ÚVOD

Následující text je urèen pro prvotní seznámení se s programem OZOGAN KLONDAIK. Podrobná nápovìda vèetnì popisu vnitøního jazyka je dostupná v programu ve formì nápovìdy.

Pro ty, kteøí dosud nikdy neprogramovali na poèítaèi se mùže zdát pojem programování záhadný a tajemný. Jedná se však o zcela bìžnou vìc, kterou vlastnì všichni známe již od dìtských let. Urèitì jste si již mnohokrát øekli: "ráno pùjdu do školy (práce), odpoledne na koupalištì a veèer se budu dívat na televizi". Tím jste si sestavili program dne. Definovali jste posloupnost akcí, které provedete. Mùžete si však také stanovit podmínky programu. Napøíklad pokud bude odpoledne hezky, pùjdete na koupalištì, jinak (bude pršet) si budete èíst nebo se uèit. V nìkterých pøípadech si mùžete stanovit opakování nìkterých akcí. Napøíklad že se budete uèit tak dlouho, dokud to nepochopíte. Jak vidíte, programovat už umíte. Sestavujete si program dne, výuky, dovolené. Nyní už zbývá pouze nauèit se pøevést program do podoby, kterou zvládne i poèítaè.

Poèítaèový program pøedstavuje definici posloupnosti akcí, podmínky jejich provedení a opakování. Aby byl poèítaè schopen požadované èinnosti provádìt, musíte mu je zadat v podobì, které on rozumí. Pokud budete chtít anglièanovi øíct svùj program dne, budete mu jej muset pøeložit do angliètiny. Stejnì tak musíte pøeložit program ze slovní podoby do poèítaèového jazyka. Poèítaè potom bude èíst jednu øádku programu za druhou a vykonávat to, co jste mu zadali.

V dalších lekcích bude popsáno, jakým zpùsobem budete s poèítaèem komunikovat, jak zapsat program. Nauèíte se také základùm použitého poèítaèového jazyka.

1.lekce

Instalace, spuštìní a ukonèení 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 šíøena ve zkomprimovaném formátu v souboru KLONDAIK.ZIP a staèí ji pouze "rozbalit" do libovolného adresáøe na disku poèítaèe. V tomto pøípadì se nezakládá programová skupina ve Windows a spuštìní programu je nutné provést spuštìním souboru KLONDAIK.EXE. Demoverze zabírá na disku asi 1 Mb prostoru.

Demoverze šíøená na disketách i plná verze programu obsahuje již instalaèní program. Program se dodává na jedné disketì formátu 3,5" HD. Instalace se spustí z diskety pøíkazem INSTALL.EXE. Zobrazí se vám øídící instalaèní panel, ve kterém musíte zadat nejprve adresáø pro instalaci programu. Standardnì je pro instalaci nastaven adresáø C:\OZOKLOND\ pro demoverzi, pøípadnì C:\OZOKLON2 pro plnou verzi programu. Cílový disk i adresáø je možné zmìnit, pøesto však doporuèujeme zachovat pøeddefinované jméno adresáøe. Kopírování souborù do zadaného adresáøe se zahájí po stisku tlaèítka "Instalace". Program zabírá na disku asi 1 Mb prostoru. Po zkopírování souborù je nabídnuta možnost založení skupiny programù Windows. Pokud budete souhlasit, založí se skupina OZOGAN, do které se nadefinuje spuštìní programu KLONDAIK, pøípadnì prohlížení helpu k programu.

Pro rozlišení pøesné definice programu OZOGAN KLONDAIK od editovaného a ladìného programu bude v následujícím textu u všech lekcí uvádìn program OZOGAN KLONDAIK jako systém a pod pojmem program se bude rozumìt ladìný a editovaný uživatelùv program.

Systém se startuje po dokonèené instalaci spuštìním souboru OZOKLOND.EXE (demoverze), pøípadnì OZOKLON2.EXE (plná verze programu). Pøi prvním spuštìní systému je integrované uživatelské prostøedí nastaveno do standardního stavu.

Program je možné ukonèit buï volbou z menu Soubor/Ukonèit program, pøípadnì ikonou z toolbaru. Pokud je editován program a zmìny nebyly dosud uloženy, budete dotázáni systémem na uložení zmìn. Pøi ukonèení programu se provede uložení rozložení oken integrovaného prostøedí do inicializaèního souboru OZOKLON*.INI (dle druhu verze programu), který se nachází v adresáøi Windows. Pøi dalším spuštìní 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 podøízeno tomuto standardu. Systém je ovládán z menu, èasto používané volby jsou pøitom pøístupné i z toolbaru pomocí ikon, pøípadnì pomocí funkèních kláves. Veškeré akce probíhají v nìkolika oknech s pøesnì definovaným urèením. Celá práce se systémem probíhá v nìkolika režimech dle kterých jsou využívána pøíslušná okna. Okna lze na ploše obrazovky uspoøádat podle aktuální potøeby. Okna je možné minimalizovat, nelze je však žádným zpùsobem zrušit.

Nejèastìji používané okno je okno s programem, které obsahuje vyvíjený a ladìný program. V dalším, pøíkazovém oknì je možné zadávat pøímo pøíkazy jazyka bez nutnosti spuštìní programu. Výsledky programu jsou zobrazovány buï v textovém nebo grafickém výstupním oknì. Pøi krokování programu je možné zobrazovat stav promìnných v oknì sledování promìnných. Pomocnou funkci má okno text, ve kterém je možné zobrazit libovolný textový soubor. Nìkterá z oken mají svou vlastní sadu ikon umístìných v toolbaru, který se zobrazí po aktivaci okna v jeho horní lištì.

Okna mùžete po pracovní ploše libovolnì pøepínat, posouvat, zvìtšovat je èi zmenšovat. Pøepínání oken je možné kliknutím na libovolnou viditelnou èást okna. Po kliknutí se stane pøíslušné okno aktivní. Pokud není okno viditelné, použijte menu Okna, kde zvolíte požadované okno. V dalších lekcích se nauèíte, že je možné zviditelnit požadované okno i pøíkazem z programu. Kliknutím na pøíslušnou ikonu v pravém horním rohu okna je možné okno maximalizovat, pøípadnì minimalizovat. Okno se pøesouvá nejlépe myší, kdy je uchopíte za horní lištu a pøesunete na požadované místo. Zmìny velikosti okna dosáhnete po uchopení pravého dolního rohu okna a nastavení na požadovanou velikost.

Vždy pøi spuštìní systému je nastaven editaèní režim, ve kterém je možné editovat vytváøený program, pøípadnì zadávat pøíkazy z pøíkazového okna. Po spuštìní editovaného programu je nastaven provozní režim, ve kterém je provozována ladìná aplikace. Zvláštním druhem provozního režimu je potom ladící režim, ve kterém lze krokovat program a amimaèní režim,

pøi kterém je možné zobrazovat prùbìh èinnosti programu. Každý režim má svá specifika, která budou popsána v následujících lekcích.

Systém obsahuje kontextovou nápovìdu, která je dostupná v nìkolika stupních. Pøi výbìru volby z menu je krátká nápovìda k vybrané volbì zobrazována ve stavovém øádku v dolní èásti systému. Tzv. bublinová nápovìda je zobrazována po najetí myší na ikony v toolbaru. Nejobsáhlejší nápovìda se zobrazí po stisku klávesy F1. Jedná se o hypertextový dokument s provázanými odkazy na zvolená hesla. Zde uvádìné pøíklady programù máte možnost si do systému z nápovìdy pøetáhnout pomocí blokového pøesunu pøes schránku Windows. To provedete tak, že najedete v nápovìdì myší na zaèátek ukázky programu, stisknete levé tlaèítko myši, držíte jej stisknuté a pøetáhnete myš na konec textu pøíkladu. Tam tlaèítko myši uvolníte. Stiskem kláves Ctrl+C se pøevede takto oznaèený blok textu do schránky Windows. Nyní se pøepnete do okna s programem a stiskem kláves Ctrl+V pøevedeme obsah schránky.

K ovládání programu je vhodné používat myš. Vìtšina voleb je sice dostupná souèasnì z menu i myší, pøesto je však ovládání myší mnohem pøíjemnìjší a operativnìjší. Pokud se v dalším textu používá myš, pøedpokládá se levé tlaèítko.

3.lekce
Práce s pøíkazovým a textovým výstupním oknem

Nejprve si ukážeme, jak se dají velmi jednoduše zadávat systému pøíkazy bez zápisu programu. Najdìte na pracovní ploše okno nazvané pøíkazy a uèiòte je aktivním - kliknìte na nìj myší. Pokud není okno viditelné , pøípadnì je minimalizované, mùžete zvolit z menu volbu Okna/Pøíkazy nebo stiskem kláves Ctrl+F2. Upravte myší velikost okna tak, aby zabíralo levou polovinu obrazovky. Obdobným zpùsobem nastavte okno Výstup-text do pravé poloviny obrazovky. Pøepnìte ze zpìt do pøíkazového okna.

Okno pøíkazy se podobá textovému editoru. Má však velmi dùležitou vlastnost. Pokud napíšete v oknì libovolný text a stisknete klávesu ENTER, pokusí se systém ihned napsaný øádek interpretovat jako známý pøíkaz. To znamená, že systém si pøeète text øádku, na kterém stál kurzor a zjišuje, zda se jedná o jemu známý povel nebo pøíkaz. Pokud ano, povel nebo pøíkaz provede. Proto se toto okno nazývá pøíkazové.

Pøíkazy a povely jsou pøitom pro systém instrukce, co má vykonat. Pøíkazem je vìtšinou oznaèen pøímý hlavní povel systému. Pøíkazy jsou v programech psány velkými písmeny. Jako povely jsou oznaèovány doplòkové pøíkazy, které jsou zabudovány jako podpùrné akce. Jsou psány malými písmeny s velkými písmeny na poèátku slova. Pokud jsou názvy povelù uvedeny z více slov, nesmí být mezi nimi uvedeny mezery.

Napište nyní v pøíkazovém oknì slovo BEEP a stisknìte klávesu ENTER. Systém zjistí, že pøíkaz BEEP je pokyn, pro pípnutí. Zajistí proto, že reproduktor poèítaèe krátce pípne. Takto systém interpretoval vámi zadaný pøíkaz do podoby srozumitelné pro poèítaè. Interpretace proto znamená pøevedení pøíkazu nebo øádku programu do formy srozumitelné pro zaøízení poèítaèe (obrazovka, tiskárna ...). Takto je v systému OZOGAN KLONDAIK interpretován každý øádek programu. Pokud je nìkterý øádek nìkolikrát opakován, provádí se i opakovanì jeho interpretace. Takovým systémùm se proto øíká interprety. Jiné systémy pøekládají zdrojový

program pouze jednou pøímo do spustitelné podoby. To jsou kompilátory a bývají rychlejší. Neumožòují ale interaktivní práci uživatele se systémem. Proto jsou pro zaèátky programování vhodnìjší interprety, které vám umožní lépe a rychleji proniknout do tajù programování.

V pøíkazovém oknì máte možnost vyzkoušet si mnoho pøíkazù a povelù systému OZOGAN KLONDAIK, aniž napíšete jedinou øádku programu. Pokud udìláte chybu, systém vás na to upozorní a vy máte možnost ihned chybu opravit bez nutnosti kompilace programu. Zapomenete-li napøíklad napsat ve výše uvedeném pøíkazu BEEP jedno 'E', oznámí vám systém, že uvedený pøíkaz nezná. Najedete proto kurzorem na chybné místo, provedete opravu a po stisku klávesy ENTER se již pøíkaz vykoná.

Jistì jste si již všimli, že klávesa ENTER neukonèí na místì kurzoru øádek, jak to bývá u textových editorù. Øádek zùstane celý zachován a kurzor se automaticky pøesune na zaèátek prázdného øádku za pøedchozí pøíkazy. Pokud zkusíte zapsat pøíkaz

WRITELN("KLONDAIK");

vypíše se do textového výstupního okna text KLONDAIK. Možnostmi pøíkazu WRITELN je vìnována následující lekce. Nyní bude staèit, pokud si zapamatujete, že pøíkaz WRITELN v uvedeném tvaru vypíše obsah textu uvedeného mezi uvozovkami do textového výstupního okna. Mezi uvozovky mùžete zkusit pøitom zadat libovolný jiný text. Zkuste text mezi uvozovkami nìkolikrát zmìnit a zmìny vždy odešlete klávesou ENTER. Jistì si všimnete, že pùvodní text pøed opravou zùstane v pøíkazovém oknì zachován a pøíkaz s novým textem bude doplnìn na konec pøíkazového okna. Díky tomu máte vždy pøehled o historii zadávaných pøíkazù. Neprovedete-li v textu žádnou zmìnu a pøíkaz opakovanì odešlete, nebude stejný pøíkaz zadávaný beze zmìny za sebou v historii pøíkazového okna za sebou znovu opakován.

Historii zadávaných pøíkazù mùžete libovolnì používat bez ohledu na poøadí pùvodního zadávání. Pokud provedete v nìkterém øádku pøíkazového okna libovolnou zmìnu a následnì stisknete klávesu ESC, pøíkaz se neprovede a text upraveného øádku bude uveden do pùvodního stavu. Pøíkazy vyzkoušené v pøíkazovém oknì máte možnost pøes schránku Windows následnì pøevést do programu.

Rùznými pokusy s pøíkazovým oknem jste dosáhli toho, že máte ve výstupním textovém oknì uvedeny rùzné texty. S každým použitím pøíkazu WRITELN pøibude na konec okna jeden øádek. To pøipomíná použití psacího stroje, kdy také není možné se vracet zpìt. V dávných dobách prahistorie výpoèetní techniky sálových poèítaèù se podobné psací stroje používaly pro komunikaci s poèítaèem. Tehdy se jim øíkalo konzole - anglicky console. Uvedené anglické oznaèení je proto také používáno pro pøíkazy pøímo související s obsluhou textového výstupního okna. Pokud napøíklad zkusíte zadat v pøíkazovém oknì pøíkaz ConsoleClear, zjistíte, že zruší obsah textového výstupního okna. Obdobného efektu dosáhnete, pokud se pøepnete do textového výstupního okna a stisknete v podøízeném toolbaru uvedeného okna ikonu znázoròující prázdnou stránku. Po kontrolním dotazu bude obsah textového výstupního okna zrušen.

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

4.lekce
Pøíkaz WRITELN, matematické výpoèty

V pøedchozí kapitole jste se nauèili zadávat z pøíkazového okna pøíkazy umožòující vypsat do textového výstupního okna znaky. Nyní si ukážeme, jak vypisovat èísla a aritmetické výpoèty.

S pøíkazem WRITELN jste již mìli možnost se krátce seznámit. Používá se na výpis hodnot. Požadovaná hodnota se pøitom uvádìla v závorce. Pokud se u pøíkazu WRITELN uvedou pouze prázdné závorky, nebude nic vypsáno a další výstup do textového výstupního okna bude provádìn na novou øádku. WRITELN je pouze pøíkaz který uvádí systému, že má nìco vypsat. Uvedené nìco se pøitom uvádí v závorce jako parametr pøíkazu. Pokud uvedete znaky v uvozovkách nebo apostrofech, pøevedou se uvedené znaky do textového výstupního okna. Pokud by jste chtìli vypsat èíslo, musíte zadat pøíkaz v následujícím tvaru:

WRITELN(10);

Staèí tedy napsat jako parametr pøíkazu do závorek pøímo požadované èíslo. Èíslo se vypíše se zarovnáním od levého kraje okna. Pokud budete chtít ponechat zleva odsazené místo, budete muset zadat požadovanou délku vypsaného èísla v poètech znakù:

WRITELN(10:5)

Délka vypsaného èísla je v tomto pøípadì pìt znakù. To je parametr uvedený v pøíkazu za dvojteèkou a udává poèet míst zobrazení èísla. Nejedná se proto o výpoèet dìlení. Výpoèty je ale samozøejmì také možné zadat. V tomto pøípadì by jste pro dìlení museli zadat:

WRITELN(10/5:5)

Jako výsledek bude zobrazeno èíslo 2 o délce pìti znakù. Zkuste si sami zadat souèet, rozdíl, pøípadnì násobek èísel:

WRITELN(10+5:5);
WRITELN(10-5:5);
WRITELN(10*5:5);

Zobrazí se vždy výsledek aritmetického výrazu o délce pìti znakù. Pøekvapeni budete možná pokud zadáte pøíkaz:

WRITELN(10/3:5);

Sami jistì poznáte, že nìco není v poøádku. Ztratila se desetinná èást výsledku. Až dosud jsme pracovali pouze s celými èísly. Výsledky se také vždy zobrazují standardnì jako celá èísla. Zkuste nyní následující pøíkaz:

WRITELN(10/3:5:2);

Zobrazí se již výsledek s uvedením dvou desetinných míst. Je to proto, že jsme za dvojteèkou uvedli délku vypisovaného èísla a souèasnì za další dvojteèkou poèet zobrazených desetinných míst. Zkuste si sami další pøíklady a seznamte se zadáváním uvedených parametrù.

S možností výpisu nenumerických údajù, tedy znakù jste se již seznámili v pøedchozí lekci. Seskupení nìkolika znakù se pøitom nazývá øetìzec a také tak již bude v následujícím textu uvádìno. Pokud by jste chtìli jedním pøíkazem WRITELN vypsat najednou øetìzec i znaky, mùžete tak uèinit uvedením nìkolika parametrù oddìlených èárkou:

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

Stejnì tak mùžete vypsat samozøejmì i nìkolik èísel nebo øetìzcù oddìlených v pøíkazu èárkami. To vám dovolí ve spojení s možností uvádìní délky èísel vhodnou grafickou úpravu textu:

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

Pøíkaz WRITELN provede po svém ukonèení vždy pøechod na novou øádku. Pokud však budete potøebovat, aby další výpis pokraèoval na stejné øádce, mùžete použít pøíkazu WRITE. Ten neprovádí ukonèení øádku, následující výpis je zahájen od pozice ukonèení pøíkazu WRITE. Možnosti a parametry pøíkazu WRITE jsou pøitom totožné jako pro uvádìný popis pøíkazu WRITELN.

Výše uvedený popis použití pøíkazu WRITELN se vám mùže zdát na první pohled složitý. Je však nutné jej pøesnì dodržet. Systém si hlídá dùslednì jeho dodržování a v pøípadì chyby odmítne pøíkaz vykonat. Pøesná 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 pøíkazù. Uvádíme zde pro pøedstavu zkrácenou definici syntaxe zápisu pøíkazu WRITELN tak, jak byla výše popsána:

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

V zápise znamená výraz libovolný výraz, Napøíklad øetìzec, èíslo nebo výpoèet. Údaje v hranatých závorkách se nemusí uvádìt. Pokud proto uvedeme dvojteèku následovanou èíslem, jedná se o délku vypisovaného èísla. Pøípadná další dvojteèka s èíslem uvádí poèet desetinných míst. Pokud si dobøe prohlédnete uvedený syntaktický zápis zjistíte, že mezi závorkami nemusíte nic uvádìt. To je dáno levou hranatou závorkou uvedenou ihned za levou kulatou závorkou. Ukonèení této volitelné èásti je provedeno pravou hranatou závorkou pøed pravou kulatou závorkou. Pokud proto uvedete pøíkaz:

WRITELN();

nevypíše se žádný text, dojde ale k vynechání prázdného øádku. Støedník uvedený v

hranatých závorkách na konci pøíkazu není nutné uvádìt. V popisu syntaxe je uveden, protože v klasickém jazyku Pascal musí být každý øádek programu ukonèen støedníkem. Uvedený zpùsob popisu syntaxe je použit u popisu všech pøíkazù a knihoven procedur a funkcí uvedených v manuálu i helpu k programu.

Pokud jste se dokonale seznámili s použitím pøíkazu WRITELN, mùžete ve spolupráci s manuálem k programu vyzkoušet nìkteré funkce matematické knihovny. Dále uvedené pøíklady uvádìjí na konci øádku mezi složenými závorkami komentáøe, které nemají na výsledek žádný vliv. Ve skuteènosti je proto nemusíte vèetnì složených závorek uvádìt.

```
WRITELN(Abs(-55));      {absolutní hodnota èísla}
WRITELN(Cos(PI));       {kosinus Ludolfova èísla PI}
WRITELN(Max(3,10);      {maximální hodnota zadaných èísel}
WRITELN(Min(3,10);      {minimální hodnota zadaných èísel}
WRITELN(Random(500);    {náhodné èíslo do 500}
WRITELN(Round(12.82);   {zaokrouhluje èíslo}
WRITELN(Sqr(5);         {vrací druhou mocninu èísla}
WRITELN(Sqrt(16);       {vrací druhou odmocninu èísla}
WRITELN(Trunc(12.82);   {odøízne desetinou èást èísla}
```

Jak vidíte, mùžete použít pøíkaz WRITELN jako docela chytrou kalkulaèku. V souèasnosti pøi použití pouze pøíkazového okna má ale jednu dosti velkou vadu. Nemá žádnou pamì, do které by jste si zaznamenali hodnoty mezivýsledkù. V nìkteré z dalších lekcí, kdy budeme probírat deklaraci promìnných v programu se ale nauèíme 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 pøedchozích kapitolách jsme se nauèili používat pøíkazové 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 libovolnì kreslit.

Uzavøete døíve používané textové výstupní okno. Mùžete tak uèinit kliknutím na uzavírací ikonu, pøípadnì mùžete zadat v pøíkazovém oknì povel ConsoleHide. Dále aktivujte grafické výstupní okno. To lze provést z menu volbou Okna/výstup grafika. Okno je možné také zobrazit povelem ImageShow. Zobrazené okno upravte opìt tak, aby pokrývalo pravou polovinu obrazovky.

Jak vidíte, každé okno je pro úèely ovládání z programu pojmenováno významovì dle anglického názvosloví. Existují pøitom povely pro zobrazení (anglicky show) a ukrytí (anglicky hide) okna. Možná se pozastavujete nad tím, že u èeského programu se používá cizojazyèných termínù. Je to proto, že systém OZOGAN KLONDAIK mùže sloužit jako nástroj pro výuku programování. Proto je podle nás vhodné si již od poèátku zvyknout na terminologii, která se pøi bìžném programování používá. Názvy procedur, funkcí a konstant pøitom vychází z používané terminologie jazykù Pascal a získané znalosti jistì dále

zužitkujete. Použitá poèítaèová angliètina je pøitom velmi jednoduchá a nemìla by nikomu èinit problémy.

Po aktivování grafického výstupního okna se vám v jeho horní lištì zobrazí øada ikon pro ovládání zabudovaného grafického editoru a ikony pro možnost kreslení na grafické ploše. Ikony jsou umístìny do tøí skupin. Levá skupina se používá pro naèítání a ukládání obrázkù v grafickém formátu *.BMP. Støední skupina slouží pro definici kresleného tvaru a v pravé skupinì naleznete nastavení typu èáry a výplnì ploch.

Nejprve se seznámíme s možností kreslení základních geometrických tvarù. Ve støední skupinì ikon jsou seskupena tlaèítka pro èáru, obdélník, kružnici a obdélník se zakulacenými hranami. Kliknìte nejprve na tlaèítko s èárou. Tím jste zadali, že budete chtít kreslit èáry. Uvidíte, že tlaèítko zùstalo stisknuto. Pøesuòte ukazatel myši na grafickou plochu. Pokud nyní na grafické ploše stisknete tlaèítko myši, podržíte jej a pøesunete na novou pozici, bude se kreslit èára z bodu stisku tlaèítka do aktuální pozice ukazatele myši. Po uvolnìní tlaèítka zùstane èára zachována. Zkuste si nakreslit i vodorovné a svislé èáry. Obdobným zpùsobem lze na grafickou plochu kreslit obdélníky, kružnice a obdélníky se zakulacenými rohy. Pro zvolení nového tvaru musíte stisknout pøíslušné tlaèítko na lištì s ikonami.

Èáry se kreslí èernou barvou a tenkou èárou. Pokud budete chtít zmìnit barvu èáry, nebo její tloušku, stisknìte ikonu v pravé horní èásti okna s vyobrazením tužky. Zobrazí se vám další øada ikon pro zadávání barvy, typu a tloušky èáry. Po opìtovném stisku ikony s tužkou se nastavení skryje. Ikona tedy pracuje jako pøepínací tlaèítko. Kliknìte si proto na tlaèítko tak, aby jste mìli zobrazeny ikony pro nastavení èar. Vlevo je umístìna tabulka barev, uprostøed tlaèítka pro výbìr typu èáry a vpravo mùžete zadat tloušku èáry. Barvy se vybírají kliknutím na požadovanou barvu. Vybraná barva je oznaèena dvoupísmennou anglickou zkratkou barvy. Typ èáry mùžete vybrat plnou èáru, teèkovanou, èárkovanou, èerchovanou nebo mùžete posledním tlaèítkem úplnì zrušit. Tlouška èáry se udává v bodech. Pro znìnu tloušky mùžete zapsat do editaèního boxu pøímo novou hodnotu, nebo mùžete použít pro nastavení šipek. Pokud bude nastavena tlouška èáry vìtší než jedna, neuplatní se zadaný typ èáry a èára se bude vykreslovat vždy plná, pøípadnì se nebude kreslit vùbec. Zkuste si nastavit parametry èáry a provìøte si úèinky zmìny pøi kreslení základních geometrických tvarù.

Dosud jsme kreslili pouze okraje geometrických tvarù. Standardnì je totiž nastaveno, že se plocha kreslených geometrických tvarù nevykresluje. Podobným zpùsobem, jako se nastavují parametry kreslení èar máte možnost nastavit parametry vykreslování ploch. Ikony pro nastavení se zobrazí po stisku tlaèítka s vyobrazením štìtce. Opìt máte možnost zadat barvu, tentokrát plochy (výplnì). Souèasnì mùžete zadat typ výplnì. Máte možnost vybrat si buï kreslení plné plochy zadanou barvou, nevykreslování plochy, nebo z nìkolika druhù vykreslení plochy èárami. Lze vybrat èáry vodorovné, svislé, vodorovné i svislé souèasnì (møížka) a rùzné druhy diagonálních èar. Tlouška èáry výplnì je pøitom vždy jeden bod.

Nyní jste již schopni nakreslit na grafické ploše pouze za použití myši jednoduché obrázky. Vyzkoušejte si rùzné možnosti nastavení èáry a plochy. Parametry nastavení èar a plochy je možné zmìnit samozøejmì také pøímo z programu, pøípadnì zadat z pøíkazového okna. To si však ukážeme až v následujících lekcích. Nyní bude pro vás jistì zajímavá možnost uložení nakreslených obrázkù na disk do souboru pro pozdìjší použití a zpìtná

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

Pokud máte grafickou plochu zaplnìnou pøedchozími pokusy, mùžete provést její výmaz pomocí ikony s obrázkem prázdné stránky. Ikona se nachází v levé skupinì ikon umístìné v grafickém výstupním oknì. Po kontrolním dotazu bude grafická plocha vymazána. Stejného efektu lze dosáhnout povelem ImageClear zadaným v pøíkazovém oknì. Povel se zadává bez parametrù, v tomto pøípadì se provede výmaz grafické plochy již bez kontrolního dotazu.

Nakreslené obrázky máte samozøejmì možnost uložit do souboru. Používá se známý a bìžný bitmapový soubor typu *.BMP. Pro uložení stisknìte ikonu s obrázkem diskety. V dialogovém oknì zadejte adresáø a jméno souboru. Obrázek mùžete uložit samozøejmì i povelem ImageSave z pøíkazového okna. Napøíklad:

ImageSave("obrazek.bmp")

Obrázky mùžete i zpìt naèíst. Použijte ikonu s obrázkem šipky smìøující do stránky. V dialogovém oknì vyberte adresáø a jméno souboru typu *.BMP. Naèíst mùžete i obrázky vytvoøené v jiných grafických systémech. Po naètení obrázku je velikost grafické plochy nastavena dle naèítaného obrázku. Obrázek mùžete naèíst i povelem z pøíkazového okna ImageLoad. Napøíklad:

ImageLoad("obrazek.bmp")

Pokud by jste chtìli vytvoøený obrázek vytisknout, mùžete tak uèinit pomocí ikony zobrazující tiskárnu. Pro tisk obrázku z pøíkazového okna mùžete použít povel ImagePrint.

V následující lekci se seznámíte s možnostmi ovládání grafického výstupního okna pomocí povelù z pøíkazového okna.

6.lekce
Ovládání grafického okna z pøíkazového okna

Pøedchozí lekce vás seznámila krátce se základními možnostmi použití grafického výstupního okna. Nauèili jste se kreslit do grafické plochy pomocí myši, ukládat, naèítat a vytisknout vytvoøený obrázek. Nyní se seznámíte s dalšími možnostmi grafického okna a s parametry okna, které budete potøebovat pro programování grafických povelù. Poznáte, že vše, co bylo možné nastavit v grafickém oknì budete mít možnost zadat pomocí povelu z pøíkazového okna.

Èinnost grafického okna je možné si pøedstavit jako malíøské plátno definovaných rozmìrù, na které se kreslí perem (anglicky pen). Vìtší plochy je možné vybarvit štìtcem (anglicky brush). Kreslí se pøitom vždy nastavenou barvou. Systém obsahuje povely pro nastavení parametrù pera i štìtce. S možnostmi se seznámíte v následujícím textu.

Pøi kreslení obrazcù do grafického okna se kreslí obrazce èárou, jejíž typ je definován povelem ImagePenStyle a tlouška èáry je definována povelem ImagePenWidth. Plocha

nakreslených geometrických obrazcù je vyplnìna stylem zadaným povelem ImageBrushStyle.

Nejjednodušší je zmìna tloušky èáry, kdy uvádíte pøímo jako parametr povelu tloušku èáry v bodech. Pro zmìnu tloušky èáry na pìt bodù použijete z pøíkazového okna následující povel:

ImagePenWidth(5);

Všimnìte si, že pokud nastavíte sílu èáry povelem z pøíkazového okna, použije se nastavená síla èáry i pro následné kreslení pomocí myši pøímo v grafickém oknì. Obdobná vlastnost je platná i pro nastavení všech parametrù grafického okna. Máte proto možnost libovolnì kombinovat zadávaní povelù z pøíkazového okna nebo jejich nastavení pomocí ikon. V budoucnu budete mít samozøejmì možnost uvedené parametry nastavit i pøímo z programu.

Styl èáry máte možnost zadávat povelem ImagePenStyle. Pøeddefinováno je šest stylù. Jako parametr povelu musíte pøitom zadat definovaný styl èáry. Parametr mùžete uvést buï èíselnou hodnotou, nebo jménem konstanty dle následující tabulky:

hodnota	konstanta	název stylu
1	psSolid	souvislá èára
2	psDash	pøerušovaná èára
3	psDot	teèkovaná èára
4	psDashDot	èerchovaná èára
5	psDashDotDot	èerchovaná èára se dvìma teèkami
6	psClear	neviditelná èára

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

```
ImagePenStyle(1);       { souvislá plná èára }
ImagePenStyle(3);       { teèkovaná èára }
ImagePenStyle(6);       { neviditelná èára }

ImagePenStyle(psSolid);   { souvislá plná èára }
ImagePenStyle(psDot);     { teèkovaná èára }
ImagePenStyle(psClear);   { neviditelná èára }
```

Existuje samozøejmì i zpùsob nastavení barvy èáry. Nejprve se však budeme muset seznámit s možnostmi použití barev v poèítaèi. Zobrazování barev závisí na vlastnostech videokarty ve vašem poèítaèi. Používáte-li barevnou VGA kartu, máte možnost zobrazit minimálnì 16 barev. Po pøíslušném nastavení videoadaptéru je možné bìžnì zobrazovat 256 barev, výjimeènì i více. Vy budete mít možnost nastavit v systému libovolnou barvu z rozsahu 16 miliónù barev. Skuteènì zobrazená barva ale závisí na vlastnostech technického zaøízení, protože se zobrazí vždy barva nejbližší.

Hodnotu barev je možné zadávat dvìma zpùsoby, které lze v programu libovolnì kombinovat. Pokud budete používat pouze základní, šestnáctibarevnou paletu, mùžete 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 pomìr kombinace barev modré, zelené a èervené. Pro každou barvu je možné volit hodnoty v rozsahu 0 až 256. Násobek tìchto hodnot (modrá x zelená x èervená) udává výslednou barvu. Výhodné je používat tzv. hexadecimálního zápisu, kdy jsou pro každou barvu vyhrazeny dvì pozice èísla s hodnotami od 00 (èíslo 0) až do FF (èíslo 256). Pøi použití hexadecimálního èísla je nutné uvést pøed èíslem rozlišovací znak $. Viz tabulka hodnot barev.

hodnota	konstanta	název barvy
$000000	clBlack	èerná
$000080	clMaroon	kaštanovì èervená
$0000FF	clRed	svìtle èervená
$008000	clGreen	tmavì zelená
$008080	clOlive	tmavì žlutá
$00FF00	clLime	svìtle zelená
$00FFFF	clYelow	žlutá
$800000	clNavy	tmavì modrá
$800080	clPurple	tmavì fialová
$808000	clTeal	tmavì modrozelená
$808080	clDkGray	tmavošedá
$C0C0C0	clLtGray	svìtle šedá
$FF0000	clBlue	modrá
$FF00FF	clFuschsia	fialová
$FFFF00	clAgua	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 povelem ImageBryshColor. 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($0000FF); { svìtle èervená barva plochy }
ImageBrushColor($FF0000); { modrá barva plochy }

Pro nastavení stylu vyplòování ploch se používá povel ImageBrushStyle, kde se jako parametr povelu udává buï èíslem 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 èárami
4	bsVertical	vyplní oblast svislými èárami
5	bsFDiagonal	diagonální èáry \\\\\\\
6	bsBDiagonal	diagonální èáry ///////
7	bsCros	vodorovné a svislé èáry
8	bsDiagCross	vodorovné a svislé èáry diagonálnì

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

ImageBrushStyle(1); {plné vybarvení plochy}
ImageBrushStyle(3); {výplò vodorovnými èárami}
ImageBrushStyle(8); {výplò diagonálními èárami}

ImageBrushStyle(bsSolid); {plné vybarvení plochy}
ImageBrushStyle(bsHorizontal);{výplò vodorovnými èárami}
ImageBrushStyle(bsDiagCros); {výplò diagonálními èárami}

Aby bylo možné kreslit v grafickém oknì do pøesnì urèených pozic, musí být zadány souøadnice pro kreslení. Souøadnice urèují polohu jednotlivých bodù kresby. Souøadnice znamená, že musíte uvést vzdálenost v bodech od levého okraje grafického okna a vzdálenost v bodech od horního okraje grafického okna. Souøadnicový systém je tedy vztažen k levému hornímu rohu, který má souøadnici 0,0. Hodnoty ve smìru osy X narùstají smìrem doprava, hodnoty ve smìru osy Y narùstají smìrem dolù. Pøi zápisu souøadnice se uvádí nejprve osa x, potom osa y. Je pøitom možné zadávat pøíkazy pro kreslení mimo plochu grafického okna, zobrazí se však pouze ta èást, která je obsažena maximálními souøadnicemi grafického okna.

Aktuální souøadnice se zobrazují ve stavovém øádku systému vždy, když máte nastavenou myš nad grafickým oknem. Souøadnice je vhodné si vyzkoušet také na povelu Point, který slouží pro zobrazení bodu na zadané souøadnici. Povel nakreslí na zadaných souøadnicích bod o zadané velikosti. Bod se nakreslí aktuální barvou pera, kterou lze nastavit procedurou ImagePenColor. Vyzkoušejte si nìkolik pøíkladù pro seznámení se se zpùsobem oznaèování souøadnic 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);    { uprostøed, vìtší bod }
```

Souøadnice grafického okna se použijí i pro kreslení geometrických tvarù pomocí povelù. Možné je kreslit èáru povelem Line, obdélník povelem Rectangle, kružnici nebo elipsu povelem Ellipse a obdélník se zaoblenými rohy povelem RoundRect. Povelem Triangle je možné nakreslit trojúhelník, což není pomocí myši možné. Pomocí povelù je možné také kreslit povelem Arc èást køivky a povelem Pie kruhovou výseè. Vyzkoušejte si kreslení základních geometrických tvarù:

```
Line(20, 20, 50, 100);          { nakreslí èáru }
Rectangle(10, 10, 100, 100);    { nakreslí ètverec }
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 ètvrtkruh}
Pie(0,0,100,100, 50,100,100,50) {pravý dolní ètvrtkruh}
```

7.lekce
Výstup do grafického okna

V pøedchozí lekci jsme se nauèili kreslit do grafického výstupního okna z pøíkazového okna a nastavovat parametry èar a ploch. Nyní se zamìøíme na nastavení grafické plochy a pøevod èásti grafického okna ze zásobníku Windows a zpìt. Seznámíme se také s možností zápisu textu do grafického okna.

Až dosud jsme mìli velikost grafické plochy nastavenou vždy podle toho, jak veliké bylo grafické okno v okamžiku aktivace grafické plochy. Další zmìnou velikosti grafického výstupního okna se již velikost grafické plochy nemìnila. Pokud proto potøebujete nastavit velikost grafické plochy na požadované rozmìry, mùžete tak uèinit povelem ImageInit. Povel inicializuje grafické okno a nastaví jeho velikost na rozmìry zadané parametry. Nastaví souèasnì bílou barvu plochy, styl štìtce pro plnì vybarvené plochy, èernou barvu pera, sílu èáry na jeden bod. Inicializací se provede výmaz pùvodního obsahu grafického okna. Vyzkoušejte si napøíklad následující hodnoty:

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

Pokud potøebujete znát parametry již inicializované grafické plochy, mùžete použít dotazu GetMaxX a GetMaxY, který vrací velikost grafické plochy zadané strany. Pokud budete napøíklad potøebovat vykreslit bod uprostøed grafické plochy bez ohledu na velikost aktuálnì inicializované plochy, mùžete tak uèinit povelem:

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

V nìkterých pøípadech by bylo vhodné kreslit èáry do grafické plochy zadáváním v absolutních pøírùstkových hodnotách místo pøesné definice souøadnic. Proto je v grafickém oknì definován tzv. grafický ukazatel, který zaznamenává pozici vykreslení posledního bodu. Poloha grafického ukazatele se pøi použití nìkterých povelù automaticky mìní. Je možné ji nastavit i z programu povelem MoveTo. Èáry je možné potom zadat povelem LineTo definicí koneèného bodu. Využitím uvedených povelù je možné napøíklad nakreslit libovolný mnohostranný mnohoúhelník. Vyzkoušejte si následující povely, které by mìly v grafické ploše nakreslit ètyøúhelník:

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

Až dosud jsme dùslednì dodržovali, že je možné textové informace vypisovat zásadnì do textového výstupního okna a grafické informace do grafického výstupního okna. Je však možné provádìt výstup textových informací i do grafického okna. Nelze pøitom použít povelu WRITELN, se kterým jsme se již døíve seznámili. Zobrazovat lze pouze textové informace. Èísla je nutné pøedem pøevést na øetìzce. K výstupu textových informací se používá povelu TextOut. Jako parametry povelu se udávají souøadnice pro zobrazení textu v grafickém oknì a øetìzec, který se má zobrazit. Povel si mùžete vyzkoušet napøíklad následujícími pøíklady:

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

Text se pøi použití povelu TextOut vypisuje pøeddefinovaným fontem. Jeho zmìna se provede buï povelem ImageSetFont, kdy je možné zadat v dialogovém oknì nejen jméno fontu, ale pøímo i jeho styl a velikost. Další možností je použití ikony z grafického výstupního okna. Pokud potøebujete nastavit pouze nìkteré atributy fontu, mùžete tak uèinit povely ImageFontColor (barva fontu), ImageFontName (jméno fontu), ImageFontSize (velikost fontu) a ImageFontStyle (styl fontu).

Pøi nastavení barvy fontu se zadává u povelu ImageFontColor jako parametr buï èíslo barvy, nebo jméno konstanty udávající barvu. Tabulka hodnot barev je stejná jako pro nastavení barvy èar a ploch. U povelu ImageFontName, který nastavuje jméno fontu se uvádí pøímo jméno fontu. Pokud není zadané jméno v systému Windows dostupné, použije se font s podobným jménem. Velikost fontu se uvádí v povelu ImageFontSize pøímo požadovanou hodnotou. Pøíklad možných nastavení si prozkoušejte vèetnì výpisu textu po každé zmìnì parametrù fontu:

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

Pro nastavení stylu fontu se v povelu ImafeFontStyle požívají jako parametr pøeddefinované

hodnoty jednotlivých stylù. Uvádí se souèet hodnot požadovaného výsledného stylu:

hodnota	konstanta	popis stylu
0	fsNormal	normální písmo
1	fsBold	tuèné písmo
2	fsItalic	naklonìné písmo
4	fsUnderline	podtržené písmo
8	fsStrikeOut	pøeškrtnuté písmo

Pro nastavení naklonìného podtrženého písma mùžete zadat jednu z následujících možností:

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

Pokud budete potøebovat pøevést nakreslený obrázek pøes schránku Windows (clipboard) do jiné aplikace, mùžete tak provést pomocí povelu InageToClip, kdy uvedete jako parametry souøadnice ohranièené plochy pro pøevod. Pokud budete chtít napøíklad pøesunout do schránky obsah celého grafického okna, zadejte pøíkaz:

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

Naètení obsahu schránky se provádí povelem ImageFromClip, kdy se jako parametry uvedou souøadnice ohranièené plochy, kam se má obsah schránky pøevést. Pokud budete chtít pøevést obsah schránky Windows na celou plochu grafického okna, zadejte pøíkaz:

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

Nìkoho z vás již urèitì napadlo, že výše uvedené povely, pomocí kterých mùžete pøevádìt obrázky do schránky a naèítat zpìt ze schránky by bylo možné použít na pøesun èásti obrázku na nové místo. Na to je ale výhodnìjší použít samostatného povelu ImageMove, který provede obì akce najednou. Zadáváte pøitom jako parametry souøadnice místa, odkud se má naètení provést a souøadnice, kam se má plocha pøenést. Neodpovídá-li pøitom pomìr stran zdrojové a cílové plochy, nebude zkopírovaný obraz uøíznut, ale bude upraven (deformován) do zadaných souøadnic. To je možné využít k mnoha zajímavým efektùm. Napøíklad ke zvìtšení èásti plochy apod.

Nìkteøí jste již možná netrpìliví, kdy se zaène programovat. Místo programování jsme zadávali pouze povely do pøíkazového okna. Tím jsme se vás snažili nauèit nejprve používat nìkolik základních povelù programovacího jazyka. Pokud by jsme zaèali ihned se základy programování vèetnì popisu použití povelù bylo by toho najednou moc. Proto jsme si nejprve probrali základy používání povelù a pøíkazù jazyka a v pøíští kapitole již zaèneme opravdu programovat.

8.lekce
První program, procedura main

V pøedchozích lekcích jsme si probrali základní informace o používání pøíkazového okna pro zadávání povelù jazyka pøímo z klávesnice. Výstup výsledkù byl provádìn do textového nebo grafického okna. V následujících lekcích využijeme získané poznatky již pøímo pøi vývoji vlastních programù.

Programy se zadávají (zapisují) do samostatného okna, které slouží jako textový editor programu. Pokud je okno na obrazovce viditelné, mùžete jej aktivovat kliknutím myši na jeho plochu. V opaèném pøípadì využijte z menu volbu Okna/Program. Pokud budete chtít použít povelu z pøíkazového okna, budete muset zadat povel ProgramShow. Pro možnost automatického vytvoøení základní kostry programu stisknìte 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, pøípadnì stiskem funkèní klávesy F9 nebo kliknutím myši na ikonì s obrázkem bìžícího panáèka. Prázdný program obsahuje pouze základní kostru bez žádného povelu, který by vykonával viditelnou èinnost a proto program skonèí, aniž by cokoliv uèinil. Využijeme proto získaných zkušeností a doplníme text programu o nám již známé povely a pøíkazy. Dávejte si pozor na to, aby jste ponechali beze zmìny první a poslední øádek programu. Pokud by se tak nestalo, program by se nemusel vùbec spustit. Proveïte proto pouze doplnìní programu o pøíkaz WRITELN dle následující ukázky:

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

Program spustíme funkèní klávesou F9 a pokud máte viditelný obsah textového výstupního okna, zobrazí se v nìm pozdrav od systému. Vyzkoušejte si další, døíve probrané povely, které zapisujte pouze do èásti programu mezi slovy PROCEDURE a ENDPROC. Takto by napøíklad 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 zpùsobem mùžete zadat i povely pro použití grafického výstupního okna. Opìt použijeme pouze døíve probrané povely, které zapíšete do èásti programu mezi slovy PROCEDURE a ENDPROC. Text ve složených závorkách nemusíte zapisovat, jsou to poznámky, které nemají na èinnost programu vliv. O tom, jak pøebírat dále uvedené pøíklady z nápovìdì k programu se dozvíte podrobnì 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 mùžete nyní vyzkoušet. Projdìte si pøedchozí lekce a pokuste se z již získaných znalostí sestavit program, který napøíklad nakreslí dùm a vedle nìj zelený strom. Aby jste to nemìli tak složité, uvádíme zde èást programu pro nakreslení domu. Povely pro nakreslení strumu již doplòte sami. Samozøejmì mùžete v programu provést i další úpravy dle vlastního uvážení. Aby jste se také nauèili nìco nového z ovládání uživatelského prostøedí systému, zkuste tentokrát spustit program klávesou F6. Tomu, co uvidíte se øíká animace programu. Podrobnìji se s ní seznámíme až v nìkteré z dalších lekcí. Nyní krátce jen to, že animace mimo interpretace programu souèasnì v oknì s programem soubìžnì zobrazuje interpretovaný øádek 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 tmavì zelené}
  Rectangle(20,60, 105, 130);  {nakreslení stìny domu}
  ImageBrushColor(clYelow);   {nastavení žluté barvy}
  Rectangle(28, 70, 48, 90);   {horní levé okno}
  Rectangle(72, 70, 52, 90);   {horní støední okno}
  Rectangle(96, 70, 76, 90);   {horní pravé okno}
  Rectangle(28, 100, 48, 120);  {levé dolní okno}
  Rectangle(72, 100, 52, 130);  {dveøe}
  Rectangle(96, 100, 76, 120);  {pravé dolní okno}
  Ellipse(140,20, 170, 50);    {sluníèko}
  ImageBrushColor(clRed);     {nastavení èervené barvy}
  Triangle(20,60, 60,25, 105,60);{støecha}
ENDPROC
```

Takže jak vypadá program již víte. Mìli by jste být již také schopni sestavit jednoduchý program z posloupnosti vykonávaných povelù. Zatím jsme si ale nic neøekli o tom, co znamená první a poslední øádek programu, ve kterých jsou uvedena klíèová slova PROCEDURE a ENDPROC. To se dozvíte v nìkteré z následujících lekcích, ve které si vysvìtlíme, co všechno program mùže a musí obsahovat a jakou má mít strukturu.

9.lekce
Editace programu

Pøi zápisu programu se zapisuje text programu textovým editorem. Jedná se o tzv. programátorský editor, který neumožòuje nastavovat promìnný 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 vytvoøí nová prázdná struktura programu, kterou mùžete bìžným zpùsobem doplòovat o nové øádky programu. Již existující program mùžete otevøít stiskem kláves Ctrl+O, nebo volbou z menu Soubor/Otevøít. Program máte možnost uložit Stiskem kláves Ctrl+S nebo volbou z menu Soubor/Uložit. Pokud se jedná o nový program, musíte v dialogovém oknì zadat jméno souboru a tím i programu pro uložení. Pokud již zadaný soubor existoval, bude pøepsán novou verzí. Neprovádí se pøitom záložní kopie pùvodního souboru a tím samozøejmì i programu !

Pokud budete chtít uložit program pod novým jménem, mùžete tak uèinit volbou z menu Soubor/Uložit pod jménem. Budete dotázáni na jméno nového souboru, do kterého se program uloží. Pùvodní soubor zùstane pøitom zachován beze zmìny. Pøi práci se soubory a tudíž i s programy vždy platí, že mùže být v jednom okamžiku aktivní vždy pouze jeden. Nemùžete editovat najednou nìkolik programù.

Základy editace programu jsou obdobné, jako u všech textových editorù. Jakým zpùsobem zapsat nebo pøepsat text snad není nutné uvádìt. Stejnì tak jak se dostat na konec programu, procházet text poøádcích a podobnì. Seznámíme se ale s blokovými operacemi, které jsou pøi programování velmi výhodné. Pokud budete potøebovat pøesunout èást textu na jiné místo, budete si muset požadovaný blok textu nejprve oznaèit. To mùžete provést buï myší, kdy se pøesunete na zaèátek bloku, stisknete tlaèítko myši, držíte je stisknuté a pøesunete ukazatel myši na konec požadovaného bloku textu. Text zapsaný v bloku se pøitom zvýrazní. Z klávesnice mùžete blok textu oznaèit tak, že najedete kurzorem na požadovaný poèátek bloku textu, stisknete klávesu Alt, podržíte ji stisknutou a pøejedete kurzorem na požadovaný konec bloku textu. Po uvolnìní klávesy Alt máte oznaèen blok textu pro další operace. Pokud j budete chtít blok textu pouze vymazat, stisknìte nyní klávesu Delete. Oznaèený blok textu bude zrušen. Pozor proto pøi práci s oznaèeným blokem na uvedenou klávesu.

Pro pøesun oznaèeného bloku textu programu na jiné místo v programu budete muset použít schránku Windows. Pokud stisknete klávesy Ctrl+C zkopíruje se text z oznaèeného bloku do schránky a blok zùstane v textu stále zachován. Naopak, pokud stisknete klávesy Ctrl+X, bude blok textu pøeveden do schránky Windows (z textu programu bude zrušen). Pokud se pøesunete na požadované místo v programu, mùžete do nìj text ze schránky Windows zkopírovat. To se provede stiskem kláves Ctrl+V. Text ve schránce zùstává pøitom stále zachován do doby dalšího naètení bloku. Pozor však na to, že obsah schránky Windows se mùže v ostatních aplikací Windows zrušit. Pro pøesun textu mezi editorem a Windows jsou v menu systému zaøazeny pøíslušné volby v menu Editace.

Stejnì jako máte možnost pøesouvat text mezi schránkou Windows a programem, mùžete podobným zpùsobem pøevádìt text povelù zadaných v pøíkazovém oknì systému. Po dobu editace programu máte stále možnost používat pøíkazové okno a tím i napøíklad zkoušet parametry povelu. Až získáte správný výsledek, pøevedete odzkoušený povel pøes schránku Windows do programu. Teoreticky mùžete pomocí bloku pøevádìt do programu v pøípadì potøeby i obsah výstupního textového okna.

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

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

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

10.lekce
Promìnné a jejich typy

Když jsme si ukazovali, jak vypsat pøíkazem WRITELN èíslo do výstupního textového okna, bylo uvedeno, že není možné jednoduchým zpùsobem uložit výsledek výpoètu do pamìti. Je to proto, že to nelze z pøíkazového øádku provést. Musí se použít program, ve kterém bude hodnota definována. V této lekci si proto ukážeme, jaké typy hodnot mùžeme do pamìti ukládat.

Hodnoty uložené v pamìti se nazývají promìnné. To proto, že jejich hodnoty se mohou programem mìnit. Každou promìnnou musíme pøed jejím prvním použitím nejprve pojmenovat a definovat typ uložené hodnoty. Velmi zjednodušenì by se dalo napsat, že typ promìnné je buï èíslo, znakový øetìzec nebo logická hodnota. Pokud systém ví, s jakým typem promìnné má pracovat, je schopen provádìt prùbìžné kontroly hodnot a hlavnì zná, jaké místo má urèité promìnné v pamìti rezervovat.

Co je èíslo, ví urèitì každý. V systému je však definováno nìkolik typù èíselných promìnných, které se liší velikostí možných hodnot zpracovávaných èísel. Nìkteré typy èíselných promìnných jsou urèeny pouze pro celá èísla, jiné mohou obsahovat i desetinnou èást. Rozlišuje se také, že èísla mohou být buï pouze kladná, nebo i záporná. Požadovaný typ èíselné promìnné je proto vždy nutno peèlivì zvážit. Znakový øetìzec pøedstavuje seskupení libovolných znakù - písmen. Logická hodnota uchovává výsledky dotazu a mùže nabývat pouze hodnot pravda nebo nepravda.

Ještì døíve, než bude promìnná v programu poprvé použita, musí se v programu deklarovat. To znamená, že musíme uvést její jméno, typ a pøípadnì i poèáteèní hodnotu. Typ promìnné popisuje pøitom množinu hodnot, které mùže nabývat a operace, které se s ní mohou provádìt.

Základní promìnné jsou nejjednodušší promìnné, které reprezentují jeden výskyt prvku údaje. Ze základních typù promìnných jsou odvozovány další deklarace promìnných typù pole, záznam a tabulky, se kterými se seznámíme až pozdìji.

Podporované typy promìnných:
BYTE - celoèíselný 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 $2.9*10^{-39}$ do $1.7*10^{38}$
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 dvì 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 $2.9*10^{-39}$ do $1.7*10^{38}$

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 èíslice 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 dvì možné hodnoty a to True (pro stav pravda) a False (pro stav nepravda). Lze mu proto v programu pøiøadit buï 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 lekcích.

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 mimi pracovat.

11.lekce
Deklarace a používání promìnných

V pøedchozí lekci jsme se seznámili s tím, že prùbìžné hodnoty výpoètù se ukládají promìnných. Popsali jsme si také, jaké typy promìnných mohou existovat. V této lekci si ukážeme, jak promìnnou v programu deklarovat a jak ji používat.

Promìnné musí být v programu deklarovány ještì pøed svým prvním použitím ve výpoètech. V deklaraci promìnné se definuje, název promìnné a její typ s možností uvedení poèáteèní hodnoty. Název promìnné musí zaèínat písmenem a nesmí obsahovat mezeru. V programu se musí pøed deklarací promìnných uvést klíèové slovo VAR a na konci ENDVAR. V tomto bloku deklarace promìnných není možné používat žádné povely. Vyzkoušejte si následující pøíklad:

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

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

Promìnné a, b a c jsou deklarovány jako typ integer, to znamená, že mohou obsahovat pouze celá èísla. Promìnná d je typu real a mùže proto obsahovat reálné èíslo vèetnì desetinných míst. Promìnná s je øetìzec. V programu jsme promìnné nejprve v bloku oznaèeném slovy VAR a ENDVAR deklarovali. U promìnné a, b jsme definovali souèasnì poèáteèní hodnoty. V dalším bloku programu oznaèeném bloky PROCEDURE a ENDPROC jsme deklarované promìnné použili.

Hodnotu promìnné stanovíme jejím pøiøazením. To se provede pomocí dvojice znakù := mezi nimiž nesmí být mezera. Na levé straní je pøitom uvedena promìnná, jejíž hodnota má být zmìnìna a na straní pravé je uvedena pøiøazovaná hodnota. Hodnotu mùžeme uvést buï pøímo (èíslo, øetìzec) nebo jako výraz. Co je to výraz si pøesnìji probereme v dalších lekcích. Nyní bude staèit, pokud si budete pamatovat, že výraz mùže být napøíklad matematický výpoèet.

Po spuštìní výše uvedeného programu si systém nejprve z èásti VAR..ENDVAR definuje použité promìnné a jejich typ a v druhé èásti programu PROCEDURE..ENDPROC s nimi provádí zadané pøíkazy. Nejprve do promìnné c uloží dìlení a/b. Stejnì tak uèiní s promìnnou d. Po výpisu promìnných do výstupního okna si mùžete myslet, že výpoèet neprobìhl správnì. Obsah promìnných se liší, u promìnné c nelze vypsat desetinnou èást výsledku. Pokud si však zkontrolujete typ promìnné c, zjistíte, že je vše v poøádku. Promìnná je deklarována jako integer, což znamená pouze celé èíslo (kladné nebo záporné). Desetinná èást výsledku se proto neukládá a není možné ji také vypsat.

Po ukonèení programu zùstanou deklarované promìnné stále aktivní. Mùžeme je proto i po ukonèení programu dále používat a vypsat napøíklad pøíkazem WRITELN jejich obsah. Nyní máte proto možnost vyzkoušet si používání nadeklarovaných promìnných z pøíkazového okna. Novým spuštìním programu dojde k jejich deaktivaci (zrušení) a nahrazení novými promìnnými. Prozkoušejte si využití i jiných typù promìnných deklarovaných v programu.

Dosud jsme sestavovali naše programy tak, že vykonávaly pouze zadanou posloupnost akcí bez možnosti vynechání nìkterých akcí. V následující lekci se proto seznámíme s možností rozdìlení èinnosti programu na základì vyhodnocení zadané podmínky.

12.lekce
Podmínky v programu, logické výrazy

Jak jsme již uvedli v úvodu probíraných lekcí, je programování vlastnì sestavování posloupnosti vykonávaných akcí. V nìkterých pøípadech musíme mít ale možnost nìkterou z akcí buï vynechat, nebo provést akce jiné. Obojí na základì zadané podmínky. Napøíklad pokud mám v kapse více než dvacet korun, pùjdu do kina, jinak pùjdu domù na televizi. Zadaná podmínka by se dala v lidské mluvì popsat následujícím vztahem:

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

Uvedenému zápisu se øíká algoritmus a popisuje schematicky postup provádìných akcí. Vztahu hotovost>20 se øíká podmínka. Podmínka nám tedy urèuje, jaká akce se bude provádìt. Zápis algoritmu je nutné pro poèítaè pøevést do programu. Napøíklad následujícím zpùsobem:

```
VAR
  hotovost : integer;
ENDVAR

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

Nejprve jsme deklarovali promìnnou se jménem hotovost. Dále jsme jí v programu pøiøadili urèitou hodnotu. V podmínce jsme potom zjišovali, zda odpovídá našim požadavkùm a podle toho jsme provedli urèitou akci. Zkuste si zadat sami rùzné 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 Kè. Zkuste upravit program tak, aby jste mohli zadávat i vìtší èástky.

Závisí to na typu promìnné.

Všimnìte si, že jsme v programu u zápisu podmínky odsadili vykonávané pøíkazy na øádku o tøi znaky. Tím vynikla struktura podmínky a na první pohled je viditelné její rozèlenìní. Není to sice nutnost, pøesto však doporuèujeme uvedenou grafickou podobu zápisu programu ve vlastním zájmu dodržovat.

Podmínka se v programu zadává klíèovým slovem IF, za kterým musí následovat vyhodnocovaná podmínka. Klíèové slovo THEN není povinné, pøesto jej však doporuèujeme uvádìt, protože to vyžaduje klasický jazyk Pascal. Dále následují povely, které se provedou pouze pøi splnìní podmínky. Za klíèovým slovem ELSE se uvedou povely, které se provedou pøi nesplnìní podmínky. Zápis podmínky je ukonèen klíèovým slovem ENDIF. V zápise podmínky mùžete vynechat klíèové slovo ELSE vèetnì akcí pro nesplnìní podmínky. Vždy ale musíte uvést ukonèení podmínky klíèovým slovem ENDIF. Pokud se tak nestane, nahlásí systém chybu v programu.

Podmínku pøedstavuje logický výraz, který udává, zda je podmínka splnìna nebo ne. Logické výrazy mohou mít proto výsledek pouze pravda nebo nepravda. V poèítaèové terminologii True (pravda) nebo False (nepravda). Pokud by jste chtìli výsledek podmínky deklarovat jako promìnnou, museli by jste použít typ BOOLEAN. Logická podmínka zpracovává nejèastìji matematický výraz. Mùže to však být i výraz zpracovávající øetìzce, to však bude psáno až se nauèíme s rìtìzci dùkladnìji pracovat.

Matematické výrazy porovnávají nejèastìji nìkolik hodnot. Ve výrazu se pøitom mùže také použít libovolného matematického výpoètu, který je v jazyce definován.

Matematické výrazy zpracovávají aritmetické operace. Výrazy se skládají z operátorù a operandù. Operátor je pøitom porovnávaná hodnota a operand je zpùsob porovnání hodnot. Operátor mùže pøedstavovat libovolný matematický výpoèet, který je v jazyce definován. Operand slouží k vyhodnocení operátorù. Pøi zápisu dodržte, aby byly operátory oddìleny od operandù mezerami ! Používají se následující dostateènì známé operandy:

>	**vìtší než**
>=	**vìtší nebo rovno než**
<	**menší než**
<=	**menší nebo rovno než**
=	**rovno**
<>	**nerovno**

Ve výrazu je samozøejmì možné používat závorky. Pøíklad výrazù zpracovávajících matematický výpoèet:

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

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

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

V pøípadì použití logických operátorù mají tyto ve vyhodnocování výrazù pøednost pøed ostatními operátory. Dále se vyhodnocují závorky a až na konci matematické výpoèty. Pokud budete chtít zapsat v programu nìkolik výrazù spojených logickým operátorem, musíte umístit výrazy do závorek. Napøíklad pro zjištìní rozsahu hotovosti od 10 Kè do 30 Kè použijete následující zápis:

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

V této lekci jsme si probrali mimo možnosti rozvìtvení èinnosti programu také zpùsob zápisu podmínek v programu. Podmínky se v programu používají i v dalších pøíkazech. Napøíklad pro zadání poètu opakování zvolené èásti programu, jak si ukážeme v následující lekci.

13.lekce
Programové cykly (FOR, REPEAT, UNTIL)

Až doposud provádìly naše programy pouze zadaný sled povelù bez možnosti opakování požadovaných akcí. Pokud by jsme potøebovali napøíklad vykonat nìjakou èást programu stokrát, museli by jsme uvedenou èást programu zapsat stokrát. To by však bylo, jak sami jistì uznáte znaènì nevhodné. V této lekci si proto probereme, jak je možné na základì vyhodnocení podmínek opakovat zadanou èást programu.

Pokud budete potøebovat v programu provést nìjakou akci s pøedem známým poètem opakování, bude nejvhodnìjší použít cyklus typu FOR..ENDFOR. Jedná se o cyklus, ve kterém je definován tzv. èítaè, který udává kolikrát se má cyklus ještì provést. Èítaè je automaticky systémem zvyšován o zadanou hodnotu. Smìr naèítání mùže být pøitom nahoru, nebo dolù:

```
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 smìrem dolù}
    WRITELN(x:10);
```

```
  ENDFOR
ENDPROC
```

V uvedeném pøíkladì jsou v programu dva cykly FOR. První z nich zvyšuje stav èítaèe x smìrem nahoru, druhý smìrem dolù. Všimnìte si, že èítaè musí být sice deklarován jako promìnná, ale pøiøazení poèáteèní hodnoty se provádí až v definici cyklu. Koneèná hodnota èítaèe se pro smìr naèítání nahoru zadává za klíèovým slovem 'to'. Pro smìr naèítání dolù je urèeno klíèové slovo 'downto'. Pokud chceme zvyšovat, nebo snižovat stav èítaèe o jinou hodnotu než jedna, musíme požadovanou hodnotu uvést za klíèovým slovem 'step'. Hodnota øídící promìnné se nesmí v cyklu mìnit. Hodnoty cyklu jsou vyhodnocovány pouze jednou a to pøi spuštìní pøíkazu FOR.

Dùležitá poznámka: promìnná x je v pøedchozím pøíkladì deklarována jako typ real, to znamená že mùže nabývat i desetinných hodnot. Pokud by jste uvedli v tomto pøíkladì promìnnou x jako typ integer, zvyšoval by se èítaè vždy sice o hodnotu 0.5 ale protože typ integer je pouze celoèíselný, ke zvýšení èítaèe by nikdy nedošlo a program by skonèil v nekoneèné smyèce. Museli by jste jej restartovat pomocí volbou z menu Program/Restart programu.

Pokud budete chtít ve svém programu použít cyklus, jehož ukonèení bude záviset na splnìní zadané podmínky, máte možnost použít cyklus typu REPEAT..UNTIL. Vnitøní èást tohoto cyklu se provede vždy minimálnì jednou, protože podmínka ukonèení cyklu je uvedena až na jeho konci:

```
VAR
  x : real
ENDVAR

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

Pøíkazy uvedené mezi klíèovými slovy REPEAT a UNTIL se provádí tak dlouho, dokud není výraz definovaný na konci podmínky pravdivý. Nesmíte pøitom zapomenout na zvýšení hodnoty èítaèe, pøípadnì provést jinou akci, která mùže mít za následek ukonèení cyklu.

Jako další cyklus mùžete použít WHILE..ENDWHILE, který vyhodnocuje podmínku opakování cyklu na jeho poèátku. To umožòuje, že pokud není podmínka splnìna, neprovedou se pøíkazy uvedené mezi klíèovými slovy WHILE a ENDWHILE ani jednou. To mùže být v nìkterých pøípadech výhodné a je proto nutné se pøi 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 pøi používání cyklù platí vše co bylo uvedeno u popisu vìtvení 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 uvádìjí 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 pokraèoval 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 výsledkù

U programù v pøedchozích lekcích jsme vždy poèítali s tím, že jsou všechna data pro výpoèty zadána pøímo v programu. Aby jsme ale mìli možnost ovlivnit prùbìh výpoètu, bylo by vhodné zadávat požadované hodnoty pøímo dotazem od uživatele. To je možné dvìma zpùsoby. Buï použijeme pøíkaz READ, který umožòuje mimo jiné uživatelské zadaní hodnoty promìnné, nebo složitìjší postup, kdy použijeme interaktivní knihovnu pro tvorbu formuláøù.

Pøíkaz READ se používá pro vstup hodnoty promìnné ze souboru, nebo od uživatele z klávesnice. Hodnotu potom mùžeme použít dále v programu k výpoètùm. Promìnná musí být nejprve definována. Možné je použít pouze èíselné a øetìzcové promìnné. Ètení hodnoty ze souboru si ukážeme pozdìji, nyní si pøedvedeme, jak zadat promìnnou z klávesnice.

Pøíkaz READ vyvolá jednoduchý formuláø s možností zadání hodnoty promìnné. Jméno formuláøe je 'vstup' a pokud by jste chtìli umístit nad editaèní box pro zadání hodnoty promìnné váš text, museli by jste text vypsat pøíkazem WRITE (pozor, ne WRITELN !) ještì pøed použitím pøíkazu READ. Zadaný text se samozøejmì 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 výpoèty a zobrazení výsledkù. Následuje pøíklad použití pøíkazu READ.

```
VAR
  s : string;
  r : real;
ENDVAR

PROCEDURE main
  ConsoleClear;
  WRITE("Zadejte celé èíslo");
  READ(s);
  WRITELN();
  WRITELN("Zadal jste èíslo:",StrToInt(s):12);
  WRITELN();

  WRITE("Zadejte libovolné èíslo");
  READ(r);
  WRITELN();
  WRITELN("Zadal jste èíslo:",r:12:4);
  WRITELN();

  WRITE("Zadejte libovolný øetìzec");
  READ(s);
  WRITELN();
  WRITELN("Zadal jste øetìzec: ",s);
  WRITELN();
```

```
ENDPROC
```

POZOR !!!

V souèasné verzi nesmíte zadat pøi vstupu do èíselné promìnné nenumerický znak. Pokud tak uèiníte, bude výsledkem nulová hodnota. V dalších verzích bude upraveno.

Se složitìjším, ale efektnìjším zpùsobem se seznámíme pøi probírání interaktivní knihovny, která umožòuje definovat v programu vzhled i obsah formuláøe. Zkuste si proto zatím alespoò následující pøíklad:

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

ROCEDURE 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, ', pøeji pìkný den !');
    GetDate(rok2, mesic, den1, den2);
    WRITE("dnes je ", den1, "/", mesic, "/", rok2, ", ");
    WRITELN("je Vám ", rok2-rok1, " rokù.");
  ENDIF
ENDPROC
```

Takže jak zadávat obsah promìnné od uživatele již známe. Víme také, jak vypsat jejich obsah s pøípadnou další úpravou do textového výstupního okna. Zatím jsme se ale nic nedozvìdìli o tom, jak je možné provést výstup informací na tiskárnu. Systém sice neobsahuje žádné pøíkazy pro pøímý tisk na tiskárnu, dovoluje vám ale vytisknout kompletní obsah textového výstupního okna. Musíte proto nejprve provést výstup požadovaných informací a textù do výstupního okna a potom pøíkazem ConsolePrint vytisknout jeho obsah. Mùžete také po ukonèení programu, kdy se zobrazí požadované výsledky ve výstupním oknì provést jeho tisk pomocí ikony s obrázkem tiskárny, umístìné v horní lištì textového výstupního okna.

15. lekce
Vícenásobné vìtvení programu

V jedné z pøedchozích lekcí jsme se seznámili s pøíkazem IF..ENDIF, který slouží pro vìtvení èinnosti programu. Poznali jsme, že mùžeme èinnost programu rozdìlit do dvou vìtví, podle pravdivosti zadané podmínky. Pokud by jste však potøebovali rozdìlit èinnost programu podle výsledku na více vìtví, byl by zápis pomocí pøíkazu IF velmi nepøehledný. Proto systém obsahuje pøíkaz SWITCH, který mùžete použít pro rozdìlení èinnosti programu do libovolného poètu vìtví na základì vyhodnocení výsledku zadané podmínky. Pokud budete napøíklad potøebovat pro každé èíslo samostatnou akci, mùžete tak uèinit podle následujícího pøíkladu:

```
VAR
  x : integer
ENDVAR

PROCEDURE main
  ConsoleClear;
  FOR x := 1 to 10;
    SWITCH x OF           {porovnávej promìnnou x }
      CASE 1:           {pokud má hodnotu 1 }
        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 nìkterému z výrazù uvedených za klíèovým slovem CASE, bude proveden blok pøíkazù za tímto výrazem. Program bude potom pokraèovat po ukonèení konstrukce pøíkazu klíèovým slovem ENDSWITCH. Pokud se výraz nebude rovnat žádnému z porovnávaných výrazù, provede se blok pøíkazù uvedený mezi klíèovými slovy ELSE a ENDCASE.

Jedná se o pomìrnì dosti složitou konstrukci jazyka. Pro názorné pøedvedení proto mùžete program spustit klávesou F6 v animaèním režimu, kdy systém zobrazuje postupnì øádek programu, který právì vykonává. Všimnìte si pøitom, že øádky, které nevyhovují zadání jsou pøeskakovány. Pokud nebudete staèit sledovat animaci, mùžete 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 èástí. V první èásti uvedené mezi slovy VAR a ENDVAR jsme deklarovali dále používané promìnné. Ve druhé èásti se mezi klíèovými slovy PROCEDURE a ENDPROC uvádìly jednotlivé pøíkazy programu. Takovým èástem programu se øíká bloky. Zaèátek i konec bloku je vždy definován pøíslušnými klíèovými slovy pro zaèátek a konec bloku. V programu mohou být mimo deklarací a hlavního pøíkazového bloku uvedeny i další èásti, se kterými jsme se dosud neseznámili. Viz následující schématická struktura programu:

```
TYPE
  {definice datových typù - záznamù}
ENDTYPE

VAR
  {deklarace promìnných}
ENDVAR

PROCEDURE ...
  {deklarace uživatelské procedury}
ENDPROC

FUNCTION ...
  {deklarace uživatelské funkce}
ENDPROC

PROCEDURE main
  {tìlo hlavního programu}
ENDPROC
```

Definice datových typù

Datové typy jsou uživatelsky definované typy promìnných, které umožòují napøíklad seskupovat nìkolik promìnných a používat je jako jednu strukturovanou promìnnou. Podrobnìji budou popsány pozdìji.

Deklarace promìnných

Již znáte. Jsou uvedeny v bloku mezi klíèovými slovy VAR a ENDVAR. Obsahují deklarace promìnných, urèení jejich typu a pøípadné stanovení poèáteèní hodnoty. Podrobnìji se k nim ještì vrátíme pozdìji.

Deklarace uživatelských procedur

Procedury jsou uživatelsky definované pøíkazy, které umožní lépe rozèlenit program do logických celkù a vícenásobné použití èásti kódu. Podrobnìji se budeme procedurami zabývat v následující lekci.

Deklarace uživatelských funkcí

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

Tìlo hlavního programu

Musí být uvedeno v každém programu a musí být obsaženo v proceduøe se jménem main, kterou doporuèujeme umístit vždy na konec programu. Jejím provádìním se zaèíná vždy èinnost programu. Pokud by nebyla procedura se jménem main v programu nalezena, systém by nahlásil chybu a program by byl ukonèen.

Standardní jazyk Pascal dodržuje mnohem pøísnìjší formát zápisu programu. Platí pøitom obecné pravidlo, že každý prvek programu se musí v jazyce Pascal nejprve deklarovat a až potom se mùže používat. Bloky v programech v jazyce Pascal jsou vyznaèeny klíèovými slovy Begin a End místo námi používaného ukonèení ENDIF, ENDFOR, ENDPROC a podobnì. Námi používané øešení vychází z jazykù používaných pro programování databází. Má tu výhodu, že je mnohem pøehlednìjší a rychleji se jej nauèíte používat.

Bloková struktura programu se dodržuje i u dalších, døíve probraných pøíkazù. Jedná se o rozhodovací pøíkaz IF..ENDIF a všechny pøíkazy cyklù FOR..ENDFOR, REPEAT..UNTIL a WHILE..ENDWHILE. Všechny struktury programu musí být ukonèeny svým pøíslušným ukonèením. Bloky mohou být do sebe vnoøovány, nesmí však docházet k pøesahùm jejich koncù. Proto je vhodné dodržovat grafickou úpravu programu, kdy jsou podøízené èásti blokù odsazeny od svého poèátku a konce. Tím se dosáhne souèasnì pøehlednosti programu.

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

17.lekce
Deklarace a používání procedur

Doposud jsme v našich programech používali pouze pøíkazy zabudované v systému. Nyní si ale pøedvedeme, jak si mùžeme naprogramovat vlastní pøíkazy. Urèitì jste se již setkali s tím, že pokud by jste seskupili nìkolik øádkù programu, které se v uvedené sestavì v programu nìkolikrát opakují do jednoho, byl by program pøehlednìjší a kratší. Na to mùžete použít procedury, kterým se také nìkdy øíká podprogramy.

Procedury jsou volány z hlavního programu, vykonají pøíkazy zadané ve svém tìle procedury a po jejím ukonèení pøedají vykonávání programu zpìt do hlavního programu. Procedury jsou v programu oznaèeny klíèovými slovy PROCEDURE a ukonèeny klíèovým slovem ENDPROC, mezi nimiž jsou uvedeny výkonné pøíkazy procedury. Každá procedura musí být pojmenována. V programu se pøitom nesmí vyskytovat více procedur se stejným jménem. Následuje pøíklad jednoduché procedury:

```
VAR
  a : integer = 10;
  b : integer =  3;
ENDVAR

PROCEDURE nadpis
  WRITELN("***************************");
  WRITELN("* poèítá systém KLONDAIK * ");
```

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

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

V pøíkladì jsme definovali proceduru se jménem nadpis, která vždy mezi výpoèty vypíše tøi øádky nadpisu. V hlavní proceduøe main již potom staèí zadat volání procedury nadpis. Systém v tom okamžiku vypíše vždy tøi øádky 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(clYelow);
  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 pøedávanými parametry, je nutné v deklaraci procedury uvést za jejím jménem v závorce seznam použitých promìnných. V seznamu se uvádí jméno promìnné a za dvojteèkou její typ. Promìnné jsou v seznamu oddìlovány èárkami. Poèet deklarovaných promìnných v proceduøe musí pøitom vždy odpovídat poètu pøedávaných parametrù ve volání procedury.

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

Pokud v programu nadeklarujete svou vlastní proceduru a program spustíte, zachová si systém informace o deklarované proceduøe i po ukonèení programu. Mùžete potom proceduru použít z pøíkazového okna jako souèást systému. To je velmi výhodná vlastnost, protože vám umožòuje seznámit se dokonale s možnostmi vlastnosti jazyka.

Pomocí deklarace procedur dosáhnete toho, že si mùžete definovat své vlastní pøíkazy. Je pøitom možné volané proceduøe zadávat pøedávané parametry. Není ale možné, aby procedura vracela zpìt výsledek provedení procedury. To vám umožní až funkce, které budou probrány v následující lekci.

18.lekce
Deklarace a používání funkcí

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

Funkce je definovaná èást programu, která je volaná z výrazu a provádí urèité výpoèty, pøípadnì akce. Pøi ukonèení funkce nám vrací požadovaný výsledek, se kterým se ve výrazu, odkud byla funkce volána dále pracuje. Uveïme si proto pøíklad deklarace a použití funkce v programu, který nám provádí výpoèet mocniny zadaným exponentem (5^3 = 5 x 5 x 5, výsledek je 125).

```
VAR
  k : integer;
  v : real;
ENDVAR

FUNCTION umocni(x : integer, y : integer): integer
  v := x;         {pøevezneme základ}
  FOR k := 1 to y
     v := v*x;    {násobíme opakovanì základem}
  ENDFOR
  umocni := v;     {pøedáme 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 ukonèením, kdy je výpoèet funkce uveden mezi klíèová slova FUNCTION a ENDFUNC. V deklaraci funkce se navíc od procedury musí uvést za seznamem pøebíraných parametrù i typ výsledku, který funkce vrací.

Další odlišností je zpùsob vytvoøení hodnoty výsledku funkce. To je výsledku, který bude pøedán zpìt. Jsou pøitom dvì možnosti. První z nich je standardní a používá ji i jazyk Pascal a proto ji doporuèujeme preferovat. Na konci zpracování funkce zapíšeme pøiøazovací pøíkaz:

```
identifikátor := výraz;
```

kde na levé stranì pøiøazovacího pøíkazu je uvedeno jméno funkce a na pravé stranì její výsledná hodnota. Pozor na to, že identifikátor pøitom není deklarován jako promìnná. Je to proto, že stejnì jako si systém deklaruje sám promìnné, které se pøedávají jako parametry procedury a funkce, stejnì tak si systém deklaruje i promìnnou se jménem deklarované procedury.

Druhou možností pro pøedání výsledné hodnoty je pøíkaz RETURN s uvedením návratové hodnoty:

```
FUNCTION umocni(x : integer, y : integer): integer
  v := x;         {pøevezneme základ}
  FOR k := 1 to y
     v := v*x;    {násobíme opakovanì základem}
  ENDFOR
  RETURN v;       {pøedáme výsledek funkce}
ENDFUNC
```

Uvedené použití vychází z jazyka BASIC a databázových jazykù. Mùže být v nìkterých pøípadech 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 ukonèení programu. Mùžete potom funkci použít u pøíkazù z pøíkazového okna jako souèást systému. To je velmi výhodná vlastnost, protože vám umožòuje seznámit se dokonale s možnostmi vlastnosti jazyka.

19.lekce
Lokální a globální promìnné

V pøedchozích lekcích jsme v programu deklarovali promìnné v samostatném bloku umístìném na zaèátku programu. Nadeklarované promìnné pøitom platily po celou dobu programu a zùstávaly aktivní dokonce i po ukonèení programu. Tomu se øíká globální

platnost promìnné, protože je dosažitelná kdykoliv z libovolné procedury v programu. Opakem je promìnná deklarovaná pouze s lokální dobou platnosti. Pro dokonalejší seznámení se s lokálními a globálními promìnnými si upravíme døíve uvedený pøíklad 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;         {pøevezmeme základ}
  WRITELN("pøedávaná hodnota x:",x:4);
  WRITELN("lokální   hodnota v:",v:4);
  FOR k:= 1 to y
     v := v*x;    {násobíme opakovanì základem}
  ENDFOR
  umocni := v;     {pøedáme výsledku funkce}
ENDPROC

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

Všimnìte si, že program obsahuje dva bloky s deklarovanými promìnnými. Deklaraèní blok na zaèátku programu obsahuje pouze promìnnou 'v' s pøidìlenou hodnotou. Tato promìnná, protože je deklarována mimo jakékoliv procedury je globální a dosažitelná v celém programu. V definované funkci umocni je uveden znovu blok definice promìnných, kde je opìt deklarována promìnná se jménem 'v' (dokonce rozdílného typu). Tato promìnné je pouze lokální, to znamená, že je platná pouze ve funkci, kde byla deklarována. Pokud deklarujeme lokální promìnou stejného jména, jaké má již existující globální promìnná, zastíní lokální promìnná po dobu své platnosti døíve definovanou globální promìnnou.

Pokud výše uvedený program spustíme, bude se mimo výpoètu vypisovat i obsah zadaných promìnných. Všimnìte si, že pokud vypíšeme stav promìnné 'v' z hlavní procedury main, bude vypsána vždy hodnota, která byla pøidìlena promìnné pøi její deklaraci. Vypsána je tedy hodnota promìnné s globální platností v celém programu. Pokud se bude vypisovat hodnota promìnné 'v' z funkce umocni, bude se vypisovat hodnota lokální promìnné, která doèasnì pøekryje globální promìnnou. Mùžeme také vypsat hodnotu promìnné 'x', kterou jsme sami ani nedeklarovali. Deklaroval si ji opìt doèasnì sám systém pro pøevedení hodnoty z nadøazeného programu.

Možná se vám bude zpoèátku zdát používání a sledování lokálních a globálních promìnných

nepøehledné. Jejich používání ale umožní zpøehlednìní programu. V nìkterých pøípadech je správné pochopení rozsahu platnosti lokálních a globálních promìnných dokonce nutné. Je to napøíklad problém rekurze, kdy z tìla funkce voláme stejnou funkci (volání funkce sebe sama).

20. lekce
Rekurze v programu

Rekurzívní funkce (nebo procedura) bìhem provádìní pøíkazù svého tìla vyvolá ještì pøed jeho ukonèením sama sebe. Znovu se tedy zaène provádìt tatáž posloupnost pøíkazù, aniž by pùvodní byla dokonèena. Pokud budeme chtít napøíklad vypoèítat faktoriál nìjakého èísla, musíme postupnì vynásobit požadované èíslo všemi èísly nižšími vždy o jednièku. Faktoriál èísla 5 se napøíklad vypoèítá vztahem 5*4*3*2*1. Pokud budeme chtít pro výpoèet faktoriálu napsat obecnou funkci, mùžeme vztah upravit do následujícího tvaru: 5*(4*(3*(2*(1)))). Zde je již vidìt, že je možné pøi násobení snížit vždy základ o jednièku a násobit opakováním výsledku volané funkce. Funkce pro výpoèet 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 èísla 1 je:', factorial(1):5); {=   1}
  WRITELN('faktoriál èísla 2 je:', factorial(2):5); {=   2}
  WRITELN('faktoriál èísla 3 je:', factorial(3):5); {=   6}
  WRITELN('faktoriál èísla 4 je:', factorial(4):5); {=  24}
  WRITELN('faktoriál èísla 5 je:', factorial(5):5); {= 120}
  WRITELN('faktoriál èísla 6 je:', factorial(6):5); {= 720}
  WRITELN('faktoriál èísla 7 je:', factorial(7):5); {=5040}
ENDPROC;
```

Všimnìte si, že uvedený program neobsahuje žádnou deklaraci promìnných a pøesto je používá. Je to proto, že promìnné jsou deklarovány systémem v deklaraci funkce.

Má-li rekurzivnì aktivovaná funkce lokální promìnné, je pøi každé aktivaci vytvoøena nová sada lokálních promìnných, pøièemž pøi nové aktivaci zùstanou lokální promìnné nadøazených aktivací zachovány. Každá aktivace funkce má tedy své lokální promìnné. K promìnným nadøazených aktivací téže procedury èi funkce není pøístup možný. Po ukonèení aktivace zaniknou odpovídající lokální promìnné a platnými se stanou lokální promìnné nadøazené aktivace.

Existují úkoly, jejichž øešení vede na zápis programù s rekurzivními procedurami nebo funkcemi. Jsou to problémy, pøi jejichž rozkladu na podproblémy vznikne problém, který je

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

21. lekce
Návrh a zápis programu

V pøedchozích lekcích jsme se seznámili se základními pøíkazy systému a strukturou programu. Poznali jsme, že program tvoøí posloupnost operací, vìtvení programu podle podmínek a opakování èástí programu. Poznali jsme, že program tvoøí i data, která program zpracovává. Nyní by bylo vhodné seznámit se s tím, jak by mìl nový program vznikat.

Ještì døíve, než zaèneme psát nový program by jsme si mìli ujasnit požadovanou èinnost programu. Nejlépe tak, že se seznámíme nejprve s tím, jaké výsledky nám má program poskytnout. Podle toho si musíme zadat poèáteèní podmínky a vstupní hodnoty. Dalším krokem by již mìl být rozklad problému tak, aby jsme si v blocích definovali èinnost programu od poèáteèních hodnot až k požadovaným výstupním hodnotám. Pokud napøíklad zjistíme, že musíme nejprve provést výpoèet vstupních hodnot a potom výsledné hodnoty zobrazit, mùžeme již zaèít psát kostru programu:

```
PROCEDURE main
  vypocet;
  zobrazeni;
ENDPROC
```

Tím máme souèasnì zadány jména procedur, které musíme naprogramovat. Dále již budeme provádìt postupnì rozklad dílèích problémù na další podproblémy. V pøípadì složitých zadání se mùže dokonce stát, že nejsme schopni sestavit ihned program pro vyøešení dílèích problémù, ale musíme znovu definovat rozklad problému na menší související posloupné akce. Uvedeným zpùsobem se snažíme rozložit poèáteèní zadání problému do postupných krokù, které dále zjemòujeme. Na konci by jsme se mìli dostat do fáze, že øešení již zapisujeme pøímo v programovém jazyce. Soubìžnì s tím upøesòujeme požadovaná data a jejich uložení v promìnných definovaného typu.

Výše uvedenému zpùsobu øešení programù se øíká metoda návrhu programu shora dolù. To proto, že od celkového zadání pøecházíme postupnì k jednotlivým dílèím úkolùm, které nakonec zadáváme v programovém jazyce.

Velmi dùležité je také zaznamenání postupu øešení ve formì poznámek. Nyní se vám zdá všechno jasné. Pokud se však dostanete k programu po delší dobì, nemusí vám být ihned jasné, proè jste postupovali uvedeným zpùsobem. Zaznamenávejte si proto maximum poznámek k øešení problému. Poznámky mohou být v programu dvojího druhu. Pokud uvedeme na zaèátku øádky dvì lomítka za sebou, bude systém celý øádek ignorovat a bude jej považovat za poznámku. Takto je možné napøíklad psát delší komentáøe, nebo pøi rùzných pokusech mùžeme zneplatnit celý øádek programu, aniž by jsme jej museli vymazávat.

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

øádku pro zápis popisu provedené èinnosti øádku. Takto je možné oznaèit jako poznámku i text uprostøed øádku, nebo nìkolik øádkù najednou.

Pøi zápisu programu se snažte dodržovat grafickou úpravu programu, kdy jsou tìla blokù odsazena od svého záhlaví o nìkolik znakù. Takový program je potom i na první pohled lépe èitelný a snáze se hledají pøípadné chyby.

Systém nerozlišuje v programu malá a velká písmena. Pøesto bude vhodné, pokud budete dodržovat jednotnou dále popsanou úpravu. Nejvhodnìjší je používat pro návìští a ukonèení bloku (FOR..ELSE..ENDFOR) velká písmena. Stejnì tak je vhodné psát jména základních pøíkazù systému (READ, WRITELN, CONTINUE ..) velkými písmeny. Jména procedur a funkcí pøebíraná z knihoven systému je vhodné zapisovat tak, aby vždy první písmeno slova bylo velké. Složeným slovem se rozumí jméno složené z nìkolika slov, mezi nimiž nesmí být mezera. Napøíklad ImageInit, ConsoleClear apod. Stejným zpùsobem je vhodné pojmenovat vlastní víceznaková jména deklarovaných promìnných.

22. lekce
Ladící a animaèní režim programu

V pøedchozí lekci jsme se seznámili s tím, jak program navrhnout a sestavit. Nikdy však není zaruèeno, že vám bude program ihned pracovat správnì. Chybu v programu vám mùže ohlásit systém v pøípadì chybnì deklarované promìnné, špatnì zadaného pøíkazu a podobnì. Avšak i v pøípadì, že program probìhne až do konce bez vypsání 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 špatnì hledají, byl systém doplnìn nìkolika pomùckami, dovolujícím chyby v programu nalézt.

V jedné z pøedchozích lekcí jsme si zbìžnì ukázali animaci programu. Pøi ní se program spouští klávesou F6 pøípadnì volbou z menu Program/Animace, nebo kliknutím na ikonu s obrázkem filmu. Program pøitom bude mimo vykonávání naprogramované èinnosti souèasnì zobrazovat v oknì s programem øádku, kterou právì vykonává. Chod programu je souèasnì zpomalen tak, aby bylo možné prùbìh animace programu sledovat. To nám umožní kontrolovat prùbìh programu pøes jednotlivé pøíkazy, volání procedur a funkcí. Animaci programu máte možnost kdykoliv pøerušit tak, že stisknete klávesu F9 a program bude pokraèovat normální rychlostí bez zobrazování vykonávaného øádku programu. Pokud by jste chtìli animaci pouze doèasnì pozastavit, mùžete tak uèinit kliknutím myší na ikonu s obrázkem STOP znaèky.

Další, velmi zajímavou možností systému vzhledem k ladìní programu je schopnost prùbìžného vypisování hodnot promìnných. Promìnné se zobrazují ve svém samostatném oknì, které otevøete z menu volbou Okna/Promìnné. Zobrazí se vám tabulka se dvìma ikonami v horní lištì. Levá ikona se používá pro pøidání promìnné do tabulky, pravá ikona promìnnou z tabulky zruší. Zkuste si proto napsat program, ve kterém použijete cyklus FOR..ENDFOR. Mimo øídící promìnné deklarujte i další promìnné. Jména promìnných zadejte i do tabulky pro sledování promìnných. To uèiníte tak, že kliknete myší na levou ikonu, a do dotazovacího okna zadejte jméno promìnnou, kterou chcete sledovat. Po jejím zadání se zobrazí promìnné v tabulce s uvedením, že hodnota promìnné není definována. stejným

zpùsobem zadejte i další promìnné, jejichž stav budete chtít sledovat. Pøi spuštìní programu v režimu animace sledujte souèasnì tabulku s hodnotami promìnných. Dokud není v programu promìnné pøidìlena hodnota, zobrazuje se v jejich stavu tzv. nedefinovaná náhodná velièina. Po pøidìlení hodnoty a její každé zmìnì je stav hodnoty promìnné v tabulce aktualizován. Mùžete proto sledovat, zda odpovídá její hodnota pøedpokládaným velièinám.

Mimo režimu animace, kdy jsou øádky programu vykonávány automaticky se zadanou èasovou prodlevou máte možnost program krokovat sami po jednotlivých øádcích. Krokování programu po øádcích se aktivuje klávesou F7. Systém pøi každém stisku klávesy F7 provede pouze jeden øádek programu se souèasným zobrazením øádky, kterou bude provádìt v následujícím kroku. Máte tak možnost sami ovládat èasové prodlevy mezi provádìním jednotlivých øádek programu. Souèasnì máte také možnost výše popsaného zobrazení hodnot promìnných v pøíslušném oknì. Pokud se z programu volá procedura, pokraèuje krokování programu ve volané proceduøe. Pokud ale nechcete proceduru krokovat, stisknìte na øádku s procedurou místo F7 klávesu F8, která umožní skok do procedury pøeskoèit. Program v takovém pøípadì vykoná volanou proceduru bìžnou rychlostí bez zobrazování provádìných øádek programu. Obdobnì to platí i pro volání funkce v programu.

Pokud budete chtít sledovat prùbìh programu až od urèitého místa, nastavte kurzor v oknì s programem na požadované místo a spuste program funkèní klávesou F4. Program se spustí bìžnou rychlostí a po dosažení øádku programu, na kterém je kurzor pøejde do ladícího režimu. Dále budete mít možnost pokraèovat buï krokováním, animací nebo napøíklad po kontrole hodnot promìnných pokraèovat bìžnou rychlostí.

Výše popsaná animace programu i jeho krokování vám dovolí sledovat prùbìh èinnosti programu a zmìny hodnot promìnných. To vám umožní poznat dùkladnìji èinnost pøíkazù pro cyklus a vìtvení programu. Krokování i animaci programu máte možnost kdykoliv pøerušit s tím, že èinnost programu buï pozastavíte nebo spustíte bìžnou rychlostí. Pøerušení se provede kliknutím myši na ikonu se STOP znaèkou. Pokraèování programu bìžnou rychlostí je možné stiskem klávesy F9. Krokování programu, jeho animaci i bìžné spuštìní je možné libovolnì kombinovat.

Protože zùstává aktivní pøíkazové okno stále pøístupné, máte možnost kdykoliv zmìnit hodnotu promìnné, vypsat si obsah promìnných, které nemáte v oknì pro sledování promìnných apod. Tím dosáhnete absolutní nadvlády nad programem, což vám kompilované programy nikdy neumožní.

DOÈASNÝ ZÁVÌR

Další lekce obsluhy programu OZOGAN KLONDAIK se pøipravují a budou dodávány s plnou verzí programu. Budou obsahovat zejména popis interaktivní knihovnny umožòující tvorbu vstupních formuláøù, popis práce s textovými soubory, zpùsob práce se záznamy a podobnì.

Liberec 18.8.1997