Moderné vzdelávanie pre vedomostnú spoločnosť/
Projekt je spolufinancovaný zo zdrojov EÚ

POUŽÍVATEĽSKÉ ROZHRANIA

Fakulta elektrotechniky a informatiky

Daniela Perduková

Táto učebnica vznikla za finančnej podpory projektu 006-005TUKE-4/2010. Je určená študentom 3.ročníka bakalárskeho štúdia študijného programu Automatizácia mechatronických systémov pre výučbu v predmete Používateľské rozhrania.

NÁZOV: Používateľské rozhrania
AUTOR: prof. Ing. Daniela Perduková, PhD.
RECENZENTI: prof. Ing. Pavol Fedor, PhD., doc. Ing. Janka Jablonská, PhD.
VYDAVATEĽ: Technická univerzita v Košiciach
ROK: 2011
ROZSAH: 199 strán
NÁKLAD: 70 ks
VYDANIE: prvé
ISBN 978-80-553-2050-2

OBSAH

1 SYSTÉMY MAN MACHINE INTERFACE –MMI

Nasledujúca kapitola dáva odpoveď na otázky týkajúce sa definície a charakteristiky používateľského rozhrania. Činnosť návrhu takéhoto rozhrania si okrem programátorských zručností vyžaduje aj určité vedomosti týkajúce sa psychologického a ergonomického hľadiska návrhu systémov Man Machine Interface – **MMI.**

1.1 Úvod a ciele

V priebehu technického rozvoja rôznych technologických procesov vystupuje do popredia čoraz častejšie otázka prenosu informácií medzi technologickým procesom a jeho okolím. V začiatkoch technického rozvoja obsluha ovládala technologický proces jednoduchým ŠTART - STOP systémom a mohla priamo sledovať stav tohto procesu pomocou svojich zmyslov. Typickým príkladom sú najjednoduchšie typy elektrických pohonov, napr. cirkulárka. Postupne sa však technologické procesy komplikovali, stávali sa rozsiahlejšími, začali ich riadiť počítačové systémy a ich stavy už nebolo možné z rôznych dôvodov pozorovať. Preto sa medzi proces a obsluhu začali umiestňovať rôzne ovládacie panely, ktoré obsahovali množstvo ovládacích a signalizačných prvkov. Tieto sa často pri komplikovaných technológiách umiestňovali priamo do schémy technológie umiestnenej na paneli. Typickým príkladom sú rôzne velíny v elektrárňach, teplárňach a pod. Tieto panely boli vždy prispôsobené konkrétnej technológií a bolo pomerne zložité ich meniť.
S rozvojom výpočtovej techniky a postupným definovaním štandardných rozhraní medzi technologickým procesom a jeho okolím sa začali vytvárať technické a hlavne programové prostriedky, umožňujúce veľmi efektívne navrhovať, resp. meniť prenos informácií medzi technologickým procesom a okolím. Dnes už existuje množstvo vizualizačných systémov, rôznej kvality, stupňa zložitosti a od rôznych výrobcov.

CIELE

Cieľom tejto kapitoly je naučiť sa:

- čo tvorí systémy **MMI** a aká je ich základná funkcia
- čo je to ergonómia a ako ovplyvňuje návrh používateľského rozhrania
- ktoré ergonomické aspekty ovplyvňujú získavanie informácií používateľom
- akú úlohu zohráva psychologické hľadisko pri návrhu systémov MMI

1.2 Čo sú systémy Man Machine Interface - MMI?

Disciplína **Man Machine Interface - MMI** študuje komunikáciu medzi strojmi a ľuďmi. Strojom môže byť všetko od náramkových hodiniek až po veľkú továreň. Komunikáciou môže byť zase napríklad stlačenie tlačidla operátorom alebo rozsvietenie signalizačnej lampy nejakým zariadením.

Často sme trošku zmätení, ak ideme používať nové zariadenie, či je to jednoduchá vec ako telefónny aparát alebo komplikovanejšia vec ako napríklad nový počítačový program: “Ktoré tlačidlo mám teraz stlačiť?“, „Prečo to teraz pípa?“, atď.

Návrh používateľského rozhrania označovaného ako systém **MMI (Man-Machine Interface)** je základným aspektom realizácie rozhrania medzi človekom a riadiacim systémom s ohľadom na zabezpečenie:

- **kvality,**
- **spoľahlivosti**
- **a bezpečnosti pomocou počítačových systémov.**

Zámerom používateľského rozhrania je uľahčiť prenos informácií medzi používateľom a systémom, ktorý má byť riadený. Dobre navrhnuté používateľské rozhranie nielen spríjemňuje pracovné podmienky, ale tiež pomáha redukovať chyby a tak obmedzuje rozsah možného poškodenia.
Štúdium rozhrania človek/stroj je veľmi potrebné preto, aby sme vedeli navrhovať veci použiteľné čo najjednoduchším spôsobom. Tejto vlastnosti hovoríme jednoducho **použiteľnosť**.

Niekoľko príkladov riešenia problémov v oblasti **MMI**:

- ako navrhnúť TV ovládač tak, aby používateľ čo najrýchlejším spôsobom našiel požadovaný kanál
- ako navrhnúť auto, aby šofér mohol používať všetky inštrumenty bez toho, aby odvrátil zrak od vozovky
- ako navrhnúť softvér tak, aby ho nový používateľ vedel rýchlo používať bez toho, aby čítal rozsiahly manuál
- ako navrhnúť riadiacu halu vo veľkej továrni tak, aby operátor mal prehľad o celej riadenej technológií a súčasne aby mal rýchly prístup k detailným informáciám vybraných častí technológie
- ako navrhnúť medicínsky prístroj tak, aby doktor nezranil pacienta stlačením nesprávneho tlačidla
- ako navrhnúť prístroje, ktoré by mohli používať handicapovaní ľudia

Definícia:

Systém MMI tvorí súhrn programových a technických prostriedkov, ktoré slúžia na maximálne zjednodušenie obsluhy riadiaceho systému, na kontrolu a testovanie všetkých zásahov obsluhy a tiež na vzdelávanie a obnovovanie odborných a pracovných schopností personálu pri riadiacom systéme.

Z uvedenej definície vyplýva aj základná **funkcia systému MMI**:

- zabezpečiť ergonomicky prijateľný styk operátorskej obsluhy s riadiacim systémom
- zabezpečenie pomocných funkcií pre výchovu, udržiavanie výkonnosti a sledovanie činností operátorského terminálu

Úroveň a komfort obsluhy zvyšuje integrovaný systém technických a programových prostriedkov, ktorý zabezpečuje nasledovné úlohy:

- zobrazenie stavu procesu pomocou grafických prostriedkov,
- archiváciu priebehu výroby a činnosti obsluhy,
- sledovanie a spracovanie alarmových situácií,
- operátorské riadenie procesu,
- sledovanie a riadenie kvality produkcie,
- plánovanie a sledovanie údržby zariadení.

SAMOHODNOTIACE OTÁZKY

- Vysvetlite, čo znamená pojem **MMI**?
- Čo je zámerom tvorby používateľského rozhrania?
- Uveďte príklady problémov, ktoré sa riešia v systémoch **MMI**?
- Vymenujte z akých troch hľadísk sa navrhuje používateľské rozhranie v systémoch **MMI**?
- Definujte systém **MMI**.
- Aká je základná funkcia systému **MMI** ?
- Aké úlohy zabezpečuje systém **MMI**?

1.3 Štruktúra systému MMI

V systéme MMI rozoznávame 3 relatívne autonómne podsystémy:

1. **Operátorská časť**
2. **Inžinierska časť**
3. **Zabezpečenie údržby a opravy systému**

OPERÁTORSKÁ ČASŤ

Zabezpečuje vhodné rozhranie (interfejs) medzi operátorskou obsluhou a riadiacim systémom. Operátorská obsluha nie je školená v programovaní softvérových prostriedkov, ale vykonáva

- monitorovanie procesu
- zasahuje do procesu pri mimoriadnych udalostiach (odstávka, nábeh technológie, výskyt havárií).

Prístup k informáciám a spôsob vykonávania riadiacich zásahov sú dané vlastnosťou programového vybavenia.

Operátorské funkcie systému MMI sa vykonávajú nasledujúcimi programovými podsystémami:

- **podsystém pre displejové obrazce**
- **podsystém pre spracovanie alarmov**
- **podsystém pre sledovanie a riadenie kvality**
- **podsystém bilancovania a protokolovania**

Podsystém pre displejové obrazce

Displejové obrazce sa najčastejšie zostavujú do hierarchickej štruktúry:

- symbolická schéma ucelenej časti technológie (1 alebo viac obrazoviek navzájom prepojených)
- detailnejšie zobrazenie časti technológie, napr. jednotlivé regulačné obvody
- detaily jednotlivých prvkov systému, napr. informácie o procesných veličinách, údaje o regulačných slučkách, ...

Pre zobrazenie technických zariadení a informácií sa na displejových obrazcoch používajú **štandardné** prvky.

Pre zobrazovanie technických zariadení sa spravidla používajú **normované symboly**, napr. symbol ventilu, motora, čerpadla, snímača. Usporiadanie týchto symbolov odráža reálne zobrazenie technologického celku.

Okrem normovaných symbolov sa pri tvorbe displejových obrazcov využívajú aj **základné geometrické elementy** (obdĺžnik, kružnica, elipsa, ...) a **farby,** ktorých zmena zobrazuje informáciu o stave technologického zariadenia alebo jeho časti.

Podsystém pre spracovanie alarmov

Je rozhodujúci pre riešenie mimoriadnych situácií v technologickom procese alebo riadiacom systéme.

Rýchlosť a **správnosť** rozhodovania operátora pri vzniku alarmu ovplyvňuje **spôsob** a **obsah** poskytovanej informácie o ňom.

Alarm označuje vznik nebezpečnej situácie v riadiacom systéme a je spojený s prekročením povolených medzí veličín technologického procesu.

Alarmy delíme na 2 skupiny:

- **systémový alarm** – vzťahuje sa na činnosť prostriedkov a zariadení riadiaceho systému. Poskytované informácie slúžia na lokalizáciu miesta poruchy
- **procesný alarm** – súvisí s prevádzkou technologického procesu. Poskytovaná informácia obsahuje údaje o čase vzniku alarmu, o type alarmu, o veľkosti prekročeného obmedzenia

Pre sledovanie a spracovanie alarmov sú v programovom vybavení systému MMI zabudované špeciálne funkcie:

- **indikácia alarmu** – vizuálne alebo akusticky
- **potvrdenie alarmu** – akceptovanie operátorom

- **alarmové správy** – jednoznačný zmysel
- **archivovanie alarmov**

Podsystém pre sledovanie a riadenie kvality

Sledovanie a udržiavanie kvality sa dostáva do kompetencie operátorov len v poslednom čase. Operátorovi sa poskytujú informácie o stave vstupných surovín, o kvalite medziproduktu a výsledného produktu. Operátor vykonáva riadiace zásahy na základe **štatistických odchýlok**.

Podsystém bilancovania a protokolovania

Tvorí štandardnú časť programového vybavenia systému MMI. Realizuje nasledujúce funkcie:

- **zber a prvotné spracovanie technologických veličín** (selekcia neplatných vzoriek, maximálne a minimálne hodinové hodnoty, ich čas výskytu,...)
- **bilančné spracovanie veličín** (priemerná hodnota, okamžitá hodnota, suma vzoriek,...)
- **generovanie bilančných protokolov** – štruktúra protokolu sa vytvára podľa špecifikácie používateľa

INŽINIERSKA ČASŤ

Je určená pre inžiniersky personál, ktorý vyvíja a udržiava aplikačné programové vybavenie riadiaceho systému.

Inžinierske funkcie MMI

Systém MMI obsahuje osobitné programové vybavenie pre riešenie inžinierskych úloh ako sú:

- **implementácia riadiacich algoritmov**
- **konfigurovanie displejových obrazcov**
- **konfigurovanie alarmov**
- **konfigurovanie protokolov**
- **definovanie prístupových práv operátorov**

ZABEZPEČENIE ÚDRŽBY A OPRAVY SYSTÉMU

Táto časť je určená pre pracovníkov technického personálu, ktorý vykonáva preventívnu údržbu a odstraňuje poruchy na technických zariadeniach.

Ciele:

- **predchádzať vzniku chybových stavov v systéme**
- **zabezpečovať do maximálnej miery tolerantnosť systému proti chybám**

- **redukovať stredný čas na opravu alebo výmenu na minimálnu možnú hodnotu**

Z pohľadu údržby rozoznávame: **chyby** a **výpadky**

Chyba – vzniká ako výsledok nesprávnej operácie zapríčinenej nejakou poruchou. Môže sa niekedy opraviť opakovaním operácie.

Výpadok – je to nespôsobilosť funkčnej jednotky vykonávať určitú úlohu v dôsledku stálej poruchy alebo častého opakovania porúch.

Základom tohto podsystému sú **autodiagnostické programy**, ktoré zabezpečia, že indikované chyby sú automaticky spracovávané do príslušných správ a na ich základe sa robí analýza a prijatie rozhodnutia pre údržbu.

SAMOHODNOTIACE OTÁZKY

- Vysvetlite, čo znamená pojem **MMI**?
- Čo je zámerom tvorby používateľského rozhrania?
- Uveďte príklady problémov, ktoré sa riešia v systémoch **MMI**?
- Vymenujte z akých troch hľadísk sa navrhuje používateľské rozhranie v systémoch **MMI**?
- Definujte systém **MMI.**
- Aká je základná funkcia systému **MMI** ?
- Aké úlohy zabezpečuje systém **MMI**?
- Aká je štruktúra systému **MMI**?
- Charakterizujte operátorskú časť systému **MMI** a vymenujte podsystémy, z ktorých sa skladá.
- Uveďte vrstvy hierarchickej štruktúry displejových obrazcov.
- Čo slúži na zobrazovanie technických zariadení a informácií v displejových obrazcoch?
- Čo je to alarm a aké druhy alarmov poznáte?
- Aké funkcie sú zabudované v systéme MMI pre sledovania a spracovanie alarmov?
- Ktoré funkcie realizuje podsystém bilancovania a protokolovania?
- Vymenujte inžinierske funkcie systému MMI.
- Aké sú ciele podsystému pre zabezpečenie údržby a opravy systému **MMI**?
- Charakterizujte chybu a výpadok. Uveďte príklady.

1.4 Ergonomické aspekty MMI

Ergonómia je veda, ktorá sa zaoberá štúdiom čo najlepšieho **využívania** ľudských schopností v pracovnom prostredí a tiež **konfigurovaním** pracovného prostredia tak, aby sa čo najlepšie prispôsobilo potrebám človeka.

Ergonómia je teda interdisciplinárna vedná disciplína, ktorej predmetom skúmania je **ergonomický systém** (**ES**). Zahŕňa v sebe rôzne vedné odvetvia, akými sú fyzika, fyziológia, technika a psychológia.

Ergonomický systém ako zovšeobecnenie prevzatého pojmu **HUMAN – MACHINE – ENVIRONMENT SYSTEM** (HMES) pozostáva:

- z ľudského faktora (**H** (**HUMAN**) – podsystém)
- prvkov, ktoré zahŕňajú všetko, čo môžeme označiť ako stroj (**M** (**MACHINE**) – podsystém)
- prvkov prostredia (**E** (**ENVIRONMENTAL**) – podsystém)
- tzv. ergonomických interakcií (**I** (**INTERACTION**)– podsystém) ako vzťahov, ktoré spájajú uvedené podsystémy
- a času **T**

Zavedením týchto pojmov môžeme ergonomický systém popísať nasledovne:

$$\mathbf{ES = \{H, M, E, I, T\}}$$

kde $H = \{h_1, h_2, ..., h_i\}$

$M = \{m_1, m_2, ..., m_j\}$

$E = \{e_1, e_2, ..., e_n\}$

$I = \{i_1, i_2, ..., i_m\}$

T – čas

Každá interakcia **(i, j)** odráža existenciu alebo neexistenciu vzťahov medzi dôležitými ľudskými (**H**) charakteristikami (fyziologické, biochemické alebo psychologické), ergonomickými charakteristikami stroja (**M**) a prvkami reprezentujúcimi fyzikálne a sociálne podmienky prostredia (**E**). Pričom každý z uvažovaných prvkov môže byť jednoduchý alebo zložitý. Napríklad prvok $\mathbf{h_1}$ môže predstavovať schopnosť človeka reagovať v čase, zatiaľ čo $\mathbf{h_3}$ odráža jednotlivé typy osobností, $\mathbf{m_2}$ sa môže vťahovať k určitému typu riadiaceho prvku a $\mathbf{m_5}$ môže predstavovať jeho fyzikálne vlastnosti.

Vzhľadom na uvedený popis **ES** môžeme povedať, že **cieľom ergonómie** ako vedy optimalizovať identifikáciu, výber a štruktúru všetkých možných prvkov **H**, **M** a **E** podsystémov s ohľadom na fyzické a psychické vlastnosti a vzťahy medzi ľuďmi, strojmi a prostredím tak, aby bola zabezpečená **bezpečnosť**, **spoľahlivosť** a **požadovaná kvalita**.

Niektoré aspekty **ergonómie** sú veľmi dôležité v aplikáciách riadiacich systémov. Človek ako súčasť riadiaceho systému sa stretáva s problémami MMI ako **používateľ** alebo ako **návrhár.**

Používateľ by mal vedieť ako pristupovať k systému, čo hľadať, čo očakávať a ako čo najrýchlejšie spoznať všeobecné princípy zadávania príkazov. Ak je systém postavený na sledovaní dôsledných logických pravidiel, používateľ bude schopný narábať s ním vo veľmi krátkom čase.

Návrhár riadiaceho systému má za úlohu definovať ako sa budú dáta zobrazovať na obrazovke alebo riadiacich paneloch a tiež navrhnúť vzhľad a logické usporiadanie príkazov určených pre používateľa. Musí zabezpečiť, aby výstupné dáta boli ľahko zrozumiteľné aj v zložitých situáciách, aby sa používateľ vedel ľahko orientovať a robiť opravné operačné rozhodnutia a zásahy.

Ak sú v systéme porušené základné princípy ergonomického návrhu, je viac ako isté, že systém ako celok musí byť zmätok. Aj na prvý pohľad veľmi pekný interfejs, môže zakrývať

zlú funkčnosť systému. Na druhej strane, ak je používateľské rozhranie dobre štruktúrované, zrozumiteľné a ľahko sa používa, potom bude určite celý systém na vysokej profesionálnej úrovni. Neexistujú preddefinované pravidlá pre návrh MMI, ale znalosti niektorých princípov by mali byť dodržané vždy, ak sa chceme vyhnúť veľkým chybám. Analýzu návrhu používateľského rozhrania by sme mali urobiť ešte predtým ako ho budeme navrhovať pomocou obrazovkového editora alebo slovníka príkazov. Improvizácia sa v tomto prípade väčšinou nevypláca a určite si vyžaduje veľmi skúseného návrhára.

V súčasnosti až 50 - 70% softvéru na riadenie predstavuje softvér pre návrh **MMI**. Je to investícia, ktorá sa oplatí a niekoľkonásobne vráti.

SAMOHODNOTIACE OTÁZKY

- Čo rozumiete pod pojmom **ergonómia**?
- Čo je predmetom skúmania **ergonómie**?
- Z čoho pozostáva **ergonomický systém**?
- Čo je cieľom **ergonómie**?
- V ktorých dvoch polohách môže vystupovať človek v systémoch MMI?
- Charakterizujte postavenie používateľa v systémoch **MMI**?
- Aká je úloha návrhára v systémoch **MMI**?

1.4.1 Fuzzy prístup v modelovaní systémov MMI

Ergonómia ako veda je na počiatočnom stupni vývoja, predovšetkým čo sa týka metodologických prístupov v jej výskume týkajúcich sa zostavenia robustného matematického modelu pre analýzu zložitých HMES systémov. To je spôsobené predovšetkým nasledujúcimi faktormi:

1. „človek“ – je príliš zložitý systém na to, aby sme mohli úplne popísať všetky jeho vlastnosti a schopnosti v pracovnom procese
2. dlho neboli dostupné vhodné matematické prostriedky a metódy, ktoré by umožňovali získať potrebné informácie o správaní človeka a adekvátne ich popísať.

Pri vyšetrovaní takýchto systémov je potrebné brať do úvahy 3 odlišné typy neurčitosti, a to: *nepresnosť*, *náhodnosť* a *vágnosť*. Nepresnosti vznikajú meraniami, zatiaľ čo náhodnosť (udalostí) je závislá od pozorovaní a často stanovuje konečnú vlastnosť reálneho systému. Neurčitosť typu vágnosť súvisí so zložitosťou systémov vzhľadom na ľudské myslenie a procesy vnímania. Hoci obvyklé matematické techniky boli a budú aplikované pre analýzu HMES systémov, je jasné, že veľká zložitosť a neurčitosť týchto systémov si vyžaduje vo svojej podstate iné prístupy. Klasické matematické postupy a logika, ako výskumné jazyky, sú široko aplikovateľné v prírodných vedách a inžinierstve, ale nie sú postačujúce pre modelovanie systémov, v ktorých vystupuje ľudský faktor predstavujúci určité rozmery neurčitosti.

Riešením pre modelovanie systémov HMES môže byť použitie fuzzy teórie množín, ktorá umožňuje manipulovať s vágnymi (nepresnými) informáciami a tak rozoznávať

a vyhodnocovať neurčitosti javov vďaka špeciálnej teórii množín, ktorá úspešne vypĺňa medzeru medzi svetom „presnosti“ a svetom „neurčitosti“.

Fuzzy prístup k riešeniu problémov z oblasti systémov HMES, v ktorých sa vyskytuje človek ako podsystém, je rozdielny od spôsobu klasického usudzovania, ktoré vychádza z aristotelovej logiky, kde tvrdenie môže byť iba „pravdivé“ alebo „nepravdivé“. Reálna skutočnosť je však nekonečne zložitá a my často nevieme presne stanoviť hranicu nejakej množiny, t.j. určiť jednoznačne, či do nej daný prvok patrí alebo nie. Profesor Zadeh sformuloval tzv. princíp „inkompatibility“, ktorý hovorí, že keď rastie zložitosť systému, klesá naša schopnosť formulovať o tomto systéme presné a významné tvrdenia. Existuje dokonca určitá hranica, za ktorou sa presnosť a významnosť vylučujú.

V ergonomických systémoch sa často používajú vágne pojmy a operácie nad nimi, tzv. ľudský jazyk. Napríklad operátor linky popisuje ako sa má chovať regulátor pri zmene danej technologickej veličiny nasledovne: Ak je rýchlosť pásu veľká potom nastav nižšie napájacie napätie. Takáto informácia je vyjadrená nepresným spôsobom, no v teórii fuzzy množín môžeme chovanie takéhoto systému popísať vo forme fuzzy výrokov, v ktorých sú obe časti výroku zviazané logickou funkciou implikácie „ak ... potom“. Pojem „veľká rýchlosť“ označujeme ako fuzzy pojem (množina), pričom rozhodnutie o tom, či daná rýchlosť patrí alebo nepatrí do tejto množiny, nahradíme určitou mierou, váhou. Táto miera príslušnosti sa volí v rozsahu <0, 1>, pričom príslušnosť „0“ znamená, že prvok vôbec nepatrí do množiny (nepravda) a príslušnosť „1“, že do nej patrí úplne (pravda). Hodnota miery príslušnosti medzi týmito hranicami vyjadruje príslušnosť objektu do danej množiny.

Príklad:

Fuzzy pojem *veľká rýchlosť*	Veľkosť	Miera príslušnosti
	150 m/s	1
	100 m/s	0,9
	70 m/s	0,7
	50 m/s	0,5
	5 m/s	0,2
	1 m/s	0

Fuzzy systémy sú teda také systémy, ktorých popis je robený na rozdiel od klasických analytických systémov pomocou fuzzy pojmov a operácií nad nimi. Vzhľadom na neurčitosť vzťahu medzi človekom a jeho pracovným prostredím, zložitosť pravidiel a daných princípov v systémoch HMES sa práve tento prístup stále častejšie využíva pre ich skúmanie, popis a modelovanie.

1.4.2 Aplikácie teórie fuzzy množín vo výskume systémov MMI

Teória fuzzy množín bola úspešne aplikovaná pri modelovaní nedostatočne definovaných systémov v rôznych disciplínach ako je psychológia, oblasť spracovania informácií a riadenia, v biologických a medicínskych vedách, v sociológií a lingvistike, v oblasti spracovania a rozpoznávania obrazov a umelej inteligencie. V súčasnosti existuje veľa aplikácií využívajúcich fuzzy prístup aj v oblasti systémov HMES.

Terano a kol. predstavil vo svojej práci fuzzy prístup pri štúdiu a analýze problému spoľahlivosti človeka z hľadiska bezpečnosti systémov HMES. Kramer a Rohr vyvinuli fuzzy model správania sa šoféra v situáciách, ktoré boli simulované prostredníctvom vizuálnych informácií. Hirsch použil fuzzifikáciu pre popis hlasového rozsahu človeka.

Willaeys a Malvach ako prví skúmali v priemyselných úlohách zostavenie fuzzy modelu riadenia technologického procesu na základe subjektívnych informácií a skúseností získaných

od operátora danej technológie vo forme lingvistických premenných, napr. ak je tlak príliš vysoký, potom zníž teplotu a pod.

Benson vyvinul interaktívny počítačový grafický program pre analytické úlohy, ktoré nie je možné s dostatočnou presnosťou definovať alebo používajú nepresné údaje. Použitie fuzzy prístupu umožňovalo identifikovať príslušnosť farby vstupnej informácie do určitej kategórie danej farebnej palety.

Karwowski a kol. vytvorili fuzzy model pre určenie prípustnej záťaže pri úlohách, ktoré si v praxi vyžadujú ručné dvíhanie zariadenia určitej váhy operátorom. Hodnoty prípustností záťaže boli vyjadrené funkciami, ktoré odrážali stupne biomechanických a fyziologických faktorov prípustných pre operátora.

Lucrak a GE aplikovali fuzzy modelovanie vo výskume vzťahov medzi fyzickou váhou operátora a závažím, ktoré môže dvíhať. Onisawa vyvinul niekoľko fuzzy prístupov pre zlepšenie štúdia spoľahlivosti ľudského faktora v systémoch HMES.

Vymenovali sme len niektoré najzaujímavejšie práce z oblasti modelovania systémov HMES, kde sa využíva fuzzy prístup. Výsledky dosiahnuté v tejto oblasti sú dôkazom toho, že teória fuzzy množín, zaoberajúca sa matematickou reprezentáciou a manipuláciou s nepresnými pojmami, je výkonným nástrojom pre analýzu systémov, v ktorých vystupuje ľudský faktor. Takéto systémy sú zložité, ich štruktúra a vzájomné vzťahy nie sú presne známe, ich popis má vo všeobecnosti lingvistickú povahu a definícia premenných a pojmov sa vyznačuje značnou mierou neurčitosti. Keďže fuzzifikácia hrá podstatnú úlohu v oblasti ľudského poznávania a činnosti, výskum v tejto oblasti stále vo väčšej miere využíva obrovský potenciál, ktorý fuzzy metódy ponúkajú, predovšetkým v oblasti analýzy ergonomických systémov.

SAMOHODNOTIACE OTÁZKY

- Ktoré faktory ovplyvňujú zostavovanie robustných matematických modelov zložitých HMES systémov?
- Čo je potrebné brať do úvahy pri vyšetrovaní zložitých HMES systémov?
- Zdôvodnite, prečo sú fuzzy systémy vhodné pre použitie v oblasti modelovania HMES systémov.
- Uveďte niektoré známe aplikácie fuzzy množín vo výskume HMES systémov.

1.5 Psychologické hľadisko

Aby sme mohli definovať efektívne spôsoby návrhu **MMI**, je potrebné najprv uviesť model ľudského uvažovania a spracovávania informácií.
Odhaduje sa, že celkové množstvo informácií, ktoré vstupujú do nášho tela je okolo 1 bilióna bitov za sekundu, z ktorých len okolo 100 bitov za sekundu sme schopní spracovať. Mozog sa vždy snaží redukovať množstvo spracovávaných informácií. Ak je príliš veľa informácií prezentovaných v rovnakom čase, potom sa naša pozornosť sústredí len na ich určitú časť. Mozog je schopný filtrovať obrazy, ale aj zvuky a hluk na pozadí.

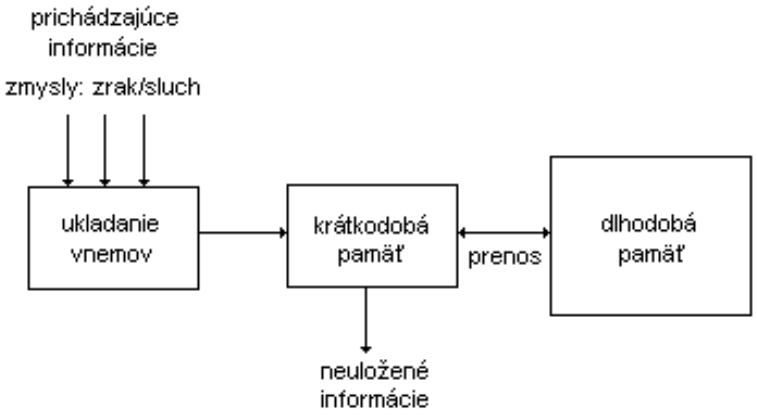
Psychológovia sa dlho zaoberali týmto problémom spracovania informácií a navrhli niekoľko modelov. V kognitívnom procese budeme rozlišovať funkčné bloky:

- **ukladania vnemov**,
- **krátkodobej pamäti**,
- **dlhodobej pamäti**,
- **plánovania a**
- **prevedenia odpovedajúceho zásahu**.

Človek sa môže koncentrovať len na **jednu** vec v danom čase, ale môže veľmi ľahko upriamiť pozornosť z jednej veci na druhú. Z hľadiska **MMI** sú najdôležitejšie vizuálne a akustické vnemy.

Informácie zbierané zmyslovými orgánmi sú prenášané do krátkodobej pamäte, kde môžeme vedome dávať na nich pozor. Z krátkodobej pamäte sa informácie, v mnohým prípadoch len vlastným úsilím, prenášajú do dlhodobej pamäte. Krátkodobá pamäť je veľmi rýchla na pripomenutie, ale aj na zabúdanie. Je to naše vedomie. Udržuje potrebné informácie v čase premýšľania a poskytuje nám základňu pre prevedenie potrebných akcií. Pripamätanie a ukladanie do dlhodobej pamäti trvá dlhšie. Informácie v krátkodobej pamäti sa môžu vybaviť za niekoľko sekúnd, ale v dlhodobej pamäti to môže trvať aj celý život.

V **krátkodobej pamäti** je miesto pre **7+2 informačných položiek**. Nové prichádzajúce informácie vymažú už existujúce, ktoré ešte neboli spracované alebo presunuté do dlhodobej pamäti. Podstatné je, že položky v krátkodobej pamäti sú na rovnakej abstrakčnej úrovni (ak rozmýšľame o futbalovom skóre, nie je problém porovnať ho v rovnakom čase s inými futbalovými skóre, ale nemôžeme rozmýšľať o futbale a súčasne plánovať dovolenku).



Obr.1 Model pamäte

Ukladanie nových informácií do dlhodobej pamäte je ľahšie, ak sa tieto spájajú s informáciami, ktoré už existujú a sú uložené. Takisto si ľahšie zapamätáme určité fakty, ak nie sú prezentované samostatne, ale v určitých príčinných vzťahoch.

Ľudská pamäť nepracuje s priamo adresovateľnými bunkami ako počítač, ale pracuje na báze **analógií a asociácií.**
V používateľských systémoch je väzba na predchádzajúcu skúsenosť založená na používaní symbolov každodenného života. Na obrazovke počítača bude preto pero symbolizovať niečo na písanie a guma nejaký prostriedok na mazanie. Symboly sú len metaforami skutočného života a dávajú do súvisu operácie (činností) známych objektov z bežného života s podobnými operáciami v počítači. Tento súvis budeme označovať pojmom **VIDITEĽNOSŤ.**

Viditeľnosť je veľmi účinná, ak sa vzťahuje ku každodenným skúsenostiam a objektom (pri šoférovaní, ak chceme ísť doľava, potom točíme volantom doľava; ak by sme navrhli auto, kde by to bolo naopak, potom by sme sa učili šoférovať oveľa ťažšie). Viditeľnosť má priame praktické využitie v systémoch MMI. Napríklad hodnota 66 samotná nič neznamená. Ak ju však nazveme teplotou a porovnáme ju s maximálne dovolenou teplotou napr. 70°C, hneď máme viac užitočných informácií. Pohľad na obr. 2 nám dá hneď odpoveď na otázku, či je teplota o 4°C nižšia ako maximálna teplota ešte prípustná.

Obr.2 Príklad intuitívneho zobrazenia výstupu

Ďalším dôležitým pojmom, ktorý budeme v systémoch MMI používať, je **DÔSLEDNOSŤ**. Tá pomáha prenášať existujúce vedomosti a poznatky do nových pojmov či obsahu. Dôslednosť prakticky znamená, že príkaz bude mať vždy rovnaký alebo ekvivalentný význam, nezávisle od špecifickej situácie. To znamená, že príkaz na nahrávanie súborov by nikdy nemal vymazať súbor v rámci toho istého programu.

Viditeľnosť a dôslednosť je definovaná štandardnými symbolmi v rámci noriem ISO a IEC. Tieto symboly sú jednoduchými grafickými reprezentáciami spoločných objektov a pomáhajú robiť operácie so zariadeniami tak intuitívne, ako je to len možné.

Dôslednosť je dosiahnutá vtedy, ak sa symboly používajú na všetkých zariadeniach, pričom žiadne zo zariadení nepoužíva iné symboly pre tú istú funkciu, pokiaľ to nevyžadujú špeciálne dôvody.

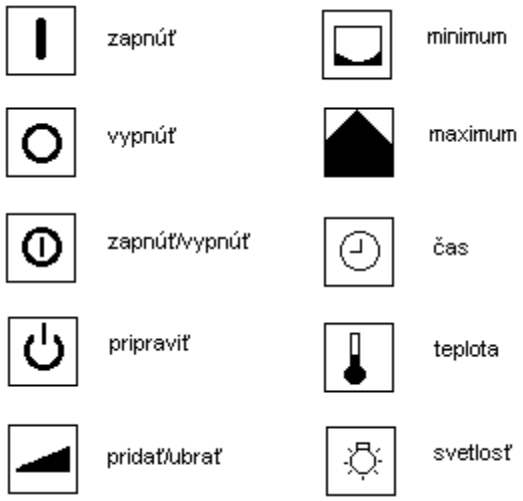
Ďalším dôležitým faktorom používateľsky orientovaného návrhu je **SPÄTNÁ VÄZBA**. Po zadaní príkazu by mala nasledovať reakcia, ktorá nie je potvrdením akcie (točíme volantom→doprava ideme doprava, stlačenie klávesy→kliknutie, ...), ale dáva užívateľovi aj odpoveď, či sa daná akcia uskutočnila alebo nie.

Získanie informácií používateľom ovplyvňujú 3 ergonomické aspekty:

- **schopnosť vnímania**,
- **kódovanie** a
- **organizovanie a štruktúrovanie**.

Schopnosť vnímania má fyzikálnu podstatu a závisí od hardvéru príslušných zariadení. Napríklad pre počítačovú obrazovku sú dôležitými faktormi: svetlosť, farebný kontrast, veľkosť symbolov,... .

Kódovanie je cesta prenosu informácií za pomoci symbolov a stôp. Kódovaná správa sa ľahko vizuálne prezentuje a pomáha prenášať pomerne veľké množstvo informácií. Napríklad rovnaký symbol zobrazený v rôznej farbe môže vyjadrovať odlišné informácie o danom objekte.



Obr.3 Príklady štandardných symbolov (IEC 417/ISO 3461/ISO 7000-DAD/1)

Organizovanie a štruktúrovanie používateľského rozhrania napomáha používateľovi ľahko sa orientovať pri práci so systémom a tým umožňuje, aby jeho pozornosť bola sústredená len na vzniknutý problém a jeho riešenie, a nie na výstavbu štruktúry jeho rozhrania

Cieľom návrhu dobrého interfejsu **MMI** je pritiahnuť pozornosť na dôležité fakty a informácie a na ich základe umožniť rýchle a správne reakcie.

Z toho dôvodu, ak operátor musí rýchlo reagovať na novú informáciu, napríklad zadať nový príkaz, prezentovaná informácia by mala byť:

- **logicky organizovaná**
- **nemala by prekročiť 5 odlišných položiek na tej istej abstrakčnej úrovni.**

Dobré používateľské rozhranie by malo byť prehľadné a málo náročné, malo by používateľovi umožňovať sústrediť sa na proces, ktorý riadi bez toho, aby bol rozptýlený interfejsom a samotným riadiacim systémom.

SAMOHODNOTIACE OTÁZKY

- Aké modely boli vypracované z psychologického hľadiska pre efektívne spôsoby návrhu systémov **MMI**?
- Ktoré ľudské vnemy sú najdôležitejšie z hľadiska systémov **MMI**?
- Popíšte spôsob spracovania informácií v ľudskej pamäti.
- Čo rozumieme pod pojmom viditeľnosť a aké má praktické využitie v systémoch **MMI**?
- Vysvetlite aký význam má pojem „dôslednosť“ pri návrhu systémov **MMI.**
- Akú úlohu zohráva pojem „spätná väzba“ v systémoch **MMI**?
- Ktoré tri ergonomické aspekty ovplyvňujú získavanie informácií používateľom v systémoch **MMI**? Uveďte ich krátku charakteristiku.

2 VŠEOBECNÉ PRINCÍPY NÁVRHU POUŽÍVATEĽSKÉHO ROZHRANIA

Táto kapitola sa zaoberá špecifikáciou všeobecných princípov používateľského rozhrania a faktorov ovplyvňujúcich jeho tvorbu. Návrhu systémov **MMI** sa nezaobíde bez poznania základných faktorov a princípov návrhu jednotlivých obrazoviek, syntaxe správ a príkazov, princípov transparentnosti a predvídateľnosti a ďalších dôležitých faktorov, ktoré zvyšujú jeho mieru **použiteľnosti.**

2.1 Úvod a ciele

Typický počítačový používateľ má snahu narábať so všetkým, čo sa objaví na obrazovke počítača. Obrazovka by mala byť preto jednoduchá a nemala by obsahovať nepotrebné informácie. Tie môžu priťahovať pozornosť používateľa, ktorú by mal venovať dôležitejším informáciám. Rovnako informačný obsah obrazovky má byť zameraný na používateľa, pretože expert v danej oblasti sa nezaujíma o triviálne informácie, ktoré však môžu byť rozhodujúce pre nováčika.

Kľúčom k funkčnosti a prezentácií veľkého množstva údajov je ich správny výber a štruktúrovanie, z čoho vyplývajú aj základné princípy návrhu používateľského rozhrania.

CIELE

Cieľom tejto kapitoly je naučiť sa:

- ktoré faktory ovplyvňujú tvorbu používateľského rozhrania
- aké sú všeobecné princípy návrhu používateľského rozhrania
- aké sú základné charakteristiky správne navrhnutého používateľského rozhrania
- akú syntax majú mať informácie zobrazované ako správy na obrazovke počítača
- aké sú najdôležitejšie princípy návrhu obrazoviek používateľských rozhraní
- ktoré pravidlá je potrebné dodržiavať pri zadávaní príkazov a tvorbe ponúk v systémoch MMI
- akú úlohu hrá spätná väzba pri návrhu používateľského rozhrania
- ktoré princípy je potrebné dodržať, aby bolo používateľské rozhranie transparentné
- čo je potrebné dodržiavať pri návrhu ovládacích tlačidiel
- čo sú to štandardy a ako súvisia s návrhom systémov **MMI**

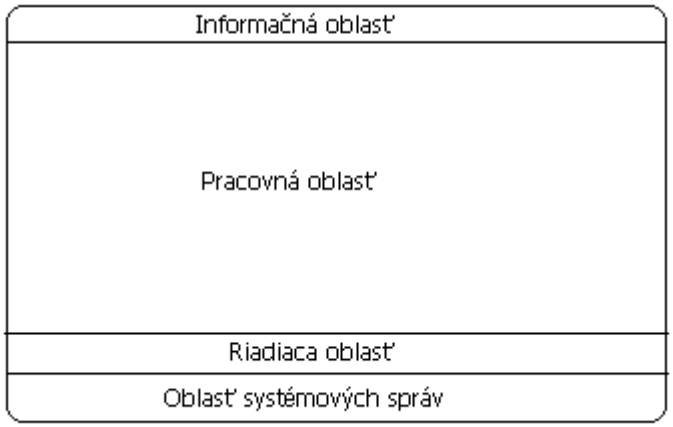
2.2 Návrh obrazovky

Niektoré princípy návrhu obrazovky boli formulované v spojení s inými účelmi, hlavne masovo-komunikačnými technikami pre reklamu, ktoré poskytli mnoho podnetov pre prezentáciu informácií na obrazovke počítača.

Najdôležitejšie princípy návrhu obrazoviek sú formulované nasledovne:

1. Pretože prirodzený pohyb očí je **zľava → doprava**, vývoj riadeného procesu by mal byť prezentovaný tiež týmto smerom. Pohyb po obrazovke je ľahšie pochopiteľný po vodorovnej ako po zvislej osi.
2. Návrh obrazovky musí byť zhodný na všetkých úrovniach. Môže byť rozdelený do **4 sekcií**:
   - **Pracovnej** (technológia, správy príkazy)
   - **riadiacej** (virtuálna klávesnica)
   - **oblasti systémových správ** (systémové alarmy)
   - **a informačnej oblasti** (v ľavom rohu je umiestnený dátum/čas a v pravom procesné alarmy)

   Ak uvažujeme štandardnú obrazovku s 24 riadkami, potom pre pracovnú oblasť je rezervovaných 16-20 riadkova 1-3 riadky pre každú ďalšiu sekciu. Jednotlivé sekcie by mali byť oddelené čiarami, zvýraznením alebo iným spôsobom. Veľkosť každej sekcie by mala byť rovnaká na každej obrazovke.



Obr.4 Funkčné rozdelenie obrazovky

3. Pre **štruktúrovanie** sa používajú metódy symetrie, podobnosti a zoskupovania, čo zjednodušuje používateľovi rýchle rozpoznanie navrhnutej koncepcie, pretože informácie zobrazované bez zoskupovania na základe určitých vzájomných vzťahov sú veľmi neprehľadné a chaotické.
4. Účinným spôsobom kódovania informácií je **používanie farieb**. Tie sa však musia používať veľmi opatrne a striedmo. Bez väčšieho úsilia sme schopní rozoznať **4-5 farieb**. Maximum, ktoré sme schopní vnímať je **7 farieb**.
5. Farby musia byť používané veľmi dôsledne, tzn. **jedna farba-jeden význam**. Variácie pri používaní farieb môžu viesť k zlým analógiám a nesprávnym predpokladom.
6. Farby sa najčastejšie používajú na zobrazovanie funkčných stavov zariadení.

Zelená - sa chápe ako označenie bezpečnosti, povolenia, korektnosti

Červená - sa vzťahuje na farby alarmov, nebezpečenstva, zákazu

Žltá - sa chápe ako upozornenie a tiež môže indikovať prítomnosť nejakého menšieho problému

7. Kombinácia farieb by mala byť pre používateľa príjemná a nie vyčerpávajúca. Všetky použité symboly by mali určitý kontrast s pozadím. Kombinácie farieb, ktoré majú silný vzájomný kontrast a sú ľahko vnímateľné:

   žltá/čierna

   červená/biela

   červená/žltá

   čierna/biela

   zelená/čierna

   Nevhodné kombinácie: **ružová/biela**

   modrá/čierna

8. Je potrebné dbať na to, aby sa na dôležité informácie upozorňovalo nielen farbami. Veľký počet ľudí nie je schopní vnímať a rozpoznávať určité farby. Naviac environmentálne faktory, ako tienenie alebo žiarenie, môžu sťažiť vnímanie niektorých farieb v prevádzkach. Preto zobrazovanie dôležitých informácií by malo obsahovať určité množstvo redundancie, napríklad návestia, texty, grafické symboly, aby sme znížili pravdepodobnosť nesprávnej interpretácie.

9. Najprirodzenejšou reprezentáciou objektu je jeho zobrazenie v podobe obrázku. Zobrazením objektu vždy povieme viac ako informáciami v textovej podobe.

10. Pri zobrazovaní textu by sa nemala používať metóda blikania, pretože blikanie sťažuje čítanie textu.

11. Navrhnuté zobrazenie by nemalo byť nudné, ale zaujímavé a motivujúce pre používateľa.

SAMOHODNOTIACE OTÁZKY

- Ktorým smerom má byť na obrazovke počítača prezentovaná postupnosť informácií, vo vodorovnej alebo zvislej osi?
- Do koľkých častí môžeme rozdeliť informácie zobrazené na obrazovke počítača? Vymenujte ich.
- Uveďte, kde by ste na obrazovke počítača umiestnili jednotlivé sekcie pre zobrazovanie informácií a koľko miesta by ste pre nich približne vyhradili.
- Aké metódy štruktúrovania sa používajú pre zjednodušenie rozpoznania navrhnutej koncepcie rozhrania?
- Aký význam má používanie farieb pri návrhu rozhrania?
- Koľko farieb na jednej obrazovke sme schopní vnímať a koľko z toho bez problémov rozoznať?
- Ktoré farby sa používajú pre znázorňovanie jednotlivých funkčných

	stavov zariadení? Uveďte význam ich použitia. • Vymenujte niektoré vhodné a nevhodné kombinácie farieb používaných pri návrhu rozhrania. • Potrebnú informáciu je vhodnejšie zobraziť vo forme obrázku alebo v textovej forme? • Ktorá metóda by sa pri zobrazovaní textu určite nemala používať?

2.3 Správy

Vizualizácia správ na obrazovke podporuje operátora pri riešení úloh riadenia procesu. Operátor správy v reálnom čase vyhodnocuje a rozhoduje o ďalšom postupe. Každá správa sa objavuje pri špecifických udalostiach.

Správa je teda **informácia** o výskyte udalosti a má obsahovať odpovede na nasledujúce otázky:

- **kedy?** – čas vzniku udalosti
- **Čo?**- typ a stav udalosti
- **Kde?** – lokalizácia udalosti
- **Aká dôležitosť?** – nutnosť reakcie operátora, priorita správy

TYPY SPRÁV

Správy môžeme klasifikovať z nasledujúcich hľadísk:

- **pôvod, miesto vzniku**
  - **procesné správy** – z technologické procesu, slúžia operátorovi na riadenie procesu
  - **správy z riadiaceho systému** – hovoria o jeho poruchách, sú určené pre údržbu
- **efekt, pôsobenie signalizovanej udalosti**
  - **prevádzkové správy** – objavujú sa v priebehu normálnej prevádzky (motor4 zapnutý)
  - **poruchové správy** – hovoria o chybách, zhoršení funkcií zariadení
  - **správy o nebezpečenstve** – hovoria o výskyte stavu, ktorý ničí zdravie, život, okolie alebo zariadenie
- **zoskupovanie, spájanie**
  - **individuálne**
  - **skupinové**
- **priorita, dôležitosť**
  - **predvýstraha**
  - **výstraha**
  - **alarm**
- **nutnosť potvrdzovania**
  - **vyžadujúce potvrdenie**
  - **bez potvrdenia**
- **smer**
  - **došlé**
  - **odídené**

Operátor by mal mať na obrazovke vizualizované iba správy **dôležité** pre proces. Tieto sa týkajú nežiadúcich zmien, na ktoré operátor musí reagovať. Rýchlosť prichádzajúcich správ by nemala byť väčšia ako **15 správ za minútu.**

SYNTAX SPRÁV

Často potrebujeme na obrazovke počítača zobrazovať informácie, o stave nejakého zariadenia, na základe ktorých sa vykonávajú určité akcie. Tieto informácie zobrazujeme pomocou:

- **fixného-statického textu**, ktorý popisuje typ zariadenia a
- **dynamickej premennej**, ktorá zobrazuje hodnotu aktuálneho stavu zariadenia.

Statický text uvádzame vždy v spojení s dynamickou premennou. Sám osamote nedáva úplnú informáciu a mohol by byť aj nesprávne pochopený. Len v kombinácií s dynamickou premennou dáva úplný zmysel.

namiesto:	**zariadenie A11 napájané** :	**áno/nie**
použijeme:	**zariadenie A11:**	**zap/vyp**

Dynamická informácia by nemala negovať statickú časť, ale mala by ju dopĺňať. Statický text by nemal mať príliš veľa možností, stavov pre zobrazenie dynamickej premennej.

namiesto:

zariadenie A12 stav:	**zap/vyp**
	Ok/Alarm
	aktívne/pripravené

použijeme:

zariadenie A12:	**zap/vyp**
zariadenie A12 pracuje:	**Ok/Alarm**
zariadenie A12 pripojené:	**aktívne/pripravené**

Jasnejšie odlíšenie medzi statickým textom a dynamickou premennou dosiahneme, ak statický text bude na obrazovke zobrazený normálnou intenzitou a dynamická premenná bude zvýraznená.

Statický text by nemal obsahovať slová s negatívnym upozornením, ako **POZOR ALARM !**, pretože tieto sa jasne nevzťahujú ku konkrétnemu stavu riadeného systému. Text zobrazovanej správy by mal motivovať a nie iritovať používateľa.

SAMOHODNOTIACE OTÁZKY

- Definujte pojem správa v systémoch MMI a vymenujte, čo má obsahovať.
- Vymenujte hľadiská, podľa ktorých klasifikujeme správy.
- Aký druh správ je potrebné zobrazovať na obrazovke operátora?
- Aký maximálny počet správ by sa mal objaviť na obrazovke operátora za minútu?
- Akým spôsobom zobrazujeme na obrazovke počítača informácie o stave nejakého zariadenia?

	• Uveďte príklady správneho a nesprávneho zobrazenia správy o stave nejakého zariadenia. • Aké vlastnosti má mať dynamická premenná v syntaxe správy a ako ju môžeme odlíšiť od statického textu?

2.4 Príkazy, pohľady, trendy

PRÍKAZY

V systémoch **MMI** prebieha komunikácia od používateľa k zariadeniu alebo naopak, pričom obidve sú rovnako dôležité. Komunikácia od používateľa k zariadeniu sa uskutočňuje prostredníctvom zariadení ako je myš, joystick, tlačidlá na paneloch alebo zadaním sekvencie príkazov z klávesnice. Návrh komunikácie **človek → stroj** by sa mal riadiť nasledujúcimi pravidlami:

- Referenčná hodnota pre určitý stav zariadenia zadaná z klávesnice musí byť reprezentovaná takým spôsobom, aby nemohla byť v žiadnom prípade zamenená za aktuálnu hodnotu zobrazovaného stavu. Referenčná hodnota sa stane aktuálnou hodnotou len v tom prípade, keď zariadenie pracuje správne.

- Akcie, ktoré nasledujú za príkazom, by mali byť rovnaké na všetkých úrovniach navrhnutého používateľského rozhrania.

- Ak je pre vstupný údaj prípustných niekoľko alternatív, tieto by mali byť jasne a zreteľne označené. Vstupné príkazy by mali byť okamžite kontrolované riadiacim systémom a výsledok by mal byť oznámený používateľovi.

- Textové príkazy zadávané z klávesnice by mali byť čo najkratšie, bez toho aby stratili na význame. Vhodná metóda je používanie prvých písmen príkazového názvu za predpokladu, že tieto sa nebudú navzájom zamieňať.

- V prípade, ak sa požaduje pre vstup len niekoľko možností, môžeme sa vyhnúť zbytočným chybám, ak

  a) zobrazíme prípustné hodnoty na pozadí obrazovky
  b) poskytneme výber správnej hodnoty pomocou menu
  c) zobrazíme chybovú správu, ak vstup nie je zadaný správne

  Alternatíva a) nie je vhodná, ak je väčší počet možných vstupov, pretože obrazovka je potom pokrytá príliš veľkým počtom informácií. Alternatíva c) môže spôsobovať oneskorenie podľa toho, ako často sa vyskytujú chyby. Riešenie b) sa preto javí ako optimálne.

- Zadávanie príkazu z klávesnice môže viesť k rôznym chybám, preto je vhodné opýtať sa prostredníctvom zobrazeného hlásenia na potvrdenie vykonania príkazu. Napr. **"Chcete naozaj vymazať knižnicu ? Áno/Nie"**

- Veľmi nebezpečné príkazy by mali byť zabezpečené heslom. Mali by sme sa však vyhnúť veľmi komplikovaným ochranným schémam, pretože dobrý riadiaci systém by mal súčasne bezpečný a jednoduchý pre používateľa.

- Je dôležité, aby bolo možné počítačom riadený stroj zastaviť okamžite v prípade nebezpečenstva. Zreteľne označené tlačidlo pre tento účel by malo byť umiestnené tak, aby operátor mohol kedykoľvek zasiahnuť. Toto tlačidlo, ktoré sa zvyčajne označuje červenou farbou, by malo byť tiež dostatočne veľké.

Princípy zreteľnosti a dôslednosti vyžadované pri návrhu obrazoviek a príkazov musia byť samozrejme dodržané aj pri návrhu menu:

- **Štruktúra menu by sa mala stať pre používateľa okamžite zrozumiteľná**. Každé menu by malo byť označené hlavičkou (titulkom), ktorá je totožná s položkou uvedenou v menu o úroveň vyššie.
- **Prerušovací príkaz by mal byť dostupný vždy prostredníctvom rovnakého kľúča**. Malo by byť možné v ľubovoľnom okamžiku prerušiť prácu, vrátiť sa späť na vyššiu úroveň alebo do základného menu.
- **Položky v menu by mali byť na rovnakej významovej úrovni**, tzn. že nemôžeme umiestniť príkaz pre mazanie a pre tlač do toho istého menu.
- **Počet položiek v menu by mal byť v rozsahu od 2 do 5**.
- **Základnú ideu štruktúry menu je potrebné dodržať na každej obrazovke**.
- **Podobné funkcie v odlišných menu by mali byť prístupné cez rovnaké funkčné kľúče**.

POHĽADY

Pohľad – schematická, symbolická vizualizácia zariadení a ich častí, komponentov, vrátane ich vzájomného prepojenia za účelom vyjadrenia ich vzťahu vo vnútri technologického zariadenia. Pohľady majú byť vytvárané tak, aby bolo možné identifikovať:

- usporiadanie technologických systémov, ich podsystémov a ich prepojenie cez štruktúru, toky materiálu, ale aj informácie od procesného riadenia
- poruchy alebo odchýlky od žiadaného stavu v mieste ich výskytu
- pohotovosť technologických zariadení a ich častí

Pohľady sú umiestnené v pracovnej časti obrazovky. Podľa významu komponentov a spojníc sú v nich vizualizované:

- technologické zariadenia – **pohľad na techológiu**
- zariadenia procesného riadiaceho systému – **pohľad na riadenie**
- kombinácia obidvoch predchádzajúcich – **kombinovaný pohľad**

Pohľad na technológiu

Je to pohľad, v ktorom sú pomocou grafických objektov (symbolov) zjednodušene znázornené komponenty technologických zariadení (zásobník, motor, čerpadlo, nádrž, kompresor,...). Transport materiálu a energií je reprezentovaný spojnicami s odporučeným výberom farieb.

Typy pohľadov:

- **základný (prehľadový)** pohľad znázorňuje technologické zariadenia v zjednodušenej forme pomocou blokov prepojených animovanými spojnicami na znázornenie toku materiálu, energie, informácie. Bloky môžu predstavovať jednotlivé systémy alebo ich podsystémy
- **technologický (vlastný)** pohľad, ktorý znázorňuje technologický proces pomocou grafických symbolov reprezentujúcich jednotlivé komponenty, ktoré sú pospájané animovanými čiarami, sú tu zobrazené všetky časti techológie
- **pohľad na úsek** obsahuje technologické skupiny, ktoré sa v ňom dajú vybrať
- **pohľad na skupinu** odpovedá skupine technológie, ktorá obsahuje viac prístrojov
- **detailný pohľad** slúži na zobrazenie jednotlivých prístrojov**,** predstavuje podporu pri konfigurácií a uvádzaní do prevádzky, pomáha operátorovi pri objasňovaní príčin poruchových stavov.

Pohľad na riadenie

symbolické zobrazenie komponentov riadiacej techniky, napr. regulátorov, vstupných zariadení pre žiadané a hraničné hodnoty. Tieto sú poprepájané animovanými spojnicami reprezentujúcimi signály.

TRENDY

Účelom vizualizácie trendov je pomáhať operátorovi pri riadení procesu, hlavne pri monitorovaní priebehov technologických veličín minulých, súčasných i predpokladaných. Trendy ukazujú priebeh jednej alebo viacerých hodnôt premenných vzhľadom na čas. Sú zobrazované ako spojité čiary alebo postupnosti bodov. V jednom poli by malo byť maximálne 6 kriviek. Farby sú voliteľné. Pre tmavý podklad sú odporúčané nasledujúce farby: **oranžová, biela, svetložltá, svetlozelená, svetlofialová, svetlomodrá.**

Typy trendov

Vzhľadom na reálny čas sa trendy delia na:

- **historické, minulé** – zobrazujú priebehy archívnych, minulých časových úsekov, hlavne pre analýzu a verifikáciu
- **súčasne, reálne** – zobrazujú aktuálne hodnoty a hodnoty v krátkom časovom úseku pred reálnym časom, krátku históriu
- **predikčné, predpokladané** – zobrazujú predpokladaný vývin, vo vizualizačnom systéme musia existovať prídavné predikčné moduly

SAMOHODNOTIACE OTÁZKY

- Vymenujte niektoré pravidlá pre návrh komunikácie človek – stroj týkajúce sa zadávania príkazov.
- Akú vhodnú metódu poznáme pre zadávanie textových príkazov z klávesnice?
- Aké riešenie je najvhodnejšie pre zadanie vstupného príkazu s výberom niekoľkých možností?
- Ako je potrebné ošetriť zadávanie príkazu z klávesnice a ako

	zabrániť zadávaniu tzv. nebezpečných príkazov? • Vymenujte niektoré základné princípy návrhu štruktúry menu na obrazovke. • V akom rozsahu by mal byť počet položiek v menu? • Čo je to pohľad a čo je možné pomocou neho identifikovať? • Uveďte klasifikáciu pohľadov. • Na čo slúži pohľad na technológiu a aké typy týchto pohľadov poznáme? • Čo sú to trendy a aké je ich postavenie v systémoch MMI? • Aké typy trendov poznáte?

2.5 Spätná väzba k používateľovi

Kedykoľvek používateľ stlačí tlačidlo, otočí vypínačom, klikne myšou, napíše príkaz alebo iným spôsobom zadá príkaz pre zariadenie, s ktorým pracuje, vždy musí existovať **spätná väzba**, ktorá ho informuje o tom, že príkaz bol prijatý správne. Spätnou väzbou môže byť zvukový alebo svetelný signál, textová správa na obrazovke a pod.

Ideálnou formou spätnej väzby je umožniť používateľovi presvedčiť sa, že zadaný príkaz bol vykonaný. Ak naštartujeme motor, počujeme jeho zvuk a vidíme, že sa kolesá otáčajú. Šofér v tom prípade nemá pochybnosť o tom, či motor beží alebo nie.

Mechanické tlačidlá a vypínače môžu jednoducho poskytovať spätnú väzbu svojou **polohou.**

Posuvný gombík pre zoslabovanie alebo zosilňovanie signálu na mixážnom pulte nám poskytuje perfektnú informáciu o zadanom príkaze svojou polohou, ktorá vyplýva z prirodzenej podstaty jeho mechanického návrhu. Otáčajúci gombík je lacnejší, ale jeho polohu môžeme zistiť len pomocou malej značky alebo šípky, ktoré sú na väčšine takýchto gombíkov umiestnené. **Hore/dole** tlačidlá umiestnené napríklad na diaľkových ovládačoch televízorov nemajú spätnú väzbu, s výnimkou ovládania zvuku .

Príklady uvedených zariadení sú na obr.5.

Obr.5 Mechanické tlačidlá – spätná väzba polohou

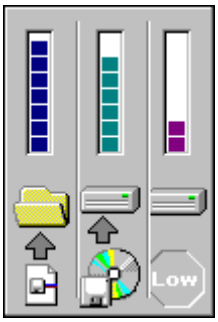
Ak zariadenie **nemá spätnú väzbu**, potom používateľ môže predpokladať, že zadaný príkaz nebol prijatý a akceptovaný. Preto môže stlačiť tlačidlo ešte raz alebo si myslieť, že zariadenie je pokazené alebo on urobil niečo nesprávne.

Doba odozvy je pri spätnej väzbe veľmi dôležitá. V každom prípade by mala byť dostatočne krátka. Predstavme si napríklad počítačový program vykonávajúci rôzne matematické výpočty. Väčšina z nich trvá pár sekúnd, ale jedna vybraná operácia nech trvá až 30 sekúnd. Ak stlačíme tlačidlo pre vykonanie tejto operácie a po krátkej chvíli neuvidíme, že sa niečo deje, tak potom sa môže stať, že stlačíme tlačidlo pre vykonanie príkazu viackrát za sebou, napríklad až 20-krát. Po 30 sekundách sa objaví na obrazovke výsledok prvého stlačenia tlačidla, ale pretože sme tlačidlo stlačili ešte 20-krát za sebou, tak systém bude opakovať zadanú operáciu a nebude schopný vykonávať nič iné ešte ďalších 10 minút.

Z toho plynie ponaučenie, že ak systém nemôže okamžite odpovedať na zadaný príkaz, potom by sa mal v systéme existovať nejaký indikátor, ktorý nás bude informovať o tom, že príkaz bol prijatý a že sa vykonáva. Napríklad veľa systémov zobrazuje ikonu presýpacích hodín, ktorá informuje používateľa, že má čakať na odpoveď. Presýpacie hodiny však poskytujú minimum informácií. Hovoria len o tom, že systém pracuje, ale čo konkrétne sa vykonáva, to nevieme.

Informatívnejšia spätná väzba by nás mala informovať o tom, ktorý príkaz systém prijal, čo sa vykonáva ako dlho to bude trvať. 

Správnym riešením môže byť napríklad **bar graf** zobrazený na obr.6. Používateľ intuitívne pochopí, že proces je ukončený, ak bar dosiahne vrchol grafu.



Obr. 6 Informatívna spätná väzba

Intenzita spätnej väzby by mala odrážať dôležitosť situácie. Obyčajné stlačenie tlačidla by malo spôsobiť napríklad slabý zvukový signál. Akcia väčšej dôležitosti, ako napríklad spustenie veľkého a nebezpečného stroja, by mala byť sprevádzaná silnejším zvukovým signálom a poprípade aj rozsvietením veľkej indikačnej lampy. Niektoré zariadenia vydávajú pípajúce zvuky po každom stlačení klávesy alebo tlačidla, čo môže pôsobiť dosť rušivo na samotného používateľa, ale ja na ľudí nachádzajúcich sa v jeho tesnej blízkosti.

SAMOHODNOTIACE OTÁZKY

- Uveďte príklady realizácie spätných väzieb v jednotlivých zariadeniach.
- Vysvetlite dôležitosť doba odozvy spätnej väzby.
- Čo znamená pojem informatívna spätná väzba?
- Čo odráža intenzita spätnej väzby?

2.6 Tolerantnosť k chybám

Ľudský jazyk je vo všeobecnosti celkom **tolerantný** k chybám.

Ak je nesprávne nejaké slovo použité vo vete alebo chýba, potom si vieme sami bez problémov domyslieť správny význam celej vety. Počítače sú ale od podstaty, celkom odlišné. Použitie **čiarky namiesto bodkočiarky** v počítačovom programe môže spôsobiť jeho celkové zlyhanie. Prax je taká, že používatelia nie sú vždy dostatočne pozorný pri zadávaní počítačových príkazov.

V ideálnom prípade by mali byť počítače tiež tolerantné k chybám. Napríklad vieme už zabezpečiť, aby počítač akceptoval alternatívne zadávanie príkazov alebo webových adries, to znamená, že pri vynechaní nejakého písmena si ho doplní sám alebo nám zobrazí správnu **www stránku**.

Problém vznikne vtedy, ak počítač neopraví našu chybu správne. V tom prípade musíme zabezpečiť, aby ponúknutá oprava bola vždy potvrdená používateľom.

AKTIVITA

- Vyhľadajte príklady alternatívneho zadávania príkazov v programe MS Internet Explorer?
- Uveďte príklady alternatívneho zadávania príkazov pri práci v iných programoch.

2.7 Návrh ovládacích prvkov

Medzi najpoužívanejšie ovládacie prvky patria **tlačidlá**.

Tlačidlá na riadiacom paneli alebo na klávesnici by mali byť usporiadané tak, že tie, ktoré sú najčastejšie používané, by mali byť najväčšie a mali by mať prominentnú polohu, zatiaľ čo tlačidlá, ktoré sa používajú zriedka, by mali byť menšie a umiestnené niekde v okrajových častiach. Tlačidlá používané pre základné nastavenia alebo konfiguráciu by mali byť umiestnené pod nejakými krytmi tak, aby bolo jasné, že to nie sú tlačidlá pre bežné používanie.

Napríklad na fotokopírke máme jedno tlačidlo, ktoré používame stále, ak chceme kopírovať. Je to veľké zelené tlačidlo, ktoré musíme stlačiť, keď chceme urobiť kópiu. Zelené tlačidlo nie je väčšie ako ostatné tlačidlá. Preto mnoho používateľov namiesto tohto tlačidla stláča sivé tlačidlo **ON/OFF** umiestnené vpravo (obr.7).

Obr. 7 Ovládacie prvky - fotokopírka

Existujú však tlačidlá, ktorých stlačenie má oveľa väčšie dôsledky ako spomínaný príklad. Napríklad hlavné tlačidlo prívodu elektrickej energie v nemocnici!!! Také tlačidlo by samozrejme v žiadnom prípade nemalo vyzerať ako ostatné tlačidlá.

Tento princíp si môžeme ilustrovať aj na príklade Rádia X. Rádio X je malé neziskové rádio určené pre menej vzdelaných poslucháčov. Noví poslucháči musia vždy stráviť určitý čas v nahrávacom štúdiu, aby sa naučili ako používať jednotlivé zariadenia. V nahrávacom štúdiu si prehrávajú hudbu, rozprávajú do mikrofónu rôzne veci a potom tie nahrávky spolu mixujú a pod. V tomto štúdiu je jedno tlačidlo, ktoré spúšťa vysielanie do éteru, a ktoré, ako im bolo povedané, oni nesmú stlačiť. Napriek tomu sa často stáva, že tlačidlo je stlačené. Výsledok je to, že tisíce ľudí počúvajú rôzne hlúposti povedané do mikrofónu a že naviac iná stanica, ktorá vysiela na rovnakej frekvencii, nemôže vysielať.

Riešením tohto problému je náhrada tohto **„on air“** tlačidla iným bezpečnejším tlačidlom. Takým tlačidlom môže byť napríklad tlačidlo so zámkom a kľúčom, ktoré hneď na prvý pohľad vyzerá odlišne od ostatných tlačidiel. Zámok umiestnený na tlačidle akoby hovoril: **„Tu je zakázaný prístup! Otočenie kľúča bude mať veľké následky!“.** Kľúč bol vždy umiestnený v zámku a predsa sa už nikdy nestalo aby bolo tlačidlo zatlačené.

Návrh tlačidla by mal odrážať jeho dôležitosť !!!

Obr. 8 Rôzne typy tlačidiel

AKTIVITA

- Čo podľa Vás vyjadrujú jednotlivé vypínače na obr.2.5? Uveďte príklady v akých situáciách alebo systémoch by ste ich používali.

SAMOHODNOTIACE OTÁZKY

- Kde by mali byť umiestnené na riadiacom paneli alebo klávesnici tlačidlá, ktoré sa používajú najčastejšie a kde naopak tlačidlá, ktoré sa používajú zriedka?
- Akým spôsobom by mali byť odlíšené tlačidlá, ktoré sa používajú na konfiguráciu a nastavovanie dôležitých parametrov systému?
- Ktorý základný princíp treba dodržiavať pri návrhu tlačidiel?.

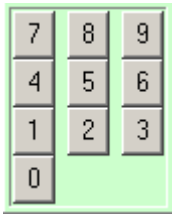
2.8 Štandardizácia

Ktorým smerom by sme otočili gombík na rádiu, aby sme ho dali hlasnejšie?

Obr. 9 Otočný gombík rádia

Samozrejme v smere hodinových ručičiek. Nepotrebujeme na to žiadne inštrukcie a nikto sa určite nepomýli, pretože ovládanie hlasitosti je vždy v smere hodinových ručičiek. Výhoda takto zaužívaného **štandardu** je zrejmá. Výnimkou takto zaužívaného štandardu je vodovodný kohútik, ktorý sa otáča v protismere hodinových ručičiek, ak ho chceme otvoriť.

Numerické klávesy na klávesnici počítača alebo kalkulačke sú umiestnené ako je uvedené na obr.10.



Obr. 10 Numerické klávesy na klávesnici počítača

Ale klávesy na diaľkovom ovládači TV sú umiestené tak, ako je zobrazené na obr. 11.

Obr. 11 Numerické klávesy na diaľkovom ovládači TV

Klávesy na telefóne môžu byť umiestnené ešte **iným spôsobom**.

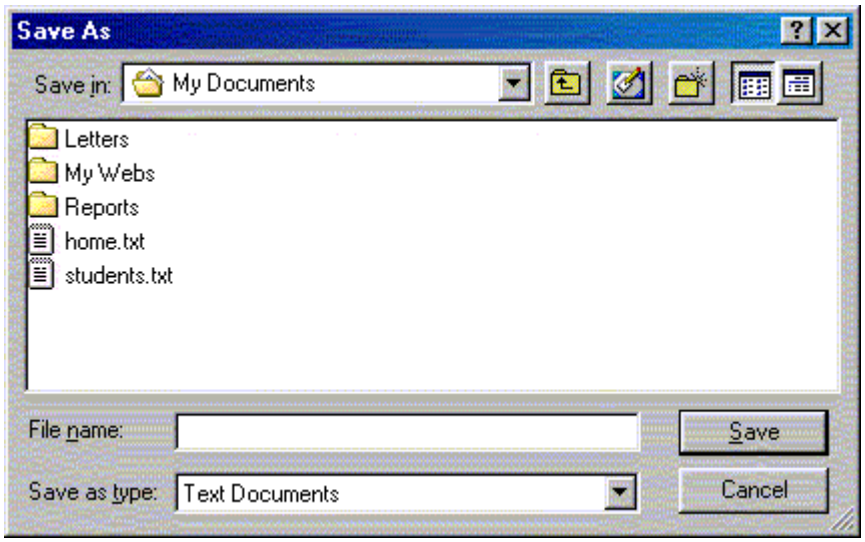
Z toho vyplýva, že používanie štandardu pri návrhu numerickej klávesnice je zrejme problémom, pričom ak je rozloženie klávesnice rovnaké, používateľ pracuje rýchlejšie a robí menej chýb. Rovnako slepí ľudia by určite uprednostnili telefóny s rovnakým rozložením kláves na klávesnici.

Obr. 12 Usporiadanie numerických klávesníc na rôznych zariadeniach

Ak navrhujeme ovládací panel alebo iné zariadenie s používateľským rozhraním, mali by sme sa predovšetkým uistiť, či v danom prípade neexistujú nejaké **štandardy** alebo **všeobecné zaužívané praktiky pre používanie farieb, tvarov, umiestnenia tlačidiel a pod.**

To isté platí aj pre užívateľské rozhranie navrhovaného softvéru. **Konzistencia** je dôležitá vo vnútri samotného systému ako aj medzi systémami navzájom.

Je Vám dialógové okno na obr. 13 známe?



Obr. 13 Dialógové okno pre ukladanie súborov

Dialógové okienko pre ukladanie súborov vyzerá a pracuje rovnako vo všetkých programoch OS Windows. Pre užívateľov je to veľká výhoda, pretože môžu pracovať v novom programe okamžite bez toho, aby sa museli učiť ako ukladať súbory a pod. Ak vytvárame nový program, potom by sme mali dbať na to, aby jeho menu bolo podobné ako menu v iných programoch (t.j. **File** naľavo, **Help** napravo) a tam, kde je to možné používať prvky rozhrania operačného systému. Niektorí programátori vytvárajú v programoch vlastné menu, aby do nich pridali nejaké zábavné prvky či dekorácie. Mali by sme sa vyhnúť takému pokušeniu z toho dôvodu, že používateľ bude musieť stráviť viac času, kým sa naučí pracovať s takýmto rozhraním.

Webové stránky sú najmenej štandardizované, ale potreba **štandrardizácie** stále stúpa, pretože neustále narastá počet nových používateľov, ktorí webové stránky navštevujú po prvýkrát. Ak má používateľ problém zistiť ako používať stránku a kde kliknúť, potom sa vráti späť a použije inú webovú stránku. Hypertexové odkazy by mali byť podčiarknuté a zvýraznené pokiaľ je to možné modrou farbou. Mnoho tvorcov web stránok však nepoužíva podčiarknutie z estetického dôvodu. Neexistujú ani štandardy pre vytváranie menu na web stránkach, čo vedie k tomu, že rôzne web stránky používajú rôzne riešenia. Menu umiestnené na hornom okraji web stránky by malo byť podobné ako menu v programoch OS Windows, tak aby ho vedel používať aj menej skúsený používateľ. Pravdaže preferuje sa používanie aj ďalších zaužívaných štandardov.

AKTIVITA

- Porovnajte rozloženie kláves na klávesniciach telefónov na obr.2.9.
- Vyberte si jednu klávesnicu, ktorá by vám vyhovoval najviac z hľadiska rozloženia kláves a vysvetlite prečo?
- Pokúste sa vo svojom okolí nájsť klávesnicu s odlišným rozložením kláves ako na obr.2.9.

SAMOHODNOTIACE OTÁZKY

- Čo rozumieme pod pojmom štandardy?
- Uveďte príklady zaužívaných štandardov, s ktorými ste sa stretli pri svojej práci?
- Uveďte príklady štandardov, s ktorými ste sa stretli na webových stránkach pri práci s Internetom.

2.9 Otvorené štandardy

Všetky technické výrobky, ktoré sú prepojené s inými výrobkami musia mať **štandardizované rozhranie**.

Napríklad:

- Telefón potrebuje štandardnú zásuvku, štandardnú úroveň napätia, štandardné číselné kódy
- Textový procesor potrebuje ukladať súbory v štandardizovanom formáte, aby boli čitateľné aj v iných textových procesoroch
- Tlačiareň potrebuje štandardnú veľkosť papiera, štandardné kódovanie textov, štandardný toner
- Všetky zariadenia pripájané k počítačovej sieti potrebujú štandardizované zásuvky a komunikačné protokoly

Existujú rôzne druhy štandardov a rôzne úrovne štandardizácie:

Utajený štandard

Výrobky majú byť kompatibilné iba s výrobkami tej istej spoločnosti, ktorá ich vyrába. Iné spoločnosti musia „**hacknúť**“ tieto výrobky, ak chcú vyrobiť niečo, čo je s nimi kompatibilné.

Patentovaný štandard

Štandard vlastní spoločnosť alebo je chránený patentom alebo autorskými právami. Iné spoločnosti musia zaplatiť licenčné poplatky, ak chcú používať rovnaké štandardy.

Štandard de facto

Ide o štandardy, ktoré sa vyvíjajú v tom prípade, ak niekoľko spoločností chce vyrábať výrobky, ktoré budú navzájom kompatibilné alebo kompatibilné s inými výrobkami. V tom prípade neexistuje nijaká oficiálna dohoda, ani snaha chrániť vyvinuté štandardy nejakým patentom alebo autorskými právami.

Oficiálny štandard

Štandard schvaľuje a uchováva konkrétna oficiálna spoločnosť. Všetky technické detaily sú presne špecifikované a zverejnené.

Otvorený zdroj

Softvérový produkt je vytváraný pre ľubovoľných používateľov a jeho zdrojový kód je zverejnený bez akýchkoľvek obmedzení a užívateľských práv. Hocikto ho môže modifikovať alebo vytvárať ďalšie s ním kompatibilné softvérové produkty.

SAMOHODNOTIACE OTÁZKY

- Uveďte príklady štandardizovaných rozhraní, s ktorými ste sa stretli pri práci alebo sa nachádzajú vo vašom okolí.
- Vymenujte jednotlivé druhy štandardov a uveďte ich krátku charakteristiku.
- Zistite, aké druhy štandardov používa firma Microsoft, Siemens, Philips.

2.10 Základné charakteristiky správne navrhnutého rozhrania

Dobré navrhnuté používateľské rozhranie by malo mať nasledovné charakteristiky:

- **Je jednoduché a zodpovedá účelu, tzn. že nezobrazuje viac alebo menej informácií ako je potrebné,**
- **je maximálne názorné a zreteľné, priťahuje pozornosť na dôležité informácie tak, aby sa používateľ mohol rýchlo a správne rozhodnúť predovšetkým v neštandardných situáciách,**
- **je zhodné na viacerých úrovniach, tzn. že rovnaké symboly či farby majú rovnaký význam na odlišných obrazovkách a užívateľ tak vie, čo môže očakávať v rôznych situáciách**
- **nápoveda je prístupná z každej úrovne pomocou toho istého funkčného kľúča, ktorý je jasne označený; moderné systémy ponúkajú tzv. kontextové nápovedy, ktoré rozpoznávajú jednotlivé činnosti a situácie a ponúkajú nápovedu práve k nim.**
- **koncepcia viacerých obrazoviek je poprepájaná pomocou funkčných kľúčov, pričom najčastejšie používané a najdôležitejšie obrazovky sú dostupné vždy a čo najjednoduchším spôsobom.**
- **pre používateľa je veľmi dôležité, aby po zadaní príkazu získal okamžite "pocit", že príkaz bol prijatý a akceptovaný. Preto:**
  - **po zadaní príkazu (najčastejšie stlačením kláves RETURN alebo ENTER) by mala nasledovať okamžitá odozva systému alebo komunikačný dialóg či potvrdzovacia správa, ak je odozva systému dlhšia;**
  - **vplyv príkazu by mal byť okamžite viditeľný a vždy by mala existovať možnosť jeho okamžitého pozastavenia.**
- **Existujú tri hlavné aspekty tvorby interfejsu ako je ergonómia, použiteľnosť a funkčnosť, pričom poradie v akom je potrebné sa nimi zaoberať pri tvorbe interfejsu záleží len od schopnosti a postavenia operátora v jednotlivých situáciách.**

- správne navrhnuté rozhranie, nielenže **spríjemňuje prácu**, ale vedie aj k **redukcii chýb** a **obmedzeniu vzniku škôd**.

V súčasnosti **neexistujú** všeobecné postupy a merania koľko práce a úsilia potrebuje vynaložiť operátor pri práci s konkrétnym interfejsom, teda neexistuje vhodné kritérium vyhodnotenia ich kvality. Avšak aspekty ako je motivácia, vyššia výkonnosť a celkové uspokojenie z práce s daným interfejsom je potrebné pri jeho návrhu zohľadňovať.

AKTIVITA

- Na webovej stránke www.baddesigns.com nájdete súbor jednoduchých príkladov zle navrhnutých používateľských rozhraní. Prezrite si jednotlivé prípady a poučte sa z nich.
- Pohľadajte vo svojom okolí zariadenie, ktoré nespĺňa základné charakteristiky dobre navrhnutého používateľského rozhrania. Uveďte, čo kde sú chyby a problémy a navrhnite ich riešenie.

3 PROCES NÁVRHU POUŽÍVATEĽSKÉHO ROZHRANIA

Táto kapitola sa zaoberá špecifikáciou jednotlivých etáp návrhu používateľského rozhrania. Vedomosti o charakteristike týchto etáp a dôležitých postupov, sú rovnako dôležité ako znalosť všeobecných princípov jeho návrhu, pretože zvyšujú mieru **použiteľnosti** takto navrhnutého rozhrania.

3.1 Úvod a ciele

Pri realizácií používateľské rozhrania sa musíme pridržiavať všeobecných princípov jeho návrhu, pričom musíme brať do úvahy jednotlivé faktory ovplyvňujúce ich výber. Samotný proces návrhu pozostáva z jednotlivých etáp a postupov, ktorých dodržiavanie je rovnako dôležité ako dodržiavanie základných princípov návrhu.

CIELE

Cieľom tejto kapitoly je naučiť sa:

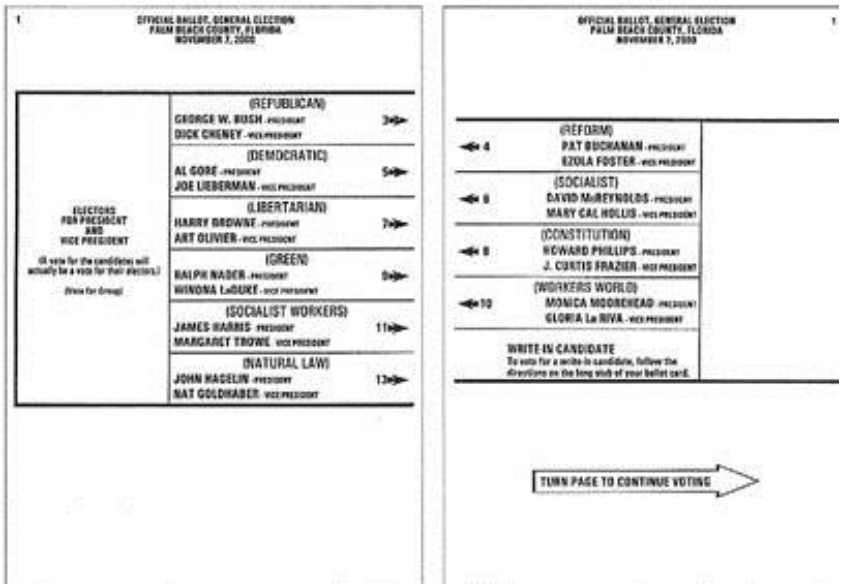
- akú úlohu hrá profil používateľa pri tvorbe používateľského rozhrania
- prečo je dôležité zahrnúť používateľa do procesu návrhu používateľského rozhrania
- aký význam má test použiteľnosti v procese tvorby používateľského rozhrania
- prečo je dôležité sledovať spätnú väzbu od používateľa a neustále sledovať jeho správanie vo vzťahu k používateľskému rozhraniu a na čo nám budú slúžiť takto získané informácie.

3.2 Profil používateľa

Pre začiatkom návrhu výrobku alebo softvéru si musíme urobiť **profil používateľa** a zistiť v akých situáciách sa bude používať:

- Budú ho všetci používatelia používať skoro každý deň alebo len príležitostne?
- Bude každý používateľ využívať len určitú špecifickú časť produktu alebo celý produkt?
- Bude produkt využívať len určitá kategória používateľov alebo bude určená pre najširšiu škálu používateľov vrátane detí, starších či handicapovaných ľudí
- Sú používatelia motivovaní používať tento produkt zlepšením ich pracovných podmienok alebo používanie produktu bude súčasťou ich pracovnej náplne?
- Budú používatelia sklamaní, ak bude dlho trvať naučiť sa produkt používať?
- Majú používatelia nejaké špeciálne školenia alebo vedomosti?
- Bolo používatelia už zaškolení do používania podobného produktu?

- Budú používatelia schopní zadávať príkazy v jednom jazyku alebo je potrebné vytvoriť zadávanie príkazov vo viacerých jazykoch?
- Rozumejú používatelia technickým výrazom používaným v príkazoch?



Obr.14 Ukážka volebného lístku

Na obr.14 je zobrazený volebný lístok prezidentských volieb v štáte Floride v roku 2000. Voliči mali v strede volebného lístka urobiť dierku vedľa mena vybraného kandidáta, ku ktorému bola priradená šípka s poradovým číslom. Len niekoľko stovák voličov odovzdalo správne označené volebné lístky. Zvyšok väčšinou urobil dierku na mieste, kde sa nedalo identifikovať, ktorému kandidátovi prislúcha, alebo sa dopustili iných chýb.

Problém bol v tom, že tvorcovia volebného lístka nezobrali do úvahy profil používateľa. Volebný lístok mal byť jasne zrozumiteľný pre všetkých dospelých, vrátane ľudí so slabším zrakom alebo iným postihnutím.

AKTIVITA

- Vytvorte formulár, pomocou ktorého by ste zistili profil používateľa pre používanie vizualizačnej aplikácie monitorovania teplôt a tlakov teplárne.

3.3 Používateľ v procese návrhu

Potenciálni používatelia by mali byť od začiatku **zahrnutí do procesu návrhu.**

Rozoznávame dva typy používateľov:

Skúsení používatelia

Sú to používatelia, ktorí majú dlhodobé skúsenosti s využívaním podobných produktov (predchádzajúce verzie produktov alebo podobné produkty iných firiem). Skúsení používatelia majú mnoho návrhov a pripomienok ako zlepšiť produkty, ktoré poznajú.

Nováčikovia

Sú to používatelia, ktorí budú náš produkt potrebovať, ale nikdy predtým nepoužívali podobný. Sú užitočný na to, aby sa pýtali rôzne nezmyselné otázky, ktoré by nás nenapadli.

Obidva typy používateľov je potrebné zahrnúť do testu použiteľnosti hneď na začiatku návrhu alebo prvých skúšok vytvoreného produktu.

SAMOHODNOTIACE OTÁZKY

- Ktoré dva základné typy používateľov rozoznávame v procese návrhu?
- Charakterizujte skúsených používateľov a nováčikov
- Uveďte dôvody prečo je potrebné potenciálnych používateľov zahrnúť do procesu návrhu.
- Do ktorej skupiny používateľov softvérových produktov by ste sa zaradili?

3.4 Test použiteľnosti

Test použiteľnosti je najdôležitejšia časť návrhu produktu, pretože na jeho základe môžeme odhaliť mnohé nedostatky a chyby.

Princíp testu použiteľnosti je veľmi jednoduchý: Požiadame niekoho, aby produkt používal a sledujeme ako sa pokúša s ním pracovať. Všetky problémy, ktoré objaví si zapíšeme. Nezabudnime, že **cieľom testu** nie je dokázať, že produkt pracuje správne, ale **zistiť chyby a problémy**.

Test použiteľnosti môžeme aplikovať na rôzne druhy hardvéru: otvárač konzerv, budík, lietadlo, a tiež ľubovoľný druh softvéru: web stránky, video hry, textový procesor alebo vedecké programy.

Druhy chýb, s ktorými sa môžeme stretnúť:

- Používateľ má problém zistiť ako produkt používať.
- Používateľ chce, aby produkt vykonával niečo, čo nemôže.
- Produkt nerobí to, čo od neho používateľ očakáva.
- Používateľovi sa nepodarilo objaviť užitočné vlastnosti produktu.
- Používateľ používa produkt zložitým spôsobom.

- Používateľ je unavený alebo má ergonomické problémy.
- Používateľ zistil funkčné chyby produktu.

Existuje niekoľko spôsobov ako urobiť **test použiteľnosti**. **Najpoužívanejšie metódy** sú:

- Pohovor s používateľom
- Poprosiť používateľa, aby hlasno rozmýšľal, keď sa pokúša zistiť ako produkt používať
- Sledovať používateľa pri narábaní s produktom

Všetky tieto metódy testovania je možné uskutočniť v bežných laboratórnych podmienkach.

Výsledok testu použiteľnosti závisí od typu ľudí, ktorých sme na test vybrali.

Príklady **používateľov vhodných na testovanie**:

Nováčikovia

Tento typ ľudí nemá žiadne predchádzajúce skúsenosti s používaním daného typu produktu, preto bude mať problémy zistiť ako ho treba používať a tak objaví mnoho chýb a nedostatkov.

Skúsení používateľ

Sú to ľudia, ktorí majú veľa skúseností s používaním podobných produktov, preto budú vedieť ako produkt vylepšiť a kde treba hľadať problémy.

Staršie neskúsené osoby

Starší ľudia sa učia pomalšie a majú tiež znížené zmyslové a motorické schopnosti. Problém im robí napríklad dvojklik myši.

Handicapovaní používatelia

Handicapovaní používatelia sú rovnako užitočný pre testovanie nášho produktu. Ak nevedia s ním pracovať jednoduchým spôsobom, potom je potrebné zabudovať do systému prvky, ktoré ho urobia pre nich prístupnejším.

Deti

Deti sú zvedavé a podnikavé. Chcú vyskúšať všetko a tak otestovať hranice nášho produktu.

Zásadoví používatelia

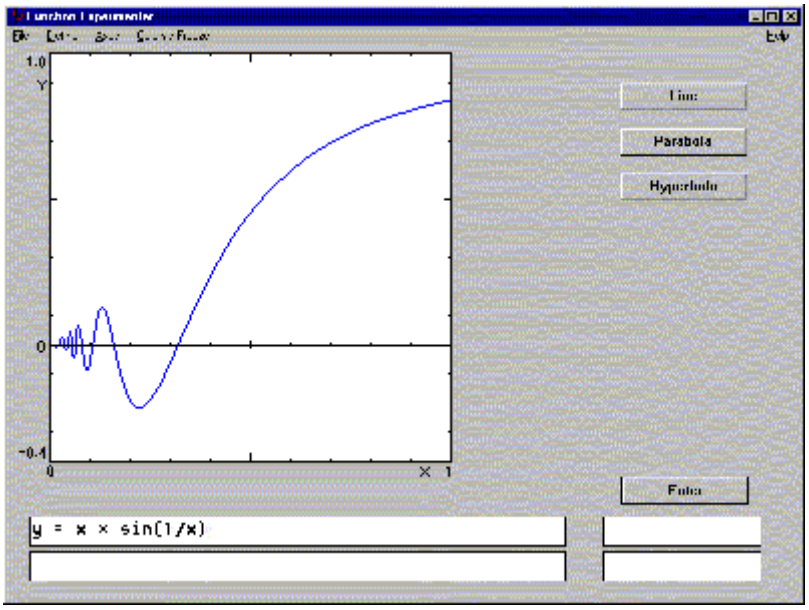
Sú to typy ľudí, ktorí predtým ako stlačia tlačidlo **ON** na našom produkte musia prečítať celý manuál a všetky jeho prílohy. Takýto typ ľudí nám pomôže nájsť všetky chyby v tlačených manuáloch.

Progresívny a nadšení mladý používatelia

Chcú vyskúšať všetky možnosti vrátane **Helpu**. Ak ich požiadame, aby našli chyby, tak to budú považovať za výzvu alebo akýsi druh hry, ktorý bude mať určite požadované výsledky.

Príklad

Predstavme si, že chceme testovať program vykresľujúci priebehy matematických funkcií



Obr. 15 Priebeh testovanej funkcie

Naša prvá testujúca osoba bol starší inžinier zbehlý v matematike.

Do riadku vzorcov napísal funkciu: **y = x + 1** a stačil kláves **Enter**

Program vykreslil priamku.

Potom napísal: **y = x +2**

Program vykreslil inú priamku rovnobežnú s predchádzajúcou.

Nakoniec napísal: **y = x +8** a skončil test so slovami: **Program pracuje v poriadku.**

Ďalšou testujúcou osobou bol mladý progresívny technik. Do riadku vzorcov hneď napísal všetky možné kombinácie čísel a symbolov bez ohľadu na syntax. Aj keď to môže znieť hlúpo, bola to jedinečná príležitosť ako otestovať, či program generuje chybové hlásenia a s akým obsahom.

Tak sme zistili, že tento muž našiel viac chýb ako prvá testujúca osoba.

Tento príklad ukazuje aké je dôležité mať **viac ako jednu testujúcu osobu**. Jedna testujúca osoba nikdy nemôže nájsť všetky chyby použiteľnosti nášho produktu. Platí, že čím viac testujúcich osôb máme, tým viac chýb a nedostatkov nájdeme.

Zaujímavé je, že tento problém nezávisí len od testujúcej osoby, ale aj od pozorovateľa. Preto dobrý test použiteľnosti by mal zahrňovať viac testujúcich osôb a tiež viac pozorovateľov.

Poznáme nasledujúce **typy pozorovateľov**:

Návrhár

Osoba, ktorá sa zúčastňuje na technickej konštrukcii zariadenia. Sleduje, či sa používateľ nespráva inak ako sa očakávalo. Často inklinuje k záverom, že chybu robí používateľ a nie zariadenie.

Iný používateľ

Osoba, ktorá má viac vedomostí o technickej štruktúre zariadenia ako testujúca osoba, preto môže lepšie porozumieť problémom, s ktorými sa stretla testujúca osoba.

Expert - používateľ

Tento typ pozorovateľa má skúsenosti s pozorovaním a sledovaním problémov testujúcej osoby, ale nie je oboznámený so špecifickými problémami v oblasti aplikácií daného zariadenia.

STUPNE VÝVOJA

Musíme si uvedomiť, že snaha vyriešiť jeden problém, môže priniesť iné problémy. To znamená, že musíme produkt testovať znova.

Úplný proces vývoja nového produktu by mal zahrňovať **test použiteľnosti**, pozostávajúci **z niekoľkých etáp**:

Stará verzia

Pred začatím vývoja nového produktu by sme mali testovať podobné produkty, aby sme zistili, čo sa dá v nich zlepšiť

Prototyp

Vývoj prototypu slúži na to, aby sme mohli urobiť test použiteľnosti. Dokonca aj návrh používateľského rozhrania na kúsku papiera môže byť použitý pre test použiteľnosti.

Beta verzia

Beta test môže obsahovať aj test použiteľnosti.

Konečný produkt

Konečný produkt by mal byť vždy testovaný.

Spätná väzba od zákazníkov

Žiaden test použiteľnosti nemôže nájsť všetky nedostatky a chyby, pretože nezahrňuje všetky situácie, ktoré sa môžu vyskytnúť v reálnom živote pri bežnom používaní produktu.

SAMOHODNOTIACE OTÁZKY

- Na čo slúži test použiteľnosti a aký je jeho princíp?
- Vymenujte aspoň 4 druhy chýb, s ktorými sa môžeme stretnúť pri teste

	použiteľnosti. • Vymenujte najpoužívanejšie metódy realizácie testu použiteľnosti. • Od čoho závisí výsledok testu použiteľnosti? • Vymenujte skupiny používateľov vhodných na testovanie a uveďte ich krátku charakteristiku. • Stačí pre test použiteľnosti vybrať jednu testujúcu osobu? • Na kom závisí výsledok testu použiteľnosti okrem testujúcej osoby? • Vymenujte typy pozorovateľov z hľadiska testu použiteľnosti. • Vymenujte etapy testu použiteľnosti a každú etapu stručne charakterizujte.

	**AKTIVITA** • Predstavte si, že ste vytvorili vizualizačnú aplikáciu pre riadenie dopravníka sypkých materiálov. Navrhnite štruktúru osôb potrebných pre realizáciu testu použiteľnosti, t.j. počet a typy používateľov ako testujúcich osôb, počet a typy pozorovateľov.

3.5 Spätná väzba od používateľa

Spätná väzba od používateľa je perfektným zdrojom informácií, ktoré môžu byť vo veľkej miere použité pre zlepšenie vztvoreného produktu. Mnoho výrobcov však nevyužíva túto možnosť. Považujú ich za otravných dotieravcov a v žiadnom prípade ich systematickým spôsobom nevyužijú pre zlepšenie vlastností produktu.

Je veľmi dôležité implementovať do procesu návrhu procedúry zohľadňujúce požiadavky zákazníka. Všetky otázky, návrhy, sťažnosti a požiadavky používateľov by mali byť zhromažďované a štatisticky vyhodnocované. Ak má niekoľko používateľov rovnaký problém, potom je to určite používateľská chyba. Používatelia by mali byť odmeňovaní za podávanie takýchto správ, napríklad ďakovným listom, kde im oznámime, že na probléme sa pracuje a v najbližšej možnej dobe sa bude pracovať na jej odstránení a pod.

Implementovanie spätnej väzby od používateľa by malo byť súčasťou odboru riadenia kvality každej organizácie.

Webové stránky môžu byť využité na informovanie zákazníkov ako odstrániť chybu.
Veľa ďalších informácií vieme získať neustálym **sledovaním správania** sa používateľov. Môžeme to dosiahnuť napríklad zverejnením **web stránky**, na ktorej budú môcť používatelia vyplniť dotazníky, týkajúce sa problémov, s ktorými sa stretli počas používania produktu alebo zapísať svoje poznatky a postrehy. Tieto informácie sa potom štatisticky vyhodnotia a mali by byť samozrejme anonymné.
Ak zo štatistického prehľadu zistíme, že mnoho používateľov robí tú istú chybu alebo má ten istý problém, potom musíme tento problém riešiť z hľadiska **použiteľnosti**.
Štatistiky nám môžu tiež pomôcť pri zatrieďovaní používateľov do jednotlivých kategórií. Ak sú napríklad používatelia, ktorí používajú funkciu A a tiež funkciu B a na druhej strane

používatelia, ktorí používajú funkcie C a D, potom môžeme využiť túto informáciu pri návrhu menu v systéme tak, že pre používateľov funkcie A a B vytvoríme jedno podmenu a pre používateľov funkcie C a D druhé podmenu.

SAMOHODNOTIACE OTÁZKY

- Na čo slúži v procese návrhu spätná väzba od používateľa?
- Akým spôsobom je potrebné vyhodnocovať zhromažďované požiadavky používateľov?
- Akým spôsobom motivovať požívateľov, aby predkladali svoje návrhy, sťažnosti a požiadavky? Uveďte aj vlastné nápady!

4 TECHNICKÉ PROSTRIEDKY POUŽÍVATEĽSKÝCH ROZHRANÍ

Táto kapitola sa zaoberá systematickým pohľadom na jednotlivé typy vstupno/výstupných zariadení v systémoch MMI, ich stručnou špecifikáciou a výhodami a nevýhodami ich použitia v priemyselných aplikáciách.

4.1 Úvod a ciele

Podľa definície systém MMI tvorí súhrn technických a programových prostriedkov. Najčastejšie používaným zariadením v systémoch MMI sú **počítačové terminály**. Väčšina systémov MMI pozostáva z počítačového terminálu, klávesnice a myši. Tento typ hardvéru je ľahko prístupný, lacný a primeraný na postavenie dobrého používateľského rozhrania, v ktorom musí byť dôraz kladený na správnu interakciu systémových komponentov a na správne kódovanie správ a príkazov. Za nesprávny návrh používateľského rozhrania v žiadnom prípade nezodpovedá hardvér.

CIELE

Cieľom tejto kapitoly je naučiť sa:

- ktoré vstupno/výstupné zariadenia sa používajú v systémoch MMI pre priamy vstup-výstup, aké sú ich základné charakteristiky a možnosti použitia,
- ktoré zariadenia s priamym riadením povrchu obrazovkového terminálu sa používajú v systémoch MMI, aké sú ich základné charakteristiky a možnosti použitia,
- ktoré zariadenia s priamym riadením kurzora na obrazovke sa používajú v systémoch MMI, aké sú ich základné charakteristiky a možnosti použitia,
- aké najmodernejšie technické zariadenia sa používajú pri návrhu systémov MMI, aké sú výhody a nevýhody ich použitia.

4.2 Zariadenia pre priamy vstup/výstup

Niektoré zariadenia MMI sa používajú veľmi často, iné zriedkavo a niektoré len v špeciálnych aplikáciách.

Najdôležitejšie I/O zariadenia **pre priamy vstup/výstup** v priemyselných aplikáciách sú:

- obrazovkové terminály
- klávesnice
- špeciálne funkčné kľúče
- terminál pre tlač
- tlačiareň
- riadiaci panel

Obrazovkové terminály - spolu s klávesnicami sú najpopulárnejšími zariadeniami výmeny dát pre praktické riadiace aplikácie. Hlavnou **nevýhodou** terminálov je ich citlivosť na prostredie s vysokou úrovňou elektromagnetického vlnenia, vibrácií, vlhkosti a prachu. Existujú špeciálne terminály, ktoré sa používajú v prostredí priemyselných prevádzok. Pri používaní klávesnice je nevyhnutná určitá voľnosť prstov, čo môže byť problém v prevádzkach, kde sa používajú a sú nevyhnutné rukavice.

Základné princípy použitia:

- Jas počítačovej obrazovky ba mal byť taký, aby boli informácie zobrazené na nej ľahko čitateľné, ale na druhej strane, aby nás neboleli oči.
- Jas by mal byť vždy ľahko nastaviteľný.
- Obrazovka by mala byť umiestnená tak, aby zozadu nedopadalo na ňu veľa svetla a neodrážalo sa.
- Tmavý text na svetlom pozadí umožňuje väčšie rozlíšenie detailov, ale veľmi svetlé pozadie je škodlivé pre oči.
- Svetlý text na tmavom pozadí je pre čítanie najvýhodnejší.

Klávesnice

V 90-tych rokoch boli v móde tzv. **tiché klávesnice**. Používatelia boli pri kúpe fascinovaní, pretože klávesnica reagovala okamžite rozsvietením svetielka pod klávesou. Ale „tichá“ klávesnica nie je vhodná pre skúsených používateľov. Ak náhodou nestlačíme klávesu úplne, potom nie sme si istí , či bola akceptovaná alebo nie. Ešte pred stlačením ďalšej klávesy sa musíme pozrieť na obrazovku počítača urobiť kontrolu, čo preruší našu prácu a môže nás to vyviesť z koncentrácie.

Preto by sme na klávesnici mali mať vždy pocit „**kliknutia**“ pri jej stlačení. Ide o tzv. **taktickú spätnú väzbu**, keď na základe kliknutia presne vieme, či sme kláves stlačili, či sme urobili dvojklik a pod.

Základné princípy návrhu:

- Veľkosť a vzdialenosť medzi klávesami musí byť prispôsobená prstom.
- Klávesy sa nesmú ťažko stláčať, ale nie ani príliš ľahko
- Klávesy by mali byť logicky organizované do funkčných skupín
- Klávesy, ktoré sa používajú zriedka, ba mali byť menšie a umiestnené na okraji
- Medzery medzi skupinami kláves umožňujú ich rozlíšenie

QWERTY rozmiestnenie alfanumerickej klávesnice nie je optimálne, ale stalo sa štandardom. Lepšie sú iné rozmiestnenia, napr. QWERTZ.

Lacným a praktickým vstupno/výstupným zariadením je tzv. **"printing" terminál**. Používa sa vtedy, ak sa informácia mení vo väčšom časovom rozsahu (1-2 udalosti/min) a každá informácia je obsahovo samostatná, t.j. nenadväzuje na iné informácie), takže používateľ nemusí dlho čakať na výstup, aby získal kompletný obraz situácie. Výhodou "printing" terminálov je, že ich zobrazovacím médiom je papier. Ten môže byť archivovaný ako záznam, ktorý nepotrebuje ďalšie spracovanie.

Riadiace panely vznikli so vznikom interfejsových zariadení. Za účelom riadenia systémov boli navzájom prepojené príkazmi pochádzajúcimi z centrály. Riadiace panely majú indikátory (lampy, ručičkové prístroje, ...) pre výstup dát a prepínače alebo klávesnice pre vstup dát. Ich nevýhodou je to, že sa môžu používať len pre obmedzené množstvo vstupno/výstupných dát.

SAMOHODNOTIACE OTÁZKY

- Vymenujte aspoň tri zariadenia, ktoré sa používajú v systémoch MMI pre priamy vstup/výstup.
- Čo je nevýhodou obrazovkových terminálov?
- Aké sú základné princípy použitia obrazovkových terminálov?
- Aký bol princíp činnosti „tichej klávesnice a prečo bol nevhodný?
- Aké sú základné princípy návrhu klávesnice?
- Kedy môžeme použiť ako vstupno/výstupné zariadenie „printing“ terminál a aké má výhody alebo nevýhody?
- Aká je nevýhoda použitia riadiacich panelov?

4.3 Zariadenia s priamym riadením povrchu obrazovkového terminálu

Medzi najčastejšie používané zariadenia s priamym riadením povrchu obrazovkového terminálu patria:

- **svetelné perá**
- **dotykové terminály**

Týmto zariadeniam sa v technickej praxi hovorí aj **priamo ukazujúce zariadenia**.

Priamo ukazujúce zariadenia (svetelné perá, dotykové obrazovky) - nenašli veľký záujem medzi používateľmi.

Nevýhody:

- neustále dvíhanie ramena z pracovného stola je veľmi únavné
- presnosť dotyku je nízka
- ukázanie prstom alebo svetelným perom na obrazovku zaberie viac času ako napríklad použitie myši.

Priamo ukazujúce zariadenia sú zaujímavé len v tom prípade, ak nie je možné používať klávesnicu a MMI dialóg je organizovaný formou menu s malým počtom základných výberov.

SAMOHODNOTIACE OTÁZKY

- Ktoré zariadenia patria do skupiny priamo ukazujúcich zariadení?
- Prečo priamo ukazujúce zariadenia nenašli široké použitie v systémoch MMI a kde ich môžeme používať?
- Stretli ste sa pri práci s takými typmi zariadení? Ak áno, skúste z vlastných skúseností popísať výhody a nevýhody ich použitia.

4.4 Zariadenia s priamym riadením kurzora na obrazovke

Medzi najčastejšie používané zariadenia s priamym riadením kurzora na obrazovke patria:

- **myš**
- **trackball**
- **joystick**

Týmto zariadeniam sa v technickej praxi hovorí aj **nepriamo ukazujúce zariadenia**.

U **nepriamo ukazujúcich zariadení** ako je **myš**, **trackball** a **joystick** je pozícia ukazovateľa kurzora na obrazovke terminálu priamo riadená ručnou manipuláciou zariadenia. Presnosť pohybu je oveľa vyššia ako u dotykových obrazoviek a svetelných pier. Ak má byť vybraná jedna z niekoľkých položiek, aktuálny výber je vždy zvýraznený, čo potvrdzuje správnosť operácie.

Myš

Myš je perfektným zariadením, ktoré uľahčuje prácu s používateľským rozhraním. Má však aj niekoľko nevýhod, ktoré vyplývajú z ergonomických princípov.

- Mnoho ľudí má problém s presným pohybom myši
- **Dvojklik** myši je najväčším problémom, pretože si vyžaduje dva rýchle kliknutia za sebou bez pohybu myši v ktoromkoľvek smere. Mnoho výrobcov už rozmýšľa o pridaní ďalšieho tlačidla, ktoré by dvojklik nahradilo

Bolo už vyrobených mnoho iných alternatívnych zariadení: **joysticky**, **trackbally**, **perá**, **dotykové obrazovky** a pod., ale lepšie zariadenie ako **myš** zatiaľ nebolo nájdené.

Veľa systémov obsahuje pre najpoužívanejšie príkazy **klávesové skratky**, aby uprednostnili používanie klávesnice pred používaním myši, pretože vo všeobecnosti je používanie klávesnice rýchlejšie ako používanie myši, ak si vieme zapamätať všetky klávesové skratky.

SAMOHODNOTIACE OTÁZKY

- Ktoré zariadenia patria do skupiny priamo ukazujúcich zariadení?
- Porovnajte priamo a nepriamo ukazujúce zariadenia z hľadiska presnosti zadávanej informácie
- Stretli ste sa pri práci s takými typmi zariadení? Ak áno, skúste z vlastných skúseností popísať výhody a nevýhody ich použitia.

4.5 Ďalšie zariadenia MMI

Medzi ďalšie dôležité vstupno/výstupné zariadenia, ktoré v poslednej dobe nachádzajú stále širšie využitie v systémoch **MMI** sú:

- **systémy rozpoznávania reči**
- **generátory reči**
- **optické, akustické alarmy**

V počítačovom priemysle bolo venované veľké úsilie vývoju **systémov na rozpoznávanie reči.** Niekoľkoročné výsledky výskumu však dodnes nie sú dostačujúce. Dnešné systémy dokážu len rozpoznávať jednotlivé slová podľa predchádzajúceho vzoru. Preto, ak je to isté slovo vyslovené iný človekom, nemusí byť správne rozpoznané. Z toho dôvodu je používanie týchto zariadení zatiaľ veľmi obmedzené.

Generátor reči je technicky jednoduchší ako prístroj na rozpoznávanie reči. Na trhu sú dostupné rôzne typy generátorov reči. Ich **nevýhodou** je to, že správy, ktoré generujú, prichádzajú často náhodne, keď ich používateľ neočakáva alebo nedáva pozor. Výnimkou je telefónny automat, kde volajúci zámerne dáva pozor na správy prichádzajúce z telefónneho automatu.

Špeciálne **alarmové situácie** môžu byť oznamované za pomoci **vizuálnych** a **akustických** zariadení riadených počítačom. Používajú sa na pritiahnutie **pozornosti operátora**, ktorý sa venuje iným úlohám. V prípade akustických zariadení by mali byť nastaviteľné niektoré parametre ako je napríklad frekvencia, hlasitosť a pod. Dôležité je, aby zariadenia mali rýchly a praktický príkaz na **RESET**.

Výhodou vizuálnej prezentácie oproti akustickej je veľké množstvo prenášaných informácií. Vizuálna prezentácia vyžaduje však stálu aktívnu účasť ľudského subjektu.

Akustické správy:

- nevyžadujú stálu pozornosť používateľa
- môžu prichádzať v ľubovoľnom čase
- sú prijaté okamžite, nezávisle od toho, čo používateľ v danom okamihu robí.

Hlasové správy si však vyžadujú zvýšenú pozornosť, aby sa správne interpretovali. Akustické signály sa nesmú vyskytovať často, pretože by pôsobili veľmi rušivo a používateľa by mohli ľahko iritovať.

SAMOHODNOTIACE OTÁZKY

- Akým spôsobom môžu byť oznamované alarmové situácie v systémoch MMI?
- Porovnajte výhody a nevýhody vizuálnej a akustickej prezentácie informácií.

5 SOFTVÉROVÉ PROSTRIEDKY POŽÍVATEĽSKÝCH ROZHRANÍ

Táto kapitola sa zaoberá popisom dôležitých softvérových prvkov, ktoré by mali byť súčasťou vizualizačnej aplikácie, ich vlastnosťami, správnosťou výberu a možnosťami použitia.

5.1 Úvod a ciele

Programové prostriedky sú neoddeliteľnou súčasťou systémov MMI. Pri tvorbe vizualizačných aplikácií musíme dodržať základné princípy návrhu používateľského rozhrania so zreteľom na možnosti interakcie medzi človekom a počítačom, pričom nesmieme zabudnúť na správne a primerané zabudovanie komponentov, ktoré musia byť súčasťou vytvorenej programátorskej aplikácie.

CIELE

Cieľom tejto kapitoly je naučiť sa:

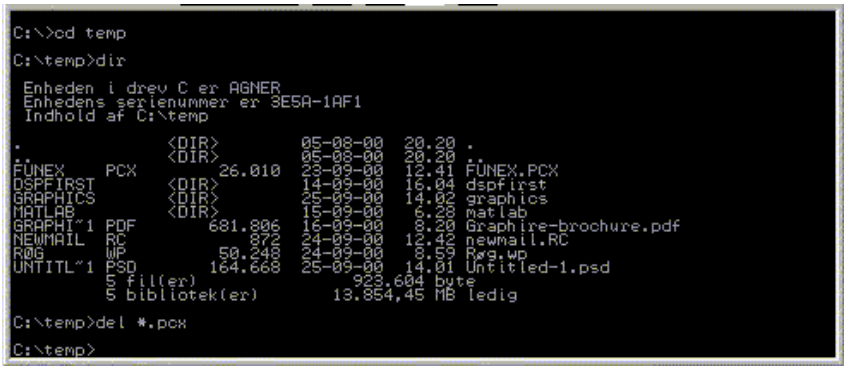
- aké možnosti komunikácie medzi človekom a počítačom máme k dispozícii, aké sú výhody a nevýhody ich použitia.
- poznať, ktoré druhy HELPu môžeme používať v systémoch.
- ako chrániť vytvorený programový produkt pred neautorizovaným kopírovaním.

5.2 Spôsoby komunikácie

Vo všeobecnosti existuje niekoľko spôsobov komunikácie medzi človekom a počítačom. Nazývame ich **interakcia**.

Medzi najznámejšie patria:

Príkazový riadok



Popis

Je to komunikácia založená výlučne na texte, typická pre terminály. Počítač zobrazí znak „**prompt**“, čo znamená, že očakáva zadanie príkazu. Používateľ napíše príkaz a stlačí **Enter**. Počítač vykoná príkaz a na nasledujúcom riadku zobrazí správu o jej vykonaní.

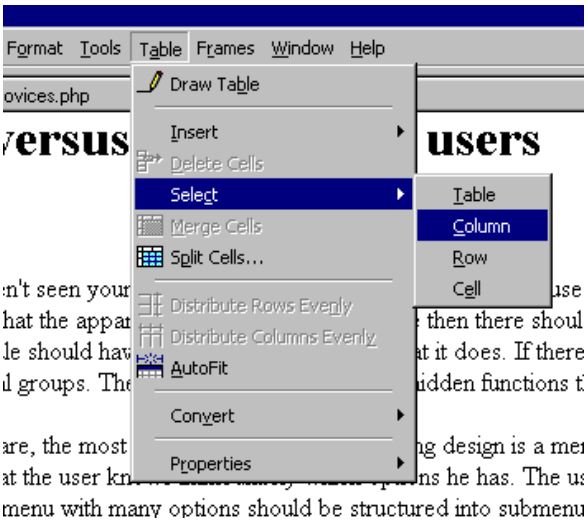
Príklady použitia

Príkazový riadok je dobre známy z DOS, UNIX a Matlab.

Výhody
Ľahko realizovateľný rýchly spôsob komunikácie. Ak používateľ používa často rovnakú skupinu príkazov, môže si ich uložiť ako jednu postupnosť príkazov, ktorá sa potom spúšťa ako jeden príkaz (DOS - batch file, UNIX - shell script, Matlab - M-file).

Nevýhody
Zložité pre začiatočníkov, potrebujeme dlhý čas, aby sme sa naučili a zapamätali príkazy. Nevidíme, aké máme možnosti. Potrebujeme tlačený manuál.

Menu

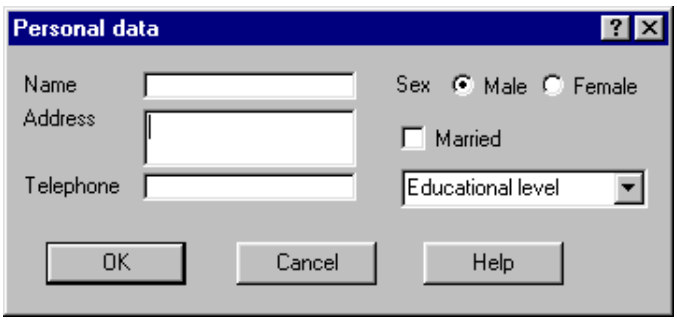


Popis
Prístupné príkazy sú zobrazené na obrazovke alebo displeji. Používateľ vyberá príslušný príkaz kliknutím pomocou myši alebo zadaním klávesovej skratky. Ak existuje viac možností, potom sú štruktúrované tak, že každá položka hlavného menu otvára podmenu s ďalšími možnosťami.

Výhody
Veľmi vhodné pre začiatočníkov, používatelia okamžite vidia, ktorú možnosť si vybrali.

Nevýhody
Skúsení používatelia, ktorí často opakujú tie isté príkazy považujú používanie menu a príslušných podmenu za únavné a stratu času.

Formuláre



Popis

Formuláre obsahujú polia, ktoré používatelia vyplňujú informáciami. Ak používateľ zadal všetky potrebné informácie, potom klikne na tlačidlo **OK** alebo tlačí kláves **Enter**.
Formuláre môžu obsahovať rôzne ovládacie prvky: polia so zoznamom, prepínače, zaškrtávacie políčka, posuvníky, číselníky a pod.

Výhody
Vhodné pre začiatočníkov aj skúsených používateľov. Rozhranie je **samovysvetľujúce**. Užívateľ hneď vidí, ktoré informácie sa od neho požadujú, môže zrušiť zadanú operáciu, vypĺňať polia v ľubovoľnom poradí, polia môžu obsahovať preddefinované hodnoty.

Nevýhody
Problém môže robiť výber preddefinovaných hodnôt.

Pri prechode medzi jednotlivými poliami by sa mala používať myš, aj keď sa jednotlivé polia vypĺňajú pomocou klávesnice. Nie sú známe klávesy pre pohyb vo formulári.

Grafické rozhranie

Popis
Objekty sú zobrazené na obrazovke a používateľ môže s nimi narábať priamo pomocou myši, joystiku a pod.

Príklady použitia
V OS Windows môžeme narábať s ikonkami umiestnenými na ploche, môžeme meniť polohu a veľkosť okien. Programy pre kreslenie, video hry.

Výhody
Intuitívne zrozumiteľné pre všetkých používateľov. Vhodné pre manipuláciu s dátami, ktoré môžu byť reprezentované graficky.

Nevýhody
Presné narábanie s myšou je pre mnohých používateľov problém. Používanie klávesnice namiesto myši nie je možné. Práca je intuitívne zrozumiteľná, ale pre začiatočníkov nie je jasné, ktorý objekt môže byť presunutý, zväčšený, zmenšený a pod.

Komunikácia pomocou reči

Popis
Systém má na vstupe zariadenie pre rozpoznávanie reči a na výstupe zariadenie pre syntézu reči. Systém pracuje podobne ako príkazový riadok. Používateľ povie príkaz a počítač potvrdí, že ho prijal a vykonal.

Výhody
Používateľ nie je pripútaný ku klávesnici a obrazovke, ale môže sa prechádzať a používať ruky a oči na iné úlohy. Vhodné pre riadenie strojov v zlých pracovných podmienkach.

Nevýhody
Ťažko realizovateľné, náchylné ku chybám, pomalé, nepoužiteľné v hlučnom prostredí. Používateľ sa musí naučiť slovník a syntax príkazov, ktorým počítač rozumie.

SAMOHODNOTIACE OTÁZKY

- Vymenujte možnosti komunikácie medzi človekom a počítačom.
- Uveďte krátku charakteristiku jednotlivých možností interakcie medzi človekom a počítačom.
- Ktoré spôsoby komunikácie sú nevhodné pre začiatočníkov a ktoré naopak pre pokročilých?
- Ktoré spôsoby komunikácie sú vhodné aj pre začiatočníkov aj pre pokročilých?
- Ktorý spôsob komunikácie by ste si vybrali pri tvorbe používateľského rozhrania?

5.3 Možnosti HELPU

Softvér môže obsahovať niekoľko druhov **HELPu**:

- **Tlačená príručka**
- **Demo verzia**
- **Všeobecný HELP**
- **Kontextový HELP**

Tlačená príručka
Tlačené príručky sú drahé a používajú sa zriedka. Mnoho používateľov dáva prednosť online **HELP**u zabudovanom v systéme, na to aby zistili, ako treba postupovať alebo na čo sa to používa. Pre nováčikov, ktorí nevedia narábať s online **HELP**om môže byť užitočná krátka príručka.

Demo verzia
Demo verzia môže byť veľmi užitočná pre začiatočníkov. Skúsenejší používatelia ju využívajú zriedka.

Všeobecný HELP
Je veľmi rozsiahly dokument, obsahujúci mnoho strán, ktorý zahŕňa všetky možnosti programu. Mal by obsahovať štruktúrovaný index a tiež vyhľadávacie možnosti, pre ľahší prístup k jednotlivým témam. Moderné systémy majú často dosť zložité používateľské rozhranie a používajú termíny, ktorým používateľ nerozumie. Ak sa používateľ pokúsi

niekoľkokrát nájsť odpoveď na hľadaný problém bez úspechu, tak už sa nikdy nepokúsi používať systém **HELP**u.

Kontextový HELP

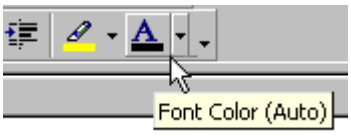
Je to možnosť, pomocou ktorej si môžeme zobraziť informácie týkajúce sa vybranej časti programu, ktorú používateľ používa.

Kontextový HELP môže byť realizovaný nesledujúcimi spôsobmi:

Tlačidlom

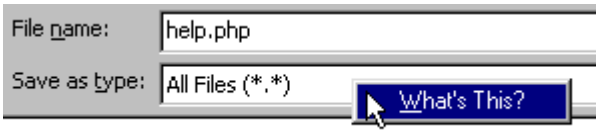
Všetky dialógové okná by mali obsahovať tlačidlo HELP, ktoré podáva informácie o účele otvoreného dialógového okna a význame jednotlivých polí.

Polohou myši



Ak sa postavíme kurzorom myši na objekt, ikonu alebo vybranú položku v menu a nemeníme polohu viac ako sekundu, potom sa zobrazí pri kurzore myši bublinová nápoveda, ktorá nás môže informovať na čo slúži ikona, ale neobsahuje detailné informácie.

Stlačením pravého tlačidla myši



Stlačením pravého tlačidla myši sa zobrazí miestna ponuka, ktorá obsahuje všetky príkazy bezprostredne sa týkajúce daného objektu.

Ovládacím tlačidlom HELP

Kliknutím myši na ovládacie tlačidlo s otáznikom v pravom rohu otvoreného okna, sa pri kurzore myši zobrazí otáznik. Ak takýmto kurzorom myši klikneme na vybraný objekt, tak sa zobrazí „**Čo je to**“ HELP pre daný objekt.

Tlačidlo je jediná zo štyroch možnosti kontextového **HELP**u, ktorá je pre používateľov dostatočne zrozumiteľná a somovysvetľujúca. Zobrazenie **HELP**u polohou myši môže byť rušivé, ak sa aplikuje na všetky objekty v programe. Pravé tlačidlo myši používajú iba používatelia, ktorí vedia na čo slúži. Ovládacie tlačidlo **HELP**u používajú iba skúsení používatelia, ktorí chcú mať informácie o všetkých zobrazených poliach v dialógových oknách.

Záver

Moderné systémy **HELP**u sú často veľmi zložité, pretože obsahujú viac problémov ako ich riešení. Je veľmi dôležité venovať špeciálnu pozornosť **HELP** systémom pri vykonávaní testu použiteľnosti.

SAMOHODNOTIACE OTÁZKY

- Vymenujte druhy HELPu, ktoré sa môžu používať v programoch.
- Uveďte krátku charakteristiku jednotlivých druhov HELPu.
- Ktorými spôsobmi môže byť realizovaný kontextový HELP?
- Vyskúšajte si všetky možnosti kontextového HELPu. Ktorá z nich vám najviac vyhovuje?

AKTIVITA

- Zistite, ktoré druhy HELPu obsahuje generátor vizualizačných aplikácií Control Web 2000.

5.4 Ochrana pred kopírovaním

Výrobcovia komerčného softvéru majú pochopiteľnú požiadavku ochrany ich produktov pred **neautorizovaným kopírovaním.** V tejto oblasti bolo vyvinutých veľa rôznych metód, ale u všetkých sa stretávame s **problémom použiteľnosti**:

Originálny disk

Disketa, ktorá podporuje časť softvérového produktu, je porušená takým spôsobom, že ju nemôžeme ľahko prekopírovať, napríklad je odstránená časť magnetickej vrstvy.

Problémy:

- Používateľ musí vždy zasunúť disketu do disketovej mechaniky, ak chce s produktom pracovať
- Používateľ si nemôže urobiť legitímnu záložnú kópiu
- Používate nemôže pracovať v sieti

Ukladanie ochranných informácií do počítača

Problémy:

- Tieto informácie sa stratia, ak používateľ zmení operačný systém
- Používateľ môže mať viac počítačov
- Nekompatibilita s iným operačným systémom

Ukladanie hardvérovej konfigurácie do počítača

Problémy:

- Konfigurácia sa stratí, ak používateľ zmení operačný systém

- Používateľ môže mať viac počítačov
- Nekompatibilita s iným operačným systémom

Hardvérový kľúč

Je to malý kúsok hardvéru, ktorý musí byť počas používania programu pripojený k počítaču.

Problémy:

- Vyžaduje priamy prístup k hardvéru alebo špeciálny driver, môže byť nekompatibilný s iným hardvérom alebo operačným systémom
- Ak má používateľ viac počítač, tak sa musí vždy vybrať a presunúť do iného počítača
- Môže sa stratiť alebo poškodiť
- Nie je chránený pred zlodejmi
- Počet hardvérových kľúčov, ktoré môžu byť pripojené k počítaču je obmedzený
- Používate nemôže pracovať v sieti

Tlačená príručka

Manuál sa ťažšie kopíruje ako softvér.

Problémy:

- Používateľ dáva prednosť online HELPu.

Registrácia

Problémy:

- Používateľ má strach pred zneužitím osobných údajov

Sériové číslo

Používateľ musí zadať sériové číslo, ak chce inštalovať softvér

Horúca linka (HOT)LINE

Len registrovaní používatelia môžu využívať HELP cez **„horúcu linku“.**

Mnoho ochranných postupov obsahuje kombinácie uvedených metód. V poslednej dobe je najpoužívanejším spôsobom ochrany sériové číslo.

SAMOHODNOTIACE OTÁZKY

- Vymenujte možnosti ochrany softvérových produktov, ktoré môžeme použiť pred neautorizovaným kopírovaním.
- Z ktorých z týchto možností ste sa už stretli?
- Ktorá z týchto možnosti je podľa vás najúčinnejšia?
- Stretli ste sa aj s inými možnosťami ochrany softvéru? Ak áno, popíšte tento spôsob.

6 VIZUALIZÁCIA

Táto kapitola sa zaoberá objasnením pojmu vizualizácie, jej definíciou podmienenou oblasťami použitia, cieľmi a základnými funkciami vizualizačného systému a popisom štruktúry technologického systému s vizualizáciou, ktorá tvorí rozhranie medzi technologickou a informačnou úrovňou riadiaceho systému.

6.1 Úvod a ciele

Technické a programové vybavenie tvoriace rozhranie **stroj - človek** často reprezentuje 50 až 75 % nákladov na riadiaci systém ako celok. Napriek tomu sa tieto investície môžu mnohonásobne vrátiť spoľahlivosťou a bezpečnosťou správne navrhnutých systémov riadenia.
Vizuálna reprezentácia je najväčšou časťou používateľského rozhrania medzi procesom, automatickým riadiacim systémom a človekom. Aj z tohto dôvodu sú pojmy **vizualizačné systémy** a **systémy SCADA/HMI** (Supervisory Control and Data Acquisition/Human Machine Interface - supervízorové riadenie, zber údajov/ rozhranie človek - stroj) v súčasnosti považované za synonymá.
V prednáške budú stručne uvedené funkcie a charakteristické črty vizualizačných systémov, podrobnejšie sa budem venovať najnovším trendom vo vývoji týchto systémov, využitiu Internetu/Intranetu v priemyselných aplikáciách a snahám o normalizáciu v oblasti SCADA/HMI aplikácií.

CIELE

Cieľom tejto kapitoly je naučiť sa:

- vysvetliť pojem vizualizácia v najvšeobecnejšej rovine,
- definovať pojem vizualizácia pre oblasť priemyselných systémov,
- aké sú ciele vizualizácie a stručne ich vysvetliť,
- aké sú funkcie vizualizačného systému,
- aká je štruktúra technologického procesu s vizualizáciou a rozoznávať jej úrovne.

6.2 Vizualizácia a jej definície

V systémoch **MMI** sa najčastejšie stretávame s potrebou **vizualizácie** riadeného procesu.
Na abstraktnej úrovni tento pojem znamená **ZVIDITEĽNOVANIE**. Proces vizualizácie je zameraný na ľudský objekt s cieľom vyvolať v ňom vizuálny vnem.
V najvšeobecnejšej rovine sa jedná o vizualizáciu požadovaných objektov a jej definícia je nasledovná:
VIZUALIZÁCIA je použitie teoretických, technických, programových a komunikačných prostriedkov pre zviditeľňovanie definovaných (abstraktných alebo reálnych) objektov.

Presnejšia definícia vizualizácie je podmienená **oblasťou použitia**, t.j. definíciou objektov a použitými prostriedkami.

Napríklad pre oblasť priemyselnej automatizácie je definícia vizualizácie nasledovná:
VIZUALIZÁCIA je použitie teoretických, technických, programových a komunikačných prostriedkov pre zviditeľňovanie definovaných (abstraktných alebo reálnych) objektov na ľubovoľnej úrovni informačného a riadiaceho systému s cieľom podpory rozhodovania.
Pri vizualizácii sa nejedná len o grafické zobrazenie objektov, ale o všetky činnosti týkajúce sa definovania, získania a spracovania objektov, ktorých grafická stránka je **používateľským rozhraním** medzi priemyselným procesom a človekom.
Pod vizualizáciou priemyselných procesov rozumieme rozhranie medzi **technologickou** časťou a **informačnou** úrovňou riadiaceho systému.

SAMOHODNOTIACE OTÁZKY

- Čo znamená pojem vizualizácia na abstraktnej úrovni?
- Definujte pojem vizualizácia v najvšeobecnejšej rovine.
- Definujte pojem vizualizácia pre oblasť priemyselnej automatizácie.
- Čo rozumieme pod pojmom vizualizácia priemyselných procesov?

6.3 Ciele a funkcie vizualizácie

Proces vizualizácie má nasledujúce **ciele**:

- **MONITOROVANIE** - jeho cieľom je zber a zaznamenávanie údajov či stavov procesu
- **DOHLIADANIE** - sledovanie činnosti systému alebo jeho časti za účelom overenia jeho správnej funkcie. Vykonáva sa sledovaním jednej alebo viacerých premenných systému a porovnaním nameraných hodnôt so stanovenými limitami.
- **OVLÁDANIE, SUPERVÍZNE RIADENIE** - ide o riadiacu a monitorovaciu činnosť systému, a tiež činnosť, ktorá zabezpečuje jeho spoľahlivosť a ochranu.
- **DIAGNOSTIKA PROCESU** - predvídanie chybových stavov na základe informácií o priebehu procesných veličín získaných z databáz.

Ako už bolo spomenuté rozhranie medzi **technologickou** a **informačnou** úrovňou riadiaceho systému označujeme pojmom **vizualizácia**. Z toho hľadiska sú základné **funkcie vizualizačného systému** sú nasledujúce

- prenášať povely od používateľa do technologického procesu
- prenášať a vo vhodnej forme zobraziť používateľovi údaje o okamžitom stave technologického procesu, t.j. o hodnote jeho stavových veličín, parametrov, prípadne okamžitej štruktúre
- zbierať a uchovávať informácie o stavoch technologického procesu, o množstve a parametroch hotových produktov vo forme rôznych databáz
- zobrazovať alebo uchovávať informácie o poruchách systému (alarmy)
- zobrazovať vývoj určitých veličín za uplynulé obdobie (trendy)

SAMOHODNOTIACE OTÁZKY

- Vymenujte ciele vizualizačného systému a uveďte ich krátku charakteristiku.
- Vymenujte funkcie vizualizačného systému s vizualizáciou.

6.4 Štruktúra technologického procesu s vizualizáciou

Technologický proces s vizualizáciou sa skladá z **3 podsystémov** (obr.16):

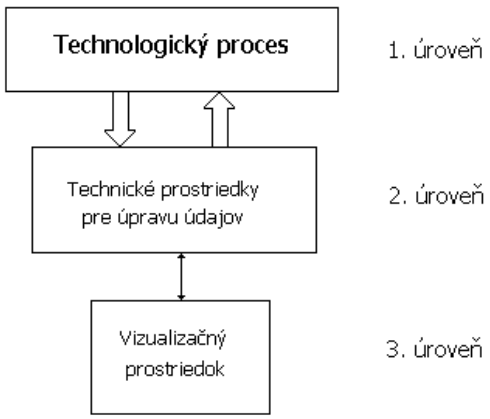
- **1.podsystém** tvorí samotný technologický proces
- **2.podsystém** tvoria technické prostriedky pre úpravu údajov. Patria sem karty pre PC, programovateľné automaty, inteligentné snímacie moduly,...
- **3.podsystém** predstavuje vlastný vizualizačný prostriedok, ktorým môže byť technologický panel, technologické PC, operátorský terminál,...

Z hľadiska vizualizácie nás zaujíma predovšetkým **3.podsystém** a jeho prepojenie na **2.podsystém**. Prepojenie týchto podsystémov vykonáva najčastejšie tzv. **driver** (ovládač), ktorý môže byť pre používateľa:

- **zakrytý** - má definované iba formálne parametre a ich činnosť
- **otvorený** - používateľ má definované rozhranie zo strany vizualizačného programu a môže si činnosť ovládača prispôsobiť programovaním v niektorom programovacom jazyku.

Najčastejšie sa jednotlivé signály z technológie zobrazujú na obrazovke vo forme tzv. **objektov**. Priradenie signálov jednotlivým objektom sa definuje väčšinou vo forme tzv. **databázy objektov** pre danú aplikáciu, kde sa objektom priradia symbolické mená, adresy, komentáre, použité kanály ovládača a pod. Pokiaľ vizualizačný prostriedok komunikuje s viacerými uzlami technológie, sú tieto prepojené do informačnej siete, napr. cez zbernicu DH 485.

Dnes už existuje množstvo rôznych vizualizačných systémov rôznej kvality, rôzneho stupňa zložitosti a od rôznych výrobcov.



Obr. 16 Štruktúra technologického procesu s vizualizáciou

SAMOHODNOTIACE OTÁZKY

- Z akých podsystémov sa skladá technologický proces s vizualizáciou? Uveďte stručnú charakteristiku každého podsystému.
- Na čo slúžia ovládače vo vizualizačnom systéme a aké typy ovládačov poznáme z hľadiska používateľa?
- Akým spôsobom sa definuje priradenie signálov jednotlivým objektom vo vizualizačnom systéme?

7 VÝMENA INFORMÁCIÍ V SYSTÉMOCH MMI

Táto kapitola sa zaoberá problémami a všeobecnými princípmi výmeny informácií v systémoch MMI, pretože ich množstvo a kvalita ovplyvňujú správne rozhodnutia ľudí-operátorov pri výskyte nepredvídaných udalostí, čo vedie k redukcii vzniku škôd a kvalite, bezpečnosti a spoľahlivosti celého systému.

7.1 Úvod a ciele

Rozhranie **človek – stroj**, označované aj ako **MMI** interfejs, je imaginárna úroveň, cez ktorú sa vymieňajú informácie medzi operátorom a strojom či zariadením technologického procesu. Ak je výmena informácií medzi používateľom a riadeným systémom jednoduchá a efektívna, čo znamená, že je správne navrhnuté rozhranie, nielenže to spríjemňuje prácu, ale vedie to aj k redukcii chýb a obmedzeniu vzniku škôd.

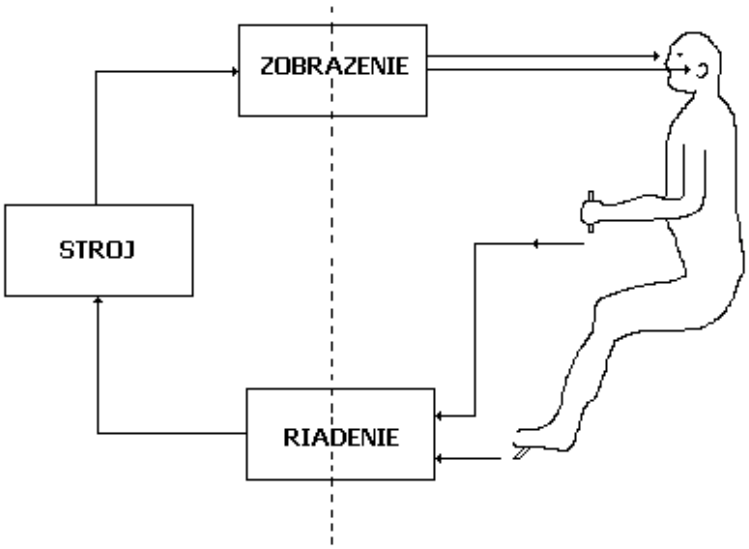
CIELE

Cieľom tejto kapitoly je naučiť sa:

- ako ovplyvňujú prenos informácií v systémoch MMI zobrazovacie a riadiace prvky,
- aký problém sa rieši pri návrhu používateľského rozhrania z hľadiska výmeny informácií,
- akým spôsobom zohľadňovať kritérium jednoduchosti a kritérium všestrannosti pri návrhu používateľského rozhrania,
- ako merať a vyhodnocovať kvalitu návrhu používateľského rozhrania,
- aké sú tri hlavné aspekty tvorby používateľského rozhrania,
- špecifikovať všeobecné princípy návrhu používateľského rozhrania z informačného hľadiska.

7.2 Problémy výmeny informácií v systémoch MMI

V systémoch MMI sa informácie dopravujú smerom od stroja ku človeku prostredníctvom **zobrazovacích prvkov** interfejsu a naopak od človeka k stroju cez **riadiace prvky** interfejsu (obr.17).
Pri návrhu interfejsu **z hľadiska výmeny informácií** vzniká vážny problém, pretože sa tu stretávame s dvoma úplne odlišnými základnými vlastnosťami rozhrania človek – stroj. Stroj je rýchly, presný, výkonný, no neprispôsobivý. Na druhej strane je tu človek, ktorý je pomalý, náchylný ku chybám, z hľadiska výkonu pomerne slabý, ale jeho činnosť môže byť veľmi prispôsobivá a mnohostranná. Odlišnosť týchto vlastností vysvetľuje, prečo **kombinácia človek – stroj** je taká užitočná, ale len v tom prípade, ak tieto od základov odlišné vlastnosti sú úspešne pospájané do jedného celku.



Obr.17 Výmena informácií v systémoch MMI

V začiatkoch technologickej revolúcie nebol tento problém taký významný, pretože vzhľadom na obmedzenia strojov a rýchlosť ich činnosti sa problémy návrhu sústredili predovšetkým na hardvérovú časť a schopnosti človeka sa využívali na kompenzáciu nedostatkov, ktoré vznikli pri návrhu interfejsu. Avšak s príchodom výkonnejších strojov a zariadení sa zistilo, že najviac nedostatkov v činnosti celej technológie vyniká práve pri návrhu tohto rozhrania. V priebehu vývoja návrhu interfejsu došlo tiež **k výmene úloh** medzi človekom a strojom. V čase, keď stroje a zariadenia predstavovali niečo nové a výnimočné, ľudia boli školení na to, aby sa im prispôsobovali. V súčasnosti sa ale úlohy **vymenili** a očakáva sa, že stroje sa budú prispôsobovať človeku s cieľom zlepšiť návrh interfejsu.

Tento problém sa stáva čoraz naliehavejším aj vďaka stále abstraktnejšej podobe výmeny informácií v systémoch MMI. Napríklad návrh **intuitívneho interfejsu** je opodstatnený pri jednoduchých technologických postupoch, akým môže byť oranie zeme pluhom, kde za jedinú vizuálnu kontrolu môžeme považovať pohyb pluhu a riadiaci zásah ručnú manipuláciu s ním. Iným extrémom sú riadiace centrá, kde je na displejoch zobrazované množstvo abstraktných parametrov a ich kombinácií, ktoré vplývajú na široký rozsah možných riadiacich zásahov.

Čím abstraktnejšou sa stáva situácia, tým dôležitejšie je návrh interfejsu prispôsobiť jeho základným princípom. Pri návrhu **zobrazovacích a riadiacich prvkov** sa berú do úvahy **populačné stereotypy**, ale pri návrhu interfejsu musíme ešte zohľadniť aj určité prvky stereotypu v ich vzájomnom vzťahu založenom na **prirodzených princípoch** a **situáciách**. Ak krútime riadiacim prvkom doprava, očakávame, že jeho pohyb bude zobrazený na displeji v smere pohybu hodinových ručičiek. To platí ale len v tom prípade, ak medzi pohybom riadiaceho prvku a jeho zobrazením neexistuje žiadna mechanická väzba. V opačnom prípade nie je tento vzájomný vzťah jednoznačný, pretože existuje viacero spôsobov jeho zobrazenia. Je jasné, že neočakávaný vzťah medzi riadením a jeho zobrazením môže spôsobiť v systémoch dlhšie časové odozvy a častejší výskyt chýb. Príkladom môže byť výmena plynového a brzdového pedálu v aute. Samozrejme, že je možné takéto auto šoférovať, bude si to však vyžadovať zvýšenú pozornosť a opatrnosť, čoho výsledkom bude pomalá jazda.

SAMOHODNOTIACE OTÁZKY

- Prečo je výmena informácií v systémoch MMI pomerne obtiažná?
- Ktoré prvky slúžia na sprostredkovanie prenosu informácií v systémoch MMI?
- Aký vážny problém vzniká pri návrhu používateľského rozhrania z hľadiska výmeny informácií?
- Popíšte rozdelenie úloh medzi človekom a strojom v procese vývoja návrhu používateľského rozhrania.
- V ktorých prípadoch je opodstatnený návrh intuitívneho interfejsu?
- Ktoré princípy sa berú do úvahy pri návrhu zobrazovacích a riadiacich prvkov?

7.3 Základné aspekty tvorby rozhrania z hľadiska výmeny informácií

Je zrejmé, že ak sa častejšie pri práci operátora vyskytuje tá istá chyba, nie je niečo v poriadku s interfejsom. Buď operátor nemá dostatok potrebných informácií alebo informácie nie sú zobrazované takým spôsobom, ktorý by podporoval a umožňoval jeho správne rozhodovanie. Pri návrhu interfejsu sa samozrejme nepredpokladá, že operátor je osoba, ktorá nie je schopná samostatne uvažovať, je **úplne neskúsená a nezodpovedná**, a preto nie je dôvod na to, že by mal byť interfejs **plne prispôsobivý**.

Na druhej strane, nemôžeme ani predpokladať, že skúsený a inteligentný operátor bude schopný kompenzovať všetky nedostatky, ktoré sme pri návrhu interfejsu zanedbali. Ak je stroj určený pre všeobecné použitie, potom musí byť **spoľahlivý**, ale ak je navrhnutý pre špecializované účely a pre použitie skúseným operátorom, potom sa ciele návrhu interfejsu stávajú viac komplikovanými a diskutabilnými. Problém je v tom, že **kritérium jednoduchosti** a **kritérium všestrannosti** sú často **nezlúčiteľné**. Napríklad, zlúčenie všetkých premenných z technologického procesu do jedného zobrazovacieho prvku a poskytnutie len jedného riadiaceho prvku operátorovi je redukovanie úlohy do jej najjednoduchšej podoby, čo ale obmedzuje operátora v jeho činnosti a nedáva mu dostatočnú voľnosť v rozhodovaní a riadiacej činnosti. Ak však bude mať k dispozícii displej zobrazujúci všetky premenné z procesu a riadiace prvky, ktoré ovplyvňujú všetky tieto premenné, bude mať väčšiu voľnosť v rozhodovaní a tiež sa zvýši aj jeho motivácia a záujem o správnu činnosť konkrétneho zariadenia. Dokonca môže ovplyvniť a dosiahnuť vyššiu pružnosť v zladení činností jednotlivých prvkov procesu.

Avšak schopnosť navrhnúť a vytvoriť rozhranie, ktoré umožní operátorovi slobodne sa rozhodnúť pre správny, nie chybný zásah, vo veľkej miere závisí od nadania a skúseností človeka – **tvorcu rozhrania**. Ak zoberieme do úvahy hlavné aspekty ako sú **ergonómia, dynamika** a **funkčnosť**, potom **neexistuje optimálne poradie**, v ktorom majú byť tieto uvažované pri návrhu každého interfejsu. Uprednostnenie jedného hľadiska môže obmedzovať možnosti ďalších, preto je najlepšie pri návrhu rozhrania začať s tým hľadiskom, ktoré zohľadňuje a rieši najpravdepodobnejšie problémy **spracovania informácií operátorom** – to je **schopnosť** jeho **vnímania**, **uvedomenia**, **časovej odozvy** a **akčného**

zásahu. Návrh rozhrania by preto mal vždy zohľadňovať všetky tieto aspekty v ľubovoľnom poradí tak, aby bola dosiahnutá maximálna funkčnosť daného zariadenia.

SAMOHODNOTIACE OTÁZKY

- Vysvetlite ako sa pri návrhu používateľského rozhrania uplatňuje kritérium jednoduchosti a kritérium všestrannosti.
- Od čoho závisí vo veľkej miere správnosť návrhu používateľského rozhrania?
- Pre akých používateľov je spoľahlivosť vhodným cieľom návrhu interfejsu?
- Vymenujte tri hlavné aspekty tvorby používateľského rozhrania.
- Od čoho závisí poradie hlavných aspektov, ktoré berieme do úvahy pri tvorbe interfejsu?

7.4 Hodnotenie kvality používateľského rozhrania

Problémy návrhu interfejsu majú vždy **informačný charakter**. Z tohto hľadiska vyplývajú aj nedostatky v určení a vyhodnotení jeho **kvality**. Meranie práce alebo úsilia vynaloženého pri práci s interfejsom je mimoriadne ťažké, ak nie nemožné.

Fyziologické merania záťaže nie sú vhodné a nie je rovnako možné používať nepriame meranie námahy, ako je napríklad meranie napätia v svaloch či srdcovej arytmie na porovnanie kvality jednotlivých rozhraní. Môžeme konštatovať, že v súčasnosti **neexistuje** uspokojujúci spôsob merania ako ťažko a na akej úrovni pracuje človek, ktorý je súčasťou takéhoto rozhrania.

Pokusy na vyhodnotenie a porovnanie kvality jednotlivých interfejsov sú preto nevhodné tak z praktického ako aj z teoretického hľadiska. Pravdaže je možné robiť rôzne psychologické merania rýchlosti a presnosti zásahov človeka ako súčasti rozhrania, ale nie je možné použiť tieto veličiny na posúdenie úrovne schopnosti učenia a stupňa prispôsobenia týchto rozhraní. Toto je pravdepodobne jeden z hlavných dôvodov, prečo sa vývoj smerom k vytvoreniu všeobecnej teórie návrhu interfejsu rozvíja pomalším tempom. Avšak systematická práca v tejto oblasti prispieva k neustálemu zlepšovaniu kvality návrhu systémov MMI a hoci je vplyv návrhu dobrého interfejsu na motiváciu, vyššiu výkonnosť a pocit uspokojenia z práce ťažko merateľný, je zrejmé, že je potrebné tieto aspekty pri jeho návrhu zohľadňovať.

SAMOHODNOTIACE OTÁZKY

- Aký charakter majú problémy spojené s návrhom interfejsu?
- Existujú v súčasnosti uspokojujúce postupy alebo merania na vyhodnocovanie kvality používateľského rozhrania?
- Poznáte nejaké metódy, ktoré by sa dali použiť na určovanie kvality navrhnutého interfejsu?

7.5 Všeobecné závery pre návrh rozhrania v systémoch MMI

Z uvedeného rozboru problematiky **výmeny informácií** v systémoch MMI vyplývajú nasledujúce všeobecné závery:

1. Výmena informácií v systémoch MMI je pomerne obťiažná vzhľadom na odlišné vlastnosti a obmedzenia vyplývajúce z úloh ľudského faktora a strojov v týchto systémoch. Z tohto dôvodu nedostatky MMI systémov môžu viesť k nevhodnému návrhu interfejsu.
2. Pri návrhu interfejsu je potrebné brať do úvahy **populačné zvyklosti**, ich porušenie môže viesť:

a k dlhšej časovej odozve
b väčším chybám či rozsahom chýb
c dlhšiemu času potrebnému na zaškolenie
d k väčšej vyčerpanosti pri obsluhe

3. Spoľahlivosť je vhodným cieľom návrhu interfejsu len pre neskúsených používateľov.
4. **Kritérium jednoduchosti** návrhu interfejsu je často **v rozpore** s **kritériom všestrannosti**.
5. Existujú tri hlavné aspekty tvorby interfejsu ako je **ergonómia**, **dynamika** a **funkčnosť**, pričom poradie v akom je potrebné sa nimi zaoberať pri tvorbe interfejsu záleží len od schopnosti a postavenia operátora v jednotlivých situáciách.
6. V súčasnosti **neexistujú** všeobecné postupy a merania koľko práce a úsilia potrebuje vynaložiť operátor pri práci s konkrétnym interfejsom, teda neexistuje vhodné kritérium vyhodnotenia ich kvality. Avšak aspekty ako je motivácia, vyššia výkonnosť a celkové uspokojenie z práce s daným interfejsom je potrebné pri jeho návrhu zohľadňovať.

Dodržiavanie **všeobecných záverov** platných pri návrhu interfejsu je kľúčom k zabezpečeniu **funkčnosti** a neustále sa zvyšujúcej **kvality** návrhu systémov MMI, pretože správne navrhnuté rozhranie, nielenže spríjemňuje prácu, ale vedie aj k redukcii chýb a obmedzeniu vzniku škôd.

SAMOHODNOTIACE OTÁZKY

- Vymenujte všeobecné závery platné pre správny návrh interfejsu.
- K čomu môže viesť nedodržiavanie populačných zvyklostí pri návrhu rozhrania?
- Aké ciele sledujeme pri dodržiavaní všeobecných záverov návrhu používateľského rozhrania?

8 STROJOVÝ INTELIGENČNÝ KVOCIENT V SYSTÉMOCH MMI

Nasledujúca kapitola sa zaoberá otázkami kvocientu umelej inteligencie – **MIQ**, ktorý predstavuje mieru inteligencie stroja a spôsobmi jeho určenia, ktoré využívajú užívateľské hľadisko a overené metódy merania mentálneho zaťaženia a množstva prenášaných údajov. V budúcnosti by mal kvocient MIQ slúžiť nielen pre teoretické účely, ale mal by sa stať praktickým indexom vyjadrujúcim ciele návrhu daného zariadenia a jeho kvality.

8.1 Úvod a ciele

Umelá inteligencia je oblasť výskumu pre implementáciu inteligencie v strojoch ako jednej z ľudských vlastností založenej na počítačovej vede, biológii, ekológii, filológii, matematike, atď. Automatizácia s prvkami umelej inteligencie začala v poslednej dobe nahrádzať ľudskú inteligenciu rovnako ako fyzickú ľudskú prácu. Od roku 1956, keď bol prvýkrát definovaný pojem **umelej inteligencie**, bolo publikovaných veľa rôznych definícií **strojovej inteligencie** V mysli mnohých ľudí, osobitne z oblasti riadenia, termín **„inteligentné riadenie“** znamená nejakú formu riadenia s využitím metodológie fuzzy prístupu alebo neurónových sietí. Preto je dôležité zaviesť nový index, ktorý predstavuje stupeň strojovej inteligencie ako požadovaný cieľ pri návrhu inteligentných strojov. Je to dôležité aj z komerčného hľadiska, pretože zavedenie tohto indexu umožní zákazníkovi porovnať jednotlivé výrobky z hľadiska miery inteligencie.

CIELE

Cieľom tejto kapitoly je naučiť sa:

- čo je to kvocient umelej inteligencie,
- čo vyjadruje pojem strojová inteligencia,
- aké podstatné vlastnosti by mali mať inteligentné stroje,
- ktoré dôležité otázky je potrebné riešiť pre zostavenie postupu praktického merania MIQ,
- ako delíme strojovú inteligenciu,,
- na čo slúži model spolupráce človek-stroj,
- na čo slúži graf inteligenčných úloh,
- ako matematicky vyjadriť kvocient MIQ,
- ktoré metódy sa používajú pre meranie mentálneho zaťaženia,
- z akých krokov pozostáva metóda určenia kvocientu MIQ.

8.2 Kvocient strojovej inteligencie – MIQ

Podobne ako pojem **inteligenčný kvocient** (**IQ**) používaný pre vyjadrenie miery ľudskej inteligencie, je **kvocient strojovej inteligencie** (**MIQ**) používaný ako index pre vyjadrenie miery inteligencie riadiaceho systému. Tento pojem sa významne odlišuje od iných známych

ukazovateľov ako je napríklad **výkon**, **spoľahlivosť**, **diagnostika chybových stavov** a pod. V súčasnosti existuje niekoľko **definícií strojovej inteligencie**, z ktorých potom môžeme formulovať výslednú definíciu pre **MIQ**.

Inteligentné stroje by mali mať nasledujúce podstatné vlastnosti:

- **strojová inteligencia** je proces analyzovania, organizovania a premeny dát na strojové vedomosti, ktoré sú definované ako štruktúrované informácie získané a aplikované s cieľom odstrániť nevedomosť a neurčitosť špecifickej úlohy vykonávanej inteligentným strojom
- aby **inteligentný stroj** mohol vykonávať odpovedajúce úlohy, mal by napodobňovať funkcie žijúcich tvorov a v podstate aj ľudské mentálne schopnosti
- **inteligentné riadenie** je disciplína, v ktorej je riadiaci algoritmus vytváraný napodobňovaním určitých vlastností inteligentných biologických systémov
- **inteligentný riadiaci systém** je riadiaci systém s konečným stupňom autonómie z hľadiska samoučenia, rekonfigurovateľnosti, dedukcie, plánovania a rozhodovania
- inteligentné stroje sú stroje navrhnuté na vykonávanie antropomorfných úloh s minimálnou interakciou ľudského faktora

Z uvedeného vyplýva, že **strojová inteligencia** je schopnosť napodobňovať ľudské mentálne schopnosti a vykonávať ich ako človek. Napríklad novú verziu riadiaceho systému nejakej technológie by sme posúdili za viac inteligentnú vtedy, ak by na jeho obsluhu bol potrebný menší počet operátorov ako v predchádzajúcej verzii.

Cieľom zavádzania **inteligentného riadenia** je zvýšiť schopnosť odozvy na nepredvídané udalosti v neznámom prostredí. Ak sa stroj stáva inteligentnejším, tak rastie jeho riadiaci výkon pri výskyte nepredvídaných okolností.

Definícia kvocientu strojovej inteligencie: MIQ je miera autonómie a výkonnosti vzhľadom na výskyt nepredvídaných udalostí.

SAMOHODNOTIACE OTÁZKY

- Vysvetlite pojem kvocient strojovej inteligencie
- Ktoré podstatné vlastnosti by mali mať inteligentné stroje?
- Čo je to strojová inteligencia?
- Čo je cieľom zavádzania inteligentného riadenia?
- Definujte kvocient strojovej inteligencie.

8.3 Problematika merania MIQ

Zostavenie postupu **praktického merania MIQ** si vyžadovalo vyriešenie nasledujúcich otázok:

- Ktoré sú hlavné komponenty strojovej inteligencie?
- Môžeme jasne oddeliť strojovú inteligenciu od vplyvu ľudského faktora?

- Ako môžeme porovnávať a priblížiť MIQ vyjadrené číselnou hodnotou a stupeň strojovej inteligencie, ktorý vnímame ako používatelia?

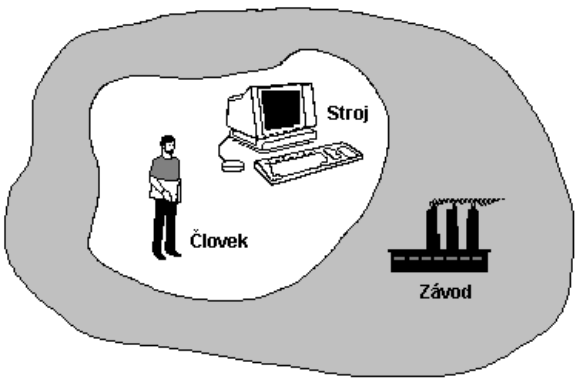
Strojová inteligencia môže byť rozdelená na dva komponenty – **inteligenciu riadenia** a **inteligenciu rozhrania** tak, ako je to zobrazené na obr.18.

Inteligencia riadenia má za úlohu vykonávať riadiace zásahy ako odozvu na výskyt nepredvídaných udalostí v neurčitom prostredí. Nepredvídané udalosti sú výnimočné alebo náhodne sa vyskytujúce stavy ako napríklad poruchy strojov alebo iné nezvyčajné stavy, do ktorých sa môže stroj počas prevádzky dostať. Často je ťažké získať dostatok informácií alebo parametrov o pracovných stavoch strojov vzhľadom na ťažkosti, ktoré vznikajú pri modelovaní ich dynamických charakteristík v neurčitom prostredí.

Ďalším komponentom **strojovej inteligencie** je **inteligencia rozhrania**, ktorá predstavuje stupeň inteligencie vzájomného pôsobenia medzi človekom a počítačom (**Human Computer Interaction – HCI**). V poslednej dobe sa oblasť **HCI** stáva stále viac a viac významnejšou predovšetkým v zložitých riadiacich systémoch ako sú napríklad riadiace centrá nukleárnych elektrární, kabíny pilotov lietadiel, kde riadiaci systém navrhnutý ako „user friendly“ môže redukovať chyby spôsobené ľudským faktorom a efektívne využívať ľudskú vynaliezavosť.

Obr.18 Inteligencia stroja

Odpoveď na druhú otázku je zobrazená na obr.19. Vo väčšine prípadov úplný riadiaci systém pozostáva z človeka vykonávajúceho nadradenú kontrolnú činnosť (supervízor) a stroja. Hranica medzi človekom a strojom nie je jasne definovaná, pretože sa musia navzájom dopĺňať. Väčšinou teda ide o systémy spolupráce medzi človekom a strojom (**Human Machine Cooperative Systems**), pretože stroj nemôže vykonávať úplne všetky riadiace úlohy bez prítomnosti človeka.



Obr.19 Komponenty úplného riadiaceho systému

Výskum v oblasti **strojovej inteligencie** dal odpoveď aj na tretiu otázku. V súčasnosti už boli vyvinuté také metódy merania kvocientu MIQ, ktoré zohľadňujú orientáciu na ľudský faktor a predstavujú stupeň strojovej inteligencie blízky ľudskému vnímaniu a cíteniu.

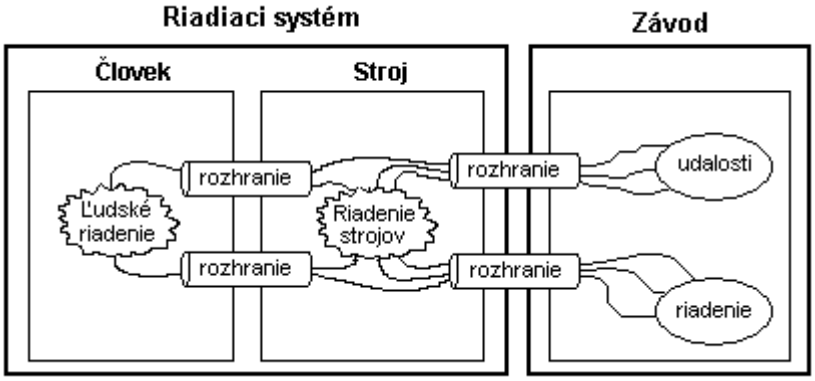
SAMOHODNOTIACE OTÁZKY

- Ktoré sú hlavné komponenty strojovej inteligencia?
- Môžeme jasne oddeliť strojovú inteligenciu od vplyvu ľudského faktora?
- Aká je úloha inteligencie riadenia?
- Čo reprezentuje inteligencia rozhrania?
- Ktoré podstatné vlastnosti by mali mať inteligentné stroje?
- Čo je to strojová inteligencia?
- Čo zohľadňujú metódy merania kvocientu MIQ?

8.4 Modelovanie systémov spolupráce človek – stroj

Model spolupráce systému človek-stroj slúži na analýzu nejasných vzťahov medzi ľuďmi, strojmi a celými prevádzkami.

Na obr.20 je znázornený model spolupráce systému človek-stroj a jeho riadiacich zásahov. Rôzne udalosti, ktoré sa vyskytnú v prevádzke alebo v riadiacom systéme pozostávajúcom z človeka a stroja by mali generovať v reálnom čase odpovedajúce **riadiace zásahy**.



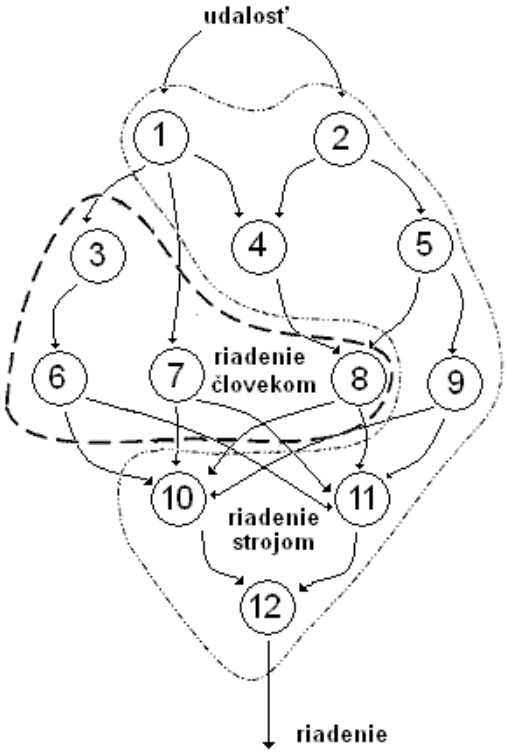
Obr.20 Model spolupráce človek-stroj

Riadiaci zásah môže byť rozdelený na niekoľko samostatných úloh, ktoré sú pridelené človeku alebo stroju. Ak stroj môže vykonať zadanú úlohu, v tom prípade nie je pridelená človeku. Avšak, ak stroj nemôže vykonať zadanú úlohu pre jej zložitosť alebo nedostatok „strojových“ informácií, potom je za jej vykonanie zodpovedný človek.

V oblasti paralelného spracovania údajov je množina úloh a ich vzájomných súvislostí zvyčajne popísaná pomocou **DFG** (**Data Flow Graph**), ktorého použitie bolo rozšírené aj na oblasť spracovania riadiacich úloh. Najznámejšie sú tzv. **rozhodovacie modely**, ktoré môžu

byť použité ako základný rámec pre zobrazenie rozdelenia úloh a ilustráciu vzájomnej komunikácie medzi operátorom a strojom. **Rozhodovací model** umožňuje viaceré možnosti správania sa systému, kde **rozhodovací** a **vykonávací** proces pozostáva s ôsmich krokov: ***detekcia***, ***pozorovanie***, ***indentifikácia***, ***realizácia***, ***vyhodnotenie***, ***definovanie úlohy***, ***výber akcie*** a ***vykonanie***. Avšak z týchto a podobných modelov nie je možné vyjadriť v číselnej forme inteligenciu vykonávania jednotlivých úloh alebo komunikácie. Pri paralelnom spracovaní procesov vieme vypočítať konečný čas spracovania aplikácie, ktorej úlohy sú rozdelené na spracovanie do niekoľkých procesorov. Z tejto myšlienky vychádza aj **metodika výpočtu strojovej inteligencie** ako zložiek **inteligencie vykonávania jednotlivých úloh**, **inteligencie vzájomného prepojenia** a **inteligencie rozdelenia úloh**.

Pre analýzu strojovej inteligencie sa vytvára tzv. **graf inteligenčných úloh** – **ITG graf**, ktorý pozostáva z kružníc a oblúkov. Riadiaci zásah môže byť rozdelený na niekoľko úloh a ich vykonanie je potom pridelené buď stroju alebo človeku. Kružniciam v ITG grafe prislúchajú jednotlivé úlohy riadiaceho zásahu a oblúky reprezentujú tok informácií medzi úlohami navzájom. Príklad ITG grafu je uvedený na obr. 21.



Obr.21 Príklad ITG grafu

Pre **výpočet strojovej inteligencie** sa definujú v grafe nasledujúce symboly:

- **množina úloh T**:

$$\mathbf{T = \{T_1, T_2, ..., T_n\}} \quad (1)$$

- **inteligencia potrebná na vykonanie úloh τ** :

$$\tau = \{\tau_1, \tau_2, ..., \tau_v\} \quad (2)$$

- **prenosová matica dát F**, kde $\mathbf{f_{i,j}}$ je množstvo dát prenášaných medzi úlohami $\mathbf{T_i}$ a $\mathbf{T_j}$:

$$\underline{F} = \begin{bmatrix} 0 & f_{12} & \cdots & f_{1n} \\ f_{21} & 0 & \cdots & f_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n1} & f_{n2} & \cdots & 0 \end{bmatrix} \tag{3}$$

- **zložitosť rozhrania c_{hm}** a **c_{mh}**: vyjadruje zložitosť prenosu dát medzi človekom a strojom prostredníctvom zobrazujúceho zariadenia, klávesnice, ovládacích prvkov, myši a pod. **c_{hm}** vyjadruje zložitosť rozhrania medzi človekom a strojom a **c_{mh}** medzi strojom a človekom.
- **inteligencia rozhrania $f_{ij}*c_{mh}$** a **$f_{ij}*c_{hm}$**: človeku musia byť zrozumiteľné dáta zobrazované z technológie, na základe ktorý robí odpovedajúce akčné zásahy do systému. Množstvo ľudskej inteligencie vynaloženej na komunikáciu so strojom predstavuje inteligenciu rozhrania úmernú tak počtu spracovávaných dát ako aj zložitosti rozhrania.
- **matica rozdelenia úloh A** : prvok tejto matice môže nadobudnúť hodnotu **0** alebo **1**. **a_{i1} = 1** ak stroj vykonáva úlohu **T_i**, **a_{i2} = 1** ak človek vykonáva úlohu **T_i**, ak úloha nemôže byť pridelená ani stroju ani človeku, tak prvok **a_{i3} = 1**. Z toho vyplýva, že **a_{i1} + a_{i2} + a_{i3} = 1** pre všetky **i** z intervalu <1, n>.

$$\underline{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} \end{bmatrix} \tag{4}$$

Kvocient **MIQ** môžeme **matematicky vyjadriť** ako **súčet kvocientu inteligencie riadenia CIQ** (Control Inteligence Quotient) a **kvocientu ľudskej inteligencie HIQ** (Human Inteligence Quotient).

Kvocient CIQ predstavuje celkovú inteligenciu riadiaceho systému pozostávajúceho z človeka a stroja. Je to suma inteligencií potrebných na vykonanie všetkých úloh:

$$CIQ = \sum_{i=1}^{n} a_{i1} \cdot \tau_i + \sum_{i=1}^{n} a_{i2} \cdot \tau_i \tag{5}$$

Kvocient HIQ vyjadruje množstvo ľudskej inteligencie potrebnej pre riadenie prevádzky. Je to suma inteligencií potrebných na vykonanie úloh určených pre človeka a inteligencie rozhrania.

$$HIQ = \sum_{i=1}^{n} a_{i2} \cdot \tau_i + c_{mh} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i1} a_{j2} f_{ij} + c_{hm} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i2} a_{j1} f_{ij} \tag{6}$$

Rozdiel týchto dvoch kvocientov určuje veľkosť kvocientu MIQ:

$$MIQ = CIQ - HIQ \quad (7)$$

Z rovnice (7) vyplýva, že ak rastie kvocient **CIQ** a a kvocient **HIQ** sa nemení, potom kvocient **MIQ** narastá. To znamená, že ak sú riadiace zásahy inteligentnejšie a používateľ vynakladá pri práci rovnakú námahu týkajúcu sa inteligenčného zaťaženia, potom by mala inteligencia stroja narastať. Rovnako platí, že ak sa nebudú meniť riadiace zásahy a nároky na inteligenciu používateľov budú nižšie, napríklad môžeme zameniť expertov za začiatočníkov alebo znížiť počet obsluhujúceho personálu, inteligencia stroja by sa mala zvyšovať.

SAMOHODNOTIACE OTÁZKY

- Na čo slúži model spolupráce systému človek-stroj?
- Aký význam má riadiaci zásah v modeloch spolupráce?
- Na čo slúži **DFG** graf?
- Na čo sa používajú rozhodovacie modely?
- Vymenujte kroky, z ktorých pozostáva rozhodovací model.
- Z čoho vychádza metodika výpočtu strojovej inteligencie?
- Na čo slúži **ITG** graf a z čoho pozostáva?
- Ktoré premenné potrebujeme pre výpočet kvocientu **MIQ**?
- Z ktorých dvoch kvocientov vieme matematicky vyjadriť kvocient MIQ?
- Čo vyjadruje kvocient **CIQ** a Čo kvocient **HIQ**?
- Aká je závislosť medzi kvocientami **CIQ** a **HIQ**?

8.5 Procedúra merania kvocientu MIQ

Meraním **mentálneho zaťaženia** a **množstva údajov** môžeme určiť premenné potrebné pre výpočet **kvocientu MIQ**: **inteligenciu potrebnú na vykonanie úloh**, **prenosovú maticu dát** a **zložitosť rozhrania**.

Problém predstavuje meranie mentálneho zaťaženia. **Mentálne zaťaženie** je definované ako stupeň kapacity spracovania informácií človekom počas vykonávania konkrétnej úlohy. Je zrejmé, že inteligencia potrebná na vykonanie úlohy a zložitosť rozhrania veľmi úzko súvisia s mentálnym zaťažením, z čoho vyplýva, že vyšší stupeň týchto premenných si vyžaduje väčšiu kapacitu spracovania informácií. Bolo vyvinutých mnoho techník a postupov pre určenie mentálneho zaťaženia, z ktorých niektoré našli aj praktické využitie pri návrhu užívateľských rozhraní.

V súčasnosti sa používa **6 vybraných metód** pre meranie **mentálneho zaťaženia**:

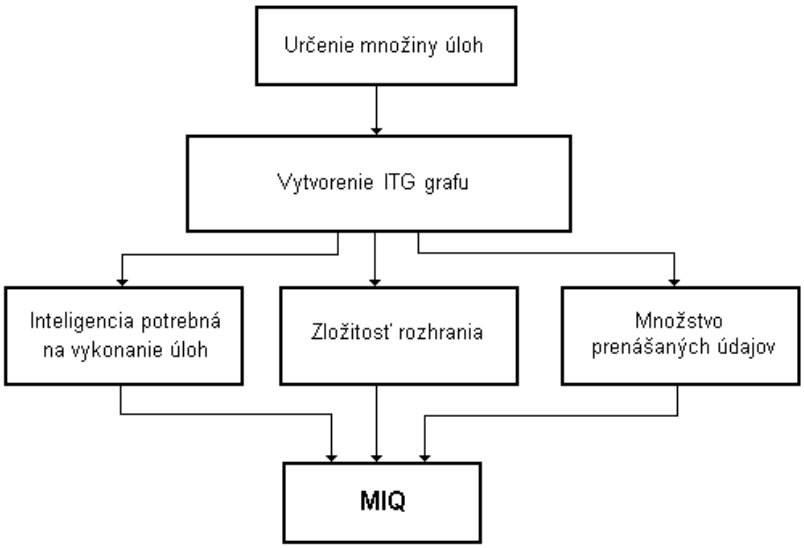
- metóda parametrov od signálov správania

- metóda duálnych úloh
- metóda merania informácií
- metóda očného snímania pohybov
- metóda subjektívneho merania
- metóda fyziologických premenných

Výber vhodnej metódy záleží na požadovanej kvalite a dostupných možnostiach.

Metóda určenia kvocientu MIQ na základe výpočtu uvedeného v kapitole **Modelovanie spolupráce systémov človek-stroj** pozostáva zo štyroch nasledujúcich krokov (obr.22):

1. určenie množiny úloh systému
2. zostavenie grafu inteligenčných úloh
3. meranie mentálneho zaťaženia pre určenie inteligencie potrebnej na vykonanie úloh a zložitosti rozhrania, ktoré ďalej slúžia na určenie prenosovej matice systému, inteligencie rozhrania a matice rozdelenia úloh
4. výpočet kvocientu MIQ



Obr.22 Metóda určenia koeficientu MIQ

SAMOHODNOTIACE OTÁZKY

- Ako môžeme určiť premenné potrebné pre výpočet kvocientu **MIQ**?
- Ako je definované mentálne zaťaženie?
- Ktoré metódy sa používajú na meranie mentálneho zaťaženia?
- Od čoho závisí výber vhodnej metódy?
- Vymenujte kroky, z ktorých pozostáva metóda určenia kvocientu **MIQ**.

9 KOMUNIKÁCIA V PRIEMYSELNEJ AUTOMATIZÁCII

Táto kapitola sa zaoberá objasnením základných pojmov a princípov z oblasti prenosu informácií v číslicovom tvare medzi jednotlivými zariadeniami a úrovňami distribuovaného riadiaceho systému ako aj rôznymi systémami navzájom z hľadiska charakteru sieti číslicových zariadení a charakteru úloh, ktorý je potrebné zabezpečiť.

9.1 Úvod a ciele

Začiatkom 60. rokov dvadsiateho storočia sa začali na riadenie technologických procesov používať číslicové počítače ako centrálne riadiace prvky v priamom číslicovom riadení (DDC – Direct Digital Control). V 60. rokoch boli vyvinuté programovateľné logické automaty (PLC), stroje riadené počítačom (CNC), a v praxi sa začali nasadzovať priemyselné roboty. Ruka v ruke so zavádzaním číslicových počítačov a zariadení vznikali špeciálne priemyselné komunikačné siete.

V polovici 70. rokov uviedla firma Honeywell prvý hierarchický distribuovaný riadiaci systém (DCS – Distributed Control System), ktorý pozostával z množstva mikroprocesorov. Takisto mnohé súčasné priemyselné automatizované systémy majú otvorenú distribuovanú architektúru s komunikáciou prostredníctvom digitálnych komunikačných sietí.

CIELE

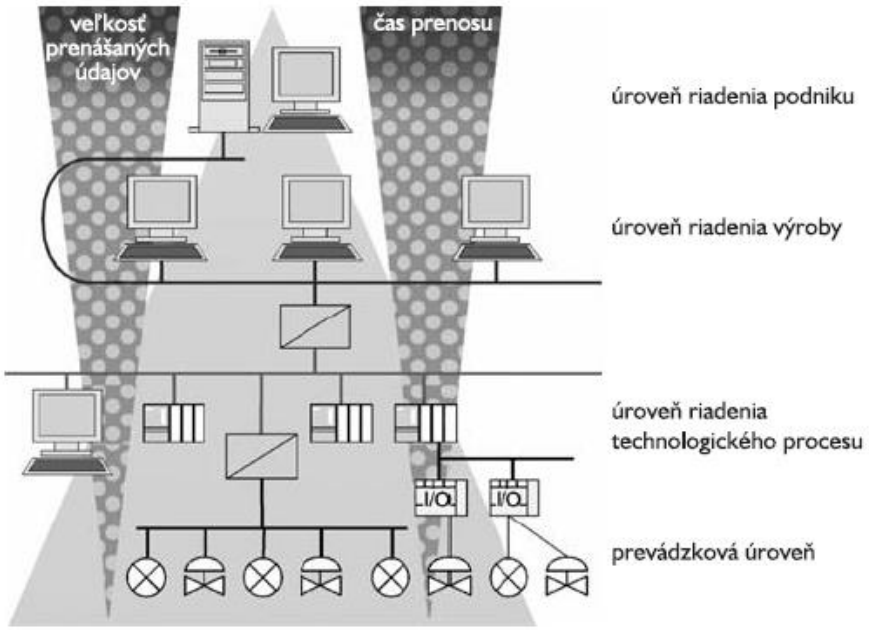
Cieľom tejto kapitoly je naučiť sa:

- vysvetliť a definovať základné pojmy ako model komunikácie OSI, protokol, priemyselná zbernica a pod.,
- z ktorých vrstiev pozostáva model komunikácie OSI,
- aké služby poskytujú jednotlivé vrstvy komunikačného modelu OSI,
- aká je štruktúra komunikačného modelu OSI pre priemyselné aplikácie a lokálne siete,
- ako sa líši komunikačný model pre internet od komunikačného modelu OSI.

9.2 Referenčný model komunikácie OSI

Automatizovaný výrobný systém sa zvyčajne člení na niekoľko hierarchických úrovní. Každá z týchto úrovní obsahuje vlastnú komunikačnú úroveň s určitými nárokmi na príslušný komunikačný systém. Príklad hierarchického usporiadania automatizovaného výrobného systému je na obr. 23.
S postupným zväčšovaním rozsahu priemyselných automatizovaných systémov sa dostávala do popredia otázka zavedenia štandardov pre systémy na prenos dát medzi rôznymi zariadeniami. Aby sa predišlo problémom súvisiacim s používaním veľkého množstva nekompatibilných štandardov pre systémy určené na prenos informácií v číslicovom tvare, medzinárodná organizácia ISO (International Organization for Standardization) definovala model na komunikáciu otvorených systémov OSI (Open System Interconnection). OSI sám o

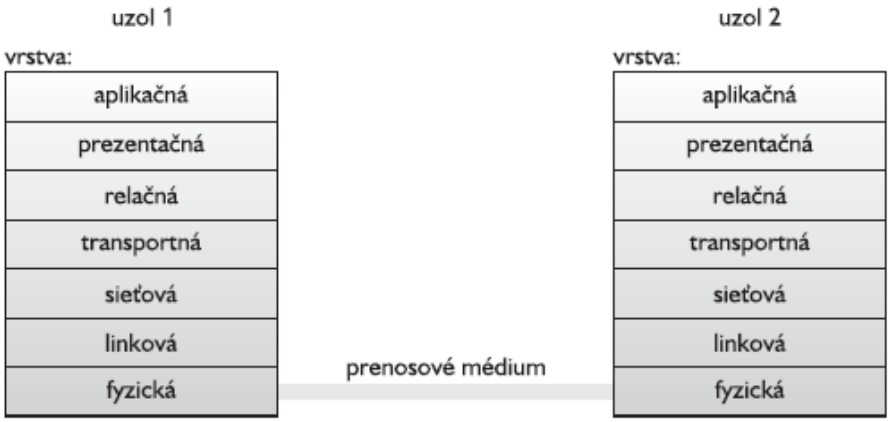
sebe nie je štandardom, ale ponúka určitý návod, ako identifikovať a oddeliť koncepčne odlišné časti komunikačného procesu.



Obr.23 Hierarchia automatizovaného výrobného systému

9.2.1 Charakteristika modelu ISO/OSI

Vzhľadom na charakter sietí číslicových zariadení a charakter úloh, ktoré treba zabezpečiť, sa ako najvhodnejšie riešenie ukázala dekompozícia základného programového vybavenia siete na hierarchicky usporiadané vrstvy. Každá vrstva má na starosti zabezpečenie presne vymedzeného okruhu úloh. Mechanizmy, pomocou ktorých tieto úlohy zaisťuje, potom poskytuje ako služby vrstve bezprostredne vyššej. Napríklad vrstva, ktorá zaisťuje prenos jednotlivých bitov, môže poskytovať svoje služby bezprostredne vyššej vrstve, ktorá pomocou nej prenáša celé bloky údajov. Vo všeobecnosti každá vrstva poskytuje určitú množinu služieb vrstve bezprostredne vyššej, pričom na ich realizáciu využíva služby vrstvy bezprostredne nižšej. Každá vrstva síce využíva služby bezprostredne nižšej vrstvy, jej partnerom pri komunikácii v sieti je však vrstva, ktorá sa v inom uzle siete nachádza na rovnakej úrovni hierarchie vrstiev. Tieto rovnoľahlé vrstvy musia byť dohovorené na spoločných pravidlách komunikácie.
Súbor pravidiel, ktoré rovnoľahlé vrstvy vrstvového modelu používajú na vzájomnú komunikáciu, tvorí tzv. **protokol.**
V modeli OSI je definovaných sedem funkčných vrstiev (obr. 24). V každej vrstve definuje služby, ktoré musí mať vrstva k dispozícii pre vyššiu vrstvu. Služby (čo spraviť) sú striktne odlíšené od protokolu (skutočná realizácia, ako to spraviť). Vzájomná prepojiteľnosť je postavená na skutočnosti, že rozdielne systémy sú štruktúrované na základe podobných služieb a že v rovnoľahlých vrstvách sú tie isté protokoly.



Obr.24 Model prepojovania otvorených systémov (OSI)

9.2.2 Funkcia a úlohy vrstiev modelu OSI

1. Fyzická vrstva. Úlohou tejto vrstvy je zaistiť prenos jednotlivých bitov medzi príjemcom a odosielateľom prostredníctvom fyzickej prenosovej cesty, ktorú táto vrstva bezprostredne ovláda. Fyzická vrstva jednotlivé prenášané bity neinterpretuje. Obsahuje elektrické, mechanické a optické rozhrania spolu s potrebnými softvérovými ovládačmi komunikačných portov. Na úrovni fyzickej vrstvy je definovaná topológia (spôsob prepojenia zariadení na úrovni prenosu signálu).

2. Linková vrstva. Táto vrstva má za úlohu zaistiť prenos celých blokov údajov, označovaných ako rámce medzi dvoma susednými uzlami (čiže zabezpečuje prenos medzi uzlami, medzi ktorými je priame spojenie, napr. uzly na tom istom segmente siete). Linková vrstva má správne rozpoznať začiatok a koniec rámca, ako aj jeho jednotlivé časti. Zabezpečuje verifikáciu toho, či postupnosť bitov prešla medzi dvoma uzlami korektne.

3. Sieťová vrstva. Zriaďuje kompletnú cestu a dohliada na to, aby cestou od zdroja do cieľa prešli všetky správy aj v prípade, že táto cesta je zostavená z odlišných vetiev prechádzajúcich niekoľkými uzlami. Musí zaistiť potrebné smerovanie (voľbu vhodnej trasy) prenášaných blokov údajov označovaných ako pakety a postupné odovzdávanie jednotlivých paketov po tejto trase od pôvodného odosielateľa až ku koncovému príjemcovi. Sieťová vrstva si teda musí „uvedomovať“ konkrétnu topológiu siete (t. j. spôsob vzájomného priameho prepojenia jednotlivých uzlov).

4. Transportná vrstva. Táto vrstva zabezpečuje koncové riadenie komunikácie (t. j. medzi pôvodným odosielateľom a konečným príjemcom) a funguje ako rozhranie medzi aplikačným softvérom, ktorý požaduje údajovú komunikáciu, a externou sieťou. Táto vrstva má prostredníctvom sieťovej vrstvy vytvorenú ilúziu, že každý uzol siete má priame spojenie s ktorýmkoľvek iným uzlom siete. Vďaka tomu sa venuje už len komunikácii medzi pôvodným odosielateľom a konečným príjemcom.

5. Relačná vrstva. Jej úlohou je nadväzovať, udržiavať a rušiť relácie medzi koncovými účastníkmi. V rámci nadväzovania relácie si táto vrstva vyžiada od transportnej vrstvy vytvorenie spojenia, prostredníctvom ktorého potom prebieha komunikácia medzi oboma účastníkmi relácie.

6. Prezentačná vrstva. Jednotlivé počítače môžu používať navzájom odlišnú vnútornú reprezentáciu údajov (napr. kódy EBCDIC, ASCII atď.). Táto vrstva zabezpečuje teda potrebné kódovanie údajov a konverziu, pomocou ktorých sú binárne údaje prevedené na to,

čo samy o sebe znamenajú: správy, texty, obrázky a podobne. Na zaistenie konverzie údajov je zavedený jazyk ASN.1.

7. Aplikačná vrstva. Koncoví používatelia využívajú počítačové siete prostredníctvom najrôznejších sieťových aplikácií – systémov elektronickej pošty, prenosov súborov, vzdialeného prihlasovania a pod. Začleňovať všetky tieto rôznorodé aplikácie priamo do aplikačnej vrstvy by pre ich veľkú rôznorodosť nebolo vhodné. Preto sa do aplikačnej vrstvy zahrnujú len tie časti aplikácií, ktoré realizujú spoločné, resp. všeobecne použiteľné mechanizmy. Táto najvyššia vrstva sa zaoberá úlohami manažmentu aplikačného systému, ako je prenos údajových súborov, činnosť distribuovaných databáz a diaľkové riadenie.

Pre každú vrstvu **OSI** existuje jeden alebo viac súborov štandardov vydaných štandardizačnými organizáciami.
Na fyzickej a linkovej úrovni boli do **OSI** zahrnuté aj niektoré zo skorších štandardov. Pre ostatné úrovne boli definované nové protokoly, ktoré sa pridŕžajú modelu **OSI.**
Plná kompatibilita medzi rôznymi vrstvami znamená, že principiálne možno zostaviť fungujúcu aplikáciu aj pomocou zariadení od rôznych výrobcov. Vo všeobecnosti je to trochu zložitejšie. Jednotlivé sprostredkujúce vrstvy **OSI** sa na trhu nepredávajú ako samostatné softvérové balíky a výrobcovia a návrhári softvéru ponúkajú namiesto toho balíky pre úrovne 3 – 4 až 6 – 7. Vnútorné rozhrania nemusia dodržiavať požiadavky **OSI.** Preto sa namiesto podpory všetkých vrstvových protokolov **OSI** navrhuje, pokiaľ možno, čo najefektívnejšie programové vybavenie.

SAMOHODNOTIACE OTÁZKY

- Vysvetlite skratky **ISO** a **OSI.**
- Vysvetlite princíp prenosu informácií v systéme hierarchicky usporiadaných vrstiev.
- Vysvetlite pojem „**protokol**“.
- Uveďte koľko vrstiev je definovaných v systéme **OSI** a vymenujte ich.
- Charakterizujte funkcie a úlohy jednotlivých vrstiev modelu **OSI.**
- Vysvetlite, či systém na prenos dát musí mať implementované všetky vrstvy modelu **OSI** a od čoho závisí nutnosť implementácie konkrétnej vrstvy?

9.3 OSI a priemyselné komunikačné zbernice

Priemyselné zbernice sú digitálne komunikačné linky, ktoré sa používajú na prenos dát na najspodnejších úrovniach riadenia (na úrovni riadenia technologického procesu a prevádzky). Prenášajú sa nimi krátke bloky dát na úrovni snímačov, akčných členov a regulátorov. Na ich činnosť nie sú potrebné (ani žiaduce) všetky vrstvy modelu **OSI** (obr.25).

Používajú sa tieto vrstvy modelu **OSI**:

Fyzická vrstva poskytuje elektrické a elektromechanické rozhranie pre prenosové médium,

ktoré zabezpečuje kódovanie a dekódovanie údajov.

Na úrovni fyzickej vrstvy má každá **komunikačná zbernica** definované **fyzické charakteristiky** komunikačných obvodov:

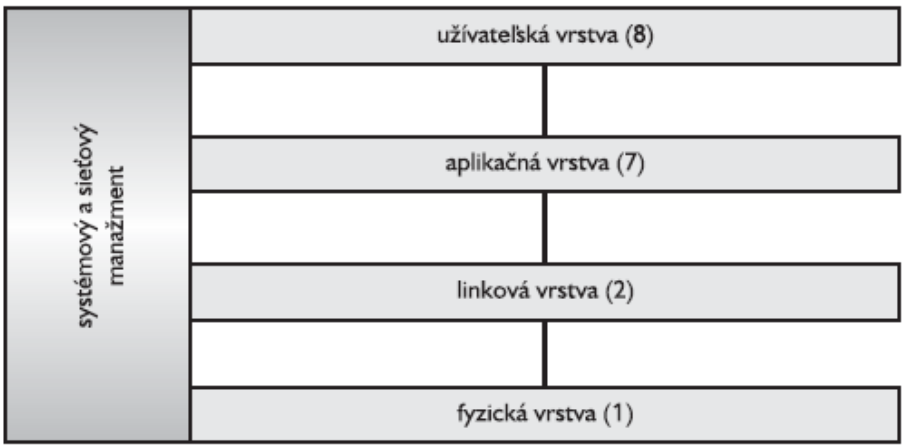
- typ prenosového média (médií)
- napäťové úrovne
- veľkosť zaťažovacích prúdov
- prenosovú rýchlosť
- topológiu
- vlastnosti prijímačov zbernice (prijímače optických signálov)
- maximálny počet pripojiteľných zariadení (uzlov) a pod.

Linková vrstva zriaďuje spojenie medzi dvoma uzlami zbernice a zabezpečuje prenos rámcov.

Aplikačná vrstva zabezpečuje preklad požiadaviek používateľskej vrstvy do linkovej vrstvy. Umožňuje prístup do množiny komunikačných služieb podporujúcich činnosť distribuovaných systémov.

V používateľskej vrstve je definovaná štruktúra zberu údajov a riadiace funkcie zariadenia pripojeného na priemyselnú zbernicu. To znamená, že definuje štruktúru databázy, ktorá sa bude nachádzať v každom meracom, resp. riadiacom zariadení pripojenom na zbernicu. Definujú sa tu rôzne funkčné bloky, ich atribúty, módy a pod.

Systémový a sieťový manažment poskytuje metódy na konfiguráciu zbernice a zotavenie sa z porúch. Monitoruje a riadi činnosť jednotlivých častí zbernice.



Obr.25 Vrstvový model priemyselnej komunikačnej zbernice

SAMOHODNOTIACE OTÁZKY

- Vysvetlite pojem **priemyselná zbernica** a uveďte na čo slúžia.
- Ktoré vrstvy modelu **OSI** obsahujú priemyselné zbernice?
- Vymenujte úlohy jednotlivých vrstiev modelu **OSI** v priemyselných zberniciach.

	• Vymenujte fyzické charakteristiky komunikačných zberníc na úrovní na úrovní fyzickej vrstvy.

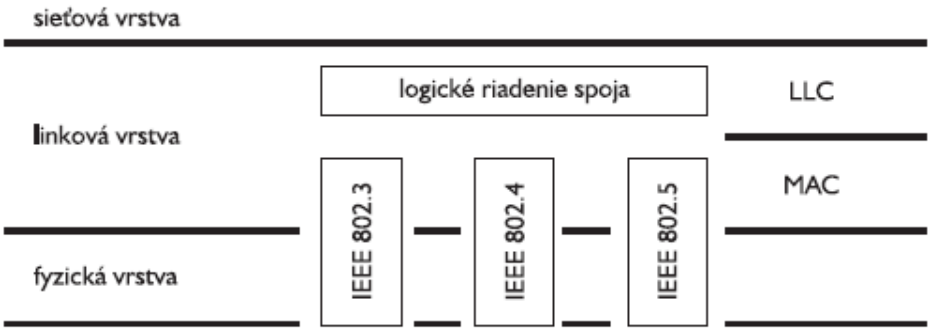
9.4 OSI a lokálne siete

Na vzájomné prepojenie viacerých miest sa využívajú **lokálne počítačové siete** (**LAN**). V automatizovanom výrobnom systéme sa používajú predovšetkým na vyšších úrovniach riadenia (úroveň riadenia podniku a výroby).

LAN pokrývajú spodné dve vrstvy modelu **OSI.**

Linková vrstva je rozdelená na dve podvrstvy:

- riadenie prístupu na médium (**MAC** – **Medium Access Control**), čo je reálne rozhranie na fyzické médium. Úlohou podvrstvy **MAC** je realizovať algoritmus prístupu na médium – prístupovú metódu, t. j. zabezpečiť bezkolízny prístup na prenosové médium.
- a logické riadenie spoja (**LLC** – **Logical Link Control**), ktorá je zodpovedná za koordináciu prístupu na sieť (detekcia chýb, riadenie toku údajov).



Obr. 26 Architektúra štandardov LAN podľa modelu OSI

Koncepcia je ilustrovaná na obr. 26. Fyzické médium môže byť to isté pre rôzne typy **LAN** (napr. koaxiálny kábel, krútená dvojlinka a iné).

Signály, typ modulácie a protokol prístupu na médium sa pri rôznych **LAN** líšia. A nakoniec, vyššie riadenie činnosti **LAN** je opäť to isté.

V **LLC** a vyšších vrstvách nie je dôležité vedieť, ktorý typ **LAN** sa používa.

Na obr. 26 sú uvedené tri štandardy LAN: IEEE 802.3 – ethernet, IEEE 802.4 – Token Bus a IEEE 802.5 – Token Ring.

SAMOHODNOTIACE OTÁZKY

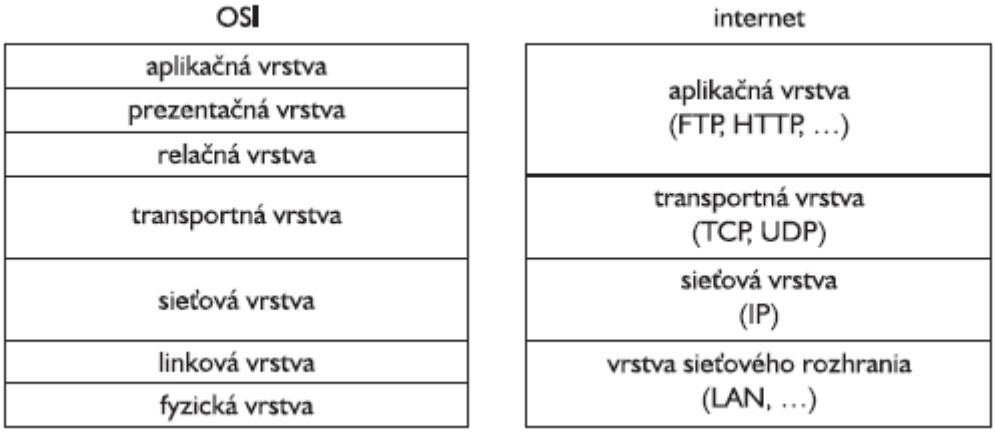
- Vysvetlite pojem **LAN** a uveďte na čo sa používajú.
- Vymenujte vrstvy modelu OSI, ktoré pokrýva **LAN.**
- Vysvetlite skratky **MAC** a **LLC.**
- Charakterizujte úlohy jednotlivých podvrstiev linkovej vrstvy v **LAN.**

9.5 OSI a Internet

Internetové protokoly nie sú realizované na základe modelu **OSI**.

Pre internet sú definované **4 vrstvy**:

Vrstva sieťového rozhrania (zodpovedá fyzickej a linkovej vrstve OSI). Zabezpečuje prenos paketov nadradenej vrstvy prostredníctvom špecifického prenosového média.



Obr. 27 Vrstvy modelu OSI a internetu

Sieťová vrstva. Používa protokol nazývaný **Internetwork Protocol** – skratka **IP** a pomocou neho zabezpečuje:

- služby doručenia správ bez závislosti od prenosového média,
- mechanizmus adresovania,
- mechanizmus smerovania správ.

Transportná vrstva. Zabezpečuje spoľahlivý prenos dát medzi koncovými uzlami. Na prenos údajov sa používajú dva typy protokolov:

- **TCP** (Transmission Control Protocol)
- a **UDP** (User Datagram Protocol).

Aplikačná vrstva (zodpovedá aplikačnej vrstve OSI). Obsahuje protokoly, ktoré špecifikujú procedúry pre používateľa (prihlasovanie k serverom, prenos súborov a iné). Procedúry (aplikácie) sú rozdelené podľa použitého protokolu transportnej vrstvy.

SAMOHODNOTIACE OTÁZKY

- Uveďte rozdiely medzi štruktúrou komunikačných vrstiev modelu **OSI** a internetu.
- Aký protokol používa sieťová vrstva internetu a ktoré úlohy pomocou neho zabezpečuje?
- Vymenujte typy protokolov používaných v transportnej vrstve internetu.
- Charakterizujte aplikačnú vrstvu internetu.

10 ISO NORMY PRE TVORBU POUŽÍVATEĽSKÝCH ROZHRANÍ

Táto kapitola sa zaoberá vymedzením štruktúry slovenskej verzie normy **STN EN ISO 9241**, ktorá získala postavenie národnej normy jej schválením Slovenským ústavom technickej normalizácie a vymedzením predmetu a stručným opisom tých jej častí, ktoré sú zamerané na tvorbu používateľských rozhraní.

10.1 Úvod a ciele

Norma STN EN ISO 9241 má byť prínosom pre **návrhárov používateľského rozhrania**. Jej princípy, uplatnené v praxi, bude však v konečnom dôsledku využívať používateľ, ktorého potreby sú určované ergonomickými požiadavkami, zohľadnenými pri tvorbe noriem.

Aplikácia normy by mala viesť k vytvoreniu rozhrania, ktoré by vďaka svojej konzistentnosti a vhodnosti pre používateľa viedlo k zvýšeniu produktivity jeho práce.

CIELE

Cieľom tejto kapitoly je naučiť sa:

- ktorá slovenská norma sa zaoberá tvorbou používateľských rozhraní,
- aká je štruktúra normy STN EN ISO 9241 a ktoré jej časti sa zaoberajú tvorbou používateľských rozhraní,
- stručne popísať predmet a obsah tých častí normy STN EN ISO 9241, ktoré sa zaoberajú tvorbou používateľských rozhraní.

10.2 Norma STN EN ISO 9241

Slovenská verzia normy **STN EN ISO 9241** získala postavenie národnej normy jej schválením Slovenským ústavom technickej normalizácie. Jednotlivé časti normy **STN EN ISO 9241,** vydané v slovenskom jazyku, majú obsah identický s obsahom **medzinárodnej normy EN ISO 9241**.

Norma **STN EN ISO 9241** pozostáva zo 17 častí, z ktorých časti 10, 13, 14 a 17 sú zamerané na tvorbu používateľských rozhraní:

- **ISO 9241-10:1996 - Základné zásady vytvárania dialógu.**
- **ISO 9241-13:1998 - Príručka používateľa.**
- **ISO 9241-14:1997 - Vedenie dialógu pomocou menu.**
- **ISO 9241-17:1998 - Vedenie dialógu pomocou obrazových formulárov.**

Jednotlivé časti normy sú spracované **v rovnakej štruktúre.**

V úvode je vymedzený predmet normy, odkazy na súvisiace normy, ďalej nasleduje vysvetlenie použitých pojmov a definícií.

V každej časti je uvedená aplikovateľnosť a vhodnosť použitia odporúčaní danej časti normy v určitých podmienkach.

Jadro tvorí zoznam podmienených odporúčaní, ktoré sú špecifické pre každú časť normy. V niektorých prípadoch sú odporúčania doplnené praktickými príkladmi.

Prílohou každej časti normy je postup na posúdenie použiteľnosti a zhody, výsledkom ktorého má byť konštatovanie, či sú odporúčania normy pre konkrétny posudzovaný prípad splnené.

AKTIVITA

- Pokúste sa nájsť pomocou Internetu celé znenie normy **STN EN ISO 9241.**
- Presvedčite sa, že časti 10, 13, 14 a 17 sú venované tvorbe používateľských rozhraní.

SAMOHODNOTIACE OTÁZKY

- Ktorá medzinárodná norma má identický obsah ako slovenská verzia normy **STN EN ISO 9241?**
- Ktoré časti normy **STN EN ISO 9241** sú venované tvorbe používateľského rozhrania?
- Aká je štruktúra normy **STN EN ISO 9241?**
- Stručne charakterizujte obsah každej časti normy **STN EN ISO 9241.**

10.3 ISO 9241-10: Základné zásady vytvárania dialógu

Táto časť normy obsahuje **všeobecné ergonomické zásady**, týkajúce sa akýchkoľvek pracovných situácií, aplikácií, prostredia a techniky.

Pre **tvorbu** a **vyhodnocovanie dialógu** norma uvádza **všeobecné zásady**:

- primeranosť úlohám,
- schopnosť vlastného opisu úlohy,
- ovládateľnosť,
- konformita očakávania,
- tolerancia chýb,
- individuálne riešenie,
- podpora schopnosti učenia.

a **používateľské parametre**:

- trvanie pozornosti (sústredenosti),

- medze krátkodobej pamäti,
- učebné zvyklosti,
- stupeň skúseností s prácou s dialógovým systémom,
- mentálny model používateľa.

a **parametre úloh**:

- pre zvýšenie efektívnosti a účinnosti realizácie pracovných úloh je nevyhnutné splnenie požiadaviek, vyplývajúcich z riešenia úloh.

Príklad odporúčania normy

Konformita očakávania:

Dialógový systém by mal používať slovnú zásobu, ktorá je používateľovi známa z realizácie pracovných úloh.

Príklad:

Odborné výrazy, ktoré sa používajú v dialógu, sú totožné s tými, ktoré sa skutočne používajú v oblasti pracovných úloh používateľa.

AKTIVITA

- Pokúste sa nájsť pomocou Internetu celé znenie normy ISO 9241-10.
- Vyberte si jednu položku zo všeobecných zásad a jednu položku z používateľských parametrov a pomocou normy ich podrobnejšie naštudujte a vysvetlite.

10.4 ISO 9241-13: Príručka používateľa

Príručka používateľa v ponímaní tejto normy predstavuje **doplňujúcu informáciu** k bežnému dialógu **používateľ – počítač**, ktorá sa poskytuje používateľovi na požiadanie alebo ju automaticky poskytuje systém. Podľa nami zaužívanej terminológie je výraz **„Príručka používateľa“** zrejme vhodnejšie nahradiť výrazom **„Pomoc používateľovi“**, príp. **„Vedenie používateľa“**, ktorý zreteľnejšie vystihuje podstatu tohto pojmu používaného v norme.

Odporúčania tejto časti **ISO 9241** sú aplikovateľné na prvky interakcie, ktoré pomáhajú používateľom **pri riešení chybových stavov**. Príručka používateľa (pomoc používateľovi, podpora používateľa) má podľa normy zahŕňať odporúčania, špecifické pre aplikačné oznamy, spätné hlásenia, chybové správy a on-line pomoc, ako aj všeobecné odporúčania, obvyklé pre všetky typy príručky používateľa.

Odporúčania sú formulované tak, aby boli nezávislé od aplikácií, prostredia alebo implementačnej technológie. Korešpondujú s typickými situáciami, kde sa vyžadujú špeciálne potreby informácií a akcií.

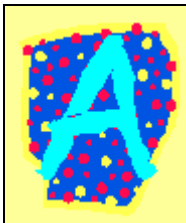
Príklad odporúčania normy

Všeobecné odporúčanie pre štylizovanie príručky používateľa:

Správy používateľskej príručky majú byť formulované tak, aby zvýšili pozornosť používateľa.

SPRÁVNE**: Na uchovanie zmien stlačte OK.**

NESPRÁVNE*:* **Systém uchová vaše zmeny, len ak stlačíte OK.**



AKTIVITA

- Pokúste sa nájsť pomocou Internetu celé znenie normy ISO 9241-13.
- Vyberte si niektoré zo všeobecných odporúčaní a pomocou normy ho podrobnejšie naštudujte a vysvetlite.

10.5 ISO 9241-14: Vedenie dialógu pomocou menu

Norma **ISO 9241-14** opisuje **princípy navrhovania dialógov pomocou menu**. Tieto princípy poskytujú navrhovateľovi doplňujúce informácie, týkajúce sa ergonomickej podstaty dizajnu dialógov pomocou menu.

V dialógoch pomocou menu **dialógový systém** predstavuje jednu alebo viac skupín možností pre používateľa. Odporúčania normy by sa mali splniť len v rámci určitého kontextu, pre ktorý sú relevantné (napr. konkrétne typy používateľov, úloh, prostredí, technológií). Odporúčania pokrývajú menu prezentované rôznymi technikami, vrátane používania okien, panelov, tlačidiel, polí atď.

Dizajn rozhrania závisí od **úlohy, používateľa, prostredia** a **dostupnej technológie**. Predpokladá sa, že návrhár rozhrania má dostupné informácie, týkajúce sa požiadaviek úlohy a používateľa a rozumie použitej technológii. To môže vyžadovať konzultácie s kvalifikovanými profesionálmi z oblasti ergonómie, ako aj empirické testovanie so skutočnými používateľmi.

Odporúčania sa týkajú troch hlavných prvkov dizajnu používateľského rozhrania, t. j. **dialógu**, **vstupu** a **výstupu**.

Dizajn dialógu určuje spôsob, akým je používateľ riadený systémom pre zadanie vstupov a ovplyvňuje rozsah kontroly, ktorú má používateľ nad dialógom.

Odporúčania normy pre **návrh vstupov** sa týkajú možností vkladania informácií do systému a používania rôznych vstupných zariadení. Možnosti menu môžu byť vybrané prostredníctvom jedného alebo viacerých vstupných zariadení, ako je alfanumerická klávesnica, funkčné klávesy, kurzorové klávesy, myš a pod.

Návrh výstupu hovorí o tom, že údaje prezentované na obrazovke by mali konzistentné a jednoznačne odlíšiteľné.

ISO 9241-14 poskytuje **podmienečné odporúčania** na rozmiestnenie možností a skupín možností, štruktúru a syntax možností a prezentáciu techník na ich vyznačenie a selektivitu.

Príklad odporúčania normy

Štruktúra menu / Logické kategórie:

Ak možnosti (voľby) nemajú konvenčné zoskupenia alebo štruktúru, ale môžu sa zoskupiť spôsobom, ktorý je jednoznačný a používateľ si ho ľahko osvojí, mali by sa usporiadať tak, aby sa minimalizoval počet úrovní a maximalizoval počet možností v jednom menu.

Príklad:

Možnosti (voľby) s objektom sú umiestnené do jednej skupiny a možnosti s činnosťami do druhej skupiny.

AKTIVITA

- Pokúste sa nájsť pomocou Internetu celé znenie normy ISO 9241-14.
- Zistite, ktoré odporúčania normy ISO 9241-14 sú v programe Control Web 2000 dodržané a ktoré nie sú dodržané.

10.6 ISO 9241-17: Vedenie dialógu pomocou obrazových formulárov

Formulárové dialógy sú dialógy, pri ktorých používateľ zadáva vstupy, vypĺňa alebo upravuje označenie polí vo formulári, ponúkanom systémom. Typické pre vypĺňanie formulára sú vstupy vo forme písaných vstupov (skratky alebo plné názvy) alebo výber možností z ponúknutého zoznamu.

Táto časť normy sa týka **formulárových dialógov**, založených buď na báze znakov, alebo bitovej mapy a vstupov, zadávaných z klávesnice alebo pomocou voliteľného ukazovacieho zariadenia (napr. myš).

Okrem toho je predmetom tejto časti normy použitie netextových metód ponuky formulárových vstupov (napr. zoznamové okná a dialógové okná). Návrh dialógu určuje spôsob, akým je používateľ vedený systémom, aby zadával vstupy a ako ovplyvňuje rozsah kontroly používateľa nad dialógom.

Formulárové dialógy by sa mali navrhovať tak, aby **podporovali** používateľa pri jeho skutočnej práci bez vytvárania ďalšej práce v dôsledku zvláštností systému a umožnili zachovanie kontroly nad priebehom práce.

Odporúčania sa vzťahujú na tri hlavné zložky používateľského rozhrania, na **dialóg**, **vstup** a **výstup**.

Pre navrhovanie vstupov uvádza norma **odporúčania**, ako možno použiť vstupné zariadenie na uľahčenie zadávania vstupov a odporúčania na úpravu polí formulára.

Odporúčania pre návrh výstupov sa týkajú toho, ako zreteľne a presne sa majú dáta prezentovať na obrazovke v zmysle ich obsahu a umiestnenia polí a skupín polí.

Príklad odporúčania normy

Požiadavky na vstup / Rozlíšiteľné vizuálne inštrukcie:

Rozlíšiteľné vizuálne inštrukcie by sa mali používať na rozlíšenie rôznych logických typov voliteľných vstupov v aplikácii.

Príklad:

Prepínače sa používajú na označenie výlučnej voľby (ovládacie prvky) a zaškrtávacie políčka (voliteľné schránky) sa používajú na označenie nevýlučnej voľby.

AKTIVITA

- Pokúste sa nájsť pomocou Internetu celé znenie normy ISO 9241-17.
- Zistite, ktoré odporúčania obsahuje norma pre navrhovanie vstupov a výstupov.

10.7 Postup na posúdenie použiteľnosti a zhody

Poskytovanie možnosti prispôsobiť rozhranie požiadavkám používateľa tak, aby vyhovovalo jeho špecifickým potrebám, sa stalo populárnym prístupom k navrhovaniu **softvérového rozhrania**. Avšak takéto riešenie nie je prijateľnou náhradou za ergonomicky navrhnuté pôvodné rozhranie. Preto by sa možnosti úpravy podľa požiadaviek používateľa mali vyhodnotiť aj s ohľadom na odporúčania tejto normy.

Vzorové postupy, ktoré podporujú **určenie aplikovateľnosti** odporúčaní normy a postupy pre zistenie, či odporúčanie normy bolo zohľadnené, sú súčasťou každej z častí **ISO 9241**.

Postup predstavuje proces s dvomi krokmi:

1. stanovenie odporúčaní, ktoré sú použiteľné
2. posúdenie, či boli relevantné odporúčania splnené

Postup posudzovania by sa mal zakladať na **analýze typických** používateľov, ich typických a podstatných pracovných úloh a na typickom pracovnom prostredí.

Posudzovanie možno vo všeobecnosti rozdeliť do dvoch kategórií:

a) Ak sú známe skupiny používateľov aj pracovné úlohy, posudzovateľ vyhodnotí produkt, alebo pozoruje reprezentatívneho používateľa, ako plní typické a podstatné pracovné úlohy v typických podmienkach použitia.

b) Ak sú skupiny používateľov a pracovné úlohy neznáme, posudzovateľ vyhodnotí všetky povely, použité pri posudzovanom produkte.

Použiteľnosť

Použiteľnosť určitého odporúčania závisí od **dvoch faktorov**:

a) Od toho, či je pravdivý podmienený výrok **„ak“,** v prípade, že je časťou odporúčania. Určité odporúčanie je (alebo nie je) použiteľné, ak je (alebo nie je) podmienený výrok pravdivý.

b) Od **používateľského kontextu**. Určité odporúčanie nemusí byť použiteľné napríklad pre špecifické podmienky skupiny používateľov, pracovnej úlohy, pracovného prostredia a technológie. Ak však prostredie navrhovania zahŕňa konkrétne používateľské charakteristiky, pracovné úlohy alebo technické parametre, ktoré boli uvedené v odporúčaní, potom sa toto odporúčanie musí uplatniť.

Vhodné **postupy** na posúdenie **použiteľnosti** určitého odporúčania sú:

a) **Systémová dokumentačná analýza**

b) **Dokumentovaná evidencia**

c) **Pozorovanie**

d) **Analytické posudzovanie**

e) **Empirické posudzovanie**

Zhoda

Ak je určité **odporúčanie použiteľné** na základe vyššie popísaných kritérií **použiteľnosti**, potom sa na základe jedného alebo niekoľkých nasledujúcich postupov musí posúdiť jeho dodržanie.

Postupy na posudzovanie **zhody**

a) **Meranie**

b) **Pozorovanie**

c) **Dokumentovaná evidencia**

d) **Analytické posudzovanie**

e) **Empirické posudzovanie**

Kontrolný zoznam

Kontrolný zoznam je pomôckou pre **návrhárov** a **posudzovateľov kvality** pri posudzovaní **použiteľnosti** a **zhody** jednotlivých podmienených odporúčaní. Obsahuje všetky odporúčania príslušnej časti ISO v stručnej podobe a poskytuje logickú štruktúru na posudzovanie použiteľnosti. Popri postupoch na určenie **použiteľnosti** a **zhody** je **kontrolný zoznam** taktiež súčasťou prílohy každej časti normy.

SAMOHODNOTIACE OTÁZKY

- Na čo slúžia vzorové postupy umiestnené v každej časti normy **ISO 9241**?
- Vymenujte, z ktorých dvoch krokov pozostáva vzorový postup.
- Na čom by sa mal zakladať postup posudzovania?

	• Do ktorých kategórií môžeme rozdeliť postup posudzovania? • Od ktorých faktorov závisí použiteľnosť určitého odporúčania? • Vymenujte postupy vhodné na posúdenie použiteľnosti. • Vysvetlite pojem „posúdenie zhody“. • Vymenujte postupy na posúdenie zhody. • Na čo slúži kontrolný zoznam, ktorý je súčasťou prílohy každej normy **ISO 9241**? • Čo obsahuje kontrolný zoznam?

11 CONTROL WEB - UNIVERZÁLNY NÁSTROJ PRE VÝVOJ VIZUALIZAČNÝCH A RIADIACICH APLIKÁCIÍ

Automatizácia je v podvedomí širokej verejnosti spojená s elektrotechnickými odbormi, kde pomocou regulátorov a regulačnej techniky riadime rôzne systémy, spravidla bez zásahu človeka. Jej popularizácia sa prezentuje úsporami energie a mzdových prostriedkov, užívateľským komfortom, alebo vo forme ukážky a možnostiach robotiky.
Robotika vplyvom vývoja vysokého stupňa integrácie a objavením nových technológií riadenia a ovládania mechanického pohybu, najmä však nástupom umelej inteligencie priam šokuje širokú verejnosť, ktorá sa začala zaoberať myšlienkou, či roboty budúcnosti nahradia inteligenciu človeka a či to bude v prospech ľudstva, alebo sa tieto stroje môžu stať vážnou hrozbou pre život.
To, čo je laickej verejnosti menej známe je skutočnosť, že princípy automatizácie a kybernetiky sa dajú aplikovať aj na také odbory, ako sú medicína a ekonomika. Predstavte si nasadenie umelej inteligencie pri vývoji nových liekov a lekárskych postupov! O koľko by sa skrátilo utrpenie človeka. V ekonomike je tu možnosť posudzovať informácie z toľkých hľadísk, až sa to vymykajú ľudským možnostiam.
Vráťme sa však späť k základom automatizácie, aké sa vyučujú na stredných školách. Základom výučby je osvojiť si základné vedomosti o vstupoch do automatizovaného procesu, o snímačoch a prevodníkoch, o princípe prenosu signálu, princípoch regulácie a ich vplyvu na kvalitu regulácie a v neposlednom rade vyhodnocovaní výsledkov regulácie, bezpečnosťou a spoľahlivosťou regulovaného systému a jeho odolnosťou proti rušivým až kritickým situáciám.
Často sa však v predmetoch automatizácie prednášajú základy výpočtovej techniky. Takto pojatá výučba nahradzuje absenciu predmetu výpočtová technika a je prezentovaná ako základy automatizácie modernej kancelárie.
Učiteľská prax poukazuje na nízke prepojenie poznatkov rôznych predmetov. Študenti môžu vykazovať celkom slušné výsledky v každom predmete, no pri zadaní komplexnej úlohy nie sú schopní efektívne využiť svoje vedomosti. Stredoškolská odborná činnosť ako záujmová činnosť oslovuje iba niektorých študentov.
Zaujímavou aplikáciou a modifikáciou predmetu automatizácia je prepojenie s výpočtovou technikou. V praxi sú klasické regulačné systémy súčasnosti aplikované v pamäti počítačov. Výhody sú evidentné: ľahká modifikovateľnosť, náhrada drahých zariadení virtuálnymi prístrojmi, absencia starnutia mechanických a elektrických prvkov, zmenšenie rozmerov „velína“, radikálne zníženie náhradných dielov, ľahká diagnostika a tak ďalej.
Pri skúmaní možností oživiť výučbu automatizácie a pripraviť študentov na komplexné a kreatívne riešenie úloh je využitie profesionálneho systému Control Web 2000. Táto voľba nebola náhodná. Systém Control Web 2000 alebo jeho DOS verzia Control Panel vznikol v bývalej Československej republike vo firme Moravské přístroje Zlín. Dodávaná verzia je lokalizovaná v anglickom jazyku, nemčine, japončine a češtine. Česká verzia odstraňuje bariéru systém–študent. Český a slovenský jazyk je natoľko príbuzný, že študentom nerobí žiadne problémy.
Ďalším faktorom pri voľbe je jeho ľahká dostupnosť – demo verziu ako v DOS, tak i Windows verzii je možné získať buď na CD-ROM alebo na Internetovej adrese firmy Moravské přístroje www.mii.cz. Demo verziu nie je možné chápať ako „okliešteńu, funkčne výrazne obmedzenú verziu“ , ale ako plnohodnotný systém pre aplikácie riadenia procesov pomocou počítačov PC bez možnosti prepojenia cez vstupno-výstupnú kartu do reálneho sveta. Všetky ostatné funkcie vrátanie simulácie sú identické. Je teda možné vytvoriť profesionálny systém na demo verzii a na konkrétnom počítači, kde je ostrá verzia, tento

spustiť. Nemenej významným faktorom je kreatívny prístup pri programovaní nie len regulačného, ale aj technologického procesu.
Láskavosťou vedenia firmy Moravské přístroje je možné získať licenčné kódy pre demonštráciu školských aplikácií a tým vytvoriť reálne systémy, ktoré by mohli dominovať na Stredoškolskej odbornej činnosti.

12 POPIS SYSTÉMU CONTROL WEB

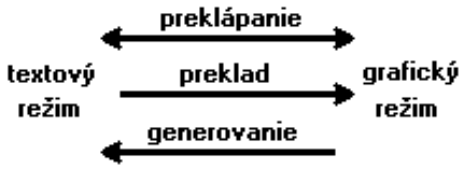
Control Web je univerzálny nástroj pre vývoj a nasadenie vizualizačných a riadiacich aplikácií, aplikácií zberu, ukladanie a vyhodnocovanie dát, aplikáciu rozhrania človek–stroj. Univerzálna, objektovo–orientovaná komponentová architektúra zaisťuje aplikáciám systému Control Web čo najširší rozsah nasadenia od prostých časovo nenáročných vizualizácií až po riadiace aplikácie reálneho času.
Virtuálne prístroje môžu pracovať nielen v sekvenčne riadených procesoch, ale aj ako prístroje reagujúce na udalosti. Komponenty systému nie sú fixne naprogramované. Existuje celý rad vizualizačných možností vrátane ďalších programátorských nástrojov. Vizualizácia procesu nie je obmedzená iba na konkrétny počítač, kde je program spustený. V možnostiach **Control Web** je vizualizácia na prepojených počítačoch, prípadne prostredníctvom internetových štandardov http a HTML prostredníctvom ľubovolného WWW klienta vytváranie plnohodnotného http servera dynamicky tvoriaceho stránky podľa stavu technológie.
Systém má neobyčajne rozsiahlu funkčnosť od vizualizácie po digitálne spracovanie signálu, od riadenia procesu po diaľkovú diagnostiku strojov po Internete.

12.1 Panely a virtuálne prístroje

Reálne velíny obsahujú celý rad prístrojov, kontroliek, tlačidiel a iných nastavovacích prvkov. Všetky tieto komponenty sú súčasťou nejakého panelu, napríklad panelu rozvádzača. Control Web 2000 umožňuje panely vytvárať na obrazovke buď ako statické, alebo dynamicky sa prekrývajúce s inými panelmi, schovávať sa, minimalizovať a opäť vystupovať do popredia. Do panelov je možné inštalovať virtuálne prístroje.
Virtuálne prístroje sú samostatné a na sebe nezávislé komponenty, no ak majú spolu tvoriť súdržný a výkonný aplikačný program, musia byť navzájom prepojiteľné. Základným prepojením virtuálnych prístrojov v aplikačnom programe je štruktúra časovania a štruktúra viditeľnosti. Štruktúra viditeľnosti určuje, kde sa na obrazovke bude prístroj nachádzať, a štruktúra časovania stanovuje, kedy a za akých podmienok bude prístroj aktivovaný.
Systém **Control Web** je zodpovedný len za stanovenie a dodržiavanie pravidiel pre také prepojenie. To, akú prístroj vyvinie činnosť pri svojej aktivácii, ako vyzerá na obrazovke a dokonca ako sa zapíše v zdrojovom texte aplikácie, je iba jeho vnútornou vecou.

12.2 Inteligentné vývojové prostredie

Kvalitu riešenia akéhokoľvek problému predpokladá isté počiatočné premyslenie, usporiadanie požiadaviek a postupné vytriedenie možných realizačných ciest. Táto fáza býva veľmi kreatívna a zábavná.
Control Web 2000 podporuje vytváranie projektov niekoľkými spôsobmi – pomocou textového režimu a grafickými nástrojmi. Oba spôsoby sú navzájom zameniteľné, pričom textový režim bol vybratý ako **zdrojový tvar aplikácie**. Tým je daná vysoká bezpečnosť a robustnosť aplikácie. Zmena režimu práce sa volá **preklápanie**, pričom prechodu z textového do grafického režimu sa hovorí **preklad** a prechodu z grafického do textového režimu **generovanie.**



Textový súbor môže obsahovať ľubovoľný text, teda aj zle napísaný program. Musí preto existovať mechanizmus, ktorý bezpečne rozlíši, či napísaný text systému vyhovuje. Takým mechanizmom je **preklad**. Prechod kontroluje formálnu správnosť textu a vytvára podľa neho binárnu podobu aplikácie. Ak prekladač zistí chybu, označí miesto chyby a ponúkne informácie o nedostatkoch textu. V žiadnom prípade chyby sám neopravuje! Takáto „automatická oprava“ alebo „ignorovanie príkazu“ by mohlo mať za následok zmenu aplikácie a tým možné porušenie bezpečnosti aplikácie. Preto oprava je zverená výhradne človeku.
Preklad môže prebiehať v niekoľkých režimoch – **preklad po častiach (inkrementálny).** Je nepríjemné, keď sa preklad zastaví niekde pred koncom zdrojového textu. Za normálnych okolností by po oprave bolo nutné celý text znovu prekladať. **Control Web** podporuje „inteligentnejšie“ riešenie. Zdrojový text sa prekladá prístroj po prístroji a ten, ktorý je bez chyby je zapamätaný. Pri budúcom preklade sa znovu neprekladá. **Jednoduchý preklad** je veľmi rýchly a kontroluje zdrojový text iba na syntaktickú správnosť. **Preklad pre preklopenie** zaisťuje prechod do grafického režimu. **Preklad pre spustenie aplikácie** kontroluje všetko a aplikácia musí byť „stopercentná“.
Generovanie je opak prekladu. Z grafickej podoby aplikácie sa generuje zdrojový text. Je teda možné, že znovu vygenerovaný text sa bude líšiť od predchádzajúceho.

12.3 Grafický editor

Aplikácie sú väčšinou vytvárané v grafickom režime, ktorý je ľahko ovládateľný pomocou myši. Programovanie aplikácie sa tak stáva veľmi kreatívne. Naviac dostávame ponuku sprievodcov, ktorý vám pomôžu na začiatku vývoja aplikácie pripraviť hlavné funkčné bloky.
Grafický editor obsahuje niekoľko logicky oddelených častí. **Dátoví inšpektori** zaisťujú napojenie aplikácie na reálny svet, nahradzujú „vodiče“ na prepojenie prístrojov a ďalšie možnosti, o ktorých si povieme neskoršie. **Paleta prístrojov** je hlavná časť grafického editora. Pomocou nej vyberáme prístroje a umiestňujeme ich na **plochu grafického editora.** Každý takto umiestnený objekt je možné zmeniť a upraviť pomocou **Inšpektora prístroja.**
Na tejto úrovni prerušíme teoretický výklad a pokúsime sa vytvoriť vlastný, jednoduchý projekt. Predpokladom úspechu je minimálna znalosť ovládania operačného systému Windows.

13 NESPOJITÉ PRÍSTROJE

13.1 Začíname projektovať

Spustite **Control Web 2000** a vytvorte novú aplikáciu buď cez roletové menu *Soubor – Nový* (analógia kombinácie kláves *Ctrl N*) alebo pomocou ikony „čistého listu papiera“ vľavo hore. Systém ponúkne v dialógovém okne sprievodcu novou aplikáciou. To je ďalšia, význačná vlastnosť systému Control Web. Programátor je odbremenený od rutinného programovania a vedený k ľahkému vytvoreniu základnej štruktúry aplikácie. No my sa touto cestou nevydáme a klikneme na kláves *Zrušiť*. K iným možnostiam sa vrátime až po zvládnutí základov projektovania. Ak nie ste, prepnite sa do režimu ***Grafický editor***. Túto voľbu nájdete na záložke v dolnej časti obrazovky.

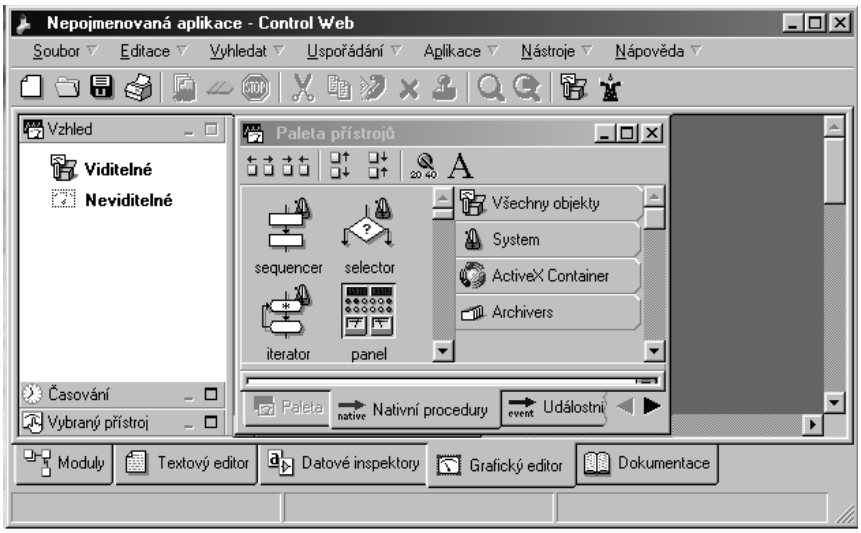
13.2 Úloha č.1

AKTIVITA

Vytvorte aplikáciu, kde na paneli bude umiestnený vypínač a žiarovka, ktorá bude „svietiť“ pri zapnutí vypínača. Vytvorte niekoľko variantov typov vypínačov, niekoľko typov „žiaroviek“ a rôzne varianty panela.

Riešenie – vytvorenie prístroja Panel

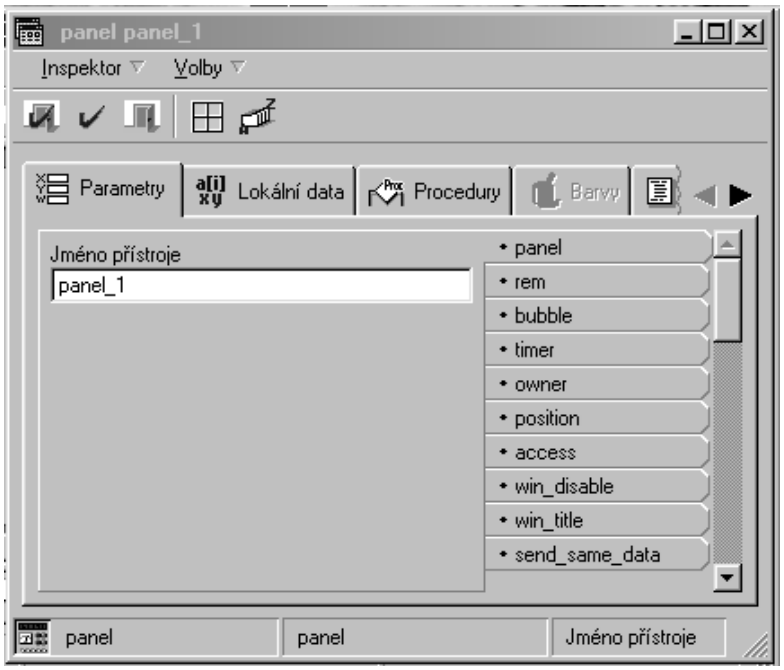
Prvým krokom bude vytvoriť na editačnej ploche **panel**. Prístroj *panel* je určený na zhromažďovanie viacerých prístrojov do jedného objektu, ktorý môže byť zobrazovaný, skrývaný alebo minimalizovaný. V našom prípade v ňom budú neskôr umiestnené prístroje vypínač a žiarovka.



Na nástrojovej lište kliknite na „Paletu prístrojov“. Tú prezentuje ikona s meracím prístrojom a krabicou. Druhý variant je cez menu *Nástroje – Paleta prístrojov*. Vyberte záložku *Všetky objekty* alebo *Systém*. Tieto záložky obsahujú okrem iných objektov aj prístroj *panel*.

Pomocou myši uchopte prístroj *panel* a preneste na pracovnú plochu. Paletu prístrojov zrušte kliknutím na krížik. Všimnite si, že novo vytvorený objekt je zaznamenaný aj do štruktúry viditeľnosti v okne **Vzhľad** ako *panel_1*.
Na pracovnej ploche sa vytvoril ohraničený obdĺžnik s úchopovými bodmi – panel je zobrazený a vybratý. Pomocou úchopových bodov nastavte pozíciu a rozmery panelu. Môžete experimentovať – polohu a rozmer panela môžete kedykoľvek upraviť. Pokiaľ sa stane, že úchopové body zmiznú, a to sa stane vtedy, keď kliknete mimo ohraničenú časť vybratého objektu, panel nie je vybratý. Ak však chcete panel opäť vybrať, kliknite na ľubovoľné miesto na objekte. Ďalšia možnosť vybratia panela je kliknite na jeho názov v štruktúre viditeľnosti. Panel, ako iný objekt, nemá konečnú podobu danou vývojovým prostredím. Projektantom je umožnený celý rad ako vizualizačných, tak funkčných zásahov do štruktúry virtuálneho prístroja. Cez pravý kláves myši, ak jej kurzor sa nachádza v objekte panel, a voľbou *Inšpektor prístroja* je umožnené zmeniť jeho názov, vzhľad, veľkosť či pozíciu, zadať podmienky ponorenia, vynorenia, nastavenie pozadia alebo iné vlastnosti dané možnosťami prístroja. Tak isto je možné *Inšpektor prístroja* vybrať pravým klávesom myši cez názov v štruktúre viditeľnosti ak na neho ukážeme.

Inšpektor prístroja
Inšpektor prístroja panel_1 obsahuje horizontálne a vertikálne záložky. Na horizontálnych nastavte záložku Parametre. Vertikálne záložky obsahujú niektoré štandardné a niektoré špecifické parametre prístroja. Každý parameter je reprezentovaný záložkou. Ak ju nevidíte, použite posuvník.



Medzi **štandardné parametre** patrí:
panel - štandardne sa doplní názov prístroja s poradovým číslom. Zmeňte meno prístroja na *Skúšobný_panel*. Upozorňujem, že v názve sa nesmú vyskytnúť medzery!
rem – poznámka, ktorá opisuje význam prístroja v aplikácii. Má iba dokumentačný charakter.
bubble – text v bublinovej nápovedi. Bublinová nápoveda sa zobrazí vtedy, ak v priebehu aplikácie zastanete na prístroji kurzorom myši.
timer – spôsob časovania prístroja. Môže obsahovať číselnú hodnotu udávajúcu periódu časovania v sekundách, alebo meno vlastníka v časovej štruktúre. Ak nie je *timer* uvedený,

nie je prístroj časovaný v rámci časovej štruktúry, no môže byť aktivovaný inými prístrojmi. Zadajte *timer = none*. Čierny trojuholník vpravo na konci editačného okna umožňuje voľbu niektorých prednastavených hodnôt, no je možné priamo písať do okna. **Offset** je hodnota posunutia mechanizmu časovania o patričný interval vpred. Posunutie je odvodené od polnoci, ktorá už prebehla. Napríklad *timer=60, offset=20* znamená, že prvý takt časovača bude 20 sekund po prebehnutej polnoci a ďalší periodicky po minúte.
owner obsahuje meno vlastníka vo vizuálnej štruktúre, ktorým môže byť meno panelu alebo kľúčové slovo *background*, pokiaľ má byť prístroj priamo na pozadí. Ak nie je parameter *owner* uvedený, prístroj je neviditeľný. Náš panel bude umiestnený priam na pozadí, preto *owner = background*.
position obsahuje u prístrojov s pevnou veľkosťou dve a u prístrojov s definovateľnou veľkosťou štyri číselné hodnoty, ktoré udávajú umiestnenie a poprípade i rozmery prístroja. Nastavenie veľkosti už poznáme pomocou myši, no ak je potrebné nastaviť presné hodnoty, použijeme tieto parametre.
access – obsahuje číslo v rozsahu 0 až 4294967295 vyjadrujúcu úroveň prístupových práv. V našom prípade bude *access = none*.
win_disable – obsahuje zoznam zakázaných vlastností okna. Tento parameter má význam iba ak je prístroj v okne. Zoznam je tvorený nasledujúcimi kľúčovými slovami

move	zákaz pohybu
zoom	zákaz zmeny veľkosti
minimize	zákaz minimalizácie
maximize	zákaz maximalizácie
lower	zákaz ponorenia

Náš panel nebude mať žiadne obmedzenie, preto nebude žiadne okienko prečiarknuté.
win_tittle – ukazuje text, ktorý bude zobrazený v titulku okna prístroja. Ak nie je parameter uvedený, do titulku okna sa prenesie meno prístroja. Tento parameter má význam iba v prípade, ak je prístroj v okne. Náš panel je dostatočne popísaný v položke *meno prístroja*, preto položku nevyplňujte.

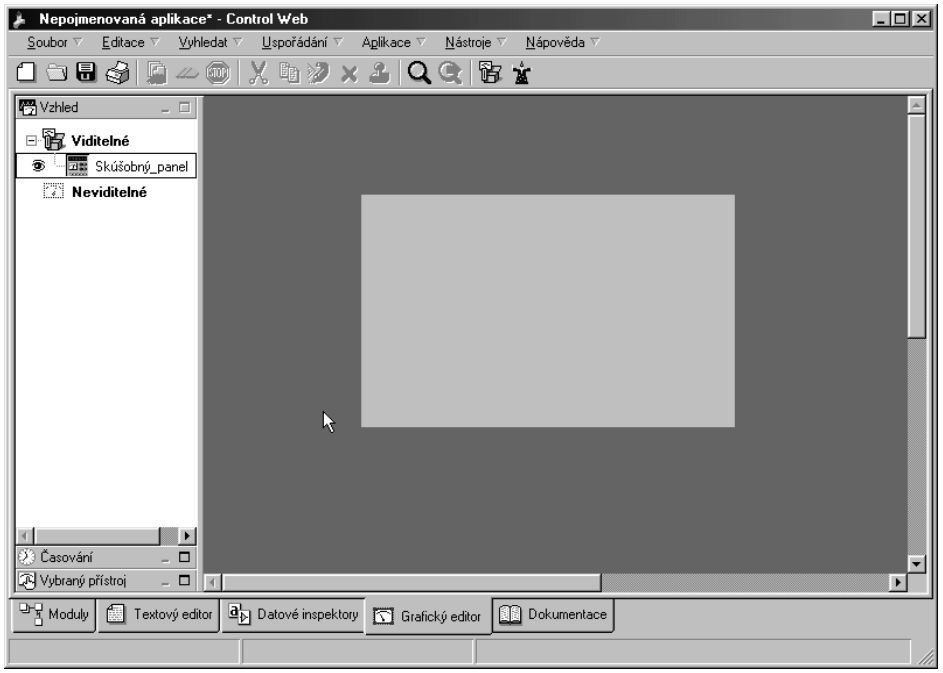
send_same_data – príznak udávajúci nutnosť behom aktivity prístroja zapisovať do výstupného kanálu hodnoty i v prípade ich zhodnosti so súčasným stavom. Údaj v našom paneli nevyplňujeme.
tab_select – obsahuje poradie výberu prístroja v rámci panelu. Tento parameter umožňuje meniť výber prístroja pomocou klávesy *Tab* prípadne *Shift Tab*. Tento parameter prístroj *panel* neobsahuje.

Medzi **špecifické parametre** prístroja patria parametre minimalizácia, vynorenie a ponorenie, viditeľnosť, farba a obrázok pozadia. Zaujímavým parametrom prístroja *panel* je **item**. Položka *item* dovoľuje zadať v rámci panelu posupnosť aktívnych obdĺžnikov citlivých na kliknutie myšou. Každá položka môže obsahovať dve kľúčové slová – **rect** – súradnice obdĺžnika a **output** – premenná typu boolean spojená s daným obdĺžnikom. Náš panel nebude obsahovať žiadnu položku *item*.

Panel môže nadobudnúť tvar okna vo Windows ak klikneme na ikonu *okno* v hornej časti inšpektora. Túto možnosť si overíme v rôznych modifikáciách našej úlohy.

Pokiaľ ste previedli požadované zmeny do štruktúry prístroja, je potrebné dialógové **okno *Inšpektor* zatvoriť. Ikona realizujúca operáciu zápis parametrov a zatvorenie má podobu prečiarknutých dverí. Výsledok vašej práce by mal byť podobný tomuto obrázku.**

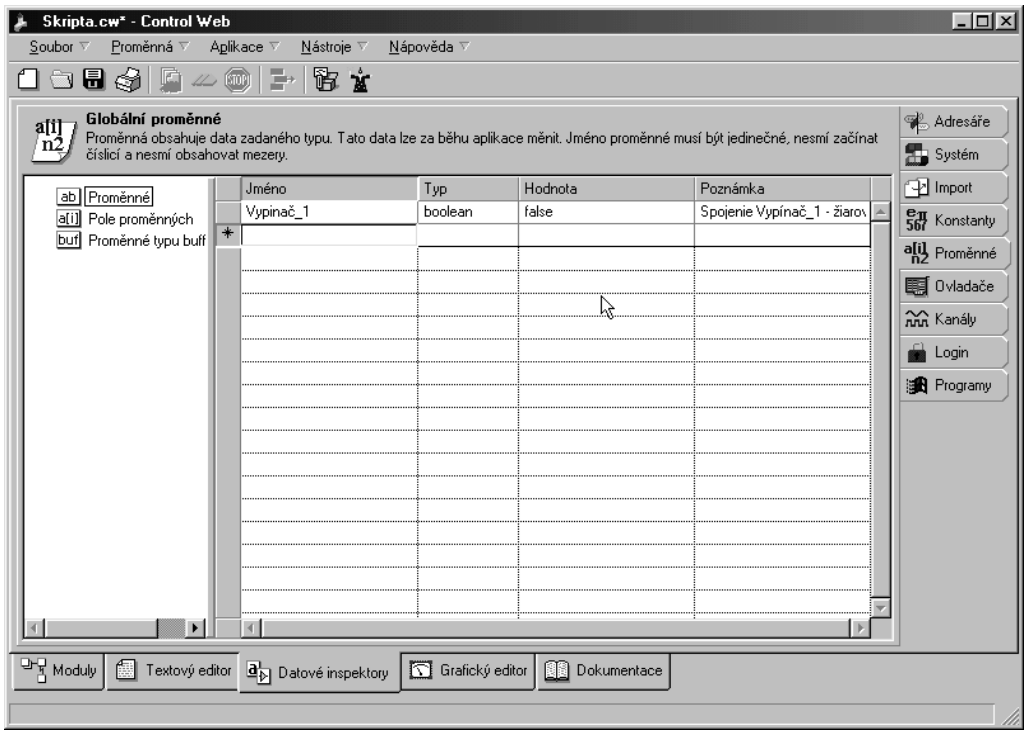
V tejto fáze vytvárania aplikácie ste vytvorili panel, do ktorého budete umiestňovať prístroje.



Dátové inšpektori
Na realizáciu skutočného zapojenia panela je potrebný okrem vypínača a žiarovky ešte vodič. V systéme **Control Web 2000** sa prepojenie prístrojov nerealizuje vodičmi, ale premennými. Preto ďalším krokom bude vytvoriť premennú, ktorá prenesie signál z vypínača do žiarovky.
Inteligentné vývojové prostredie podporuje vytváranie premenných v kapitole Dátoví inšpektori, prístupné cez záložku v dolnej časti obrazovky. Dátoví inšpektori obsahujú niekoľko záložiek, o ich funkcii budeme hovoriť neskôr. Nás zaujíma záložka *Premenné*.
V ľavom okne zvolíme jednoduché *Premenné*. Pole premenných ani premenné typu buffer zatiaľ nebudeme naplňovať.

Kliknutím do prázdneho riadku položky *Meno* vložte názov premenné (nesmie obsahovať medzery!), napríklad *Vypínač_1*. Každej premennej ktorú definujete je nutné dať *Typ*. Keďže našou úlohou bude zažínať a zhasínať žiarovku, naša premenná ponesie signál „zopnutý – vypnutý“, teda dvojhodnotovú informáciu, preto zvolíme typ *boolean*. *Hodnota* je stav, ktorý sa dosadí do premennej v okamihu štartu aplikácie. Vyberte *false* – vodič je bez signálu. Aby ste mali prehľad o použitých premenných, vyplňte položku *Poznámka* textom, k čomu bude premenná slúžiť, napr. *Spojenie vypínač_1 – žiarovka_1*.

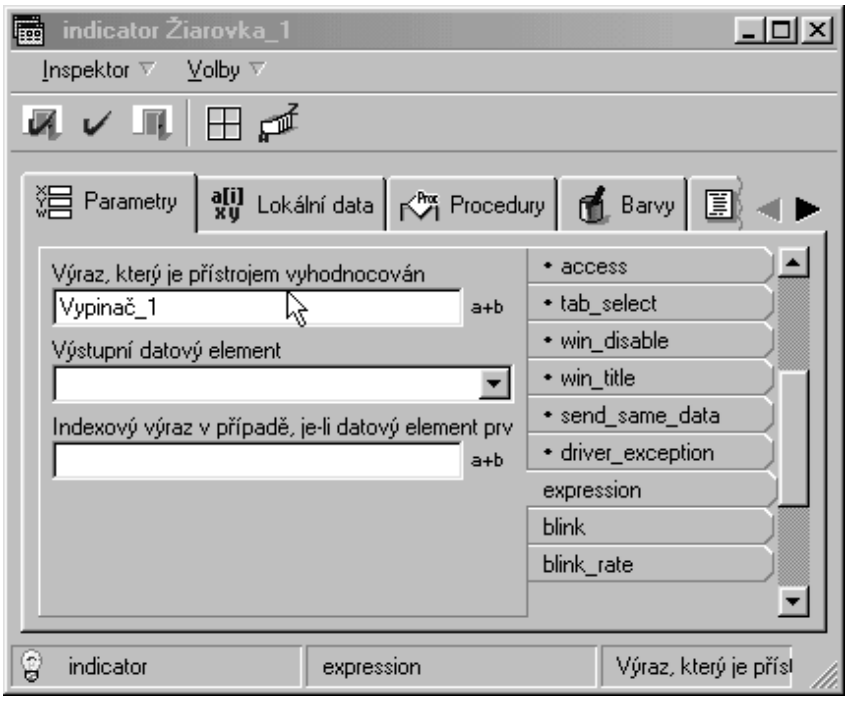
Tým práca v dátových inšpektoroch skončila.



Riešenie – vytvorenie prístroja Žiarovka

Prepnite sa znova do grafického editora. Kto zabudol, nech si pozrie kapitolu 3. Znovu otvorte Paletu prístrojov. V záložke *Všetky objekty,* alebo *Main* sa nachádza prístroj *Indicator*, reprezentovaný ikonou žiarovka. Presuňte tento prístroj do vytvoreného panela.
Prístroj **Indicator** zobrazuje výsledky logického výrazu a jeho prípadné priradenie do premennej alebo výstupného kanálu. Interpretačné výrazy tohoto prístroja sa nastavujú v Inšpektorovi *prístroja* (3.1.2.). Pre našu úlohu sú zaujímavé parametre
indicator = Žiarovka_1 (názov prístroja)
timer = none (zatiaľ nebudeme časovať)
owner = Skúšobný_panel. Vlastníkom prístroja bude panel, podobne ako by sme žiarovku priskrutkovali.
expression – do poľa *Výraz, ktorý je prístrojom vyhodnocovaný* sa píše logický výraz, ktorého hodnotu bude prístroj zobrazovať. Žiarovka bude závislá na polohe jedného vypínača, to znamená na hodnote jednej logickej premennej, ktorou je *Vypínač_1*. Zápis tejto premennej je možné priamo cez klávesnicu, je však potrebné dodržať presne názov premennej spolu s veľkými a malými písmenami. My sa však pozrieme na lepšiu možnosť zápisu pomocou **Editora výrazov**.

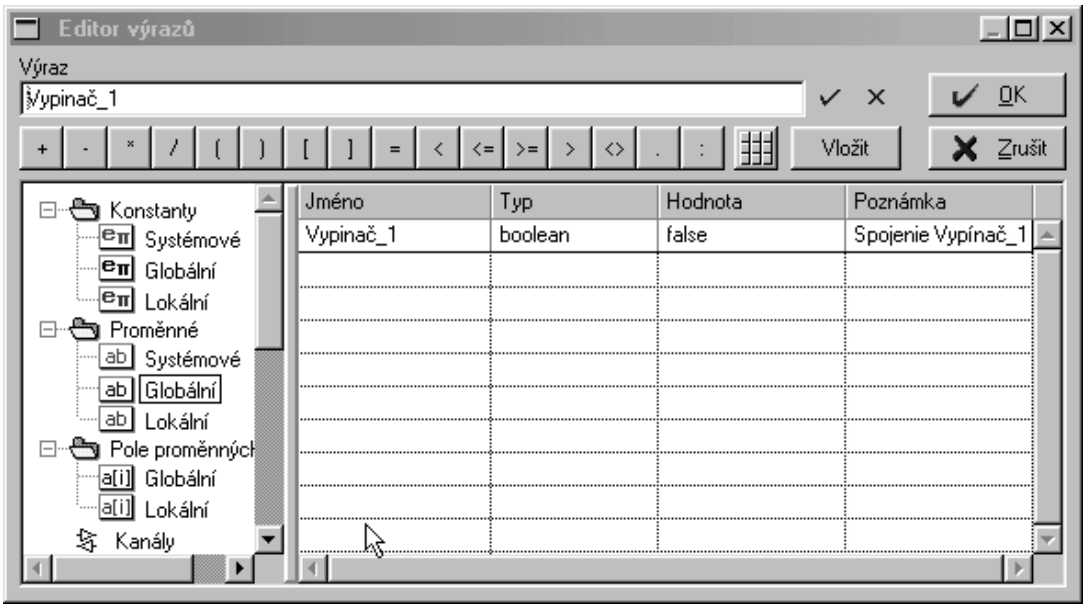
Inšpektor prístroja zatvoríme (ako inak než kliknutím na prečiarknuté dvere).



Editor výrazov

Iba so samostatnými dátovými elementami v aplikačných programoch ťažko vystačíme. Často je potrebné vyriešiť matematický výraz a výsledkom riadiť ďalší beh programu. V oblastiach matematických výrazov je systém **Control Web 2000** veľmi silný.
Výrazy môžeme podľa typu výsledku rozdeliť na tri typy:
1.číselné výrazy – výsledkom je číselná hodnota typu *real* a môžu obsahovať číselné typy, aritmetické operátory a všetky matematické, prevodné, textové a bitové funkcie, ktorých výsledkom je číslo
2.logické výrazy – výsledkom je logická hodnota typu *boolean* a môžu obsahovať logické typy, logické operátory a relácie medzi číselnými podvýrazmi
3.textové výrazy – výsledkom je textová hodnota typu *string* a môžu obsahovať dátové funkcie s výsledkom *string*

Pre maximálne uľahčenie pri zostavovaní výrazov je grafické vývojové prostredie vybavené editorom výrazov, ktorý umožní akýkoľvek zložitý výraz „napísať“ myšou. Editor vyvoláme z Inšpektora prístrojov klepnutím na kláves **a+b**, ktorý je umiestnený vždy napravo vedľa editačného riadku s výrazom.
Editor výrazov sa skladá
- z výstupného okna *Výraz*
- z poľa základných aritmetických a porovnávacích kláves a kláves pre zobrazenie číselnej klávesnice
- z okna pre výber množiny konštánt, premenných, kanálov, funkcií, operátorov, atribútov a modulov zo stromovej štruktúry
- z okna hodnôt príslušnej množiny



Ak napríklad potrebujeme vybrať globálnu premennú *Vypínač_1*, klikneme v stromovej štruktúre na *Premenné – Globálne*, vyberieme požadovanú premennú a stačíme *Vložiť* (alebo prevedieme dvojklik na premennej). Premenná sa prepíše do výstupného okna.

Ak potrebujeme vytvárať matematické operácie alebo vyberať logické a relačné operátory, vyberieme najskôr požadovanú oblasť v stromovej štruktúre a potom príslušnú funkciu alebo operátor.

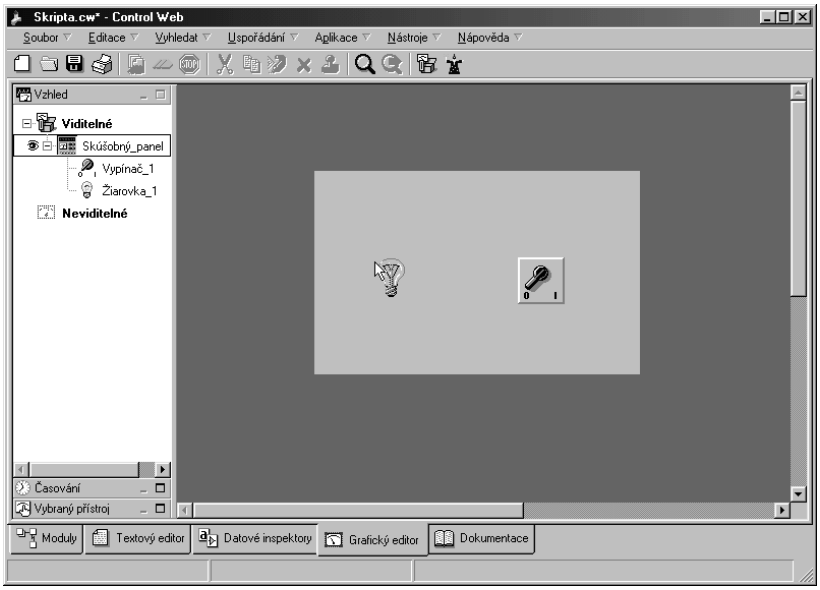
Zostavený výraz odošleme klávesom *OK*.

Riešenie – vytvorenie prístroja Vypínač_1

V grafickom režime otvorte Paletu prístrojov, v záložke *Všetky objekty,* alebo *Main* nájdite prístroj *switch*, reprezentovaný ikonou vypínač. Presunte tento prístroj do vytvoreného panela.

Prístroj **Switch** nastavuje logické hodnoty do premennej alebo výstupného kanálu typu boolean pomocou klávesy. V *Inšpektori prístroja* (3.1.2.) sú zaujímavé tieto parametre
switch = Vypínač_1 (názov prístroja)
timer = none (zatiaľ nebudeme časovať)
owner = Skúšobný_panel. Vlastníkom prístroja bude panel, podobne ako u žiarovky.
output – udáva názov premennej alebo výstupného kanálu, kde sa bude hodnota nastavená prístrojom switch posielať. Naša premenná sa volá *Vypínač_1*. V poli *Výstupný dátový element* namiesto priameho vpisovania hodnoty použime roztváraciu ponuku (čierny trojuholník), a premennú vyberieme pomocou myši. *, ktorý je prístrojom vyhodnocovaný.* Ak ju v ponuke nenájdeme, s najväčšou pravdepodobnosťou ju nemáme vytvorenú v Dátových inšpektoroch (3.1.3) alebo premenná nemá typ boolean.
receivers – obsahuje zoznam všetkých prístrojov, ktorým je posielaná správa pri zmenách stavu prístroja. Tu sa zastavme a vysvetlime si niektoré pojmy. Prístroje v **Control Web 2000** môžu pracovať buď v dátovo riadenom režime alebo v niektorom z časových režimov. Dátovo riadený režim informuje všetkých príjemcov až v okamihu zmeny stavu, teda podobne, ako keď stlačíme vypínač na stene. Časovo riadené režimy periodicky „ohmatávajú“ prístroje a vypytujú sa, v akom stave sa nachádzajú. A podľa toho potom reagujú. Časový režim zavádza do aplikácií synchronizáciu. Táto periodicita však niečo stojí – počítač sa zaoberá skúmaním v akom stave sa nachádzajú prístroje. A to stojí strojový čas o ktorý budú ukrátené ostatné súbežne spustené programy. Predstavme si, že by sme chodili neustále

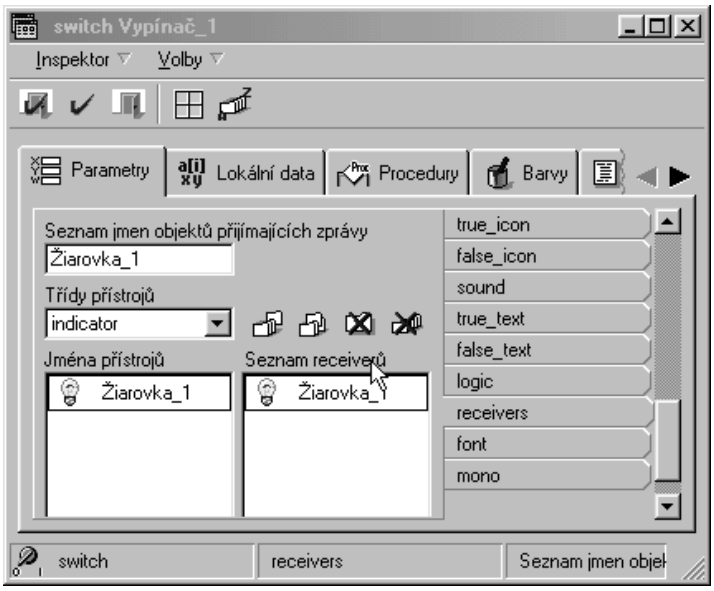
skúmať, či je vypínač osvetlenia v miestnosti zapnutý namiesto toho, že si vypočujeme jeho cvaknutie a pritom robíme niečo iné. Preto, ak to nie je skutočne opodstatnené, vždy volíme režim dátovo riadený! Samozrejme pri rozsiahlych prácach, alebo v prípade, že chcete mať prístroje pod kontrolou, sa časovaniu nevyhneme a dokonca nám uľahčí veľa práce. Ako na udalosťou riadené procesy, to sa dozviete v kapitole 3.1.5.1.



Pokiaľ sú parametre nastavené, zatvorte Inšpektor prístroja.

Nastavenie receivers

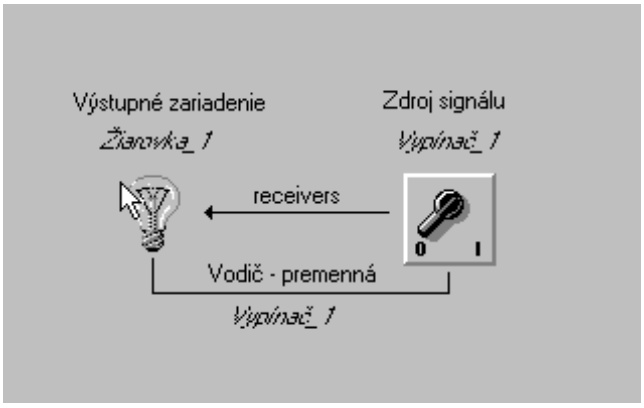
Panel *receivers* predovšetkým zobrazuje *Triedy prístrojov*. Rozbalením ponuky môžeme vybrať triedu prístrojov, do ktorej spadá náš výstupný člen – *Žiarovka_1*. Táto trieda sa nazýva Indicator. V okne *Mena prístrojov* vyberieme konkrétny prístroj, ktorý bude prijímať správu o zmene stavu vypínača. Dvojklikom na jeho meno alebo kliknutím na ikonku Pridať prepíšeme prístroj do pravého okna. Pokiaľ by vypínač ovládal ďalšie prístroje, rovnakým spôsobom ich prepíšeme do pravého okna.



A to je všetko. Nezabudnite uložiť zmenu cez ikonu *Použiť a uzatvoriť*, inak by bola neplatná.

Zhrnutie

Na obrázku je principiálna schéma, aká bolo naprogramovaná v úlohe č.1.



Na obrazovke sú tri nečasované prístroje (timer=none)
– panel *Skúšobný panel*, vlastníkom je pozadie (owner=background)
– indicator *Žiarovka_1*, vlastníkom je *Skúšobný panel a* vyhodnocuje premennú *Vypínač_1*(expression)
– switch *Vypínač_1*, vlastníkom je *Skúšobný panel*, vysiela zmenu stavu do premennej *Vypínač_1*(output), ktorý o zmene svojho stavu informuje prístroj *Žiarovka_1* (receivers).

Ak projekt úspešne odladíme a spustíme (3.1.7.), bude pracovať nasledovne:
- kliknutím na vypínač sa prístroj *Vypínač_1* aktivuje, prestaví polohu páčky na I, vyšle zmenu stavu do premennej *Vypínač_1* a upozorní prístroj *Žiarovka_1*, že sa udiala zmena
- prístroj *Žiarovka_1* sa udalosťou aktivuje a vyhodnotí stav premennej *Vypínač_1*. Ak je v aktívnom stavu (true), žiarovka zmení vzhľad – „rozsvieti sa“.
- Ďalším kliknutím na vypínač dôjde k prestaveniu páčky na O, zmení sa hodnota premennej na neaktívny stav (false) a udalosťou sa opäť aktivuje žiarovka, ktorá stav vyhodnotí. Žiarovka „zhasne“.

Projekt pracuje na základe zmeny udalostí, v našom prípade na zmene stavu vypínača.

Spustenie a ukončenie aplikácie

Ak sú parametre prístrojov nastavené, môžeme pristúpiť k spusteniu aplikácie z grafického editora. Skúsení užívatelia Windows vedia, že teraz je vhodná chvíľa na uloženiu súboru.

Aplikácia sa spúšťa ikonou *Spustiť aplikáciu* na nástrojovej lište. Pred spustením sa prevedie kompilácia, a pokiaľ je všetko v poriadku, spustí sa. Vývojové prostredie sa odloží do panelu úloh v dolnej časti obrazovky.
Ak chceme aplikáciu ukončiť, obnovíme vývojové prostredie z panelu úloh a voľbou ikony Zastaviť aplikáciu na nástrojovej lište aplikáciu ukončíme. Inak beží do resetu počítača!
Ak pri preklade aplikácie vznikne chyba, vývojové prostredie sa prepne do textového režimu a systém oznámi výskyt chyby. Bez opravy nie je možné prejsť späť do grafického režimu.

Vyskúšajte si prechod do textového editora pomocou záložky v spodnej časti obrazovky. Uvidíte textové moduly, ktoré opisujú jednotlivé prístroje a premenné v úlohe. No, pokiaľ nezískate viac skúseností, neprepisujte ich. Práca v textovom režime je oveľa náročnejšia ako v grafickom a chybný program nie je možné preložiť do grafického režimu.

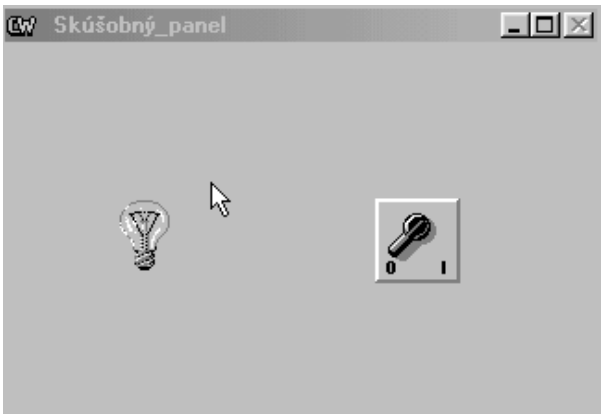
```
const
end_const;

var
  Vypínač_1 = boolean, false, 'Spojenie Vypínač_1 – žiarovka_1';
end_var;

driver
end_driver;

channel
end_channel;

timer
end_timer;

instrument

  panel Skúšobný_panel;
    owner = background;
    position = 135, 85, 320, 200;
  end_panel;

  switch Vypínač_1;
    owner = Skúšobný_panel;
    position = 200, 85, 46, 46;
    output = Vypínač_1;
    receivers = Žiarovka_1;
  end_switch;

  indicator Žiarovka_1;
    owner = Skúšobný_panel;
    position = 55, 85;
    expression = Vypínač_1;
  end_indicator;

end_instrument;
```

Modifikácia úlohy č.1

V tejto kapitole si na modifikáciách úlohy č.1 ukážeme vizualizačné možnosti prístrojov panel, indicator a switch. Ďalej si ukážeme použitie časovania prístrojov. Budeme sa zatiaľ pohybovať iba v grafickom editori.

Modifikácia prístroja *panel*
Prístroj *panel* má veľké množstvo výrazových prostriedkov a modifikácií. Uváženou kombináciou je možné vytvoriť skutočne realistické prostredie. Na tomto mieste uvedieme základné manipulácie s panelom, ktoré poslúži našej úlohe. Ďalšie možnosti budú demonštrované v iných úlohách.

Panel v okne
Panel je možné umiestniť do okna a ako s oknom vo Windows s ním manipulovať. Ako to môžeme dosiahnuť?

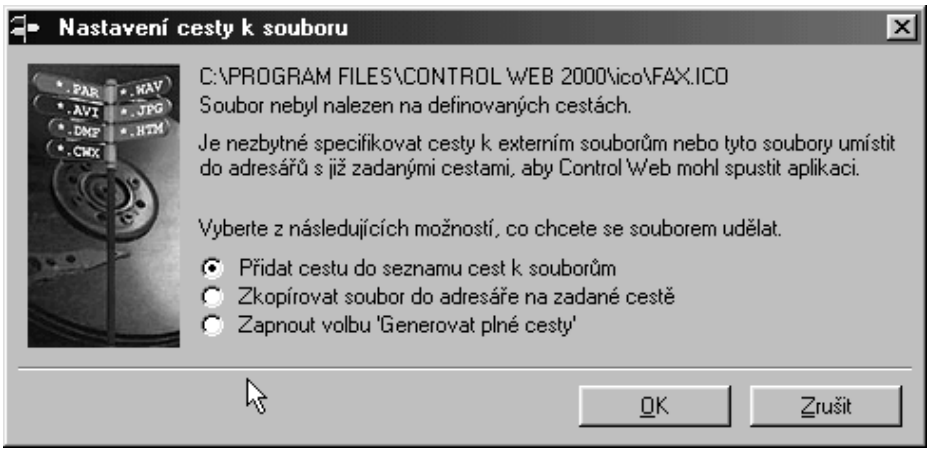


Vyberte panel (kto zabudol, tak kliknutím na neho), spustite Inšpektor prístroja (pravý kláves myši). Všimnite si ikony *okna* v nástrojovej lište. Ak na ňu kliknete, vytvorili ste nastavenie. Uložte zmenu (prečiarknuté dvere) a program spustite. Vyskúšajte si manipuláciu s oknom – úprava výšky, šírky, polohy na ploche, minimalizáciu, maximalizáciu. To, čo sa vám nepodarí, je zatvorenie okna.
Teraz ukončite beh aplikácie (obnoviť systém z nástrojovej lišty a kliknúť na ikonu *Stop*).

Obmedzenie pohybu panela
Komu by vadila niektorá vlastnosť okna, môže ju potlačiť. Ako? V inšpektorovi prístroja vyberte záložku *win_disable*. Prešktrnutím okienka potlačíme pohyb, veľkosť, minimalizáciu, maximalizáciu a ponorenie v ľubovolnej kombinácii. Vyskúšajte.

Zmena názvu panela v okne
V zložitejších aplikáciách môže byť niekoľko panelov, ktorým priraďujeme názov podľa určitej schémy. Ak tento názov nie je ten pravý, ktorý by sa mal zobraziť v záhlaví okna (vieme, že v názvu prístroja sa nesmie vyskytovať medzera), vyberte v inšpektorovi záložku *win_title* a napíšte text záhlavia.

Nastavenie pozadia panela
Panelu je možné priradiť i inú farbu pozadia. Zmeníte ju v záložke *color*.



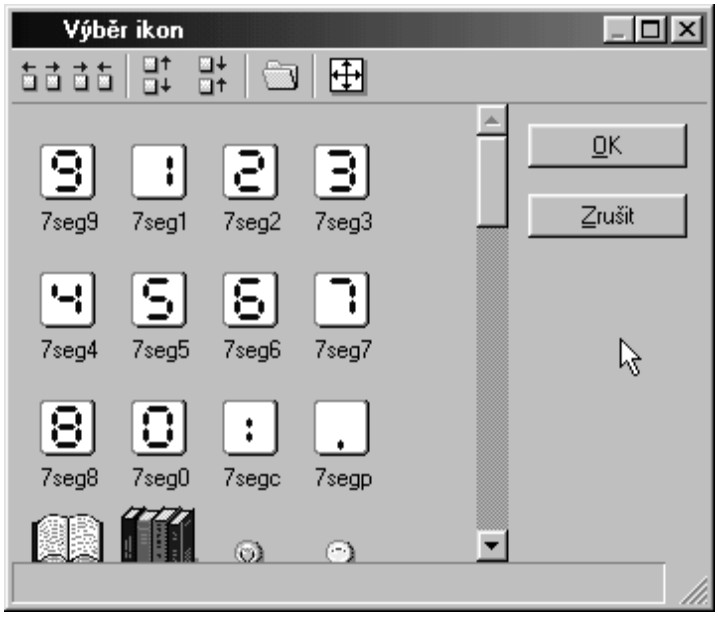
Pre realistické aplikácie je možné na pozadie panelu umiestniť obrázok uložený v súbore. V záložke *dv_id* volíme typ grafického súboru a v záložke *dv_file* názov súboru. Pokiaľ súbor nie je v implicitnom adresári, pridáme cestu do zoznamu ciest k súborom. To zaručí, že aplikácia bezpečne grafický súbor nájde. Všimnite si, že ak manipulujeme s rozmermi panelu v bežiacej aplikácii, mení sa rozmer pozadia – zmenšuje a zväčšuje sa a rôzne deformuje pri zmene rozmeru v jednom smere.
V aplikáciách, kde pozadie panelu vyjadruje reálne prostredie, je možné veľkosť pozadia panelu zachovať v pôvodnom rozmere. Voľbou záložky *container* a prečiarknutím okna *DataView v kontajneri* vytvoríme rolovacie lišty dobre známe z Windows.
Ak na realistický obraz pozadia panelu nadväzujú prístroje umiestnené v okne, potom by bolo logické, aby sa pohybom rolovacích líšt pohybovali aj tieto prístroje. To zabezpečíme, keď v záložke *follow_scrollbars* prečiarkneme okno *Prístroje budú sledovať pohyb rolovacích líšt*.

Zavedenie obrázka pozadia do panelu pri spustení aplikácie môže byť časovo náročné. Najmä v aplikáciách, kde sú panely skryté, predstavuje tento krok zbytočnú stratu operačného výkonu. V záložke *load_on_show* je okno *Obsah DataView zavedený až pri zobrazení*. Prečiarknutie spôsobí zavedenie pozadia až v okamihu zobrazenia panelu, napríklad jeho vyplávanie na povrch.

Modifikácia prístroja *indicator*
Prístroj indicator patrí medzi skupinu prístrojov, ktoré zobrazujú stav premennej alebo logického výrazu. Musí preto obsahovať rad vlastností, ktoré priblížia jeho tvar realite. Medzi základné vlastnosti patrí zmena vzhľadu prístroja.

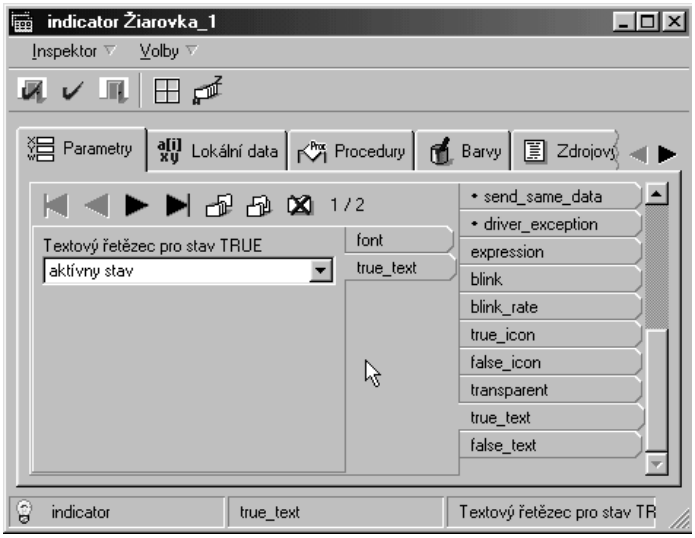
***Indicator* bez pozadia**
Klasické zobrazenie prístroja pomocou ikony je spojené s pozadím, ktoré ikonu obklopuje. Táto vlastnosť môže byť rušivá v prípade, že vlastník tohto prístroja má na pozadí obrázok.
Ak chceme potlačiť zobrazenia okolia ikony, vyberieme v *Inšpektori prístroja* záložku *transparent* a prečiarkneme okno *Objekt bez pozadia.* V aplikácii bude zobrazená iba ikona s pozadím vlastníka.

Zmena ikony prístroja *indicator*
Presunutím prístroja *indicator* z Palety prístrojov na plochu sa zavedie štandardná ikona zobrazujúca žiarovku. No ak naše požiadavky na vzhľad projektu vyžadujú iný tvar ikony, môžeme ju vymeniť za jeden zo štandardných tvarov alebo vytvoriť ikonu vlastnú. Zmena vzhľadu prístroja však vyžaduje, aby sme definovali jednu ikonu pre stav neaktívneho stavu (zhasnutá) a druhú pre aktívny stav (rozsvietená).



Definovanie nového vzhľadu prístroja zabezpečujú záložky *true_icon* (pre aktívny stav) a *false_icon* (pre neaktívny stav). V oboch prípadoch naplňujeme *Cestu k súboru pre stav true* (*false*). No cestu nevyplňujeme priamo cez klávesnicu, ale použijeme kláves vpravo od dialógového okna – *Výber ikon*. Je možné vybrať akúkoľvek ikonu pre aktívny a neaktívny stav. Zvážte, či je výhodné napríklad pre aktívny stav voliť ikonu *clock* a neaktívnu *fax*.
Ak vašim požiadavkám nevystačí paleta ikon, môžete si ju vytvoriť sami. **Control Web** má na túto operáciu Editor ikon, ktorý nájdete cez kláves *Štart* vľavo dole, potom *Programy, Control Web 2000 Demo* a *Editor ikon*. Ovládanie editora je ľahké, podobné ovládanie programu Skicár vo Windows. Ak ikony uložíte do adresára Program Files\Control Web 2000 Demo\Ico, ponúknu sa vám vo *Výberu ikon*.
Keď máte ikony uložené v inom adresári, je možné priamo napísať cestu a názov *ICO* súboru. Ručný zápis môže byť ale zdrojom chýb.

Výmena ikony prístroja *indicator* za text
Niektoré aplikácie požadujú, aby bol operátor upozorňovaný na zmenu stavu textovou správou. Aj na túto možnosť je prístroj *indicator* pripravený. V *Inšpektorovi prístroja* vyberte *true_text* pre oznam aktívneho stavu a *false_text* pre oznam neaktívneho stavu. Ak je vybraná záložka *font*, môžeme pomocou klávesu *Abc* špecifikovať veľkosť, rez a font písma. Cez voľbu záložky *true_text* píšeme vlastný text.
Pokiaľ požadujeme, aby bol text napísaný na viacej riadkov, je nutné zaradiť ich do zoznamu. K tomu slúžia tri tlačidlá nad písaným textom. Prvé z nich znamená *Pridať* a pridáva vetu na koniec zoznamu. Druhá znamená *Vložiť* a zaraďuje vetu na aktuálne miesto zoznamu. Tretia vymaže aktuálnu vetu. Čísla vpravo od tlačidiel, napríklad 1/3, znamená, že máme v zozname tri položky a aktuálne nastavená je prvá. Význam šípok vľavo slúži na posun v zoznamu – na začiatok, vľavo, vpravo, na koniec zoznamu.



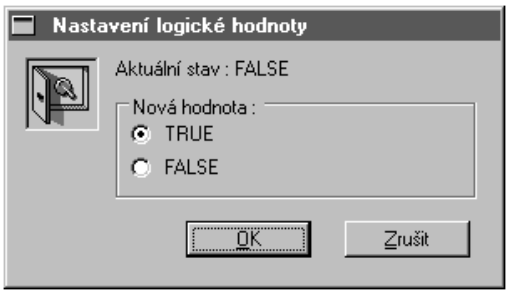
Každý riadok textu v zoznamu môže mať rôzny vzhľad.

Bublinové napovedanie

Niekedy nie je možné textom, ani názornou ikonou plne vystihnúť význam prístroja. Text, ktorý by opisoval taký prístroj by mohol rušivo pôsobiť na celú aplikáciu a časom by sa stal neakceptovateľným.
Control Web umožňuje každému prístroju pridať bublinovú nápovedu. V Inšpektorovi prístroja k takému účelu slúži záložka *bubble*, kde môžeme dodatočne popísať jeho funkciu.
Nápoveda sa zobrazí počas behu aplikácie vo forme bubliny po ukázaní myšou do plochy prístroja.

Modifikácia prístroja *switch*
Switch patrí medzi skupinu prístrojov, ktoré vysielajú stav o svojej aktivite do premennej alebo výstupného kanálu. Reálne vypínače a tlačidlá pracujú na rôznych princípoch. Preto i virtuálny prístroj *switch* obsahuje rad nastavení a funkcií.
Medzi základné, obecné vlastnosti je zmena ikony prístroja (3.2.2.2.), zmena ikony za text (3.2.2.3.) a bublinová nápoveda (3.2.2.4.). Tieto zmeny boli dostatočne popísané u prístroja Indicator a sú platné v celom rade prístrojov. Odlišnosť je pri písaní textu pre aktívny a neaktívny stav. Tu je nastavenie fontu v samostatnej záložke *font.* Taktiež nie je možné opisovať stav prístroja niekoľkými riadkami textu.
Medzi špecifické parametre prístroja patrí najmä mode, logic, init_value a change_icon a sound.

Voľba typu prístroja *switch* - mode
Výrazové prostriedky ako zobraziť tvar prístroja switch sú voliteľné v parametri *mode*.
Ponúkajú sa tri možnosti: icon_button, text-button a change_box.
icon-button
Zmena ikony bola dostatočne popísaná v 3.2.2.2.
text_button
Zmena ikony za text bola dostatočne popísaná v 3.2.2.3.
change_box



Implicitný tvar prístroja je podobný bezpečnostnej schránke. Tento mód umožňuje aktivovať stav vypínača pomocou zvláštneho okna, ktoré sa zobrazí dvojklikom na ikonu. Aktuálny stav sa volí kliknutím na príslušný stav.
Ak nevyhovujú anglické názvy pre aktívny a neaktívny stav, je možné pomocou parametrov *true_text* a *false_text* tieto zmeniť.
Voľbou *mode=change_box* znemožníme nekontrolovanú zmenu stavu vypínača. Všade tam, kde je nutné zabrániť náhodnej zmene stavu vypínača, je tento režim veľmi vhodný.
Ak predvolená ikona v podobe bezpečnostnej schránky nesplňuje predstavy projektu, je možné pomocou parametra **change_icon** vybrať ikonu inú.

Zvukový doprovod prístroja *switch* - sound
Ak projekt vyžaduje, aby pri zmene stavu vypínača bol vydaný doprovodný zvukový signál, je možné názov súboru spolu s cestou k tomto súboru voliť v parametri *sound*. Nastavenie cesty je obdobné, ako je popísané v 3.2.1.4. Systém podporuje všetky štandardné zvukové protokoly známe vo Windows.
Zvukový signál môže okrem krátkeho zvuku aj hlasovo informovať o spúšťanej akcii. Úloha tak môže nadobudnúť „nové rozmery“.

Nastavenie počiatočnej hodnoty prístroja *switch* - init_value
V niektorých prípadoch je potrebné, aby v počiatočnom stave bol vypínač nastavený na aktívnu hodnotu. Ak prečiarkneme v parametri *init_value* okno Inicializacná hodnota prístroja, táto úloha je splnená.

Nastavenie logiky prístroja *switch* - logic
Prístroj switch je vybavený niekoľkými variantami logiky funkcie:

set_flip_flop
Je to obdoba reálného vypínača, ktorý má dva stabilné stavy – zapnutý a rozopnutý. Každým kliknutím na prístroj sa tento prepne do opačného stavu, zmení hodnotu výstupnej premennej a pomocou *receivers* spúšťa iné prístroje.

set_true_on_press

Je to obdoba reálnej klávesy typu „domový zvonček“. Kláves je po dobu „stlačenia“ aktívny. Pri ukončení stlačenia sa prístroj vracia do stabilného kľudového stavu.

set_false_on_press

Je to obdoba reálneho klávesa typu „rozpínacie tlačítko“. Kláves je po dobu „stlačenia“ neaktívny. Pri ukončení stlačenia sa prístroj vracia do stabilného aktívneho stavu. Pre správnu počiatočnú funkciu nezabudnite nastaviť parameter *init_value.*

set_true

Funkcia tejto logiky vychádza z požiadavky „bezpečnostnej klávesy“. Pri kliknutí na prístroj tento prechádza do trvalého aktívneho stavu a nie je možné ďalším kliknutím tento stav zmeniť.

set_false

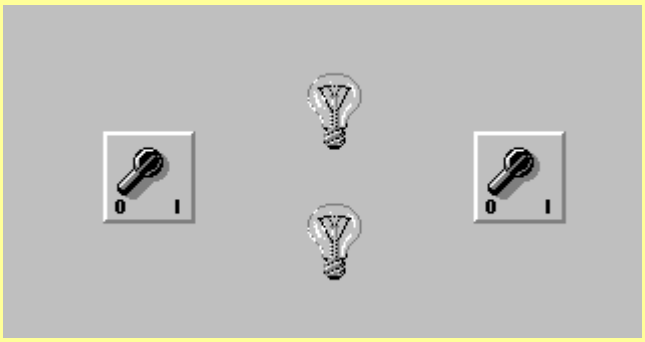
Funkcia tejto logiky vychádza z požiadavky „bezpečnostnej klávesy“. Pri kliknutí na prístroj tento prechádza do trvalého neaktívneho stavu a nie je možné ďalším kliknutím tento stav zmeniť. Nezabudnite nastaviť aj *init_value*.

Vyskúšajte, ako sa bude chovať úloha č. 1 pri nastavení logiky prístroja *switch.*

13.3 Úloha č.2

AKTIVITA

Vytvorte aplikáciu, kde na paneli budú umiestnené dva vypínače a dve žiarovky. Jedna žiarovka bude „svietiť“ pri zopnutí oboch vypínačov do aktívneho stavu a druhá pri zopnutí vypínačov do opačných polôh.



Riešenie – vytvorenie prístroja *panel*, *switch* a *indicator*

Vytvorenie týchto prístrojov bolo popísané v kapitolách 3.1.1., 3.1.4., 3.1.5.. V tejto fáze riešenia je nutné upozorniť na tieto aspekty:
- aplikácia obsahuje dva vysielače informácií (switch), preto je nutné pre každý z nich definovať premennú typu Boolean (3.1.3.)

- vytvoriť logické názvy prístrojov, napr. Vypínač_1, Vypínač_2, Žiarovka_1, Žiarovka_2 a názvy premenných napr. Vodič_1, Vodič_2.
- každý vypínač ovláda obe žiarovky, preto *receivers* musí odkazovať na oba prijímače správ - indicator (3.1.5.1.)

Riešenie – zostavenie logických výrazov

Žiarovka_1
Žiarovka_1 v príklade bude aktivovaná vtedy, ak polohy oboch vypínačov budú v polohe I, teda v aktívnom stave. Preto veta, ktorá opisuje aktívny stav žiarovky bude:
ak bude zapnutý Vypínač_1 a súčasne bude zapnutý Vypínač_2, Žiarovka_1 sa rozsvieti.
Pretože vypínače vysielajú svoj stav do logických premenných, bude logický výraz opisujúci túto vetu znieť:

Vodič_1 and Vodič_2

Uvedený príklad simuluje sériovo zapojené vypínače.

Spôsob nastavenia parametra *expression* je popísaný v kapitole 3.1.4.1. Pokiaľ nevyužijete editor výrazov a budete logický výraz písať, tak pamätajte, že logické spojky sa píšu vždy malými písmenami! Ak toto nedodržíte, výraz bude vyhodnotený ako chybný!

Žiarovka_2
Žiarovka_2 v príklade bude aktivovaná vtedy, ak polohy vypínačov budú v nesúhlasných polohách. Veta, ktorá opisuje aktívny stav žiarovky bude:
ak bude zapnutý Vypínač_1 a súčasne nebude zapnutý Vypínač_2, alebo ak nebude zapnutý Vypínač_1 a súčasne bude zapnutý Vypínač_2, Žiarovka_1 sa rozsvieti.

Pretože vypínače vysielajú svoj stav do logických premenných, bude logický výraz opisujúci túto vetu znieť:

Vodič_1 and not Vodič_2 or not Vypínač_1 and Vypínač_2

Uvedený príklad simuluje schodišťové zapojenie vypínačov.

Riešenie – fragmenty programu

V tejto časti sú uvedené kritické časti programu, ktoré môžu poukazovať na chybne zostavený program.

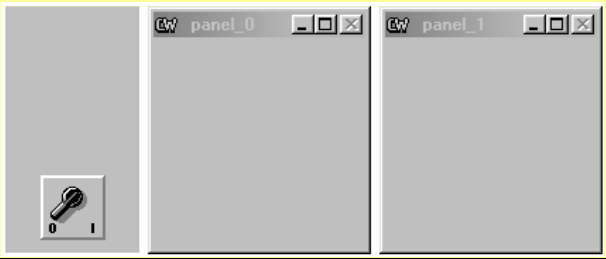
```
var
  Vodič_1 = boolean, false;
  Vodič_2 = boolean, false;
end_var;
instrument

  panel Skúšobný_panel;
    owner = background;
    position = 125, 195, 320, 200;
  end_panel;
```

```
  indicator Žiarovka_2;
    owner = Skúšobný_panel;
    position = 145, 100;
    expression = Vodič_1 and not Vodič_2 or not Vodič_1 and Vodič_2;
  end_indicator;

  switch Vypínač_1;
    owner = Skúšobný_panel;
    position = 50, 65, 46, 46;
    output = Vodič_1;
    receivers = Žiarovka_1, Žiarovka_2;
  end_switch;

  switch Vypínač_2;
    owner = Skúšobný_panel;
    position = 235, 65, 46, 46;
    output = Vodič_2;
    receivers = Žiarovka_1, Žiarovka_2;
  end_switch;

  indicator Žiarovka_1;
    owner = Skúšobný_panel;
    position = 145, 35;
    expression = Vodič_1 and Vodič_2;
  end_indicator;

end_instrument;
```

13.4 Úloha č. 3

AKTIVITA

Vytvorte aplikáciu, kde vypínač umiestnený na paneli bude polohou "0" zviditeľňovať Panel_0 a polohou "1" Panel_1. Vytvorte modifikácie rôznych podmienok viditeľnosti panelov.



Riešenie - nastavenie parametra *visibility*

Vypínač a panely vytvoríme štandartným spôsobom. Nezabudnite na parameter *receivers* vypínača, ktorý musí ukazovať na *Panel_0* a *Panel_1*, na deklaráciu premennej *Vypínač_1* a na jej umiestnenie do parametra *Output* vypínača. To, čo je pre túto úlohu nové, je použitie parametra *visibility* na oboch paneloch.
Parameter *visibility* stanovuje podmienku viditeľnosti prístroja *panel*. Ak ho použijeme, zobrazenie panela sa bude riadiť podľa neho.
V zadaní je požiadavka, aby *Panel_0* bol viditeľný pri polohe "0" vypínača, preto *visibility= not Vypínač_1* (premenná nie je aktivovaná). Prístroj *Panel_1* bude viditeľný pri polohe "1" vypínača, preto *visibility=Vypínač_1*.
Úlohu spustite a presvedčte sa, ako reagujú paneli na polohu vypínača.

Kritické úseky programu:

```
instrument

  window panel Panel_1;
    owner = background;
    position = 294, 37, 150, 150;
    visibility = Vypínač_1;
  end_panel;

  window panel Panel_0;
    owner = background;
    position = 130, 37, 150, 150;
    visibility = not Vypínač_1;
  end_panel;

  panel Skúšobný_panel;
    owner = background;
    position = 25, 15, 95, 175;
  end_panel;

  switch Vypínač;
    owner = Skúšobný_panel;
    position = 25, 120, 46, 46;
    output = Vypínač_1;
    receivers = Panel_0, Panel_1;
  end_switch;

end_instrument;
```

Takýmto spôsobom môžeme vytvárať zložité projekty, kde umiestnenie všetkých prístrojov je z estetického hľadiska nevhodné, alebo jednoducho chceme časti projektu štrukturovať a nechať zobrazené iba v danej chvíli potrebné objekty.

Riešenie - nastavenie parametra *minimize*

Parameter *minimize* prístroja Panel umožní minimalizáciu, ak je táto podmienka logickým výrazom splnená. Ďalšou podmienkou je, aby panel bol v režime "okno". V priebehu spustenej úlohy je možné panel opäť umiestniť na plochu obrazovky do okna kliknutím na jeho minimalizovaný tvar..
Zmodifikujeme úlohu č. 3.4.1. takým spôsobom, že odstránime logické výrazy z parametrov *viziblity* a umiestnime ich do parametrov *minimize.*

Textový tvar úlohy v kritickej časti bude:

```
window panel panel_1;
   owner = background;
   position = 294, 37, 150, 150;
   minimize = Vypínač_1;
 end_panel;

 window panel panel_0;
   owner = background;
   position = 130, 37, 150, 150;
   minimize = not Vypínač_1;
 end_panel;
```

Riešenie - nastavenie parametrov *ascend* a *descend*

Často sa stáva, že pre nedostatok miesta na ploche monitora sa jednotlivé panely navzájom prekrývajú, alebo z hľadiska prevádzky nie je potrebné mať na ploche ovládacie prvky, ktoré v danom čase nie sú potrebné, napríklad servisný panel. V takom prípade je vhodné použiť parameter *ascend* pre "vyplávanie“ panela nad všetky objekty a parameter *descend* pre "ponorenie" panela pod všetky objekty.

Našu úlohu č. 3.4.1. upravme takým spôsobom, že umiestnime paneli na seba a odstránime hodnoty parametrou *visibility.* Ak sa vám nepodarí dostať sa na panel, ktorý je pod, a nie je vidieť, tak stačí kliknúť na príslušný názov panela v okne Vzhľad (vľavo od pracovnej plochy).

Vlastné nastavenie je potom nasledujúce:

```
Panel_0 - ascend=not Vypínač_1   (vynor sa, keď je vypínač v polohe "0")
          descend=Vypínač_1      (ponor sa, keď je vypínač v polohe "1")
Panel_1 - ascend=Vypínač_1       (vynor sa, keď je vypínač v polohe "0")
          descend=not Vypínač_1  (ponor sa, keď je vypínač v polohe "1")
```

Modifikovaná úloha vyzerá v kritickej časti takto:

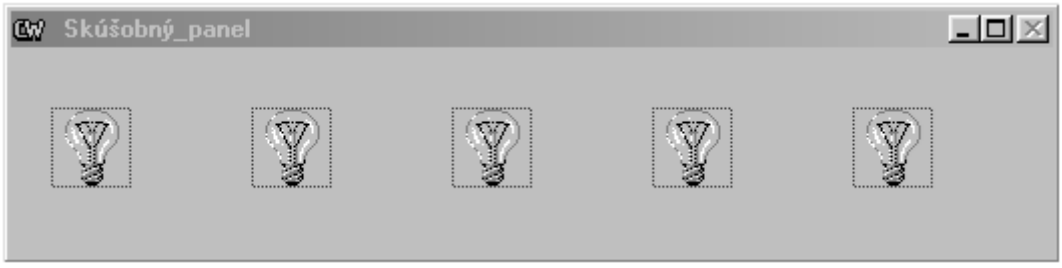
```
window panel panel_1;
   owner = background;
   position = 129, 37, 150, 150;
   ascend = Vypínač_1;
```

```
  descend = not Vypínač_1;
end_panel;

window panel panel_0;
  owner = background;
  position = 129, 37, 150, 150;
  ascend = not Vypínač_1;
  descend = Vypínač_1;
end_panel;
```

13.5 Úloha č. 4

Vytvorte panel s citlivými miestami na kliknutie myši, ktoré rozsvieti príslušnú žiarovku. Príklad riešte s piatimi citlivými miestami, na ktorých sú umiestnené žiarovky.

Riešenie - zložka *item*



Panel vytvoríme obvyklým spôsobom, taktiež žiarovky *Žiarovka_1* až *Žiarovka_5* a premenné *Vypínač_1* až *Vypínač_5*. Nezabudnite nastaviť *receivers* panela na všetky žiarovky. To, čo je v tejto úlohe nové, je zložka *item* prístroja *Panel.*

Zložka *item* dovoľuje zadať v rámci panela postupnosť aktívnych obdĺžnikov (alebo štvorcov) citlivých na kliknutí myšou. Položka obsahuje dve kľúčové slová:

rect - súradnice obdĺžnika v obrazových bodoch relatívne sa vzťahujúce k ľavému hornému rohu panelu. Ak sa kurzor myši nachádza nad aktívnou plochou, zmení sa na tvar ukazujúcej ruky. Kliknutím sa najprv nastavia logické hodnoty FALSE na premenných ostatných citlivých miestach panela, a potom logická hodnota TRUE na premennej, spojenej s vybraným citlivým miestom.
output - premenná typu boolean spojená s daným obdĺžnikom

Nastavenie zložiek *item* je obdobné ako bolo popísané v kapitole 3.2.2.3.

Pretože citlivé miesta nie sú viditeľné, je vhodné na ne nejakým spôsobom upozorniť. V našom prípade budú na týchto miestach umiestené žiarovky. V praxi to môže byť obrázok, napísaný text a podobne.

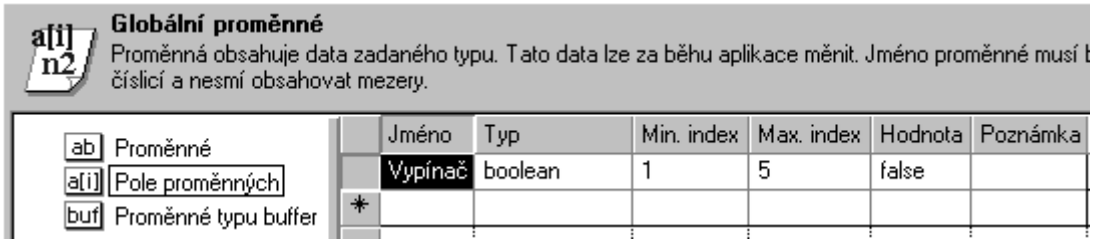
Ak spustíme program, kliknutím na príslušnú žiarovku, táto sa rozsvieti a ostatné zhasnú.

Kritická časť programu:

```
window panel Skúšobný_panel;
  owner = background;
  position = 19, 97, 520, 104;
  item
    rect = 20, 30, 40, 40;
    output = Vypínač_1;
  end_item;
  item
    rect = 120, 30, 40, 40;
    output = Vypínač_2;
  end_item;
  item
    rect = 220, 30, 40, 40;
    output = Vypínač_3;
  end_item;

 item
    rect = 320, 30, 40, 40;
    output = Vypínač_4;
  end_item;
  item
    rect = 420, 30, 40, 40;
    output = Vypínač_5;
  end_item;
  receivers = Žiarovka_1, Žiarovka_2, Žiarovka_3, Žiarovka_4, Žiarovka_5;
end_panel;
```

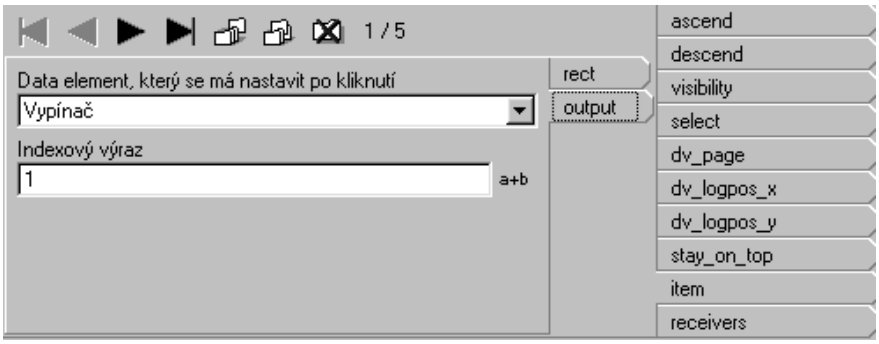
Riešenie - použitie premennej typu pole

Úlohy tohto typu sú predurčené k používaniu premenných typu pole. Premenná typu pole používa jeden názov pre skupinu premenných rovnakého typu, no každej jej zložke je priradený index, cez ktorý je jednoznačne indetifikovateľná. Pri deklarácii je nutné stanoviť počiatočný a koncový index.

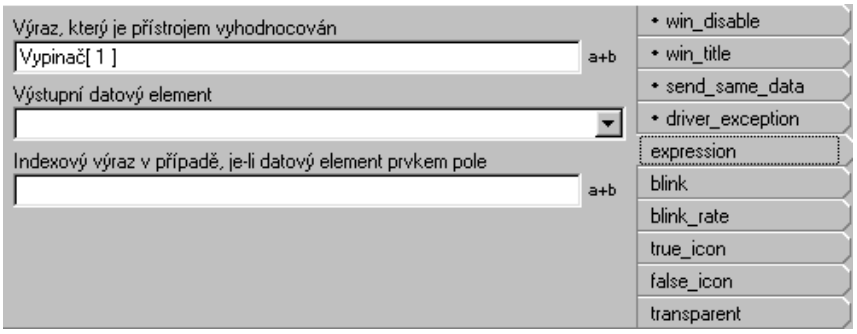


Nastavenie premennej sa uskutoční cez Dátových inšpektorov, zložku premenné, časť Pole premenných. V stĺpci *Min. index* definujeme počiatočné číslo indexu, v stĺpci *Max. index* koncové čislo indexu. Ich veľkosť je daná počtom zložiek, ktoré má premenná obsahovať. Nie je nevyhnutne nutné, aby počiatočné číslo bolo jedna alebo nula. Až sa naučíte používať konštanty, nahraďte tieto čísla nimi.

V prístroji *panel* v zložke *item* ukazuje *Data element* na názov premennej typu pole a *Indexový výraz* na časť pole. Pre prvú zložku je definovaná premenná *Vypínač* nasledovne



V prístroji *indicator* v zložke *expression* sa zapisuje výraz typu pole v podobe úplného zápisu, to je názvu premennej spolu s číslom indexu uzatvoreného do hranatých zátvoriek.

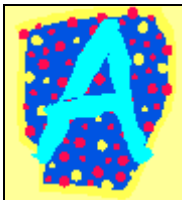


Kritické časti programu:

```
var
  Vypínač = array[ 1..5 ] of boolean, false;
end_var;

instrument

  window panel Skúšobný_panel;
    owner = background;
    position = 19, 97, 520, 104;
    item
      rect = 20, 30, 40, 40;
      output = Vypínač[ 1 ];
    end_item;
    item
      rect = 120, 30, 40, 40;
      output = Vypínač[ 2 ];
    end_item;
    item
      rect = 220, 30, 40, 40;
      output = Vypínač[ 3 ];
    end_item;
    item
```

```
    rect = 320, 30, 40, 40;
    output = Vypínač[ 4 ];
   end_item;
   item
    rect = 420, 30, 40, 40;
    output = Vypínač[ 5 ];
   end_item;
   receivers = Žiarovka_1, Žiarovka_2, Žiarovka_3, Žiarovka_4, Žiarovka_5;
  end_panel;

  indicator Žiarovka_5;
   owner = Skúšobný_panel;
   position = 420, 30;
   expression = Vypínač[ 5 ];
  end_indicator;

  indicator Žiarovka_4;
   owner = Skúšobný_panel;
   position = 320, 30;
   expression = Vypínač[ 4 ];
  end_indicator;

  indicator Žiarovka_3;
   owner = Skúšobný_panel;
   position = 220, 30;
   expression = Vypínač[ 3 ];
  end_indicator;

  indicator Žiarovka_2;
   owner = Skúšobný_panel;
   position = 120, 30;
   expression = Vypínač[ 2 ];
  end_indicator;

  indicator Žiarovka_1;
   owner = Skúšobný_panel;
   position = 20, 30;
   expression = Vypínač[ 1 ];
  end_indicator;

end_instrument;
```

13.6 Úloha č. 5



AKTIVITA

Doplňte do úlohy č.1 text vyjadrujúci názov vypínača a žiarovky

Riešenie - prístroj *label*

Súčasťou každého projektu je riešenie grafickej stránky. Spôsob, ako do úlohy vložiť obrázok alebo zmeniť ikonu bol už popísaný. Nie sú to všetky možností ako obohatiť výtvarnú stránku. V mnohých prípadoch je vhodné k prístrojom doplniť text a umocniť tak ich zmysel. Riešenie je ponúkané v podobe prístroja *label*.
Prístroj *label* zabezpečí zobrazenie ľubovoľného textu a jeho význam je iba dokumentačný a dekoračný. Nijakým spôsobom nepracuje s dátami systému.
Do našej úlohy doplňte dva prístroje *label*, kde jeden bude umiestnený nad vypínač a druhý nad žiarovku. Dajme im názov *Txt_Vypínač* a *Txt_Žiarovka*. Text, ktorý má byť zobrazený napíšeme do zložky *text.* Spôsob zápisu textu, výber fontu a veľkosti bolo popísané v kapitole 3.2.2.3.

Kritická časť programu:

```
label Txt_Vypínač;
  owner = Skúšobný_panel;
  position = 235, 15;
  font = 'MS Sans Serif (Central European)', 8, bold;
  text = 'Vypínač';
end_label;

label Txt_Žiarovka;
  owner = Skúšobný_panel;
  position = 20, 20;
  font = 'MS Sans Serif (Central European)', 8, bold;
  text = 'Žiarovka';
end_label;
```

Riešenie – modifikácia prístroja *label – blink, blink rate, transparent, frame*

Pokiaľ chceme zvýrazniť zmenu pomocou prístroja *label*, môžeme nechať text po splnení určitej podmienky blikať.
Pre túto variantu doplňte v záložke prístroja *Txt_Žiarovka* logický výraz – v našom prípade logickú premennú *Vypínač_1* (pripomeňme, že text je ovládaný premennou, preto nezabudnite nastaviť *receivers* prístroja *Vypínač*).

Pomocou záložky *blink_rate* môžeme ovplyvniť rýchlosť blikania. Ak vás ruší zmena pozadia za textom, môžete nastaviť priehľadné pozadie pomocou záložky *transparent.*
Text môže nadobudnúť aj 3-D okraje doplnením záložky *frame* o hodnotu výšky rámčeka.

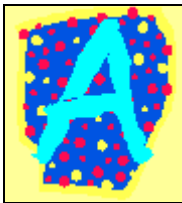
Kritická časť programu:

```
label Txt_Žiarovka;
  owner = Skúšobný_panel;
  position = 20, 20;
  frame = 1;
  blink = Vypínač_1;
  blink_rate = quick;
  transparent;
  font = 'MS Sans Serif (Central European)', 8, bold;
  text = 'Žiarovka';
end_label;

switch Vypínač;
  owner = Skúšobný_panel;
  position = 235, 40, 46, 46;
  output = Vypínač_1;
  receivers = Žiarovka, Txt_Žiarovka;
end_switch;
```

Riešenie – modifikácia prístroja *label – icon*

Text nie je jediný výrazový prostredok, ktorý je možné použiť v prístroji *label*. Ďalšou variantou je použitie záložky *icon* a zvoliť si z preddefinovaných ikon, alebo vytvoriť vlastnú pomocou Editora ikon.

13.7 Úloha č. 6

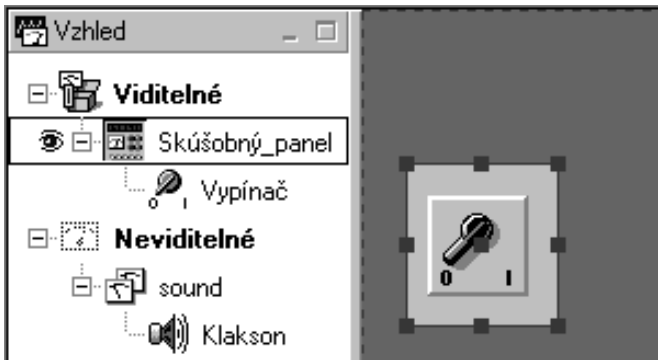


AKTIVITA

Vytvorte vypínač na paneli, kde sa po jeho zapnutí prehraje zvuk.

Riešenie – prístroj *sound*

Nie vždy je grafické zobrazenie tou najvhodnejšou metódou upozorniť na udalosť v riadiacom procese. Obsluha spravidla vykonáva aj inú činnosť a nepretržité sledovanie obrazovky monitoru je nemožné aj z hygienického hľadiska.
Riešením, ktoré sa ponúka, je prehratie zvukovej nahrávky pri výskyte udalosti.
Panel, vypínač a premennú vytvoríme štandardným spôsobom. To, čo sa štandardným spôsobom neumožní je umiestniť prístroj *sound* na plochu. Prístroj *sound* totiž patrí medzi neviditeľné prístroje. Preto ak chceme tento prístroj použiť, musíme ho umiestniť medzi neviditeľné prístroje a to takým spôsobom, že ho z panela prístrojov pretiahneme až na položku stromu **Neviditeľné**.



Prístroj pomenujeme *Klakson.* V záložke *file* definujeme názov a cestu k zvukovom súboru rovnako, ako to bolo popísané v kapitole 3.2.3.2. Nezabudnite na parameter *receivers* u prístroja *Vypínač_1*.

Ak úlohu spustíme, každou zmenou sa prehraje zvuk. Avšak našou požiadavkou je, aby sa zvuk spustil pri prechode vypínača do polohy „I“. Preto definujeme podmienku pre spustenie v parametri *Štart*. Tou bude samotná premenná.

Kritická časť programu:

```
switch Vypínač;
  owner = Skúšobný_panel;
  position = 10, 15, 46, 46;
  output = Vypínač_1;
  receivers = Klakson;
end_switch;

sound Klakson;
  file = 'Buzzer.wav';
  start = Vypínač_1;
end_sound;
```

Pokiaľ požadujeme predčasné ukončenie prehrávania zvukového súboru, definujeme podmienku v parametri *Stop*. Pri jej splnení nemajú ostatné parametre význam.

Modifikácia – viacnásobné prehrávanie zvukového záznamu

Zvukový signál určený na prehrávanie môže byť krátky, preto variant viacnásobného prehrávania zvyšuje pravdepodobnosť, že si toho obsluha všimne.

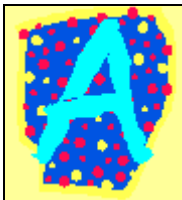
Pre stanovení počtu opakovaní zvukového záznamu zadáme do parametra *num_repeat* celé číslo. Ďalej je nutné nastaviť aj periódu, za ktorú sa znova spustí prehrávanie – parameter *period*.

Kritická časti programu:

```
sound Klakson;
  file = 'Buzzer.wav';
```

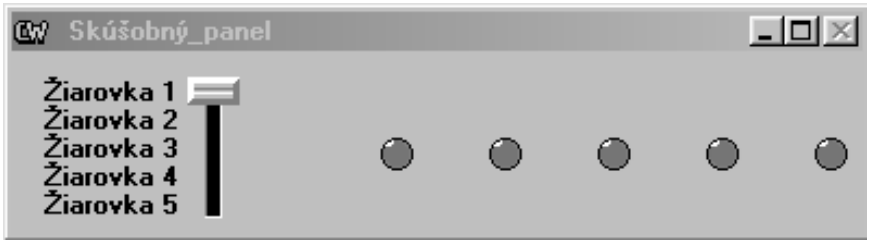
```
  period = 1;
  num_repeat = 3;
  start = Vypínač_1;
end_sound;
```

13.8 Úloha č. 7



AKTIVITA

Vytvorte panel s päťnásobným prepínačom a žiarovkami.
Žiarovky sa budú rozsvecovať príslušnou polohou prepínača.
Vytvorte modifikácie prepínača.



Riešenie- prepínač v tvare *slider*

Na vytvorený panel so žiarovkami pretiahnite z Panela prístrojov prístroj *multi_switch*. V Inšpektorovi prístroja zvoľte jeho názov, prípadne skontrolujte, kto je vlastníkom prístroja. Tým by mal byť Skúšobný panel. Nezabudnite nastaviť zložku *receivers* na všetky žiarovky. Vytvorte v Dátových inšpektoroch premenné, pre každú polohu osobitne, alebo najlepšie premennú typu pole s piatimi zložkami.
V položke *item* prístroja, v záložke *text* definujte názov príslušnej polohy a v položke *output* názov premennej s prípadným indexom. Pridávaním zložiek do zoznamu určujete počet polôh vypínača. Tvar písma je možné definovať v záložke *font*.

V záložke *mode* definujte typ prístroja - slider. Slider má tvar prepínača s bežcom, ktorý má vždy aktívnu iba jednu polohu. Znamená to, že v našom príklade bude svietiť vždy iba jedna žiarovka.
Úlohu spustite. Zmenou polohy prístroja sa budú rozsvecovať príslušné žiarovky.

Kritická časť prístroja:

```
multi_switch Prepínač;
  owner = Skúšobný_panel;
  position = 11, 8, 98, 73;
  receivers = Žiarovka_1, Žiarovka_2, Žiarovka_3, Žiarovka_4, Žiarovka_5;
  item
    text = 'Žiarovka 1';
    output = Vypínač[ 1 ];
  end_item;
  item
    text = 'Žiarovka 2';
```

```
    output = Vypínač[ 2 ];
  end_item;
  item
    text = 'Žiarovka 3';
    output = Vypínač[ 3 ];
  end_item;
  item
    text = 'Žiarovka 4';
    output = Vypínač[ 4 ];
  end_item;
  item
    text = 'Žiarovka 5';
    output = Vypínač[ 5 ];
  end_item;
end_multi_switch;
```

Riešenie- prepínač v tvare *combo_box*

Ak zmeníme parameter *mode* na variant *combo_box*, tak pri kliknutí na prístroj sa otvorí ponuka, z ktorej volíme polohu prepínača. Aj v tomto prípade sa prístroj chová ako prepínač pre voľbu jedného stavu z preddefinovaných.

Riešenie- prepínač v tvare *check_box*

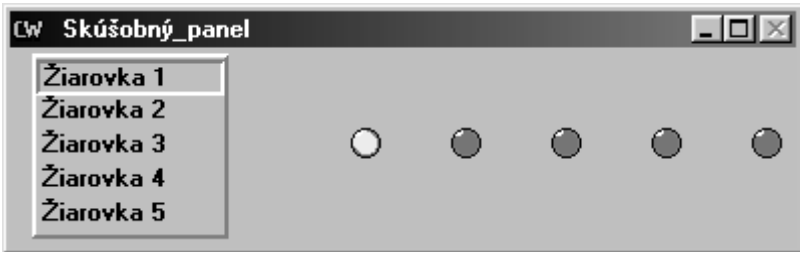
Vo variante, kde *mode=check_box,* volí sa ponuka stlačením príslušného tlačidla vedľa popisu. Tento mód prístroja umožňuje voľbu položiek súčasne, to je nezávisle jeden na druhom. Výsledkom je, že žiarovky môžu svietiť v ľubovoľnej kombinácii.

Riešenie- prepínač v tvare *radio_button*

Mode=radio_button ponúka tlačidlá v rade pod sebou. Pri voľbe sa prístroj chová ako prepínač jedného stavu z možných.

Riešenie- prepínač v tvare *menu*

Voľbu v tvare menu ponúka tento variant. Prístroj sa chová ako prepínač jedného stavu z možných.



Riešenie- prepínač v tvare *menu_bar*

Voľba *mode=menu_bar* ako jediná ponúka horizontálnu ponuku. Aj v tomto prípade ide o voľbu jedného stavu z možných.

13.9 Úloha č. 8

AKTIVITA

Vytvorte panel s desaťnásobným prepínačom a multifunkčným displejom, ktorý zobrazí čísla. Vytvorte modifikácie displeja.

Riešenie – multifunkčný dispej *multi_label* s ikonami

Prístroj *multi_label* je určený na zobrazovanie ľubovoľných ikon alebo textov v závislosti na výsledkoch vyhodnotenia logických výrazov. Prístroj môže podľa stanovených podmienok meniť svoj vzhľad aj polohu.
Na plochu preneste panel a do neho aplikujte prístroj *multi*_switch a *multi*_label. Nezabudnite vytvoriť premennú vodič, v najlepšom prípade ako indexovanú premennú s desiatimi stavmi (0 – 9). Nezabudnite ani na zložku item a receivers v prístroji *multi*_switch.
Ak má displej zobrazovať viac ikon, je potrebné toto nadefinovať v zložke *item*. Cestu k icon súboru vyberte obvyklým spôsobom. Pre stav prepínača 0 vyberte ikonu sedemsegmentového displeja 0, pre stav 1 ikonu s jedničkou a tak ďalej. Súčasne s definíciou ikon definujte i parameter *expression* – pre stav 0 premennú Vodič [0], … .Spôsob pridávania zložiek bol popísaný v kapitole 3.2.2.3. a spôsob zápisu indexovanej premennej v kapitole 3.5.2.

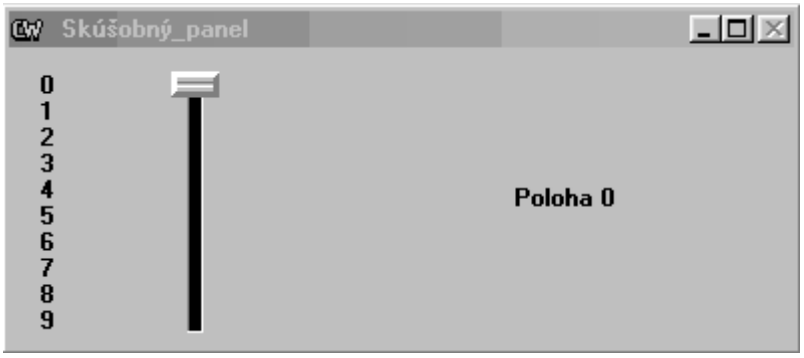
Spustená úloha bude na displeji zobrazovať pozíciu nastaveného prepínača.

Kritické časti programu:

```
multi_label Displej;
   owner = Skúšobný_panel;
   position = 251, 58;
   item
     icon = '7SEG0.ICO';
     expression = Vypínač[ 0 ];
     paper = lgray;
     ink = black;
     blink_paper = white;
     blink_ink = red;
   end_item;
   item
     icon = '7SEG1.ICO';
     expression = Vypínač[ 1 ];
     paper = lgray;
     ink = black;
     blink_paper = white;
     blink_ink = red;
   end_item;
.....
.....
   item
     icon = '7SEG9.ICO';
     expression = Vypínač[ 9 ];
     paper = lgray;
     ink = black;
     blink_paper = white;
     blink_ink = red;
   end_item;
 end_multi_label;
```

Riešenie – multifunkčný dispej *multi_label* s textom

Ďalšou možnosťou ako zobraziť sled informácií závislých na stavu premenných je použitie textových správ. Ďalej budú opísané iba zmeny oproti predošlej úlohe. Ak budete úlohu modifikovať z predošlej úlohy, odstráňte všetky položky zo záložky *item – icon*.



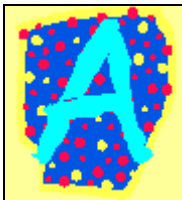
V položke *item* nastavte *expression* pre prvý stav, to je Vypínač [0]. Zatiaľ neprechádzajte na novú zložku pomocou ikony *Pridaj*, ale prejdite na záložku *text*, kde napíšte text a prípadne vyberiete príslušný font. Všimnite si, že ak má mať text viacej riadkov, použijeme ikonu *Pridaj*, ako to bolo popísané v 3.2.2.3. Upozorňujem, že toto pridanie ďalšieho riadku textu nemá nič spoločné s pridaním nového stavu prístroja. Nový stav prístroja získame, ak prejdeme naspäť do záložky *expression* a stlačíme ikonu *Pridať*. Taký postup opakujeme, až stanovíme všetky možnosti pre zobrazenie stavov prepínača.

Kritická časť programu:

```
multi_label Displej;
   owner = Skúšobný_panel;
   position = 251, 68;
   item
     font = 'MS Sans Serif (Central European)', 8, bold;
     text = 'Poloha 0';
     expression = Vypínač[ 0 ];
     paper = lgray;
     ink = black;
     blink_paper = white;
     blink_ink = red;
   end_item;
   item
     font = 'MS Sans Serif (Central European)', 8, bold;
     text = 'Poloha 1';
     expression = Vypínač[ 1 ];
     paper = lgray;
     ink = black;

     blink_paper = white;
     blink_ink = red;
   end_item;
   item
  .....
```

Úlohu je možné modifikovať o farebné kompozície a voľbou fontov. K možnostiam prístroja s požitím parametrov *x_position* a *y_position* sa vrátime neskôr, až preberieme potrebné doplňujúce prístroje.

13.10 Úloha č. 9



AKTIVITA

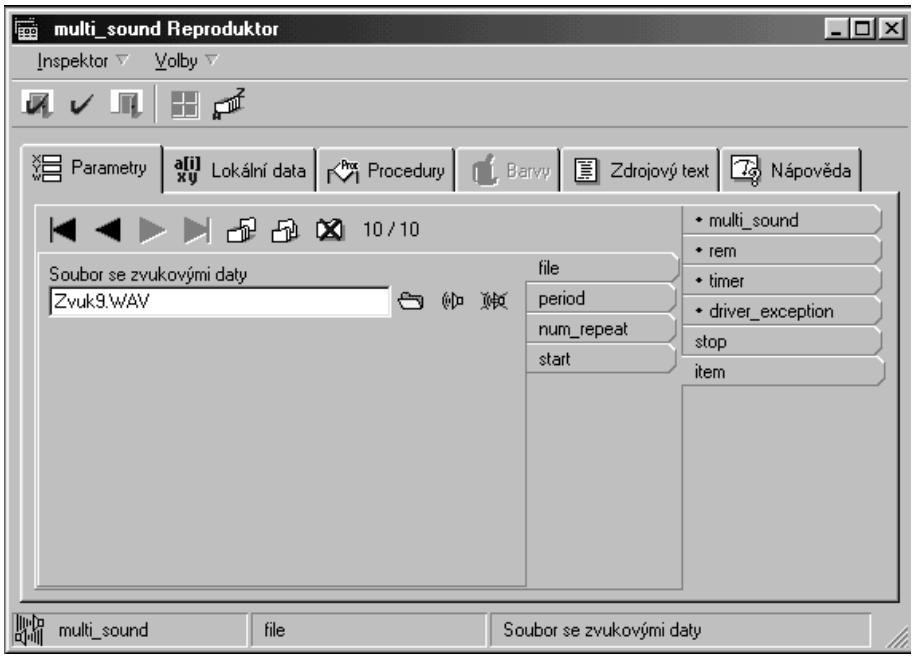
Na paneli vytvorte prepínač, ktorý zvukovou prehrávkou oznámi prechod z jednej polohy do druhej.

Riešenie – prístroj *multi_sound*

Dôvody použitia „zvukových prístrojov“ boli popísané v kapitole 3.7.1. u prístroja *sound*. Prístroj *multi*_sound však môže obsahovať niekoľko odkazov na nahrávky, ktoré je možné prehrávať na základe rôznych udalostí.

Panel, prepínač a premenné vytvoríme štandardným spôsobom (upozorňujem na možnosť definovať premennú typu pole). Podobne ako prístroj *sound* nepatrí ani prístroj *multi_sound* medzi viditeľné prístroje. Umiestnime ho teda medzi neviditeľné prístroje v strome *Vzhľad* tak, že ho z panela prístrojov pretiahneme až na položku stromu **Neviditeľné**.

Prístroj pomenujeme *Reproduktor*. V záložke *item* definujeme štandardným spôsobom parameter *file* (3.2.3.2.), *period, num_repeat a štart* (3.7.2.). Aj pridanie ďalších zvukových záznamov sa robí štandardne ikonou *Pridať*. Nezabudnite na parameter *receivers* u prístroja *Prepínač*.
Ak úlohu spustíme, každou zmenou polohy prepínača sa prehrá príslušný zvuk.



Kritická časť programu:

```
multi_sound Reproduktor;
 item
   file = 'Zvuk0.WAV';
   period = 0;
   num_repeat = 0;
   start = Vypínač[0];
 end_item;
 . . . . .
 . . . . .
 item
   file = 'Zvuk9.WAV';
   period = 0;
   num_repeat = 0;
   start = Vypínač[9];
 end_item;
end_multi_sound;
```

Pokiaľ požadujeme predčasné ukončenie prehrávania zvukového súboru, definujeme podmienku v parametri *Stop*. Pri jej splnení nemajú ostatné parametre význam.
Viacnásobné prehrávanie bolo popísané v kapitole 3.7.2. Upozorňujem, že použitie prístroja *multi_sound* nespadá iba do oblasti prepínačov. V praxi môžu byť jednotlivé zvukové nahrávky aktivované z ľubovolných logických prístrojov a ich kombinácií.

13.11 Úloha č. 10

AKTIVITA

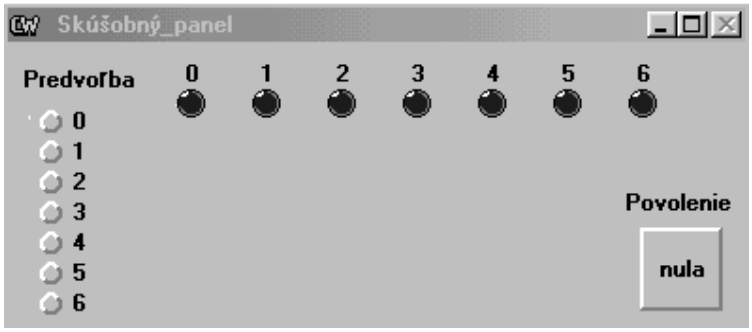
Na paneli vytvorte prepínač v tvare predvoľby a vypínač, v ktorom sa bude stav predvoľby zobrazovať. Na stave vypínača *Povolenie* bude závisieť, či kontrolky 0 až 3 budú signalizovať stav predvoľby. Žiarovky 4 až 6 budú zobrazovať stav prepínača nezávisle od vypínača.

Riešenie – prístroj *switch_label*

Panel, prepínač a pomocné texty vytvoríme štandardným spôsobom. Ďalej vytvoríme sedem premenných (pole typu boolean), ktoré budú ovládané prepínačom, a ešte jednu, tiež boolean, ktorú bude ovládať prístroj typu *switch_label*.

Prístroj typu *switch_label* umožňuje nastaviť do výstupnej premennej hodnotu podobne, ako je to u prístroja *switch* a zároveň sa chová ako prístroj *multi_label*, ktorý zobrazuje text (alebo ikonu) na základe logických podmienok. Ak nie je logická podmienka splnená, prístroj sa nezobrazí.

Z dôležitých parametrov prístroja switch_label sú to najmä *output, receivers, logic* a *item.* Nastavenie *output* a *receivers* je zhodné s prístrojom switch (3.2.3.4.) a *item* je zhodné s prístrojom *multi_label* (3.9.2). Nová položka je *fixed_size*, ktorá zaručí, že sa nebude meniť veľkosť prístroja ak sú texty v položke *item* rôzne dĺžky.



Kritické fragmenty programu:

```
Štart = boolean, false;
Vypínač = array[ 0..6 ] of boolean, false;

indicator Žiarovka6;
  owner = Skúšobný_panel;
  position = 330, 25;
  expression = Vypínač[ 6 ];
  true_icon= 'LED16RON.ICO';
  false_icon = 'LED16ROF.ICO';
end_indicator;

 indicator Žiarovka3;
  owner = Skúšobný_panel;
  position = 210, 25;
  expression = Štart and Vypínač[ 3 ];
  true_icon = 'LED16RON.ICO';
  false_icon = 'LED16ROF.ICO';
end_indicator;

 switch_label Povolenie;
  owner = Skúšobný_panel;
  position = 336, 98, 44, 44;
  output = Štart;
  receivers = Žiarovka0, Žiarovka1, Žiarovka2, Žiarovka3;
  logic = set_flip_flop;
  fixed_size;
  item
    font = 'MS Sans Serif (Central European)', 8, bold;
    text = 'nula';
    expression = Vypínač[ 0 ];
    paper = lgray;
    ink = black;
    blink_paper = white;
```

```
    blink_ink = red;
  end_item;

  item
    font = 'MS Sans Serif (Central European)', 8, bold;
    text = 'Tri';
    expression = Vypínač[ 3 ];
    paper = lgray;
    ink = black;
    blink_paper = white;
    blink_ink = red;
  end_item;
end_switch_label;

multi_switch Prepínač;
  owner = Skúšobný_panel;
  position = 11, 33, 98, 114;
  mode = radio_button;
  receivers = Žiarovka0, Žiarovka1, Žiarovka2, Žiarovka3, Povolenie, Žiarovka4,
Žiarovka5, Žiarovka6;
  item
    text = '0';
    output = Vypínač[ 0 ];
  end_item;
  . . . . . .
  . . . . . .
end_multi_switch;
```

13.12 Úloha č. 11

AKTIVITA

Na paneli vytvorte zadávacie textové pole pre voľbu sériového komunikačného portu COM1 až COM4. Voľbou bude možné zadávať buď z klávesnice, alebo pomocou ponuky. Ošetrite zle definovaný komunikačný port.

Riešenie – prístroj *string_switch*

Na štandardne vytvorený panel umiestnite prístroj *string_switch, label* a *multi_label.* V dátovom inšpektorovi definujte premennú COM_port typu string (textový typ).

Prístroj typu *string_switch* umožňuje do dialógového okna zadávať pomocou klávesnice text, alebo prednastavený text z ponuky do okna umiestňovať.

Z dôležitých parametrov prístroja *string_switch* sú to najmä *output, receivers, enter_button, item a selected.*

output je štandardný parameter definujúci výstupný dátový element, do ktorého sa bude text zapisovať (*Output* = COM_port).

receivers, jeden z najdôležitejších parametrov „vysielacích" prístrojov už nie je nutné predstavovať. V príklade bude aktivovať *multi_label* (*receivers* = Text).

item zložka definuje textové varianty, ktoré budú prístrojom ponúknuté. Spôsob ukladania textov do zložky *item* je štandardný. V úlohe budú naplnené štyri zložky hodnotami 0 až 4 (pre voľbu COM1 až COM4). Parameter **selected** určí, aký variant bude ponúknutý ako prvý pri spustení úlohy. Ak určíme u viacerých položiek parameter *selected*, bude vybraný prvý variant z poradí, ak parameter *selected* nedefinujeme u žiadnej položky, bude vybraný prvý variant z celkového poradia.

enter_button je príznak rozhodujúci o zobrazení potvrdzovacej klávesy. Kláves je umiestnený na koniec dialógového okna. Ak je aktivovaný, potom je možné poslať hodnotu aj pomocou myši.

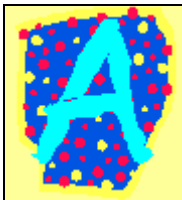
Na príklade je ešte zaujímavé nastavenie item zložky prístroja *Text* (multi_switch). Parameter **expression** definuje zobrazovaný text a podmienky zobrazenia. Prvé štyri pozície definujú podmienku pre zobrazenie jedného z možných stavov (napr. COM_port = '0'). Piata pozícia, označená textom COM infinite, definuje zakázaný stav. Podmienka pre ošetrenie tohto stavu sa ponúka v relačnom zápise *ord (COM_port) > 3*. Ord je prevodná funkcia, prevádzajúca textový znak na číslo. Nájdeme ju štandardne v Editori výrazov.

Kritické fragmenty programu:

```
var
  COM_port = string, '';
end_var;

multi_label Text;
    item
      font = 'MS Sans Serif (Central European)', 8, bold;
      text = 'COM 0';

      expression = COM_port = '0';
end_item;
   item
      text = 'COM 1';
      expression = COM_port = '1';
    end_item;
    item
```

```
    text = 'COM 2';
    expression = COM_port = '2';
  end_item;
  item
    text = 'COM 3';
    expression = COM_port = '3';
  end_item;
  item
    text = 'COM infinite';
    expression = ord( COM_port ) > 3;
  end_item;
 end_multi_label;

 string_switch COM_port;
  output = COM_port;
  receivers = Text;
  item
    text = '0';
    selected;
  end_item;
  item
    text = '1';
  end_item;
  item
    text = '2';
  end_item;
  item
    text = '3';
  end_item;
 end_string_switch;
end_instrument;
```

13.13 Úloha č. 12



AKTIVITA

Na paneli vytvorte zadávacie textové pole pre voľbu sériového komunikačného portu COM1 až COM4. Voľbu bude možné zadávať buď z klávesnice, alebo pomocou ponuky. Zobrazte hodnotu výstupnej textovej premennej.

Riešenie – prístroj *string_display*

Na štandardne vytvorený panel umiestnite prístroj *string_switch, label* a *string_display*. V dátovom inšpektorovi definujte premennú COM_port typu string (textový typ).

Prístroj typu *string_switch* definujte ako v kapitole 3.12.1. iba s tým rozdielom, že parameter item bude obsahovať text COM 0 až COM 4.

Z dôležitých parametrov prístroja *string_display* sú to najmä expression a justify.

expression je parameter definujúci textový výraz alebo premennú, ktorú bude prístroj zobrazovať. Pretože prístroj bude zobrazovať hodnotu premennej, bude *expression* = COM_port.

justify určuje, akým spôsobom sa text zobrazí – vľavo, vpravo a centrovane na stred..

Kritické fragmenty programu:

```
var
  COM_port = string, '';
end_var;

string_display Text;
  owner = Skúšobný_panel;
  position = 276, 28, 85, 18;
  expression = COM_port;
end_string_display;

  string_switch COM_port;
    owner = Skúšobný_panel;
    position = 11, 28, 90, 18;
    output = COM_port;
    receivers = Text;
    item
      text = 'COM 0';
      selected;

  end_string_switch;
end_instrument;
```

13.14 Úloha č. 13

AKTIVITA

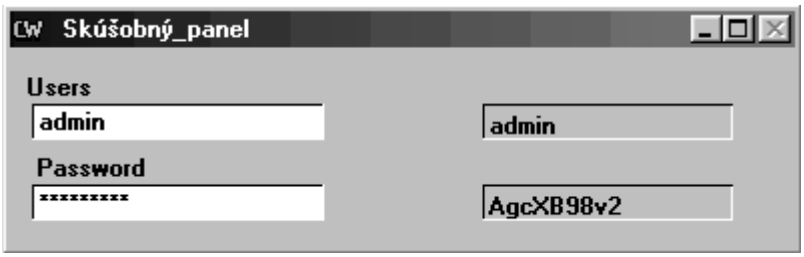
Na paneli vytvorte zadávacie textové pole pre vkladanie užívateľského mena a hesla. Užívateľské meno sa bude zobrazovať v normálnom tvare, text hesla bude skrytý. Zobrazte hodnoty výstupných textových premenných.

Riešenie – prístroj *string_control*

Na štandardne vytvorený panel umiestnite prístroj *string_control*, *label* a string_*display*. V dátovom inšpektorovi definujte premenné *Users* a *Password* typu string (textové typy).

Prístroje typu *string_display* definujte ako v kapitole 3.13.1.. V prístroji *Users* bude definovaná premenná *expression* = *Users*, v prístroji Password bude *expression* = *Password*.

Z dôležitých parametrov prístroja *string_control* sú to najmä *output*, *receivers* *a password_mode.*



output je štandardný parameter definujúci výstupný dátový element, do ktorého sa bude text zapisovať (o*utput* = Users, prípadne output = Password).

receivers aktivuje *string*_display. U prístroja *Users* definujte *receivers* = Text_users, v prístroji *Password* definujte *receivers* = Text_password.

password_mode v prístroji *Password* zaistí, že namiesto normálnych znakov sa budú zobrazovať znaky „hvezdička“.

Kritické fragmenty programu:

```
var
  Users = string, '';
  Password = string, '';
end_var;

  string_control Password;
    output = Password;
    password_mode;
    receivers = Text_password;
  end_string_control;

  string_control Users;
    output = Users;
    receivers = Text_users;
  end_string_control;
. . . . .
. . . . .
end_instrument
```

14 SPOJITÉ PRÍSTROJE

V číslicových počítačoch je každá spojitá informácia transformovaná na rad diskrétnych entít, na číslicový signál. Túto úlohu preberajú analógovo-číslicové prevodníky umiestené na vstupných, výstupných alebo vstupne/výstupných kartách počítača. Na tejto úrovni teda nemá zmysel hovoriť o spojitých či nespojitých prístrojoch.
Ak sa na prístroje pozrieme z iného uhla, môžeme rozlíšiť funkciu rôznych vypínačov a prepínačov, žiaroviek a displejov (ako predstaviteľov diskrétnych prístrojov) od potenciometrov a meracích prístrojov (ako predstaviteľov analógových prístrojov).
Spojité prístroje vo svojej funkcii využívajú digitalizovanú hodnotu analógového signálu, pričom úroveň diskrétneho kroku je možné nastaviť ako jeden z parametrov opisujúci správanie virtuálneho prístroja.

Control Web 5 je vybavený celým radom premenných, ktoré prenášajú digitalizovanú hodnotu spojitého signálu. V tabuľke je uvedený prehľad premenných spolu s rozsahom prenášaných hodnôt.

Typ premennej	Prenášané hodnoty
shortint	-128..127
shortcard	0..255
integer	-32768..32767
cardinal	0..65535
longint	-2147483648..2147483647
longcard	0..4294967295
shortreal	+-1.2E-38..3.4E+38
real	+-2.3E-308..1.7E+308

Z dôvodov čo najpresnejšej reprezentácie analógových veličín sú v systéme **Control Web 5** všetky čísla skladované s presnosťou na 15 platných čísel a s rozsahom exponentu od E-308 do E+308.

Jednotná a najpresnejšia možná reprezentácia čísel (na súčastných počítačoch) prináša autorovi aplikačného programu veľké výhody. Hlavnou výhodou je neobmedzená kompatibilita všetkých číselných typov. Pri priraďovaní hodnôt do číselných premenných nemusíte dodržovať typ číselných zapisovaných hodnôt. Rovnako sa pri zápise matematických výrazov nemusíte zaťažovať číselnými typmi jednotlivých použitých dátových elementov. Neriskujete pritom prípadnú stratu presnosti výsledku výpočtu.

Táto vlastnosť nesporne prináša značné zjednodušenie vývoja a väčšiu robustnosť aplikačných programov. Je tu však jeden dôsledok, ktorý je dobré mať pri zostavovaní niektorých špeciálnych programov na pamäti. Keď napríklad postupne pripočítame jednotku do premennej typu *shortcard* (osem bitové celé nezáporné číslo), obsahuje táto premenná celé čísla v rozsahu od 0 do 255. Ak pripočítame jednotku k číslu 255, dôjde k takzvanému pretečeniu číselného typu a hodnota premennej sa zmení na nulu. Ak priradíme túto premennú do premennej typu *real*, bude obsahovať hodnotu 256 a nie hodnotu 0. Toto je logickým dôsledkom faktu, že hodnota premennej je vnútorne uchovávaná v maximálnom možnom rozlíšení a rozsahu.

Pomocou analógových virtuálnych prístrojov vytvárame na obrazovke počítača reálne riadiace pulty a velíny. Virtuálne vstupné prístroje nie sú zaťažené neúmerným prúdom ani napätím, neopaľujú sa bežce potenciometrov ako je to bežné v reálnych systémoch. Vo virtuálnych výstupných prístrojoch nedochádza k starnutiu, zadrhávaniu rúčkových systémov ani k rozbitiu sklenných teplomerov. Sú to ideálne prístroje, poskytujúce trvalo presné hodnoty reálnej veličiny. Samozrejme presnosť prístroja je daná obdobne ako v reálnych prístrojoch správnou voľbou meracej metódy, ďalej správnou voľbou meracej karty, kvalitou analogovo-číslicového prevodníka a frekvenciou vzorkovania analógového signálu.

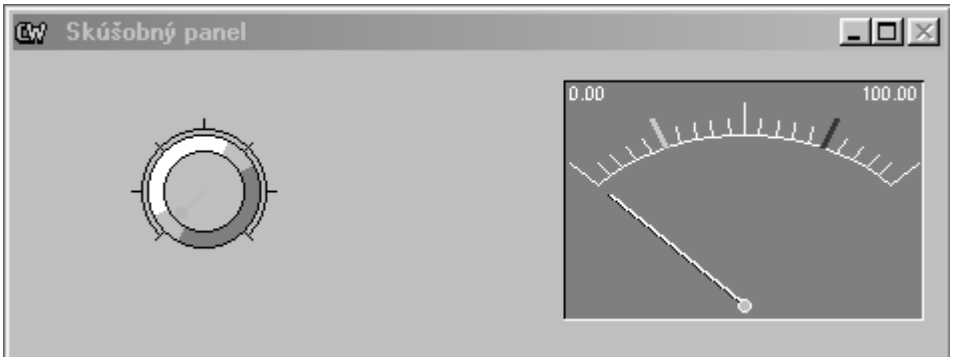
14.1 Úloha č. 14

AKTIVITA

Na paneli vytvorte otočný potenciometer ako simulovaný zdroj signálu a rúčkový merací prístroj. Rozsah výstupného signálu potenciometra a rozsah stupnice meracieho prístroja nastavte od 0 do 100. Dráha potenciometra bude rozdelená na úseky po 0.1 (krok potenciometra).

Riešenie – prístroj *control a meter*

Na štandardne vytvorený panel umiestnite prístroj *control* a *meter*. V dátovom inšpektorovi definujte premennú *Signál* typu *shortreal* alebo *real*.



Riešenie – prístroj *meter*

Prístroje typu *meter* zobrazujú výsledok numerického výrazu pomocou rúčkového prístroja, grafu, poprípade číslicového zobrazovača.

Dôležité parametre prístroja *meter*:

expression - už známy parameter. Jeho obsahom bude numerický výraz, ktorého hodnotu bude prístroj zobrazovať.
mode - určuje grafickú podobu prístroja
pointer - napodobuje rúčkový prístroj zobrazujúci spojitú veličinu
digital - digitálny výstup zobrazený pomocou bodového zobrazovača
dot_digital - digitálny výstup zobrazený pomocou segmentového zobrazovača

vertical_bar, horizontal_bar - farebná plocha, ktorá sa zapĺňa farbou v pomeru hodnoty výrazu *expression* k maximálnemu rozsahu prístroja

text_display - výpis hodnoty spolu s textom. Údaj má väzbu na parameter **mask**

flow_graph, sweep_graph - vykresľujú graf historického trendu. Mód *flow_graph* vykresľuje poslednú hodnotu vždy na koniec grafu, pričom sa celý graf priebežne posúva. Mód *sveep_graph* vykresľuje vždycky od začiatku grafu, po dosiahnutí konca začne prepisovať hodnoty opäť od začiatku, graf sa neposúva.

line flow_graph – graf podobný módu *folw_graph***.** V tomto móde ale nie je využívaný grafický posuv časti grafu, ktorý je pri moderných grafických kartách veľmi rýchly. Graf je po častiach prekresľovaný priamkami. To prináša možnosť stojacej mriežky pri posunujúcom sa grafe. Nevýhodou tohto módu je nutnosť vykresľovania vysokého počtu čiar v každom časovom kroku a tým značná spotreba času na prekresľovanie.

mask - šablóna pre vpisovanie číselnej hodnoty do ľubovolnej textovej masky (pre *mode = text_display*)**.**

Pokiaľ masku zapisujeme pomocou Inšpektora prístroja (a to je náš prípad), tak nepoužívame apostrofy. A by sme zapisovali masku v textovom editori, musí byť maska uzatvorená v apostrofoch.

Napríklad môže maska vyzerať takto: *mask = Napätie ### Voltov.* Znaky # alebo @ udávajú umiestnenie a presnosť zobrazovanej numerickej hodnoty. Ak je použitý znak #, namiesto prípadných nevýznamných núl pre číslom sú doplnené medzery. Ak je použitý znak @ sú v pevnom formáte vpisované aj tieto nuly.

disable_menu - zrušenie menu, ak je prístroj v okne. Ak umiestnime prístroj do okna (ako inak, než stlačením "okna" v Inšpektorovi prístroja), tak sa zobrazí roletové menu **Mod.** Pomocou neho sa môžeme prepnúť na rôznu podobu prístroja. Ak si neželáme prepínanie módu, označíme parameter *disable_menu***.**

range_from, range_to - parametre udávajúci rozsah stupnice prístroja a tým aj rozsah výstupných hodnôt. Ak nebude vstupná hodnota v rozsahu prístroja, nastaví sa medzná hodnota rozsahu**.**

low_limit, high_limit - dva parametre, pre nastavenie limitov. Ak je hodnota menšia ako *low_limit*, použije sa farba uvedená v parametri *colors_low_limit,* ak presiahne hodnota *high_limit*, použije sa farba uvedená v parametri *color_high_limit***.** V medziach medzi limitami sa hodnota vykresluje farbou *color_value*. V reálnych meracích systémoch (a rovnako aj tu) sa tieto parametre používajú napríklad u pece pre nastavenie teplotného rozsahu (teplotnej hysteréze).

dec_places - počet zobrazovaných desatinných miest na číselnom zobrazovači.

real_step - krok pre nastavenie hodnôt. Jeho veľkosť udáva prírastok alebo úbytok hodnoty inkrementálnych a dekrementálnych šípok.

history - udáva počet pamätaných a zobrazených hodnôt v grafe. Je dĺžkou historického trendu**.**

content - obsažnosť vzhľadu prístroja. Ponúka tri možnosti:

min - minimálne zobrazenie prístroja

med - k základnému vzhľadu prístroja sa pridá napríklad digitálny číselník, stupnica alebo rámik (podľa módu prístroja)

max - maximálna podoba prístroja. Oproti *med* sú pridané prvky pre nastavenie limitov**.**

h_grid, v_grid - počet čiar pre vodorovný a zvislý raster

V našej úlohe nastavíme parametre:

meter = Voltmeter

mode = pointer
expression = Signál
low_limit = 25
high_limit = 75
range_from = 0
range_to = 100
dec_places = 1

Riešenie – prístroj *control*

Prístroje typu *control* vysielajú hodnotu numerického výrazu do premennej, ktorej hodnota závisí na polohe ovládacieho prvku k maximálnemu rozsahu prístroja.

Dôležité parametre prístroja *meter*:

output - už známy parameter. Jeho obsahom je názov premennej, kde bude hodnota nastavená prístrojom vysielaná.
receivers - zoznam príjemcov správ (veľmi dôležitý parameter).
mode - určuje grafickú podobu prístroja
knob - tvar prístroja má podobu gombíka
turn - prístroj má podobu ikony gombíka s maximálnym zobrazením (*content* = *max*)
Oba módy zobrazujú gombík, ktorý má nastavovaciu rysku. Ak klikneme myšou na plochu gombíka, pootočí sa gombík tak, že jeho ryska prechádza bodom kliknutia myši. Odpovedajúcim spôsobom sa zmení aj výstupná hodnota. Ak podržíte ľavé tlačidlo myši stlačené, môžeme pohybom myši plynulo (s nastaveným krokom) meniť výstupnú hodnotu prístroja.
horizontal_slider - ťahový potenciometer uložený vodorovne
vertical_slider - - ťahový potenciometer uložený zvisle
Oba módy sa ovládajú rovnakým spôsobom. Umiestnite kurzor myši na bežec, stlačte ľavé tlačidlo myši a ťahaním presuňte bežec požadovaným smerom. Ďalšia možnosť je kliknutím na smerové šípky (posun sa uskutoční práve v dĺžke jedného kroku).
count_box - prístroj má podobu numerického okienka so šípkami. Zápis je umožnený buď priamo cez klávesnicu, alebo inkrementáciou a dekrementáciou pomocou šípok
edit_box - prístroj má podobu numerického okienka. Zápis je možný iba cez klávesnicu.

change_box - bezpečnostné zadávanie hodnoty. Používame, ak chceme vylúčiť náhodný zápis novej hodnoty do výstupnej premennej. Na miesto ovládacieho prvku sa zobrazí ikona trezoru, ktorú však môžeme nahradiť vlastnou ikonou v parametri *change_icon*.
Po dvojkliku na plochu ikony sa objaví modálne okno, v ktorom je možné zadať novú numerickú hodnotu pomocou klávesnice.
content - obsažnosť vzhľadu prístroja. Ponúka tri možnosti:
min - minimálne zobrazenie prístroja
med - k základnému vzhľadu prístroja sa pridá niektorá jeho zložka (podľa módu prístroja)
max - maximálna podoba prístroja.
range_from, range_to - parametre udávajúci rozsah prístroja a tým aj rozsah výstupných hodnôt

init_value - nastavenie počiatočnej hodnoty prístroja pri jej inicializácii
real_step - krok zmeny pre nastavenie hodnôt. Veľkosť udáva prírastok alebo úbytok výstupnej hodnoty
dec_places - počet zobrazovaných desatinných miest na číselnom zobrazovači

V našej úlohe nastavíme parametre:

control = Potenciometer
mode = knob
output = Signál
range_from = 0
range_to = 100
dec_places = 1

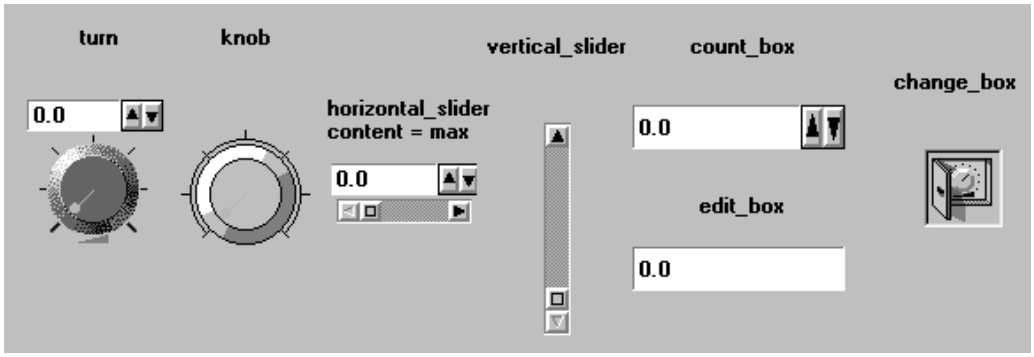
Kritické časti programu:

```
var
  Signál = shortreal, 0;
end_var;

instrument

  window panel Skúšobný_panel;
    owner = background;
    position = 30, 35, 500, 200;
    win_title = 'Skúšobný panel';
  end_panel;

  meter Voltmeter;
    owner = Skúšobný_panel;
    position = 290, 40, 180, 120;
    expression = Signál;
    dec_places = 1;
    real_step = 0.1;
  end_meter;

  control Potenciometer;
    owner = Skúšobný_panel;
    position = 85, 55, 82, 80;
    output = Signál;
    real_step = 0.1;
    dec_places = 1;
    receivers = Voltmeter;
  end_control;
end_instrument;
```

14.2 Modifikácie úlohy č. 14

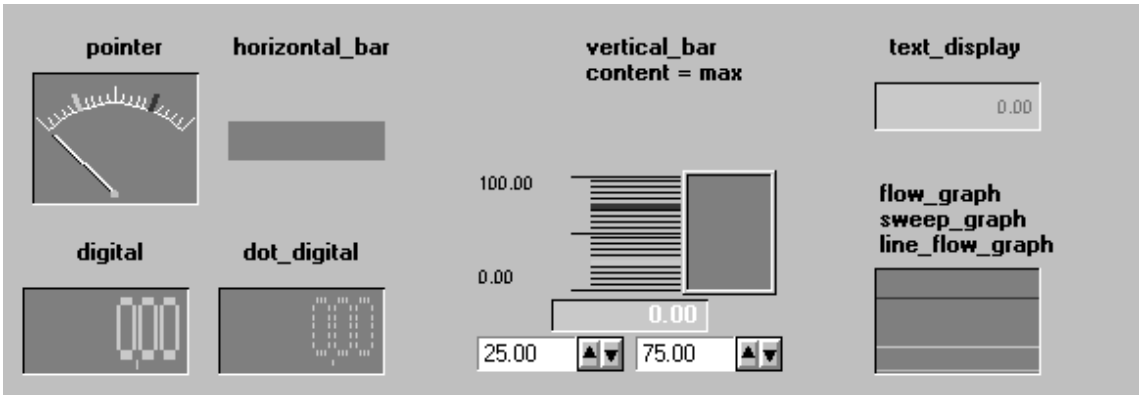
	**AKTIVITA** Modifikujte prístroj *control* a *meter* z úlohy č. 14. Vyskúšajte všetky varianty parametra *mode*.

Modifikácie prístroja *control*



Pomocou parametra *mode*, *content*, *range_from*, *range_to*, *init_value*, *real_step* a *dec*_places preverte možnosti prístroja.

Modifikácie prístroja *meter*



Pomocou parametra *mode*, *content*, *mask*, *disable_menu*, *range_from*, *range_to*, *low_limit*, *high_limit*, *real_step*, *history*, *h_grid*, *v_grid* a *dec*_places preverte možnosti prístroja.

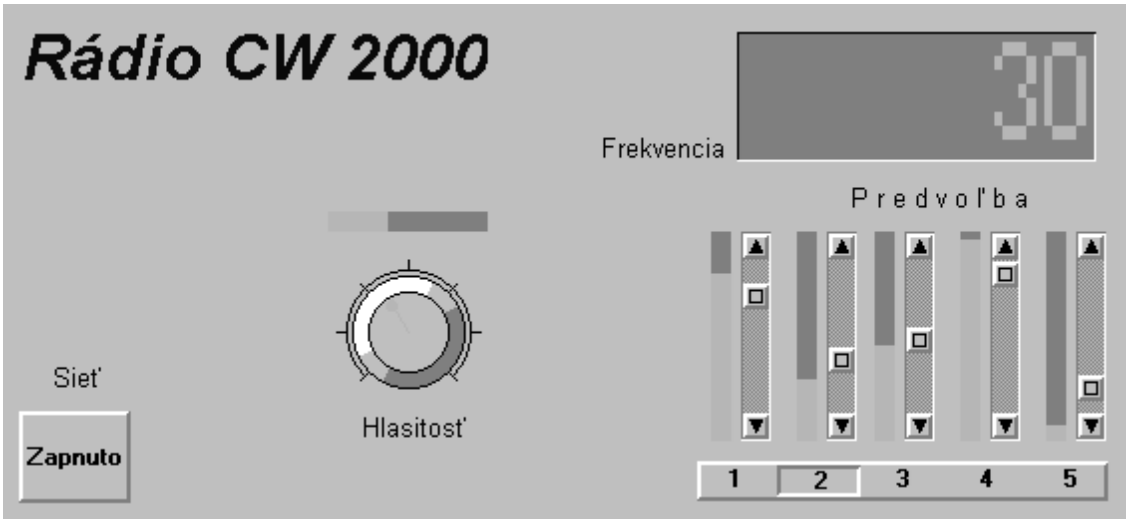
14.3 Úloha č. 15

	**AKTIVITA** Simulujte funkciu autorádia s predvoľbami staníc.

Rozbor úlohy

Autorádio sa skladá z týchto častí:

- sieťového vypínača
- regulátora hlasitosti
- indikátora hlasitosti
- ladiaci prvky predvoľby 1 až 5
- indikátory predvoľby 1 až 5
- ukazateľa kmitočtu
- popisky panela
- panela



Riešenie – premenné

Úloha vyžaduje niekoľko premenných. Pre sieťový vypínač použijeme jednoduchú premennú typu boolean, pre hlasitosť jednoduchú reálnu premennú. Údaje o stave prepínača predvoľby ponesie premenná pole s piatimi položkami typu boolean. Stav ovládacích prvkov predvoľby ponesie pole s piatimi prvkami typu real.

```
var
  Sieť = boolean, false;
  Hlasitosť = real, 0;
  Prepínač = array[ 1..5 ] of boolean, false;
  Predvoľba = array[ 1..5 ] of real, 0;
end_var;
```

Riešenie – sieťový vypínač

Sieťový vypínač, tak ako u skutočného rádia, privádza napätie do autorádia. V našej simulácii bude svojou funkciou ovládať ukazovateľ kmitočtu. Bez zapnutia vypínača sa nesmie zobraziť kmitočet predvolenej stanice na dispeji.

```
switch Sieť;
  output = Sieť;
```

```
  mode = text_button;
  true_text = 'Zapnuto';
  false_text = 'Vypnuto';
  receivers = Frekvencia;
end_switch;
```

Riešenie – hlasitosť

Časť programu pre ovládanie hlasitosti sa skladá z regulátora a indikátora hlasitosti. Táto časť je programovaná štandardným, už známym spôsobom, a nie je nutné ju podrobne rozvádzať. Pre indikáciu zvoľme mód prístroja *horizontal_bar*.

```
meter IndikátorH;
  expression = Hlasitosť;
  mode = horizontal_bar;
  low_limit = 0;
  high_limit = 100;
end_meter;

control Hlasitosť;
  output = Hlasitosť;
  receivers = IndikátorH;
end_control;
```

Riešenie – prepínač pre predvoľbu

Prepínač predvoľby je tvorený prístrojom *multi*_switch v módu *menu*_bar. To, čo je na definícii prístroja zaujímavé, je časť *text*. Prístroj totiž automaticky volí šírku na základe veľkosti textov v zložke *item*. Aby sme sa „trafili“ do šírky medzi predvoľbami, doplníme do textu s číslami medzery. V časti receivers bude Prepínač ovládať predvoľby 1 až 5.

```
multi_switch Prepínač;
  mode = menu_bar;
     receivers = Predvoľba_1, Predvoľba_2, Predvoľba_3, Predvoľba_4, Predvoľba_5;
  item
    text = '  1   ';
    output = Prepínač[ 1 ];
  end_item;
  item
    text = '   2   ';
    output = Prepínač[ 2 ];
  end_item;
  item
    text = '   3   ';
    output = Prepínač[ 3 ];
  end_item;
  item
    text = '   4   ';
    output = Prepínač[ 4 ];
  end_item;
```

```
    item
      text = '   5  ';
      output = Prepínač[ 5 ];
    end_item;
  end_multi_switch;
```

Riešenie – predvoľba 1 až 5

Predvoľbu tvoria prístroje *meter* a *con*trol v prevedení *vertical*. Aj tato časť je vytvorená štandardným spôsobom. Prístroj *control* bude ovládať jak vlastný indikátor predvoľby, tak frekvenciu na displeji.

```
  meter Predvoľba_1;
    expression = Predvoľba[ 1 ];
    mode = vertical_bar;
    low_limit = 0;
    high_limit = 100;
  end_meter;

  control Predvoľba_1;
    output = Predvoľba[ 1 ];
    mode = vertical_slider;
    receivers = Predvoľba_1, Frekvencia;
  end_control;
```

Riešenie – prístroj *multiplexer*

Prístroj *multiplexer* je veľmi podobný prístroji *meter*. To, v čom sa líši, je možnosť výberu vstupných výrazov na základe podmienky. Zameriame sa na tieto odlišnosti:

input_item – skladá sa so zložky *condition* pre stanovení podmienky na vstup zložky *expression* do prístroja. Pokiaľ podmienka nebude splnená, zložka *expression* sa neuplatní. Celkový počet zložiek *item* nie je obmedzený.
default_input –pokiaľ nebude žiadna so zložiek *item* vybratá, zobrazí sa hodnota tejto premennej
show_inputs – ak je prečiarknutá, zobrazia sa symboly vstupov
show_numbers – ak je prečiarknutá, zobrazia sa čísla vstupov
disable_menu – zrušenie menu, ak je prístroj v okne.

Multiplexer Frekvencia obsahuje päť zložiek *item.* Podmienka, pre výstup príslušnej premennej Predvoľba je daná v podmienke *condition*. Je zrejmé, že nastavenú frekvenciu predvoľby prístroj zobrazuje len pri vybratí príslušnej sekcie prepínača a zapnutom sieťovom vypínači. Ďalšou zaujímavosťou je použitie *default_input*. Ak vypneme sieťový vypínač, žiadna z podmienok v zložke *item* nebude splnená. A práve hodnota *default_input=0* zaistí „vynulovanie“ displeja.

```
  multiplexer Frekvencia;
    input_item
      condition = Sieť and Prepínač[ 1 ];
```

```
  expression = Predvoľba[ 1 ];
end_input_item;
.............
input_item
  condition = Sieť and Prepínač[ 5 ];
  expression = Predvoľba[ 5 ];
end_input_item;
default_input = 0;
mode = digital;
low_limit = 0;
high_limit = 100;
history = 50;
dec_places = 0;
end_multiplexer;
```

14.4 Úloha č. 16

AKTIVITA

Simulujte funkciu prístrojovej dosky automobilu Favorit.

Rozbor úlohy

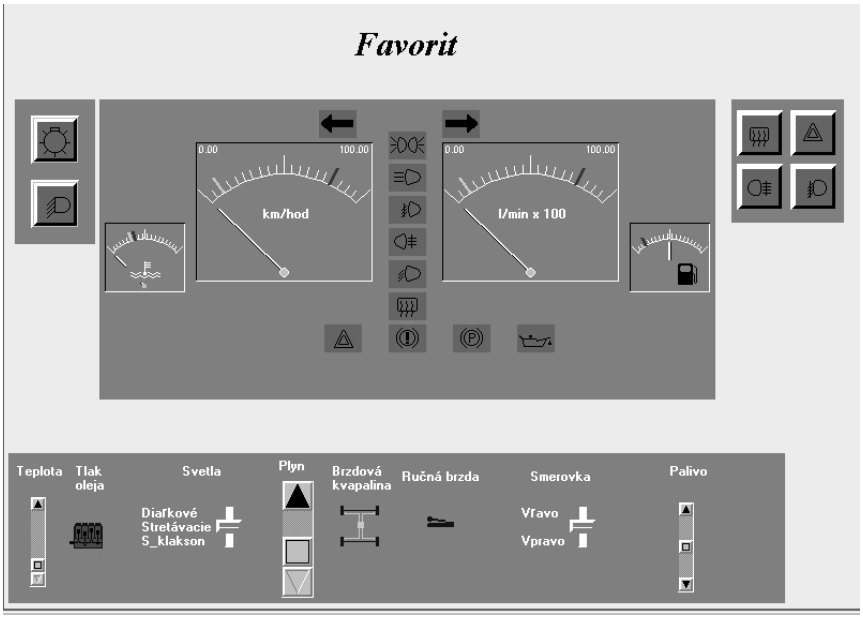
Prístrojová doska osobného automobilu Favorit sa skladá z týchto prvkov:

- ukazovateľ teploty chladiacej vody motoru (meter Teplota)
- ukazovateľ rýchlosti (meter Rýchlosť)
- ukazovateľ obsahu palivovej nádrže (meter Nádrž)
- ukazovateľ kritického množstva palivovej nádrže (indicator Rezerva)
- smerovka vľavo (indicator S_Vľavo)
- smerovka vpravo (indicator S_Vpravo)
- stav ručnej brzdy (indicator Brzda)
- ukazovateľ parkovacích svetiel (indicator Parkovacie)
- ukazovateľ stretávacích svetiel (indicator Potkávacie)
- ukazovateľ diaľkových svetiel (indicator Ďiaľkové)
- ukazovateľ hmlových svetiel (indicator Hmla)
- ukazovateľ tlaku oleja v motoru (indicator Olej)
- ukazovateľ výstrahy (indicator S_Výstraha)

Ovládacie prvky, ktoré majú vplyv na ukazatele prístrojovej dosky:

- teplomer chladiacej vody (control Teplota)
- spínač tlaku oleja (switch Olej)
- prepínač diaľkových svetiel, stretávacie svetla, sveteľná húkačka (multi_switch Svetlo)
- pedál plynu (control Plyn)
- stav hladiny brzdovej kvapaliny (switch B_kvapalina)
- ručná brzda (switch Brzda)

- smerové svetla (multi_switch Smerovka)
- spínač paliva (control palivo)
- spínač vyhrievania zadného okna (switch_label Okno)
- spínač výstrahy (switch Výstraha)
- spínač zadných hmlových svetel (switch_label Z_Hmla)
- spínač predných hmlových svetiel (switch Hmla)
- spínač parkovacích svetiel (switch ParkovacieTlačítko
- spínač tlmených svetiel (multi_switch TlmenéTlačítko)



V popise riešenia bude upozornené iba na tie časti, ktoré zásadným spôsobom modifikujú daný prístroj.

Riešenie – otáčkomer a rýchlomer

Plynový pedál je simulovaný prístrojom *control* v prevedení *vertical_slider*. To, čo je v demonštrácii zaujímavé, je ovládanie otáčkomera. Otáčkomer preberá hodnotu z prístroja „plynový pedál“ upravenú koeficientom konštanty *Otáčky*.

Konštanta je definovaná v záložke *Dátoví inšpektori,* položka *Konštanty*. Je to teda aplikácia na využitie konštanty v projekte.

Kritická časť programu:

```
const
  Otáčky = 0.8;
end_const;

meter Rýchlosť;
   owner = Prístrojová_doska;
   position = 320, 40, 165, 130;
   expression = Plyn * Otáčky;
   low_limit = 10;
```

```
    high_limit = 80;
  end_meter;
```

Riešenie – stav palivovej nádrže

Stav palivovej nádrže je strážený dvomi spôsobmi:

- ručičkový prístroj ukazuje pomer celej nádrže k množstvu paliva
- svetelný indikátor indikuje kritické množstvo paliva v nádrži.

Prístroj *Rezerva* je ovládaný logickou hodnotou, tvorenou výsledkom porovnaní *Palivo <= 15*. Premenná, ani číslo, nie sú logické hodnoty, ich spojením do logickej rovnice získame logický výsledok.

Kritická časť programu:

```
  indicator Rezerva;
    owner = Prístrojová_doska;
    position = 530, 150;
    expression = Palivo <= 15;
    true_icon = 'Svetlo13b.ICO';
    false_icon = 'Svetlo13a.ICO';
    transparent;
  end_indicator;

  meter Palivo;
    owner = Prístrojová_doska;
    position = 495, 115, 75, 65;
    expression = Palivo;
    low_limit = 15;
    high_limit = 15;
  end_meter;
```

Riešenie – panel spínačov pomocných svetiel

Tlačidlá pomocných svetiel sú realizované pomocou prístrojov *switch_label*. Ich grafické vyjadrenie sa riadi podľa toho, či sú zopnuté tlačidlá spínajúce svetlá. Stav animujú štyri ikony:
- tlačidlo vypnuté a nesvieti
- tlačidlo zapnuté a nesvieti
- tlačidlo vypnuté a svieti
- tlačidlo zapnuté a svieti

Stav prístroja *switch_label* ovláda tak spolu so spínačmi svetiel výber ikon pre vlastné zobrazenie. Paradoxne preto aktivuje sám seba v zložke *receivers*. Zložka *item* v parametri *expression* rieši všetky štyri stavy.

Kritická časť programu:

```
  switch_label Okno;
```

```
  receivers = Sklo, Okno;
  logic = set_flip_flop;
  item
    icon = 'Tlac6a.ICO';
    expression = not Parkovacie and not Okno;
.....
  end_item;
  item
    icon = 'Tlac6b.ICO';
    expression = Parkovacie and not Okno;
.....
  end_item;
  item
    icon = 'Tlac6c.ICO';
    expression = not Parkovacie and Okno;
.....
  end_item;
  item
    icon = 'Tlac6d.ICO';
    expression = Parkovacie and Okno;
.....
  end_item;
```

Riešenie – ikona *Voda*

Ikona *Voda* je realizovaná pomocou prístroja *multi_label.* Tvar ikony je ovplyvňovaný spínačom svetiel. Pri zapnutí sa ikona „rozsvieti“

Kritická časť programu:

```
multi_label Voda;
  item
    icon = 'Svetlo14a.ICO';
    expression = not Parkovacie;
  end_item;
  item
    icon = 'Svetlo14b.ICO';
    expression = Parkovacie;
  end_item;
end_multi_label;
```

Riešenie – prepínanie svetiel, smerové svetla

Prepínanie stretávacích a diaľkových svetiel respektíve spínanie svetelného klaksónu je prevedené štandardným spôsobom. Zaujímavé je iba nastavenie počiatočného stavu prístroja *multi_switch.*
Prepínač *Svetla* nemá východziu polohu na prvej polohe, ale až na druhej. Je to zabezpečené parametrom *selected* v zložke *item.*

Kritická časť programu:

```
multi_switch Svetla;
  item
    text = 'Diaľkové';
    output = Diaľkové;
  end_item;
  item
    text = 'Stretávacie';
    output = Stretávacie;
    selected;
  end_item;
  item
    text = 'S_klakson';
    output = S_klakson;
  end_item;
end_multi_switch;
```

Záver

Pomocou **Control Web 5** je možné realizovať nielen regulačné a riadiace aplikácie, ale aj simulačné úlohy a trenažéry. Náš prípad prístrojovej dosky Favoritu je toho dôkazom.

To, čo nie je zatiaľ možné vyriešiť pomocou súčasných znalostí **Control Web 5** sú aplikácie odvíjajúce sa v čase. V projekte prístrojovej dosky je to napríklad blikanie smerových svetiel a výstražného trojuholníka. Blikanie je možné síce zaistiť pomocou alternatívnych farieb prístroja (vyskúšajte), ale my sa pokúsime o ďalšie riešenie pomocou programových sekvencií. Preto prerušíme praktické úlohy a pozrime sa opäť na časť teórie.

15 ČASOVANIE PRÍSTROJOV

Úloha č. 1 je riadená udalosťou – zapnutím vypínača. Existuje rad úloh, ktoré je potrebné časovať. Uveďme príklad výrobnej automatickej linky, na ktorej sa musí predmet nachádzať v určitom čase na určitom mieste. Ďalším, veľmi výrazným aspektom aplikácie je použitie časovačov pri periodickom snímaní dôležitých vstupných veličín.

Aplikácie môžu byť veľmi rozsiahle a okamih, kedy si počítač pre daný prístroj vyhradí strojový čas, sa v časovom horizonte líši. Výsledkom nesynchrónneho režimu býva často „stratenie“ informácie alebo omeškanie procesu, čo vedie k nestabilnosti systému a dokonca i k havárii.

Ak takým nestabilnosťám chceme predísť, zvýšime bezpečnosť aplikácie synchronizáciou. Pokiaľ sa budeme venovať bezpečnému snímaniu informácie vstupnej veličiny, potom časový interval periodického snímania musí byť 10x až 20x kratší ako najkratší časový interval periódy snímanej veličiny.

Než sa začneme venovať časovaniu, je nutné si pripomenúť, že časovanie obecne podstatným spôsobom zvyšuje nároky na výkonnosť počítačov. Preto časovanie použite iba v odôvodnených prípadoch, kde udalostná procedúra môže viesť k nestabilnosti systému. Veľkosť časovej periódy voľte v optimálnej veľkosti. Príliš krátka perióda je zbytočná a často vedie k sklzu aplikácie (meškanie periódy časovania).

15.1 Relatívne časovanie

Základným spôsobom ako časovať je časovanie časovou periódou. To znamená, že prístroj je aktívny vždy po uplynutí zvolenej časovej periódy **bez vzťahu k aktuálnemu času**. Systém zaisťuje iba periodicitu, nič viac. Pretože počiatok časovania sa vždy relatívne vzťahuje k okamihu štartu, hovoríme o relatívnom časovaní.

Prístroj bude relatívne časovaný, ak jeho parameter *timer* bude mať ako hodnotu číslo alebo konštantu udávajúcu čas periódy v sekundách. Parameter *receivers* (napr. u switch) je nutné odstrániť.

Perióda časovania môže byť udaná v rozmedzí od nuly do „nekonečna“. Nulová perióda má špecifický význam – prístroj bude časovaný maximálnou možnou rýchlosťou systému a jeho veľkosť bude ovplyvňovaná momentálnym stavom celého operačného systému. Nie je vhodné používať takúto periódu. Perióda „nekonečno“ sa ako parameter zapisuje kľúčovým slovom *infinite* a má takisto špecifický význam – prístroj bude spustený iba jedenkrát pri štarte aplikácie.

Znova si pripomeňme, že relatívne časovaný prístroj ignoruje absolútny čas. Za počiatok časovania je stanovený okamihom štartu aplikácie, a preto nie je možné aktivovať prístroj v rovnakých okamihoch absolútneho času (napríklad každú celú hodinu). Aj keby sa to podarilo, tak zmena systémového času všetko pokazí.

15.2 Úloha č. 17

AKTIVITA

Modifikujte úlohu č.1 na relatívne časovanie.

Riešenie – nastavenie relatívneho časovania hodnotou

Pred nastavením časovania úlohy odstráňte z prístroja Vypínač_1 z parametra *receivers* odkaz na Žiarovka_1. Dosiahnete to ľahko – kliknutím na uvedený prístroj v pravom okne a stlačením tlačidla s preškrtnutým priečinkom (odstrániť). Ak odpoviete na otázku, či chcete zmazať označený receivers kladne a zatvoríte Inšpektor prístroja s potvrdením zmeny, tak od tejto chvíle nebude prístroj Žiarovka_1 aktivovaný udalosťou z Vypínač_1.

V inšpektorovi prístroja Žiarovka_1 nastavte parameter *timer* na hodnotu 2 (dve sekundy) jednoduchým zápisom do položky Časovač alebo časový krok. Inšpektor uzatvorte a spustite úlohu. Čo zistíte? Keď zmeníme polohu vypínača, žiarovka túto zmenu vyhodnotí s maximálne dvoj sekundovým oneskorením. Prečo? Časovanie žiarovky síce prebieha v stanovenom časovom intervale a prístroj v takých intervaloch sleduje stav premennej na svojom vstupe (expression), no okamih zmeny stavu vypínača určujeme sami a je časovo nezávislý. Výsledkom bude, že sa so zmenou stavu vypínača stretneme niekde medzi minimálnym (teoreticky nulovým meškaním) a maximálnym, teda dvoj sekundovým intervalom.

Kritické časti programu:

```
switch Vypínač;
   owner = Skúšobný_panel;
   output = Vypínač_1;
 end_switch;

 indicator Žiarovka;
   timer = 2;
   owner = Skúšobný_panel;
   expression = Vypínač_1;
 end_indicator;
```

Riešenie – nastavenie relatívneho časovania konštantou

Ak modifikujete úlohu č. 1, odstráňte položku *receivers* ako v 5.2.1.

Prejdite do záložky Dátoví inšpektori (3.1.3) do zložky *Konštanty*. Za parameter *Meno* napíšte názov konštanty, napr. *Čas*. V položke *Hodnota* zadajte časový krok v sekundách, napr. 2 .

V inšpektorovi prístroja Žiarovka, v položke *timer* otvorte ponuku a vyberte práve konštantu *Čas*. Úlohu spustite.

Zistíte, že úloha pobeží s oneskorením dvoch sekúnd rovnako, ako úloha v 5.2.1. Rozdiel je iba v tom, že časový krok je stanovený konštantou, ktorú môže využívať niekoľko prístrojov súčasne. Ak neskôr zistíme, že časový krok bol stanovený nesprávne, jednoducho prepíšeme konštantu a v celom programe, kde sa toto časovanie využíva, sa zmena uskutoční. Používaním relatívneho časovania predídeme mnohým neskorším problémom a navyše táto technika patrí medzi základy modulárneho programovania.

Kritické časti programu:

```
const
  Čas = 2;
end_const;


switch Vypínač;
    owner = Skúšobný_panel;
    position = 235, 65, 46, 46;
    output = Vypínač_1;
  end_switch;

  indicator Žiarovka;
    timer = Čas;
    owner = Skúšobný_panel;
    position = 45, 65;
    expression = Vypínač_1;
  end_indicator;
```

15.3 Absolútne časovanie

Jednoduchým rozšírením relatívneho časovania vznikne **časovanie absolútne**, alebo inak povedané **časovanie s posunutím.** Parameter *timer* prístroja bude obsahovať dve časové hodnoty oddelené čiarkou, napr.

```
timer = 60,10;
```

Prvé číslo definuje periódu ako pri relatívnom časovaní, druhé posunutie. Časovanie s periódou sa vzťahuje k skutočnému absolútnemu času. Počiatočný okamih všetkých období je stanovený na polnoc dňa, kedy bola aplikácia spustená, posunutie určuje omeškanie začiatku prvej periódy.

V prípade nulového posunutia prvá perióda začína presne o polnoci. Ak stanovíme *timer = 60,0;* , potom interval časovania od polnoci bude po jednej minúte. Časovanie *timer = 259200,0;* znamená „prvý krok o polnoci a ďalší nasledujúci každú tretiu polnoc“.

Nenulové parametre celý mechanizmus časovania posunie o udaný interval vpred. Časovanie *timer = 60,10;* znamená, že prvý časový krok je 10 sekúnd po polnoci a ďalší nasleduje po minúte, teda v 00:01:10, 00:02:10, ...

Vzťah periódy a posunutia je obmedzený iba prirodzenou podmienkou, že posunutie nemôže byť väčšie ako perióda. Pokiaľ taký posun použijete, výsledkom posunu bude zvyšok po delení periódou.

Absolútne časované prístroje môžu byť použité napríklad k zobrazovaniu behu aplikácie. Väčšie množstvo relatívne časových prístrojov prevádza svoju činnosť v skupinách, v medziobdobiach period sa opäť nič nedeje. Môže byť preto rozumné prístroj pomocou absolútneho časovania rozložiť posunutím v ich časových intervaloch tak, aby bol výsledný beh plynulejší – častejšie sa budú vykonávať menšie množstvá prístrojov.

V súvislosti s absolútnym časovaním majte na pamäti časové variácie spôsobené prepínaním prevádzajúcich tokov. Predstavte si prístroj s časovačom 1,0 , ktorý zobrazuje sekundy skutočného času. Taký prístroj by mal ukazovať každú sekundu novú hodnotu. No vždy tomu tak nebude, milisekundové variácie period sa orezaním prejavia ako sekundová chyba. Je preto potrebné s týmto mechanizmom rátať. Ak chcete zaistiť správne zobrazovanie sekúnd v prístroji, ktorý ukazuje v aplikácii reálny čas, použite časovač s rezervou – napríklad 1,0.1 . Ak chcete do archívu zapísať skutočne absolútnu novú hodnotu, posuňte archiver kúsok dopredu – použite napríklad časovanie 3600,1.

A na záver teórie: možno ste si všimli, že ak je potrebné zadať desatinné číslo, uvádzame desatinnú bodku a nie desatinnú čiarku.

15.4 Úloha č. 18

AKTIVITA

Modifikujte úlohu č.1 na absolútne časovanie.

Riešenie – nastavenie absolútneho časovania hodnotou

Odstráňte hodnotu parametra receivers v prístroji *Žiarovka_1*. V inšpektorovi prístroja Žiarovka_1 nastavte parameter *timer* na hodnotu 5 a *offset* na hodnotu 1 (jedna sekunda). Inšpektor uzatvorte a spustite úlohu.

Reakcia na zmenu stavu vypínača bude: prístroj *Žiarovka*_1 sa bude aktivovať každých päť sekúnd reálneho času s jedno sekundovým oneskorením.

Kritické časti programu:

```
switch Vypínač;
   owner = Skúšobný_panel;
   position = 235, 65, 46, 46;
   output = Vypínač_1;
 end_switch;
```

```
indicator Žiarovka;
  timer = 5, 1;
  owner = Skúšobný_panel;
  position = 45, 65;
  expression = Vypínač_1;
end_indicator;
```

Riešenie – nastavenie absolútneho časovania konštantou

Prejdite do záložky Dátoví inšpektori (3.1.3) do zložky *Konštanty*. Za parameter *Meno* napíšte názov konštanty, napr. *Čas*. V položke *Hodnota* zadajte absolútny časový krok v sekundách, napr. 5. Ďalšiu konštantu nazvite napríklad *Posun* s hodnotou 1.

V inšpektorovi prístroja Žiarovka, v položke *timer* otvorte ponuku a vyberte konštantu *Čas* a v položke offset nastavte konštantu *Posun.* Úlohu spustite.

Zistíte, že prístroj *Žiarovka*_1 bude aktivovaný každú piatu sekundu reálneho času s oneskorením jednej sekundy ako v príklade 5.4.1. Rozdiel je iba v tom, že časový krok a posun je stanovený konštantou, ktorú môže využívať niekoľko prístrojov súčasne. Ak neskôr zistíme, že časový krok či posun bol stanovený nesprávne, jednoducho tieto konštanty prepíšeme. V celom programe, kde sa toto časovanie využíva, sa zmena prevedie.

Kritické časti programu:

```
const
  Čas = 5;
  Posun = 1;
end_const;

  switch Vypínač;
    owner = Skúšobný_panel;
    position = 235, 65, 46, 46;
    output = Vypínač_1;
  end_switch;


  indicator Žiarovka;
    timer = Čas, Posun;
    owner = Skúšobný_panel;
    position = 45, 65;
    expression = Vypínač_1;
  end_indicator;
```

15.5 Nepriame časovanie

Relatívne a absolútne časovanie prístroja prevádzajú svoje časové kroky v rytme, ktorý im prikazuje časovací tok aplikácie. Sú časované priamo samotným systémom.

Control Web 5 definuje ešte aj iný spôsob časovania - nepriamy, kde prístroje prevádzajú svoje časové toky vo vnútri časového kroku iného logicky nadriadeného prístroja. Prístroje, ktoré môžu zastrešovať pri časovaní iné prístroje, sa nazývajú časovače a spôsob časovania prístrojov s využitím časovačov **časovanie časovačom.**

Princíp je jednoduchý - časovač vo svojom časovom kroku prechádza jednotlivé podriadené prístroje a podľa určitých pravidiel im povolí uskutočniť ich časové kroky. Prístroj časovaný časovačom obsahuje vo svojom parametri *timer* meno príslušného časovača a prípadne poradie časovania.

Všetky časovače môžu obsahovať viac prístrojov, prípadne i časovačov. Podriadené prístroje sú v časovačoch usporiadané za sebou v sekvencii, ktorú časovače striktne dodržujú. Aktivácia prístrojov je postupná a pokiaľ prístroj neukončí svoju činnosť, nie je aktivovaný ďalší. Táto sekvenčnosť je vlastná všetkým časovačom a je tou hlavnou, čo odlišuje periodické časovanie od časovania časovačom.

Control Web 5 ponúka tri možnosti časovačou - *sequencer, selector a iterator.*

15.6 Nepriame časovanie prístrojom *sequencer*

Sequencer je základný systémový prístroj, ktorý umožňuje časovanie iných prístrojov. Patrí medzi neviditeľné prístroje, preto sa nám nepodarí umiestniť ho na plochu. Dokonca ho ani nie je možné umiestniť do časti Neviditeľné v okne Vzhľad (vľavo od pracovnej plochy).
Pre umiestnenie sequenceru použijeme okno Časovanie. Pomocou ikony maximalizácie okno zväčšíme a pretiahnutím prístroja z Palety prístrojov na vetvu Časované umiestnime prístroj do projektu. Inšpektor prístroja otvárame priamo z okna Časovanie. V parametre *timer* stanovíme časový krok časovača. Všetky prístroje, ktoré budú mať v parametri *timer* zadaný názov časovača, budú týmto prístrojom periodicky časované. Poradie časovania prístrojov môžeme ovplyvniť uvedením čísla poradia za názvom časovača v parametri *timer.* Napríklad prístroj, ktorý má hodnotu *timer=Časovač,2* bude časovaný až druhý v poradí.

15.7 Úloha č. 19

AKTIVITA

Vytvorte panel s vypínačom a piatimi žiarovkami, ktoré budú časované časovačom s periódou 2 sekundy. Poradie aktivácie bude Svetlo1, Svetlo2, Svetlo3, Svetlo4, Svetlo5.

Riešenie - *sequencer*

Základom úlohy je sequencer *Časovač*, s periódou časovania *timer*=2. Žiarovky a vypínač vytvoríme štandardným spôsobom. V prístroji *Vypínač* nebudeme naplňovať parameter *receivers.* Ten ostane prázdny. Do parametra *timer* prístrojov Svetlo1 až Svetlo5 naplníme názov seqenceru *Časovač*. Za ním, za čiakou, uvedeme poradie aktivácie prístroja.

Kritické časti programu:

```
  sequencer Časovač;
    timer = 2;
  end_sequencer;

end_timer;


instrument

  switch Vypínač;
    output = Vodič;
    mode = text_button;
  end_switch;

  indicator Svetlo4;
    timer = Časovač, 4;
    expression = Vodič;
.............
  end_indicator;

  indicator Svetlo5;
    timer = Časovač, 5;
.............
    expression = Vodič;
  end_indicator;

  indicator Svetlo3;
    timer = Časovač, 3;
.............
    expression = Vodič;
  end_indicator;

  indicator Svetlo2;
    timer = Časovač, 2;
.............
    expression = Vodič;
  end_indicator;

  indicator Svetlo1;
    timer = Časovač, 1;
.............
```

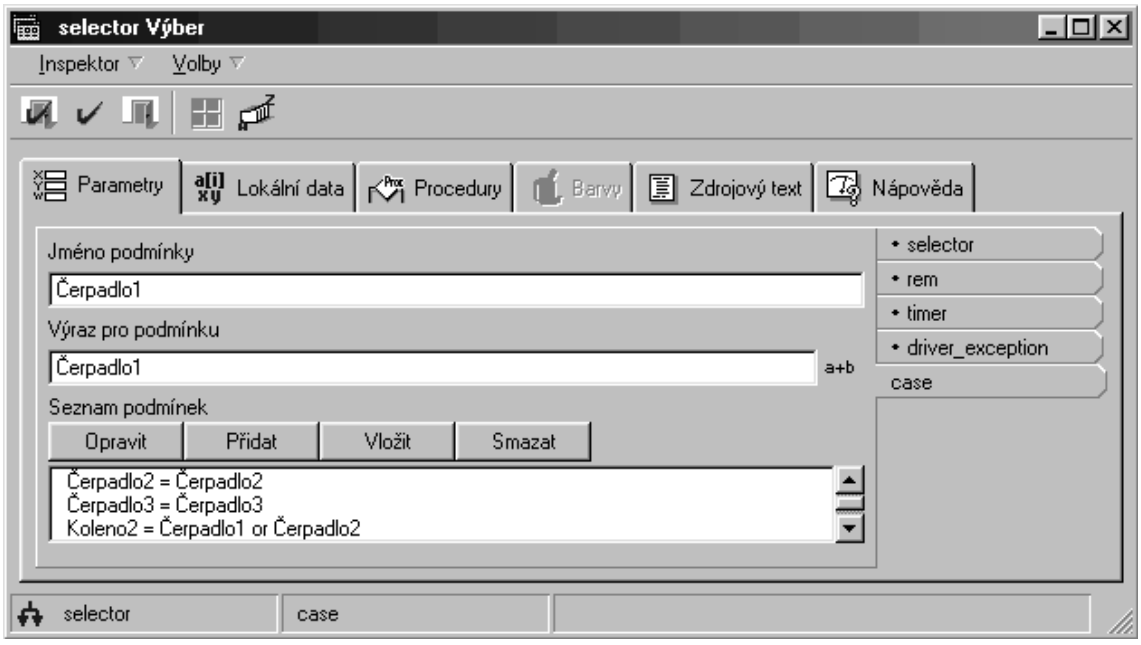
```
    expression = Vodič;
  end_indicator;

end_instrument;
```

15.8 Nepriame časovanie prístrojom *selector*

Už na začiatku kapitoly o časovaní prístrojov bolo zdôraznené, že zbytočne časované prístroje kladú zvýšené nároky na rýchlosť počítača. Vezmime do úvahy rozsiahly projekt, ktorý má niekoľko panelov a na každom z nich je umiestnených niekoľko časovaných prístrojov. Ak máme dodržať prehľadnosť aplikácie, bude nutné v danom čase nevýznamné panely skryť. No skryté prístroje budú v takej aplikácii naďalej časované. Práve tento stav vedie k nehospodárnemu využívaniu strojového času počítača. Omnoho efektívnejšie by bolo skryté prístroje jednoducho nečasovať. Túto úlohu môže na seba vziať prístroj *selector*.

Selector na základe podmienky zapína alebo vypína časovanie prístrojov uvedených v takzvanej selekcii, vetvi. Týchto vetví môže selector obsahovať niekoľko. Za behu aplikácie je každá vetva vyhodnocovaná osobitne logickým výrazom. V prípade skrývajúcich sa panelov môže informácie „schovaj panel“ zastaviť časovanie v ňom obsiahnutých prístrojov.

Prístroj selector umiestňujeme do stromu časovania ako časový prístroj. Stanovenie podmienok vo vetvách sa nastavujú v *Inšpektorovi prístrojov*, v kapitole *case*.



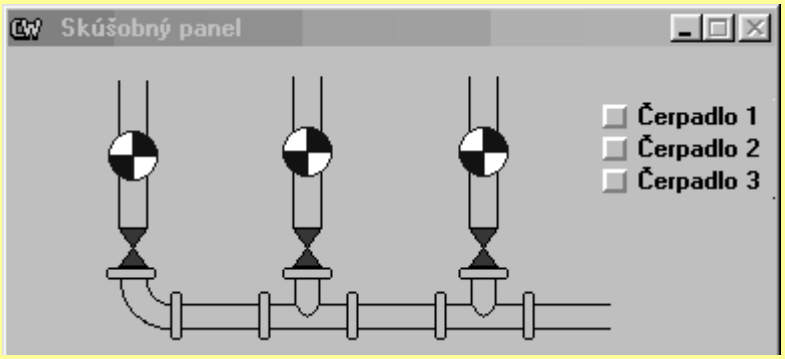
Každá vetva začína menom podmienky nasledovaná logickým výrazom. Ak sú stanovené tieto parametre, vytvoríme vetvu pridaním do zoznamu podmienok.

Prístroje časované selectorom majú v parametri *timer* uvedený názov selectoru spolu s menom podmienky, napr. *Výber.Čerpadlo1*.

15.9 Úloha č. 20

AKTIVITA

Vytvorte model potrubia s tromi odbočkami demonštrujúci čerpanie a pohyb kvapaliny. Čerpadlá zapínajte nezávislými spínačmi. Ak je zapnutý spínač, animujte otvorenie ventilu a pohyb čerpadla a kvapaliny.



Riešenie – premenné a prístroj *multi_switch*

V dátovom inšpektorovi definujte premenné *Čerpadlo1, Čerpadlo2 a Čerpadlo3* typu boolean. Na štandardne vytvorený panel umiestnite prístroj *multi_switch.* V Inšpektorovi prístroja zadajte meno *Rozvádzač* a v *item* definujte tri zložky s názvom Čerpadlo1, Čepadlo2 a Čerpadlo3 spolu a premennými rovnakého názvu. Do zložky *receivers* neskôr zapíšte všetky použité prístroje (až budete poznať ich názvy). Poslednú vetu treba vyznačiť výkričníkmi (**!!!**), lebo práve toto je zdrojom mnohých problémov pri ladení projektu. Prístroj bude typu nezávislé spínače (*mode = check_box*).

Kritické časti programu:

```
var
  Čerpadlo1 = boolean, false;
  Čerpadlo2 = boolean, false;
  Čerpadlo3 = boolean, false;
end_var;

  multi_switch Rozvádzač;
    owner = Skúšobný_panel;
    mode = check_box;
    receivers = Čerpadlo1, Čerpadlo2, Čerpadlo3, Koleno1, Koleno2, Koleno3, Trúbka1,
Trúbka2, Trúbka3, Trúbka4, Trúbka5, Trúbka6, Trúbka7, Trúbka8, Trúbka9, Ventil1,
Ventil2, Ventil3;
    item
      text = 'Čerpadlo 1';
      output = Čerpadlo1;
    end_item;
```

```
    item
      text = 'Čerpadlo 2';
      output = Čerpadlo2;
    end_item;
    item
      text = 'Čerpadlo 3';
      output = Čerpadlo3;
    end_item;
  end_multi_switch;
```

Riešenie – prístroj *selector*

Selector pomenujte Výber. Zvoľte časový krok 100 ms. V *case* definujte niekoľko vetví:

- vetve Čerpadlo1, Čerpadlo2 a Čerpadlo3 budú časovať animáciu pohybu čerpadiel, ventilov a pohyb kvapaliny v separátnych častiach potrubia.
- Koleno2 je vetva, ktorá bude animovat pohyb kvapaliny v trúbke prichádzajúci z vetve prvého a druhého čerpadla.
- Koleno3 je vetva pre animáciu pohybu kvapaliny v trúbke spoločnej pre všetky tri čerpadla.

V tejto úlohe bude teda *selector* slúžiť výhradne k animácii.

Kritické časti programu:

```
timer

  selector Výber;
    timer = 0.1;
    case
      Čerpadlo1 = Čerpadlo1;
      Čerpadlo2 = Čerpadlo2;
      Čerpadlo3 = Čerpadlo3;
      Koleno2 = Čerpadlo1 or Čerpadlo2;
      Koleno3 = Čerpadlo1 or Čerpadlo2 or Čerpadlo3;
    end;
  end_selector;

end_timer;
```

Riešenie – prístroj *engine*

Prístroj *engine* zobrazuje symbol motora pre zostavenie technologických schém. Umožňuje animáciu rotácie v oboch smeroch otáčania.

Medzi špecifické parametre patrí:

expression – notoricky známy parameter. Ak bude vyhodnotený logický výraz ako pravda, symbol bude rotovať. Ak nebude výraz uvedený, symbol bude v pokoji.

mode – určuje činnosť prístroja uvedením jedného z názvov módu: rotate_left (motor rotuje doľava) a rotate_right (motor rotuje doprava)
step – uhlový krok pre animáciu rotácie v stupňoch (asi by nebolo vhodné zvoliť krok 360)
timer – časový krok, v ktorom sa vykoná jeden uhlový krok.

V našom príklade bude animáciu rotácie motora riadiť vetva časového prístroja *Výber*, preto v parametri *timer* bude uvedená jedna z týchto vetví.

Kritické časti programu:

```
engine Čerpadlo3;
  timer = Výber.Čerpadlo3, 2;
  expression = Čerpadlo3;
  mode = rotate_right;
  step = 30;
end_engine;
```

Riešenie – prístroj *pipe*

Pipe je prístroj pre zobrazenie symbolu potrubie na zostavenie technologických schém. Prístroj umožňuje animáciu pohybu média v trúbke.

Medzi špecifické parametre patrí:

expression – notoricky známy parameter. Ak bude vyhodnotený logický výraz ako pravda, symbol bude animovat pohyb. Ak nebude výraz uvedený, prístroj bude trvalo neaktívny.
mode – určuje činnosť prístroja uvedením jedného z názvov módu:
pipe_right – vodorovná trúbka s možnosťou animácie zľava doprava
pipe_left - vodorovná trúbka s možnosťou animácie sprava doľava
pipe_up – zvislá trúbka s možnosťou animácie zdola nahor
pipe_down - zvislá trúbka s možnosťou animácie zhora na dol
pipe_horizontal – vodorovná trúbka bez animácie
pipe_vertical – zvislá trúbka bez animácie
content – určuje spôsob vykreslovania trúbky uvedením jedného z názvov
shaded – tieňová kresba
line_shaded – kresba jednoducho tieňovaná pomocou čiar
flat – plochá kresba
flow_step – šírka pohybujúcich sa pásikov pri animácii
no_run_shadow – obmedzenie vykreslovania horného a dolného tieňa pri animácii
run – trúbka v aktívnom stave
stop – trúbka v neaktívnom stave
run_top_shadow – horný tieň v aktívnom stave
run_bottom_shadow – dolný tieň v aktívnom stave
stop_top_shadow – horný tieň v neaktívnom stave
stop_bottom_shadow – dolný tieň v neaktívnom stave
timer – časový krok, v ktorom sa vykoná jeden krok.

V našom pograme bude animáciu pohybu média v trúbke riadiť vetva časového prístroja *Výber*, preto v parametri *timer* bude uvedená jedna z týchto vetví.

Kritické časti programu:

```
pipe Trúbka9;
  timer = Výber.Koleno3, 1;
  expression = Čerpadlo1 or Čerpadlo2 or Čerpadlo3;
  flow_step = 2;
  colors
    run = lblue;
    stop = lgray;
  end_colors;
  blink_colors
    run_bottom_shadow = black;
    stop_top_shadow = white;
    stop_bottom_shadow = black;
  end_blink_colors;
end_pipe;
```

Riešenie – prístroj *valve*

Prístroj *valve* zobrazuje symbol ventilu pre zostavovanie technologickej schémy. Okolie prístroja je priehľadné, takže môže byť umiestnený napríklad nad obrázkom.

Medzi špecifické parametre patrí:

expression – notoricky známy parameter. Ak bude vyhodnotený logický výraz ako pravdivý, aktívny symbol zmení farbu. Ak nebude výraz uvedený, prístroj bude trvalo neaktívny.
mode – určuje orientáciu ventilu uvedením jedného z názvov módu: *valve_horizontal* (vodorovný ventil) a *valve_vertical* (zvislý ventil)
run – ventil v aktívnom stave
stop – ventil v neaktívnom stave

Kritické časti programu:

```
valve Ventil1;
  expression = Čerpadlo1;
  mode = valve_vertical;
end_valve;
```

Riešenie – prístroj *knee*

Prístroj *knee* zobrazuje symbol potrubného kolena pre ohyb, spojovanie a rozdeľovanie potrubia pri zostavovaní technologickej schémy. Okolie prístroja je priehľadné, takže môže byť umiestnený napríklad nad obrázkom.

Medzi špecifické parametre patrí:

expression – notoricky známy parameter. Ak bude vyhodnotený logický výraz ako pravda, aktívny symbol zmení farbu. Ak nebude výraz uvedený, prístroj bude trvalo neaktívny.
mode – určuje typ potrubného kolena uvedením jedného z názvov módu:
knee_left_up – potrubné koleno s prírubou vľavo a hore
knee_left_down – potrubné koleno s prírubou vľavo a dole
knee_right_up – potrubné koleno s prírubou vpravo a hore
knee_right_down – potrubné koleno s prírubou vpravo a dole
knee_left_up_right – potrubné koleno s prírubou vľavo, hore a vpravo
knee_left_down_right – potrubné koleno s prírubou vľavo, dole a vpravo
knee_left_up_down – potrubné koleno s prírubou vľavo, hore a dole
knee_right_up_down – potrubné koleno s prírubou vpravo, hore a dole
knee_left_right_up_down – potrubné koleno s prírubou vľavo, vpravo, hore a dole
content – určuje spôsob vykreslovania potrubného kolena uvedením jedného z názvov
shaded – tieňová kresba
line_shaded – zjednodušená tieňovaná pomocou čiar
flat – plochá kresba
run – koleno v aktívnom stave
stop – koleno v neaktívnom stave
run_top_shadow – horný tieň v aktívnom stave
run_bottom_shadow – dolný tieň v aktívnom stave
stop_top_shadow – horný tieň v neaktívnom stave
stop_bottom_shadow – dolný tieň v neaktívnom stave

Kritické časti programu:

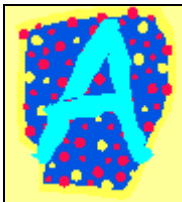
```
knee Koleno1;
  timer = Výber.Čerpadlo1, 4;
  expression = Čerpadlo1;
  mode = knee_right_up;
  colors
    stop = lgray;
  end_colors;
end_knee;
```

15.10 Nepriame časovanie prístrojom *iterator*

Iterator, tak ako predošlé časovače, je základný systémový prístroj, ktorý umožňuje časovanie iných prístrojov. Jeho funkcia je však zložitejšia. K sekvenčnému spracovaniu prístrojov naviac pridáva možnosť ich opakovania – cyklovania. Na počiatku svojho časového kroku najprv vyhodnotí podmienku, či má podriadené prístroje rozbehnúť, či nie. Ak nie je podmienka splnená, prístroje postupne aktivuje. Ak všetky prístroje obvolá, opäť podmienku testuje a v prípade jeho nesplnenia pokračuje v obvolávaniu prístrojov znovu od začiatku. Je zrejmé, že cyklus prístroja v iterátoru musí nejako ovplyvňovať podmienku výskoku. Inak sa môže stať, že sa cyklus stane nekonečným. Celá slučka bude až do splnenia výstupnej podmienky vykonávaná v jedinom časovom kroku. Časovo náročné slučky tak môžu výrazne ovplyvniť časovanie celej aplikácie

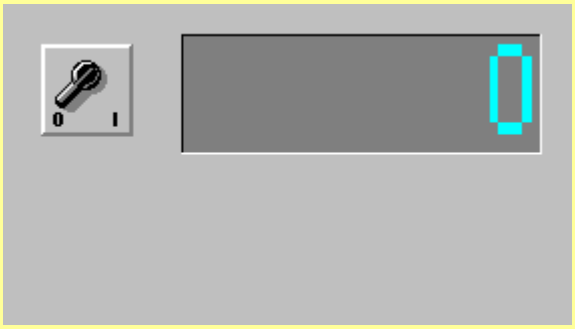
Iterator sa v aplikáciách často nepoužíva, jednoduchší je vytvárať slučky priamo programátorským zápisom vo vnútri nejakej procedúry.

15.11 Úloha č. 21



AKTIVITA

Vytvorte aplikáciu, kde po zapnutí vypínača bude displej maximálnou rýchlosťou pričítať jednotku až do konečného čísla 100.



Riešenie - premenné

Aplikácia bude mať dve premenné. Jedna bude sledovať stav vypínača a v druhej sa bude cyklicky pričítať jednotka.

```
var
  x = real, 0;
  konec = boolean, false;
end_var;
```

Riešenie - vypínač

Vypínač vytvoríme štandardným postupom a umiestnime na panel. Jeho stav bude vysielaný do premennej *koniec*.

```
switch Vypínač;
  owner = panel_1;
  output = konec;
end_switch;
```

Riešenie – prístroj *iterator*

Iterator, ako už bolo povedané, patrí medzi časované prístroje. Preto jeho umiestnenie patrí do sekcie Časovanie v ľavej časti obrazovky. Funkcia prístroja bola popísaná, zameriame sa teda iba na jeho parametre.

timer – časový krok prístroja. S touto periódou testuje *iterator* podmienku, či sa budú prístroje v tzv. iterácii znova aktivovať.
exit – podmienka ukončenia cyklu. Ak je podmienka splnená, *iterator* preruší vykonávanie slučky.

```
iterator Cykl;
  timer = 1;
  exit = ( x >= 100 ) or not konec;
end_iterator;
```

Riešenie – zobrazenie čísla

Zobrazenie výsledku iterácie prevedieme prístrojom *meter*. Jeho definícia bude štandardná. Nóvum je použitie iterátora v parametri *timer*.

Ďalšie nóvum je použitie programovej sekvencie *OnActivate()*. Predbiehame výklad, preto iba krátko: v *Inšpektorovi prístroja* prejdite na záložku *Procedúry* a doplňte matematický vzťah x = x+ 1.

Významu procedúr bude venovaná samostatná kapitola.

Kritická časť programu:

```
meter Zobrazovač;
  timer = Cykl, 1;
  owner = panel_1;
  expression = x;
  mode = digital;
  range_to = 200;
  dec_places = 0;

  procedure OnActivate();
  begin
    x = x + 1;
  end_procedure;

end_meter;
```

Záver

Ak spustíme aplikáciu a prepneme vypínač do polohy „zapnuto“, čítač v rýchlom sledu pričíta jednotku do premennej *x*. Po napočítaní do 100 bude splnená podmienka ukončenia iterácie a funkcia čítača sa zastaví.

16 KANÁLY

Aplikácie systému **Control Web 5** používajú názov **kanály** pre dátové elementy, ktoré majú vzťah k reálnym snímačom alebo riadiacim prvkom. Ak je kanál pripojený na snímač (tlačidlo,...), hovoríme o **vstupnom** kanále, ak je kanál pripojený na riadiaci prvok (motor, ventil,...), hovoríme o **výstupnom** kanále. Vstupné kanály snímajú stav technológie, vysielacie zasahujú do technológie. Vzájomnou väzbou medzi vstupným kanálom – riadiacim softvérom – výstupným kanálom ovládame danú technológiu..

Aplikácie reálneho času, respektíve **jadra v spolupráci s prístrojmi** rozhoduje, ktorý kanál bude v konkrétnom časovom kroku aktivovaný. Časový krok jadra je **elementárna** jednotka behu aplikácie a je preto nutné zaručiť, aby behom neho pracovali všetky prístroje so zhodnými hodnotami kanálov. V opačnom prípade by dva zhodne časované grafy ukázali rôzne hodnoty kanála, a to by bola neprípustná situácia. **Každý vstupný kanál je preto v jednotlivom časovom kroku jadra meraný iba jedenkrát, a to aj v prípade, že je prístrojom požadovaný viackrát**. Existuje však výnimka, kedy vstupný kanál behom jedného kroku nadobudne dve hodnoty. Tá nastáva v prípade, že komunikácia dobehla s oneskorením počas behu aktivácie prístrojov. Prvá hodnota je stará (z predchádzajúceho časového kroku), druhá sa objaví behom prerušenia aktivácie. Dvojaká hodnota kanála sa **nikdy nevyskytne** v časových krokoch, ktoré domerajú všetky dáta pred vypršaním komunikačnej prodlevy.

Odlišným spôsobom sa pracuje s **výstupnými** kanálmi. Prístroje behom svojich aktivácií môžu výstupný kanál rôzne nastavovať v jednom časovom kroku. Jadro všetky požiadavky rešpektuje a hodnota výstupného kanálu sa preto behom jedného časového kroku jadra môže meniť. Po dokončení aktivácie prístrojov je do technológie **zapísaná posledná hodnota výstupného kanálu**. Konkrétne: systém môže behom jedného časového kroku merať aj zapisovať hodnoty do kanálov, pričom vstupné meranie ovplyvňuje hodnoty výstupných kanálov. Meranie aj zápis sa tak môže odohrať v jednom časovom kroku.
Výstupné kanály nemajú vymedzený čas pre ukončenie požiadavky. Ak stlačíte kláves, je táto udalosť spracovaná v okamihu, bez väzby na časované jadro. Prebehne teda **asynchrónne.** Synchrónny zápis vykoná jadro až na kanál.
Pokiaľ asynchrónny zápis vznikne **mimo časový krok**, vykoná sa mimoriadny časový krok, v ktorom sa uskutoční iba komunikácia asynchrónne vzniknutých požiadaviek. Je teda možné navrhnúť aplikácie, ktoré nebudú mať žiadne časované prístroje. Tieto aplikácie boli popísané v kapitolách 3.x

16.1 Kanály a ovládače

Komunikácia kanálov prebehá pomocou ovládačov. Kanály sú dátové elementy, avšak ovládače sú samostatné komponenty, alebo inak povedané malé samostatné programy, ktoré realizujú prenášanie kanálov z a do technológie.

Každý ovládač je spojený s niektorým zariadením, s ktorým dokáže komunikovať. Existujú takisto ovládače, ktoré nepatria ku konkrétnemu zariadeniu, ale k protokolu a je preto možné ich aplikovať pre komunikáciu s rôznymi zariadeniami. Obecne možno povedať, že ovládač vždy komunikuje s niektorým protokolom.

Ovládač je pomocou súboru s parametrami nastavený na určitú množinu dát – vnútorných kanálov – ktoré odpovedajú nejakým spôsobom dátam priamo v technologickom zariadení. Každý vnútorný kanál má svoje číslo – číslo kanálu v ovládači – ktoré sa používa pri definícii kanála v aplikácii. Číslo kanála v ovládači je základná informácia, ktorá sa pri výmene dát medzi jadrom a ovládačom používa k rozlíšeniu kanála.

Ovládač sa skladá z niekoľkých častí:

- súboru DLL. Je to vlastne program ovládača v podobe knižnice.
- Súbor DMF. Textový súbor obsahujúci zoznam všetkých kanálov ovládača Jeho zápis môže vypadať nasledovne:

```
begin
        1  boolean  input
        2  boolean  output
    3 –  9  longreal  bidirectional
   10 – 19  longreal  output
   20 – 34  integer   input
   35 – 48  integer   bidirectional
       49  string          input
       50  string          output
end.
```

Najprv je uvedené číslo kanála, poprípade rozsah zhodných kanálov. Každý kanál musí mať vlastné číslo, ktoré sa nesmie opakovať u iného kanálu rovnakého ovládača. Potom nasleduje typ kanála. Nakoniec je uvedený príznak smeru kanála – pri vstupných kanáloch bude uvedený *input* alebo jeho skratka *in,* u výstupných *output* alebo *out* a u obojsmerných kanálov *bidirectional (bidirect)*. Doporučujem však aby ste skratky nepoužívali z estetického hľadiska.

Súbor PAR. Niektoré ovládače môžu meniť vlastnosti svojich kanálov, ako rozsah, adresy, módy,...). K zápisu týchto parametrov slúži práve tento súbor.
Napríklad:

```
BEGIN
    REAL_FROM 5
    REAL_TO 5
END
```

Ide opäť o textový súbor, ktorého obsahom je zoznam parametrov. Každý ovládač môže mať iné parametre a preto i definícia súborov s parametrami sa líši.

16.2 Virtuálny ovládač

Virtuálny ovládač VSOURCE.DLL je špeciálny ovládač, ktorý sa neviaže s žiadnym konkrétnym fyzickým zariadením. Signály ovládača sa vytvárajú iba v pamäti počítača – virtuálne.

Virtuálny ovládač slúži na generovanie priebehu rôznych signálov a poskytuje rôzne systémové údaje. Tak, ako každý ovládač, má i virtuálny ovládač svoj mapovací súbor.

Štandardný tvar mapovacieho súboru VSOURCE.DMF má podobu:

```
begin
      1 – 96      real input
         97       real output
         98       boolean output
         99       real output
   101 – 200      real output
   201 – 300      boolean output
   301 – 400      integer output
   501 – 600      boolean input
   601 – 700      string output
end.
```

Kanály 1 – 96 sú vyhradené pre generovanie vstupných signálov a preto sa nedoporučuje ich nastavenie meniť.

Virtuálny ovládač poskytuje mimo iného tieto signály a zložky (súbor VSOURCE.PAR):

kanál č. 1 harmonický signál v závislosti na čase
kanál č. 2 trojuholníkový signál v závislosti na čase
kanál č. 3 náhodný signál
kanál č. 4 pílový signál v závislosti na čase
kanál č. 5 obdĺžnikový signál v závislosti na čase
kanál č. 6 sínusový signál v závislosti na počte čítaní
kanál č. 7 trojuholníkový signál v závislosti na počte čítaní
kanál č. 8 náhodný signál
kanál č. 9 pílový signál v závislosti na počte čítaní
kanál č. 10 obdĺžnikový signál v závislosti na počte čítaní

kanál č. 11 počet sekúnd od začiatku dňa s presnosťou na stotiny
kanál č. 12 Juliánske dátum
kanál č. 13 počet dní od 1.1.1900
kanál č. 14 rok
kanál č. 15 mesiac (1..12)
kanál č. 16 deň (1..31)
kanál č. 17 deň v týždni (pondelok – nedeľa -> 1..7)

Rozsah signálov v kanále je tiež súčasťou súboru a jeho hodnoty je možné meniť za behu programu.

```
BEGIN
  SINE_MIN  0
  SINE_MAX  100
  SINE_PERIOD  5.1
  TRIANGLE_MIN  0
  TRIANGLE_MAX  100
  TRIANGLE_PERIOD  1.7
  NOISE_MIN  0
  NOISE_MAX  100
```

```
  SAW_MIN  0
  SAW_MAX  100
  SAW_PERIOD  1
  SQUARE_MIN  0
  SQUARE_MAX  100
  SQUARE_PERIOD  2.3
END.
```

16.3 Modelový ovládač

Modelový ovládač MODEL.DLL slúži ako náhrada za reálnu sústavu pri testovaní správnej funkcie riadiaceho modelu, najmä regulačných sústav.

Mapovací súbor MODEL.DMF má podobu:

```
begin
        1   real output
    2 – 3   boolean output
        5   real input
end.
```

Parametrický súbor modelového ovládača MODEL.PAR nastavuje signály na tieto hodnoty:

```
begin
  param_type          LT_1
  param_k             8.000000E+0
  param_T1            8.000000E-1
  noise               0.000000E+0
  boolean_range_low   0.000000E+0
  boolean_range_high  2.000000E+1
  step_range_low      0.000000E+0
  step_range_high     1.000000E+1
  step_number         7.000000E+0
end.
```

16.4 Simulačný ovládač

Simulačný ovládač DUMMY.DLL nahradzuje pôvodný reálny ovládač pre potreby testovania a ladenia aplikácie. Ovládač nekomunikuje so žiadnym zariadením, ale vracia hodnoty vstupných kanálov podľa toho, ako je definujete v jeho konfiguračnom súboru typu PAR.

16.5 ASCII ovládač

ASCII ovládač ASCDRV.DLL slúži pre komunikáciu cez sériové rozhranie počítača a umožňuje komunikovať s celým radom prístrojov, ktoré používajú štandardný textový formát.

Mapovací súbor ASCDRV.DMF má podobu:

```
begin
        1    real input
        2    real input
        3    string input
        4    string output
        5    boolean output
        6    string input
        7    boolean output
        8    boolean output
        9    boolean output
       10    real input
       11    boolean input
       12    boolean output
end.
```

Parametrický súbor modelového ovládača ASCDRV.PAR nastavuje signály na tieto hodnoty:

```
[comm]
baudrate = 9600
parity = no
databits = 8
stopbits = 1
rx_buffer = 260
tx_buffer = 260
rx_frame_buffer = 260
tx_frame_buffer = 260
cts_flow = true
dsr_flow = false
dtr_control = enable


rts_control = handshake
dsr_sense = high
rx_interchar_timeout = 50
rx_char_timeout = 20
rx_timeout = 50
tx_char_timeout = 20
tx_timeout = 50

[ascdrv]
com_driver = CWCOMM.DLL COM2
terminator = CRLF
ena_multi_str = true
```

Zhrnutie

Ako bolo napísané, systém **Control Web 2000** disponuje ovládačmi na ladenie a simuláciu aplikácii. No nie je to všetko. Do systému je možné zaradiť ďalšie ovládače, ktoré nie sú súčasťou systému, ale dodávajú sa ako samostatné produkty, ovládajúce konkrétne hardvérové vstupne/výstupné dosky.

Medzi špecifické ovládače patrí TCP/IP a DDE ovládač. Slúžia pre dynamické výmeny dát na lokálnej počítačovej sieti, prípadne na komunikáciu cez Internet.

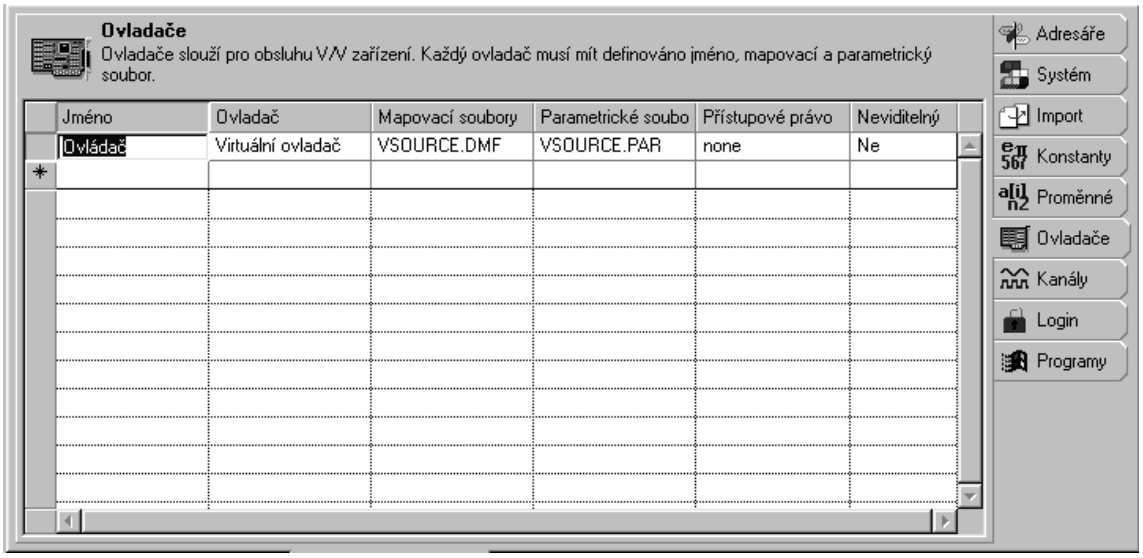
16.6 Úloha č. 22

AKTIVITA

Namodelujte harmonický signál. Jeho priebeh zobrazte.

Riešenie – virtuálny ovládač VSOURCE.DLL

Virtuálny ovládač sa inštaluje v Dátových inšpektoroch, záložke Ovládače.



V položke *Meno* definujte pod akým názvom budete ovládač používať. V príklade je definované meno *Ovládač*.

V položke *Ovládač* roztvorte ponuku a vyberte Virtuálny ovládač. Jeho fyzickou reprezentáciou je súbor VSOURCE.DLL.
V záložke *Mapovacie* súbory vyberte z ponuky položku VSOURCE.DMF. Všimnite si, že ak ukážete na mapovací súbor, zobrazí sa v pravom okne jeho obsah.

Podobne je to aj s výberom parametrického súboru. Vyberte VSOURCE.PAR v záložke *Parametrické* súbory. V pravom okne sa dozviete, aký signál poskytuje konkrétny kanál.

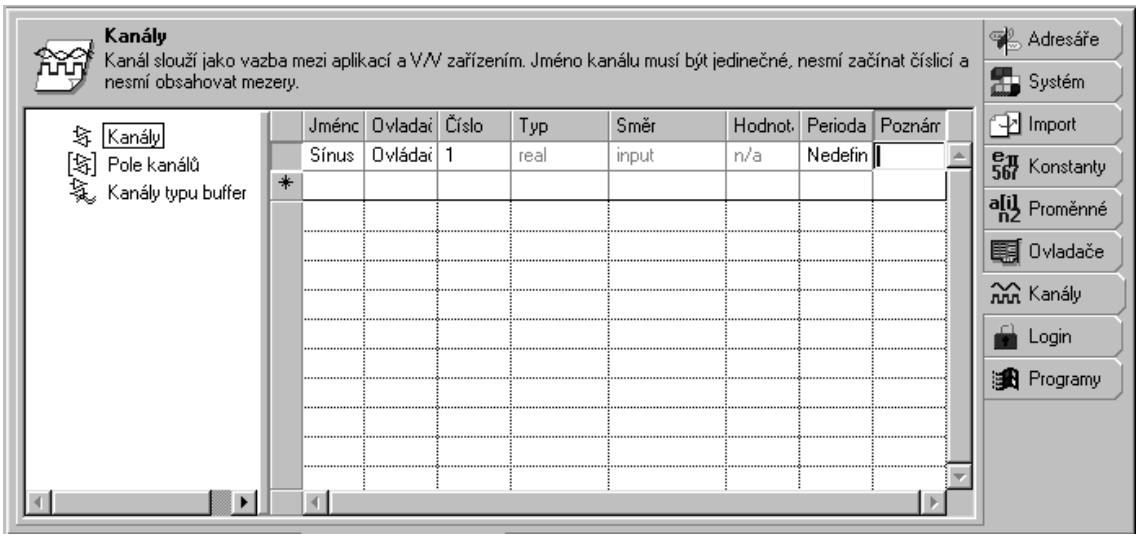
A ešte nebola špecifikovaná cesta k týmto súborom, objaví sa dialógové okno pre nastavenie cesty k súboru (kapitola 3.2.1.4.).

Riešenie – definícia kanálov

Výber ovládača je iba jednou etapou, ako so signálmi pracovať. Je potrebné definovať kanály, ktoré požadované signály poskytnú. Tieto definície poskytuje záložka *Kanály*.

V záložke *Kanály* definujeme meno kanála, na ktoré sa budeme odkazovať v programe. Zvoľte meno *Sinus*.

V položke *Ovládač* definujeme názov zvoleného ovládača. Ovládač sme definovali v minulej kapitole, zvolíme si ten istý.



Číslo reprezentuje kanál, ktorý nám poskytne požadovanú službu. Podľa obsahu parametrického súboru pre náš účel vyhovuje kanál č. 1.

Ostatné položky nevyplňujeme.

Pokiaľ sa pozriete do textového editora, mali by ste v zdrojovom texte nájsť tieto fragmenty:

```
directories
  '*.DMF' = 'C:\PROGRAM FILES\CONTROL WEB 2000\DMF';
  '*.PAR' = 'C:\PROGRAM FILES\CONTROL WEB 2000\PAR';
end_directories;

driver
  Ovládač = 'vsource.dll', 'VSOURCE.DMF', 'VSOURCE.PAR';
end_driver;

channel
  Sínus = real, 1, Ovládač, input;
end_channel;
```

Riešenie – prístroj *chart*

Prístroj *chart* zobrazuje časové priebehy signálov v grafe. Je teda veľmi vhodný na zobrazovanie priebehov kolísania teploty, tlaku, hladiny a podobne.

Medzi najdôležitejšie parametre prístroja *chart* patrí:

timer – známý parameter pre časovanie prístroja.
mode – určuje grafickú podobu prístroja
- **flow_graph** – posúvajúci sa graf. Vykresľuje poslednú hodnotu vždy na koniec grafu, pričom celý graf sa priebežne posúva.
- **sweep_graph** – prekreslujúci sa graf. Vykresľuje graf postupne zľava doprava. Ak dosiahnete koniec, prepisovanie začne opäť zľava doprava. Graf sa neposúva.
- **line_flow_graph** – graf podobný módu *flow_graph*. V tomto móde ale nie je využívaný grafický posun časti grafu, ktorý je u moderných grafických kartách veľmi rýchly. Graf je po častiach prekresľovaný, čo prináša možnosť stojacej mriežky. Nevýhodou tohto módu je nutnosť vykresľovania vysokého počtu čiar v každom časovom kroku a tým značná spotreba času na prekresľovanie.

content – obsažnosť vzhľadu prístroja. Ponúka tri možnosti:
- **Min** – minimálne zobrazenie prístroja
- **Med** – k základnému vzhľadu prístroja sa pridá číslicový zobrazovač a numerické riadky pre zmenu horného a dolného limitu.

ranfe_from, range_to – rozsah stupnice prístroja a tým aj rozsah výstupných hodnôt.
low_limit, high_limit – dolné a horné limity, ktoré sa vykresľujú v grafe podľa ktorých sa v číslicovém zobrazovači mení farba pozadia vykresľovaného čísla.
history – udáva počet spätne pamätaných a zobrazovaných hodnôt v grafe. Je dĺžkou historického trendu zobrazovanej veličiny.
dec_places – počet zobrazovaných desatinných miest na číselnom zobrazovači
real_step – je krokom nastavovania limitných hodnôt pomocou numerického riadku. Jeho veľkosť udáva prírastok alebo úbytok hodnoty pri použití inkrementálnych a dekrementálnych šipok.
h_grid – počet vodorovných čiar rastra. Ak nie je parametr zadaný, alebo je 0, vodorovný raster nebude vykresľovaný.
v_grid - počet zvislých čiar rastra. Ak nie je parameter zadaný, alebo je 0, vodorovný raster nebude vykresľovaný.
clear – označuje logický výraz, pri ktorom splneniu bude vyrovnávacia pamäť prístroja zmazaná a všetky hodnoty nastavené na inicializačné hodnoty.
activity – označuje logický výraz s podmienkou činnosti prístroja. Pri nesplnenej podmienke sú grafy zastavené. Ak nie je výraz zadaný, je prístroj stále aktívny.
item – zložka *item* je známa z mnohých prístrojov. Parameter *expression* obsahuje numerický výraz, ktorý bude prístrojom vyhodnocovaný.

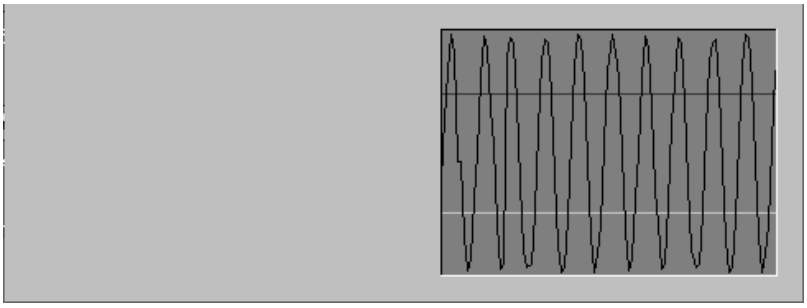
Ak je prístroj *chart* v okne, je možné pomocou menu meniť mód prístroja počas behu programu.

V našej úlohe nastavíme parametre:

```
chart Zapisovač;
  timer = 0.1;
  mode = flow_graph;
```

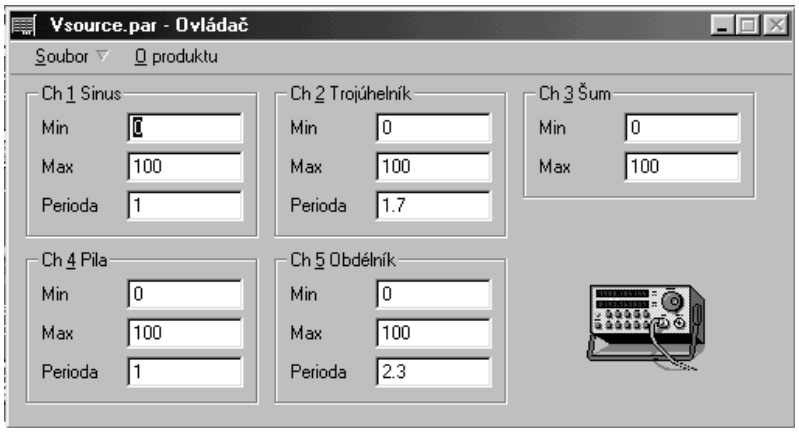
```
  history = 100;


  item
    expression = Sínus;
    color = black;
  end_item;
end_chart;
```

Ak úlohu spustíme, bude prístroj *Zapisovač* zobrazovať sínusový signál.



Riešenie – zmena parametrov signálu počas behu programu

V bežiacom programe, ktorý používa virtuálny ovládač, je možné operatívne meniť vlastnosti vstupných signálov. Všimnite si, že ak program beží, na spodnej nástrojovej lište sa objaví ikona s názvom Vsource.par – Ovládač. Poklepaním na ikonu sa ikona maximalizuje na dialógové okno.



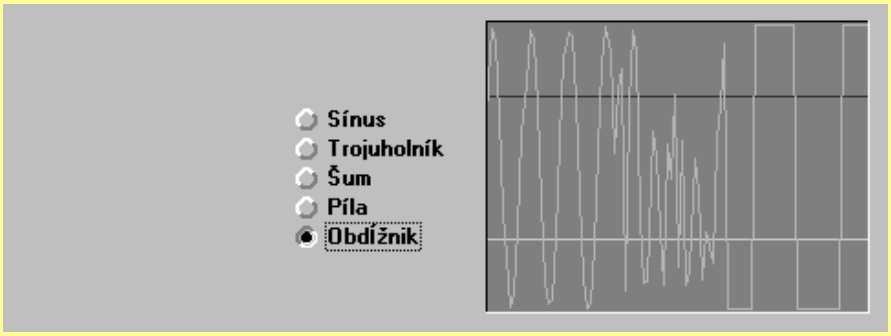
Dialógové okno je rozdelené na päť častí podľa druhov signálov, ktoré virtuálny ovládač ponúka.

U každého signálu je možné v okne nastaviť minimálnu hodnotu, maximálnu hodnotu a periódu zmeny signálu. Dovoľuje tak experimentovať s programom, analyzovať ho a dokazovať jeho správnosť.

16.7 Úloha č. 23

AKTIVITA

Namodelujte zobrazovanie rôznych typov signálov. Zobrazovanie signálu ovládajte prepínačom.



Riešenie – definícia vstupných signálov a premenných

Cez *Dátoví inšpektori* definujte premenné:
- Signál – zobrazovaná hodnota
- Prepínač – pole piatich premenných pre prepínač
- Kanály 1 až 5 virtuálneho ovládača

```
var
  Signál = real, 0;
  Prepínač = array[ 1..5 ] of boolean, false;
end_var;
driver
  Ovládač = 'vsource.dll', 'VSOURCE.DMF', 'VSOURCE.PAR';
end_driver;

channel
  Obdĺžnik = real, 5, Ovládač, input;
  Píla = real, 4, Ovládač, input;
  Sínus = real, 1, Ovládač, input;
  Trojuholník = real, 2, Ovládač, input;
  Šum = real, 3, Ovládač, input;
end_channel;
```

Riešenie – prepínač kanálov

Prepínač kanálov v módu *radio_button* je realizovaný prístrojom *multi*_switch.

```
multi_switch Prepínač;
   mode = radio_button;
   item
```

```
      text = 'Sínus';
      output = Prepínač[ 1 ];
    end_item;
.....................
    item
      text = 'Obdĺžnik';
      output = Prepínač[ 5 ];
    end_item;
  end_multi_switch;
```

Riešenie – zobrazenie signálu

Pre zobrazenie vybratého signálu použijeme prístroj *multiplexer*.

V zložkách *input_item* definujeme podmienky pre vybratie príslušného signálu. Podmienka je určená polohou prepínača, teda príslušnou zložkou premennej *Prepínač* typu pole.

Pre zobrazenie grafickej podoby prístroja použijeme mód *flow_graph* – posúvajúci sa graf. Parameter *history* (počet pamätaných hodnôt v grafu) nastavte na hodnotu 100.

To, čo je v tejto úlohe zaujímavé a podstatné, je naplnenie parametra *Output*.

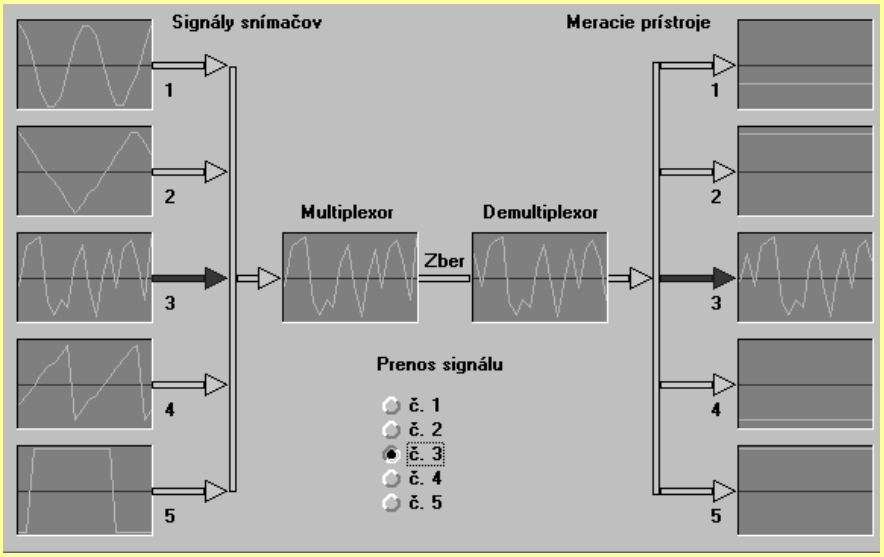
V parametri *Output* sa definuje výstupná premenná alebo výstupný kanál, kde sa bude zobrazovaná hodnota posielať. Ak by sme tento parameter nezadali, nebude sa signál v grafe zobrazovať!

```
multiplexer Zapisovač;
    timer = 0.1;
    input_item
      condition = Prepínač[ 1 ];
      expression = Sínus;
    end_input_item;
.........................
    input_item
      condition = Prepínač[ 5 ];
      expression = Obdĺžnik;
    end_input_item;
    output = Signál;
    mode = flow_graph;
    history = 100;
  end_multiplexer;
```

16.8 Úloha č. 24

AKTIVITA

Namodelujte činnosť zberu informácií zo snímačov. Prenos realizujte pomocou jednej signálnej linky. Prenášaný signál voľte prepínačom a v animácii demonštrujte.



Riešenie – definícia signálov, premenných a konštánt

V Dátových inšpektoroch definujeme tieto objekty:
- signály snímačov – vonkajšie vplyvy na snímače sú definované ako kanály 1 až 5 Virtuálneho ovládača.
- Výstup – signály zo snímačov. V deklarácii je použitá premenná pole typu real.
- Zber – spoločná signálová linka
- Vstup – vstup signálov do meracích prístrojov. V deklarácii je použitá premenná typu real.
- Prepínač – premenná pre voľbu prenášaného signálu
- Čas – konštanta časového kroku prístrojov
- History – počet spätne pamätaných a zobrazovaných hodnôt v grafe
- Limit – konštanta pre nastavenie horného a spodného limitu grafu.

```
const
  Čas = 0.1;
  History = 20;
  Limit = 50;
end_const;

var
  Zber = real, 0;
  Prepínač = array[ 1..5 ] of boolean, false;
  Výstup = array[ 1..5 ] of real, 0;
  Vstup = array[ 1..5 ] of real, 0;
```

```
end_var;

driver
  Ovládač = 'vsource.dll', 'VSOURCE.DMF', 'VSOURCE.PAR';
end_driver;

channel
  Obdĺžnik = real, 5, Ovládač, input;
  Píla = real, 4, Ovládač, input;
  Sínus = real, 1, Ovládač, input;
  Trojuholník = real, 2, Ovládač, input;
  Šum = real, 3, Ovládač, input;
end_channel;
```

Riešenie – časovanie prístrojov

Prístroje sú časované časovačom typu *sequencer* s názvom *Časovač*. Čas je stanovený konštantou.

```
timer

  sequencer Časovač;
    timer = Čas;
  end_sequencer;

end_timer;
```

Riešenie – grafy vstupných hodnôt

Grafy vstupných hodnôt sú použité iba z dôvodu demonštrácie vstupného signálu a pre generovanie výstupného signálu zo snímača. Výstupné signály vystupujú z meracích prístrojov s premennej Výstup (Výstupný dátový element).

Časovanie, limity a veľkosť histórie zobrazovaných hodnôt sú nastavené pomocou konštánt.

```
  meter Sínus;
    timer = Časovač, 1;
    owner = Skúšobný_panel;
    position = 10, 10, 90, 60;
    Výstup[ 1 ] = Sínus;
    mode = flow_graph;
    low_limit = Limit;
    high_limit = Limit;
    history = History;
  end_meter;
```

Riešenie – príprava signálov na prenos po signálovej ceste

Signály so snímačov sa sústreďujú v prístroji multiplexor, kde po splnení podmienky v zložke *item* vystupujú pomocou premennej *Zber* von z prístroja na signálovú cestu.

```
multiplexer Zapisovač_vstup;
  timer = Časovač, 11;
  input_item
    condition = Prepínač[ 1 ];
    expression = Sínus;
  end_input_item;
.................
  input_item
    condition = Prepínač[ 5 ];
    expression = Obdĺžnik;
  end_input_item;


  output = Zber;
  mode = flow_graph;
  low_limit = Limit;
  high_limit = Limit;
  history = History;
end_multiplexer;
```

Riešenie – prístroj *demultiplexor*

Príjem signálu zo signálovej cesty a jeho smerovanie na príslušný merací prístroj sa realizuje pomocou prístroja *demultiplexor*.

Prístroj *demultiplexor* zobrazuje *ľubovoľný* numerický výraz pomocou rúčkového prístroja, grafu alebo displeja. Hodnota výrazu je nielen zobrazená, ale taktiež vyslaná na výstupné premenné, ktoré majú splnenú podmienku výberu.

Medzi špecifické parametre prístroja patrí:
timer – známy parameter pre časovanie prístroja
output_item - v položke *condition* sa definuje podmienka výberu výstupnej premennej, v položke *output* sa definuje výstupná premenná. Počet zložiek *item* nie je obmedzený.
input – vstupný numerický výraz, ktorý sa zobrazí na prístroji a prípadne sa vyšle na výstup, ktorý má splnenú podmienku výstupu.
mode – grafická podoba prístroja zhodná s prístrojom typu *meter*
content – obsažnosť vzhľadu prístroja
range_from, range_to – rozsah stupnice prístroja a tým aj rozsah výstupných hodnôt
low_limit, high_limit – dolné a horné limity ako u prístroja *meter*
history – dĺžka historického trendu ako u prístroja *meter*
dec_places – počet zobrazovaných desatinných miest
h_grid, v_grid – počet vodorovných a zvislých čiar rastra
mask – šablona pre vypisovanie ľubovoľného textu súčasne s numerickou hodnotou prístroja. Parameter má význam pri móde *text_display*. Text sa zadáva v Inšpektorovi prístroja

bez úvodzoviek. Môže mať napríklad tvar ##.## m/s, kde znak mriežka udáva umiestnenie a veľkosť zobrazovanej numerickej hodnoty.

Kritické fragmenty programu:

```
demultiplexer Demultiplexer_1;
   timer = Časovač, 12;
   output_item
     condition = Prepínač[ 1 ];
     output = Vstup[ 1 ];
   end_output_item;
.....................
   output_item
     condition = Prepínač[ 5 ];
     output = Vstup[ 5 ];
   end_output_item;
   input = Zber;
   mode = flow_graph;
   low_limit = Limit;
   high_limit = Limit;
   history = History;
 end_demultiplexer;
```

Riešenie – zobrazenie preneseného signálu

Zobrazenie preneseného signálu prevádza prístroj *meter*. Bol dostatočne popísaný v predchádzajúcich kapitolách.

Kritické časti prístroja:

```
 meter Merač_1;
   timer = Časovač, 10;
   expression = Vstup[ 1 ];
   mode = flow_graph;
   low_limit = Limit;
   high_limit = Limit;
   history = History;
 end_meter;
```

Riešenie – prepínanie signálov

Prepínanie signálov zaisťuje prístroj *multi_switch*. Bol dostatočne popísaný v predchádzajúcich kapitolách.

Kritické fragmenty programu:

```
multi_switch Prepínač;
   mode = radio_button;
```

```
    item
      text = 'č. 1';
      output = Prepínač[ 1 ];
    end_item;
.....
    item
      text = 'č. 5';
      output = Prepínač[ 5 ];
    end_item;
  end_multi_switch;
```

Riešenie – prístroj *box*

V úlohe je použitý prístroj *box* pre grafické vyznačenie spojovacích ciest a ich animáciu. Jedná sa teda o grafický symbol obdĺžnik.

Medzi špecifické parametre prístroja patrí:
mode – určuje grafickú podobu prístroja
- **interior_and_border** – vyplnený obdĺžnika s okrajom
- **interior_only** – vyplnený obdĺžnik bez okraja
- **border_only** – priehľadný obdĺžnik s okrajom

blink – logický výraz. Pri jeho splnení bude prístroj blikať.

Kritické fragmenty prístroja:

```
box Čiara_1;
  blink = Prepínač[ 1 ];
end_box;
```

Riešenie – prístroj *triangle*

V úlohe je použitý prístroj *triangle* pre grafické vyznačenie smeru spojovacích ciest. Prístroj má podobu trojuholníka.

Medzi špecifické parametre prístroja patrí:
mode – určuje grafickú podobu prístroja
- *interior_and_border* – vyplnený trojuholník s okrajom
- *interior_only* – vyplnený trojuholník bez okraja
- *border_only* – priehľadný trojuholník s okrajom

blink – logický výraz. Pri jeho splnení bude prístroj blikať.
site_mode – určuje natočenie trojuholníka. Môžeme použiť symboly *left, right, up, down.*

Kritické fragmenty prístroja:

```
triangle Šipka_1;
  blink = Prepínač[ 1 ];
```

```
    side_mode = left;
end_triangle;
```

Záver

Cieľom úlohy bolo demonštrovať použitie prístrojov multiplexor a demultiplexor pre zníženie počtu prenosových ciest medzi zdrojom signálu (snímačom) a vyhodnocujúcim prvkom (napr. meracím prístrojom).
Tento mechanizmus môžeme použiť v prípade, že v jednom časovom okamihu nie je potrebné snímať všetky signály, napríklad pre informatívne dozeranie na teplotu kotla.

16.9 Úloha č. 25

AKTIVITA

Vytvorte panel zobrazujúci aktuálny čas.

Riešenie – systémové premenné

Control Web 5 disponuje celým radom systémových premenných. Systémové premenné sú špeciálne premenné, ktoré nie je možné meniť, iba čítať.

Jméno	Typ	Hodn	Poznámka
hour	real	0	aktuální hodina ve dni
minute	real	0	aktuální minuta ve dni
second	real	0	aktuální sekunda ve dni
hsec	real	0	aktuální násobek deseti milisekund ve dni
msec	real	0	aktuální milisekunda ve dni
year	real	0	aktuální rok
month	real	0	aktuální měsíc v roku
day	real	0	aktuální den v roku
day_in_week	real	0	aktuální den v týdnu
julian_date	real	0	aktuální juliánské datum
sec_in_day	real	0	aktuální sekunda ve dni
run_time_hsec	real	0	aktuální násobek deseti milisekund od okamžiku spuštění aplikace
run_time_msec	real	0	aktuální milisekunda od okamžiku spuštění aplikace
scan_code	real	0	kód stisknuté klávesy
user_level	real	0	úroveň přístupových práv přihlášeného uživatele
user_name	string	''	jméno přihlášeného uživatele
sound_freq	real	440	proměnná pro kompatibilitu se systémem Control Panel - ve Windows nepoužívat
sound_len	real	0.1	proměnná pro kompatibilitu se systémem Control Panel - ve Windows nepoužívat
sound_start	boolean	false	proměnná pro kompatibilitu se systémem Control Panel - ve Windows nepoužívat
wave_player_busy	boolean	false	příznak aktivity přehrávače WAV souborů, má hodnotu true, je-li právě zvuk přehráván
mouse_x	real	0	aktuální vodorovná pozice kurzoru myši
mouse_y	real	0	aktuální svislá pozice kurzoru myši

V našej úlohe využijeme systémové premenné:
- *hour* – systém vracia aktuálnu hodinu dňa
- *minute* – systém vracia aktuálnu minútu v hodine
- *second* – systém vracia aktuálnu sekundu v minúte

Údaje o čase sa berú zo systémového času počítača. Preto pre aktuálne časové informácie je nutné nastaviť správny čas vo Windows.

K zobrazeniu času použijeme prístroj *meter*. Vzorkovanie a nastavenie základných parametrov prístrojov *meter* definujeme v konštantách. Časovanie prístrojov prevádza časovač *sequencer*.

Kritické časti programu:

```
const
  Limit = 60;
  History = 1;
  DecPlaces = 0;
  Vzorka = 0.5;
end_const;

timer
  sequencer Vzorkovanie;
    timer = Vzorka;
  end_sequencer;
end_timer;

instrument

  window panel Skúšobný_panel;
    owner = background;
    win_title = 'Hodiny';
  end_panel;

  meter Sekundy;
    timer = Vzorkovanie, 1;
    owner = Skúšobný_panel;
    expression = second;
    mode = digital;
    range_to = 59;
    low_limit = Limit;
    high_limit = Limit;
    history = History;
    dec_places = 0;
    frame = 2;
  end_meter;

  meter Minuty;
    timer = Vzorkovanie, 2;
    owner = Skúšobný_panel;
```

```
    expression = minute;
    mode = digital;
    range_to = 59;
    low_limit = Limit;
    high_limit = Limit;
    history = History;
    dec_places = DecPlaces;
  end_meter;

  meter Hodiny;
    timer = Vzorkovanie, 3;
    owner = Skúšobný_panel;
    expression = hour;
    mode = digital;
    range_to = 23;
    low_limit = Limit;
    high_limit = Limit;
    history = History;
    dec_places = DecPlaces;
  end_meter;
end_instrument;
```

17 PROGRAMOVANIE A PROCEDÚRY OCL

Vytváranie projektu pomocou vizuálnych techník je silným nástrojom rýchleho programovania. Programátor sa nestará o syntax príkazového riadku, je odbremenený od detailov textového zápisu.

Skúsenosť nám však vraví, že každá strana má dve mince. Rýchle programovanie prináša určité obmedzenie a v určitých detailoch i nemožnosť ošetriť všetky zamýšľané požiadavky.

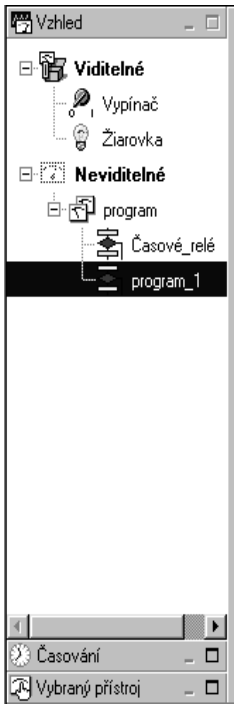
Control Web 5 ponúka i druhú stranu mince – programovanie a procedúry OCL.

Programovanie pod **Control Web 5** sa líši od zabehnutého, klasického programovania. Pri klasickom programovaní (Pascal, Modula 2, ...) existuje určitá časť programu, ktorá je vyhlásená ako "hlavná" a ktorá volá ostatné časti programu - funkcie a procedúry. **Control Web 5**, ako objektovo orientovaný systém reálneho času, by s touto technikou nevystačil. Logické celky programov sú v systéme zapracované do prístroja *program*, ktoré sa správa ako samostatný komponent riadený udalosťou alebo časovaním. Nie je tu nič, čo by sme mohli prehlásiť za tzv. "hlavný program". Systém však ponúka aj niečo naviac - OCL metódy.

OCL metódy (Object Control Language) je jazyk riadenia objektov. Dokážu ovládať prístroje priamo, pomocou definovania ich vlastností. Dostávame tak silný nástroj na zmenu prístrojov "za behu aplikácie". Každý prístroj má definované tzv. natívne procedúry, zasahujúce korektným spôsobom do komponenty a doplňujúce aplikáciu o potrebnú funkčnosť. Pri programovaní máme vždy projekt "pod kontrolou", to znamená, že nemôžeme zasahovať nekorektným spôsobom do objektov a spôsobiť tak haváriu celého projektu. V klasickom programovaní je taký zásah úplne bežný.

17.1 Prístroj program

Prístroj *program* sa umiestňuje v strome vzhľadu aplikácie medzi neviditeľné prístroje.



Inšpektor prístroja ponúka v záložke *parametre* veľmi málo možností. Zo známych sú to názov programu a *timer* pre časovanie prístroja, pokiaľ nebude riadený udalosťou. Doména jeho deklarácie sa skrýva v inej záložke – *Procedúry*.

Záložka *Procedúry*, ak použijeme časť *OnActivate*, obsahuje tieto údaje:

```
procedure OnActivate();
  begin
  end_procedure;
```

Procedúra s názvom *OnActivate ()* je výnimočná udalostná procedúra.

Prvý dôvod jej výnimočnosti spočíva v tom, že je to jediná procedúra, volaná vždy pri aktivácii prístroja, alebo presne povedané, **pred každou aktiváciou prístroja**. Procedúra má teda možnosť testovať a modifikovať dátové elementy ovplyvňujúce činnosť prístroja.

Druhým dôvodom výnimočnosti je skutočnosť, že môžeme poznať dôvody aktivácie procedúry. Ak zapíšeme tvar

Procedure OnActivate (ByTimer, ByInstrument, ByDriver, ByData : boolean);

budú procedúre predané štyri logické parametre, určujúce dôvody aktivácie. Vo vlastnom kóde môžeme príčinu testovať a prevádzať nejakú činnosť.

V určitých situáciách môže dôjsť k súbehu niekoľkých dôvodov. V tom prípade je prístroj aktivovaný iba raz a príslušné parametre sú nastavené do hodnoty *true*.

Systém **Control Web 5** dokáže pri aktivácii prístroja omnoho viac – odovzdaním bitovej masky. Bitová maska predstavuje jedno číslo, ktoré nesie logické hodnoty podobne ako logické parametre. Pre zistenie, či konkrétny bit daného čísla bol nastavený, možno použiť funkciu *bitget*. Napríklad:

```
procedure OnActivate (ActiveMode : cardinal);
procedure
  if bitget (ActivateMode,2) = 1
     then ..... (* aktivácia od ovládača *)
  end;
end_procedure;
```

Význam jednotlivých bitov je nasledujúca:

bit	význam
0	prístroj je aktivovaný časovačom
1	prístroj je aktivovaný iným prístrojom
2	prístroj je aktivovaný ovládačom
3	rezervované, vždy 0
4	prístroj je aktivovaný zmenou dát pri dátovo riadených aplikáciách
5	prístroj je v časovom skĺze
6	prístroj je aktivovaný ako prvý prístroj aplikácie
7	prístroj je aktivovaný ako posledný prístroj aplikácie
8	aplikácia je v časovom skĺze (a nemusí to byť práve tento prístroj)
9	aplikácia končí
10 až 31	je rezervované pre ďalšie využitie.

Okrem procedúry OnActivate disponuje prístroj program časťou *Nová procedúra*. V tejto časti môžeme definovať ďalšie procedúry. Skôr ako sa vrhneme do programovania, ukážeme si, aká je obecná štruktúra programu.

Štruktúra prístroja *program*

Hlavička komponenty *program* začína kľúčovým slovom **program**, za ktorým nasleduje názov prístroja. Program končí kľúčovým slovom **end_program***;*

Telo procedúry alebo procedúr začína kľúčovým slovom **procedure**, nasleduje jeho meno, zátvorky pre prípadné parametre (tie sú podstatné, aj keď procedúra nemá žiadne parametre), kľúčové slovo **begin** a procedúra končí kľúčovým slovom **end_procedure**. Jednotlivé bloky príkazov sú oddelené bodkočiarkou. Medzi kľúčové slová **begin** a **end_procedure** budú umiestnené výkonné príkazy procedúry.

Minimálna konfigurácia vyzerá takto:

```
program Pokus;

  procedure Nič();
  begin
  end_procedure;

end_program;
```

Taký program skutočne neurobí nič. Dokonca sa sám nikdy nespustí, pokiaľ ho nebude volať iný prístroj.

Časovanie procedúry

Časovanie prístroja deklarujeme štandardne v premennej *timer*. V takom prípade bude periodicky aktivovaná. V udalostných procedúrach sa procedúra aktivuje, ak ju zavoláme z inej procedúry alebo prístroja.

```
program Časové_relé;
  timer = 0.8;
```

Algoritmus v procedúre prebehne v jednom časovom kroku, ak to nie je stanovené inak. Výnimku môžu spôsobiť príkazy **pause**, **yield** a **wait**.

Návestie

Pokiaľ sú v procedúre použité skokové inštrukcie **goto**, je nutné ešte pred kľúčové slovo **begin** uviesť kľúčové slovo **label** a zoznam návestí oddelených čiarkami. V programe sa cieľ skoku označí uvedením návestí nasledované dvojbodkou, napríklad takto:

```
  procedure Nič();
  label
    Skok1, Skok2;

  begin
    (* príkazy *)
    Skok1:
    (* príkazy *)
    Skok2:
   (* príkazy*)
  end_procedure;
```

O škodlivosti používania príkazov skoku bolo popísané mnoho. Obecne povedané, príkaz skoku v programe nevadí, dokonca môže v niektorých prípadoch skrátiť algoritmus. To, čo je na tomto príkaze škodlivé je, že sa stráca prehľad v algoritme, v jeho modularite. Každý programátor musí zvážiť užitočnosť každého príkazu, teda aj príkazu **goto**. Je programátorskou cťou napísať akúkoľvek procedúru bez použitia uvedeného príkazu. Je to hodená rukavica do programátorského ringu.

Deklarácia lokálnych konštánt a premenných

Okrem globálnych premenných a konštánt, deklarovaných v Dátových inšpektoroch, umožňuje prístroj *program* deklaráciu aj lokálnych dátových elementov. Rozdiel medzi globálnym a lokálnym elementom je v tom, že globálny element je viditeľný v celom projekte a každý prístroj ho môže využívať, napriek tomu lokálny element je viditeľný iba v prístroji alebo procedúre, v ktorej je deklarovaný. **Ak však použijeme rovnaký názov globálneho a lokálneho elementu, lokálny element má prednosť a zatieni globálny!**

Lokálne elementy v prístroji program definujeme cez Inšpektor prístroja v položke *Lokálne dáta*. Lokálne elementy procedúry definujeme priamo v procedúre.

Deklaráciu konštánt predchádza kľúčové slovo **const**, deklaráciu premenných slovo **var.** Hodnoty premenné, uvedené v deklarácii **var**, budú zabudnuté pri ukončení činnosti procedúry (alebo prístroja). Pokiaľ chcete, aby sa hodnoty lokálnych premenných uchovali aj medzi jednotlivými volaniami procedúry (prístroja), musíte je deklarovať za kľúčovým slovom **static**, nie **var.**

Ak potrebujeme, aby procedúra pracovala s rôznymi parametrami, importujeme tieto parametre pri deklarácii názvu procedúry.

Príklad:

```
program Pokus;
 const
  DĺžkA = 10; (*konštanta viditeľná v celom prístroji Pokus*)
 end_const;

var
 Vysledok = real, 0; (*premenná viditeľná v celom prístroji Pokus*)
end_var;

static
 Otáčky = real, 0;(*Hodnota sa bude pamätať aj po ukončení *)
end_static;      (*činnosti prístroja.*)

timer = 0.8;

procedure Proc_1(Param_1, Param_2 : real);(* predaní parametrov *)
label
 Skok;

const
```

```
Zero = 0;  (* konštanta viditeľná v procedúre Proc_1*)

var
Číslo = boolean, false; (*premenná viditeľná v procedúre Proc_1*)

static
  Pamäť = real, 0; (*premenná viditeľná v procedúre Proc_1*)
```

Pri programovaní obecne nie je možné deklarovať dve procedúry s rovnakým názvom. Pri ich volaní inou procedúrou by systém nevedel, ktorú z nich má zavolať. **Control Web 5** však pozná ešte niečo naviac. **Pokiaľ definujeme dve procedúry rovnakého názvu, ale s rozdielnym počtom parametrov, je to v poriadku.** Táto odlišnosť je postačujúca k rozlíšení o ktorú procedúru ide!

Príkaz **if**

Príkaz **if** slúži k podmienenému vykonávaniu určitých častí programu. Za kľúčovým slovom musí byť uvedený logický výraz a kľúčové slovo **then**. Nasledujúci blok príkazov je vykonaný len v tom prípade, pokiaľ logický výraz bude vyhodnotený ako *true*.

Blok príkazov môže byt ukončený troma spôsobmi:

- kľúčovým slovom **end**
- kľúčovým slovom **else**. Za **else** nasleduje blok príkazov, ktorý bude vykonaný len v prípade nesplnení logickej podmienky. Blok musí byť ukončený kľúčovým slovom **end**.
- kľúčovým slovom **elsif**. Za týmto slovom nasleduje podmienka a kľúčové slovo **then** (a všetko sa opakuje, ako bolo popísané).

Príklad:

```
if Číslo = 0 then
  Pamäť = 0
elsif Číslo = 1 then
  Pamäť = 1
elsif Číslo = 2 then
  Pamäť = 2
else  Pamäť = 9
end;
```

Príkaz **loop**

Príkaz **loop** patrí do skupiny príkazov cyklu. Presne povedané, je to nekonečný cyklus. Pokiaľ nemá byť cyklus nekonečný, musí vo vnútri cyklu existovať algoritmus s kľúčovým slovom **exit**.

Pokiaľ je v sebe zanorených viacej cyklov **loop**, príkaz **exit** sa vzťahuje vždy na najvnútornejší cyklus. Vo vnútri cyklu je možné použiť kľúčové slovo **continue**, ktoré prenesie riadenie na počiatok cyklu (preskočí zvyšok príkazov v cyklu).

Príklad_1:

```
  loop
```

```
    Číslo = Číslo + 1;  (* nekonečný cyklus. Pozor na tento cyklus, NIKDY
  NESKONČÍ! *)
  end;

  Príklad_2:
loop
  Číslo = Číslo + 1;
  if Číslo = 255 then
    exit;  (* cyklus končí po dosiahnutí 255 *)
  end;

  Príklad_3:
    loop
      Číslo = Číslo + 10;
      Číslo = Číslo - Pamäť;

      if Číslo < Zero then
        continue; (* zvyšok príkazov do konca slučky preskoč
      end;        (* a vráť sa na prvý príkaz v loop *)

      if Číslo = 255 then
        exit;
      end;
    end;
```

Príkaz **while**

Príkaz **while** patrí do skupiny príkazov cyklu. Je to cyklus s podmienkou pri vstupu do slučky. Za kľúčovým slovom **while** nasleduje logický výraz a kľúčové slovo **do**. Pokiaľ je logický výraz vyhodnotený ako *true*, vykoná sa nasledujúci blok príkazov až po kľúčové slovo **end**. Pokiaľ sa v bloku príkazov nevyskytujú príkazy **pause** alebo **wait**, neprebieha v ďalších priechodoch slučkou komunikácia s vonkajším zariadením.

Pokiaľ nie je logický výraz pravdivý pred vstupom do cyklu, cyklus sa neprevedie.

Príklad:

```
  Číslo = 1;

  while Číslo < Pamäť do
     Číslo = Číslo + 1;
  end;
```

Príkaz **repeat until**

Príkaz **repeat until** patrí do skupiny príkazov cyklu. Je to cyklus s podmienkou na konci slučky. Za kľúčovým slovom **repeat** nasleduje blok príkazov, ukončený kľúčovým slovom **until** s logickým výrazom. Ak je logický výraz vyhodnotený ako pravdivý, cyklus končí.

Je teda zrejmé, že blok príkazov sa prevedie vždycky minimálne jedenkrát.

Príklad:
```
Číslo = 1;

repeat
  Číslo = Číslo + 1;
until Číslo >= Pamäť
```

Príkaz **for**

Príkaz **for** patrí do skupiny príkazov cyklu. Je to cyklus s dopredu známym počtom cyklov. Za kľúčovým slovom **for** nasleduje zápis číselnej premennej, znak = (rovná sa) a číselný výraz. Potom nasleduje kľúčové slovo **to** a ďalší číselný výraz nasledovaný kľúčovým slovom **do** a blok príkazov ukončený kľúčovým slovom **end.**

Pri vstupe do cyklu sa nastaví do číselnej premennej hodnota prvého čísla a potom sa vykoná blok príkazov. Nakoniec sa zvýši číselná premenná o jednotku. Tento cyklus sa bude opakovať dovtedy, kým číselná premenná nebude väčšia ako druhé číslo.

Príklad:
```
for Číslo = 1 to 10 do
   Číslo = Číslo + 1;
end;
```

Nie vždy je požadované, aby sa číselná premenná zvyšovala v každom cykle o jednotku. Niekedy je nutné krok zväčšiť. Preto je možné cyklus **for** doplniť o krok. Krok sa zapisuje kľúčovým slovom **by** pred kľúčovým slovom **do.**

Príklad:
```
for Číslo = 1 to 10 by 2 do
   Číslo = Číslo + 1;
end;
```

Ak požadujeme, aby sa premenná zmenšovala, dosadíme do prvej číselnej premennej väčšiu hodnotu ako do druhej a za kľúčové slovo **by** dosadíme zápornú číselnú hodnotu.

Príklad:
```
for Číslo = 11 to 1 by -2 do
   Číslo = Číslo + 1;
end;
```

Príkaz **yield**

Ak je spustená procedúra, jej telo sa vykoná v jednom časovom kroku a teda s rovnakou hodnotou na vstupne/výstupných kanáloch. Niekedy je však potrebné časový krok ukončiť a umožniť systému znovu zmerať hodnoty na kanáloch.

Príkaz **yield** ukončí vykonávanie procedúry v aktuálnom časovom kroku a umožní systému obsluhu všetkých prístrojov, ktoré majú byť v prebiehajúcim časovom kroku aktivované. Vykonávanie procedúry bude pokračovať v najbližšom systémovom časovom kroku.

Príklad:

```
while Číslo < 20.0 do (*čakanie na dosiahnutí stanovenej hladiny*)
  yield;    (* ukončenie časového kroku. Umožní sa komunikácia pri meraní nových
hodnôt *)
end;
```

Príkaz yield je možné použiť výhradne v procedúre OnActivate. V žiadnych iných udalostných alebo užívateľských procedúrach táto inštrukcia nie je povolená! Ak však chceme obísť tento zákaz, môžeme nastaviť parameter *independent_procedure_execution* na false buď na začiatku projektu v kapitole settings, alebo v integrovanom vývojovom prostredí v Dátových inšpektoroch - Systém.

Príkaz **pause**

Príkaz **pause** pozastavuje beh programu na určitú dobu. Za kľúčovým slovom nasleduje numerický výraz udávajúci počet sekúnd zadržanie programu.

Príkaz **pause** 0 nepozastaví program, ale poskytne systému čas k prípadnej obsluhe ďalších prístrojov, rovnako ako **yield.**

Príklad:

```
while Číslo < 20.0 do
  pause 0; (* môže sa nahradiť príkazom yield *)
end;

while Číslo < 20.0 do
  pause 5; (* pozastavenie činnosti programu na 5 sekúnd *)
end;
```

Príkaz pause je možné použiť výhradne v procedúre OnActivate. V žiadnych iných udalostných alebo užívateľských procedúrach táto inštrukcia nie je povolená! Ak však chceme obísť tento zákaz, môžeme nastaviť parameter *independent_procedure_execution* na *false* buď na začiatku projektu v kapitole *settings*, alebo v integrovanom vývojovom prostredí v Dátových inšpektoroch - Systém.

Príkaz **wait**

Príkaz **wait** pozastaví činnosť programu do splnenia podmienky za týmto kľúčovým slovom. Ak je hodnota logickej podmienky *true*, program pokračuje nasledujúcim príkazom v tom istom časovom kroku. Ak je výsledok logickej podmienky *false*, program poskytne systému čas k prípadnej obsluhe ďalších prístrojov a opäť testuje podmienku.

Príklad:

```
wait Číslo >= 20.0; (* čakanie na splnenie podmienky *)
```

Príkaz wait je možné použiť výhradne v procedúre OnActivate. V žiadnych iných udalostných alebo užívateľských procedúrach táto inštrukcia nie je povolená! Ak však chceme obísť tento zákaz, môžeme nastaviť parameter *independent_procedure_execution* na *false* buď na začiatku projektu v kapitole *settings*, alebo v integrovanom vývojovom prostredí v Dátových inšpektoroch - Systém.

Príkaz **stop**

Príkaz **stop** spôsobí ukončenie procedúry. Ďalšie volanie procedúry spôsobí jej rozbehnutie znovu od začiatku.

Príkaz **send**

Pomocou kľúčového príkazu **send** voláme iné prístroje v aplikácii. Príkaz aktivuje uvedený prístroj ihneď po skončení práve aktuálneho časového kroku. Pokiaľ chcete, aby prístroj aktivoval sám seba, môžete vypísať jeho meno, alebo použiť zástupné meno *self*. Príkaz **send** *self* znamená**:** „aktivuj sám seba v najbližšom systémovom časovom kroku“ (používa sa pre komunikáciu s V/V kanálmi).

Príkaz **send** má v niektorých prístrojoch ekvivalent v podobe parametra *receivers*, no ak voláme niekoľko prístrojov, musíme použiť viacej príkazov **send.**

Príkaz **send** je veľmi mocný. Ak ho použijeme, nemusíme prístroje vôbec časovať.

Príklad:

```
send Vypínač;  (* volanie prístroja Vypínač *)
send Žiarovka; (* volanie prístroja Žiarovka*)
send self;     (* prístroj volá sám seba    *)
```

Príkaz **sound**

Prístroj **sound** umožní prehrávanie všetkých zvukových súborov, ktoré podporuje Windows 9x a vyššie verzie. Za kľúčovým slovom je reťazový výraz vyjadrujúci meno súboru. Jeho funkcia je podmienená prítomnosťou príslušného hardvéru

Príklad:

```
if Číslo = 1 then
  sound 'gong1.wav'; (* volanie prístroja Vypínač *)
elsif Číslo = 2 then
  sound 'gong2.wav';
else sound 'gong3.wav';
end;
```

Príkaz **move**

Príkaz **move** slúži k vzájomnému priradeniu polí, alebo ich častí. Je rýchlejší ako vykonanie tejto požiadavky cez programovú slučku. Veľmi rýchly prenos sa dosiahne pri prenose medzi poľom kanálov a poľom premenných. Prvým parametrom je zdrojové pole, druhým cieľové pole a tretím počet priradení.

Príklad:

```
x = 5;
move a[1],b[3],x; (*pole a[1..5] sa prenesie do pole b[3..7] *)
```

Príkaz **priradenia**

Kdekoľvek v zápise algoritmu sa môže vyskytovať priraďovací príkaz, ktorý má obecnú podobu *dátový_element = výraz*. Dátovým elementom môže byť premenná, prvok alebo pole premenných, výstupný alebo obojsmerný kanál alebo pole výstupných alebo obojsmerných kanálov. Vpravo od symbolu „rovná sa“ je ľubovoľný výraz, ktorý však musí typovo súhlasiť s dátovým elementov vľavo. Výnimkou sú dátové elementy *shortint, integer, longint, shortcard, cardinal, longcard, shortreal, real*, pretože majú všetky rovnakú reprezentáciu v pamäti počítača (o tom bolo už písané v kapitole 4.0).

Príklad:

```
x = 1;
x = x + 1;
Text = ‘Čas’;
```

Volanie procedúr

Zápis programu dovoľuje odkazovať sa v algoritme na iné procedúry, ktoré týmto spúšťame. Zápis sa prevádza v tvare: meno prístroja, bodka, názov procedúry.

Napríklad:

```
Časové_relé.Proc_1();
```

Ak chceme volať procedúry, ktoré sú v inom module (o moduloch bude písané neskôr), píše sa ako prvé názov modulu.

Napríklad:

```
ModulČasovačov.Časové_relé.Proc_1();
```

V niektorých prípadoch sa v procedúre odvolávame na inú procedúru toho istého prístroja. V takom prípade nemusí byť pre názvom procedúry meno prístroja, ale toto meno môže zastupovať kľúčové slovo **self.**

Príklad:

```
self.Časové_relé.Proc_1();
```

Vykonávanie volaných procedúr sa deje v tom istom časovom kroku. Znamená to, že sa hodnoty na kanáloch nemenia. To je zásadný rozdiel od použitia príkazu **send**! Ak vo volanej procedúre použijeme pozastavujúce inštrukcie **pause**, **yield** alebo **wait**, vráti sa vykonávanie inštrukcií na volaný program bez dobehnutia inštrukcií volané procedúry! Po ukončení pozastavení pokračuje volaná procedúra v najbližšom časovom kroku (po *yield*), po uplynutí stanovenej doby po (*pause*) alebo po splnení podmienky (po *wait*) samostatne vrátanie prípadného merania kanálov (teda ako u *send*).

Procedúry nie je možné volať rekurzívne a to ani cez iné prístroje.

Komentáre

Komentár je doprovodný text, oboznamujúci s funkciou prístrojov alebo ich častí. Každý skúsený programátor nešetrí na komentároch. Veď kto si bude po nejakom čase pamätať, prečo práve tú procedúru (alebo premennú...) použil!

Komentár je od zápisu algoritmu oddelený komentárovými zátvorkami (* pre začiatok a *) pre koniec. Všetko, čo bude zapísané vo vnútri, je chápané ako komentár. Je teda možné pri ladení programu zazátvorkovať určité časti programu, na miesto ich mazaní.

Komentárové zátvorky môžu byť aj vnorené.

Príklad:

```
procedure OnActivate();
begin
 if Vypínač then        (* ak bude stlačený vypínač *)
   Svetlo = not Svetlo;  (* invertuj stav premennej *)
   send Žiarovka;              (* aktivuj aplikáciu Žiarovka *)
 else Svetlo = false;       (* pri vypnutí vypínača musí zhasnúť *)
   send Žiarovka;              (* aktivuj aplikáciu Žiarovka *)
end;
(* Pokus.Nič(); volanie procedúry Nič(*v Pokus*)zatiaľ nefunkčné*)
end_procedure;
```

Rekurzívne volanie a oneskorujúce inštrukcie

Ako už bolo povedané, nie je možné, aby procedúra volala sama seba a to ani cez iné procedúry. Výnimkou je volanie procedúry *OnActivate()*, ktorá v takom prípade pokračuje v testovaní podmienky behu a prípadne pokračuje od miesta prerušenia ďalej.

Zvlášť časté problémy môžu vznikať pri použití oneskorovacích inštrukcií v udalostných procedúrach. Pokiaľ sa vykonávanie udalostnej procedúry pozastaví a daná udalosť nastane znovu, opäť dôjde k rekurzívnemu volaniu.

Ako bolo spomenuté v predošlých kapitolách (oneskorovacie inštrukce), je možné toto obmedzenie obísť nastavením parametra *independent_procedure_execution* na *false_settings*, alebo v integrovanom vývojovom prostredí v Dátových inšpektoroch - Systém.

17.2 Udalostné a užívateľské procedúry

Teraz už vieme, že každú procedúru je možné volať z inej procedúry. Niekedy je nutné, aby sa niektoré procedúry zavolali iným mechanizmom. Túto možnosť poskytujú udalostné procedúry.

Udalostné procedúry sú dopredu naprogramované a vyvolávané priamo systémom. Je teda na nás, akým spôsobom ich v projekte využijeme.

Každý prístroj má zoznam možných udalostných procedúr. Ich zoznam nájdeme v *Inšpektori prístroja* v záložke *Procedúry*.

17.2.1 Štandardné udalostné procedúry

Množstvo udalostných procedúr je dané jednotlivými prístrojmi. Napriek tomu existuje určitá skupina zhodná v rámci celého systému.

OnActivate() bola popísaná v kapitole 7.1. Je spustená pri aktivácii prístroja.

OnMouseDown (*MouseX, MouseY : integer; LeftButton, MiddleButton, RightButton : boolean***)**
Bude vyvolaná pri stlačení tlačidla myši na ploche prístroja. Parametre *MouseX, MouseY* obsahujú súradnice kurzora myši vzhľadom k ploche prístroja. Ostatné parametre vracajú *true* pri stlačení príslušného klávesa myši.

OnMouseDoubleClick (*MouseX, MouseY : integer; LeftButton, MiddleButton, RightButton : boolean***)**
Bude vyvolaná pri dvojitom stlačení tlačidla myši (dvojklik) na ploche prístroja. Parametre *MouseX, MouseY* obsahujú súradnice kurzora myši vzhľadom k ploche prístroja. Ostatné parametre vracajú *true* pri dvojkliku na príslušnú kláves myši.

OnMouseUp (*MouseX, MouseY : integer; LeftButton, MiddleButton, RightButton : boolean***)**
Bude vyvolaná, ak ukončíme stlačenie tlačidla myši na ploche prístroja. Parametre *MouseX, MouseY* obsahujú súradnice kurzora myši vzhľadom k ploche prístroja. Ostatné parametre vracajú *true* pri ukončení stlačenia na príslušnom klávesu myši.

OnMouseMove (*MouseX, MouseY : integer; LeftButton, MiddleButton, RightButton : boolean***)**
Bude vyvolaná pri pohybe myši na ploche prístroja. Parametre *MouseX, MouseY* obsahujú súradnice kurzora myši vzhľadom k ploche prístroja. Ostatné parametre vracajú *true* pri stlačení na príslušnom klávesu myši.

OnKeyDown (*Charakter : cardinal***)**
Procedúra bude aktivovaná, keď je prístroj vybratý a bol stlačený kláves. ASCII kód klávesa je v parametri *Charakter*.

OnKeyRepeat (*Charakter : cardinal***)**

Procedúra bude aktivovaná, keď je prístroj vybratý a je držaný stlačený kláves. ASCII kód klávesa je v parametri *Charakter*.

OnKeyUp (*Charakter : cardinal***)**
Procedúra bude aktivovaná, keď je prístroj vybratý a stlačený kláves sa uvoľnuje. ASCII kód klávesa je v parametri *Charakter*.

OnSelect ()
Bude volaná pri výbere prístroja (začína prijímať vstup z klávesnice).

OnDeselect ()
Bude volaná pri strate výberu prístroja (končí príjem vstupu z klávesnice).

OnShow()
Bude volaná pri zobrazení predtým skrytého prístroja.

OnHide()
Bude volaná pri skrytí predtým viditeľného prístroja.

OnNewPosition(*RectX, RectY, RectW, RectD : integer***)**
Bude volaná pri zmene pozície a veľkosti prístroja. Nové súradnice sú predané v parametroch *RectX, RectY, RectW* a *RectD*. *RectX* a *RectY* je vztiahnuté k vizuálnemu vlastníkovi prístroja (panela).

OnWindowsMinimize()
Ak je prístroj v okne, procedúra bude volaná pri minimalizácii okna.

OnWindowsMaximize()
Ak je prístroj v okne, procedúra bude volaná pri maximalizácii okna.

OnWindowsRestore(*WasMinimized, WasMaximized : boolean***)**
Procedúra bude volaná pri obnovení veľkosti okna. Parametre *WasMinimized* a *WasMaximized* informujú, aký bol stav okna pred volaním procedúry (minimalizované alebo maximalizovaný).

OnWindowsClose()
Bude volaná pri pokuse o zatvorení okna. Štandardne nie je možné okna aplikácie zatvoriť, iba skryť alebo minimalizovať. Pokiaľ je v prístroji deklarovaná táto procedúra, okno stále nie je možné zatvárať, ale je povolené tlačidlo s krížikom v pravom hornom rohu titulku okna a táto procedúra bude aktivovaná.

17.3 Natívne procedúry

Natívne procedúry (skôr sa im hovorilo OCL metódy), sú vlastnosti prístrojov, ktoré môžeme ovládať prostredníctvom programu, teda inak povedané, za behu programu.

Definícia tej ktorej natívnej procedúry je závislá na triede prístroja. Typickým príkladom je ovládanie viditeľnosti prístroja. Každý panel má možnosť definovanie výrazu ovládajúceho jeho viditeľnosť.

Zoznam typických natívnych procedúr:
Hide() - skrytie viditeľného panela

Show() - zobrazenie viditeľného panela

Restore() - ak je panel v okne a toto okno je minimalizované alebo maximalizované, bude obnovená veľkosť okna pred minimalizáciou alebo maximalizáciou.

Update(*SetOutput : boolean* **)** - nastavenie prístroja podľa aktuálneho stavu výstupneho dátového elementu. Parameter SetOutput určuje, či budú nové dáta naspäť zapísané do výstupného dátového elementu.

SetValue(*Value : boolean* **)** - zápis novej hodnoty do prístroja spolu s prípadným nastavením výstupného dátového elementu.

Move(*dx : number, dy : number* **)** - presunie panel o zadané *dx, dy* relatívne k súčasnej pozícii.

MoveTo(*x, y : real* **)** - posunutie prístroja na pozíciu *x, y*. Veľkosť hodnôt *x* a *y* sú v bodoch obrazovky.

Disable() - zablokovanie prístroja. Zavolaním tejto metódy aktívna plocha ovládacieho prvku zosivie (bude prekrytá 50% rastrom) a prístroj nebude reagovať na niektoré udalosti od myši a klávesnice.

Enable() - odblokovanie prístroja skôr zablokovaného metódou Disable(). Objekt sa prekreslí do pôvodného vzhľadu.

Select() - prístroj je vybraný a prichádzajú do neho všetky udalosti od klávesnice. Prístroj má zmysel vybrať, len ak je vo viditeľnom paneli a je aj sám viditeľný. Ak má prístroj nastavený parameter *tab_select*, je možné ho vo vybranom stave ovládať klávesnicou.

AnimateClick(*MiliSec : number* **)** - animácia stlačenia klávesu po nastavenú dobu, danú parametrom *MiliSec* v milisekundách.

Blink(*BlinkingActive : boolean* **)** - zapnutie a vypnutie blikania. Volaním procedúry *Blink* s parametrom *BlinkingActive* nastaveným na *true* začne prístroj blikať (striedavo sa prekresľuje s použitím farebných sad *colors* a *blink_colors*). Blikanie je možno zastaviť volaním procedúry *Blink* s parametrom *BlinkingActive* nastavenom na hodnotu *false*. Frekvencia blikania je daná parametrom prístroja *blink_rate*.

ToggleBlink() - prepnutie vzhľadu prístroja do alternatívnej sady farieb, teda z *colors* do *blink_colors* alebo naopak podľa okamžitého stavu. Po zastavení blikania sa prístroj vždy vráti do štandardnéj farebnej sady bez ohľadu na predchádzajúceho volania procedúry *ToggleBlink*.

Size(*w : number, d : number* **)** - zmení veľkosť panela na hodnoty w a d. Pozícia ľavého horného rohu zostane zachovaná.

Natívne procedúry sú mocným nástrojom programovania a pomocou nich je možné napísať ľubovolný algoritmus.
Spôsob volania natívnej procedúry predstavuje zápis prístroja, bodka a názov natívnej procedúry.

Napríklad:

```
Ľavý_panel.Show();
```

17.4 Úloha č. 26

AKTIVITA

Modifikujte projekt simulácie prístrojovej dosky automobilu Favorit. Vytvorte programové časti, pomocou ktorých budú smerové svetla a výstražný trojuholník blikať

Rozbor úlohy

V riešeniach sa budeme odvolávať na Kapitolu 14.4 – Úloha č.16.

Blikanie smerových svetiel a výstražného trojuholníka realizujeme pomocou programov časovaných časovačom. Pre každé ovládané svetlo vytvoríme osobitný prístroj *program.*

Riešenie – premenné a konštanty

V *Dátových inšpektoroch* doplníme konštantu pre čas blikania a premenné, ovládajúce aktuálny stav svetiel.

```
const
.............
  Čas = 0.8;
end_const;

var
.................
  Relé_Vpravo = boolean, false;
  Relé_Vľavo = boolean, false;
  Relé_Výstraha = boolean, false;
end_var;
```

Riešenie – Smerovka

Pretože budú smerové svetlá ovládané priamo príslušným programov, odstránime z prístroja *Smerovka,* z parametra *receiver,* odkaz na prístroje S_Vľavo a S_Vpravo.

Riešenie – časovač

Ponúka sa možnosť použiť časovač *selector* pre zníženie záťaže systému, ktorý by časoval len práve potrebný prístroj *program*. No nepoužijeme ho. Dôvod je prostý. Pri zrušení blikania svetla sa ukončí časovanie príslušného programu a stav premennej ovládajúci svetlá by mohol zostať v stave *true*. Svetlo by nezhaslo a to by bola chyba. Preto v úlohe použijeme prístroj *seguencer* riadený časovou konštantou, ktorý bude ovládať prístroje *program*.

```
timer

  sequencer Časovač;
    timer = Čas;
  end_sequencer;

end_timer;
```

Riešenie – prístroje S_Vľavo, S_Vpravo, S_Výstraha

Úprava týchto prístrojov spočíva v úprave parametra *expression*. Zavedieme novo definované premenné, ktoré budú ovládané programom.

```
  indicator S_Výstraha;
.....
    expression = Relé_Výstraha;
.....
  end_indicator;


  indicator S_Vpravo;
 .....
    expression = Relé_Vpravo;
.....
  end_indicator;


  indicator S_Vľavo;
  .....
    expression = Relé_Vľavo;
.....
  end_indicator;
```

Riešenie – prístroje *program*

V prístrojoch *program* definujeme sekvenciu blikania príslušného svetla. Popíšme funkciu jedného z nich.

Program *Časové_relé*_Vľavo bude časovaný časovačom. Každý časový krok spustí procedúru *OnActivate()*, ktorá otestuje prítomnosť signálu *Vľavo,* pričom signál *Vľavo* je riadený prepínačom *Smerovka*.

Ak prepneme prepínač smeroviek do polohy Vľavo, hodnota premennej *Vľavo* sa nastaví na *true*. V tom istom časovom kroku sa prevedie inverzia hodnoty premennej *Relé_Vľavo* z hodnoty *false* na hodnotu *true* a aktivuje sa prístroj S_Vľavo. Ten skutočný stav zobrazí (smerové svetlo sa „rozsvieti“). V ďalšom časovom kroku sa prestaví premenná na opačnú

hodnotu, prístroj zobrazí nový stav, ... a tak ďalej do doby, než prepneme prepínač smerových svetiel do inej polohy.

Pri prepnutí prepínača smerových svetiel do inej polohy sa zmení hodnota premennej *Vľavo* z *true* na *false*. Podmienka *if* vykoná časť *else*. V časti else sa hodnota premennej *Relé_Vľavo* zmení na *false* a následne sa aktivuje prístroj *S_Vľavo*, ktorý tento stav zobrazí. Smerové svetlo „zhasne“.

```
program Časové_relé_Vľavo;
  timer = Časovač, 1;

 procedure OnActivate();
 begin
   if Vľavo then
      Relé_Vľavo = not Relé_Vľavo; (* blikaj *)
   else Relé_Vľavo = false;(* pri vypnutí vypínača musí zhasnúť *)
   end;
   send S_Vľavo;   (* aktivuj prístroj S_Vľavo                *)
 end_procedure;

end_program;
```

LITERATÚRA

[1] Antsaklis, P. J.: *Defining intelligent control*. IEEE Control Syst. Mag., vol. 14, pp. 4–66, 1994

[2] Benson, H.: Application of fuzzy set theory to data display. Fuzzy Sets and Systems, 1995, 2, 57-75

[3] Bezdek, J.: Pattern recognition with Fuzzy Objective Function Algorithms. International Journal Of Human-Computer Interaction, London, 1992, 2, 287-305

[4] Bien, Z.: How to measure the machine intelligence quotient (MIQ):Two methods and applications. IEEE Trans. Syst., Man, Cybern. A, vol. 27, pp. 256–262

[5] Brychta, J. – Šmejkal, L.: Výběr vizualizačního systému pro dispečerská pracovište. Automatizace 39, č.2, 1996, s.47-52. ISSN 0005-125X

[6] Dix, A. et. al.: Human-Computer interaction. Pearson Education 2003.

[7] Foulloy, L. – Zavidovique, B.: Toward symbolic process control. Automatica, vol. 30, no. 3, pp. 379–390, 1994.

[8] Hirsch, G.: Phonemic classification using a fuzzy dissimilitude relation. Fuzzy Sets and Systems, 1994, 5, 267-276

[9] Karwowski, W.: Fuzzy approach in psychological modeling of human operator manual lifting system. Fuzzy Sets and Systems, 1996, 14, 65-76

[10] Karwowski, W.: Fuzzy modelling of stresses in manual lifting system. Fuzzy Sets and Systems, 1997, 27, 341-349

[11] Kim, S.W. – Kim, B. K.: MIQ (Machine Intelligence Quotient) for process control system. IEEE Trans. Syst., Man, Cybern. , vol. 28, pp. 325–332

[12] Korteling, J. E. – Borg, W.: Partial camera automation in an unmanned air vehicle. IEEE Trans. Syst., Man, Cybern. A, vol. 27, pp. 256–262, Mar. 1997

[13] Kramer, U: Between Experience and Metaphysics. Journal of Experimental Psychology, Oxford, 1986, 105, 254-276

[14] Kramer, U. – Rohr, R.: A model of driver behavior.Analysis, Design and Evaluations of Man-Machine Systems, Oxford, 1993, 31-35

[15] Landryová, L. – Zolotová, I.: Integrated Environment of SCADA/HMI for Process Simulation. In: INES´99, 3rd IEEE International Conference on Intelligent Engineering Systems 1999, High Tatras, 1999, 469-472

[16] Lucrak, J. - GE, O.:A study on the interactions between stresses involved in manual lifting tasks. In: Proceedings of the Ergonomics Society Conference, London, 1997, 2, 95-100

[17] Mudrončík, D., Zolotová, I.: Priemyselné programovateľné regulátory. Konfigurovanie, vizualizácia, kvalita softvéru.. 1. vyd. Elfa, s.r.o., Košice : STU Bratislava, 2000. 169 s. ISBN 80-88964-45-8.

[18] Newman, W. M., Lamming, M G: Interactive System Design. Addison-Wesley 1995.

[19] Nishitani, H.: Human–computer interaction in the new process technology. J. Proc. Contr., vol. 6, no. 2/3, pp. 111–117, 1996.

[20] Onisawa, M.: Cognitive aspects of man-machine interactions. In: Proceedings of the Third European Annual Conference on Human Decision-Making and Manual Control, Denmark, 1998, 313-317

[21] Park, H. J. et al.: Measuring the Machine Intelligence Quotient (MIQ) of Human-Machine Cooperative Systems. IEEE Trans. On systems, Man and Cybernetics, part A, vol. 31, No.2, 2001. pp. 89-96

[22] Passino, K. M.: Intelligent control for autonomous systems. IEEE Spectrum, vol. 32, pp. 55–62, 1995.

[23] Perduková, D.: Vizualizácia technologických procesov. Študijný materiál pripravený v rámci projektu Leonardo da Vinci No SK/98/2/05381/PI/II.1.1c/CONT: Training in Electrical Engineering for Industry Automation, "ELINA". Mercury-Smékal, Košice 2001, pp.92, ISBN 80-89061-12-5

[24] Raskin, J: The Humane Interface: New Directions for Designing Interactive Systems. Addison-Wesley 2000

[25] Rasmussen, J.: Information Processing and Human–Machine Interation—An Approach to Cognitive Engineering. Computer, vol. 15, no. 2, pp.15–25, 1982

[26] Rencken, W. D. - Durrant-Whyte, H. F.: A decision-making human computer interface. In: Proc. IEEE Int. Conf. Systems, Man, Cybernetics, June 1988, pp. 148–150

[27] Rouse, W. B. et al.: Modeling the dynamics of mental workload and human performance in complex systems. IEEE Trans. Syst., Man, Cybern, vol. 23, pp. 1662–1671, Nov. 1993

[28] Shoureshi, R.: Intelligent control systems: Are they for real? J. Dynam. Syst., Meas., Control, vol. 115, pp. 395–401, 1993.

[29] Stassen, H. G. – Johannsen, G. – Moray, N.: Internal representation,internal model, human performance model and mental workload. Automatica, vol. 26, no. 4, pp. 811–820, 1999.

[30] Zimmerman H.J.: Fuzzy sets theory and its applications. IEEE Transactions on Systems, Man and Cybernetics, 1987 SMC-14, 112-120

[31] Zadeh L.A.: Fuzzy sets. Inf.Control, 8, 6, 1965, 338-353.

[32] Terano, W.: A fuzzy rule-based model of human problem solving. In: Proccedings of the 10th Congress of the International Ergonomic Association, London, 1991, 756-758

[33] Zolotová, I., Flochová, J.: Vizualizačné prostriedky, systémy SCADA/HMI (1). AT&P Journal, vol. 8, 2001, No.12, s. 28-29.

[34] Zolotová, I., Flochová, J.: Vizualizačné prostriedky, systémy SCADA/HMI (2). AT&P Journal, vol. 9, 2002, No.1, s. 26.

[35] Zolotová, I., Flochová, J.: Vizualizačné prostriedky, systémy SCADA/HMI (3). AT&P Journal, vol. 9, 2002, No.2, s. 62-63.

[36] Willaeys, S. – Malvach, T.: A framework for developement of fuzzy model for human-computer interaction. International Journal of Human-Computer Interaction, 2, 287-305

ZDROJE WWW

http://www.sigchi.org/

http://www.useit.com/

http://www.w3.org/TR/WAI-WEBCONTENT/

http://www.baddesigns.com/

http://www.iarchitect.com/mshame.htm

NÁZOV: Používateľské rozhrania
AUTOR: prof. Ing. Daniela Perduková, PhD.
RECENZENTI: prof. Ing. Pavol Fedor, PhD., doc. Ing. Janka Jablonská, PhD.
VYDAVATEĽ: Technická univerzita v Košiciach
ROK: 2015
ROZSAH: 199 strán
NÁKLAD: 70 ks
VYDANIE: prvé
ISBN 978-80-553-2050-2