SGP SYSTEMS

SGP BALTAZAR 5.0

PŘÍRUČKA PROGRAMU

FORMOU PŘÍPRAV

PRO UČITELE A RODIČE

+

SBÍRKA PŘÍKLADŮ

Témata jednotlivých příprav

SGP Baltazar – Základní pojmy Soukupova programování

Téma: Základní pojmy, které by si měl uvědomit a znát KAŽDÝ člověk.

1. **Každý člověk programuje, ale málokdo se to učil. Učme proto děti programovat od mala!**
2. **Počítač** - je stroj, který umí vykonávat programy. Člověk také umí vykonávat programy.
3. **Program** - je **P**osloupnost **P**říkazů (**P=PP**).
4. **Příkazy** dělíme dle vykonavatele příkazu na :
   a) proveditelné
   b) neproveditelné.
5. **Příkazy** dělíme dle účinku pro zadavatele příkazu na:
   a) "+" kladné (pozitivní, konstruktivní)
   b) "-" záporné (negativní, destruktivní)
   c) "0" nulové (neutrální, bez účinku).
6. V našich programech budeme používat pouze příkazy **proveditelné** a **kladné**.
7. **Postup při řešení libovolné úlohy** (programu).
   a) Nalezneme všechna podstatná jména.
   b) Nalezneme všechna přídavná jména
   c) Nalezneme všechna slovesa.

   Z podstatných a přídavných jmen sestavíme strukturální diagram a slovesa umístíme k jednotlivým (příslušným) prvkům (objektům). Vše přesně tak, jak to vyplývá ze zadání. Ke každému podstatnému jménu musíme nalézt v zadání alespoň jedno sloveso.

   Pokud zjistíme, že zadání neobsahuje všechna potřebná podstatná jména nebo slovesa, požádáme o doplnění zadání.
8. **Podstatná jména** jsou podstatou všeho! Takže, není-li podstatné jméno = není problém = není co řešit! Proto se při komunikaci s druhými lidmi snažme používat správná podstatná jména.
9. **Fyzické objekty** = podstatná jména se skládají zase pouze a jenom z fyzických objektů, tj. z podstatných a přídavných jmen. Nikdy ne ze sloves (z činností)! Například: Silnice se neskládá z kroků (chůze), ale z částí silnice (dlaždičky, políčka apod.).
9. Dále by měl **každý žák po ukončení základní školní docházky** umět tyto práce na počítači:

a) první stupeň ZŠ
- udělat jednoduchý program
- nakreslit obrázek
- napsat dopis
- poslat dopis elektronickou poštou

b) druhý stupeň ZŠ
- udělat složitější program (s rozhodováním)
- zvládnout základní práci s tabulkovým kalkulátorem
- zvládnout základní práci s databází - rozeslat hromadný dopis
- vytvořit jednoduchou multimediální aplikaci

Děti, které zvládnou toto minimum, **nebudou mít v budoucnu nikdy problém** s pochopením a použitím výpočetní techniky v libovolné formě.

A na závěr jednu hezkou větičku z knížky pana Drbala:
"Správný kluk hraje fotbal, je zamilovaný do spolužačky a umí programovat."

SGP Baltazar – příprava č. 0

Téma: Popis programovacího a kreslicího systému SGP Baltazar

Programovací a kreslicí systém SGP Baltazar je určen především pro snadnou tvorbu programů na počítači a dětem pak pro výuku programování, kreslení a rozvoj logického myšlení.

Systém SGP Baltazar vznikl na jaře v roce 1993 spojením strukturovaného programovacího systému SGP, interpretu jazyka C a knihovny hotových funkcí jazyka Baltazar.

Jádrem systému SGP (Soukup Graphic Programming) je dnes jediný program, který zahrnuje tyto části (podsystémy):

Manažer souborů a projektů SGP	- SGP Manager
Editor struktur	- SGP Sedit
Editor textu	- SGP Tedit
Preprocesor do jazyka C	- SGP Preprocessor
Interpret jazyka C	- SGP Cinter
Knihovna funkcí jazyka Baltazar	- SGP Baltazar
Grafický editor předmětů	- SGP BaltEdit
Celostránkový grafický editor	- SGP Paint
Grafický editor ikon Windows	- SGP IconEdit
Editor baltazarovských scén	- SGP ScenEdit
Integrovaný systém nápovědy	- SGP Helper

SGP je programovací nástroj a Baltazar je programovací jazyk (na bázi C jazyka).

SGP Baltazar je tedy programovací nástroj pro jazyk Baltazar.

Samotný nástroj SGP byl vyvinut v roce 1983 a postupně byly vytvořeny preprocesory pro jazyky Basic, C, Clipper, Cobol, dBASE, Fortran, FoxBASE, Informix, Modula, Paradox a Pascal.

Od roku 1995 již firma SGP Systems podporuje pouze verzi SGP pro jazyk Baltazar a preprocesory pro ostatní jazyky již nemá v nabídce. V roce 1997 byl dokončen zcela nový originální programovací nástroj SGP Baltík, který je určen pro děti od 4 do 9 let.

Doporučená minimální sestava počítače je PC-386 kompatibilní s IBM, 560 kB volné operační paměti RAM, grafická karta VGA, pružný disk 3,5", operační systém MS-DOS 5.0 a vyšší.

Pokud používáte operační systém MS-DOS, je třeba mít nainstalovánu češtinu Latin-2 (kódová stránka 852).

Systém lze provozovat i pod operačními systémy Windows 3.11 a Windows 95, jako standardní dosová aplikace.

Dosud zjištěné potíže ve Windows 95

1. Po instalaci instalační verze programu se může stát, že při spuštění program hlásí ! PROGRAM ERROR! nebo ?LICENCE?. Proto je doporučeno, po instalaci instalační verze, znovu spustit Windows (restart Windows).
2. Po obnovení minimalizovaného okna SGP (při kreslení v SGP Paintu) se na obrazovce někdy objeví poškozený obrázek. K odstranění této závady Windows95 stačí stisknout v SGP Paintu ikonu překreslení obrazovky (druhý řádek v menu, osmá ikona zleva - obrázek obrazovky).

Žádné jiné potíže s Windows 95 zatím nebyly zjištěny.

Instalace pro MS-DOS

1. Zkontrolujte bezchybnost licenční diskety č.2 - viz návod na disketě (program TESTDISK).
2. Vložte do počítače disketu č. 1 a na klávesnici napište:

   `a:inst-sgp` a pak stiskněte Enter.

Spustí se instalační program, který vám ukáže kolik máte volného místa na disku v počítači a kolik místa zabere instalovaný balík programů. Programový balík se vybere stisknutím Enter. Pokud na disku není dostatek místa, instalační program vás na toto upozorní a instalace se nezahájí. Předtím, než potvrdíte výběr programového balíku k instalaci, můžete si také nastavit adresář, kam se má program nainstalovat. Doporučujeme však nechat přednastavený adresář.

Nejjednodušší tedy je, pokud nechcete nic měnit, stisknout třikrát po sobě tlačítko Enter. Program se začne instalovat. V průběhu instalace budete požádáni o vložení licenční diskety č.2.

Instalace pro Windows 95

1. Klepněte (jednou) pravým tlačítkem myši na ikonu "Tento počítač", vyberte "Prozkoumat". Spustí se Průzkumník.
2. V levé části Průzkumníka klepněte na disketu A:
3. V pravé části Průzkumníka poklepejte na program Inst-sgp a pak třikrát stiskněte Enter (myš nereaguje). Začne se instalovat systém SGP Baltazar. V průběhu instalace budete požádáni o vložení instalační diskety č. 2.
4. Po dokončení instalace zavřete okno DOSu "Dokončeno - INST-SGP" (klepněte myší vpravo nahoře na křížek okna DOSu, nebo stiskněte Alt+F4).
5. V Průzkumníkovi klepněte znovu na disketu A: (protože nyní je v počítači disketa č. 2).
6. Poklepejte na program Inst-grp. Klepněte na tlačítko "OK". Nainstaluje se složka SGP Baltazar s ikonami.

Program spustíte obvyklým způsobem buď v dosovském okně, z Windows přes tlačítko "Start" nebo v Průzkumníkovi, nebo si můžete vytvořit na pracovní ploše Windows95 zástupce složky SGP Baltazar.

Vytvoření zástupce složky SGP Baltazar na ploše ve Windows95:

Klepněte pravým tlačítkem myši na tlačítko "Start" (většinou v levém dolním rohu obrazovky).

Potom zvolte "Otevřít", poklepejte na Programy, pravým tlačítkem myši přetáhněte složku SGP Baltazar na pracovní plochu, zvolte "Vytvořit zde zástupce". Složka SGP Baltazar je vytvořena.

SERVIS

Budete-li mít nějaké potíže s našimi produkty, volejte:

technickou podporu SGP Systems:
tel. : 0632/551089, 554432
fax : 0632/551089
e-mail: sgp@sgp-systems.com

Připravte si tyto informace:
- sériové číslo produktu
- typ počítače a typ procesoru počítače
- typ grafické karty
- popis problému.

Na naší stránce http://www.sgp-systems.com, najdete poslední informace o produktu, našich aktivitách a stav soutěží.

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 nejprve vyzkoušíme v interaktivním režimu pomocí programu CARUJ.

Učitel spustí žákům program CARUJ.SGP v prostředí SGP Baltazara (viz příprava č. 3) nebo přímo příkazem v DOSu: **`sgpbalt caruj /r`** *(/r - spusť program, /? - všechny parametry)*.

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.

- L - vlevo vbok
- R - vpravo vbok
- P - pohyb vpřed (Baltazar popojde o 1 políčko)
- V - výběr předmětu pro příkaz čaruj - z palety všech předmětů
- C - vyčaruje předmět, který byl naposledy vybraný
- B - BaltEdit - jednoduchý grafický editor Baltazarových obrázků
- N - načti scénu
- U - ulož scénu

Zbývající část hodiny budou žáci samostatně pracovat s programem CARUJ, sestaví si svůj vlastní obrázek. Jako námět na kresbu obrázku může posloužit např. dům, ve kterém bydlí, či škola, do které chodí. Vzorovou scénu na obrázku získáte příkazem Načti a CARUJ.S00.

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.

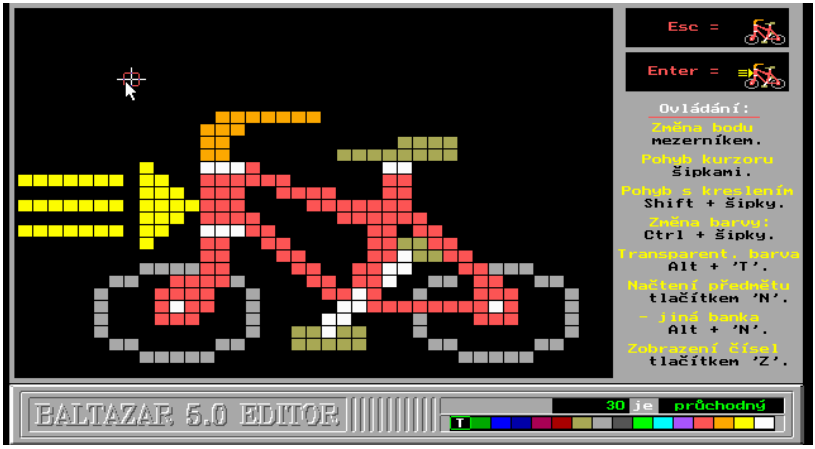
Zobrazí se paleta obrázků-předmětů, kterou již známe z programu CARUJ. Tato paleta obsahuje 150 předmětů. V části palety jsou již hotové předměty. Každý předmět má své číslo, kterým je při čarování volán. Číslo předmětu se zobrazuje vpravo dole. Můžeme také zapnout zobrazení čísel u všech předmětů (klávesou Z). U všech předmětů je možné nastavit průchodnost (klávesa P). Neprůchodné předměty musí Baltazar 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 upravovat, nejprve vybereme pomocí čtvercového kurzoru ovládaného kurzorovými klávesami, myší, nebo zadáním čísla zvoleného předmětu. Po vybrání předmětu stiskneme Enter nebo levé tlačítko myši, předmět se zvětší a můžeme jej upravovat. Jak z klávesnice, tak pomocí myši. Dále postupujeme již dle nabídky programu.

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 levým tlačítkem myši, nebo pomocí kombinace kláves Ctrl+→ nebo Ctrl+←. Bod dané barvy se udělá Mezerníkem nebo levým tlačítkem myši. Souvislou rovnou čáru uděláme kombinací kláves Shift+šipka.

Pro kreslení nového obrázku (po stisku Enter na prázdném políčku) můžeme také využít možnosti načtení již hotového obrázku. Stiskneme klávesu N a vybereme si předmět pro načtení do našeho políčka.



Chceme-li obrázek načíst z jiného souboru obrázků, stiskneme Alt+N a zadáme název souboru obrázků včetně adresáře (např.: C:\SGP\BALTAZAR\SGPBANK.B01).

Transparentní barva.

Od verze 5.0 lze v každém předmětu stanovit tzv. transparentní barvu, tj. barvu, která při položení předmětu bude průsvitná, to znamená, že bude vidět barva, která je pod ní. Transparentní barva je označena písmenem T v příslušném políčku barevné palety. Transparentní barva se nastavuje/ruší pravým tlačítkem myši na dané barvě v paletě barev nebo klávesami Alt+T na dané barvě v ploše předmětu.

Transparentní barva se používá v animovaných předmětech pro dosažení dokonalé animace. Například pták letící mezi mraky, autíčko jedoucí po silnici v krajině apod.

Ukončení BaltEditu.

Nejprve ukončíme úpravu předmětu buď z klávesnice nebo myší. Ukončení úpravy předmětu a uložení upravovaného 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. Totéž můžeme provést klepnutím levým tlačítkem myši na příslušném zmenšeném předmětu v pravém horním rohu obrazovky.

Celý BaltEdit ukončíme klávesou Esc nebo pravým tlačítkem myši.

V další části výuky se žáci seznámí s prací v BaltEditu na konkrétním příkladu, když obrazec č. 16 překopírují do předmětu 100 a 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

Při práci v síti by měl mít každý žák nakopírován do svého adresáře soubor SGPBANK.B00, aby si tak mohl tvořit své vlastní předměty. Spuštění programu například pro žáka 5:

c:\>**g:**

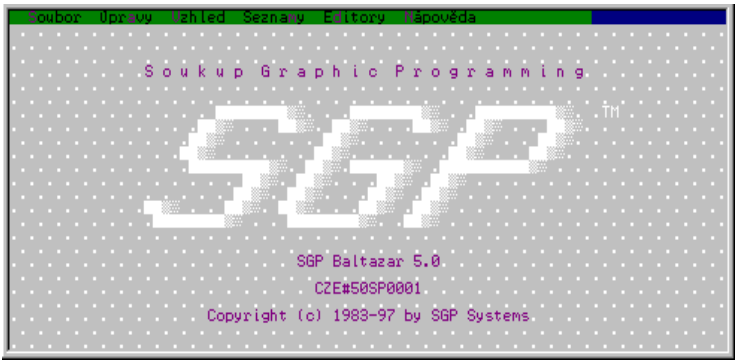
g:\>**cd \skupina\zak5**

Program SGP Baltazar spustíme:

g:\skupina\zak5**>sgpbalt**

nebo

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



Na monitoru se objeví úvodní obrazovka systému SGP s menu v prvním řádku.

Program čeká na naši volbu. Pokud se nejprve chceme pouze seznámit s jednotlivými možnostmi volby, stiskneme klávesu F10 a projdeme si jednotlivé položky menu.

Nejprve si ale vysvětlíme, co to je projekt SGP.

Projekt SGP je souhrn souborů, které mají stejné jméno jako soubor s příponou .SGP. Projekt je tedy tvořen vlastním programem, neboli zdrojovým souborem s příponou .SGP a dále pak všemi dalšími soubory, příslušnými k tomuto projektu (.CIN, .C, .EXE, .B00-.B99, .C00-.C99, .S00-.S99, .TXT apod.).

Příklad rozsáhlejšího projektu SGP:

Název soub.	*Popis*	*Vytváří*
`HRA1.SGP`	– zdrojový kód programu ve tvaru SGP	uživatel (Sedit)
`HRA1.INC`	– vkládaný zdrojový soubor (příkazem #include)	uživatel (Tedit)
`HRA1.B00`	– stránka 0 s prvními 150 předměty z banky HRA1	uživatel (Paint)
`HRA1.B01`	– stránka 1 s druhými 150 předměty z banky HRA1	uživatel (Paint)
`HRA1.B*`	– další stránky, každá se 150 předměty z banky HRA1	uživatel (Paint)
`(HRA1.C00`		– stránka 100 s novými animacemi a rámečky pro Baltazara uživatel) (Paint)
`(HRA1.C01`		– stránka 101 s dalšími 150 předměty z banky HRA1 uživatel) (Paint)
`(HRA1.C*`	– další stránky, každá se 150 předměty z banky HRA1	uživatel) (Paint)
`HRA1.S*`	– Baltazarovy scény	uživatel (ScenEdit)
`HRA1.TXT`	– doplňující textový soubor - například příručka	uživatel (Tedit)
`HRA1.C`	– přeložený program do tvaru jazyka C	program SGP
`HRA1.CIN`	– přeložený program do tvaru CIN - lze spustit	program SGP
`HRA1.EXE`	– přeložený program do tvaru EXE - lze spustit	program SGP
`HRA1.BAK`	– nějaký záložní soubor (třeba z textového editoru)	nějaký program

SGP Manažer

Pro co nejjednodušší práci se soubory a s celými projekty je v systému SGP zabudován jednoduchý a efektivní manažer souborů (SGP Manažer). Tento manažer umí vytvářet, kopírovat, přejmenovávat a mazat jednotlivé soubory, celé projekty i celé adresáře.

SGP Manažer má 3 pohledy (filtry) na soubory:
pohled Projekty - jsou zobrazeny pouze projekty SGP v daném adresáři (Alt+1)
pohled Soubory projektu - jsou zobrazeny pouze soubory daného projektu (Alt+2)
pohled Soubory - jsou zobrazeny všechny soubory v daném adresáři (Alt+3)

Můžeme si však nastavit libovolný pohled (filtr) sami, zadáním tohoto filtru do jména souboru. Chceme-li například vybrat v aktuálním adresáři pouze všechny soubory s příponou .EXE, zadáme do jména souboru pouze: *.exe (stačí také jenom .exe).

Mezerníkem se lze snadno přepínat mezi prvními dvěma pohledy, tj. mezi Projekty a Soubory projektu.

SGP Manažer také umožňuje "senzitivní" spuštění příslušného podsystému SGP dle přípony daného souboru. Pokud tedy stiskneme Enter (nebo levé tlačítko myši) na nějaké položce v adresáři, spustí se následující aplikace:

Přípona soub.	***Činnost po Enter***
.SGP	SGP Sedit - editor struktur
.B00-B99	SGP Paint - celostránkový grafický editor
.C00-C99	SGP Paint - celostránkový grafický editor
.BMP	SGP Paint - celostránkový grafický editor (umožňuje převod BMP <>B00)
.S00-S99	SGP ScenEdit - editor baltazarovských scén (úsporný formát obrázků)

.ICO SGP IconEdit - editor ikon pro Windows
.CIN SGP Cinter - C interpret = spustí program CIN v integrovaném prostředí
.EXE SGP Cinter - C interpret = spustí program CIN v integrovaném prostředí
Ostatní přípony - spustí se SGP Tedit - textový editor.

Pokud chceme *spustit textový editor na jakémkoliv souboru (i SGP)*, stiskneme *Ctrl+Enter*.

Spuštění hotového programu, např.: Čaruj.

Spustíme SGP Manažer souborů. Z menu vybereme Soubor/Manažer (nebo stiskneme Ctrl+M).

Objeví se seznam projektů (programů) dodávaných na instalační disketě.

Jsou celkem 3 možnosti, jak spustit program z SGP Manažera.

1. ***Nejjednodušší způsob:*** Kurzor nastavíme na projekt CARUJ a stiskneme Alt+P (Spustit). Spustí se program CARUJ.

2. ***Ze strukturogramu,*** *z Seditu:* Pokud na projektu CARUJ stiskneme Enter, program CARUJ (soubor CARUJ.SGP) se zavede do paměti a zobrazí se jeho strukturogram. Spuštění programu CARUJ přímo ze struktury provedeme stisknutím klávesy F9 - Spustit program nebo pro jistotu Ctrl+F9 - Přeložit znovu a spustit program.

3. ***Z textového editoru,*** *z Teditu:* Na projektu CARUJ stiskneme Enter (přepneme se do Seditu viz výše), ve strukturogramu stiskneme Ctrl+Enter (přepneme se do Teditu). Z Teditu můžeme, stejně tak jako z Seditu, program spouštět klávesami F9 a Ctrl+F9.

My budeme používat zásadně první způsob, protože je nejrychlejší, nejjednodušší a nejbezpečnější.

Ve Windows95 můžeme program spustit stejným způsobem, nebo můžeme použít výhody Windows95 a program spouštět tak, že pouze přetáhneme ikonu souboru, který chceme spustit nad ikonu SGP Baltazara 5.0. Dle přípony souboru se vždy spustí automaticky příslušný podsystém SGP, viz tabulka "Přípona souboru - Činnost po Enter" výše.

V MS-DOSu můžete dělat totéž, spouštěním programu `sgpbalt` s parametry v příkazovém řádku. Například:
`>sgpbalt caruj.ico` (nebo sgpbalt /i) - spustí editor ikon
`>sgpbalt caruj.cin` (nebo sgpbalt caruj.exe, nebo sgpbalt caruj /r) - spustí program Čaruj
Všechny možné parametry programu sgpbalt získáte příkazem: `>sgpbalt /?`

Po spuštění programu CARUJ, vidíme na obrazovce Baltazara, jak netrpělivě poklepává nožkou – čeká, až stiskneme nějakou klávesu. Vyzkoušíme si klávesy nabídnuté v menu, v dolní části obrazovky.

Jakýkoliv spuštěný baltazarovský program můžeme ukončit předčasně stisknutím klávesy Esc (nebo Ctrl+Break). Po přerušení se hledá místo přerušení programu ve zdrojovém textu. Protože to nás v tuto chvíli nezajímá, stiskneme ihned klávesu Esc, pro ukončení hledání místa přerušení.

Takovýmto způsobem si můžeme vyzkoušet spustit všechny vzorové programy.

Pozor! Pokud máte pouze DEMO verzi SGP Baltazara, nepůjdou vám již přeložit a spustit větší programy než je program CARUJ!

Žáky ponecháme, aby si sami vyzkoušeli spouštění ostatních, již hotových programů, z aktuálního adresáře a baltazarovské hry, které učitel může připravit i do jiných adresářů.

Ukončení programu SGP Baltazar

Program SGP Baltazar ukončíme buď pomocí menu nebo kombinací kláves Alt+X.

SGP Baltazar – příprava č. 4

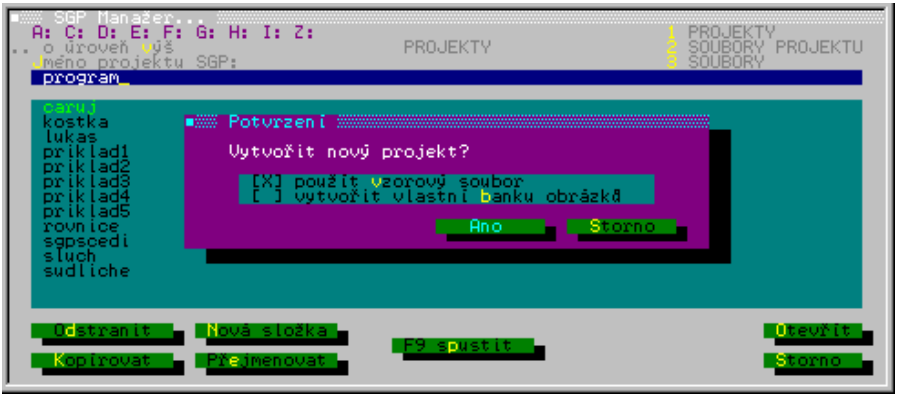
Téma: Vytvoření nového programu

Spustíme program SGP Baltazar a z menu vybereme volbu Soubor/Manažer. Do jména projektu napíšeme jméno nového programu.

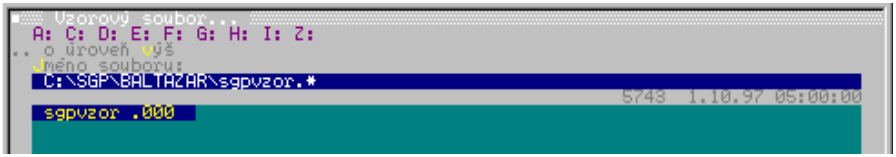
Poznámka: Pokud zadáme jméno programu, který již existuje, nevytvoří se nový soubor, ale načte se ten, jehož jméno jsme zadali, neboť se předpokládá, že chceme pracovat se zadaným souborem.

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

Program nabídne způsob vytvoření nového projektu, tj. zda chceme použít vzorový soubor (implicitně) a zda se má vytvořit vlastní banka obrázků (implicitně potlačeno).



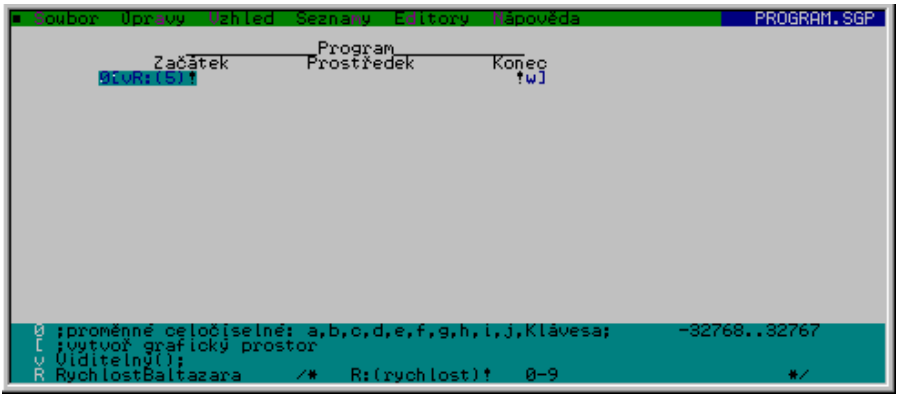
Stiskneme tedy pouze Enter (tj. Ano) a protože jsme měli zatrženu volbu "použít vzorový soubor" nabídne se nám seznam vzorových souborů. K systému se dodává jeden vzorový soubor, který stačí na většinu žákovských programů. Učitel si může připravit další vzory.



My si vybereme standardní vzorový soubor sgpvzor.000 a potvrdíme Enter. Svůj vlastní vzorový soubor vytvoříme tak, že soubor sgpvzor.000 zkopírujeme s novou příponou a tento nový soubor upravíme dle potřeby (stiskneme Ctrl+Enter na souboru sgpvzor.xxx).

Náš nový projekt je vytvořen.
Na obrazovce se objeví jednoduchý strukturogram připravený dle vzoru. Vysvětlíme dětem, že strukturogram je grafický zápis našeho programu (neboli grafický zápis

algoritmu). Obsahuje časově uspořádané prvky se jmény objektů a jejich postupné členění na menší objekty. Názvy objektů jsou podstatná jména, začínají velkým písmenem a jsou zobrazeny černě.



Hlavní struktura - objekt Program - je tvořena posloupností tří částí (objektů): Začátek, Prostředek, Konec (vykonávají se 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 také 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 strukturogramu se můžeme pohybovat kurzorem pomocí kurzorových kláves nebo pomocí myši. Nové prvky strukturogramu 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 (zatím nebudeme potřebovat)
- [- vytvoř grafický prostor s Baltazarem, Baltazar je ještě neviditelný
- v - učiň Baltazara viditelným
- R:(5) - nastav rychlost Baltazara, 0 – je nejmenší rychlost, 9 – je maximální rychlost

Operace pod objektem s názvem Konec:

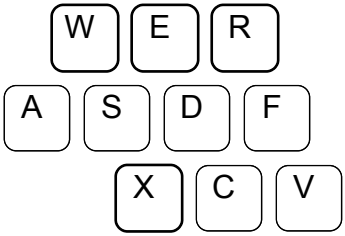
- w - čekej na stisknutí libovolné klávesy
-] - zruš grafický prostor

Nyní si ukážeme, jak se do strukturogramu vkládají operace (příkazy). Chceme například, aby Baltazar popošel o 1 políčko.

Příkazy, které má Baltazar vykonat mezi Začátkem a Koncem, uvedeme pod objektem Prostředek. Kurzor nastavíme na Prostředek.

Pro vložení nového prvku se používají tyto klávesy:

buď



Alt+šipka - vloží nový prvek ve směru šipky

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

Standardní způsob vkládání nových prvků do strukturogramu je pomocí kláves WERX nebo Alt+šipka.

Jak tedy vložíme příkaz Popojdi(1).

1. Standardní postup - vhodný pro začátečníky.

Stiskneme klávesu "X" (vložit pod prvek), pod Prostředkem začne blikat kurzor a my bychom mohli přímo psát příkaz pro Popojdi. Zatím však nevíme jak, takže budeme postupovat dle nápovědy. Vpravo je zobrazena tabulka možných znaků, které můžeme v tomto místě strukturogramu použít. Nastavíme kurzor na "!", to znamená, že budeme vkládat příkaz a stiskneme klávesu Ins (přenést znak do strukturogramu a pokračovat v nabídce), klávesa Enter by také přesunula "!" do strukturogramu, ale vkládání by se ukončilo. Museli bychom znovu stisknout Enter. Po stisku klávesy Ins se pod Prostředkem objeví "!" a tabulka povolených znaků vpravo se změní na seznam definovaných operací (maker). Všimněme si, že makra již použitá ve strukturogramu (tzv. lokální makra) jsou zobrazena bílo-černě, zatímco ostatní makra (zatím nepoužitá) jsou tmavě šedá. Tato "šedá" makra jsou tzv. globální makra, která jsou definována na začátku souboru a my je můžeme používat v každé funkci, aniž bychom je museli pokaždé znovu definovat. Co je to funkce programu se dozvíme v dalším.

Nás zajímá příkaz Popojdi(1), takže jej začneme hledat pomocí kurzorových kláves tak, že šipkou dolů se budeme pohybovat seznamem globálních maker dolů, dokud nenarazíme na makro "p Popojdi(1)".

Všimněme si, že v každé chvíli vidíme ve strukturogramu právě zobrazovanou zkratku makra (1 písmeno) a dole celou definici daného makra. Tentokrát vložíme makro do

strukturogramu klávesou Enter, čímž současně také ukončíme celé vkládání nového prvku-operace do strukturogramu.

Po stisku klávesy Enter se vlastně stalo přesně toto: makro "p Popojdi(1)" se přesunulo z globálního seznamu maker do lokálního seznamu a odtud pak do strukturogramu. Nakonec se ukončilo celé vkládání prvku "p!" a editor prvku se přepnul zpět do editoru struktury (Seditu), takže se můžeme opět pohybovat po strukturogramu a vkládat jeho další prvky.

2. Rychlejší postup - vhodný pro pokročilé.

Namísto "X" můžeme stisknout rovnou "!" (tím systému sdělíme, že chceme vkládat pod daný prvek příkazy, takže se pravé straně obrazovky automaticky zobrazí seznam všech maker, která můžeme v tuto chvíli použít) a pak stiskneme "p" a Enter. Pokud "p" již je lokální makro, rovnou se vloží před "!".

Pokud makro "p" ještě není lokální, ale je globální, zkopíruje se nejprve do lokálního seznamu a pak se vloží do strukturogramu.

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

Pokud bychom chtěli vytvořit nové lokální makro stikneme Ctrl+Enter, tj. spustí se textový editor na makru "p" pro opravu definice tohoto makra. Příkazy nového makra zapíšeme přesně dle syntaxe jazyka C a ukončíme textový editor klávesami F2 (konec s uložením) nebo Esc (konec s dotazem na uložení).

Vytváření nového makra můžeme kdykoliv přerušit klávesou Esc. Pokud chceme vytvořit nové makro, ale nechat jej zatím prázdné (tj. bez textu), stiskneme v textovém editoru pouze Ctrl+Shift+šipka dolů a takto zvýrazněný blok smažeme klávesou Del.

Poznámka!: ***Při zadávání myší stačí pouze klepnout levým tlačítkem myši pod, nad, vpravo nebo vlevo od vyznačeného prvku a ze seznamu vpravo vybrat příslušné makro. Přerušení vkládání myší lze klepnutím pravým tlačítkem myší na ploše struktury - mimo strukturu. Ukončení vkládání s potvrzením pak provedeme klepnutím levého tlačítka myši kdekoliv na volné ploše.***

Hotový program spustíme klávesou F9 nebo myší (z lokálního menu, nebo celkového menu).

Pokud chceme vytvořit samospustitelný soubor PROGRAM.EXE stiskneme CTRL-E (platí pouze pro verzi Profi). Jinak se vždy vytváří soubor .CIN, který lze spouštět i z DEMA.

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 strukturogram. 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.

Poznámka: Pro lepší orientaci učitelů, jsou příkazy (makra) ve vzorovém souboru označeny dle následujícího pravidla.
Příkazy, které jsou definovány včetně parametrů jsou označeny malými písmeny. Např.: `p Popojdi(1)` se použije takto `p!`.
Příkazy, které je nutno doplnit parametrem jsou označeny velkými písmeny. Např.: `P Popojdi` se použije takto `P:(1)!`.

SGP Baltazar – příprava č. 5

Téma: Sedit – práce s editorem strukturogramů

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

Způsob čtení a formu zápisu strukturogramů

Připomeneme způsob čtení a formu zápisu elementárního strukturogramu, se kterým se pracovalo v minulém tématu – členění jednotlivých objektů ve strukturogramu na další menší objekty směrem dolů s tím, že objekty stejné úrovně jsou uvedeny na stejné úrovni i ve strukturogramu 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 strukturogramu
- prvek strukturogramu 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

Úprava 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.

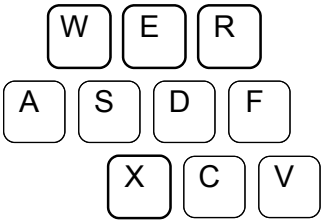
```
_____________________PROGRAM_____________________
Začátek               Cesta                   Konec
0[vR:(5)!               p!                       !w]
```

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:



Alt+šipka

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

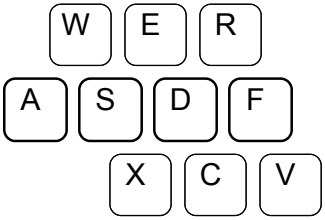
nebo

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 strukturogramu:

Posouvání prvků a bloků

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



S - posun názvu prvku vlevo
D - posun názvu prvku vpravo

A - posun bloku vlevo nebo Ctrl+šipka
F - posun bloku vpravo nebo Ctrl+šipka

posun bloku nahoru - Ctrl+šipka nahoru
posun bloku dolů - Ctrl+šipka dolů

Mazání bloku

Celý blok můžeme smazat pomocí kláves Ctrl+Del.

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 schránky nad, pod, vlevo nebo vpravo od aktuálního prvku
Ctrl+Del - smaže blok

Posouvání struktury po ploše myší

Shift + → - posun struktury vpravo po ploše mimo
Shift + ← - posun struktury vlevo
Shift + - posun struktury nahoru
Shift + ↓ - posun struktury dolů

levé tlačítko myši stiskneme na strukturu a táhneme myší v požadovaném směru

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

Práci s editorem strukturogramů ukončíme pomocí

F2 - strukturogram bude zapsán na disk a zobrazí se seznam funkcí
Esc - zobrazení seznamu funkcí, po výběru nové funkce se program se zeptá, jestli strukturogram má být zapsán na disk.

SGP Baltazar – příprava č. 6

Téma: Struktury posloupnost a opakování

Na tabuli nakreslíme strukturogram z již vyzkoušeného programu.

```
                  _____________PROGRAM_____________
      Začátek                Cesta                Konec
  0[vR:(5)!               pppp!                   !w]
```

Struktura posloupnost

Zopakujeme, že jednotlivé objekty jsou v nakresleném strukturogramu uvedeny v takovém pořadí, v jakém se budou vykonávat. 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.*

```
  _________Cesta________                    Cesta
 Pole   Pole   Pole   Pole       =           #N:(a,4)
                                            Pole
```

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

```
                  _____________PROGRAM_____________
      Začátek                Cesta                Konec
  0[vR:(5)!                   #N:(a,4)              !w]
                              Pole
                              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 tzv. „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 „*“.

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

Cesta je 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 se má opakovat, je nutno zjistit, zdali se v opakování má ještě pokračovat. Do strukturogramu 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 může v chůzi pokračovat.

Po názorném vysvětlení přistoupíme k práci s počítačem. Zadáme úkol, ve kterém má Baltazar přejít celou obrazovku (nevíme, kolik polí je široká). Řešení bude odpovídat výše uvedenému strukturogramu 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 tak, aby byl natočen doprava.

```
                    _____________________PROGRAM_____________________
     Začátek   Cesta     Pravý~okraj      Cesta     Levý~okraj  Konec
0[vR:(5)!        *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)!            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.

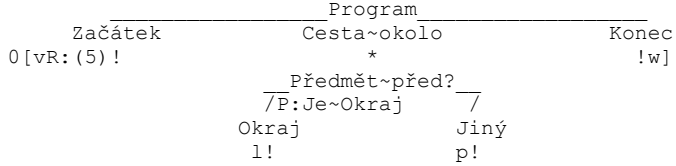
SGP Baltazar – příprava č. 7

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.

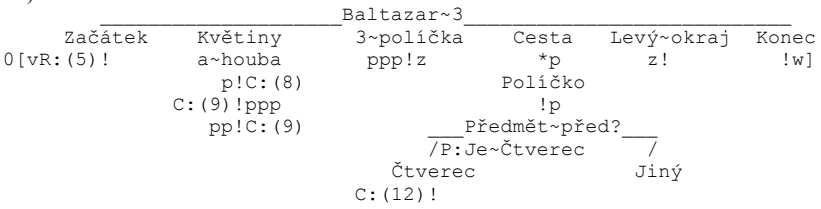
Řešení: Když dorazí k okraji prostoru, otočí se doleva a bude pokračovat dál.



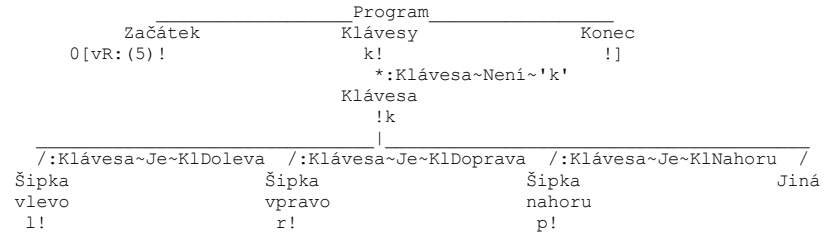
Baltazar bude chodit stále dokola obrazovky, protože ve strukturogramu není u „*“ uvedena podmínka pro ukončení opakování. Program přerušíme klávesou Esc.

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 strukturogramu 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.

```
     Strukturogram          Odpovídající kód v Pascalu

        Soubor              {Soubor}
      :Čti!                 Čti;
         *                  while True do begin
       Záznam                   zpracuj;
  :zpracuj!:Čti                 Čti;
                            end
```

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

Krásnou ukázkou programu pro čtení a zpracování kláves je program CARUJ.SGP v adresáři s SGP Baltazarem.

Poznámky

SGP Baltazar – příprava č. 8

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

Příkazy a podmínky definované ve vzoru SGPVZOR.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 Alt+L volby Lokální definice.

Seznam příkazů a podmínek:

```
-ol-Globals --Makropříkazy - operace -------------------------------------
0 ;proměnné
  ;globální proměnné (přepnutí do seznamu globální proměnných Ctrl+G):
    ;proměnné celočíselné: gA,gB,gC,gD,gE,gF,gG,gH,gI,gJ,gKlávesa;
    ;ostatní předdefinované globální proměnné mají stejné názvy jako dále
    ;definované lokální proměnné, mají však předponu g... (např. gDČíslo2)
  ;lokální:
    ;proměnné celočíselné:
       int     a,b,c,d,e,f,g,h,i,j,Klávesa,      // rozsah -32768..32767
               IČíslo1,IČíslo2,IČíslo3,PůvBarvy;
    ;proměnné reálné:
       double  DČíslo1,DČíslo2,DČíslo3;          // rozsah 5.0E-324..1.7E308
    ;proměnná znak:
       char    Znak;
    ;proměnná řetězec:
       string  Řetězec;
[ ;vytvoř Baltazarův grafický prostor
   VytvořProstor(); MyšZačátekObsluhy();
] ;zruš Baltazarův grafický prostor
   MyšKonecObsluhy(); ZrušProstor();
a Klávesa= ČtiKlávesuZFronty();
b PřepniNaBaltazara();
c Čaruj(Čtverec);
d GrČtiZnak("?",Znak,1,0x07);             /*      edit ano, černá/svšedá*/
e BezObláčku();
f VyprázdniFrontuKláves();
g GrČtiŠetězec("?",Řetězec,40,1,0x3f);   /*40zn, edit ano, modrá/bílá */
h GrČtiIČíslo("?",IČíslo1,6,0,0x0e);      /* 6zn, edit  ne, černá/žlutá*/
i GrČtiDČíslo("?",DČíslo1,20,1,0x0f);     /*20zn, edit  ne, černá/bílá */
j GrPišZnak(Znak);
k Klávesa= ČtiKlávesuZFrontySČekáním();
l VlevoVbok();
m ;vypiš Řetězec do grafického okénka
   PůvBarvy=GrNastavBarvy(Modrá<<4|Žlutá); /* žlutá na modré          */
   GrPišŘetězec(Řetězec);            /* vypíše obsah proměnné Řetězec */
   GrNastavBarvy(PůvBarvy);          /* nastaví původní barvy         */
n Neviditelný();
o SObláčkem();
p Popojdi(1);
q GrPišLČíslo(IČíslo1,-1);           /* -1 = nejkratší výpis          */
r VpravoVbok();
t PřepniNaText();
u GrPišDČíslo(DČíslo1,10,2);         /* 10 číslic, z toho 2 desetinné */
v Viditelný();
w ;čekej na stisk klávesy nebo tlačítka myši
   VyprázdniFrontuKláves();
   MyšVyprázdniFrontuUdálostí();
   Čekej(NaKlávesu|NaMyš);
   VyprázdniFrontuKláves();
```

```
   MyšVyprázdniFrontuUdálostí();
z ;čelem vzad
   VlevoVbok(); VlevoVbok();
B NastavBarvuČarování   /*       B:(barva)!   0-15                    */
C Čaruj                 /*     C:(předmět)!   1-150                   */
L NačtiScénu            /*    L:("soubor")!   "c:\\pavel\\soubor.s00" */
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.s00" */
W Čekej                 /*  W:(milisekund)!   0-32000                 */
X ČarujNaPozici         /* X:(předmět,x,y)!   1-150, 1-15, 1-10       */
-c1-Globals -Podmínky--(? lze použít jako náhrada pouze pro /,*,+) ----------
?n PrůchodnostPředmětuPředB() Je 0
?p PrůchodnostPředmětuPředB() Je 1
?B BarvaČtvercePředB()     /* Je, Není, <, >, <= ,>= */
?P PředmětPředB()          /* Je, Není, <, >, <= ,>= */
?S SměrBaltazara()         /* Je, Není, <, >, <= ,>= */
#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
===Globals ------------------------------------------------------------------
```

Jednotlivé příkazy a podmínky využívají funkce, které jsou součástí knihovny Baltazara (podmínky # jsou makra). Jednotlivé funkce zapíšeme v textovém editoru s využitím integrované nápovědy příkazů (SGP Helper), 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 strukturogramu, 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“ 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“.

Okno "Detaily - Náhled C" (zobrazení zdrojového kódu C)

Pokud stiskneme Mezerník (Detaily) na prvku Název (černé písmo), zobrazí se náhled na vygenerovaný zdrojový kód tohoto prvku, velice podobný skutečně generovanému zdrojovému kódu v jazyce C. V náhledu jsou vypuštěny pouze některé znaky, které znepřehledňují kód (např.: ";", složené závorky a pod.).

SGP Baltazar – příprava č. 9

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

Název programové jednotky – komentář

Prvek strukturogramu, který není 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 může být použito pouze uvnitř názvu.

Název programové jednotky (bloku) 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 strukturogramu použita vícekrát. Na dalších místech již není nutné rozkreslovat tutéž programovou jednotku, ale stačí se pouze odkázat na její definici.

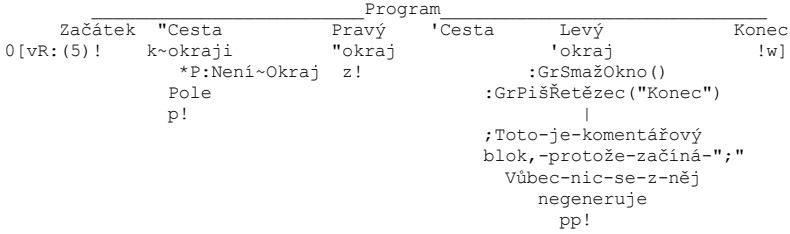
Znak ' (apostrof) – volání podstruktury
Znamená odkaz na programovou jednotku, která je definována na jiném místě ve strukturogramu. Může být definována kdekoliv před voláním i za voláním.

Znak : (dvojtečka) – přímý text
Přímý text (příkaz), používá se výjimečně, většinou pro zápis parametrů makra. Přímý text se beze změny přepisuje do generovaného kódu.

Znak ; (středník) – komentářová programová jednotka (komentářový blok)
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. Komentářová jednotka je celá šedá.

Samostatná práce

Žákům zadáme některá 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!
Pravý "okraj z!
'Cesta
Levý 'okraj
:GrSmažOkno()
:GrPišŘetězec("Konec")
|
;Toto-je-komentářový blok,-protože-začíná-";"
Vůbec-nic-se-z-něj negeneruje
pp!
Konec
!w]

SGP Baltazar – příprava č. 10

Téma: Tedit – práce s textovým editorem příkazů

Doposud jsme ve strukturogramech používali již připravené příkazy, protože jsme při vytváření nového programu použili vzorový soubor (např. SGPVZOR.000) nebo přímé vkládání příkazů z SGP Helper-u (Příkazového pomocníka - klávesa F3, nebo Shift+Mezerník nebo Ctrl+Mezerník).

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 (editoru struktur), kdy kurzor je nastaven na prvku název (není nastaven na žádnou operaci či podmínku), otevře se nám okno s kurzorem, nastaveným na hlavičce funkce, 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, že kurzor v Seditu stál pouze jedné operaci nebo na podmínce, otevře se Tedit přímo, s kurzorem nastaveným na začátku příslušné definice operace/podmínky.

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 stisku 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[vR:(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 nebo novější dle C++ dvě lomítka "//", tj. komentář od lomítek až po konec řádku.

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. Doporučujeme však, zvyknout si psát středníky za každým příkazem.

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--Seznam operací---), podmínky v seznamu podmínek (-c1--Seznam podmínek) 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 (nebo kombinací Alt+X) 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 SGPVZOR.000 se opět z tohoto vzoru přenese operace* `z ;čelem vzad` *do našeho nového programu.*

Příkazový pomocník (SGP Helper)

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 v grafickém režimu, zadáte-li "kl" - vyhledají se všechny příkazy, které mají něco společného se zpracováním klávesy, zadáte-li "kons" - vyhledají se všechny definice konstant, atd. Doporučujeme - **vyzkoušejte si Příkazového pomocníka SGP !**

Hledání funkce - novinka od verze 5.0

Pokud v Teditu na nějakém názvu funkce stisknete Ctrl+F1 vyhledá se automaticky definice této funkce. A to i vaší vlastní. Algoritmus je následující: nejprve se hledá funkce definovaná v programu, když se nenalezne, hledá se v knihovně Baltazar.

Vyzkoušejte si všechny příkazy Teditu! Nápověda - klávesou F1.

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
   GrPišŘetězec('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	PředmětPředB() Je Čtverec
!=	Není	BarvaČtvercePředB()Není Černá
<	JeMenšíNež	
>	JeVětšíNež	
<=	JeMenšíNeboRovno	
>=	JeVětšíNeboRovno	
!	Neplatí	Neplatí PrůchodnostPředmětuPředB()
&&	ASoučasně	PředmětPředB() Je Čtverec ASoučasně BarvaČtvercePředB() Je Bílá

| | nebo Směr() Je Sever nebo Směr() Je Jih

Konstanty

Použijte vysvětlení konstant v SGP Helperu - Jazyk C / Základní pojmy v jazyce C.

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

Výčet funkcí knihovny Baltazar, relační operátory a konstanty je možno opět zobrazit stisknutím F3 nebo Shift+Mezerník v Seditu nebo Teditu a zadání klíčového slova "operátor" nebo "konst".

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.

Proměnné

Proměnné si představme jako přihrádky, do nichž budeme něco ukládat. Podle toho, co do nich budeme ukládat, budeme potřebovat různě veliké přihrádky. Někdy malé pro čísla, jindy větší pro řetězce a někdy celou skříň přihrádek pro pole čísel nebo řetězců.

Každou proměnnou si musíme pojmenovat, nejlépe tak, abychom vždy přesně věděli, co jsme si do ní uložili. Takže namísto nic neříkajícího FX7abR použijeme raději zcela jasný název JménoOsoby apod.

Velikost každé proměnné je tedy dána typem dat, která do ní budeme ukládat. Jazyk C poskytuje tzv. typovou kontrolu, to znamená, že nemůžeme vkládat třeba text do přihrádky určené pro čísla.

Blíže o konstantách, proměnných a funkcích se dočtete v Příkazovém pomocníku SGP (SGP Helperu) po stisku Ctrl+Mezerník v Seditu, Teditu nebo z hlavní obrazovky.

Poznámky

SGP Baltazar – příprava č. 12

Téma: Vytvoření programu bez vzorových příkazů

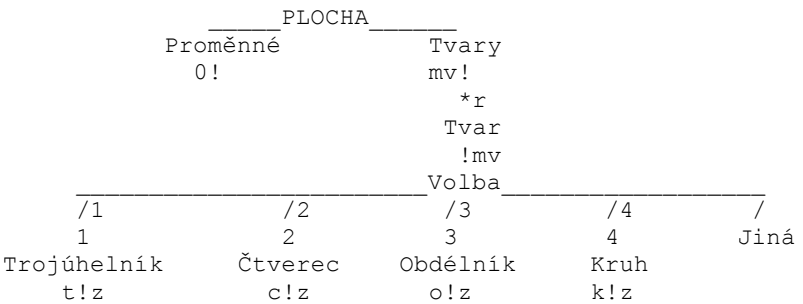
Nyní, když jsme se naučili jak upravovat, rušit a vytvářet své vlastní operace a podmínky v seznamu operací a podmínek, který byl vytvořen podle vzoru, když jsme si ukázali, jak v našich vlastních operacích a podmínkách používat funkce, které jsou v knihovně Baltazara, ukážeme si nyní, jak vytvořit program,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

Uděláme to jednoduše tak, že při vytváření nového souboru zrušíme volbu "použít vzorový soubor". Vytvoří se prázdný soubor, ve kterém již nejsou žádná globální makra, takže se nám již nebude nic automaticky nabízet.

Pokud chceme smazat globální definice maker z již existujícího programu, který byl vytvořen ze vzorového souboru, stiskneme Ctrl+G (globální definice) a nechtěné definice maker odstraníme. Pak se nám již při vkládání příkazů ve strukturogramu nebudou žádná globální makra 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.

Strukturogram programu PLOCHA:



Seznam operací struktury PLOCHA:

```
-o1-PLOCHA
0 ; Deklarace proměnných
    int      volba;    /* Volba tvaru      */
    double   a, b,     /* Parametry tvaru  */
             P;        /* Plocha           */
c ; Zadej délku strany a vypočítej obsah čtverce
    TxtPišŘetězec("Zadej délku strany čtverce: ");
    TxtČtiDČíslo(a);
    // plocha čtverce
       P = a * a;
```

```
k ; Zadej poloměr a vypočítej obsah kruhu
    TxtPišŘetězec("Zadej poloměr kruhu: ");
    TxtČtiDČíslo(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: ");
    TxtČtiDČíslo(a);
    TxtPišŘetězec("Zadej stranu b obdélníku: ");
    TxtČtiDČíslo(b);
    P = a * b;
t ; Zadej délku strany, výšku a vypočítej obsah trojúhelníku
    TxtPišŘetězec("Zadej délku strany trojúhelníku: ");
    TxtČtiDČíslo(a);
    TxtPišŘetězec("Zadej výšku trojúhelníku       : ");
    TxtČtiDČíslo(b);
    P = a * b / 2;
v ; Čti volbu z klávesnice
    TxtČtiČíslo(volba);
z ; Zobraz výsledek
    TxtPišŘetězec("Plocha je: ");
    TxtPišDČíslo(P, -1, 2);
    TxtPišŘetězec("\n-----------------------------------------")
```

Seznam Podmínek struktury PLOCHA:

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

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

Seznam všech lokálních maker (tj. operací a podmínek) zobrazíme stiskem Ctrl+L.

Do části globálních definic se přepneme stiskem Ctrl+G.

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 strukturogram.

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. Od verze 5.0 již nemusíme při umisťování nové funkce dbát na to, aby tato funkce byla před funkcí, která ji volá. Od SGP Baltazara verze 5.0 již můžeme funkce umisťovat dle abecedy, tedy tak, jak se nám nejlépe budou hledat.

Novou funkci vytvoříme v seznamu funkcí stisknutím kombinace kláves Alt+N a zadáním jména nové funkce.

Jméno funkce může obsahovat pouze povolené znaky pro jména v jazyce C, rozšířené v SGP Baltazarovi o diakritiku, tj. nesmí začínat číslicí a může obsahovat pouze znaky abecedy, číslice a znak podtržítko (_).

Při zadávání nové funkce (nebo i při přejmenování) je kontrolováno, zda se nejedná o duplicitní název.

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 po vytvoření nové funkce do textového editoru (Ctrl+Enter) a upravíme hlavičku funkce.

Příklad 4 možných definic:
Pozn.:
`void=()= nic`

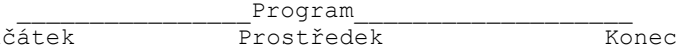
```
void Nuluj()
void PišRSouřadnice(double x, double y)
int PočetSekund()
double ObvodKruhu(double r)
```

Definice funkce

Programování nové funkce provádíme již známým způsobem v Seditu. V novém strukturogramu je na počátku jen jeden prvek, kterým je název 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 strukturogramu musí být uvedena operace, která funkci volá stejným způsobem, jako by se jednalo o funkci z knihovny Baltazara. Pozor, nezapomeňte na závorky za názvem funkce! Musí být uvedeny vždy, i když funkci nepředáváme žádný parametr.



Když začneme ve svých programech vytvářet nové funkce, brzy zjistíme, že seznamy operací a podmínek, které jsou neustále nabízeny při vkládání (pokud jsme vytvořili nový soubor dle 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, nebo vytvořit nový soubor bez použití vzoru. Ačkoli musíme vytvořit úplně celý strukturogram 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 vkládáme do seznamu operací dané funkce (-o1--Seznam operací--) a podmínky do seznamu podmínek dané funkce (-c1--Seznam podmínek--). Seznam všech použitých operací a podmínek se zobrazí po stisku Ctrl+L.

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

Nepotřebnou funkci smažeme ze seznamu funkcí klávesou Del nebo Alt+D.

Seznam funkcí

V seznamu funkcí můžeme s funkcemi provádět tyto operace:
- vytvořit novou funkci
- smazat funkci

Nové od verze 5.0
- přejmenovat funkci v celém souboru, tj. i všechna volání funkce
- přesunout funkci (můžeme je bez problémů poskládat třeba podle abecedy)
- zkopírovat funkci

Seznam funkcí vyvoláme z Seditu klávesou Esc a ukončíme také klávesou Esc.

Kombinací kláves Ctrl+U se můžeme v Seditu přepínat mezi dvěma posledně zobrazenými funkcemi.

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

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

```
    ___________MojeFunkce1__________
Zpracování                      Návrat
 vov!                            n!
```

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

```
o ; Přiřaď proměnné a hodnotu 5
    a = 5;
v ; Vypiš na obrazovku funkci a hodnotu proměnné a
    TxtPišŘetězec("Moje funkce : a = "); TxtPišIČíslo(a, -1);
    TxtPišNR();
n ; Navraťse do volající funkce a předej hodnotu 3
    return 3;
```

Strukturogram 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 = "); TxtPišIČíslo(a, -1);
    TxtPišŘetězec(", b = "); TxtPišIČíslo(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 ...
```

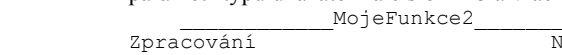
Z výše uvedeného výpisu vidíme, že hodnota `a` (je 1) byla předána funkci `MojeFunkce1`, která provedla příkaz `a = 5;`, ale po návratu z funkce `MojeFunkce1` zůstala hodnota proměnné `a` nezměněna. Program i funkce mají definovanou proměnnou `a`, ale jedná se o dvě různé proměnné. Funkci `MojeFunkce1`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

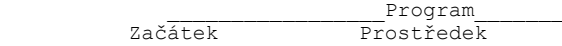
Strukturogram funkce `int MojeFunkce2(int *a)` (tj. funkce, která vyžaduje jeden parametr typu ukazatel na číslo `int` a vrací číslo `int`):



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 ");
    TxtPišIČíslo(*a, -1);
    TxtPišNR();
n ; Návrat do volající funkce
    return 3;
```

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



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 = "); TxtPišIČíslo(a, -1);
    TxtPišŘetězec(", b = "); TxtPišIČíslo(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 ...
```

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 přiřaď 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; */
TxtPišIČíslo(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 bez použití vzoru. Ve strukturogramu (Seditu) nebyly provedeny žádné úkony, nebyly definovány žádné operace či podmínky v Teditu. Z hlediska bezpečnosti je celý soubor .SGP přístupný pouze zvnějšku nějakým externím textovým editorem. Riziko externího přístupu, pak nese sám uživatel.

Jak tedy "prázdný soubor .SGP" vypadá:

```
SGPC   5.00 CZ0#5PN00000 23.09.1997 16:48:57

# include <sgpbalt.h>

/***** Globální proměnné *****/

/***** Globální makra *****/

-o1-Globals

-c1-Globals

===Globals

;-
h-------------------------------------------------------------

void main( void )

-s--TEST1

TEST1

-o1------------SEZNAM OPERACÍ-----TENTO ŘÁDEK
NEMĚNIT!-------

-c1------------SEZNAM PODMÍNEK----TENTO ŘÁDEK
NEMĚNIT!-------

=SP-----------------------------------------------------------
-

/***** Konec souboru *****/
```

Na prvním řádku jsou uvedeny následující údaje preprocesoru SGP:

- typ preprocesoru (SGP pro programovací jazyk C)
- verze preprocesoru
- číslo licence programu
- datum vytvoření souboru .SGP

- čas vytvoření souboru .SGP

Pro správnou funkci preprocesoru jsou nutné jen první dva údaje.

Na druhém řádku je příkaz (direktiva `#include`) pro kompilátor jazyka C, který zajistí, že v daném místě se zpracuje soubor, uvedený v lomených závorkách (soubor SGPBALT.H). Tento soubor se nachází v adresáři Baltazara a obsahuje deklarace funkcí, definice konstant a maker. Příkaz pro kompilátor (direktiva `#include`) může být využit pro začlenění dalšího souboru (knihovny vlastních funkcí) do programu. V tomto případě by však název souboru 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 a je také rozpoznávacím řádkem pro samotný program SGP. Proto s tímto řádkem neprovádějte žádné změny!

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_funkce
je část obsahující strukturogram
-o1-hlavička seznamu operací
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-hlavička seznamu operací
je část uvádějící seznam podmínek včetně jejich definic

Úsek zpracovávaný preprocesorem je ukončen řádkem:
=SP----------

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   5.00 CZ0#5PN00000 23.09.1997 22:48:57
# include <sgpbalt.h>
;-h-----------------------------------------------------
int Velké(int c)
-s--Velké
          _____Velké________
  __Písmeno?_          Návrat
  /m        /           n!
 Malé      Jiné
 p!
-o1--Seznam operací-------------------------------------
n ; Vrať hodnotu ASCII písmena
    return c;
p ; Převeď malé písmeno na velké
    c = c - ('a'-'A');
-c1--Seznam podmínek------------------------------------
/m ; Je to malé písmeno
     c >= 'a' ASoučasně c <= 'z'
=SP-----------------------------------------------------
;-h-----------------------------------------------------
void main( void )
{
-s--Program
     _____Program_____
Deklarace         Písmena
   0!               c!
                     *k
                  Písmeno
                     !c
                _Zpracování_
             Převod      Tisk
              v!         t!
-o1--Seznam operací-------------------------------------
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--Seznam podmínek------------------------------------
*k ; Není konec
     znak Není KlEnter
=SP-----------------------------------------------------
}
/***** konec programu *****/
```

Práci v celém souboru .SGP pomocí externího textového editoru nedoporučujeme! Úkony uvedené na začátku je možné provádět spuštěním Teditu z Seditu.

Poznámka: Pro editování strukturogramu slouží výhradně příkaz Sedit, který automaticky kontroluje a zajišťuje správnost strukturogramu. Poškození strukturogramu 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.

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 nebo Alt+šipka dolů, pro vkládání nového prvku pod prvek, na kterém se nachází kurzor, a potom jsme napsali příslušný znak. V případě podmínek očekával Sedit 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 Teditu, tak z Seditu.

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

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

Program, který není zřejmý ze strukturogramu (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. Praktický 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 strukturogramy tvořili přehledně, aby výstižně pojmenovávali prvky strukturogramu 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 a prohlášení, že je skutečným autorem zaslaného díla.

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

Struktura dat na disketě:

```
\Město-škola
 |-\1-PRIBEH
 |  |-\PRIBEH1
 |  | |-PRIBEH1.SGP - zdrojový soubor programu
 |  | |-PRIBEH1.B*  - soubory s obrázky
 |  | |-PRIBEH1.C*  - (výjimečně)
 |  | |-PRIBEH1.S*  - soubory se scénami
 |  | |-PRIBEH1.TXT - popis programu
 |  | |-...........
 |  |-\PRIBEH2
 |.................
 |-\2-HRA
 |-\3-VYUKA
 |-\4-GRAFIKA
 |-\5-PRAXE
```

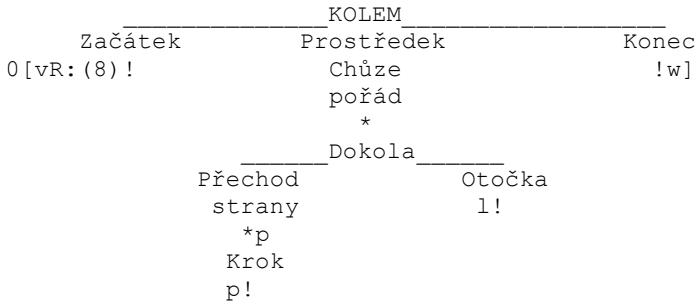
Bližší informace o soutěžích a aktuální soutěžní kategorie naleznete na domovské stránce firmy SGP Systems: http://www.sgp-systems.com .

Střední průmyslová škola, Uherské Hradiště
Řešené příklady
SGP BALTAZAR

Při řešení následující úkolů byl použit při tvorbě strukturogramu SGPVZOR.000.

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

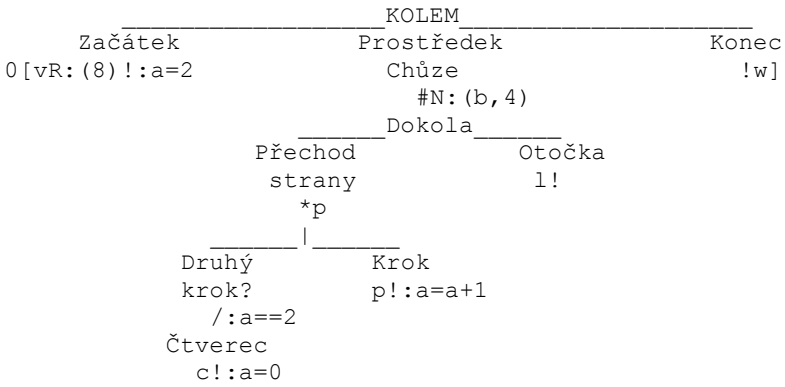
Řešení:



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.

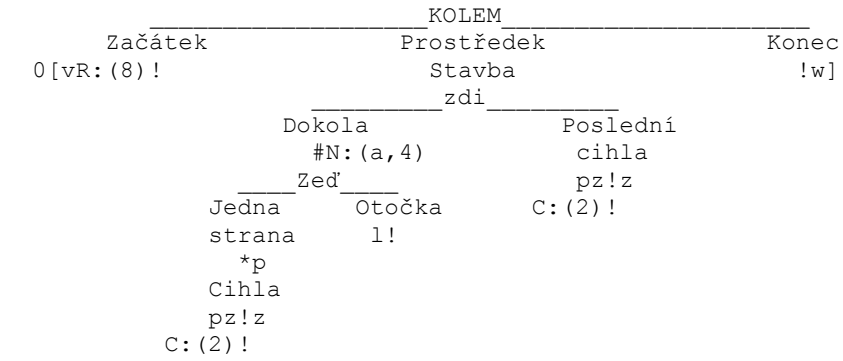
Řešení:



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.

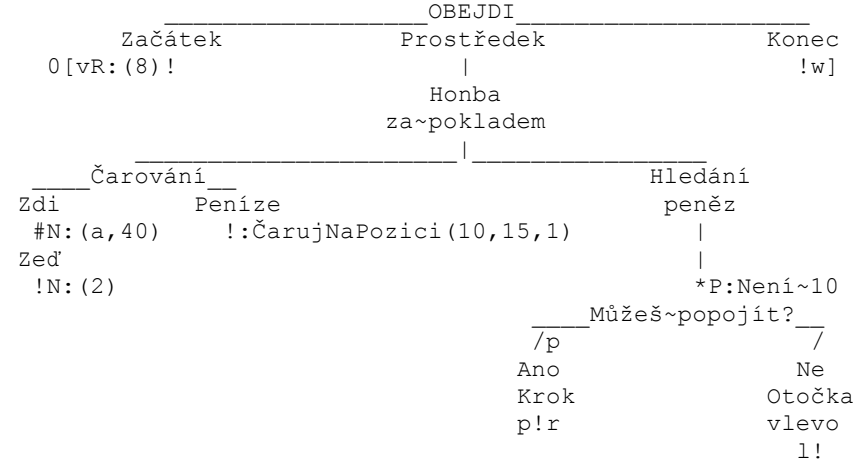
Řešení:



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[vR:(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 mohu 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[vR:(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 mohu 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[vR:(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).

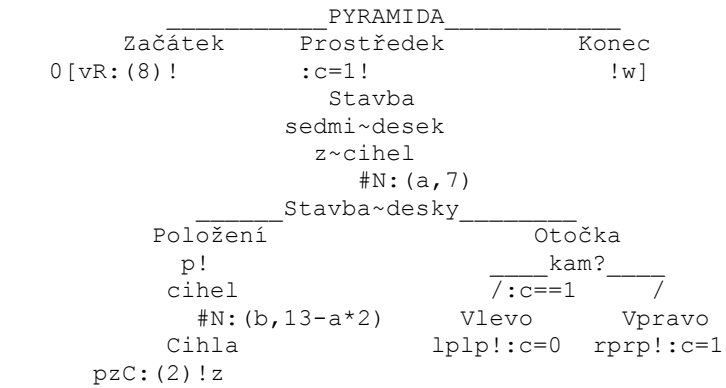
Úkol: Postavte stejnou pyramidu jako v předcházejícím případě, ale ne z průchodných čtverců, nýbrž z neprůchodné zdi.

Řešení 1:

```
             _____________PYRAMIDA_____________
       Začátek       Prostředek          Konec
  0[vR:(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.

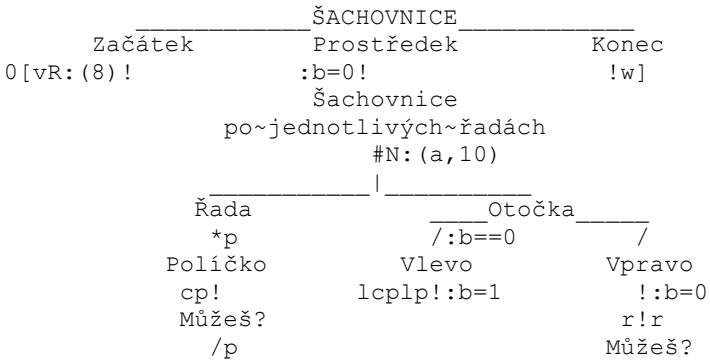
Řešení 2:



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:



```
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[vR:(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[vR:(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[vR:(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

Poslední úpravy: 31. ledna 1998

Vaše připomínky a náměty nám laskavě zasílejte na adresu:

SGP Systems, s.r.o.
L. Janáčka 180
686 01 Uherské Hradiště
Česká republika

telefon : 0632/551089, 554432
tel./fax : 0632/551089
e-mail : sgp@sgp-systems.com
url : http://www.sgp-systems.com