

Programování v Pascalu

Ing. Jana Pšenčíková

Nakladatelství a vydavatelství

Computer Media[®]

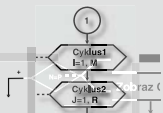
Vz dělávání, které baví
www.computermedia.cz

Obsah

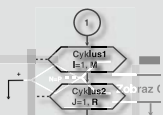
VYSVĚTLIVKY K PRVKŮM POUŽITÝM V KNIZE	7
POJMY A PRVKY POUŽITÉ V TEXTU.....	7
CO NAJDETE V TĚTO KNIZE	8
VĚNOVÁNÍ	8
SOFTWARE A JEHO ŽIVOTNÍ CYKLUS.....	9
ŽIVOTNÍ CYKLUS VÝROBKU	9
Zadání	10
Vývoj/Konstrukce	10
Výroba	10
Testování / výstupní kontrola	10
Dokumentace	10
Prodej a předání zákazníkovi	10
Provoz	11
Sběr nových požadavků a příjem reklamací	11
A všechno začne znovu	11
ŽIVOTNÍ CYKLUS SOFTWARE	11
Zadání	11
Analýza a návrh řešení (algoritmizace)	12
Programování	12
Testování / výstupní kontrola	12
Dokumentace	13
Prodej a předání zákazníkovi – zaškolení a ověřovací provoz	13
Ostrý provoz	13
Sběr nových požadavků a příjem reklamací	13
Zase znovu	14
ŽIVOTNÍ CYKLUS SOFTWARE VE ŠKOLNÍCH ÚLOHÁCH A PROJEKTECH	14
Co bude ve školních úlohách a projektech jiné a co se musí zachovávat	14
PROGRAMOVACÍ JAZYKY A METODY PROGRAMOVÁNÍ	15
CO JE PROGRAM A CO JE PROGRAMOVACÍ JAZYK	15
ROZDĚLENÍ PROGRAMOVACÍCH JAZYKŮ	15
Nižší a vyšší programovací jazyky	15
Překládané (kompilované) a interpretované programovací jazyky	16
PROGRAMOVACÍ METODY	17
Strukturované programování	17
Objektové programování	17
Vizuální programování	17
Programovací jazyk Pascal	17
CO MUSÍTE ZNÁT NEŽ ZAČNETE PROGRAMOVAT	18
STRUKTURA PROGRAMU.....	18
Hlavička programu	18
Tělo programu	19
CO VŠECHNO MŮŽETE NAJÍT V PROGRAMU.....	19
Klíčová slova	19
Identifikátory	20
Čísla	20
Celá čísla v desítkové soustavě	21
Racionální čísla v pevné desetinné čáře	21
Racionální čísla v plovoucí čáře	21
ČÍSLA V ŠESTNÁCTKOVÉ SOUSTAVĚ	22
Znaky a textové řetězce	23
Matematické a logické operátory, přiřazovací příkaz	23
Názvy funkcí a procedur jazyka Pascal	24
Komentáře	24
DATOVÉ TYPY	24
Logické datové typy	25
Celočíselné datové typy:	25
Desetinné datové typy	25
Znakové typy	25
Ordinální datové typy	25
PRAKTICKÝ POSTUP PŘI TVORBĚ PROGRAMU.....	26
Typy chyb	26
Psaní programu	27
Překlad	27
VÝVOJOVÉ PROSTŘEDÍ	27
Spuštění programu Free Pascal	27

Nastavení adresáře pro ukládání souborů	28
Vytvoření nového souboru	28
Uložení souboru	28
Ukázka vývojového prostředí Free Pascal	29
Překlad	30
Odstraňování syntaktických chyb	30
Příprava dat pro ladění programu	31
Ladění programu	32
TVORBA NEJJEDNODUŠŠÍCH PROGRAMŮ	34
KOMENTÁŘ	34
PŘÍKAZY VSTUPU A VÝSTUPU	35
Read, readln	35
Write, writeln	35
Jednoduchá sekvence s jedním sekvenčním blokem – příkaz výstupu	36
Sekvence se dvěma sekvenčními bloky	37
Součet obsahu dvou buněk, přiřazovací příkaz	37
Výměna obsahu dvou buněk	38
Další typické sekvenční programy	39
Rozdíl	39
Součin	40
Obdélník – obvod a plocha	41
Obvod kružnice a plocha kruhu	41
Formátování výstupů	42
Obvod kružnice a plocha kruhu – formátovaný výstup	43
Objem a plocha válce	44
Rovnostranný trojúhelník – obvod a plocha	44
Šestiúhelník	45
Pro hloubavé: Výměna hodnot ve dvou buňkách bez pomocné buňky	46
Chyták - Pythagorova věta – výpočet přepony pravouhlého trojúhelníka	46
PRÁCE S ORDINÁLNÍMI DATOVÝMI TYPY	47
Funkce nad ordinálním typem Celé číslo	47
Funkce nad ordinálním typem Znak	48
VĚTVENÍ	49
SYNTAXE VĚTVENÍ	49
Úplný příkaz větvení	49
Neúplný příkaz větvení	51
OŠETŘENÍ NEŽÁDOUCÍCH DŮSLEDKŮ	52
Podíl	52
Obecnější výraz s dělením	53
Odmocnina	53
Obecný výraz	54
VÝRAZY S ABSOLUTNÍ HODNOTOU, VLASTNOSTI ČÍSEL	55
Absolutní hodnota	55
Zjištění, zda je číslo kladné či záporné	56
Zjištění, zda je číslo sudé či liché	56
Dělitelnost	57
Procedura Exit	58
POROVNÁVÁNÍ A ŘAZENÍ ČÍSEL, MAXIMUM A MINIMUM	59
Porovnání dvou čísel podle velikosti – bez pomocné buňky	59
Seřazení dvou čísel s pomocnou buňkou	59
Největší ze tří čísel – bez pomocné buňky	60
Maximum ze tří čísel s použitím pomocné buňky	61
Maximum ze tří čísel s použitím dočasného maxima	62
Seřazení tří čísel podle velikosti bez pomocné buňky	63
Seřazení tří čísel s použitím pomocné buňky	64
Seřazení tří čísel podle velikosti s respektováním výsledku předchozího kroku	65
Seřazení čtyř čísel podle velikosti – s pomocnou buňkou	66
ÚLOHY Z GEOMETRIE	67
Trojúhelník	67
Test na trojúhelník rovnoramenný, rovnostranný, obecný	69
Case – vícenásobné větvení	70
Podprogram typu Procedura	71
Test na trojúhelník rovnoramenný, rovnostranný, obecný – řešení s procedurou a vícenásobným větvením	73
Test na pravouhlý trojúhelník	75
KOMBINOVANÉ ALGORITMY	76
Lineární rovnice	76
Soustava dvou lineárních algebraických rovnic	77
Kvadratická rovnice s reálnými kořeny	78
Kvadratická rovnice s komplexně sdruženými kořeny	79
Funkce Uppcase	80

Rychlost, dráha, čas.....	80
Kalkulačka.....	82
Pohyb rovnoměrně zrychlený.....	83
Vlaky.....	84
CYKLY	86
CYKLUS A JEHO TYPY	86
Cykly s pevným počtem opakování – cyklus For.....	86
Cyklus řízený podmínkou – podmínka je na začátku cyklu - While.....	87
Cyklus řízený podmínkou – podmínka je na konci cyklu - Repeat.....	88
Příkaz Keypressed.....	88
Čekací smyčka.....	88
ÚLOHY S OPAKOVÁNÍM	89
Paralelní odpory.....	90
Kalkulačka.....	91
SUMY, PROHLÉDÁVÁNÍ ŘADY ČÍSEL, MAXIMUM A MINIMUM.....	92
Zobrazení čísel od jedničky do desítky.....	92
Zobrazení čísel od dvojky do dvacítky, jen sudé.....	93
Suma čísel od 1 do 10.....	93
Suma 10 různých čísel, resp. libovolného počtu čísel.....	94
Suma N čísel.....	94
Maximum z deseti kladných čísel.....	95
Vylepšený algoritmus.....	96
Maximum a minimum z celých čísel, jejichž počet bude zadán předem.....	96
Suma čísel, jejichž počet není znám předem – s testem na začátku.....	97
Suma čísel, jejichž počet není znám předem – s testem na konci cyklu.....	98
Maximum z neznámého počtu kladných čísel.....	99
Maximum a minimum z neznámého počtu celých čísel.....	100
PROHLÉDÁVÁNÍ ŘADY ČÍSEL A ZNAKŮ	101
Zjištění, kolik čísel je kladných a kolik záporných – celkový počet znáte.....	101
Zjištění, kolik čísel je kladných, kolik záporných. Přejde-li nula, pak skončí.....	102
Zjištění, kolikrát se v textu objeví zadané písmeno – text zadáván po znacích.....	103
Datový typ řetězec (string).....	103
Zjištění délky textového řetězce.....	104
Zjištění, kolikrát se v textu objeví zadané písmeno – text zadán najednou.....	104
Zjištění počtu slov ve větě – znak po znaku.....	105
Zjištění počtu slov ve větě – načte se celá věta najednou.....	106
Součin pomocí součtu.....	106
Dělení pomocí odečítání.....	107
Největší společný dělitel.....	108
Součet číslic v čísle.....	109
Test, kolikrát se vyskytuje určitá číslice v daném čísle.....	110
Datový typ pole (array).....	111
ODDECHOVÉ ÚLOHY.....	113
Písnička „Když jsem já sloužil“.....	113
Společenská hra „Severní pól“.....	115
Zajíci a bažanti.....	116
ČÍSELNÉ SOUSTAVY A PŘEVODY MEZI NIMI	117
Převod z desítkové soustavy do dvojkové – nové cifry se vypisují od nejnižších.....	117
Převod z desítkové soustavy do dvojkové – nové cifry se vypisují od nejvyšších.....	118
Podprogram typu funkce.....	119
Převod čísla z dvojkové soustavy do desítkové.....	120
ŘADY – ARITMETICKÉ, GEOMETRICKÉ, DALŠÍ	121
Aritmetická řada – výpočet hodnoty prvků řady, prvky se zobrazují průběžně.....	121
Aritmetická řada – výpočet hodnoty prvků řady, všechny prvky se zobrazí nakonec.....	122
Aritmetická řada – součet.....	123
Geometrická řada – výpočet hodnoty prvků řady, prvky se zobrazují průběžně.....	124
Geometrická řada – výpočet hodnoty prvků řady, všechny prvky se zobrazí nakonec.....	125
Geometrická řada – Suma.....	126
Složitě úrokování.....	127
Stavební spoření.....	128
Faktoriál.....	130
Výpočet faktoriálu pomocí rekurentní funkce.....	131
Co se děje při rekurzi.....	132
Program pro výpočet Ludolfova čísla π	134
Program pro výpočet hodnoty přirozeného logaritmu čísla 2.....	135
Program pro výpočet funkce sinus x pomocí mocninné řady.....	136
Program pro výpočet funkce ex pomocí mocninné řady.....	137
OPERACE S VEKTORY A MATICEMI	138
Součet vektorů.....	138
Skalární součin vektorů.....	139
Minimum z řady čísel, metoda přímého výběru – řešení pomocí vektoru.....	140



Maximum z řady čísel, metoda probublávání – řešení pomocí vektoru	141
Maticе – načtení prvků	142
Maticе – počet kladných a záporných prvků	143
Největší a nejmenší prvek matice	144
Pozice největšího a nejmenšího prvku matice	145
Trojúhelníková matice	147
Diagonální matice	149
Maticе otočená kolem hlavní diagonály	151
Součet matic	152
Součin matic	153
TŘÍDICÍ ALGORITMY	155
Select Sort	155
Bubble Sort (třídění probubláváním) – zjednodušený	157
Bubble Sort (třídění probubláváním) – klasický	158
Shake Sort (třídění přetřásáním)	159
DATOVÝ TYP ZÁZNAM (RECORD)	161
Program na použití záznamů	162
DATOVÝ TYP SOUBOR	164
Typy souborů	164
Deklarace souborů	164
Vysvětlení struktury	165
Práce se soubory	165
Otevření souboru	165
Práce se souborem	167
Užitečné příkazy pro práci se všemi soubory	167
Příkazy pro práci s typovými soubory	167
Příkazy pro práci s textovými soubory	167
Zavření souboru	167
Program na použití typového souboru	168
SOUHRNNÉ INFORMACE	170
STRUKTURA PROGRAMU	170
KLÍČOVÁ SLOVA	171
IDENTIFIKÁTORY	171
ČÍSLA	172
ZNAKY A TEXTOVÉ ŘETĚZCE	174
DATOVÉ TYPY	176
OPERACE , SE KTERÝMI SE MŮŽETE V PROGRAMU SETKAT	180
PROCEDURY VSTUPU A VÝSTUPU	182
FORMÁTOVÁNÍ VÝSTUPŮ	183
KOMENTÁŘE	184
ZÁKLADNÍ STRUKTURY PROGRAMU	184
PODPROGRAMY	189
UŽITEČNÉ FUNKCE KNIHOVNY CRT	191



CO NAJDETE V TÉTO KNIZE

Tato kniha je „dvočetem“ nedávno vydané knihy **Algoritmizace** – vlastně je její praktickou částí. Nejsilnější stránkou knihy **Programování v Pascalu** jsou bohatě komentované zdrojové texty programů (programy), které jsou vypracovány podle algoritmů z knihy **Algoritmizace**. Najdete zde více než 100 zdrojových textů.

Cílem knihy není dokonale naučit čtenáře určitý programovací jazyk, ale seznámit ho se zákonitostmi programování do takové hloubky, aby si v budoucnu dokázal rychle osvojit každý další programovací jazyk. Vývoj v oblasti informačních technologií jde totiž tak rychle dopředu, že nemá význam učit encyklopedické znalosti, které zastarají dřív, než přijdou studenti do praxe, ale naučit je myslet určitým způsobem.

Kniha obsahuje tyto kapitoly:

Software a jeho životní cyklus

- Nejprve se seznámíte s obecným životním cyklem výrobku, s fázemi, kterými musí každý výrobek při svém vývoji a výrobě projít.
- Poté jsou obecné fáze životního cyklu výrobku aplikovány na software a jeho životní cyklus. Dozvíte se, co všechno se musí udělat, aby vznikl software, který je použitelný pro zákazníka, kam patří například algoritmizace a programování.
- Na závěr kapitoly budou formulovány zásady pro tvorbu školních úloh a projektů, které vyplývají z životního cyklu software.

Programovací jazyky a metody programování

- Snad každý intuitivně tuší, co je programovací jazyk, ale pro jistotu se hned na začátku představy sjednotí.
- Dále se dozvíte, jaké jsou typy programovacích jazyků a podle jakých kritérií je lze rozdělit.
- Budou popsány základní programovací metody, dozvíte se, co je to strukturované programování, objektové či vizuální programování.
- Výběr programovacího jazyka - na závěr kapitoly bude vysvětleno, proč byl zvolen právě výukový jazyk **Pascal**.

Co musíte znát, než začnete programovat

- Seznámíte se nejprve s nejhrušší strukturou programu.
- Poté bude věnována pozornost všemu, co se může objevit v programu (například identifikátory, čísla, znaky, klíčová slova...), a na kterém místě se to může objevit. Syntaxe jazyka je vysvětlena jednoduše a jen do takové hloubky, abyste mohli začít pracovat. Podrobnosti jsou vysvětlovány až v dalších kapitolách, kde zároveň následují příklady.
- Pak následuje krátká instruktáž do datových typů – jen tolik, kolik budete potřebovat v nejbližší době (složitější datové typy jsou vysvětlovány průběžně až v místě, kde je k nim zrovna uveden příklad).
- Na závěr kapitoly se seznámíte s vývojovým prostředím jazyka Pascal a s praktickými kroky, které budete potřebovat k vytvoření programu a odladění programu.

Pak následují tři kapitoly:

Tvorba nejjednodušších programů – zde najdete jednoduché programy a základní programátorské obraty založené na sekvenčních algoritmech.

Větvení – typické programy a programátorské obraty, které jsou založeny na větvení.

Cykly – tvorba typických programů na všechny tři typy cyklů, které jazyk Pascal podporuje.

Tyto tři kapitoly tvoří stěžejní část knihy. Obsahují zdrojové texty programů s bohatými komentáři a vysvětlivkami – jsou prakticky 1:1 k vývojovým diagramům z knihy **Algoritmizace**. Syntaxe jazyka je probírána nenásilnou formou v průběhu řešení. Datové typy jsou v této části vysvětlovány pouze tak, jak jsou zapotřebí při tvorbě programů.

Souhrnné informace – v poslední kapitole najdete přehledný popis všech důležitých pojmů jazyka Pascal, které se vám mohou hodit, například přehled datových typů a jejich deklarace, ASCII tabulka, přehled formátování, klíčová slova a jejich význam, typy podprogramů atd. Vše je co nejstručnější, formou „taháku“.

CD s podklady k příkladům

Součástí knihy je i doprovodné CD, které obsahuje jednak samotné zdrojové kódy programů a také potřebný software, ve kterém je možné příklady programování zpracovávat.



Software a jeho životní cyklus

CÍL KAPITOLY: SEZNÁMENÍ S JEDNOTLIVÝMI FÁZEMI VÝVOJE SOFTWARE

Důležité pojmy:

- Životní cyklus výrobku
- Životní cyklus software:
zadání, analýza a návrh řešení, programování, testování, prodej/předání zákazníkovi, ověřovací provoz, ostrý provoz, sběr nových požadavků
- Dokumentace na všech úrovních vývojového cyklu
- Životní cyklus software ve školních úlohách a projektech

ŽIVOTNÍ CYKLUS VÝROBKU

Název této kapitoly zní na první pohled učeně, ale posléze zjistíte, že se jedná o pojmy, se kterými jste se už v nějaké míře setkali.

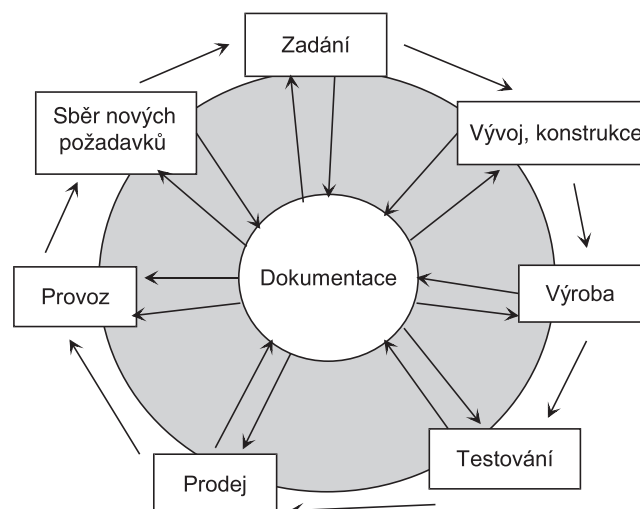
V matematice jste se už na základní škole setkali se slovními či konstrukčními úlohami. Při jejich řešení po vás učitelé požadovali tyto body:

- Zadání,
- Rozbor řešení,
- Diskuse (vymezení podmínek řešitelnosti)
- Vlastní řešení (což byl buď výpočet nebo konstrukce, podle typu úlohy),
- Zkouška (zpětné dosazení výsledků do původního zadání),
- Odpověď.

Někteří žáci považovali tyto zvyklosti za zbytečné a otravné (k těmto vy určitě nepatříte). Divili se, že jim učitel strhl body v písemce, i když měli všechno správně vypočítáno, jenom neměli nikde uvedeno,

- co vlastně počítají,
- proč to počítají právě tak – jak přišli na princip řešení,
- za jakých okolností bude výsledek platit,
- neověřili si správnost výsledku,
- nebo nakonec nenapsali, co je výsledkem jejich snažení.

Tento standardní postup neplatí jenom v matematice, ale uplatňuje se prakticky při vzniku každého výrobku. Říká se mu životní cyklus výrobku a zjednodušeně vypadá takto:



Programovací jazyky a metody programování

CÍL KAPITOLY: ZÍSKÁNÍ PŘEHLEDU O TYPECH PROGRAMOVACÍCH JAZYKŮ

Důležité pojmy:

Co je programovací jazyk

Rozdělení programovacích jazyků

- Vyšší a nižší
- Překládané a interpretované

Programovací metody

- Strukturované programování
- Objektové programování
- Vizualní programování

Jazyk Pascal – jeho vlastnosti, proč se začíná při výuce programování právě jím

CO JE PROGRAM A CO JE PROGRAMOVACÍ JAZYK

Při výuce algoritmizace jste se dozvěděli, že program je jedním z možných zápisů algoritmů. Má tu vlastnost, na rozdíl od ostatních forem zápisů, že mu kromě člověka rozumí i počítač. Program však ještě není psán v jazyce počítače, tím je strojový kód. Vytvářet program přímo ve strojovém kódu by sice šlo také (tak se pracovalo kdysi dávno na prvních počítačích), ale bylo by to nesmírně pracné a zdlouhavé.

Právě pro usnadnění komunikace člověka – programátora s počítačem byly vyvinuty programovací jazyky.

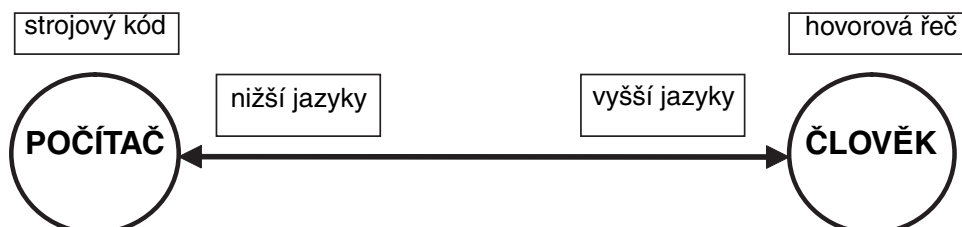
ROZDĚLENÍ PROGRAMOVACÍCH JAZYKŮ

Programovací jazyky lze rozdělit podle několika různých hledisek, například:

- Na nižší a vyšší programovací jazyky.
- Na jazyky překládané (kompilované) a interpretované.

Nižší a vyšší programovací jazyky

Toto rozdělení nemá nic společného s „hloupějším“ a „chytřejším“ programovacím jazykem, ale s tím, **jak blízko má jazyk ke strojovému kódu**. Jejich postavení lze znázornit níže uvedeným obrázkem.



Rovněž se nedá říci, který typ je „lepší“ či „horší“, každý se hodí na něco jiného a špatný bude pouze v případě, když se nevhodně použije.

Pokud se například budete chtít rychle dostat z Prahy do Bruselu, nejspíš použijete letadlo. Budete-li se chtít dostat z jednoho velkého města v rámci naší republiky do jiného, pak nejspíš pojedete rychlíkem či dálkovým autobusem. Pro cestování z jedné menší vesnice do jiné bude nejlepší osobní vlak či obyčejný autobus (letišť tam nemají a rychlík ani dálkový autobus tam nezastavuje). Půjdete-li někam, kam nevede ani silnice, pak nezbude než jít pěšky.

Co musíte znát než začnete programovat

CÍL KAPITOLY: SEZNÁMENÍ SE STRUKTUROU JAZYKA PASCAL

Důležité pojmy:

- Struktura programu – hlavička a tělo programu
- Co všechno můžete najít v programu:
 - Klíčová slova
 - Identifikátory
 - Čísla
 - Znak a textové řetězce
 - Matematické a logické operátory, přiřazovací příkaz
 - Funkce a procedury jazyka Pascal
 - Komentáře
- Datové typy
- Praktický postup při tvorbě programu
 - Činnosti ve fázi Programování – psaní programu, ladění, překlad
 - Typy chyb – syntaktické a logické
 - Integrované vývojové prostředí

STRUKTURA PROGRAMU

Představíte-li si jakýkoliv technologický postup, pro názornost například recept z kuchařské knihy, vidíte, že obsahuje

- název receptu (jídla),
- seznam surovin a polotovarů, včetně jejich množství,
- a teprve pak postup (algoritmus) k přípravě pokrmu.

Recept totiž musí být nějak označen, aby se rozlišil od ostatních. Předtím, než se začne vařit, si kuchař nejprve zjistí, co bude potřebovat za suroviny, a nachystá si je.

Při tvorbě programu je to podobné. Program v Pascalu (resp. TurboPascalu, dále jen Pascalu), stejně jako ve většině programovacích jazyků, se skládá ze dvou částí:

- z hlavičky,
- z těla programu.

Hlavička programu

Hlavička obsahuje tyto údaje v následujícím pořadí:

	Údaj	Příklad
1.	Název programu	<code>Program Zkus;</code>
2.	Použité programové jednotky (knihovny)	<code>uses Crt;</code>
3.	Definice konstant	<code>const A = 5;</code>
4.	Deklarace proměnných	<code>var X, Y, Z: integer;</code>
5.	Všechny použité procedury a funkce	<code>procedure...;</code> <code>function...;</code>

Bližší vysvětlení:

ad 1. Slovo **Program**, za ním následuje **název programu** a **středník**.

ad 2. Programovací jazyk Pascal má připraveno mnoho funkcí a procedur, které budete používat, aniž byste je museli sami vymýšlet. Jsou rozděleny do několika programových celků, kterým se říká **jednotky (units)** nebo také **knihovny**. Automaticky je přítomna vždy jen základní jednotka **System** – v té je většina funkcí,

Při psaní zdrojového textu občas vše ukládejte. Můžete použít jednu z možností:

- Stiskněte klávesovou zkratku **F2**.
- Klepněte na panelu klávesových zkratk na **Save**.
- Z roletové nabídky **File** vyberte položku **Save**.

Až jste s psaním programu hotovi, nezapomeňte naposledy **uložit**.

Překlad

Cílem překladu zdrojového textu je:

- Odstranění syntaktických chyb ze zdrojového textu
- Vytvoření binárního kódu programu (ten se vytvoří v případě, že program už neobsahuje žádné syntaktické chyby).

Máte-li zdrojový text **uložen**, můžete přistoupit k překladu.

Postup:

- Klepněte na roletovou nabídku **Compile** a odtud vyberte volbu **Compile**.

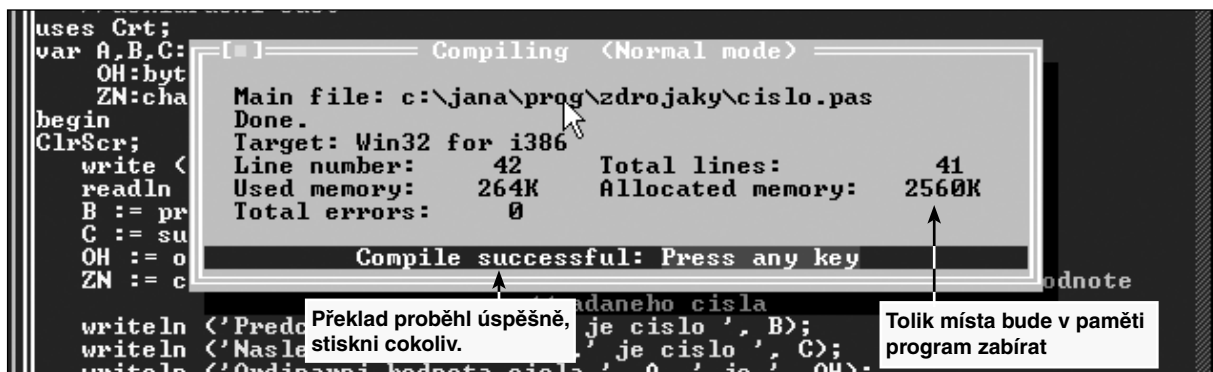


Tip: Pro spuštění překladu můžete také použít klávesovou zkratku **Alt+F9**.

Odstraňování syntaktických chyb

Pokud vše proběhne v pořádku a zdrojový kód je napsán úplně bez chyb, zobrazí se okno s informací, že překlad proběhl úspěšně. V tom případě se už vytvořil binární (spustitelný) soubor, který má příponu **.exe**, před příponou je **stejný název, s jakým jste uložili zdrojový soubor** (tedy nikoliv název programu, který jste napsali v hlavičce za slovo **Program**).

V informačním okně jsou také další zprávy – například kolik řádků zabírá zdrojový text a kolik místa v paměti bude binární soubor (program ve strojovém kódu zabírat).



Tvorba nejjednodušších programů

CÍL KAPITOLY: TVORBA PROGRAMŮ ZALOŽENÝCH NA SEKVENCÍCH

Důležité pojmy a nové dovednosti:

- Komentáře
- Příkazy vstupu a výstupu
- Přiřazovací příkaz
- Operace, které se mohou vyskytovat v sekvencích
- Součet obsahu dvou buněk
- Výměna obsahu dvou buněk

KOMENTÁŘ

V minulých kapitolách jste se dozvěděli o důležitosti komentářů. Nyní si ukážeme prakticky, jak se takový komentář vytvoří.

Následující program neobsahuje nic jiného než komentáře, proto nebude dělat vůbec nic, po spuštění hned skončí.

- Hned pod názvem programu bývá zvykem umístit komentář, ve kterém se vysvětlí, co má program dělat, popřípadě na jakých principech pracuje. Také zde bývá zvykem popsat význam všech proměnných, které se v programu vyskytují.
- V těle programu bývá zvykem komentovat všechny důležité programové struktury (větvení, cykly) a důležité programátorské obraty.

```

Program Koment;
{*****}
{
{   Program Komentar
{
{   Tento program nedela vubec nic. Text, ktery vidite,
{   je komentar pro programatora a je videt jen
{   ve zdrojovem textu.
{   Muze byt umisten kdekoliv v programu, v hlavicce
{   i v tele.
{   Komentovat program lze dvema zpusoby:
{       - Toto je jeden zpusob - text je uzavren
{         mezi slozene zavorky (zde je uzavren
{         do slozenych zavorek kazdy radek, ale muzete
{         take vytvorit nekolikaradkovy komentar, kde
{         na zacatku prvnio radku je leva slozena
{         zavorka, pak komentar pres nekolik radku
{         a na jeho konci prava slozena zavorka.
{         Pozor, pocet levych a pravych slozenych
{         zavorek se musi rovnat.
{
{*****}
begin
// Toto je take komentar - druhy zpusob. Muze se objevit kdekoliv na radku
// (v hlavicce i v tele programu) a musi byt uveden na zacatku
// dvema lomitky. Na jeho konci se zadny ukoncovaci znak nedela,
// ukonci se koncem radku. Na rozdil od predchoziho typu komentare
// je tento jen jednoradkovy. Tento typ se pouziva pro kratši komentare.
// Pokud byste jej presto chteli pouzít pro delši komentar, musíte napsat
// dve lomítka na začátku každého řádku (jako na této ukázce).
end.
```

Jednoduchá sekvence s jedním sekvenčním blokem – příkaz výstupu

Zadání:

Program po spuštění vypíše na obrazovku větu **Jsi chytrý a krásný** a poté skončí. Vývojový diagram najdete v knize **Algoritmizace** na straně 19. V této úloze si prakticky ukážeme použití procedury **write**.

```

Program Chytry;
{*****}
{
{   Jednoduchá sekvence - prikaz vystupu   }
{                                           }
{*****}
//Program nema zadnou deklaracni cast, protoze nepracuje
//se zadnymi promennymi.
//Ma jen telo

begin                                     //zacatek tela programu
    write ('Jsi chytry a krasny')         //prikaz vystupu
end.                                       //konec tela programu
    
```

Tento program zdánlivě „nic nedělá“, po spuštění hned skončí. Program sice skutečně vypíše to, co měl, jenomže skončí dřív, než si stačíte přečíst, co je na obrazovce.

Abyste program pozdrželi, za jeho jediný výkonný příkaz (za příkaz **write**) doplníte ještě jeden další příkaz, tzv. „prázdné čtení“.

```

Program Chytry1;
{*****}
{
{   Jednoduchá sekvence - prikaz vystupu   }
{   po prikazu write nasleduje "prazdne cteni" }
{                                           }
{*****}
//Program nema zadnou deklaracni cast, protoze nepracuje
//se zadnymi promennymi.
//Ma jen telo

begin                                     //zacatek tela programu
    write ('Jsi chytry a krasny');         //prikaz vystupu
    readln;                               //prazdne cteni
end.                                       //konec tela programu
    
```

Program už bude fungovat, ale na obrazovce se budou objevovat všechny staré výpisy, které vás budou mást. Proto by bylo dobré je odstranit, abyste na obrazovce vždy viděli jen aktuální informace. Provedete to pomocí příkazu **ClrScr** (z anglického **Clear Screen** - výmaz obrazovky). Tento příkaz však není součástí standardního souboru instrukcí jazyka ani není součástí knihovny **System**, která se přilinkovává automaticky. Je to procedura knihovny **Crt**, která obsahuje nástroje pro práci s obrazovkou. Musíte proto překladači říci, že budete tuto knihovnu potřebovat, a že ji tedy musí k programu připojit (přilinkovat). Do hlavičky programu v její deklarační části napíšete **uses Crt**;

Pozor, tento příkaz se musí objevit ještě před deklaracemi proměnných.

```

Program Chytry2;
{*****}
{
{   Jednoduchá sekvence - prikaz vstupu   }
{                                           }
{*****}
//deklaracni cast, pouze knihovna
uses Crt;                               //Bude potrebovat knihovnu Crt

//telo

begin                                     //Zacatek tela programu
    ClrScr;                               //Vymaz obrazovku
    write ('Jsi chytry a krasny');         //prikaz vystupu
    readln;                               //prazdne cteni
end.                                       //konec tela programu
    
```

Větvení

CÍL KAPITOLY: TVORBA TYPICKÝCH PROGRAMŮ S POUŽITÍM VĚTVENÍ

Nové dovednosti a znalosti:

- Syntaxe větvení - úplný a neúplný příkaz větvení, jeden příkaz ve větvi, více příkazů ve větvi
- Ošetřování nežádoucích důsledků v programech
- Typické programy na větvení z důvodu několika žádoucích možností
 - výrazy s absolutní hodnotou, zjišťování vlastností čísla (lichá, sudá, kladná, záporná, dělitelnost,...)
 - vyhledávání maxima a minima, řazení čísel podle velikosti
 - úlohy z geometrie
- Kombinované úlohy - rozvětvení programu z důvodu více korektních řešení v kombinaci s ošetřováním nežádoucích důsledků
- Podprogram typu procedura, lokální a globální parametry, procedury volané hodnotou a odkazem.

Nové příkazy:

- **abs** - absolutní hodnota
- **odd** - lichost
- **case** - vícenásobné větvení
- **upcase** - převod malého písmena velké

SYNTAXE VĚTVENÍ

V předchozí kapitole jste se seznámili s programy, které byly tvořeny samými sekvenčními bloky. S těmi však v praxi nevystačíte. Nyní se společně podíváme na větvení, které bylo v zápisu algoritmů realizováno rozhodovacím blokem.

Pro zápis programu bude důležité,

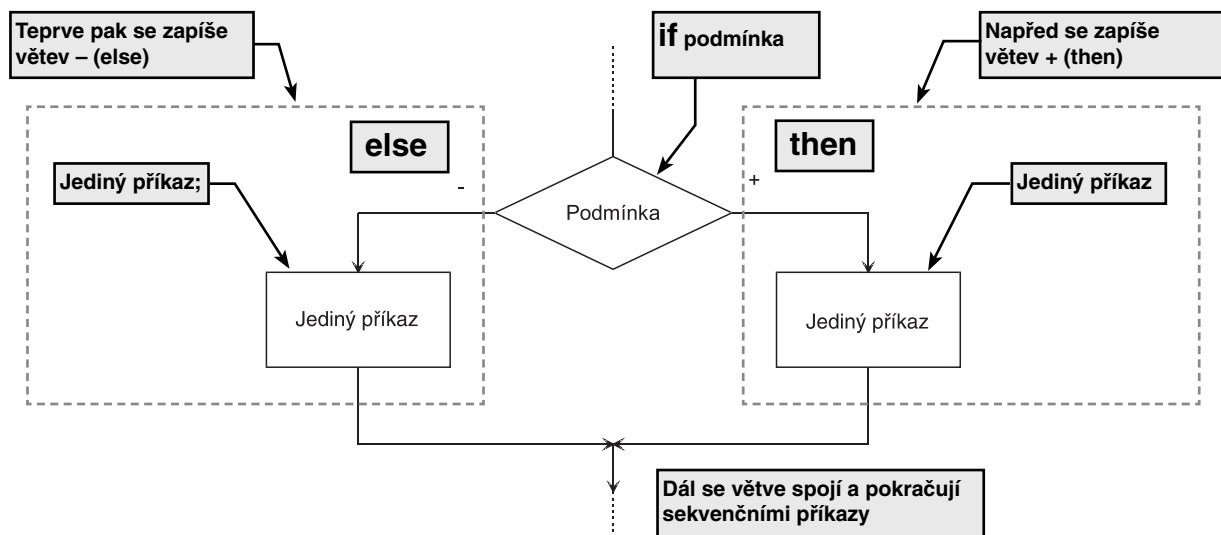
- zda se jedná o úplný či neúplný příkaz větvení,
- zda je ve větvích pouze jediný příkaz, nebo jestli jich tam je víc.

Úplný příkaz větvení

Jsou-li nějaké příkazy v obou větvích, pak se jedná o **úplný příkaz větvení**.

Je-li podmínka splněna,
pak udělej něco,
není-li podmínka splněna,
pak udělej něco jiného.

Je-li ve větvích jediný příkaz, pak má větvení v programu následující strukturu:



POROVNÁVÁNÍ A ŘAZENÍ ČÍSEL, MAXIMUM A MINIMUM

Porovnání dvou čísel podle velikosti – bez pomocné buňky

Typickou elementární úlohou, která je založena na větvení, je **porovnání dvou čísel podle velikosti**. Bez ní by se neobešla řada užitečných programů, jako například hledání maxima či minima z řady čísel (nebo jiných prvků) nebo také řazení prvků podle velikosti (třídící programy).

Zadání:

Načtete dvě čísla. Zjistěte, které číslo je menší a které větší. Nakonec je zobrazte – napřed menší číslo a potom větší.

Použité proměnné:

- A** první číslo (zadané zvenčí),
- B** druhé číslo (zadané zvenčí).

Algoritmus najdete v knize **Algoritmizace** na straně 32.

```

Program PorovD;
{*****}
{
{   Program Porovnej dve
{
{   Porovna dve cisla podle velikosti a zobrazi
{   vysledek.
{   Nepouziva pomocnou bunku.
{
{   Pouzite promenne:
{   A, B ... dve cela cisla, ktera mate porovnat
{
{*****}
//deklaracni cast
uses Crt;
var A,B: integer;           //deklarace promennych (cela cisla A, B)
//telo
begin                       //Zacatek tela programu
  ClrScr;
  write ('Zadej cislo A: ');
  readln (A);
  write ('Zadej cislo B: ');
  readln (B);
  if A > B
    then write (B, ', ', A)           //Porovnani A s B
    else write (A, ', ', B);         //Napred B, potom A
  readln;
end.                             //konec tela programu

```

Seřazení dvou čísel s pomocnou buňkou

Princip použitý v předchozím programu by byl výhodný pro porovnání právě dvou čísel a použitelný (i když ne výhodný) pro porovnání menšího počtu čísel. Naproti tomu postup, ve kterém použijete pomocnou buňku k výměně hodnot prvků, bude v budoucnu použitelný i pro vysoký počet porovnání.

Algoritmus najdete v knize **Algoritmizace** na straně 33 (nahore).

Použité proměnné:

- A** první číslo (zadané zvenčí),
- B** druhé číslo (zadané zvenčí),
- POM** pomocná buňka.

Cykly

CÍL KAPITOLY: NAUČIT SE VYTVÁŘET PROGRAMY S POUŽITÍM CYKLŮ

Nové pojmy:

- Cyklus **for** (s předem známým počtem opakování)
- Cykly řízené podmínkou
 - **while** - podmínka na začátku cyklu
 - **repeat** - podmínka na konci cyklu

Nové dovednosti – typické programy, jejichž základem je cyklus

- Úlohy s opakováním – programy, které fungovaly jednorázově, budou rozšířeny o možnost opakování
- Sumy, hledání maxima a minima
- Prohledávání řady čísel a znaků
- Oddechové úlohy – písničky, hry, hádanky
- Číselné soustavy a převody mezi nimi
- Řady – aritmetické, geometrické, mocninné a další
- Operace s vektory a maticemi
- Třídící programy

Nové strukturované datové typy:

- Znakový řetězec - **string**
- Pole – **array**
- Záznam – **record**
- Soubor (soubory textové, typové a netypové)

Nové knihovní funkce: **keypressed** (funkce knihovny **Crt**), **length** (funkce knihovny **System**)

Podprogramy typu funkce, rekurze

CYKLUS A JEHO TYPY

Z algoritmizace už víte, že jedním z nejsilnějších nástrojů algoritmů jsou cykly. Téměř všechny programovací jazyky, se kterými se v praxi setkáte, cykly podporují.

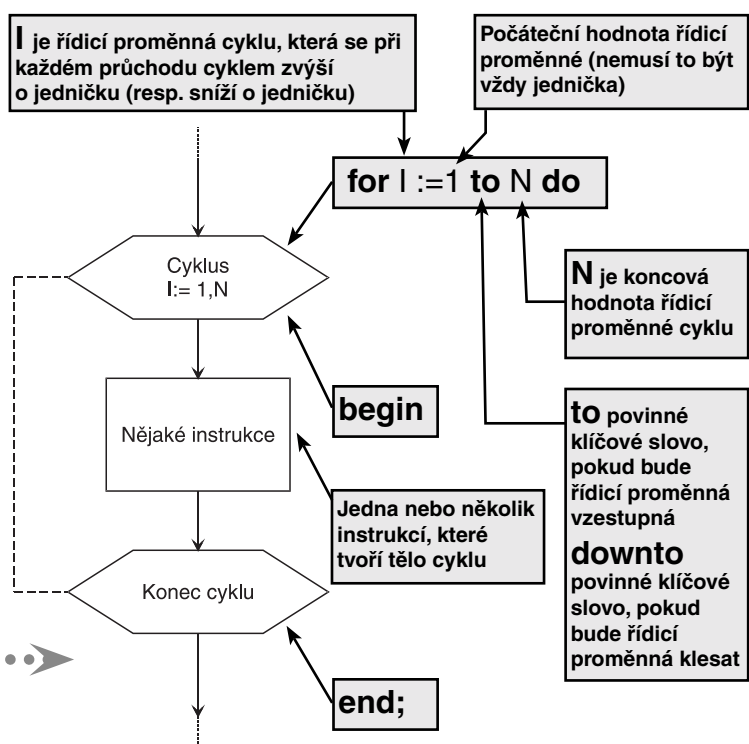
V Pascalu se můžete setkat se třemi typy cyklů, které korespondují s látkou vysvětlenou v knize Algoritmizace.

Jsou to:

- Cykly s pevným počtem opakování
- Cykly řízené podmínkou, přičemž podmínka může být
 - na začátku cyklu,
 - nebo na konci cyklu.

Cykly s pevným počtem opakování – cyklus for

Syntaxe cyklu s pevným počtem opakování:



Kalkulačka

Algoritmus najdete v knize **Algoritmizace** na straně 59 a 60.

Program bude umět:

- sečítat,
- odečítat,
- násobit,
- dělit.

Uživatel nejprve zadá kód, na základě kterého si určí, jakou operaci chce provádět. Potom zadá dvě čísla a požadovaná operace se s nimi provede.

Po zobrazení výpočtu se objeví dotaz, zda už chce skončit. Nechce-li skončit, bude moci znovu zadat nový kód operace a nová dvě čísla, se kterými se mu provede požadovaná operace.

```

Program KalkC;
{*****}
{
{   Program Kalkulacka s cyklem
{
{   Umi secitat, odecitat, nasobit a delit. Na zaklade
{   zadaneho kodu provadi operace se dvema cisly
{   zadanymi zvenci. Pocita tak dlouho, dokud nebude
{   zadan pozadavek na skonzeni.
{
{   Pouzite promenne:
{   KOD   ... kod operace
{           1 - pro secitani
{           2 - pro odecitani
{           3 - pro nasobeni
{           4 - pro deleni.
{   SKONC... kod skonzeni operace (znak)
{           'A'- skoncit
{           jakykoliv jiny znak - pokracovat
{   A, B ... dve cela cisla, s kterymi se bude provadet
{           pozadovana operace
{   C     ... vysledek (realne cislo)
{
{*****}
//deklaracni cast
uses Crt;
var A,B,KOD: integer;
    C: real;           //Kvuli deleni musi byt vysledek real
    SKONC: char;      //znakova promenna
//telo
begin
  ClrScr;
  // zacatek cyklu s podminkou na konci
  repeat
    write ('Zadej KOD: ');
    readln (KOD);           //precti kod operace
    write ('Zadej cislo A: ');
    readln (A);
    write ('Zadej cislo B: ');
    readln (B);
    case KOD of             //podle hodnoty kodu vybere vetev
    1: begin
        C := A + B;        //soucet
        writeln ('C = ',C:3:2)
      end;
    2: begin
        C := A - B;        //rozdil
        writeln ('C = ',C:3:2)
      end;
  end;

```