SGP SYSTEMS

SGP BALTAZAR

PŘÍRUČKA PROGRAMU

FORMOU PŘÍPRAV

PRO UČITELE A RODIČE

+

SBÍRKA PŘÍKLADŮ

Témata jednotlivých příprav

1. **Jednoduché ovládání Baltazara z klávesnice**
2. **Baltedit – práce s grafickým editorem předmětů**
3. **Spouštění programu SGP Baltazar, spouštění hotových programů v prostředí SGP Baltazar, ukončení práce**
4. **Vytvoření nového programu, překlad, spuštění a vytvoření EXE**
5. **Sedit – práce s editorem struktogramů**
6. **Struktury posloupnost a opakování**
7. **Struktura rozhodování**
8. **Příkazy definované ve vzoru VZORBC.000 a jejich zobrazení**
9. **Speciální znaky v názvu prvků struktogramu, samostatná práce**
10. **Tedit – práce s textovým editorem příkazů**
11. **Další funkce knihovny Baltazar**
12. **Vytvoření prázdného programu dle vzoru VZORBC.PRG**
13. **Vytvoření nové funkce**
14. **Předávání parametrů funkcím a vracení hodnoty**
15. **Popis souboru .SGP**
16. **Další možnosti editoru struktur – Seditu, samostatná práce, soutěž**

Příloha: Sbírka příkladů

SGP Baltazar – příprava č. 1

Téma: Jednoduché ovládání Baltazara z klávesnice

Baltazar je kouzelník, který čaruje na obrazovce počítače podle přání a povelů, které mu zadávají žáci. Ovládání Baltazara si nejprve vyzkoušíme v interaktivním režimu pomocí programu vytvořeného v prostředí SGP Baltazar.

Učitel spustí žákům program KRESLI.SGP v prostředí SGP Baltazar (viz příprava č. 3). Tento program se nachází v adresáři, do kterého byl nainstalován SGP Baltazar.

Na obrazovce se objeví pracovní prostor a v něm kouzelník Baltazar, který si poklepává nožkou a čeká na stisk klávesy s povelem, jenž má vykonat.

Učitel vysvětlí význam jednotlivých kláves pro ovládání kouzelníka v interaktivním režimu podle zobrazené nabídky.

↑	- pohyb vpřed
←	- vlevo vbok
→	- vpravo vbok
C	- čaruje čtverec
B	- spouští editor obrázků předmětů Baltedit
*xx*Mezerník	- vyčaruje předmět číslo *xx* (tj. zadáme číslo a stiskneme Enter)
V	- umožní volbu předmětu pro čarování z palety všech předmětů
U	- uložení aktuální scény do souboru; jméno souboru můžeme zadat včetně názvu disku a cesty. (Není-li zadána přípona, program automaticky doplní příponu ".BSC".)
N	- načtení dříve uložené scény (viz volba U – načtení souboru KRESLI.BSC)

Zbývající část hodiny budou žáci samostatně pracovat s programem KRESLI, sestaví si svůj vlastní obrázek, který se naučí ukládat a načítat. Jako námět na kresbu obrázku může posloužit např. dům, ve kterém bydlí, či škola, do které chodí.

Učitel dbá na to, aby žáci pochopili, co je to příkaz, a že jimi zadaný příkaz musí být pro kouzelníka proveditelný.

Poznámka: Jako průpravu může učitel, ještě před zahájením práce s počítačem, vysvětlit pojem „příkaz“ tak, že jeden žák (žák-robot) vykonává před celou třídou příkazy jiného žáka (například: popojdi, vpravo vbok apod.) tak, aby žák-robot například přešel na určité místo ve třídě, nebo aby přenesl nějaký předmět z jednoho místa na druhé.

SGP Baltazar – příprava č. 2

Téma: Baltedit – práce s grafickým editorem předmětů

Baltedit umožňuje žákům vytvářet vlastní obrázky předmětů, které potom může Baltazar čarovat.

Učitel spustí žákům program SGP Baltazar a v něm Baltedit pomocí klávesy (Ctrl+B), nebo z menu.

Zobrazí se paleta obrázků-předmětů, kterou již známe z programu KRESLI. Tato paleta obsahuje 150 předmětů. V části palety jsou již hotové předměty. Každý předmět má své číslo, kterým je při čarování volán. U všech předmětů je možné nastavit průchodnost. Předmět, který je neprůchodný, Baltazar musí obejít a předměty průchodné může přejít. Neprůchodný předmět je také neprůhledný! Toto má velký význam při sestavování různých bludišť, kterými má Baltazar procházet apod.

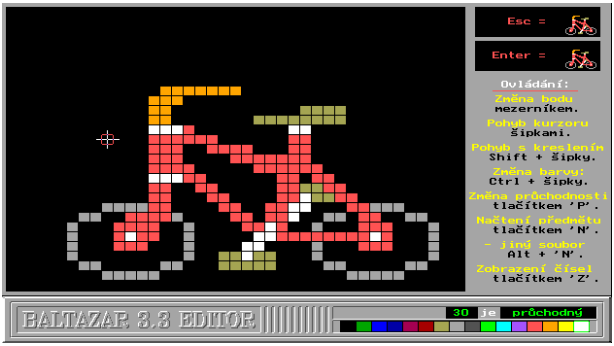
Předmět, který chceme editovat, vybereme pomocí čtvercového kurzoru ovládaného kurzorovými klávesami, myší nebo zadáním čísla zvoleného předmětu. Vlastní editor s vybraným obrázkem spustíme stisknutím klávesy Enter, nebo stisknutím levého tlačítka myši. Předmět se zobrazí ve zvětšené podobě.

V pravém dolním rohu obrazovky se zobrazuje, zda je předmět průchodný, či neprůchodný. Průchodnost měníme stiskem klávesy P.

Pod informací o průchodnosti předmětu je umístěna paleta barev, které je možné používat při editování předmětů. Výběr barvy z této palety se provádí pomocí kombinace kláves Ctrl+→ nebo Ctrl+←. Výběr barvy je také možno provádět pohybem myši při stisknutém pravém tlačítku myši. Bod, na němž je křížek, vybarvíme zvolenou barvou stiskem mezerníku nebo levým tlačítkem myši. Souvislou kresbu vytvoříme, držíme-li klávesu Shift společně s příslušnou kurzovou klávesou. Další možností je pohyb myší při stisknutém levém tlačítku.

Pro kreslení nového obrázku můžeme také využít možnosti načtení již hotového obrázku. Stiskneme klávesu N a již známým způsobem provedeme volbu předmětu pro načtení. Chceme-li obrázek načíst z jiného souboru obrázků, stiskneme Alt+N a zadáme název souboru včetně adresáře.

Ukončení práce a uložení vytvořeného předmětu se provádí stiskem klávesy Enter. Pokud obrázek nechceme uložit, stiskneme klávesu Esc. V každém okamžiku se v pravém horním rohu obrazovky zobrazuje předmět tak, jak bude vypadat, když stiskneme Enter, a jak bude vypadat po stisku Esc.



Pro konec editování v Balteditu slouží klávesa Esc.

V další části výuky se žáci seznámí se základy práce s Balteditem na konkrétním příkladu, když na obrazci č. 16 přetvoří usmívajícího se pána v zamračeného. Tuto práci bude celá skupina provádět společně pod vedením učitele.

Celá zbývající část hodiny bude vyplněna samostatnou prací žáků, kteří budou v tomto editoru připravovat svůj vlastní předmět.

Program SGP Baltazar ukončí žáci pod vedením učitele.

Poznámky

SGP Baltazar – příprava č. 3

Téma: Spouštění programu SGP Baltazar, spouštění hotových programů v prostředí SGP Baltazar, ukončení práce

Spuštění programu SGP Baltazar

Před spuštěním programu nastavíme aktuální adresář na pracovní adresář, ve kterém bude žák pracovat. Předpokládáme, že SGP Baltazar je řádně nainstalován a je k němu vytvořena cesta. Do pracovního adresáře by měl být nakopírován soubor BALTPRED.B00, aby si žák mohl tvořit vlastní předměty. Spuštění programu například pro žáka 5:

c:\>**g:**

g:\>**cd \skupina\zak5**

Program SGP Baltazar spustíme:

g:\skupina\zak5**>sgpbc**

nebo

g:\skupina\zak5>**h:\sgpbc\sgpbc** – v případě, že není nastavena cesta k SGP a SGP je v h:\sgpbc.

Na monitoru se objeví úvodní obrazovka systému SGP.

Nyní provedeme výběr souboru. Můžeme vybrat existující, nebo vytvořit nový program (přípona souboru .SGP). Vybereme z menu Soubor/Otevřít (nebo stiskneme klávesy Ctl+O) Objeví se seznam programů dodávaných na instalační disketě. Kurzor nastavíme na program LUKAS.SGP a stiskneme Enter.

Program LUKAS byl zaveden do paměti. Jeho spuštění provedeme stisknutím klávesy F9 -proveď program. Na obrazovce uvidíme Baltazara, jak netrpělivě poklepává nožkou – čeká, až stiskneme libovolnou klávesu. Běžící baltazarovský program můžeme ukončit předčasně stisknutím klávesy Esc (nebo Ctrl+Break). Po přerušení se hledá místo přerušení programu ve zdrojovém textu, a protože nás to v tuto chvíli nezajímá, stiskneme klávesu Esc pro ukončení hledání místa přerušení.

Takovýmto způsobem si můžeme vyzkoušet všechny vzorové programy.

Pozor! Pokud máte pouze demo verzi SGP Baltazara, větší programy vám již nepůjdou přeložit a spustit!

Žáky ponecháme, aby si sami vyzkoušeli spouštění ostatních, již hotových programů z aktuálního adresáře a hry, které učitel předem připraví do jiných adresářů podřízených adresáři, kde je Baltazar nainstalován.

Ukončení práce

Program ukončíme buď z menu nebo kobinací kláves Alt+X.

Poznámky

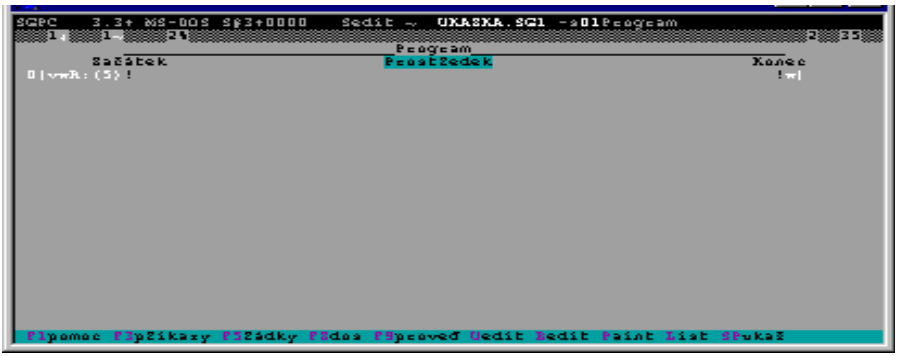
SGP Baltazar – příprava č. 4

Téma: Vytvoření nového programu

Spustíme program SGP Baltazar a z menu vybereme volbu Soubor/Otevři soubor. Namísto potvrzení napíšeme jméno nového programu.

Pozn: Pokud zadáme jméno programu, který již v daném adresáři existuje, nevytvoří se nový soubor, ale načte se ten, jehož jméno jsme zadali, neboť se předpokládá, že chceme pracovat se zadaným souborem.

Tentokráte napíšeme název nového programu (např. Program). Žáci napíší třeba svá vlastní jména bez diakritiky (maximální počet znaků je 8) a stisknou Enter.



Na obrazovce se objeví jednoduchý struktogram připravený dle vzoru. Vysvětlíme, že struktogram popisující program obsahuje jednotlivé prvky se jmény objektů s jejich postupným členěním na menší objekty (jsou zobrazeny černě).

Hlavní struktura (objekt) Program se skládá ze tří částí (objektů): Začátek, Prostředek, Konec (zleva doprava). Když objekty už nemohou být dále rozčleněny, přiřazují se k těmto „elementárním“ objektům operace (jsou zobrazeny bíle), které říkají, co se má s daným objektem udělat. Tyto operace se vykonávají postupně zleva doprava. Abychom stále věděli, co uvedené zkratky operací představují, zobrazují se neustále na dolním okraji obrazovky hlavičky definicí těchto maker (operací a podmínek).

Po struktogramu se můžeme pohybovat kurzorem pomocí kurzorových kláves nebo pomocí myši. Nové prvky struktogramu můžeme vkládat buď klávesami, nebo také myší. Od verze SGP Baltazar 5.0 je ovládání editoru struktur plně intuitivní. S žáky si však probereme i všechny ostatní možnosti.

```
                  _________________PROGRAM_____________________
      Začátek              Prostředek                  Konec
 0[vR:(5)!                                             !w]
```

Vysvětlíme jednotlivé operace pod objektem s názvem Začátek:

- 0 - deklarace proměnných (v našem případě není nutné)
- [- vytvoř grafický prostor s Baltazarem, Baltazar je zatím neviditelný
- v - učiň Baltazara viditelným
- R:(5) - nastav rychlost Baltazara, 0 – je nejmenší rychlost, 9 – je maximální rychlost

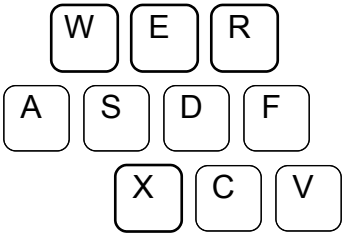
Operace pod objektem s názvem Konec:

- w - čekej na stisknutí klávesy
-] - zruš grafický prostor

Příkazy, které má Baltazar vykonat mezi objekty Začátek a Konec, budou uvedeny pod objektem Prostředek. Kurzor nastavíme na Prostředek.

Hotové nadefinované příkazy, které jsou k dispozici podle vzoru, zobrazíme stisknutím klávesy Ctrl+L – List. V seznamu najdeme příkaz `p Popojdi(1)`. Zobrazení seznamu ukončíme pomocí Esc. Příkazy jsou vlastně rozkazy, a proto se pro označení místa, kde mají být uvedeny, používá vykřičník.

Pro vložení nového prvku, vzhledem k prvku, na kterém máme kurzor, se používá kombinace Alt+šipka, nebo můžete použít klávesy:



Alt+šipka

nebo

W - vlož nový prvek vlevo od aktuálního prvku
R - vlož nový prvek vpravo od aktuálního prvku
E - vlož nový prvek nad aktuální prvek
X - vlož nový prvek pod aktuální prvek

Stiskneme klávesu X a napíšeme p! a Enter. Můžeme také ale napsat nejprve ! (tím systému sdělíme, že chceme vkládat příkazy, takže se pravé straně obrazovky automaticky zobrazí seznam všech maker, která můžeme v tuto chvíli použít. Příkazy můžeme uvádět s použitím zkratek, nebo v tzv. přímém zápisu. Zkratky mají stejnou délku (1 znak).

```
                  _________________PROGRAM_____________________
       Začátek              Prostředek                  Konec
0[vwR:(5)!                     p!                        !w]
```

Poznámka!: **Můžeme také použít tzv. zrychlené zadávání operací, tj. stačí pouze stisknout následující posloupnost kláves:** ***! p Enter.***
Při zadávání myší stačí pouze kliknout levým tlačítkem myši pod, nad, vpravo nebo vlevo od vyznačeného prvku a ze seznamu vpravo vybrat příslušné makro.

SGP Baltazar – příprava č. 4 – dokončení

Program spustíme pomocí klávesy F9.

Pokud chceme vytvořit samostpustitelný soubor PROGRAM.EXE stiskneme CTRL-F9.

Chceme-li příkaz opět editovat, nastavíme na něj kurzor a stiskneme Enter. Nyní můžeme vkládat další příkazy. Prvek operace s jediným příkazem `p` upravíme tak, že k příkazu `p` přidáme další tři příkazy `p`. Po opětovném stisknutí klávesy Enter je změněný prvek uložen, po stisknutí Esc zůstane takový, jaký byl před editováním. Upravený program spustíme opět klávesou F9.

```
_________________PROGRAM_____________________
     Začátek            Prostředek              Konec
0[vR:(5)!               pppp!                     !w]
```

Přímý zápis příkazů

Přímý zápis příkazů se provádí pomocí znaku „**:**“. Takto zapsaný příkaz se uvádí buď zcela samostatně, *nebo* ***musí být uveden jako poslední příkaz*** v případě většího počtu příkazů u jednoho vykřičníku. Prvek se čtyřmi samostatnými příkazy `p` upravíme tak, že jej nahradíme přímým zápisem ekvivalentního příkazu.

```
_________________PROGRAM_____________________
     Začátek            Prostředek              Konec
0[vR:(5)!              :Popojdi(4)                !w]
                          nebo
                 :Popojdi(4)!
```

Přímý zápis pokud možno nepoužíváme, protože svou zbytečnou podrobností značně znepřehledňuje struktogram. Používá se hlavně pro zadávání parametrů. Jednodušší zápis je tento:

```
_________________PROGRAM_____________________
     Začátek            Prostředek              Konec
0[vR:(5)!             P:(4)!                     !w]
```

Žákům umožníme, aby si vyzkoušeli i jiné příkazy ze seznamu příkazů, jako jsou například `c Čaruj(Čtverec)`, `C:(Číslo_Předmětu)` apod. Kouzelníka mohou nechat vyčarovat předmět, který dříve vytvořili v Balteditu.

Příkazy mohou uvádět také napravo od vykřičníku. POZOR – nezapomeňte na to, že **příkazy se vždy provádějí zleva doprava**. To znamená, že nejprve se provedou příkazy NALEVO od vykřičníku, pak všechny příkazy a bloky POD vykřičníkem a nakonec příkazy NAPRAVO od vykřičníku.

Prvek, na kterém je kurzor, můžeme zrušit tak, že stiskneme klávesu Del a Enter, nebo Ctrl+Y.

SGP Baltazar – příprava č. 5

Téma: Sedit – práce s editorem struktogramů

Zopakujeme téma minulé přípravy, ve kterém se soustředíme zejména na:

Způsob čtení a formu zápisu struktogramů

Připomeneme způsob čtení a formu zápisu elementárního struktogramu, se kterým se pracovalo v minulém tématu – členění jednotlivých objektů ve struktogramu na další menší objekty směrem dolů s tím, že objekty stejné úrovně jsou uvedeny na stejné úrovni i ve struktogramu v takovém pořadí, v jakém se s nimi pracuje, tj. vytváří *posloupnost* objektů.

```
_____________________PROGRAM_____________________
Začátek              Prostředek                 Konec
```

Po nadefinování jednotlivých objektů k nim přiřadíme operace, abychom vyjádřili, co se má v rámci daných objektů provádět.

```
_____________________PROGRAM_____________________
Začátek              Prostředek                 Konec
0[vR:(5)!               p!                        !w]
```

Pohyb kurzoru
- pomocí kurzorových kláves

Prvek struktogramu
- prvek struktogramu je posloupnost zobrazitelných znaků bez mezery.

V systému SGP se rozlišují tři typy prvků:
a) prvek **podmínka** – pro opakování (*, #, +) či větvení (/, \,%)
b) prvek **operace** či skupina operací – přiléhá zleva nebo zprava k „!“ bez mezery
c) prvek **název** (komentář) – všechny ostatní prvky

Editace prvku, na kterém se nachází kurzor
- pomocí klávesy Enter. Stisknutím klávesy Esc vrátíme prvek do jeho původního stavu, klávesou Enter změnu uložíme. Prvek Prostředek můžeme například změnit na Cesta.

```
_____________________PROGRAM_____________________
Začátek               Cesta                     Konec
0[vR:(5)!               p!                        !w]
```

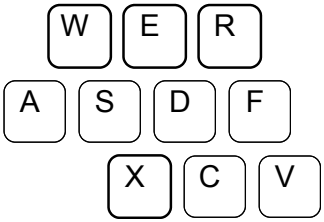
Zrušení prvku, na kterém se nachází kurzor (aktuálního prvku)

- pomocí Ctrl+Y (každý prvek)
- pomocí kláves Del s potvrzením Enter

Vkládání nových prvků

- pro vložení nového prvku vzhledem k aktuálnímu prvku, na kterém máme nastavený kurzor, se používají následující klávesy klávesy:

Alt+šipka



nebo levým tlačítkem myši kdekoliv nad, pod, vlevo, vpravo do aktuálního prvku

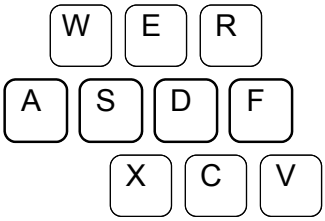
nebo

W - vložit vlevo od prvku
R - vložit vpravo od prvku
E - vložit nad prvek
X - vložit pod prvek

Po zopakování a upřesnění funkce výše uvedených kláves si probereme další klávesy, sloužící pro úpravu vzhledu struktogramu:

Posouvání prvků a bloků

Blok je skupina prvků, které jsou hierarchicky podřízeny aktuálnímu prvku, včetně tohoto aktuálního prvku. Pro posuv prvků a bloků slouží následující klávesy:



S - posuv názvu prvku vlevo
D - posuv názvu prvku vpravo

A - posuv bloku vlevo nebo Ctrl+šipka
F - posuv bloku vpravo nebo Ctrl+šipka
posuv bloku nahoru - Ctrl+šipka nahoru
posuv bloku dolů - Ctrl+šipka dolů

Mazání bloku

Celý blok můžeme smazat pomocí kláves Ctrl+Del.

SGP Baltazar – příprava č. 5 – dokončení

Kopírování bloku a přesun bloku

Ctrl+C - vloží blok do schránky
Ctrl+X - vyjme blok a vloží do schránky
Ctrl+V - vloží blok ze schránky na místo blikajícího kurzoru, jinak pod aktuální prvek
Ctrl+Alt+šipka - vloží blok ze chránky nad, pod, vlevo nebo vpravo od aktuálního prvku
Ctrl+Del - smaže blok

Posouvání struktogramu plochy

Shift + → - posun vpravo
Shift + ← - posun vlevo
Shift + - posun nahoru
Shift + ↓ - posun dolů

Pro vyzkoušení výše uvedených úkonů při práci s editorem struktogramů, mohou jako úlohu žáci zakreslit třeba svůj denní program.

Práci s editorem struktogramů ukončíme pomocí

F2 - struktogram bude zapsán na disk a zobrazí se seznam funcí
Esc - zobrazení seznamu funkcí, po výběru nové funkce se program se zeptá, jestli struktogram má být zapsán na disk. Odpovíme „Ano“ nebo „Ne“ stiskem klávesy A nebo N.

Poznámky

SGP Baltazar – příprava č. 6

Téma: Struktury posloupnost a opakování

Na tabuli nakreslíme struktogram z již vyzkoušeného programu.

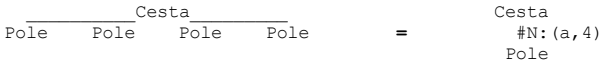
Struktura posloupnost

Zopakujeme, že jednotlivé objekty jsou v nakresleném struktogramu uvedeny v takovém pořadí, v jakém se s nimi bude pracovat. Nejprve Začátek, potom Cesta a nakonec Konec. V případě více operací, které jsou uvedeny u určitého objektu, se tyto operace také provádějí jedna za druhou v pořadí, v jakém jsou uvedeny u daného vykřičníku, a to zleva doprava.

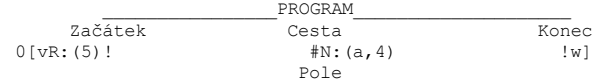
Při vysvětlování posloupnosti můžeme žáky vybídnout, aby uvedli různé příklady posloupnosti činností z jejich života.

Struktura opakování

Opakování je speciální případ posloupnosti. *Opakování je posloupnost stejných prvků.*

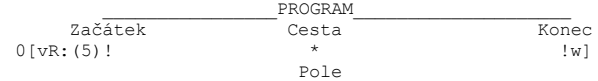


Vrátíme se opět k výše uvedenému struktogramu. Posloupnost čtyř příkazů `p` můžeme také nahradit opakováním jediného příkazu `p`.



Zápis `#N:(a,4)` znamená: opakuj čtyřikrát prvek Pole, tj. čtyřikrát se provede příkaz `Popojdi(1)`. Proměnná `a` bude postupně nabývat hodnot 0,1,2,3. Toto je příklad „počítaného“ opakování – opakování s předem daným počtem kroků. Používáme jej tehdy, známe-li předem počet opakování.

V případě, kdy nevíme dopředu, kolikrát má být určitý prvek nebo určitá činnost opakována, použijeme jiný typ opakování – obecnou strukturu opakování „*“.



Cesta se skládá z opakování polí. Jedno pole Baltazar přejde na příkaz ***p***.

Symbol „*“ znamená: *opakuj, když…* nebo také *opakuj dokud platí…*

Baltazar však musí vědět, kdy má v opakování činnosti přestat. Před vykonáním příkazu, který má být opakován, je nutno zjistit, zdali se v opakování má ještě pokračovat. Do struktogramu doplníme tedy ještě potřebnou podmínku.

```
                    _____PROGRAM_____
     Začátek          Cesta                Konec
0[vR:(5)!             *p                    !w]
                      Pole
                      p!
```

Struktura `*p` znamená opakuj, dokud předmět před Baltazarem je průchodný. Činnost, která má být opakována, se provede jenom v tom případě, když předmět před Baltazarem je průchodný. Když předmět nebude průchodný, bude třeba Okraj, opakování se automaticky ukončí a program pokračuje ve zpracování další části naší posloupnosti, tedy části Konec.

Oba typy opakování je možné nejprve vyzkoušet s žáky ve třídě formou hry, kdy žák má přejít určitý úsek ve třídě. Žák se chová, jako kdyby neviděl. V prvním případě opakování kroky počítá sám, ve druhém případě musí žák před uděláním každého kroku pokládat jinému žákovi stále stejnou otázku, např. jestli je před ním lavice, aby věděl, zda má v chůzi pokračovat.

Po názorném vysvětlení přistoupíme k práci s počítačem. Zadáme úkol, ve kterém Baltazar má přejít celou obrazovku (nevíme, kolik polí je široká). Řešení bude odpovídat výše uvedenému struktogramu včetně podmínky pro ukončení opakování.

Zadání můžeme rozšířit tak, aby Baltazar přešel celou obrazovku (nevíme, kolik polí je široká), otočil se, vyčaroval květinu, přešel obrazovku nazpět a otočil se, aby byl otočen směrem dovnitř prostoru.

```
                          _____PROGRAM_____
     Začátek   Cesta     Pravy~okraj      Cesta     Levý~okraj  Konec
0[vR:(5)!w      *p          z!C:(9)         *p          z!         !w]
               Políčko                    Políčko
                p!                         p!
```

Důležité!

Protože správné zpracování prvního a posledního pole je v praxi nejčastějším problémem, zadáme žákům ještě jednu typickou úlohu. Tuto úlohu je třeba důkladně vysvětlit a několikrát zopakovat. Úkolem Baltazara bude přejít celou obrazovku a na každém poli vyčarovat květinu.

```
                    _____Baltazar~3_____
     Začátek              _____Cesta_____          Konec
0[vR:(5)!w          První                Ostatní     !w]
                     pole                 pole
                 pzC:(12)!pz               *p
                                          Políčko
                                    C:(12)!p
```

Cesta je tedy posloupností Prvního pole a pak Ostatních polí. První pole, tj. pole, na němž stojí Baltazar, se musí zpracovat jinak, než všechna ostatní (stejná) pole.

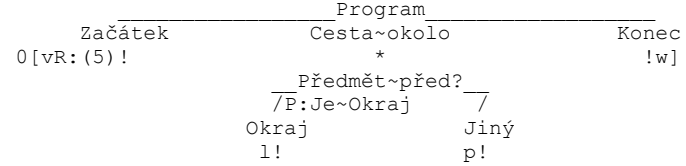
SGP Baltazar – příprava č. 7

Téma: Struktura rozhodování

Strukturu rozhodování používáme v programu vždy, když dopředu nevíme, kterou z různých činností budeme v daném okamžiku provádět. Až v daném okamžiku, za určité podmínky, provedeme určitou činnost. Jestliže podmínka splněna není, provedeme jinou činnost.

Naprogramujeme Baltazara tak, aby chodil stále okolo obrazovky.

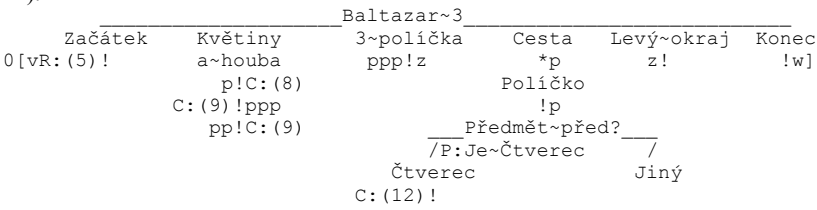
Řešení: Když dorazí k okraji prostoru, otočí se doleva a bude pokračovat dál.



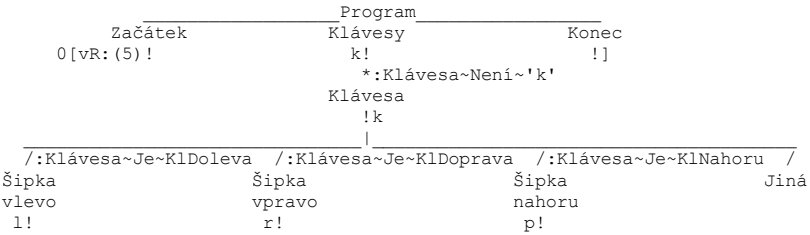
Baltazar bude chodit stále dokola obrazovky, protože ve struktogramu není u „*“ uvedena podmínka pro ukončení opakování. Program přerušíme klávesou Esc.

Poznámka: Kdyby se měl Baltazar zastavit na výchozí pozici, museli bychom při zpracování prvního pole před opakováním činnosti zjistit jeho souřadnice a ty vyhodnocovat v podmínce opakování pro zpracování ostatních polí. Protože takováto podmínka není ve vzoru připravená, museli bychom ji definovat v Teditu, který ještě nebyl probrán.

Rozhodování předvedeme ještě na dalším příkladu, ve kterém Baltazar nejprve vyčaruje květiny a houbu tak, že mezi nimi zůstanou volná místa, která budou vyplněna černými čtverci. Baltazar dále postoupí a obrátí se. Při zpáteční cestě bude zjišťovat, jestli předmět před ním je čtverec. V případě, že ano, vyčaruje na jeho místě stromeček (předmět číslo 12).



Nyní si zkusíme naprogramovat Baltazara tak, aby poslouchal naše povely zadávané z klávesnice. Program bude opakovaně zjišťovat povely tak, že bude číst klávesnici a zjišťovat, jaká klávesa byla stisknuta. Když bude stisknuta šipka doleva, natočí se Baltazar doleva. Při stisknuté šipce doprava se natočí vpravo. Stiskneme-li šipku nahoru, popojde o jedno políčko. Program ukončíme stisknutím klávesy malé „k“.

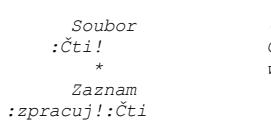


Ve výše uvedeném struktogramu jsou výrazy pro vyhodnocení podmínek poněkud dlouhé, protože jsou napsány jako přímý text. Až začneme používat Tedit pro jejich definování, budou moci být i složité podmínky nahrazeny pouze jedním znakem.

Povšimněme si též, jakým způsobem je provedeno opakování. Před opakováním je přečtena klávesa z klávesnice, aby vůbec nedošlo k provádění opakování v případě stisknutí klávesy „k“. Jako poslední operace bloku „Klávesa“ je opět operace „k“.

Poznámka pro učitele, kteří znají PASCAL

Tento způsob programování se nazývá „čtení napřed“. Záznam ze vstupního souboru nejprve načteme do paměti, a teprve pak jej zpracováváme. Obecné schéma opakování při čtení vstupního souboru (klávesnice, textový soubor na disku apod.) je následující.




```
Odpovídající kód v Pascalu

{Soubor}
Čti;
while True do begin
    zpracuj;
    Čti;
end
```

Žáci mohou program upravit tak, aby Baltazar při stisknutí určité klávesy vyčaroval čtverec nebo jiný, předem určený, předmět.

Poznámky

SGP Baltazar – příprava č. 8

Téma: Příkazy definované ve vzoru VZORBC.000 a jejich zobrazení

Příkazy a podmínky definované ve vzoru VZORBC.000 jsou k dispozici v každém novém programu, který byl vytvořen podle tohoto vzoru. V prostředí Baltazara můžeme zobrazit seznam příkazů a podmínek v Seditu stisknutím klávesy L volby List.

Seznam příkazů a podmínek:

```
-o1-VZOR  operace
0 ;proměnné celočíselné: a,b,c,d,e,f,g,h,i,j,Klávesa;      -32768..32767
                    int  a,b,c,d,e,f,g,h,i,j,Klávesa,
                         CČíslo1,CČíslo2,CČíslo3,PůvBarvy;

  ;proměnné reálné: RČíslo1,RČíslo2,RČíslo3;          5.0E-324..1.7E308
             double  RČíslo1,RČíslo2,RČíslo3;
  ;proměnná znak:   Znak;
             char    Znak;
  ;proměnná řetězec:Řetězec[41];              řetězec maximálně 40 znaků
                char Řetězec[41];
[ VytvořProstor()
] ZrušProstor()
a Klávesa= ČtiKlávesuZFronty()
b PřepniNaBaltazara()
c Čaruj(Čtverec)
d GrČtiZnak("?",Znak,1,1,0x07)           /* 1zn, edit ano, černá/šedá*/
e BezObláčku()
f VyprázdniFrontuKláves()
g GrČtiŘetězec("?",Řetězec,40,1,0x3f)    /*40zn, edit ano, modrá/bílá */
h GrČtiČíslo("?",CČíslo1,6,0,0x0e)       /* 6zn, edit  ne, černá/žlutá*/
i GrČtiReálnéČíslo("?",RČíslo1,20,1,0x0f)/*20zn, edit ano, černá/bílá */
j GrPišZnak(Znak)
k Klávesa= ČtiKlávesuZFrontySČekáním()
l VlevoVbok()
m ;vypiš Řetězec do grafického okénka
   PůvBarvy=GrBarvy(Modrá<<4|Žlutá); /* modrý podklad, žluté písmo    */
   GrPišŘetězec(Řetězec);            /* vypíše obsah proměnné Řetězec */
   GrNastavBarvy(PůvBarvy);          /* nastaví původní barvy         */
n Neviditelný)
o SObláčkem()
p Popojdi(1)
q GrPišČíslo(CČíslo1,-1)             /* -1 = nejkratší výpis          */
r VpravoVbok()
t PřepniNaText()
u GrPišReálnéČíslo(RČíslo1,10,2)     /* 10 číslic, z toho 2 desetinné */
v Viditelný()
w ;čekej na klávesu, smaž klávesu
   ČekejNaKlávesu();
   VyprázdniFrontuKláves();
x
y
z ;čelem vzad
   VlevoVbok(); VlevoVbok()
B NastavBarvuČarování   /*       B:(barva)!   0-15                    */
C Čaruj                 /*     C:(předmět)!   1-150                   */
L NačtiScénu            /*    L:("soubor")!   "c:\\pavel\\soubor.bsc" */
N NáhodněČaruj          /*     N:(předmět)!   1-150                   */
P Popojdi               /*  P:(počet_polí)!   0-14                    */
R RychlostBaltazara     /*    R:(rychlost)!   0-9                     */
S UložScénu             /*    S:("soubor")!   "c:\\pavel\\soubor.bsc" */
W Čekej                 /*  W:(milisekund)!   0-32000                 */
X ČarujNaPozici         /* X:(předmět,x,y)!   1-150, 1-15, 1-10       */
-c1-VZOR  podmínky  /=if *=while #=for          příklad použití
```

```
n ;předmět před Baltazarem je neprůchodný
    Neplatí PrůchodnostPředmětuPředB()

p ;předmět před Baltazarem je průchodný
    PrůchodnostPředmětuPředB()
B ;barva čtverce před Baltazarem ....      /B:Je~barva    0-15
    BarvaČtvercePředB()
P ;předmět před Baltazarem ....            /P:<=předmět   1-150
    PředmětPředB()
S ;směr Baltazara ....                     /S:Není~4      1-4
    SměrBaltazara()
#A ;opakuj vzestupně pro proměnnou od,do,krok   #A:(i,2,10,2)  i=2,4..10
    OdDoKrok
#D ;opakuj sestupně  pro proměnnou od,do,krok   #D:(i,10,2,2)  i=10,8..2
    OdDolůDoKrok
#M ;opakuj n-krát sestupně  pro proměnnou       #M:(i,8)  (8x) i=7,6..0
    DolůDo0
#N ;opakuj n-krát vzestupně pro proměnnou       #N:(i,8)  (8x) i=0,1..7
    Od0
===VZOR
```

Jednotlivé příkazy a podmínky využívají funkce, které jsou součástí knihovny Baltazara (podmínky # jsou makra). Jednotlivé funkce popíšeme s využitím integrovaného helpu příkazů, který se spouští klávesou F3 (nebo Shift+Mezerník, případně Ctrl+Mezerník).

Okno „Plné detaily“

Když máme kurzor nastavený na operaci (skupině operací) nebo podmínce ve struktogramu, můžeme stisknout mezerník, aby se nám zobrazilo okno „Plné detaily“. Toto okno zobrazí definice operací, na kterých se nachází kurzor. Definice jednotlivých operací jsou odděleny čárou. Po opětném stisknutí mezerníku nebo jiné klávesy toto okno zmizí.

Srovnejme si okno „Plné detaily“ srovnáme s oknem „Detaily“, které je zobrazováno průběžně v dolní části obrazovky. Okno „Detaily“ zobrazuje jenom první řádky definic jednotlivých operací. Tuto skutečnost si můžeme ověřit na příkazu „z“.

SGP Baltazar – příprava č. 9

Téma: Speciální znaky v názvu programové jednotky v Seditu, samostatná práce

Název programové jednotky – komentář

Prvek struktogramu, který není vykřičníkem, podmínkou nebo operací je název. Název se skládá z vazebních čar pro podřízené prvky a z těla názvu. Tělo názvu je posloupnost libovolných znaků s výjimkou mezery a vykřičníku. Podtržítko lze v těle názvu použít kdekoliv.

Název programové jednotky může být zvolen libovolně. Neměl by být příliš dlouhý kvůli přehlednosti zápisu, ale je výhodné, pokud už sám název napovídá o funkci programové jednotky. Je-li potřeba lépe popsat programovou jednotku, je možno název rozepsat do více řádků (názvů) pod sebe.

Speciální znaky

Je-li jako první znak použit některý z následujících znaků, mají tyto znaky speciální význam:

Znak " (uvozovky) – definice programové jednotky
Definuje programovou jednotku, která může být ve struktogramu použita vícekrát. Na dalších místech již není nutné rozkreslovat tutéž programovou jednotku, ale stačí pouze odkázat se na její definici.

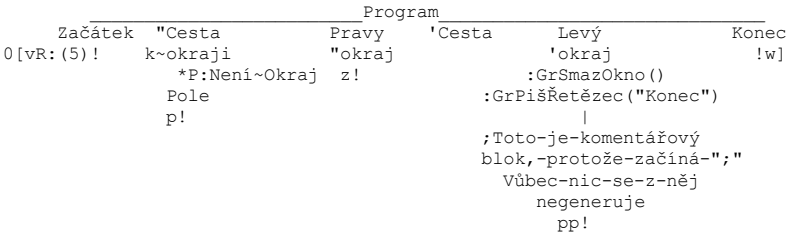
Znak ' (apostrof) – volání podstruktury
Znamená odkaz na programovou jednotku, která je definována na jiném místě ve struktogramu.

Znak : (dvojtečka) – přímý text
Přímý text (příkaz) nemusí být uveden v seznamech operací a podmínek.

Znak ; (středník) – komentářová programová jednotka
Název, jehož první znak je středník, znamená, že celá programová jednotka označená tímto názvem je pouze komentář. Komentářová jednotka je preprocesorem ignorována. Použití komentářových struktur je zvláště výhodné při testování a ladění částí programu, kdy nechceme, aby se vykonávalo všechno.

Samostatná práce

Žákům zadáme některé lehčí zadání ze sbírky příkladů, jako jsou např. pyramida či šachovnice a necháme je vyzkoušet si všechny speciální znaky.


Program
Začátek
0[vR:(5)!
"Cesta
k~okraji
*P:Není~Okraj
Pole
p!
Pravy
"okraj
z!
'Cesta
Levý
'okraj
:GrSmazOkno()
:GrPišŘetězec("Konec")
;Toto-je-komentářový
blok,-protože-začíná-";"
Vůbec-nic-se-z-něj
negeneruje
pp!
Konec
!w]

Poznámky

SGP Baltazar – příprava č. 10

Téma: TEdit – práce s textovým editorem příkazů

Doposud jsme ve struktogramech používali již připravené příkazy, protože jsme při vytváření nového programu jsme použili vzorový soubor (např. VZORBC.000) nebo přímé vkládání příkazů z Příkazového pomocníka (klávesa F3).

Prostředí SGP Baltazara však umožňuje existující příkazy upravovat a vytvářet nové vlastní příkazy. K definování příkazů slouží uživatelův textový editor – Tedit (Textový editor).

Tedit může být spuštěn téměř kdykoliv stiskem kombinace kláves Ctrl+Enter. Spustíme-li Tedit z prostředí Seditu, kdy kurzor není nastaven na žádnou operaci či podmínku, otevře se nám okno, s kurzorem nastaveným na funkci, kterou máme zobrazenou v Seditu. Byl-li však kurzor v Seditu nastaven na skupinu operací, otevře se nám nejprve okno pro volbu operace, a pak, po stisknutí klávesy Enter, se otevře okno Teditu s kurzorem nastaveným na začátek definice příslušné operace. V případě kurzoru na jedné operaci nebo podmínce v Seditu se Tedit otevře přímo s nastavením kurzoru na začátek příslušné definice.

Tedit můžeme také spustit, máme-li zobrazeno okno „Plné detaily“, nebo seznam operací a podmínek pomocí klávesy Ctrl+L. I tady si v případě většího počtu operací a podmínek můžeme nejprve vybrat v příslušném okně, kterou operaci či podmínku chceme editovat. Po stisknutí klávesy Enter se kurzor v Teditu nastaví přímo na danou operaci nebo podmínku.

Práci s Teditem si vyzkoušíme tak, že nejprve vytvoříme nový program. Pod Prostředek vložíme posloupnost příkazů `pcpz`.

```
_____________PROGRAM_____________
      Začátek          Prostředek          Konec
0[vwR:(5)!             pcpz!                 !w]
```

Program si můžeme spustit (F9). Baltazar popojde, vyčaruje čtverec, vstoupí na něj a udělá čelem vzad. Kurzor ponecháme na operacích `pcpz` a stiskneme Ctrl+Enter. V zobrazeném okně najedeme kurzorem na operaci `z` a stiskneme Enter. Otevře se nám Tedit s kurzorem nastaveným na:

```
z ;čelem vzad
   VlevoVbok(); VlevoVbok()
```

Jméno operace je tvořeno jedním písmenem na první pozici prvního řádku definice operace. Za jménem operace následuje alespoň jedna mezera, a za ní je uvedena vysvětlující poznámka, která bude zobrazována jak v seznamu operací a podmínek, tak v okně „Detaily“. Poznámka na tomto prvním řádku definice je v našem případě uvedena středníkem, což je "SGP poznámka". Můžeme samozřejmě použít i komentářových závorek jazyka C; pro vložení poznámky kdekoli v definici příkazu pak používáme céčkovské komentářové závorky – dvojici znaků „/*“ pro začátek poznámky a „*/“ pro konec poznámky.

Do operace zapisujeme příkazy v souladu se syntaxí programovacího jazyka C podle normy ANSI. Operace se tedy může sestávat z jakéhokoliv kódu v jazyce C s voláním funkcí knihovny Baltazara. My však můžeme využití pravidel syntaxe jazyka C prakticky

omezit na volání funkcí z knihovny Baltazara, které musí být odděleny středníkem (viz operace `z`), používání výše uvedených poznámek typu `/* Poznámka */`, deklarace proměnných, budeme-li je potřebovat, a přiřazovací příkazy typu `a = 3`. Za posledním příkazem v definici operace v Teditu, středník být uveden nemusí, systém SGP jej doplňuje automaticky do generovaného kódu, neboť jazyk C vyžaduje středník za každým příkazem, a tedy i za posledním.

Operace a podmínky v Teditu můžeme editovat jako v běžném editoru. To znamená, že v seznamu operací a podmínek mohou být upravovány, rušeny nebo vkládány další. Musíme dát jenom pozor, abychom kurzor neposunuli na jiné místo, než které skutečně chceme editovat. Operace editujeme v seznamu operací (-o1-název_struktury), podmínky v seznamu podmínek (-c1-název_struktury) pro danou strukturu. Klíčové řádky -s, -o1, -c1, =SP nesmíme v žádném případě žádným způsobem měnit! Došlo by k narušení souboru .SGP. Výše uvedenou operaci `z` upravíme následujícím způsobem:

```
z ;zobraz zprávu
   GrSmažOkno();
   GrPišŘetězec("Ahoj! Stiskni nějakou klávesu...");
```

Tedit ukončíme stisknutím klávesy F2, po kterém se provedené změny zapíší. Klávesou Esc ukončíme Tedit s tím, že budeme muset odpovědět na otázku, jestli chceme či nechceme, provedené změny zapsat na disk - stiskneme klávesu A.

Program si vyzkoušíme. Potom zkusíme najít operaci „z“ v seznamu operací a podmínek. Buď po stisknutí klávesy Ctrl+L, nebo v okně „Detaily“, anebo v okně „Plné detaily“ po stisknutí mezerníku.

Poznámka: *Provedená úprava operace* `z` *bude platná pouze pro tento program. Při vytvoření nového programu dle vzoru VZORBC.000 se opět z tohoto vzoru přenese operace* `z ;čelem vzad` *do našeho nového programu.*

Příkazový pomocník (SGP Helper) - novinka od verze SGP Baltazar 3.3 Plus.

Pokud chceme vložit do operace nový příkaz z knihovny Baltazara, můžeme s výhodou použít novou funkci systému SGP, a to Příkazového pomocníka.

Stiskněte klávesu F3 (příp. Shift+Mezerník nebo Ctrl+Mezerník) a zobrazí se seznam všech příkazů a konstant Baltazara. Kurzorovými klávesami si vyhledejte potřebný příkaz a klávesou Enter jej vložte do své operace v Teditu. Klávesami Ctrl+Enter můžete vložit celý příklad, na kterém si můžete vyzkoušet použití tohoto příkazu.

Druhý způsob použití Příkazového pomocníka SGP je, že pokud alespoň přibližně víte, jak se hledaný příkaz jmenuje, stačí zadat pouze několik písmen z jeho jména a ze seznamu se vyberou pouze ty příkazy, které obsahují vámi zadaný řetězec.

Tak například, zadáte-li "gr" - vyhledají se všechny příkazy, které pracují s grafickým okénkem, zadáte-li "kl" - vyhledají se všechny příkazy, které mají něco společného se zpracováním klávesy, atd. Doporučujeme - **vyzkoušejte si Příkazového pomocníka SGP !**

SGP Baltazar – příprava č. 11

Téma:

Další funkce knihovny Baltazar, relační operátory a konstanty

S pomocí Teditu můžeme definovat vlastní operace-příkazy. Jak již bylo řečeno, nový příkaz definujeme tak, že na první pozici prvního řádku definice uvedeme jeho označení (jeden znak), mezeru, středník, a za středník stručně popíšeme činnost, kterou příkaz provádí.

```
5 ;napsání výzvy a přečtení klávesy
```

Na další řádky napíšeme kód v syntaxi jazyka C, který bude spočívat především v přiřazovacích příkazech a voláních funkcí z knihovny Baltazara, oddělených středníky.

```
5 ;napsání výzvy a přečtení klávesy
   PišŘetězec('Zadej volbu: ');
   Klávesa = ČtiKlávesuZFrontySČekáním()
```

S využitím seznamu příkazů pro Baltazara a Příkazového pomocníka SGP, vysvětlíme funkce z níže uvedených skupin příkazů tak, aby o nich žáci získali rámcovou představu, tj. uměli se v nich orientovat. Skupiny funkcí:

- **příkazy pro Baltazara – doplníme je o další, které nejsou uvedeny v seznamu příkazů**
- **příkazy poskytující formou dotazu různé informace**
- **příkazy pro čtení z klávesnice a psaní na obrazovku v textovém režimu**
- **příkazy pro čtení z klávesnice a psaní do grafického okna (vysvětlíme rozdíl mezi použitím příkazů pro grafické okno a pro obrazovku v textovém režimu)**
- **příkazy pro práci s časem**
- **příkazy pro práci s porty (budeme-li je chtít využívat)**
- **matematické funkce**
- **ostatní funkce**

Dále vysvětlíme relační operátory tak, aby jim žáci dobře rozuměli.

Relační operátory

Operátor „C“	Možná náhrada	Příklad použití
==	Je	PředmětPřed() Je Čtverec
!=	Není	BarvaČtverce() Není Černá
<	JeMenšíNež	
>	JeVětšíNež	
<=	JeMenšíNeboRovno	
>=	JeVětšíNeboRovno	
!	Neplatí	Neplatí Průchodný()

&&	ASoučasně	PředmětPředBaltazarem() Je Čtverec ASoučasně BarvaČtverce() Je Bílá
||	nebo	Směr() Je Sever nebo Směr() Je Jih

Konstanty

Rámcově vysvětlíme různé konstanty, které mohou být v programech pro Baltazara použity.

Výčet funkcí knihovny Baltazar, relační operátory a konstanty je možno zobrazit stisknutím F3 nebo Shift+Mezerník v Seditu nebo Teditu.

Použití jednotlivých funkcí, relačních operátorů a konstant můžeme demonstrovat na jednoduchých příkladech.

Po vysvětlení látky žákům umožníme, aby si použití některých funkcí a konstant vyzkoušeli na vlastních jednoduchých příkladech.

Poznámky

SGP Baltazar – příprava č. 12

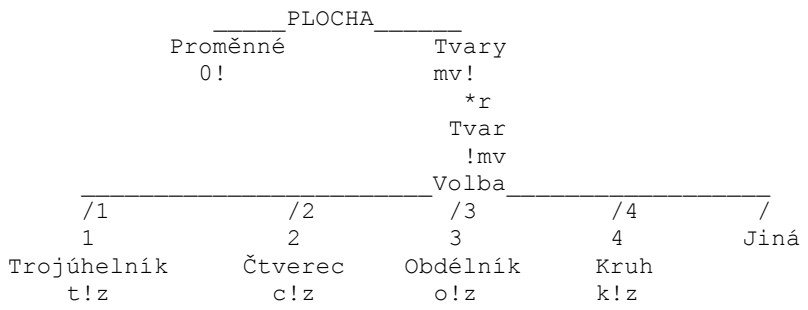
Téma: Vytvoření programu bez vzorových příkazů

Nyní, když jsme se naučili jak upravovat, rušit a vytvářet své vlastní operace a podmínky v seznamu operací a podmínek, který byl vytvořen podle vzoru, když jsme si ukázali, jak v našich vlastních operacích a podmínkách používat funkce, které jsou v knihovně Baltazara, ukážeme si nyní, jak vytvořit program, ve kterém budou jenom ty operace a podmínky, které sami napíšeme.

Uděláme to jednoduše tak, že po vytvoření nového souboru stiskneme Ctrl+G (globální definice) a po Enter jsme v Teditu. Pak už jenom stačí smazat všechny definice globálních operací a již se nám při vkládání příkazů ve struktogramu nebudou žádné globální operace nabízet.

Jako ukázku uvedeme program, na kterém si současně předvedeme programování v textovém režimu obrazovky. Program PLOCHA bude počítat plochu jednoduchých geometrických útvarů, jako jsou trojúhelník, čtverec, obdélník a kruh. Program musí opakovaně řešit různé úlohy podle příslušné volby. V nabídce možností je pamatováno také na ukončení programu, tj. program je ukončen, jakmile je zadán tvar 0.

Struktogram programu PLOCHA:



Seznam operací struktury PLOCHA:

```
-o1-PLOCHA
0 ; Deklaruj proměnné
    int      volba;    /* Volba tvaru      */
    double   a, b,     /* Parametry tvaru  */
             P;        /* Plocha           */
c ; Zadej délku strany a vypočítej obsah čtverce
    TxtPišŘetězec("Zadej délku strany čtverce: ");
    TxtČtiReálnéČíslo(a);
    P = a * a;
k ; Zadej poloměr a vypočítej obsah kruhu
    TxtPišŘetězec("Zadej poloměr kruhu: ");
    TxtČtiReálnéČíslo(a);
    P = 2 * pi * a;
```

```
m ; Zobraz nabídku
    TxtPišŘetězec("\nZadej číslo úlohy (1-trojúhelník, 2-čtverec,"
    " 3-obdélník, 4-kruh, 0-konec): ");
o ; Zadej délky stran a vypočítej obsah obdélníku
    TxtPišŘetězec("Zadej stranu a obdélníku: ");
    TxtČtiReálnéČíslo(a);
    TxtPišReálnéŘetězec("Zadej stranu b obdélníku: ");
    TxtČtiReálnéČíslo(b);
    P = a * b;
t ; Zadej délku strany, výšku a vypočítej obsah trojúhelníku
    TxtPišŘetězec("Zadej délku strany trojúhelníku: ");
    TxtČtiReálnéČíslo(a);
    TxtPišŘetězec("Zadej výšku trojúhelníku       : ");
    TxtČtiRČíslo(b);
    P = a * b / 2;
v ; Čti volbu z klávesnice
    TxtČtiČíslo(volba);
z ; Zobraz výsledek
    TxtPišŘetězec("Plocha je: ");
    TxtPišReálnéČíslo(P, -1, 2);
    TxtPišŘetězec("\n-----------------------------------------")
```

Seznam Podmínek struktury PLOCHA:

```
-c1-PLOCHA
/1 ; Volba Je 1
     volba == 1
/2 ; Volba Je 2
     volba == 2
/3 ; Volba Je 3
     volba == 3
/4 ; Volba Je 4
     volba == 4
*r ; Volba Není 0
     volba       /* v jazce C je to totéž jako volba != 0 */
```

Znovu připomínáme, že operace editujeme v seznamu operací (-o1-název_struktury), podmínky v seznamu podmínek (-c1-název_struktury) pro danou funkci.

SGP Baltazar – příprava č. 13

Téma: Vytvoření nové funkce

V případě, že vytváříme složitější program, je mnohdy výhodné rozdělit program do více samostatných funkcí. Této možnosti využíváme obzvláště, je-li třeba provést určitou činnost na různých místech programu. V systému SGP Baltazar můžeme vytvářet nové funkce tak, že nové funkci je automaticky přiřazen příslušný struktogram.

Vytvoření nové funkce

Novou funkci vytváříme zásadně pouze v seznamu struktur (funkcí), který se zobrazí po stisknutí klávesy Esc z editoru struktur. Po otevření okna seznamu struktur je kurzor nastaven na naposledy zpracovávané funkci. Nová funkce se vkládá automaticky před funkci, na níž stojí kurzor. Při umístění nové funkce dbáme na to, aby nová funkce byla vždy umístěna před funkcí, ve které je nová funkce volána.

Novou funkci vytvoříme stisknutím klávesy F4 a zadáním jména nové funkce. V případě, že funkce nemá mít žádný parametr, a ani nemá vracet žádnou hodnotu, postačí uvést jenom její jméno. Pokud má funkce vracet nějakou hodnotu, nebo jí chceme předat nějaký parametr, přepneme se do textového editoru (Ctrl+Enter) a upravíme hlavičku funkce. Jméno funkce musí obsahovat pouze povolené znaky pro jména v jazyce C, tj. nesmí začínat číslicí a může obsahovat pouze znaky abecedy, číslice a znak podtržítko (_).

Například: `void Nuluj(void)` *je stejné, jako kdybyste po F4 zadali pouze* `Nuluj`

```
void PišRSouřadnice(double x, double y)
int PočetSekund(void)
double ObvodKruhu(double r)
```

Definice funkce

Vlastní definici nové funkce provádíme již známým způsobem v Seditu. V novém struktogramu je na počátku jen jeden prvek, kterým je název struktogramu nové funkce.

Volání funkce

Aby nová funkce v programu něco vykonávala, musí být v některé jiné části programu volána. To znamená, že v některém jiném struktogramu musí být uvedena operace, která funkci volá stejným způsobem, jako by se jednalo o funkci z knihovny Baltazara.

```
_____________Program_____________
   Začátek          Prostředek              Konec
0[vR:(5)!         :MojeFunkce()               !w]
```

Když začneme ve svých programech vytvářet nové funkce, brzy zjistíme, že seznamy operací a podmínek, které jsou k novým funkcím kopírovány při použití vzoru, nám zbytečně znepřehledňují orientaci v často jednoduchých funkcích, ve kterých můžeme potřebovat jen několik málo vlastních operací a podmínek. V tomto případě je výhodnější globální definice maker smazat. Ačkoli musíme vytvořit úplně celý struktogram (ani volba podle VZORBC.000 nevytváří žádnou strukturu pro novou funkci) a definovat úplně všechny operace, zajistíme si tím, že v seznamech operací a podmínek jsou opravdu uvedeny jenom ty, které jsme si vědomě pro danou funkci vytvořili. Znovu připomínáme,

že operace editujeme v seznamu operací dané funkce (-o1-název_funkce) a podmínky v seznamu podmínek dané funkce (-c1-název_funkce).

Žáci si vyzkouší vytvářet nové funkce bez parametrů a bez vracené hodnoty.

Nepotřebnou funkci smažeme ze seznamu struktur klávesou F8.

Poznámky

SGP Baltazar – příprava č. 14

Téma: Předávání parametrů funkcím a vracení hodnoty

Při volání funkce je mnohdy potřebné, aby volaná funkce měla přístup k některým datům z funkce volající. Navíc někdy je třeba, aby volaná funkce předala volající funkci nějakou informaci po zpracování.

Předávání parametrů

Předávání parametru si vyzkoušíme na jednoduchém programu.

Předávání parametrů hodnotou

Struktogram funkce `int MojeFunkce1(int a)` (tj. funkce, která má jeden parametr typu číslo `int` a také vrací číslo `int`):

```
    _____int_MojeFunkce1(int_a)_____
Zpracování                      Návrat
 vov!                            n!
```

Seznam operací funkce `int MojeFunkce1(int a)`:

```
o ; Přiřad proměnné a hodnotu 5
    a = 5;
v ; Vypiš na obrazovku funkci a hodnotu promenné a
    TxtPišŘetězec("Moje funkce : a = "); TxtPišČíslo(a, -1);
    TxtPišNR();
n ; Navrať se do volající funkce a předej hodnotu 3
    return 3;
```

Struktogram programu (hlavní funkce `main()`):

```
   ___________________Program___________________
Začátek           Prostředek              Konec
  0!               vfv!                    c!
```

Seznam operací pro program (hlavní funkce `main()`):

```
0 ; Deklaruj a nastav proměnnou
    int      a = 1,
             b = 2;
c ; Čekej na stisknutí klávesy
    TxtPišŘetězec("Stiskni nějakou klávesu ...\n");
    ČekejNaKlávesu();
f ; Volej novou funkci
    b = MojeFunkce(a);
v ; Vypiš na obrazovku program a hodnotu proměnné a
    TxtPišŘetězec("Program: a = "); TxtPišČíslo(a, -1);
    TxtPišŘetězec(", b = "); TxtPišČíslo(b, -1);
    TxtPišNR();
```

Program spustíme a na obrazovce se objeví následující výpis:

```
Program: a = 1, b = 2
Funkce : a = 1
Funkce : a = 5
Program: a = 1, b = 3
Stiskni nějakou klávesu ...
```

Z výše uvedeného výpisu vidíme, že hodnota `a` (je 1) byla předána funkci `MojeFunkce1`, která provedla příkaz `a = 5;`, ale po návratu z funkce `MojeFunkce1` zůstala hodnota proměnné `a` nezměněna. Program i funkce mají definovanou proměnnou `a`, ale jedná se o dvě různé proměnné. Funkci `MojeFunkce1` byla předána hodnota `a` (je 1) z programu a tuto hodnotu si uchovala ve své proměnné `a`. Funkce `MojeFunkce1` si

vytvořila jenom kopii proměnné, která jí byla předána. Po návratu z funkce její proměnná `a` (je 5) zanikla.

Předávání parametrů odkazem

Vytvoříme nový program, který se podobá předcházejícímu. Program napíšeme znovu, nepokoušejte se ho upravovat. Úprava vyžaduje znalost struktury souboru .SGP, která bude probírána později.

Struktogram funkce `int MojeFunkce2(int *a)` (tj. funkce, která vyžaduje jeden parametr typu ukazatel na číslo `int` a vrací číslo `int`):

```
    _____int_MojeFunkce2(int_*a)_____
Zpracování                        Návrat
 vov!                              n!
```

Seznam operací funkce `int MojeFunkce2(int *a)`

```
o ; Zapiš na adresu a hodnotu 5
    *a = 5;
v ; Vypiš na obrazovku funkci a hodnotu proměnné a
    TxtPišŘetězec("Funkce : hodnota na adrese a je ");
    TxtPišČíslo(*a, -1);
    TxtPišNR();
n ; Návrat do volající funkce
    return 3;
```

Struktogram programu (hlavní funkce `main()`):

```
  _________________Program_________________
Začátek          Prostředek             Konec
  0!              vfv!                   c!
```

Seznam operací programu (hlavní funkce `main()`):

```
0 ; Deklaruj a nastav proměnnou
    int      a = 1,
             b = 2;
c ; Čekej na stisknutí klávesy
    TxtPišŘetězec("Stiskni nějakou klávesu ...\n");
    ČekejNaKlávesu();
f ; Volej novou funkci
    b = MojeFunkce2(&a);
v ; Vypiš na obrazovku program a hodnotu proměnné a
    TxtPišŘetězec("Program: a = "); TxtPišČíslo(a, -1);
    TxtPišŘetězec(", b = "); TxtPišČíslo(b, -1);
    TxtPišNR();
```

Program spustíme a na obrazovce se objeví následující výpis:

```
Program: a = 1, b = 2
Funkce : a = 1
Funkce : a = 5
Program: a = 5, b = 3
Stiskni nějakou klávesu ...
```

SGP Baltazar – příprava č. 14 – dokončení

Z výše uvedeného výpisu vidíme, že volaná funkce MojeFunkce2 změnila původní hodnotu proměnné `a` (je 1) v části Program na hodnotu 5. Funkci byla jako parametr předána adresa, na které je uložena hodnota `a` (je 1). Že se jedná o adresu a ne o hodnotu, je dáno znakem „&“ ve volání funkce. Skutečnost, že parametr funkce je adresa (ukazatel), musí být uvedena v deklaraci parametrů funkce pomocí znaku „*“, např. `int *a;`. Ve funkci, které byla jako parametr předána adresa (ukazatel), jméno proměnné představuje adresu, na které je daná hodnota uložena. Máme-li pracovat s hodnotou uloženou na adrese a ne s adresou, musíme použít znak (operátor) „*“, např. `*a = 5;`.

Deklaraci proměnných, ukazatelů, ukazujících na proměnné daného typu a práci s nimi si ukážeme v následující ukázce:

```
int n;       /* deklaruj proměnnou n typu int */
int *s;      /* deklaruj ukazatel na proměnnou typu int */
n = 0;       /* n přiraď hodnotu 0 */
s = &n;      /* nastav ukazatel s na adresu, kde je uložena
                proměnná n */
*s = 5;      /* na adresu, kam ukazuje ukazatel s, zapiš
                hodnotu 5, tj. n = 5; */
TxtPišČíslo(n); /* vypiš hodnotu n na obrazovku
                   (n je nyní 5) */
```

Poznámky

SGP Baltazar – příprava č. 15

Téma: Soubor .SGP

Ačkoli práce s celým souborem .SGP by měla být spíše výjimečnou, může se stát, že v některých případech se k ní můžeme uchýlit. Takovými případy mohou být například deklarace globálních proměnných, přejmenování funkce nebo vytvoření nové funkce zkopírováním již existující.

Nejprve si popíšeme soubor .SGP, který byl vytvořen v systému SGP Baltazar založením nového souboru s použitím vzoru VZORBC.PRG. Ve struktogramu (Seditu) nebyly provedeny žádné úkony, nebyly definovány žádné operace či podmínky v Teditu. Soubor .SGP je pro daný program přístupný po stisknutí kláves Ctrl+U v hlavní nabídce.

```
SGPC   3.3+ S#3+0000 12.08.1996 14:08:42
; Vytvořen ze vzoru: C:\SGPBC\VZORBC.PRG

# include <baltazar.h>

;-h-----------------------------------------------
/***** globální proměnné *****/

;-h-----------------------------------------------
/***** hlavní program *****/
void main( void )
-s--PROGRAM
  ___________________PROGRAM_____________________
Začátek             Prostředek                 Konec
-o1-PROGRAM  operace
-c1-PROGRAM  podmínky
=SP=PROGRAM
/***** konec programu *****/
```

Na prvním řádku jsou uvedeny následující údaje preprocesoru SGP:

- typ preprocesoru (SGP pro programovací jazyk C. Pro profesionální použití existují preprocesory pro všechny nejběžnější programovací jazyky)
- verze preprocesoru
- číslo licence programu
- datum vytvoření souboru .SGP
- čas vytvoření souboru .SGP

Pro správnou funkci preprocesoru jsou nutné jen první dva údaje.

Na čtvrtém řádku je příkaz (direktiva `#include`) pro kompilátor jazyka C, který zajistí, že v daném místě se zpracuje soubor, uvedený v lomených závorkách (soubor BALTAZAR.H). Tento soubor se nachází v adresáři Baltazara a obsahuje deklarace funkcí, definice konstant a maker. Příkaz pro kompilátor může být využit pro začlenění dalšího souboru (knihovny vlastních funkcí) do programu. V tomto případě však název souboru by měl být uveden v uvozovkách, případně včetně adresáře.

Příklad: `#include "\\knihovna\\k1.c"`

Znak "\" zpětné lomítko je v jazyce C tzv. escape sekvence (přeřaďovací znak), takže má-li být obsažen v řetězci, musí se zdvojit.

V případě, že v programu chceme používat globální proměnné, můžeme je deklarovat na dalších řádcích vložených za řádkem s poznámkou „globální proměnné“ (v jazyce C obecně mimo funkci před jejím použitím).

Před každou definicí funkce je oddělovací čára začínající znaky „;-h---“. Tato čára slouží pro snadnější orientaci v souboru. Pro zpracování souboru není nutná.

Nachází-li se v souboru .SGP na první pozici řádku středník, není tento řádek preprocesorem zpracováván – je považován za poznámku (tzv. SGP poznámka).

Za oddělovací čarou následuje definice funkce v jazyce C. V případě více funkcí jsou nejprve uvedeny volané funkce tak, aby v okamžiku volání funkce byla tato již definována. Jako poslední funkce je uvedena hlavní funkce, která má jméno `main`.

Každá funkce začíná hlavičkou funkce, která sestává z deklarace vracené hodnoty, názvu funkce a deklarací parametrů (které oddělujeme čárkou) uzavřených v kulatých závorkách.

Za hlavičkou funkce následuje v jazyce C tělo funkce, které je vymezeno znaky „{“ a „}“. Tyto znaky za nás vygeneruje SGP; nemusíme je tedy psát.

Tělo funkce sestává z následujících částí zpracovávaných preprocesorem:

-s--Název_struktury
je část obsahující struktogram

-o1-Název_struktury
je část uvádějící seznam operací včetně jejich definic
o1 značí, že operace budou označovány jedním ASCII znakem, tj. asi 200 různých znaků
o2 značí, že operace budou označovány dvěma ASCII znaky, tj. asi 40.000 kombinací
Upozornění: je-li použito o1 - musí být <u>všechny</u> operace jednoznakové
je-li použito o2 - musí být <u>všechny</u> operace dvouznakové

-c1-Název_struktury
je část uvádějící seznam podmínek včetně jejich definic

Úsek zpracovávaný preprocesorem je ukončen řádkem:
=SP=Název_struktury

Za definicí hlavní funkce main je už jenom poznámka, která označuje konec programu.

Jako další si vytvoříme program, který bude na obrazovku zobrazovat znaky stisknutých kláves tak, že malá písmena převede na velká. Pro převádění malých znaků na velké bude použito samostatné funkce. Po naprogramování provedeme rozbor příslušného souboru .SGP.

SGP Baltazar – příprava č. 15 – dokončení

```
SGPC   3.00 S#3D0003 30.09.1994  9:14:03
# include <baltazar.h>

;-h-----------------------------------------------------
int Velké(int c)
-s--int Velké(int c)
       _int_Velké(int_c)_
  __Písmeno?_          Návrat
  /m        /           n!
 Malé      Jiné
 p!
-o1-int Velké(int c)
n ; Vrať hodnotu ASCII písmena
    return c;
p ; Převeď malé písmeno na velké
    c = c - ('a'-'A');
-c1-int Velké(int c)
/m ; Je to malé písmeno
     c >= 'a' ASoučasně c <= 'z'
=SP-int Velké(int c)

;-h-----------------------------------------------------
void main( void )
{
-s--Program
     _____Program_____
Deklarace        Písmena
   0!              c!
                    *k
                  Písmeno
                     !c
               _Zpracování_
             Převod      Tisk
              v!         t!
-o1-Program
0 ; Deklaruj proměnnou
    int      znak;
c ; Čti znak z klávesnice
    znak = ČtiKlávesuZFrontySČekáním();
v ; Převeď na velké písmeno
    znak = Velké(znak);
t ; Vytiskni písmeno na obrazovku
    TxtPišZnak(znak);
-c1-Program
*k ; Není konec
     znak Není KlEnter
=SP=Program
}
/***** konec programu *****/
```

Práci v celém souboru .SGP pomocí textového editoru (včetně Teditu) nedoporučujeme! Úkony uvedené na začátku je možné provádět spuštěním Teditu z Seditu. Při spuštění Seditu jsou ze souboru .SGP vytvořeny dva pracovní soubory:

- .SG1 – obsahuje struktogramy, které editujeme pomocí Seditu
- .SG2 – obsahuje zbývající části, které editujeme pomocí Teditu spuštěného z Seditu

Při spuštění preprocesoru, nebo po opuštění Seditu, jsou oba soubory opět sloučeny v jeden soubor .SGP.

Poznámka: Pro editaci struktogramu slouží výhradně příkaz Sedit, který automaticky kontroluje a zajišťuje správnost struktogramu. Poškození struktogramu textovým editorem, ke kterému může dojít při práci se souborem .SGP, může mít za následek vážné poruchy zaviněné nejpravděpodobněji vznikem izolovaných prvků, nebo jejich špatným přiřazením k nadřízeným prvkům.

Domácí úkol: vymyslet téma jednoduchého programu.

Poznámky

SGP Baltazar – příprava č. 16

Téma: Další možnosti editoru struktur – Seditu, samostatná práce, soutěž

Další možnosti editoru struktur

Doposud jsme vkládali vykřičník stejně jako znaky pro označení podmínek tak, že nejprve jsme stiskli klávesu X, pro vkládání nového prvku pod prvek, na kterém se nachází kurzor, a potom jsme napsali příslušný znak. V případě podmínek Sedit očekával jméno podmínky, v případě vykřičníku se operace musely vkládat po dalším stisknutí klávesy nalevo, pod nebo napravo od vykřičníku.

Výše uvedené operace můžeme podstatně zkrátit pomocí zrychleného vkládání. Když stiskneme klávesu vykřičník, nebo znak příslušné podmínky, Sedit vloží příslušný znak automaticky pod prvek, na kterém spočíval kurzor. Navíc v případě vykřičníku Sedit očekává vkládání operací, v případě podmínek očekává vložení názvu programové jednotky, která má být zpracovávána v případě splnění dané podmínky.

Samostatná práce

S jednotlivými žáky, nebo s malými skupinkami projednáme téma programu, který hodlají vytvořit. V případě, že by neměli připraveno přiměřené téma, zadáme nějaké podle vlastní úvahy, nebo podle sbírky příkladů.

Žáci mohou získat podrobnější informace k jednotlivým funkcím z knihovny Baltazar, které hodlají použít ve svém programu pomocí zabudované nápovědy Příkazového pomocníka SGP, do kterého se dostanou stiskem klávesy F3 (případně Shift+Mezerník nebo Ctrl+Mezerník). Tento Příkazový pomocník je dostupný jak z Uditu, tak z Seditu.

Jednotlivé žáky či skupinky průběžně kontrolujeme. Dbáme na to, aby struktogramy tvořili přehledně, aby výstižně pojmenovávali prvky struktogramu a využívali probrané možnosti, zejména možnosti rozdělení programu do více struktur (funkcí).

Především dbáme na to, aby ve struktogramu všechny černé názvy byly podstanými jmény, a aby algoritmus programu byl zřejmý pouze ze struktogramu, tj. bez nutnosti zobrazení jednotlivých operací!

Program, který není zřejmý ze struktogramu (tj. ze schématu) je chybný! I kdyby pracoval správně.

Správný program není takový, který občas udělá to, co má udělat, ale takový, který to udělá vždy, rychle, ale hlavně - JE DOBŘE ČITELNÝ!

Zadání soutěžních témat

Žáci si vyberou jedno z následujících témat:
1. Příběh
2. Hra
3. Výukový program
4. Grafická animace
5. Užitkový program – v praxi použitelný program

S jednotlivými žáky nebo malými skupinkami projednáme téma, na které chtějí vytvořit samostatnou práci.

Práce na soutěžních tématech

Jednotlivé žáky či skupinky průběžně kontrolujeme. Dbáme na to, aby struktogramy tvořili přehledně, aby výstižně pojmenovávali prvky struktogramu a využívali probrané možnosti, zejména možnosti rozdělení programu do více struktur (funkcí).

Vyhodnocení soutěžních prací

Žáci sami provedou vyhodnocení jednotlivých prací a vyberou nejlepší práce.

Učitel zašle disketu s nejlepšími pracemi firmě SGP Systems spolu se souhlasem o volném šíření těchto prací. Na disketě musí být zřetelně napsána zpáteční adresa, autor programu, rodné číslo autora, bydliště autora.

Firma SGP Systems, s.r.o. vybrané práce umístí na své internetovské WWW stránce, případně zveřejní jiným způsobem. Žáci tak budou mít motivaci k tvůrčí práci se systémem SGP Baltazar.

Struktura dat na disketě:

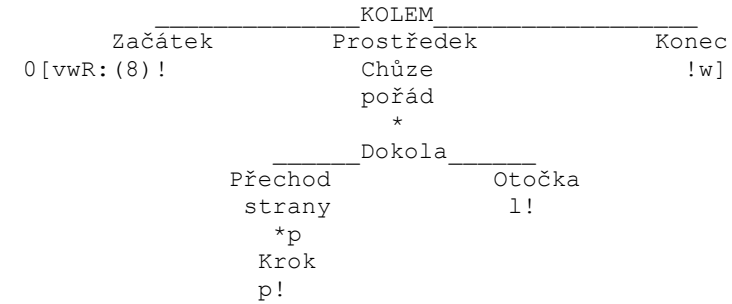
```
\Město-škola
 |-\1-PRIBEH
 |  |-\PRIBEH1
 |  | |-PRIBEH1.SGP
 |  | |-PRIBEH1.TXT - popis programu
 |  | |-PRIBEH1.B00
 |  | |-...........
 |  |-\PRIBEH2
 |.................
 |-\2-HRA
 |-\3-VYUKA
 |-\4-GRAFIKA
 |-\5-UZITEK
```

Střední průmyslová škola, Uherské Hradiště
Řešené příklady
SGP BALTAZAR

Při řešení následující úkolů byl použit při tvorbě struktogramu VZORBC.000.

Úkol: Baltazar má za úkol neustále obcházet kolem okraje obrazovky.

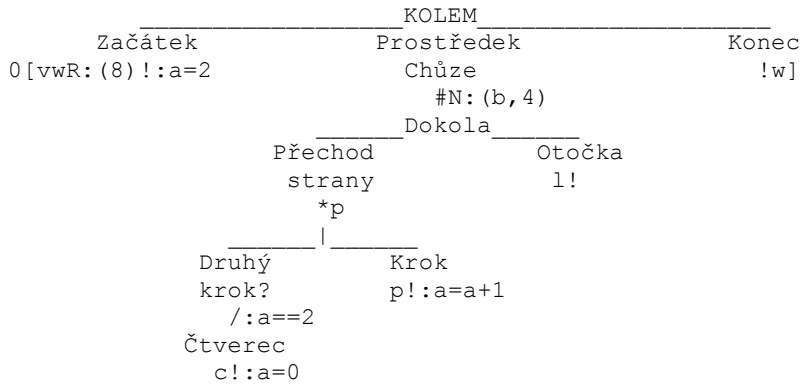
Řešení:



V nekonečném cyklu (* bez podmínky) Baltazar přejde stranu a udělá otočku vlevo. Stranu přechází krokem, dokud je před ním průchodný předmět.

Úkol: Baltazar má za úkol jednou obejít kolem okraje obrazovky a při každém druhém kroku vyčarovat čtverec.

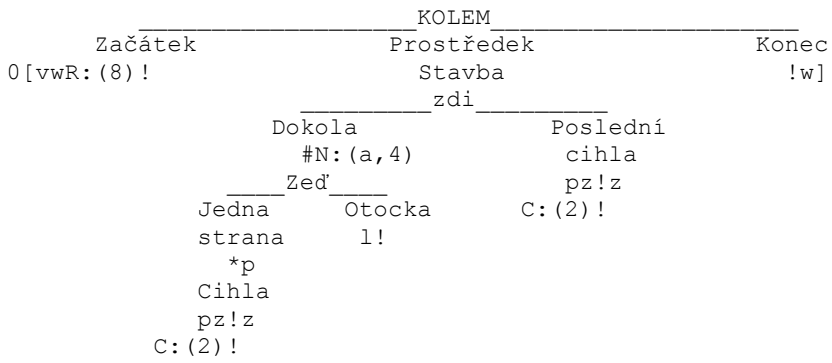
Řešení:



Obrazovku obcházíme obdobně jako v předcházejícím úkolu, jen počet opakování cyklu je pevně dán (čtyřikrát). V proměnné a si uchováváme informaci, kolikátý krok děláme, při druhém kroku vyčarujeme čtverec.

Úkol: Baltazar má kolem okraje obrazovky postavit zeď z neprůchodných cihel.

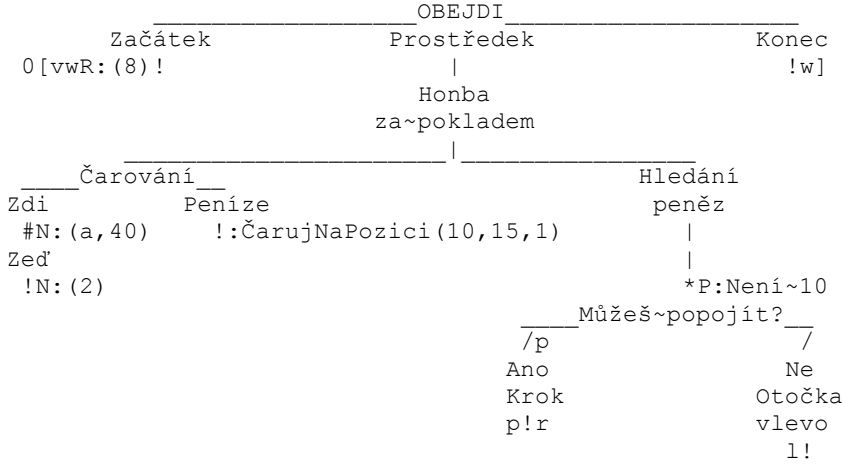
Řešení:



Stavbu zdi provádíme opět v čtyřikrát se opakujícím cyklu, jednu stranu stavíme tak, že Baltazar popojde, otočí se čelem vzad, vyčaruje cihlu a otočí se nazpět čelem vzad. Nakonec ale musí ještě "zasadit" obdobným způsobem poslední cihlu do zdi.

Úkol: Na začátku vyčarujte náhodně 40 políček s neprůchodnou zdí a v pravém horním rohu obrazovky pytlík s penězi. Úkolem Baltazara je přejít z dolního levého rohu, kde se objeví na začátku do pravého horního rohu a zastavit se před pytlíkem s penězi.

Řešení:



Nejdříve vyčarujeme v cyklu s pevným počtem opakování pomocí operace N čtyřicet náhodných zdí, pak pytlík s penězi na pozici 15,1. Hledat budeme v cyklu, který bude ukončen, když předmět před Baltazarem bude 10 = pytlík s penězi. A jak budeme chodit? Vždy jen o krok kupředu a pak se hned otočíme doprava. Budeme se totiž snažit obejít bludiště zdí při pravém okraji. Když je ale předmět před námi neprůchodný, otočíme se vlevo a pokračujeme dál. Baltazar sebou při pohybu bude "cukat", protože si neustále před sebou ohmatává cestu.
Úkol: Postavte pyramidu z bílých čtverců, základna má velikost 13 čtverců a další řada je vždy o dva čtverce menší (celkem 7 řad).

Řešení 1:

```
          _________________PYRAMIDA_________________
      Začátek          Prostředek              Konec
0[vwR:(8)!               Stavba                  !w]
                        pyramidy
                        Sedm~desek
                            #N:(a,7)
                   ______Jedna~deska_______
              Položení                  Přesun
               kostek                   nazpět
                 #N:(b,13-a*2)           z!rprp
               Kostka                     #N:(b,13-a*2)
               cp!                       Krok
                                         p!
```

Pyramidu můžu stavět tak, že postavím vždycky jednu řadu zleva doprava, otočím se, přejdu nazpět nalevo a další řadu stavím opět zleva doprava. Vlastní položení kostek v řadě provádím v cyklu s pevným počtem opakování, který se ale zmenšuje se zvyšujícím se patrem.

Řešení 2:

```
          _______________PYRAMIDA_______________
      Začátek         Prostředek            Konec
0[vwR:(8)!              Stavba                !w]
                       pyramidy
                          #N:(a,4)
      ______Stavba~dvou~desek~zároveň_______
 První~deska     Otočka     Druhá~deska     Otočka
   Pokládání    lpl!         Pokládání     rpr!
    kostek                    kostek
      #N:(b,13-a*4)             #N:(b,11-a*4)
    Kostka                    Kostka
    cp!                       cp!
```

Stavbu můžu provádět i tak, že stavím i při cestě zprava doleva, pouze otočky na pravé a levé straně jsou různé (a také počet opakování při pokládání kostek).

Řešení 3:

```
          ___________PYRAMIDA_________
      Začátek      Prostředek       Konec
0[vwR:(8)!         :c=1!              !w]
                     Stavba
                  sedmi~desek
                      #N:(a,7)
          ________Jedna~deska______
        Položení                  Otočka
         kostek                 ____kam?____
           #N:(b,13-a*2)       /:c==1     /
         Kostka              Vlevo       Vpravo
         cp!                lpl!:c=0    rpr!:c=1
```

V tomto řešení provádím stavbu opět při cestě zleva doprava i zprava doleva, pouze se musím správně rozhodnout, na kterou stranu se budu otáčet, až položím kostky v příslušné řadě. Kam se otáčím říká proměnná c (c je 0 - vpravo, c je 1 - vlevo).

Úkol: Postavte stejnou pyramidu jako v předcházejícím případě, ale ne z průchodných čtverců, nýbrž z neprůchodné zdi.

Řešení 1:

```
             ____________PYRAMIDA____________
        Začátek       Prostředek          Konec
   0[vwR:(8)!           Stavba              !w]
                       pyramidy
                       z~cihel
                          #N:(a,4)
        ______Stavba~dvou~desek~zároveň________
   První~deska       Otočka   Druhá~deska     Otočka
    lprp!         lplpp!        Pokládání     rrp!
    Pokládání                     cihly
      cihly                         #N:(b,11-a*4)
        #N:(b,13-a*4)               Cihla
      Cihla                   lC:(2)!rp
  rC:(2)!lp
```

Způsob stavby si mohu zvolit kterýkoliv z předcházejících řešení, musím však jiným způsobem pokládat cihly. Musím být o řadu výš a provést otočku k nižší řadě, položit cihlu, otočit se zpět a popojít.

Řešení 2:

```
          ___________PYRAMIDA___________
      Začátek     Prostředek          Konec
0[vwR:(8)!        :c=1!                !w]
                    Stavba
                 sedmi~desek
                   z~cihel
                      #N:(a,7)
            ______Stavba~desky_______
         Položení                  Otočka
           p!                    ____kam?____
          cihel                 /:c==1     /
            #N:(b,13-a*2)     Vlevo      Vpravo
          Cihla             lplp!:c=0  rprp!:c=1
     pzC:(2)!z
```

A nebo stavět jakoby za sebou, tj. popojít, otočit se čelem vzad, položit cihlu, otočit se nazpět ... (Toto je možné opět realizovat kterýmkoli ze tří dříve uvedených způsobů).

Úkol: Vyplnit celou obrazovku šachovnicí.

Řešení:

```
          ____________ŠACHOVNICE____________
      Začátek         Prostředek          Konec
0[vwR:(8)!            :b=0!                !w]
                      Šachovnice
                po~jednotlivých~řadách
                         #N:(a,10)
                  ___________|__________
              Řada                ____Otočka_____
               *p                /:b==0       /
            Políčko           Vlevo          Vpravo
             cp!            lcplp!:b=1         !:b=0
             Můžeš?                           r!r
               /p                            Můžeš?
```

```
Krok                          /p
p!                          Krok
                            p!
```

Obrazovku budeme zaplňovat po jednotlivých řádcích, tj. 10 řádků. Na řádku děláme vždy čtverec a poté dva kroky, dokud je před námi průchodný předmět. Pozor, při druhém kroku se musíme zeptat, jestli tento můžeme udělat, jestli je před námi pořád průchodný předmět. Po vyplnění řady se musíme otočit a přejít do další řady. Rozlišujeme, jestli přecházíme zleva doprava nebo naopak.

Úkol: Na spodním okraji obrazovky bude přejíždět autíčko sem a tam, zleva doprava a zprava doleva, pořád dokola.

Řešení 1:

```
          _____________________AUTO_____________________
      Začátek              Prostředek               Konec
0[vwR:(9)!neB:(0)              |                      !w]
                             Auto
                            jezdí
                              *
                        _____|_____
                      Sem           Otočka
                      tam            z!
                       *p
                      Auto
                C:(22)!
                      p!zcz
```

Baltazar v nekonečném cyklu vždycky vyčaruje auto, popojde, otočí se a přemaže černým čtvercem auto na předcházející pozici. To pořád dokola, dokud nenarazí na okraj obrazovky. V jednom okamžiku jsou na obrazovce dvě auta, takže to potom vypadá jako by auto mělo ducha. Na začátku Baltazara zneviditelníme a budeme čarovat bez obláčku, barvu čtverce nastavíme na černou.

Řešení 2:

```
          _______________AUTO_____________
      Začátek          Prostředek       Konec
0[vwR:(9)!neB:(0)          |              !w]
                         Jízda
                           *
                     ______|______
                   Sem          Otočka
```

```
                    tam          z!
                     *p
                    Auto
                C:(22)!cp
```

Řešení je obdobné jako v předcházejícím případě, ale Baltazar vyčarované auto hned také vymaže. Při velkých rychlostech to ani nepostřehneme, při malých auto problikává.

Řešení 3:

```
          ____________________AUTO2_________________
      Začátek              Prostředek           Konec
0[vwR:(5)!nelpB:(0)            |                  !w]
                             Auto
                            jezdí
                              *
                     _____________|_____________
                   Tam                       Zpět
                    #N:(a,15)                 #M:(a,15)
                  "Auto                     'Auto
     X:(22,a+1,10)!X:(1,a+1,10)
```

V tomto řešení nenecháváme čarovat Baltazara (a proto ho na začátku "uklidíme" do druhé řady), nýbrž čarujeme auto přímo na příslušné pozice.

Poznámky

Vaše připomínky a náměty nám laskavě zasílejte na adresu:

SGP Systems, s.r.o.
Masarykovo nám 21
696 01 Uherské Hradiště
Česká republika

telefon : 0632/40592
tel./zázn./fax: 0632/551089
e-mail : sgp@brn.pvtnet.cz
www : http://www.pvtnet.cz/www/sgp