

ÚVOD

Následující text je určen pro prvotní seznámení se s programem OZOGAN KLONDAIK. Podrobná nápověda včetně popisu vnitřního jazyka je dostupná v programu ve formě nápovědy.

Pro ty, kteří dosud nikdy neprogramovali na počítači se může zdát pojem programování záhadný a tajemný. Jedná se však o zcela běžnou věc, kterou vlastně všichni známe již od dětských let. Určitě jste si již mnohokrát řekli: "ráno půjdu do školy (práce), odpoledne na koupališti a večer se budu dívat na televizi". Tím jste si sestavili program dne. Definovali jste posloupnost akcí, které provedete. Můžete si však také stanovit podmínky programu. Například pokud bude odpoledne hezky, půjdete na koupališti, jinak (bude přšet) si budete jíst nebo se učit. V některých případech si můžete stanovit opakování některých akcí. Například že se budete učit tak dlouho, dokud to nepochopíte. Jak vidíte, programovat už umíte. Sestavujete si program dne, výuky, dovolené. Nyní už zbývá pouze naučit se převést program do podoby, kterou zvládne i počítač.

Počítačový program představuje definici posloupnosti akcí, podmínky jejich provedení a opakování. Aby byl počítač schopen požadované činnosti provádět, musíte mu je zadat v podobě, které on rozumí. Pokud budete chtít angličtinou říct svůj program dne, budete mu jej muset přeložit do angličtiny. Stejně tak musíte přeložit program ze slovní podoby do počítačového jazyka. Počítač potom bude jíst jednu řádku programu za druhou a vykonávat to, co jste mu zadali.

V dalších lekcích bude popsáno, jakým způsobem budete s počítačem komunikovat, jak zapsat program. Naučíte se také základům použitého počítačového jazyka.

1.lekce

Instalace, spuštění a ukončení programu

Postup instalace programu záleží na verzi programu, kterou máte k dispozici a na použitém zdrojovém médiu. Demoverze, kterou máte možnost získat na internetu je šířena ve zkomprimovaném formátu v souboru KLONDAIK.ZIP a stačí ji pouze "rozbalit" do libovolného adresáře na disku počítače. V tomto případě se nezakládá programová skupina ve Windows a spuštění programu je nutné provést spuštěním souboru KLONDAIK.EXE. Demoverze zabírá na disku asi 1 Mb prostoru.

Demoverze šířená na disketách i plná verze programu obsahuje již instalaci programu. Program se dodává na jedné disketě formátu 3,5" HD. Instalace se spustí z diskety příkazem INSTALL.EXE. Zobrazí se vám řídicí instalační panel, ve kterém musíte zadat nejprve adresář pro instalaci programu. Standardně je pro instalaci nastaven adresář C:\OZOKLOND\ pro demoverzi, případně C:\OZOKLON2 pro plnou verzi programu. Cílový disk i adresář je možné změnit, přesto však doporučujeme zachovat předdefinované jméno adresáře. Kopírování souborů do zadaného adresáře se zahájí po stisku tlačítka "Instalace". Program zabírá na disku asi 1 Mb prostoru. Po zkopírování souborů je nabídnuta možnost založení skupiny programů Windows. Pokud budete souhlasit, založí se skupina OZOGAN, do které se nadefinuje spuštění programu KLONDAIK, případně prohlížení helpu k programu.

Pro rozlišení přesné definice programu OZOGAN KLONDAIK od editovaného a laděného programu bude v následujícím textu u všech lekcí uváděn program OZOGAN KLONDAIK jako systém a pod pojmem program se bude rozumět laděný a editovaný uživatelův program.

Systém se startuje po dokončené instalaci spuštěním souboru OZOKLOND.EXE (demoverze), případně OZOKLON2.EXE (plná verze programu). Při prvním spuštění systému je integrované uživatelské prostředí nastaveno do standardního stavu.

Program je možné ukončit buď volbou z menu Soubor/Ukončit program, případně ikonou z toolbaru. Pokud je editován program a změny nebyly dosud uloženy, budete dotázáni systémem na uložení změn. Při ukončení programu se provede uložení rozložení oken integrovaného prostředí do inicializačního souboru OZOKLON*.INI (dle druhu verze programu), který se nachází v adresáři Windows. Při dalším spuštění programu bude potom obnoven stav rozložení jednotlivých oken na pracovní ploše systému.

2.lekce

Základy ovládání programu

Systém OZOGAN KLONDAIK je aplikace Windows a proto je ovládání programu podřízeno tomuto standardu. Systém je ovládán z menu, často používané volby jsou přitom přístupné i z toolbaru pomocí ikon, případně pomocí funkčních kláves. Veškeré akce probíhají v několika oknech s přesně definovaným určením. Celá práce se systémem probíhá v několika režimech dle kterých jsou využívána příslušná okna. Okna lze na ploše obrazovky uspořádat podle aktuální potřeby. Okna je možné minimalizovat, nelze je však žádným způsobem zrušit.

Nejčastěji používané okno je okno s programem, které obsahuje vyvíjený a laděný program. V dalším, příkazovém okně je možné zadávat přímo příkazy jazyka bez nutnosti spuštění programu. Výsledky programu jsou zobrazovány buď v textovém nebo grafickém výstupním okně. Při krokování programu je možné zobrazovat stav proměnných v okně sledování proměnných. Pomocnou funkcí má okno text, ve kterém je možné zobrazit libovolný textový soubor. Některá z oken mají svou vlastní sadu ikon umístěných v toolbaru, který se zobrazí po aktivaci okna v jeho horní liště.

Okna můžete po pracovní ploše libovolně přepínat, posouvat, zvětšovat je či zmenšovat. Přepínání oken je možné kliknutím na libovolnou viditelnou část okna. Po kliknutí se stane příslušné okno aktivní. Pokud není okno viditelné, použijte menu Okna, kde zvolíte požadované okno. V dalších lekcích se naučíte, že je možné zviditelnit požadované okno i příkazem z programu. Kliknutím na příslušnou ikonu v pravém horním rohu okna je možné okno maximalizovat, případně minimalizovat. Okno se přesouvá nejlépe myší, kdy je uchopíte za horní lištu a přesunete na požadované místo. Změny velikosti okna dosáhnete po uchopení pravého dolního rohu okna a nastavení na požadovanou velikost.

Vždy při spuštění systému je nastaven editační režim, ve kterém je možné editovat vytvářený program, případně zadávat příkazy z příkazového okna. Po spuštění editovaného programu je nastaven provozní režim, ve kterém je provozována laděná aplikace. Zvláštním druhem provozního režimu je potom ladící režim, ve kterém lze krokovat program a animací režim,

při kterém je možné zobrazovat průběh činnosti programu. Každý režim má svá specifika, která budou popsána v následujících lekcích.

Systém obsahuje kontextovou nápovědu, která je dostupná v několika stupních. Při výběru volby z menu je krátká nápověda k vybrané volbě zobrazována ve stavovém řádku v dolní části systému. Tzv. bublinová nápověda je zobrazována po najetí myši na ikony v toolbaru. Nejobsáhlejší nápověda se zobrazí po stisku klávesy F1. Jedná se o hypertextový dokument s provázanými odkazy na zvolená hesla. Zde uváděné příklady programů máte možnost si do systému z nápovědy přetáhnout pomocí blokového přesunu přes schránku Windows. To provedete tak, že najedete v nápovědě myši na začátek ukázky programu, stisknete levé tlačítko myši, držíte jej stisknuté a přetáhnete myš na konec textu příkladu. Tam tlačítko myši uvolníte. Stiskem kláves Ctrl+C se převede takto označený blok textu do schránky Windows. Nyní se přepnete do okna s programem a stiskem kláves Ctrl+V převedeme obsah schránky.

K ovládání programu je vhodné používat myš. Většina voleb je sice dostupná současně z menu i myši, přesto je však ovládání myši mnohem příjemnější a operativnější. Pokud se v dalším textu používá myš, předpokládá se levé tlačítko.

3. lekce

Práce s příkazovým a textovým výstupním oknem

Nejprve si ukážeme, jak se dají velmi jednoduše zadávat systému příkazy bez zápisu programu. Najděte na pracovní ploše okno nazvané příkazy a udeřte je aktivním - klikněte na něj myši. Pokud není okno viditelné, případně je minimalizované, můžete zvolit z menu volbu Okna/Příkazy nebo stiskem kláves Ctrl+F2. Upravte myši velikost okna tak, aby zabíralo levou polovinu obrazovky. Obdobným způsobem nastavte okno Výstup-text do pravé poloviny obrazovky. Přepněte ze zpět do příkazového okna.

Okno příkazy se podobá textovému editoru. Má však velmi důležitou vlastnost. Pokud napíšete v okně libovolný text a stisknete klávesu ENTER, pokusí se systém ihned napsaný řádek interpretovat jako známý příkaz. To znamená, že systém si přečte text řádku, na kterém stál kurzor a zjišťuje, zda se jedná o jemu známý povel nebo příkaz. Pokud ano, povel nebo příkaz provede. Proto se toto okno nazývá příkazové.

Příkazy a povely jsou přitom pro systém instrukce, co má vykonat. Příkazem je většinou označen prvním hlavní povel systému. Příkazy jsou v programech psány velkými písmeny. Jako povely jsou označovány doplňkové příkazy, které jsou zabudovány jako podpůrné akce. Jsou psány malými písmeny s velkými písmeny na počátku slova. Pokud jsou názvy povelů uvedeny z více slov, nesmí být mezi nimi uvedeny mezery.

Napište nyní v příkazovém okně slovo BEEP a stiskněte klávesu ENTER. Systém zjistí, že příkaz BEEP je pokyn, pro pípnutí. Zajistí proto, že reproduktor poěítače krátce pípne. Takto systém interpretoval vámi zadaný příkaz do podoby srozumitelné pro poěítač. Interpretace proto znamená převedení příkazu nebo řádku programu do formy srozumitelné pro zařízení poěítače (obrazovka, tiskárna ...). Takto je v systému OZOGAN KLONDAIK interpretován každý řádek programu. Pokud je některý řádek několikrát opakován, provádí se i opakování jeho interpretace. Takovým systémům se proto říká interprety. Jiné systémy překládají zdrojový

program pouze jednou přímo do spustitelné podoby. To jsou kompilátory a bývají rychlejší. Neumožňují ale interaktivní práci uživatele se systémem. Proto jsou pro začátky programování vhodnější interprety, které vám umožní lépe a rychleji proniknout do tajů programování.

V příkazovém okně máte možnost vyzkoušet si mnoho příkazů a povelů systému OZOGAN KLONDAIK, aniž napíšete jedinou řádku programu. Pokud uděláte chybu, systém vás na to upozorní a vy máte možnost ihned chybu opravit bez nutnosti kompilace programu. Zapomenete-li například napsat ve výše uvedeném příkazu BEEP jedno 'E', oznámí vám systém, že uvedený příkaz nezná. Najedete proto kurzorem na chybné místo, provedete opravu a po stisku klávesy ENTER se již příkaz vykoná.

Jistě jste si již všimli, že klávesa ENTER neukončí na místě kurzoru řádek, jak to bývá u textových editorů. Řádek zůstane celý zachován a kurzor se automaticky přesune na začátek prázdného řádku za předchozí příkazy. Pokud zkusíte zapsat příkaz

```
WRITELN("KLONDAIK");
```

vypíše se do textového výstupního okna text KLONDAIK. Možnostmi příkazu WRITELN je vlnována následující lekce. Nyní bude stačit, pokud si zapamatujete, že příkaz WRITELN v uvedeném tvaru vypíše obsah textu uvedeného mezi uvozovkami do textového výstupního okna. Mezi uvozovky můžete zkusit přitom zadat libovolný jiný text. Zkuste text mezi uvozovkami několikrát zmínit a změny vždy odešlete klávesou ENTER. Jistě si všimnete, že původní text před opravou zůstane v příkazovém okně zachován a příkaz s novým textem bude doplněn na konec příkazového okna. Díky tomu máte vždy přehled o historii zadávaných příkazů. Neprovedete-li v textu žádnou změnu a příkaz opakovaně odešlete, nebude stejný příkaz zadávaný bez změny za sebou v historii příkazového okna za sebou znovu opakován.

Historii zadávaných příkazů můžete libovolně používat bez ohledu na pořadí původního zadávání. Pokud provedete v některém řádku příkazového okna libovolnou změnu a následně stisknete klávesu ESC, příkaz se neprovede a text upraveného řádku bude uveden do původního stavu. Příkazy vyzkoušené v příkazovém okně máte možnost přes schránku Windows následně převést do programu.

Různými pokusy s příkazovým oknem jste dosáhli toho, že máte ve výstupním textovém okně uvedeny různé texty. S každým použitím příkazu WRITELN přibude na konec okna jeden řádek. To připomíná použití psacího stroje, kdy také není možné se vracet zpět. V dávných dobách prehistorie výpočetní techniky sálových počítačů se podobné psací stroje používaly pro komunikaci s počítačem. Tehdy se jim říkalo konzole - anglicky console. Uvedené anglické označení je proto také používáno pro příkazy přímo související s obsluhou textového výstupního okna. Pokud například zkusíte zadat v příkazovém okně příkaz ConsoleClear, zjistíte, že zruší obsah textového výstupního okna. Obdobného efektu dosáhnete, pokud se přepnete do textového výstupního okna a stisknete v podřízeném toolbaru uvedeného okna ikonu znázorňující prázdnou stránku. Po kontrolním dotazu bude obsah textového výstupního okna zrušen.

V následující kapitole se seznámíte, jak vypisovat v textovém výstupním okně za použití příkazu WRITELN i jiné informace a hodnoty.

4.lekce

Příkaz WRITELN, matematické výpočty

V předchozí kapitole jste se naučili zadávat z příkazového okna příkazy umožňující vypsát do textového výstupního okna znaky. Nyní si ukážeme, jak vypisovat čísla a aritmetické výpočty.

S příkazem WRITELN jste již měli možnost se krátce seznámit. Používá se na výpis hodnot. Požadovaná hodnota se přitom uvádí v závorce. Pokud se u příkazu WRITELN uvedou pouze prázdné závorky, nebude nic vypsáno a další výstup do textového výstupního okna bude prováděn na novou řádku. WRITELN je pouze příkaz který uvádí systému, že má něco vypsát. Uvedené něco se přitom uvádí v závorce jako parametr příkazu. Pokud uvedete znaky v uvozovkách nebo apostrofech, převedou se uvedené znaky do textového výstupního okna. Pokud by jste chtěli vypsát číslo, musíte zadat příkaz v následujícím tvaru:

WRITELN(10);

Stačí tedy napsat jako parametr příkazu do závorek přímo požadované číslo. Číslo se vypíše se zarovnáním od levého kraje okna. Pokud budete chtít ponechat zleva odsazené místo, budete muset zadat požadovanou délku vypsání čísla v počtech znaků:

WRITELN(10:5)

Délka vypsání čísla je v tomto případě pět znaků. To je parametr uvedený v příkazu za dvojtečkou a udává počet míst zobrazení čísla. Nejedná se proto o výpočet dělení. Výpočet je ale samozřejmě také možné zadat. V tomto případě by jste pro dělení museli zadat:

WRITELN(10/5:5)

Jako výsledek bude zobrazeno číslo 2 o délce pěti znaků. Zkuste si sami zadat součet, rozdíl, případně násobek čísel:

WRITELN(10+5:5);

WRITELN(10-5:5);

WRITELN(10*5:5);

Zobrazí se vždy výsledek aritmetického výrazu o délce pěti znaků. Překvapení budete možná pokud zadáte příkaz:

WRITELN(10/3:5);

Sami jistě poznáte, že něco není v pořádku. Ztratila se desetinná část výsledku. Až dosud jsme pracovali pouze s celými čísly. Výsledky se také vždy zobrazují standardně jako celá čísla. Zkuste nyní následující příkaz:

WRITELN(10/3:5:2);

Zobrazí se již výsledek s uvedením dvou desetinných míst. Je to proto, že jsme za dvojtečkou uvedli délku vypisovaného čísla a současně za další dvojtečkou počet zobrazených desetinných míst. Zkuste si sami další příklady a seznamte se zadáváním uvedených parametrů.

S možností výpisu nenumerických údajů, tedy znaků jste se již seznámili v předchozí lekci. Seskupení několika znaků se přitom nazývá šetizec a také tak již bude v následujícím textu uváděno. Pokud by jste chtěli jedním příkazem WRITELN vypsat najednou šetizec i znaky, můžete tak učinit uvedením několika parametrů oddělených čárkou:

WRITELN('Výsledek výpočtu je:', 10/3:5:2);

Stejně tak můžete vypsat samozřejmě i několik čísel nebo šetizců oddělených v příkazu čárkami. To vám dovolí ve spojení s možností uvádění délky čísel vhodnou grafickou úpravu textu:

WRITELN('Seznam výsledků:', 12/3:5:2, 45-26:5:2, 3*3:5:2);

Příkaz WRITELN provede po svém ukončení vždy přechod na novou řádku. Pokud však budete potřebovat, aby další výpis pokračoval na stejné řádce, můžete použít příkazu WRITE. Ten neprovádí ukončení řádku, následující výpis je zahájen od pozice ukončení příkazu WRITE. Možnosti a parametry příkazu WRITE jsou přitom totožné jako pro uváděný popis příkazu WRITELN.

Výše uvedený popis použití příkazu WRITELN se vám může zdát na první pohled složitý. Je však nutné jej přesně dodržet. Systém si hlídá důsledně jeho dodržování a v případě chyby odmítne příkaz vykonat. Přesná definice jakéhokoliv používaného jazyka se nazývá syntaktická pravidla. Pokud zadá uživatel chybný zápis, dojde k porušení syntaxe a odmítnutí systému k vykonání chybného zadání. Byla proto sestavena pravidla pro použití všech příkazů. Uvádíme zde pro představu zkrácenou definici syntaxe zápisu příkazu WRITELN tak, jak byla výše popsána:

WRITELN([výraz [:délka] [:desetiny]]) [;]

V zápise znamená výraz libovolný výraz, Například šetizec, číslo nebo výpočet. Údaje v hranatých závorkách se nemusí uvádět. Pokud proto uvedeme dvojtečku následovanou číslem, jedná se o délku vypisovaného čísla. Případná další dvojtečka s číslem uvádí počet desetinných míst. Pokud si dobře prohlédnete uvedený syntaktický zápis zjistíte, že mezi závorkami nemusíte nic uvádět. To je dáno levou hranatou závorkou uvedenou ihned za levou kulatou závorkou. Ukončení této volitelné části je provedeno pravou hranatou závorkou před pravou kulatou závorkou. Pokud proto uvedete příkaz:

WRITELN();

nevypíše se žádný text, dojde ale k vynechání prázdného řádku. Středník uvedený v

hranatých závorkách na konci příkazu není nutné uvádět. V popisu syntaxe je uveden, protože v klasickém jazyku Pascal musí být každý řádek programu ukončen středníkem. Uvedený způsob popisu syntaxe je použit u popisu všech příkazů a knihoven procedur a funkcí uvedených v manuálu i helpu k programu.

Pokud jste se dokonale seznámili s použitím příkazu WRITELN, můžete ve spolupráci s manuálem k programu vyzkoušet některé funkce matematické knihovny. Dále uvedené příklady uvádí na konci řádku mezi složenými závorkami komentář, které nemají na výsledek žádný vliv. Ve skutečnosti je proto nemusíte větně složených závorek uvádět.

```
WRITELN(Abs(-55));    {absolutní hodnota čísla}  
WRITELN(Cos(PI));   {kosinus Ludolfova čísla PI}  
WRITELN(Max(3,10)); {maximální hodnota zadaných čísel}  
WRITELN(Min(3,10)); {minimální hodnota zadaných čísel}  
WRITELN(Random(500)); {náhodné číslo do 500}  
WRITELN(Round(12.82)); {zaokrouhluje číslo}  
WRITELN(Sqr(5));    {vrací druhou mocninu čísla}  
WRITELN(Sqrt(16));  {vrací druhou odmocninu čísla}  
WRITELN(Trunc(12.82)); {odřízne desetinnou část čísla}
```

Jak vidíte, můžete použít příkaz WRITELN jako docela chytrou kalkulačku. V současnosti při použití pouze příkazového okna má ale jednu dosti velkou vadu. Nemá žádnou paměť, do které by jste si zaznamenali hodnoty mezivýsledků. V některé z dalších lekcí, kdy budeme probírat deklaraci proměnných v programu se ale naučíme i to.

V následujících lekcích se seznámíte s dalším výstupním oknem, které se používá pro výstup a kreslení grafiky.

5.lekce

Grafické výstupní okno, grafický editor

V předchozích kapitolách jsme se naučili používat příkazové okno a textové výstupní okno. Systém ale obsahuje i další, velmi zajímavé výstupní okno. Je to grafické výstupní okno, které vám umožní do své plochy libovolně kreslit.

Uzavřete dříve používané textové výstupní okno. Můžete tak učinit kliknutím na uzavírací ikonu, případně můžete zadat v příkazovém okně příkaz ConsoleHide. Dále aktivujte grafické výstupní okno. To lze provést z menu volbou Okna/výstup grafika. Okno je možné také zobrazit příkazem ImageShow. Zobrazené okno upravte opět tak, aby pokrývalo pravou polovinu obrazovky.

Jak vidíte, každé okno je pro účely ovládní z programu pojmenováno významově dle anglického názvosloví. Existují přitom příkazy pro zobrazení (anglicky show) a ukrytí (anglicky hide) okna. Možná se pozastavujete nad tím, že u českého programu se používá cizojazyčných termínů. Je to proto, že systém OZOGAN KLONDAIK může sloužit jako nástroj pro výuku programování. Proto je podle nás vhodné si již od počátku zvyknout na terminologii, která se při běžném programování používá. Názvy procedur, funkcí a konstant přitom vychází z používané terminologie jazyků Pascal a získané znalosti jistě dále

zužitkujete. Použitá počítačová angličtina je přitom velmi jednoduchá a neměla by nikomu dělat problémy.

Po aktivování grafického výstupního okna se vám v jeho horní lišti zobrazí řada ikon pro ovládání zabudovaného grafického editoru a ikony pro možnost kreslení na grafické ploše. Ikony jsou umístěny do tří skupin. Levá skupina se používá pro načítání a ukládání obrázků v grafickém formátu *.BMP. Střední skupina slouží pro definici kresleného tvaru a v pravé skupině naleznete nastavení typu čáry a výplně ploch.

Nejprve se seznámíme s možností kreslení základních geometrických tvarů. Ve střední skupině ikon jsou seskupena tlačítka pro čáru, obdélník, kružnici a obdélník se zakulacenými hranami. Klikněte nejprve na tlačítko s čarou. Tím jste zadali, že budete chtít kreslit čáry. Uvidíte, že tlačítko zůstalo stisknuto. Přesuňte ukazatel myši na grafickou plochu. Pokud nyní na grafické ploše stisknete tlačítko myši, podržíte jej a přesunete na novou pozici, bude se kreslit čára z bodu stisku tlačítka do aktuální pozice ukazatele myši. Po uvolnění tlačítka zůstane čára zachována. Zkuste si nakreslit i vodorovné a svislé čáry. Obdobným způsobem lze na grafickou plochu kreslit obdélníky, kružnice a obdélníky se zakulacenými rohy. Pro zvolení nového tvaru musíte stisknout příslušné tlačítko na lišti s ikonami.

Čáry se kreslí černou barvou a tenkou čarou. Pokud budete chtít změnit barvu čáry, nebo její tloušťku, stiskněte ikonu v pravé horní části okna s vyobrazením tužky. Zobrazí se vám další řada ikon pro zadávání barvy, typu a tloušťky čáry. Po opětovném stisku ikony s tužkou se nastavení skryje. Ikona tedy pracuje jako přepínací tlačítko. Klikněte si proto na tlačítko tak, aby jste měli zobrazeny ikony pro nastavení čar. Vlevo je umístěna tabulka barev, uprostřed tlačítka pro výběr typu čáry a vpravo můžete zadat tloušťku čáry. Barvy se vybírají kliknutím na požadovanou barvu. Vybraná barva je označena dvoupísmennou anglickou zkratkou barvy. Typ čáry můžete vybrat plnou čarou, tečkovanou, čárkovanou, čerchovanou nebo můžete posledním tlačítkem úplně zrušit. Tloušťka čáry se udává v bodech. Pro změnu tloušťky můžete zapsat do editačního boxu přímo novou hodnotu, nebo můžete použít pro nastavení šipek. Pokud bude nastavena tloušťka čáry větší než jedna, neuplatní se zadaný typ čáry a čára se bude vykreslovat vždy plná, případně se nebude kreslit vůbec. Zkuste si nastavit parametry čáry a prověřte si účinky změny při kreslení základních geometrických tvarů.

Dosud jsme kreslili pouze okraje geometrických tvarů. Standardně je totiž nastaveno, že se plocha kreslených geometrických tvarů nevykresluje. Podobným způsobem, jako se nastavují parametry kreslení čar máte možnost nastavit parametry vykreslování ploch. Ikony pro nastavení se zobrazí po stisku tlačítka s vyobrazením štitce. Opět máte možnost zadat barvu, tentokrát plochy (výplně). Současně můžete zadat typ výplně. Máte možnost vybrat si buď kreslení plné plochy zadanou barvou, nevykreslování plochy, nebo z několika druhů vykreslení plochy čarami. Lze vybrat čáry vodorovné, svislé, vodorovné i svislé současně (mřížka) a různé druhy diagonálních čar. Tloušťka čáry výplně je přitom vždy jeden bod.

Nyní jste již schopni nakreslit na grafické ploše pouze za použití myši jednoduché obrázky. Vyzkoušejte si různé možnosti nastavení čáry a plochy. Parametry nastavení čar a plochy je možné změnit samozřejmě také přímo z programu, případně zadat z příkazového okna. To si však ukážeme až v následujících lekcích. Nyní bude pro vás jistě zajímavá možnost uložení nakreslených obrázků na disk do souboru pro pozdější použití a zpitná

možnost načtení obrázku z disku do grafického okna pro provedení úprav. Ukážeme si také, jak je možné obrázek vytisknout.

Pokud máte grafickou plochu zaplněnou předchozími pokusy, můžete provést její výmaz pomocí ikony s obrázkem prázdné stránky. Ikona se nachází v levé skupině ikon umístěné v grafickém výstupním okně. Po kontrolním dotazu bude grafická plocha vymazána. Stejného efektu lze dosáhnout povelom ImageClear zadaným v příkazovém okně. Povel se zadává bez parametrů, v tomto případě se provede výmaz grafické plochy již bez kontrolního dotazu.

Nakreslené obrázky máte samozřejmě možnost uložit do souboru. Používá se známý a běžný bitmapový soubor typu *.BMP. Pro uložení stisknete ikonu s obrázkem diskety. V dialogovém okně zadejte adresář a jméno souboru. Obrázek můžete uložit samozřejmě i povelom ImageSave z příkazového okna. Například:

```
ImageSave("obrazek.bmp")
```

Obrázky můžete i zpět načíst. Použijte ikonu s obrázkem šipky směřující do stránky. V dialogovém okně vyberte adresář a jméno souboru typu *.BMP. Načíst můžete i obrázky vytvořené v jiných grafických systémech. Po načtení obrázku je velikost grafické plochy nastavena dle načítaného obrázku. Obrázek můžete načíst i povelom z příkazového okna ImageLoad. Například:

```
ImageLoad("obrazek.bmp")
```

Pokud by jste chtěli vytvořený obrázek vytisknout, můžete tak učinit pomocí ikony zobrazující tiskárnu. Pro tisk obrázku z příkazového okna můžete použít povel ImagePrint.

V následující lekci se seznámíte s možnostmi ovládání grafického výstupního okna pomocí povelů z příkazového okna.

6.lekce

Ovládání grafického okna z příkazového okna

Předchozí lekce vás seznámila krátce se základními možnostmi použití grafického výstupního okna. Naučili jste se kreslit do grafické plochy pomocí myši, ukládat, načítat a vytisknout vytvořený obrázek. Nyní se seznámíte s dalšími možnostmi grafického okna a s parametry okna, které budete potřebovat pro programování grafických povelů. Poznáte, že vše, co bylo možné nastavit v grafickém okně budete mít možnost zadat pomocí povelu z příkazového okna.

Činnost grafického okna je možné si představit jako malířské plátno definovaných rozměrů, na které se kreslí perem (anglicky pen). Větší plochy je možné vybarvit štětcem (anglicky brush). Kreslí se přitom vždy nastavenou barvou. Systém obsahuje povely pro nastavení parametrů pera i štětce. S možnostmi se seznámíte v následujícím textu.

Při kreslení obrazců do grafického okna se kreslí obrazce čárou, jejíž typ je definován povelom ImagePenStyle a tloušťka čáry je definována povelom ImagePenWidth. Plocha

nakreslených geometrických obrazců je vyplněna stylem zadaným povelu ImageBrushStyle.

Nejjednodušší je změna tloušky čáry, kdy uvádíte přímo jako parametr povelu tloušku čáry v bodech. Pro změnu tloušky čáry na pět bodů použijete z příkazového okna následující povel:

ImagePenWidth(5);

Všimněte si, že pokud nastavíte sílu čáry povelu z příkazového okna, použije se nastavená síla čáry i pro následné kreslení pomocí myši přímo v grafickém okně. Obdobná vlastnost je platná i pro nastavení všech parametrů grafického okna. Máte proto možnost libovolně kombinovat zadávání povelů z příkazového okna nebo jejich nastavení pomocí ikon. V budoucnu budete mít samozřejmě možnost uvedené parametry nastavit i přímo z programu.

Styl čáry máte možnost zadávat povelu ImagePenStyle. Předdefinováno je šest stylů. Jako parametr povelu musíte přitom zadat definovaný styl čáry. Parametr můžete uvést buď číselnou hodnotou, nebo jménem konstanty dle následující tabulky:

hodnota	konstanta	název stylu
1	psSolid	souvislá čára
2	psDash	přerušovaná čára
3	psDot	tečkovaná čára
4	psDashDot	čárkovaná čára
5	psDashDotDot	čárkovaná čára se dvěma tečkami
6	psClear	neviditelná čára

Příklady použití definice stylu čáry (text mezi složenými závorkami nemusíte psát, jedná se o poznámku):

```
ImagePenStyle(1);    { souvislá plná čára }  
ImagePenStyle(3);    { tečkovaná čára }  
ImagePenStyle(6);    { neviditelná čára }
```

```
ImagePenStyle(psSolid); { souvislá plná čára }  
ImagePenStyle(psDot);   { tečkovaná čára }  
ImagePenStyle(psClear); { neviditelná čára }
```

Existuje samozřejmě i způsob nastavení barvy čáry. Nejprve se však budeme muset seznámit s možnostmi použití barev v počítači. Zobrazování barev závisí na vlastnostech videokarty ve vašem počítači. Používáte-li barevnou VGA kartu, máte možnost zobrazit minimálně 16 barev. Po příslušném nastavení videoadaptéru je možné běžně zobrazovat 256 barev, výjimečně i více. Vy budete mít možnost nastavit v systému libovolnou barvu z rozsahu 16 miliónů barev. Skutečně zobrazená barva ale závisí na vlastnostech technického zařízení, protože se zobrazí vždy barva nejbližší.

Hodnotu barev je možné zadávat dvěma způsoby, které lze v programu libovolně kombinovat. Pokud budete používat pouze základní, šestnáctibarevnou paletu, můžete tak

uèinit zadáváním pøeddefinované konstanty udávající anglické jméno barvy. Stejnou barvu máte možnost zadat i pomocí tzv. RGB hodnoty.

Hodnota barev zadávaná definicí RGB znamená, že každá barva je definována jako poměr kombinace barev modré, zelené a èervené. Pro každou barvu je možné volit hodnoty v rozsahu 0 až 256. Násobek těchto hodnot (modrá x zelená x èervená) udává výslednou barvu. Výhodné je používat tzv. hexadecimálního zápisu, kdy jsou pro každou barvu vyhrazeny dvě pozice èíslo s hodnotami od 00 (èíslo 0) až do FF (èíslo 256). Při použití hexadecimálního èíslo je nutné uvést pøed èíslem rozlišovací znak \$. Viz tabulka hodnot barev.

hodnota	konstanta	název barvy
\$000000	clBlack	èerná
\$000080	clMaroon	kaštanová èervená
\$0000FF	clRed	svìtle èervená
\$008000	clGreen	tmavì zelená
\$008080	clOlive	tmavì žlutá
\$00FF00	clLime	svìtle zelená
\$00FFFF	clYellow	žlutá
\$800000	clNavy	tmavì modrá
\$800080	clPurple	tmavì fialová
\$808000	clTeal	tmavì modrozelená
\$808080	clDkGray	tmavošedá
\$C0C0C0	clLtGray	svìtle šedá
\$FF0000	clBlue	modrá
\$FF00FF	clFuchsia	fialová
\$FFFF00	clAqua	modrozelená
\$FFFFFF	clWhite	bílá

Pøíklady použití definice barvy èáry (text mezi složenými závorkami nemusíte psát, jedná se o poznámku):

```
ImagePenColor(clWhite); { bílá barva èáry }  
ImagePenColor(clRed); { svìtle èervená barva èáry }  
ImagePenColor(clBlue); { modrá barva èáry }
```

```
ImagePenColor($FFFFFF); { bílá barva èáry }  
ImagePenColor($0000FF); { svìtle èervená barva èáry }  
ImagePenColor($FF0000); { modrá barva èáry }
```

Podobnì, jako je možné nastavit parametry kreslené èáry je možné nastavit parametry vykreslovaných ploch geometrických obrazcù. Barva plochy se pøitom zadává povelom ImageBrushColor. Používá se pøitom výše zadaných hodnot a konstant pro definici parametru barvy. Pøíklady použití definice barvy plochy (text mezi složenými závorkami nemusíte psát, jedná se o poznámku):

```
ImageBrushColor(clWhite); { bílá barva plochy }  
ImageBrushColor(clRed); { svìtle èervená barva plochy }  
ImageBrushColor(clBlue); { modrá barva plochy }
```

```
ImageBrushColor($FFFFFF); { bílá barva plochy }
ImageBrushColor($0000FF); { světle červená barva plochy }
ImageBrushColor($FF0000); { modrá barva plochy }
```

Pro nastavení stylu vyplňování ploch se používá povel ImageBrushStyle, kde se jako parametr povelu udává buď číslem definovaný styl, nebo jméno konstanty dle následující tabulky:

hodnota	konstanta	název stylu
1	bsSolid	vyplní oblast jednou barvou
2	bsClear	vyplní oblast barvou pozadí
3	bsHorizontal	vyplní oblast vodorovnými čarami
4	bsVertical	vyplní oblast svislými čarami
5	bsFDiagonal	diagonální čáry \\\\\\\
6	bsBDiagonal	diagonální čáry /////
7	bsCros	vodorovné a svislé čáry
8	bsDiagCross	vodorovné a svislé čáry diagonální

Příklady použití definice stylu plochy (text mezi složenými závorkami nemusíte psát, jedná se o poznámku):

```
ImageBrushStyle(1); {plné vybarvení plochy}
ImageBrushStyle(3); {výplň vodorovnými čarami}
ImageBrushStyle(8); {výplň diagonálními čarami}
```

```
ImageBrushStyle(bsSolid); {plné vybarvení plochy}
ImageBrushStyle(bsHorizontal);{výplň vodorovnými čarami}
ImageBrushStyle(bsDiagCros); {výplň diagonálními čarami}
```

Aby bylo možné kreslit v grafickém okně do přesně určených pozic, musí být zadány souřadnice pro kreslení. Souřadnice určují polohu jednotlivých bodů kresby. Souřadnice znamená, že musíte uvést vzdálenost v bodech od levého okraje grafického okna a vzdálenost v bodech od horního okraje grafického okna. Souřadnicový systém je tedy vztažen k levému hornímu rohu, který má souřadnici 0,0. Hodnoty ve směru osy X narůstají směrem doprava, hodnoty ve směru osy Y narůstají směrem dolů. Při zápisu souřadnice se uvádí nejprve osa x, potom osa y. Je přitom možné zadávat příkazy pro kreslení mimo plochu grafického okna, zobrazí se však pouze ta část, která je obsažena maximálními souřadnicemi grafického okna.

Aktuální souřadnice se zobrazují ve stavovém řádku systému vždy, když máte nastavenou myš nad grafickým oknem. Souřadnice je vhodné si vyzkoušet také na povelu Point, který slouží pro zobrazení bodu na zadané souřadnici. Povel nakreslí na zadaných souřadnicích bod o zadané velikosti. Bod se nakreslí aktuální barvou pera, kterou lze nastavit procedurou ImagePenColor. Vyzkoušejte si několik příkladů pro seznámení se se způsobem označování souřadnic grafické polohy:

```
Point( 0, 0, 2); { levý horní roh }
```

```
Point( 0, 100, 2); { levý dolní roh }
Point(100, 0, 2); { pravý horní roh }
Point(100, 100, 2); { pravý dolní roh }
Point( 50, 50, 10); { uprostřed, větší bod }
```

Souřadnice grafického okna se použijí i pro kreslení geometrických tvarů pomocí povelů. Možné je kreslit čáru povelem Line, obdélník povelem Rectangle, kružnici nebo elipsu povelem Ellipse a obdélník se zaoblenými rohy povelem RoundRect. Povel Triangle je možné nakreslit trojúhelník, což není pomocí myši možné. Pomocí povelů je možné také kreslit povel Arc část křivky a povel Pie kruhovou výseč. Vyzkoušejte si kreslení základních geometrických tvarů:

```
Line(20, 20, 50, 100); { nakreslí čáru }
Rectangle(10, 10, 100, 100); { nakreslí čtverec }
Ellipse(30, 30, 120, 120); { nakreslí kružnici }
Ellipse(30, 30, 120, 60); { nakreslí elipsu }
Triangle(10,100,55,10,100,100); { nakreslí trojúhelník }
```

```
Arc(0,0,100,100, 50,0,0,50); {levý horní čtvrtkruh}
Pie(0,0,100,100, 50,100,100,50) {pravý dolní čtvrtkruh}
```

7.lekce

Výstup do grafického okna

V předchozí lekci jsme se naučili kreslit do grafického výstupního okna z příkazového okna a nastavovat parametry čar a ploch. Nyní se zaměříme na nastavení grafické plochy a převod části grafického okna ze zásobníku Windows a zpět. Seznámíme se také s možností zápisu textu do grafického okna.

Až dosud jsme měli velikost grafické plochy nastavenou vždy podle toho, jak velké bylo grafické okno v okamžiku aktivace grafické plochy. Další zmínou velikosti grafického výstupního okna se již velikost grafické plochy neměnila. Pokud proto potřebujete nastavit velikost grafické plochy na požadované rozměry, můžete tak učinit povel ImageInit. Povel inicializuje grafické okno a nastaví jeho velikost na rozměry zadané parametry. Nastaví současně bílou barvu plochy, styl štítky pro plně vybarvené plochy, černou barvu pera, sílu čáry na jeden bod. Inicializací se provede výmaz původního obsahu grafického okna. Vyzkoušejte si například následující hodnoty:

```
ImageInit( 50, 200);
ImageInit(120, 60);
```

Pokud potřebujete znát parametry již inicializované grafické plochy, můžete použít dotazu GetMaxX a GetMaxY, který vrátí velikost grafické plochy zadané strany. Pokud budete například potřebovat vykreslit bod uprostřed grafické plochy bez ohledu na velikost aktuálně inicializované plochy, můžete tak učinit povel:

```
Point(GetMaxX/2, GetMaxY/2, 5);
```

V některých případech by bylo vhodné kreslit čáry do grafické plochy zadáváním v absolutních přírůstkových hodnotách místo přesné definice souřadnic. Proto je v grafickém okně definován tzv. grafický ukazatel, který zaznamenává pozici vykreslení posledního bodu. Poloha grafického ukazatele se při použití některých povelů automaticky mění. Je možné ji nastavit i z programu povelu MoveTo. Čáry je možné potom zadat povelu LineTo definicí konečného bodu. Využitím uvedených povelů je možné například nakreslit libovolný mnohostranný mnohoúhelník. Vyzkoušejte si následující povely, které by měly v grafické ploše nakreslit čtyřúhelník:

```
MoveTo( 10, 10); { přesun na počáteční bod }  
LineTo(100, 10); { horní hrana }  
LineTo(100, 100); { pravá hrana }  
LineTo(100, 10); { spodní hrana }  
LineTo( 10, 10); { levá strana }
```

Až dosud jsme důsledně dodržovali, že je možné textové informace vypisovat zásadně do textového výstupního okna a grafické informace do grafického výstupního okna. Je však možné provádět výstup textových informací i do grafického okna. Nelze přitom použít povelu WRITELN, se kterým jsme se již dříve seznámili. Zobrazovat lze pouze textové informace. Číslo je nutné předem převést na řetězec. K výstupu textových informací se používá povelu TextOut. Jako parametry povelu se udávají souřadnice pro zobrazení textu v grafickém okně a řetězec, který se má zobrazit. Povel si můžete vyzkoušet například následujícími příklady:

```
TextOut(10, 10, "OZOGAN");  
TextOut(10, 30, "KLONDAIK");
```

Text se při použití povelu TextOut vypisuje předdefinovaným fontem. Jeho změna se provede buď povelu ImageSetFont, kdy je možné zadat v dialogovém okně nejen jméno fontu, ale přímo i jeho styl a velikost. Další možností je použití ikony z grafického výstupního okna. Pokud potřebujete nastavit pouze některé atributy fontu, můžete tak učinit povelu ImageFontColor (barva fontu), ImageFontName (jméno fontu), ImageFontSize (velikost fontu) a ImageFontStyle (styl fontu).

Při nastavení barvy fontu se zadává u povelu ImageFontColor jako parametr buď číslo barvy, nebo jméno konstanty udávající barvu. Tabulka hodnot barev je stejná jako pro nastavení barvy čar a ploch. U povelu ImageFontName, který nastavuje jméno fontu se uvádí přímo jméno fontu. Pokud není zadáno jméno v systému Windows dostupné, použije se font s podobným jménem. Velikost fontu se uvádí v povelu ImageFontSize přímo požadovanou hodnotou. Příklad možných nastavení si prozkoušejte včetně výpisu textu po každé změně parametru fontu:

```
ImageFontColor(clRed); { nastaví červenou barvu fontu }  
ImageFontColor($00FFFF); { nastaví žlutou barvu fontu }  
ImageFontName("Arial CE");{ nastaví font Arial CE }  
ImageFontSize(12); { nastaví velikost fontu 12 bodů }
```

Pro nastavení stylu fontu se v povelu ImageFontStyle používají jako parametr předdefinované

hodnoty jednotlivých stylů. Uvádí se součet hodnot požadovaného výsledného stylu:

hodnota	konstanta	popis stylu
0	fsNormal	normální písmo
1	fsBold	tučné písmo
2	fsItalic	nakloněné písmo
4	fsUnderline	podtržené písmo
8	fsStrikeOut	přeškrtnuté písmo

Pro nastavení nakloněného podtrženého písma můžete zadat jednu z následujících možností:

```
ImageFontStyle(2 + 4);           {součet hodnot}
ImageFontStyle(6);              {součet 2 + 4}
ImageFontStyle(fsItalic + fsUnderline); {uvedení konstanty}
```

Pokud budete potřebovat převést nakreslený obrázek přes schránku Windows (clipboard) do jiné aplikace, můžete tak provést pomocí příkazu `ImageToClip`, kdy uvedete jako parametry souřadnice ohraničené plochy pro převod. Pokud budete chtít například přesunout do schránky obsah celého grafického okna, zadejte příkaz:

```
ImageToClip(0,0,GetMaxX,GetMaxY);
```

Načtení obsahu schránky se provádí příkazem `ImageFromClip`, kdy se jako parametry uvedou souřadnice ohraničené plochy, kam se má obsah schránky převést. Pokud budete chtít převést obsah schránky Windows na celou plochu grafického okna, zadejte příkaz:

```
ImageFromClip(0,0,GetMaxX,GetMaxY);
```

Nikoho z vás již určitě napadlo, že výše uvedené příkazy, pomocí kterých můžete převádět obrázky do schránky a načítat zpět ze schránky by bylo možné použít na přesun části obrázku na nové místo. Na to je ale výhodnější použít samostatného příkazu `ImageMove`, který provede obě akce najednou. Zadávejte přitom jako parametry souřadnice místa, odkud se má načtení provést a souřadnice, kam se má plocha přenést. Neodpovídá-li přitom poměr stran zdrojové a cílové plochy, nebude zkopírován obraz uříznut, ale bude upraven (deformován) do zadaných souřadnic. To je možné využít k mnoha zajímavým efektům. Například ke zvětšení části plochy apod.

Nikteží jste již možná netrpěliví, kdy se začne programovat. Místo programování jsme zadávali pouze příkazy do příkazového okna. Tím jsme se vás snažili naučit nejprve používat několik základních příkazů programovacího jazyka. Pokud by jsme začali ihned se základy programování včetně popisu použití příkazů bylo by toho najednou moc. Proto jsme si nejprve probrali základy používání příkazů a příkazů jazyka a v příští kapitole již začneme opravdu programovat.

8.lekce

První program, procedura main

V předchozích lekcích jsme si probrali základní informace o používání příkazového okna pro zadávání povelů jazyka přímo z klávesnice. Výstup výsledků byl prováděn do textového nebo grafického okna. V následujících lekcích využijeme získané poznatky již přímo při vývoji vlastních programů.

Programy se zadávají (zapisují) do samostatného okna, které slouží jako textový editor programu. Pokud je okno na obrazovce viditelné, můžete jej aktivovat kliknutím myši na jeho plochu. V opačném případě využijte z menu volbu Okna/Program. Pokud budete chtít použít povelu z příkazového okna, budete muset zadat povel ProgramShow. Pro možnost automatického vytvoření základní kostry programu stisknete klávesy Ctrl+N, nebo zadejte volbu menu Soubor/Nový. Do okna program se vygeneruje kostra prázdného programu.

Nový, prázdný vygenerovaný program je možné ihned spustit volbou z menu Program/Spustit, případně stiskem funkční klávesy F9 nebo kliknutím myši na ikonu s obrázkem bílého panáčka. Prázdný program obsahuje pouze základní kostru bez žádného povelu, který by vykonával viditelnou činnost a proto program skončí, aniž by cokoliv učinil. Využijeme proto získaných zkušeností a doplníme text programu o nám již známé povely a příkazy. Dávejte si pozor na to, aby jste ponechali beze změny první a poslední řádek programu. Pokud by se tak nestalo, program by se nemusel vůbec spustit. Proveďte proto pouze doplnění programu o příkaz WRITELN dle následující ukázky:

```
PROCEDURE main  
  // zde запиšte váš program  
  WRITELN("Ahoj, zdraví Vás systém KLONDAIK");  
ENDPROC
```

Program spustíme funkční klávesou F9 a pokud máte viditelný obsah textového výstupního okna, zobrazí se v něm pozdrav od systému. Vyzkoušejte si další, dříve probrané povely, které zapisujete pouze do části programu mezi slovy PROCEDURE a ENDPROC. Takto by například mohla být upravena procedura main:

```
PROCEDURE main  
  ConsoleShow;   { zobrazí a aktivuje textový výstup }  
  ConsoleClear;  { vymaže plochu textového výstupu }  
  WRITELN("Ahoj, zdraví Vás systém KLONDAIK");  
ENDPROC
```

Obdobným způsobem můžete zadat i povely pro použití grafického výstupního okna. Opět použijeme pouze dříve probrané povely, které zapíšete do části programu mezi slovy PROCEDURE a ENDPROC. Text ve složených závorkách nemusíte zapisovat, jsou to poznámky, které nemají na činnost programu vliv. O tom, jak přebírat dále uvedené příklady z nápovědi k programu se dozvíte podrobně ve druhé lekci.

```
PROCEDURE main  
  ImageShow;      {aktivuje textový výstup}
```



```

ImageInit(150, 200);    {inicializuje plochu}
Rectangle(10, 10, 140, 190); {nakreslí obdélník }
ImageFontColor(clBlue);    {nastaví modrou barvu fontu}
ImageFontSize(16);        {nastaví velikost fontu}
TextOut(50, 30, "KLONDAIK"); { vypíše text }
ENDPROC

```

Jak jsme již hned v úvodu probíraných lekcí uvedli, je základem programování zadání posloupností akcí. To si můžete nyní vyzkoušet. Projděte si předchozí lekce a pokuste se z již získaných znalostí sestavit program, který například nakreslí dům a vedle něj zelený strom. Aby jste to neměli tak složité, uvádíme zde část programu pro nakreslení domu. Povelů pro nakreslení stromu již doplňte sami. Samozřejmě můžete v programu provést i další úpravy dle vlastního uvážení. Aby jste se také naučili něco nového z ovládání uživatelského prostředí systému, zkuste tentokrát spustit program klávesou F6. Tomu, co uvidíte se říká animace programu. Podrobněji se s ní seznámíme až v některé z dalších lekcí. Nyní krátce jen to, že animace mimo interpretace programu současně v okně s programem souběžně zobrazuje interpretovaný řádek programu.

PROCEDURE main

```

ImageShow;            {aktivace grafické plochy}
ImageInit(200, 150);  {inicializace graf.plochy}
ImageBrushColor(clLime);    {nastavení zelené barvy}
Rectangle(0, 130, 200, 150); {nakreslení zahrádky}
ImageBrushColor(clOlive);  {nastavení tmaví zelené}
Rectangle(20,60, 105, 130); {nakreslení stíny domu}
ImageBrushColor(clYellow); {nastavení žluté barvy}
Rectangle(28, 70, 48, 90);  {horní levé okno}
Rectangle(72, 70, 52, 90);  {horní střední okno}
Rectangle(96, 70, 76, 90);  {horní pravé okno}
Rectangle(28, 100, 48, 120); {levé dolní okno}
Rectangle(72, 100, 52, 130); {dveře}
Rectangle(96, 100, 76, 120); {pravé dolní okno}
Ellipse(140,20, 170, 50);   {sluníčko}
ImageBrushColor(clRed);     {nastavení červené barvy}
Triangle(20,60, 60,25, 105,60);{střecha}
ENDPROC

```

Takže jak vypadá program již víte. Měli by jste být již také schopni sestavit jednoduchý program z posloupnosti vykonávaných povelů. Zatím jsme si ale nic neřekli o tom, co znamená první a poslední řádek programu, ve kterých jsou uvedena klíčová slova PROCEDURE a ENDPROC. To se dozvíte v některé z následujících lekcí, ve které si vysvětlíme, co všechno program může a musí obsahovat a jakou má mít strukturu.

9.lekce

Editace programu

Při zápisu programu se zapisuje text programu textovým editorem. Jedná se o tzv. programátorský editor, který neumožňuje nastavovat proměnný druh ani velikost písma. Není

to ani nutné, protože zapsaný text programu slouží pouze jako zápis posloupnosti instrukcí pro systém interpretu.

Zápis nového programu zahájíte stiskem kláves Ctrl+N nebo volbou z menu Soubor/Nový. Do okna program se vytvoří nová prázdná struktura programu, kterou můžete běžným způsobem doplňovat o nové řádky programu. Již existující program můžete otevřít stiskem kláves Ctrl+O, nebo volbou z menu Soubor/Otevřít. Program máte možnost uložit stiskem kláves Ctrl+S nebo volbou z menu Soubor/Uložit. Pokud se jedná o nový program, musíte v dialogovém okně zadat jméno souboru a tím i programu pro uložení. Pokud již zadaný soubor existoval, bude přepsán novou verzí. Neprovádí se přitom záložní kopie původního souboru a tím samozřejmě i programu !

Pokud budete chtít uložit program pod novým jménem, můžete tak učinit volbou z menu Soubor/Uložit pod jménem. Budete dotázáni na jméno nového souboru, do kterého se program uloží. Původní soubor zůstane přitom zachován beze změny. Při práci se soubory a tudíž i s programy vždy platí, že může být v jednom okamžiku aktivní vždy pouze jeden. Nemůžete editovat najednou několik programů.

Základy editace programu jsou obdobné, jako u všech textových editorů. Jakým způsobem zapsat nebo přepsat text snad není nutné uvádět. Stejně tak jak se dostat na konec programu, procházet text po řádcích a podobně. Seznámíme se ale s blokovými operacemi, které jsou při programování velmi výhodné. Pokud budete potřebovat přesunout část textu na jiné místo, budete si muset požadovaný blok textu nejprve označit. To můžete provést buď myší, kdy se přesunete na začátek bloku, stisknete tlačítko myši, držíte je stisknuté a přesunete ukazatel myši na konec požadovaného bloku textu. Text zapsaný v bloku se přitom zvýrazní. Z klávesnice můžete blok textu označit tak, že najedete kurzorem na požadovaný počátek bloku textu, stisknete klávesu Alt, podržíte ji stisknutou a převedete kurzorem na požadovaný konec bloku textu. Po uvolnění klávesy Alt máte označen blok textu pro další operace. Pokud j budete chtít blok textu pouze vymazat, stisknete nyní klávesu Delete. Označený blok textu bude zrušen. Pozor proto při práci s označeným blokem na uvedenou klávesu.

Pro přesun označeného bloku textu programu na jiné místo v programu budete muset použít schránku Windows. Pokud stisknete klávesy Ctrl+C zkopíruje se text z označeného bloku do schránky a blok zůstane v textu stále zachován. Naopak, pokud stisknete klávesy Ctrl+X, bude blok textu převeden do schránky Windows (z textu programu bude zrušen). Pokud se přesunete na požadované místo v programu, můžete do něj text ze schránky Windows zkopírovat. To se provede stiskem kláves Ctrl+V. Text ve schránce zůstává přitom stále zachován do doby dalšího načtení bloku. Pozor však na to, že obsah schránky Windows se může v ostatních aplikacích Windows zrušit. Pro přesun textu mezi editorem a Windows jsou v menu systému zařazeny příslušné volby v menu Editace.

Stejně jako máte možnost přesouvat text mezi schránkou Windows a programem, můžete podobným způsobem převádět text povelů zadaných v příkazovém okně systému. Po dobu editace programu máte stále možnost používat příkazové okno a tím i například zkoušet parametry povelu. Až získáte správný výsledek, převedete odzkoušený povel přes schránku Windows do programu. Teoreticky můžete pomocí bloku převádět do programu v případě potřeby i obsah výstupního textového okna.

Až budete psát rozsáhlejší programy, budete občas potřebovat nalézt v textu programu určitý text. Po stisku kláves Ctrl+F, nebo položkou z menu Editace/Hledat se vám zobrazí dialogový box pro zadání parametrů hledání. Zadáte požadovaný text pro hledání, požadavek zda hledat pouze celá slova, jestli rozlišovat velká a malá písmena a směr hledání.

Podobným způsobem máte možnost zadat náhradu textu jiným textem. Dialogový box se zobrazí po stisku kláves Ctrl+L nebo po zadání volby menu Editace/Změnit. Nahrazovat přitom můžete pouze požadované výskyty hledaného textu, nebo komplet.

Pokud budete chtít program vytisknout, stiskněte klávesy Ctrl+P, nebo použijte volbu menu Editace/Tisk.

10.lekce

Proměnné a jejich typy

Když jsme si ukazovali, jak vypsát příkazem WRITELN číslo do výstupního textového okna, bylo uvedeno, že není možné jednoduchým způsobem uložit výsledek výpočtu do paměti. Je to proto, že to nelze z příkazového řádku provést. Musí se použít program, ve kterém bude hodnota definována. V této lekci si proto ukážeme, jaké typy hodnot můžeme do paměti ukládat.

Hodnoty uložené v paměti se nazývají proměnné. To proto, že jejich hodnoty se mohou programem měnit. Každou proměnnou musíme před jejím prvním použitím nejprve pojmenovat a definovat typ uložené hodnoty. Velmi zjednodušeně by se dalo napsat, že typ proměnné je buď číslo, znakový řetězec nebo logická hodnota. Pokud systém ví, s jakým typem proměnné má pracovat, je schopen provádět přibližné kontroly hodnot a hlavně zná, jaké místo má určité proměnné v paměti rezervovat.

Co je číslo, ví určitě každý. V systému je však definováno několik typů číselných proměnných, které se liší velikostí možných hodnot zpracovávaných čísel. Některé typy číselných proměnných jsou určeny pouze pro celá čísla, jiné mohou obsahovat i desetinnou část. Rozlišuje se také, že čísla mohou být buď pouze kladná, nebo i záporná. Požadovaný typ číselné proměnné je proto vždy nutno pečlivě zvážit. Znakový řetězec představuje seskupení libovolných znaků - písmen. Logická hodnota uchovává výsledky dotazu a může nabývat pouze hodnot pravda nebo nepravda.

Ještě dříve, než bude proměnná v programu poprvé použita, musí se v programu deklarovat. To znamená, že musíme uvést její jméno, typ a případně i počáteční hodnotu. Typ proměnné popisuje přitom množinu hodnot, které může nabývat a operace, které se s ní mohou provádět.

Základní proměnné jsou nejjednodušší proměnné, které reprezentují jeden výskyt prvku údaje. Ze základních typů proměnných jsou odvozovány další deklarace proměnných typů pole, záznam a tabulky, se kterými se seznámíme až později.

Podporované typy proměnných:

BYTE - celočíselný typ s rozsahem 0 až 255

INTEGER - celoèíselný typ s rozsahem od -32768 do +32767
WORD - celoèíselný typ s rozsahem od 0 do 65535
LONGINT - celé èíslo od -2 miliard do + 2 miliard
REAL - èíslo v rozsahu od $2.9 \cdot 10^{-39}$ do $1.7 \cdot 10^{38}$
STRING - øetìzec znakù o délce 255 znakù
CHAR - jeden znak
BOOLEAN - logická promìnná typu **BOOLEAN** (True/False)
TEXT - textový soubor typu Pascal TEXT
ARRAY - jednorozmìrné pole promìnných

Typ **BYTE** zabírá v pamìti jednu slabiku a mùže proto obsahovat pouze celá èísla pro hodnoty od 0 do 255.

Datový typ **INTEGER** je v pamìti uložen ve dvou slabikách a mùže proto nabývat hodnot od -32768 do +32767.

Typ **WORD** zabírá v pamìti také dvě slabiky, ale protože mùže obsahovat pouze kladná èísla, mùže obsahovat èísla od 0 do 65535.

Typ **LONGINT** mùže nabývat záporných hodnot a protože obsahuje v pamìti ètyøi slabiky, mùže nabývat hodnot od mínus dvou miliard do plus dvou miliard.

Datový typ **REAL** se používá pro vyjádøení hodnot s pohyblivou øádovou èárkou. Èíslo je uloženo v šesti slabikách a mùže nabývat hodnot od $2.9 \cdot 10^{-39}$ do $1.7 \cdot 10^{38}$

Datový typ znak **CHAR** má velikost jedné slabiky a uchovává jeden znak. Znakové konstanty se vyjadøují tak, že se znak uzavøe do apostrofù, napøíklad 'A', '1', '&'. Zápis '1' v tomto pøípadì znamená znak pro vyjádøení èíslíce jedna nikoli èíselnou hodnotu. Na rozdíl od klasického jazyka **PASCAL** je možné znaky uzavírat i do uvozovek.

Datový typ **STRING** pøedstavuje posloupnost znakù s pevnou délkou 255 znakù typu **CHAR** bezprostøednì za sebou. Vytváøí tím tzv. øetìzec. Øetìzec musí být ohranièen apostrofy nebo dvojitou uvozovkou. Øetìzec mùže obsahovat maximálnì 255 znakù.

Datový typ **BOOLEAN** má pouze dvě možné hodnoty a to True (pro stav pravda) a False (pro stav nepravda). Lze mu proto v programu pøiøadit buñ konkrétní hodnotu True nebo False, pøípadnì výsledek vyhodnocení logického výrazu. Logické výrazy najdou uplatnìní zejména v podmínìných pøíkazech, pøíkazech cyklu apod. a probereme si je v nìkteré z dalších lekcí.

Dále jsou definovány ještì promìnné typu **TEXT** a **ARRAY**. Jimi se budeme podrobnìji zabývat až pozdìji. V následující lekci si probereme, jak se promìnné v programu deklarují a jak s nimi pracovat.

11.lekce

Deklarace a používání promìnných

V předchozí lekci jsme se seznámili s tím, že přibližné hodnoty výpočtů se ukládají proměnných. Popsali jsme si také, jaké typy proměnných mohou existovat. V této lekci si ukážeme, jak proměnnou v programu deklarovat a jak ji používat.

Proměnné musí být v programu deklarovány ještě před svým prvním použitím ve výpočtech. V deklaraci proměnné se definuje, název proměnné a její typ s možností uvedení počáteční hodnoty. Název proměnné musí začínat písmenem a nesmí obsahovat mezeru. V programu se musí před deklarací proměnných uvést klíčové slovo VAR a na konci ENDVAR. V tomto bloku deklarace proměnných není možné používat žádné povely. Vyzkoušejte si následující příklad:

```
VAR
  a : integer = 100;
  b : integer = 3;
  c : integer;
  d : real;
  s : string;
ENDVAR

PROCEDURE main
  c := a/b;          {výpočet hodnoty}
  d := a/b;          {výpočet hodnoty}
  s := 'OZOGAN KLONDAIK'; {přiznání hodnoty}
  WRITELN(c:10:3);    {vypíše výsledek 3.000 }
  WRITELN(d:10:3);    {vypíše výsledek 3.333 }
  WRITELN(s);
ENDPROC
```

Proměnné a, b a c jsou deklarovány jako typ integer, to znamená, že mohou obsahovat pouze celá čísla. Proměnná d je typu real a může proto obsahovat reálné číslo včetně desetinných míst. Proměnná s je řetězec. V programu jsme proměnné nejprve v bloku označeném slovy VAR a ENDVAR deklarovali. U proměnné a, b jsme definovali současně počáteční hodnoty. V dalším bloku programu označeném bloky PROCEDURE a ENDPROC jsme deklarované proměnné použili.

Hodnotu proměnné stanovíme jejím přiznáním. To se provede pomocí dvojice znaků := mezi nimiž nesmí být mezera. Na levé straně je přitom uvedena proměnná, jejíž hodnota má být změněna a na straně pravé je uvedena přiznávaná hodnota. Hodnotu můžeme uvést buď přímo (číslo, řetězec) nebo jako výraz. Co je to výraz si přesněji probereme v dalších lekcích. Nyní bude stačit, pokud si budete pamatovat, že výraz může být například matematický výpočet.

Po spuštění výše uvedeného programu si systém nejprve z části VAR..ENDVAR definuje použité proměnné a jejich typ a v druhé části programu PROCEDURE..ENDPROC s nimi provádí zadané příkazy. Nejprve do proměnné c uloží dělení a/b. Stejně tak učiní s proměnnou d. Po výpisu proměnných do výstupního okna si můžete myslet, že výpočet neprobíhl správně. Obsah proměnných se liší, u proměnné c nelze vypsat desetinnou část výsledku. Pokud si však zkontrolujete typ proměnné c, zjistíte, že je vše v pořádku. Proměnná je deklarována jako integer, což znamená pouze celé číslo (kladné nebo záporné). Desetinná část výsledku se proto neukládá a není možné ji také vypsat.

Po ukončení programu zůstanou deklarované proměnné stále aktivní. Můžeme je proto i po ukončení programu dále používat a vypsat například příkazem WRITELN jejich obsah. Nyní máte proto možnost vyzkoušet si používání nadeklarovaných proměnných z příkazového okna. Novým spuštěním programu dojde k jejich deaktivaci (zrušení) a nahrazení novými proměnnými. Prozkoušejte si využití i jiných typů proměnných deklarovaných v programu.

Dosud jsme sestavovali naše programy tak, že vykonávaly pouze zadanou posloupnost akcí bez možnosti vynechání některých akcí. V následující lekci se proto seznámíme s možností rozdělení činnosti programu na základě vyhodnocení zadané podmínky.

12.lekce

Podmínky v programu, logické výrazy

Jak jsme již uvedli v úvodu probíraných lekcí, je programování vlastně sestavování posloupnosti vykonávaných akcí. V některých případech musíme mít ale možnost některou z akcí buď vynechat, nebo provést akce jiné. Obojí na základě zadané podmínky. Například pokud mám v kapse více než dvacet korun, půjdu do kina, jinak půjdu domů na televizi. Zadaná podmínka by se dala v lidské mluvě popsat následujícím vztahem:

```
POKUD hotovost > 20 POTOM  
  půjdu do kina  
JINAK  
  budu se dívat na televizi  
KONEC
```

Uvedenému zápisu se říká algoritmus a popisuje schematicky postup prováděných akcí. Vztahu hotovost>20 se říká podmínka. Podmínka nám tedy určuje, jaká akce se bude provádět. Zápis algoritmu je nutné pro počítač převést do programu. Například následujícím způsobem:

```
VAR  
  hotovost : integer;  
ENDVAR
```

```
PROCEDURA main  
  hotovost := 15;      {zadejte vaši hotovost}  
  IF hotovost > 20 THEN  
    WRITELN("Kino")    {pokud je podmínka splněna}  
  ELSE  
    WRITELN("Televize") {pokud není podmínka splněna}  
  ENDIF;  
ENDPROC
```

Nejprve jsme deklarovali proměnnou se jménem hotovost. Dále jsme jí v programu přiřadili určitou hodnotu. V podmínce jsme potom zjišťovali, zda odpovídá našim požadavkům a podle toho jsme provedli určitou akci. Zkuste si zadat sami různé hodnoty stavu vaší hotovosti. Pozor ale na to, že hotovost je deklarována jako typ integer. Nesmíte být proto moc bohatí a mít v hotovosti více než 32767 Kč. Zkuste upravit program tak, aby jste mohli zadávat i větší částky.

Závisí to na typu proměnné.

Všimněte si, že jsme v programu u zápisu podmínky odsadili vykonávané příkazy na úroveň o tři znaky. Tím vynikla struktura podmínky a na první pohled je viditelné její rozdělení. Není to sice nutnost, přesto však doporučujeme uvedenou grafickou podobu zápisu programu ve vlastním zájmu dodržovat.

Podmínka se v programu zadává klíčovým slovem IF, za kterým musí následovat vyhodnocovaná podmínka. Klíčové slovo THEN není povinné, přesto jej však doporučujeme uvádět, protože to vyžaduje klasický jazyk Pascal. Dále následují povely, které se provedou pouze při splnění podmínky. Za klíčovým slovem ELSE se uvedou povely, které se provedou při nesplnění podmínky. Zápis podmínky je ukončen klíčovým slovem ENDIF. V zápisu podmínky můžete vynechat klíčové slovo ELSE včetně akcí pro nesplnění podmínky. Vždy ale musíte uvést ukončení podmínky klíčovým slovem ENDIF. Pokud se tak nestane, nahlásí systém chybu v programu.

Podmínku představuje logický výraz, který udává, zda je podmínka splněna nebo ne. Logické výrazy mohou mít proto výsledek pouze pravda nebo nepravda. V počítačové terminologii True (pravda) nebo False (nepravda). Pokud by jste chtěli výsledek podmínky deklarovat jako proměnnou, museli by jste použít typ BOOLEAN. Logická podmínka zpracovává nejčastěji matematický výraz. Může to však být i výraz zpracovávající řetězec, to však bude psáno až se naučíme s řetězcem důkladněji pracovat.

Matematické výrazy porovnávají nejčastěji několik hodnot. Ve výrazu se přitom může také použít libovolného matematického výpočtu, který je v jazyce definován.

Matematické výrazy zpracovávají aritmetické operace. Výrazy se skládají z operátorů a operandů. Operátor je přitom porovnávaná hodnota a operand je způsob porovnání hodnot. Operátor může představovat libovolný matematický výpočet, který je v jazyce definován. Operand slouží k vyhodnocení operátorů. Při zápisu dodržte, aby byly operátory odděleny od operandů mezerami ! Používají se následující dostatečně známé operandy:

- > **větší než**
- >= **větší nebo rovno než**
- < **menší než**
- <= **menší nebo rovno než**
- = **rovno**
- <> **nerovno**

Ve výrazu je samozřejmě možné používat závorky. Příklad výrazů zpracovávajících matematický výpočet:

```
hotovost > 20
a/2 < 10
2*(a+b) = 2*a+2*b
```

Výrazy můžeme dále v jedné podmínce spojit logickým operátorem s dalším výrazem a vyhodnocovat tak složenou podmínku. Používají se přitom následující logické operátory:

AND a zároveň platí
OR platí jeden nebo druhý výraz
NOT není pravda, negace výrazu

V případě použití logických operátorů mají tyto ve vyhodnocování výrazů přednost před ostatními operátory. Dále se vyhodnocují závorky a až na konci matematické výpočty. Pokud budete chtít zapsat v programu několik výrazů spojených logickým operátorem, musíte umístit výrazy do závorek. Například pro zjištění rozsahu hotovosti od 10 Kč do 30 Kč použijete následující zápis:

```
IF (hotovost >= 10) and (hotovost <= 30) THEN  
  WRITELN("Kino") {pokud je podmínka splněna}  
ELSE  
  WRITELN("Televize") {pokud není podmínka splněna}  
ENDIF;
```

V této lekci jsme si probrali mimo možnosti rozvětvení činnosti programu také způsob zápisu podmínek v programu. Podmínky se v programu používají i v dalších příkazech. Například pro zadání počtu opakování zvolené části programu, jak si ukážeme v následující lekci.

13.lekce

Programové cykly (FOR, REPEAT, UNTIL)

Až doposud prováděly naše programy pouze zadaný sled povelů bez možnosti opakování požadovaných akcí. Pokud by jsme potřebovali například vykonat nějakou část programu stokrát, museli by jsme uvedenou část programu zapsat stokrát. To by však bylo, jak sami jistě uznáte značně nevhodné. V této lekci si proto probereme, jak je možné na základě vyhodnocení podmínek opakovat zadanou část programu.

Pokud budete potřebovat v programu provést nějakou akci s předem známým počtem opakování, bude nejvhodnější použít cyklus typu FOR..ENDFOR. Jedná se o cyklus, ve kterém je definován tzv. čítač, který udává kolikrát se má cyklus ještě provést. Čítač je automaticky systémem zvyšován o zadanou hodnotu. Směr načítání může být počtem nahoru, nebo dolů:

```
VAR  
  x : real  
ENDVAR
```

```
PROCEDURE main  
  ConsoleClear;  
  FOR x := 1 to 3 step 0.5 {cyklus nahoru, udán krok}  
    WRITELN(x:10:1);  
  ENDFOR
```

```
FOR x := 5 downto 1 {cyklus směrem dolů}  
  WRITELN(x:10);
```


ENDFOR ENDPROC

V uvedeném příkladě jsou v programu dva cykly FOR. První z nich zvyšuje stav x směrem nahoru, druhý směrem dolů. Všimněte si, že x musí být sice deklarován jako proměnná, ale přiřazení počáteční hodnoty se provádí až v definici cyklu. Konečná hodnota x se pro směr naštání nahoru zadává za klíčovými slovy 'to'. Pro směr naštání dolů je určeno klíkové slovo 'downto'. Pokud chceme zvyšovat, nebo snižovat stav x o jinou hodnotu než jedna, musíme požadovanou hodnotu uvést za klíčovými slovy 'step'. Hodnota x se nesmí v cyklu měnit. Hodnoty cyklu jsou vyhodnocovány pouze jednou a to při spuštění příkazu FOR.

Důležitá poznámka: proměnná x je v předchozím příkladě deklarována jako typ real, to znamená že může nabývat i desetinných hodnot. Pokud by jste uvedli v tomto příkladě proměnnou x jako typ integer, zvyšoval by se x vždy sice o hodnotu 0.5 ale protože typ integer je pouze celočíselný, ke zvýšení x by nikdy nedošlo a program by skončil v nekonečné smyčce. Museli by jste jej restartovat pomocí volbou z menu Program/Restart programu.

Pokud budete chtít ve svém programu použít cyklus, jehož ukončení bude záviset na splnění zadané podmínky, máte možnost použít cyklus typu REPEAT..UNTIL. Vnitřní část tohoto cyklu se provede vždy minimálně jednou, protože podmínka ukončení cyklu je uvedena až na jeho konci:

```
VAR  
  x : real  
ENDVAR  
  
PROCEDURE main  
  ConsoleClear;  
  x := 1;  
  REPEAT      {začátek cyklu}  
    WRITELN(x/(x+1):10:3);  
    x := x+1; {zvýšení hodnoty  $x$ }  
  UNTIL x = 5 {dokud není výraz pravdivý}  
ENDPROC
```

Příkazy uvedené mezi klíčovými slovy REPEAT a UNTIL se provádí tak dlouho, dokud není výraz definovaný na konci podmínky pravdivý. Nesmíte přitom zapomenout na zvýšení hodnoty x , případně provést jinou akci, která může mít za následek ukončení cyklu.

Jako další cyklus můžete použít WHILE..ENDWHILE, který vyhodnocuje podmínku opakování cyklu na jeho počátku. To umožňuje, že pokud není podmínka splněna, neprovedou se příkazy uvedené mezi klíčovými slovy WHILE a ENDWHILE ani jednou. To může být v některých případech výhodné a je proto nutné se při vlastním programování rozhodnout, který druh cyklu má být použit.

```
VAR  
  x : integer
```

ENDVAR

PROCEDURE main

```
ConsoleClear;  
x := 1;  
WHILE x < 5      {dokud je výraz pravdivý}  
  WRITELN(x*3:10);  
  x := x+1;      {zvýšení hodnoty èítaèe}  
ENDWHILE        {konec cyklu}  
ENDPROC
```

Pro zápis podmínek při používání cyklù platí vše co bylo uvedeno u popisu vřtvení programu pøíkazem IF..ENDIF. Stejnì tak je velmi výhodné a hlavnì pøehledné v textu programu odsadit øádky programu uvnitø cyklù o tøi prázdné znaky doprava. Viz výše uvedené pøíklady. Programy tím získají na pøehlednosti. Souèasnì doporuèujeme uvádìt v programu poznámky ve formì komentáøe. Jak jste si již mohli všimnout, poznámky se uvádíjí ve složených závorkách. Pokud by jste na zaèátku øádku uvedli dvì lomítka, byl by text až do konce øádku v programu ignorován.

V nìkterých pøípadech mùžete potøebovat, aby se cyklus WHILE, REPEAT nebo FOR pøedèasnì pøerušil a pokračoval dalším cyklem od zaèátku. K tomu se používá pøíkaz CONTINUE, který je vyhodnocen jako ENDWHILE, UNTIL nebo ENDFOR a øízení programu je pøedáno zpìt na zaèátek cyklu. Klíèové slovo je pøitom uvedeno až za pøíkazy, které se mají provést vždy, ještì pøed ukonèením (pøerušením) cyklu. Pokud se pøíkaz použije mimo smyèku, je vyvoláno chybové hlášení. Následující pøíklad kreslí pomocí vykreslování bodù èáry s pøerušením uprostøed.

VAR

```
x : integer  
ENDVAR
```

PROCEDURE main

```
ImageInit(100,100); {Inicializuje grafickou plochu}  
ImageShow;          {zobrazí grafický výstup }  
ImagePenColor(clBlue); {nastaví modrou barvu}  
FOR x := 1 to 100;  
  Point(25,x,3);  
  Point(75,x,3);  
  IF (x > 25) and (x < 75) then  
    CONTINUE;      {pøeruší cyklus s návratem}  
  ENDIF  
  Point(50,x,3);  {pro x od 25 do 75 se neprovede !}  
ENDFOR  
ImagePenColor(clRed);      {nastaví èervenou barvu}  
Point(GetmaxX/2,GetmaxY/2,30); {nakreslí bod doprostøed}  
ENDPROC
```

V této lekci jsme se zabývali nìkolika pøíkazy, které však provádìly podobnou èinnost. Pøíkazy cyklu jsou v programech velmi èasto používané. Je však nutné si podle požadovaných akcí vybrat nevhodnìjší pøíkaz pro cyklus.

14.lekce

Zadání vstupních hodnot, tisk výsledků

U programů v předchozích lekcích jsme vždy počítali s tím, že jsou všechna data pro výpočty zadána přímo v programu. Aby jsme ale měli možnost ovlivnit průběh výpočtu, bylo by vhodné zadávat požadované hodnoty přímo dotazem od uživatele. To je možné dvěma způsoby. Buď použijeme příkaz READ, který umožňuje mimo jiné uživatelské zadání hodnoty proměnné, nebo složitější postup, kdy použijeme interaktivní knihovnu pro tvorbu formulářů.

Příkaz READ se používá pro vstup hodnoty proměnné ze souboru, nebo od uživatele z klávesnice. Hodnotu potom můžeme použít dále v programu k výpočtům. Proměnná musí být nejprve definována. Možné je použít pouze číselné a řetězové proměnné. Čtení hodnoty ze souboru si ukážeme později, nyní si předvedeme, jak zadat proměnnou z klávesnice.

Příkaz READ vyvolá jednoduchý formulář s možností zadání hodnoty proměnné. Jméno formuláře je 'vstup' a pokud by jste chtěli umístit nad editační box pro zadání hodnoty proměnné váš text, museli by jste text vypsát příkazem WRITE (pozor, ne WRITELN !) ještě před použitím příkazu READ. Zadaný text se samozřejmě také vypíše do textového výstupního okna. Toho lze využít pro následné zadání výstupních hodnot. Pokud by vám výpis textu vadil, je možné nejprve navstupovat požadované vstupní hodnoty, vymazat obsah výstupního okna a až potom provést výpočty a zobrazení výsledků. Následuje příklad použití příkazu READ.

VAR

s : string;

r : real;

ENDVAR

PROCEDURE main

ConsoleClear;

WRITE("Zadejte celé číslo");

READ(s);

WRITELN();

WRITELN("Zadal jste číslo:",StrToInt(s):12);

WRITELN();

WRITE("Zadejte libovolné číslo");

READ(r);

WRITELN();

WRITELN("Zadal jste číslo:",r:12:4);

WRITELN();

WRITE("Zadejte libovolný řetězec");

READ(s);

WRITELN();

WRITELN("Zadal jste řetězec: ",s);

WRITELN();

ENDPROC

POZOR !!!

V současné verzi nesmíte zadat při vstupu do číselné proměnné nenumerický znak. Pokud tak učiníte, bude výsledkem nulová hodnota. V dalších verzích bude upraveno.

Se složitějším, ale efektnějším způsobem se seznámíme při probírání interaktivní knihovny, která umožňuje definovat v programu vzhled i obsah formuláře. Zkuste si proto zatím alespoň následující příklad:

VAR

```
jmeno : string = '';  
rok1 : integer = 1900; rok2 : integer;  
mesic : integer;  
den1 : integer; den2 : integer;
```

ENDVAR;

PROCEDURE formular;

```
createForm('form1','Vítám Vás !', 100, 100, 270, 180);  
addStatic ('form1','Dobrý den,', 10, 15, 200, 14);  
addStatic ('form1','jaké je Vaše jméno ?',10,45,200,14);  
addEditBox('form1','jmeno', 'jmeno', 10, 80, 250, 20);  
addStatic('form1','narodil jste se v roce:',10,120,200,14);  
addEditBox('form1','rok1', 'rok1', 210, 120, 40, 20);
```

ENDPROC;

PROCEDURE main

```
formular;  
IF (formDialog("form1"))  
  ConsoleClear;  
  WRITELN('Dobrý den ' + jmeno, ', přeji pěkný den !');  
  GetDate(rok2, mesic, den1, den2);  
  WRITE("dnes je ", den1, "/", mesic, "/", rok2, ", ");  
  WRITELN("je Vám ", rok2-rok1, " rokù.");
```

ENDIF

ENDPROC

Takže jak zadávat obsah proměnné od uživatele již známe. Víme také, jak vypsat jejich obsah s případnou další úpravou do textového výstupního okna. Zatím jsme se ale nic nedozvěděli o tom, jak je možné provést výstup informací na tiskárnu. Systém sice neobsahuje žádné příkazy pro přímý tisk na tiskárnu, dovoluje vám ale vytisknout kompletní obsah textového výstupního okna. Musíte proto nejprve provést výstup požadovaných informací a textů do výstupního okna a potom příkazem ConsolePrint vytisknout jeho obsah. Můžete také po ukončení programu, kdy se zobrazí požadované výsledky ve výstupním okně provést jeho tisk pomocí ikony s obrázkem tiskárny, umístěné v horní liště textového výstupního okna.

15. lekce

Vícenásobné vtvění programu

V jedné z předchozích lekcí jsme se seznámili s příkazem IF..ENDIF, který slouží pro vtvění èinnosti programu. Poznali jsme, že můžeme èinnost programu rozdílit do dvou vtví, podle pravdivosti zadané podmínky. Pokud by jste však potøebovali rozdílit èinnost programu podle výsledku na více vtví, byl by zápis pomocí příkazu IF velmi nepřehledný. Proto systém obsahuje příkaz SWITCH, který můžete použít pro rozdílení èinnosti programu do libovolného počtu vtví na základě vyhodnocení výsledku zadané podmínky. Pokud budete například potøebovat pro každé èíslo samostatnou akci, můžete tak uèinit podle následujícího příkladu:

```
VAR
  x : integer
ENDVAR

PROCEDURE main
  ConsoleClear;
  FOR x := 1 to 10;
    SWITCH x OF
      CASE 1:      {porovnávej promìnnou x }
        WRITELN("èíslo jedna");
      ENDCASE
      CASE 5:      {pokud má hodnotu 5 }
        WRITELN("èíslo pìt");
      ENDCASE
      ELSE         {pokud má jinou hodnotu}
        WRITELN(x:10);
      ENDCASE
    ENDSWITCH
  ENDFOR
ENDPROC
```

Za klíèové slovo SWITCH se v programu uvádí hodnota, nebo výraz, který se potom následnì povelem CASE vyhodnocuje. Pokud se hodnota, nebo výsledek výrazu rovná nikterému z výrazù uvedených za klíèovým slovem CASE, bude proveden blok příkazù za tímto výrazem. Program bude potom pokračovat po ukonèení konstrukce příkazu klíèovým slovem ENDSWITCH. Pokud se výraz nebude rovnat žádnému z porovnávaných výrazù, provede se blok příkazù uvedený mezi klíèovými slovy ELSE a ENDCASE.

Jedná se o pomìrnì dosti složitou konstrukci jazyka. Pro názorné pøedvedení proto můžete program spustit klávesou F6 v animaèním režimu, kdy systém zobrazuje postupnì øádek programu, který právnì vykonává. Všimnìte si pøitom, že øádky, které nevyhovují zadání jsou pøeskakovány. Pokud nebudete staèit sledovat animaci, můžete zkusit krokování programu po jednotlivých øádcích. Po stisku klávesy F7 se vykoná vždy pouze jeden øádek programu.

16. lekce

Struktura programu

Až dosud se naše programy skládaly pouze ze dvou částí. V první části uvedené mezi slovy VAR a ENDVAR jsme deklarovali dále používané proměnné. Ve druhé části se mezi klíčovými slovy PROCEDURE a ENDPROC uváděly jednotlivé příkazy programu. Takovým částem programu se říká bloky. Začátek i konec bloku je vždy definován příslušnými klíčovými slovy pro začátek a konec bloku. V programu mohou být mimo deklarací a hlavního příkazového bloku uvedeny i další části, se kterými jsme se dosud neseznámili. Viz následující schématická struktura programu:

TYPE

{definice datových typů - záznamů}

ENDTYPE

VAR

{deklarace proměnných}

ENDVAR

PROCEDURE ...

{deklarace uživatelské procedury}

ENDPROC

FUNCTION ...

{deklarace uživatelské funkce}

ENDPROC

PROCEDURE main

{tílo hlavního programu}

ENDPROC

Definice datových typů

Datové typy jsou uživatelsky definované typy proměnných, které umožňují například seskupovat několik proměnných a používat je jako jednu strukturovanou proměnnou. Podrobněji budou popsány později.

Deklarace proměnných

Již znáte. Jsou uvedeny v bloku mezi klíčovými slovy VAR a ENDVAR. Obsahují deklarace proměnných, určení jejich typu a případné stanovení počáteční hodnoty. Podrobněji se k nim ještě vrátíme později.

Deklarace uživatelských procedur

Procedury jsou uživatelsky definované příkazy, které umožní lépe rozdělit program do logických celků a vícenásobné použití části kódu. Podrobněji se budeme procedurami zabývat v následující lekci.

Deklarace uživatelských funkcí

Funkce se používají pro uživatelskou definici zpracování výrazů s možností vícenásobného použití v programu. Podrobněji se budeme funkcemi zabývat v následující kapitole.

Tělo hlavního programu

Musí být uvedeno v každém programu a musí být obsaženo v proceduře se jménem main, kterou doporučujeme umístit vždy na konec programu. Jejím prováděním se začíná vždy činnost programu. Pokud by nebyla procedura se jménem main v programu nalezena, systém by nahlásil chybu a program by byl ukončen.

Standardní jazyk Pascal dodržuje mnohem přesnější formát zápisu programu. Platí přitom obecné pravidlo, že každý prvek programu se musí v jazyce Pascal nejprve deklarovat a až potom se může používat. Bloky v programech v jazyce Pascal jsou vyznačeny klíčovými slovy Begin a End místo námi používaného ukončení ENDIF, ENDFOR, ENDPROC a podobně. Námi používané řešení vychází z jazyků používaných pro programování databází. Má tu výhodu, že je mnohem přehlednější a rychleji se jej naučíte používat.

Bloková struktura programu se dodržuje i u dalších, dříve probraných příkazů. Jedná se o rozhodovací příkaz IF..ENDIF a všechny příkazy cyklů FOR..ENDFOR, REPEAT..UNTIL a WHILE..ENDWHILE. Všechny struktury programu musí být ukončeny svým příslušným ukončením. Bloky mohou být do sebe vnořovány, nesmí však docházet k překrývením jejich konců. Proto je vhodné dodržovat grafickou úpravu programu, kdy jsou podřízené části bloků odsazeny od svého počátku a konce. Tím se dosáhne současné přehlednosti programu.

Jak již bylo uvedeno, musí být v programu vždy definována procedura se jménem main. Co jsou to procedury a jak je můžeme využít si ukážeme v následující lekci.

17.lekce

Deklarace a používání procedur

Doposud jsme v našich programech používali pouze příkazy zabudované v systému. Nyní si ale předvedeme, jak si můžeme naprogramovat vlastní příkazy. Určitě jste se již setkali s tím, že pokud by jste seskupili několik řádků programu, které se v uvedené sestavě v programu několikrát opakují do jednoho, byl by program přehlednější a kratší. Na to můžete použít procedury, kterým se také říká podprogramy.

Procedury jsou volány z hlavního programu, vykonají příkazy zadané ve svém těle procedury a po jejím ukončení předají vykonávání programu zpět do hlavního programu. Procedury jsou v programu označeny klíčovými slovy PROCEDURE a ukončeny klíčovým slovem ENDPROC, mezi nimiž jsou uvedeny výkonné příkazy procedury. Každá procedura musí být pojmenována. V programu se přitom nesmí vyskytovat více procedur se stejným jménem. Následuje příklad jednoduché procedury:

VAR

a : integer = 10;

b : integer = 3;

ENDVAR

PROCEDURE nadpis

WRITELN("***");**

WRITELN("* počítá systém KLONDAIK * ");

```
WRITELN("*****");  
ENDPROC;
```

```
PROCEDURE main
```

```
  nadpis;  
  WRITELN("součet :", a+b:10:3);  
  nadpis;  
  WRITELN("rozdíl :", a-b:10:3);  
  nadpis;  
  WRITELN("násobek:", a*b:10:3);  
ENDPROC
```

V příkladi jsme definovali proceduru se jménem nadpis, která vždy mezi výpoèty vypíše tøi řádky nadpisu. V hlavní proceduøe main již potom staèí zadat volání procedury nadpis. Systém v tom okamžiku vypíše vždy tøi řádky nadpisu. Tím jsme ušetøili místo a program je pøehlednìjší.

Jistì budete souhlasit s tím, že možnost používání procedur je zajímavá vlastnost systému, že by ale byla ještì zajímavìjší, pokud by procedura dokázala pracovat s rùznými vstupními hodnotami. To je také možné, musíme však ve volání procedury i v její deklaraci uvést seznam parametrù a upravit pøíkazy uvnitø procedury tak, aby dokázaly s promìnnými parametry pracovat. Pokud by jste napøíklad potøebovali nakreslit do grafického výstupního okna na zadanou pozici ètverec s délkou hrany 20 bodù, mohli by jste použít následující pøíklad:

```
VAR  
  k : integer;  
ENDVAR
```

```
PROCEDURE ctverec(x : integer, y : integer);  
  Rectangle(x, y, x+20, y+20);  
ENDPROC
```

```
PROCEDURE main  
  ImageInit(100,100);  
  ImageBrushColor(clYellow);  
  Rectangle(5, 5, 95, 95);  
  ImageBrushColor(clRed);  
  FOR k:= 10 to 70 step 30  
    ctverec(10, k);  
    ctverec(40, k);  
    ctverec(70, k);  
  ENDFOR  
ENDPROC
```

Procedura ctverec je volána se dvìmi parametry uvedenými v závorce, oddìlenými èárkou. Procedura má ve své definici deklarovány promìnné x a y, které se dále v proceduøe používají. Do uvedených promìnných se pøevedou hodnoty zadané ve volání procedury. Procedura potom se zadanými hodnotami pracuje. Tak je umožnìno pøedávat proceduøe ke zpracování vstupní hodnoty.

Pokud má procedura pracovat s předávanými parametry, je nutné v deklaraci procedury uvést za jejím jménem v závorce seznam použitých proměnných. V seznamu se uvádí jméno proměnné a za dvojtečkou její typ. Proměnné jsou v seznamu oddělovány čárkami. Počet deklarovaných proměnných v proceduře musí přitom vždy odpovídat počtu předávaných parametrů ve volání procedury.

Zkuste nyní upravit proceduru ctverec i její volání tak, aby jste mohli zadávat ve volání procedury i délku strany čtverce. Musíte proto přidat do volání procedury další parametr. Stejně tak budete muset přidat parametr do deklarace procedury. Parametr potom použijete pro zadání délky hrany čtverce.

Pokud v programu nadeklaruji svou vlastní proceduru a program spustíte, zachová si systém informace o deklarované proceduře i po ukončení programu. Můžete potom proceduru použít z příkazového okna jako součást systému. To je velmi výhodná vlastnost, protože vám umožní seznámit se dokonale s možnostmi vlastnosti jazyka.

Pomocí deklarace procedur dosáhnete toho, že si můžete definovat své vlastní příkazy. Je přitom možné volané proceduře zadávat předávané parametry. Není ale možné, aby procedura vracela zpět výsledek provedení procedury. To vám umožní až funkce, které budou probírány v následující lekci.

18.lekce

Deklarace a používání funkcí

V předchozí lekci jsme si ukázali, jak si nadefinovat vlastní příkaz ve formě procedury. Mnohdy by ale bylo výhodné, pokud by jsme mohli volat proceduru jako součást výrazu tak, aby nám vrátila procedura výsledek, se kterým by jsme ve výrazu dále pracovali. Nic takového je sice možné, používají se k tomu ale funkce.

Funkce je definovaná část programu, která je volána z výrazu a provádí určité výpočty, případně akce. Při ukončení funkce nám vrátí požadovaný výsledek, se kterým se ve výrazu, odkud byla funkce volána dále pracuje. Uvěme si proto příklad deklarace a použití funkce v programu, který nám provádí výpočet mocniny zadaným exponentem ($5^3 = 5 \times 5 \times 5$, výsledek je 125).

VAR

k : integer;

v : real;

ENDVAR

FUNCTION umocni(x : integer, y : integer): integer

v := x; {převzeme základ}

FOR k := 1 to y

v := v*x; {násobíme opakovaně základem}

ENDFOR

umocni := v; {předáme výsledku funkce}

ENDFUNC

```

PROCEDURE main
  ConsoleClear;
  WRITELN(umocni(2,9):10); {vypíše hodnotu 1024}
  WRITELN(umocni(3,3):10); {vypíše hodnotu 81}
  WRITELN(umocni(5,2):10); {vypíše hodnotu 125}
ENDPROC

```

Funkce se od procedury liší svým zahájením a ukoněním, kdy je výpočet funkce uveden mezi klíčovými slovy **FUNCTION** a **ENDFUNC**. V deklaraci funkce se navíc od procedury musí uvést za seznamem přebíraných parametrů i typ výsledku, který funkce vrátí.

Další odlišností je způsob vytvoření hodnoty výsledku funkce. To je výsledku, který bude předán zpět. Jsou přítomny dvě možnosti. První z nich je standardní a používá ji i jazyk Pascal a proto ji doporučujeme preferovat. Na konci zpracování funkce zapíšeme příkaz:

identifikátor := výraz;

kde na levé straně příkazu je uvedeno jméno funkce a na pravé straně její výsledná hodnota. Pozor na to, že identifikátor přitom není deklarován jako proměnná. Je to proto, že stejně jako si systém deklaruje sám proměnné, které se předávají jako parametry procedury a funkce, stejně tak si systém deklaruje i proměnnou se jménem deklarované procedury.

Druhou možností pro předání výsledné hodnoty je příkaz **RETURN** s uvedením návratové hodnoty:

```

FUNCTION umocni(x : integer, y : integer): integer
  v := x;      {převezneme základ}
  FOR k := 1 to y
    v := v*x;  {násobíme opakovaně základem}
  ENDFOR
  RETURN v;  {předáme výsledek funkce}
ENDFUNC

```

Uvedené použití vychází z jazyka BASIC a databázových jazyků. Může být v některých případech jednodušší, jak však již bylo uvedeno, nejedná se o standardní postup jazyka Pascal.

Pokud v programu nadeklarujete svou vlastní funkci a program spustíte, zachová si systém informace o deklarované funkci i po ukonění programu. Můžete potom funkci použít u příkazů z příkazového okna jako součást systému. To je velmi výhodná vlastnost, protože vám umožňuje seznámit se dokonale s možnostmi vlastnosti jazyka.

19. lekce

Lokální a globální proměnné

V předchozích lekcích jsme v programu deklarovali proměnné v samostatném bloku umístěném na začátku programu. Nadeklarovány přitom platily po celou dobu programu a zůstávaly aktivní dokonce i po ukonění programu. Tomu se říká globální

platnost proměnné, protože je dosažitelná kdykoliv z libovolné procedury v programu. Opakem je proměnná deklarovaná pouze s lokální dobou platnosti. Pro dokonalejší seznámení se s lokálními a globálními proměnnými si upravíme dříve uvedený příklad z lekce zabývající se funkcemi do následující podoby:

```
VAR
  v : integer = 999;
ENDVAR

FUNCTION umocni(x : integer, y : integer): integer
  VAR
    k : integer;
    v : real;
  ENDVAR
  v := x;      {převzeme základ}
  WRITELN("předávaná hodnota x:",x:4);
  WRITELN("lokální hodnota v:",v:4);
  FOR k:= 1 to y
    v := v*x;  {násobíme opakovaně základem}
  ENDFOR
  umocni := v; {předáme výsledku funkce}
ENDPROC

PROCEDURE main
  ConsoleClear;
  WRITELN("globální hodnota v:",V:4);
  WRITELN(umocni(2,9):10); {vypíše hodnotu 1024}
  WRITELN(umocni(3,3):10); {vypíše hodnotu 81}
  WRITELN("globální hodnota v:",V:4);
ENDPROC
```

Všimněte si, že program obsahuje dva bloky s deklarovanými proměnnými. Deklarace blok na začátku programu obsahuje pouze proměnnou 'v' s přidělenou hodnotou. Tato proměnná, protože je deklarována mimo jakékoliv procedury je globální a dosažitelná v celém programu. V definované funkci umocni je uveden znovu blok definice proměnných, kde je opět deklarována proměnná se jménem 'v' (dokonce rozdílného typu). Tato proměnná je pouze lokální, to znamená, že je platná pouze ve funkci, kde byla deklarována. Pokud deklarujeme lokální proměnnou stejného jména, jaké má již existující globální proměnná, zastíní lokální proměnná po dobu své platnosti dříve definovanou globální proměnnou.

Pokud výše uvedený program spustíme, bude se mimo výpočet vypisovat i obsah zadaných proměnných. Všimněte si, že pokud vypíšeme stav proměnné 'v' z hlavní procedury main, bude vypsána vždy hodnota, která byla přidělena proměnné při její deklaraci. Vypsána je tedy hodnota proměnné s globální platností v celém programu. Pokud se bude vypisovat hodnota proměnné 'v' z funkce umocni, bude se vypisovat hodnota lokální proměnné, která dočasně překryje globální proměnnou. Můžeme také vypsát hodnotu proměnné 'x', kterou jsme sami ani nedeclarovali. Deklaroval si ji opět dočasně sám systém pro převedení hodnoty z nadřazeného programu.

Možná se vám bude zpočátku zdát používání a sledování lokálních a globálních proměnných

nepøehledné. Jejich používání ale umožní zpøehlednit program. V některých případech je správné pochopení rozsahu platnosti lokálních a globálních proměnných dokonce nutné. Je to například problém rekurze, kdy z těla funkce voláme stejnou funkci (volání funkce sebe sama).

20. lekce

Rekurze v programu

Rekurzivní funkce (nebo procedura) během provádění příkazů svého těla vyvolá ještě před jeho ukončením sama sebe. Znovu se tedy začne provádět tatáž posloupnost příkazů, aniž by původní byla dokončena. Pokud budeme chtít například vypočítat faktoriál nějakého čísla, musíme postupně vynásobit požadované číslo všemi čísly nižšími vždy o jedničku. Faktoriál čísla 5 se například vypočítá vztahem $5*4*3*2*1$. Pokud budeme chtít pro výpočet faktoriálu napsat obecnou funkci, můžeme vztah upravit do následujícího tvaru: $5*(4*(3*(2*(1))))$. Zde je již vidět, že je možné při násobení snížit vždy základ o jedničku a násobit opakováním výsledku volané funkce. Funkce pro výpočet faktoriálu bude mít proto následující zápis:

```
FUNCTION factorial(n: real): real;
  IF (n = 0) then
    factorial := 1
  ELSE
    factorial := n*factorial(n-1); { REKURZE ! }
  ENDIF
ENDFUNC;

PROCEDURE main
  ConsoleClear;
  WRITELN('Test rekurze:');
  WRITELN('faktoriál čísla 1 je:', factorial(1):5); {= 1}
  WRITELN('faktoriál čísla 2 je:', factorial(2):5); {= 2}
  WRITELN('faktoriál čísla 3 je:', factorial(3):5); {= 6}
  WRITELN('faktoriál čísla 4 je:', factorial(4):5); {= 24}
  WRITELN('faktoriál čísla 5 je:', factorial(5):5); {= 120}
  WRITELN('faktoriál čísla 6 je:', factorial(6):5); {= 720}
  WRITELN('faktoriál čísla 7 je:', factorial(7):5); {=5040}
ENDPROC;
```

Všimněte si, že uvedený program neobsahuje žádnou deklaraci proměnných a přesto je používá. Je to proto, že proměnné jsou deklarovány systémem v deklaraci funkce.

Má-li rekurzivně aktivovaná funkce lokální proměnné, je při každé aktivaci vytvořena nová sada lokálních proměnných, přičemž při nové aktivaci zůstanou lokální proměnné nadřazených aktivací zachovány. Každá aktivace funkce má tedy své lokální proměnné. K proměnným nadřazených aktivací též procedury či funkce není přístup možný. Po ukončení aktivace zaniknou odpovídající lokální proměnné a platnými se stanou lokální proměnné nadřazené aktivace.

Existují úkoly, jejichž řešení vede na zápis programů s rekurzivními procedurami nebo funkcemi. Jsou to problémy, při jejichž rozkladu na podproblémy vznikne problém, který je

obdobný původnímu, avšak je jednodušší. S tím také souvisí následující lekce, kde se budeme vřnovat návrhu programu a rozkladu zadání na jednotlivé procedury.

21. lekce

Návrh a zápis programu

V předchozích lekcích jsme se seznámili se základními příkazy systému a strukturou programu. Poznali jsme, že program tvoří posloupnost operací, vřtvení programu podle podmínek a opakování částí programu. Poznali jsme, že program tvoří i data, která program zpracovává. Nyní by bylo vhodné seznámit se s tím, jak by měl nový program vznikat.

Ještě dříve, než začneme psát nový program by jsme si měli ujasnit požadovanou řinnost programu. Nejlépe tak, že se seznámíme nejprve s tím, jaké výsledky nám má program poskytnout. Podle toho si musíme zadat počáteční podmínky a vstupní hodnoty. Dalším krokem by již měl být rozklad problému tak, aby jsme si v blocích definovali řinnost programu od počátečních hodnot až k požadovaným výstupním hodnotám. Pokud například zjistíme, že musíme nejprve provést výpočet vstupních hodnot a potom výsledné hodnoty zobrazit, můžeme již začít psát kostru programu:

PROCEDURE main

vypocet;

zobrazeni;

ENDPROC

Tím máme současně zadány jména procedur, které musíme naprogramovat. Dále již budeme provádět postupně rozklad dílčích problémů na další podproblémy. V případě složitých zadání se může dokonce stát, že nejsme schopni sestavit ihned program pro vyřešení dílčích problémů, ale musíme znovu definovat rozklad problému na menší související posloupné akce. Uvedeným způsobem se snažíme rozložit počáteční zadání problému do postupných kroků, které dále zjemňujeme. Na konci by jsme se měli dostat do fáze, že řešení již zapisujeme přímo v programovém jazyce. Souběžně s tím upřesňujeme požadovaná data a jejich uložení v proměnných definovaného typu.

Výše uvedenému způsobu řešení programů se říká metoda návrhu programu shora dolů. To proto, že od celkového zadání přecházíme postupně k jednotlivým dílčím úkolům, které nakonec zadáváme v programovém jazyce.

Velmi důležité je také zaznamenání postupu řešení ve formě poznámek. Nyní se vám zdá všechno jasné. Pokud se však dostanete k programu po delší době, nemusí vám být ihned jasné, proč jste postupovali uvedeným způsobem. Zaznamenávejte si proto maximum poznámek k řešení problému. Poznámky mohou být v programu dvojího druhu. Pokud uvedeme na začátku řádky dvě lomítka za sebou, bude systém celý řádek ignorovat a bude jej považovat za poznámku. Takto je možné například psát delší komentáře, nebo při různých pokusech můžeme zneplatnit celý řádek programu, aniž by jsme jej museli vymazávat.

Druhou možností zápisu vlastního komentáře do programu je použití složených závorek. Vše, co je v programu uvedeno mezi levou a pravou složenou závorkou je považováno za komentář a je systémem ignorováno. Tento druh poznámek je výhodné používat na konci

řádku pro zápis popisu provedené činnosti řádku. Takto je možné označit jako poznámku i text uprostřed řádku, nebo několik řádků najednou.

Při zápisu programu se snažte dodržovat grafickou úpravu programu, kdy jsou řádky odsazena od svého záhlaví o několik znaků. Takový program je potom i na první pohled lépe čitelný a snáze se hledají případné chyby.

Systém nerozlišuje v programu malá a velká písmena. Proto bude vhodné, pokud budete dodržovat jednotnou dále popsanou úpravu. Nejvhodnější je používat pro návěští a ukončení bloku (FOR..ELSE..ENDFOR) velká písmena. Stejně tak je vhodné psát jména základních příkazů systému (READ, WRITELN, CONTINUE ..) velkými písmeny. Jména procedur a funkcí používaná z knihoven systému je vhodné zapisovat tak, aby vždy první písmeno slova bylo velké. Složeným slovem se rozumí jméno složené z několika slov, mezi nimiž nesmí být mezera. Například ImageInit, ConsoleClear apod. Stejným způsobem je vhodné pojmenovat vlastní víceznaková jména deklarovaných proměnných.

22. lekce

Ladící a animační režim programu

V předchozí lekci jsme se seznámili s tím, jak program navrhnout a sestavit. Nikdy však není zaručeno, že vám bude program ihned pracovat správně. Chybu v programu vám může ohlásit systém v případě chybně deklarované proměnné, špatně zadaného příkazu a podobně. Avšak i v případě, že program probíhá až do konce bez vypsaní chybového hlášení, neznamená to, že je program bez chyb. Chyba nemusí být v samotném zápisu programu, ale v chybném sestavení algoritmu. Protože se takové chyby špatně hledají, byl systém doplněn několika pomůckami, dovolujícími chyby v programu nalézt.

V jedné z předchozích lekcí jsme si zblíží ukázali animaci programu. Při ní se program spouští klávesou F6 případně volbou z menu Program/Animace, nebo kliknutím na ikonu s obrázkem filmu. Program přitom bude mimo vykonávání naprogramované činnosti současně zobrazovat v okně s programem řádku, kterou právě vykonává. Chod programu je současně zpomalen tak, aby bylo možné průběh animace programu sledovat. To nám umožní kontrolovat průběh programu přes jednotlivé příkazy, volání procedur a funkcí. Animaci programu máte možnost kdykoliv přerušit tak, že stisknete klávesu F9 a program bude pokračovat normální rychlostí bez zobrazování vykonávaného řádku programu. Pokud by jste chtěli animaci pouze dočasně pozastavit, můžete tak učinit kliknutím myši na ikonu s obrázkem STOP značky.

Další, velmi zajímavou možností systému vzhledem k ladění programu je schopnost průběžného vypisování hodnot proměnných. Proměnné se zobrazují ve svém samostatném okně, které otevřete z menu volbou Okna/Proměnné. Zobrazí se vám tabulka se dvěma ikonami v horní liště. Levá ikona se používá pro přidání proměnné do tabulky, pravá ikona proměnnou z tabulky zruší. Zkuste si proto napsat program, ve kterém použijete cyklus FOR..ENDFOR. Mimo řídící proměnné deklarujte i další proměnné. Jména proměnných zadejte i do tabulky pro sledování proměnných. To učiníte tak, že kliknete myši na levou ikonu, a do dotazovacího okna zadejte jméno proměnnou, kterou chcete sledovat. Po jejím zadání se zobrazí proměnné v tabulce s uvedením, že hodnota proměnné není definována. stejným

způsobem zadejte i další proměnné, jejichž stav budete chtít sledovat. Při spuštění programu v režimu animace sledujte současně tabulku s hodnotami proměnných. Dokud není v programu proměnné přidělena hodnota, zobrazuje se v jejich stavu tzv. nedefinovaná náhodná veličina. Po přidělení hodnoty a její každé změně je stav hodnoty proměnné v tabulce aktualizován. Můžete proto sledovat, zda odpovídá její hodnota předpokládaným veličinám.

Mimo režimu animace, kdy jsou úkony programu vykonávány automaticky se zadanou časovou prodlevou máte možnost program krokovat sami po jednotlivých úkonech. Krokování programu po úkonech se aktivuje klávesou F7. Systém při každém stisku klávesy F7 provede pouze jeden úkon programu se současným zobrazením úkonu, kterou bude provádět v následujícím kroku. Máte tak možnost sami ovládat časové prodlevy mezi prováděním jednotlivých úkonů programu. Současně máte také možnost výše popsaného zobrazení hodnot proměnných v příslušném okně. Pokud se z programu volá procedura, pokračuje krokování programu ve volané proceduře. Pokud ale nechcete proceduru krokovat, stiskněte na úkon s procedurou místo F7 klávesu F8, která umožní skok do procedury přeskočit. Program v takovém případě vykoná volanou proceduru běžnou rychlostí bez zobrazování prováděných úkonů programu. Obdobně to platí i pro volání funkce v programu.

Pokud budete chtít sledovat průběh programu až od určitého místa, nastavte kurzor v okně s programem na požadované místo a spusťte program funkcí klávesou F4. Program se spustí běžnou rychlostí a po dosažení úkonu programu, na kterém je kurzor přejde do ladícího režimu. Dále budete mít možnost pokračovat buď krokováním, animací nebo například po kontrole hodnot proměnných pokračovat běžnou rychlostí.

Výše popsaná animace programu i jeho krokování vám dovolí sledovat průběh činnosti programu a změny hodnot proměnných. To vám umožní poznat důkladněji činnost příkazů pro cyklus a vstavení programu. Krokování i animaci programu máte možnost kdykoliv přerušit s tím, že činnost programu buď pozastavíte nebo spustíte běžnou rychlostí. Přerušování se provede kliknutím myši na ikonu se STOP značkou. Pokračování programu běžnou rychlostí je možné stiskem klávesy F9. Krokování programu, jeho animací i běžné spuštění je možné libovolně kombinovat.

Protože zůstává aktivní příkazové okno stále přístupné, máte možnost kdykoliv změnit hodnotu proměnné, vypsát si obsah proměnných, které nemáte v okně pro sledování proměnných apod. Tím dosáhnete absolutní nadvlády nad programem, což vám kompilované programy nikdy neumožní.

DOČASNÝ ZÁVĚR

Další lekce obsluhy programu OZOGAN KLONDAIK se připravují a budou dodávány s plnou verzí programu. Budou obsahovat zejména popis interaktivní knihovny umožňující tvorbu vstupních formulářů, popis práce s textovými soubory, způsob práce se záznamy a podobně.

Liberec 18.8.1997