**Struktura programu v Pascalu**

**hlavička** - v TP (Turbo Pascalu) nepovinná **program** Soucet;

deklarace - (za klíčovým slovem **var** ) popis a pojmenování všech prostředků používaných

dále v programu,

 proměnné, konstanty, typy, návěstí,

 procedury a funkce

 **var** A, B: integer;

**tělo** - zápis algoritmu (posloupnost příkazů)

 **začíná begin a končí end. (s tečkou) begin**

 read(A); read(B);

 write(A+B)

 **end.**

**read**( proměnná) – čtení dat, **write** ( proměnná ) – tisk dat (na obrazovce PC)

**Program** *NázevProgramu*;

{Tohle je jenom vzorové schéma programu}

{V tomto místě končí hlavička a začínají definice s deklaracemi}

**uses** *NázevJednotek*;

**label** *NázvyNávěští*;

**const** *Konstanta1 = Výraz1;*

 *Konstanta2 = Výraz2;*

 ...

**type** *NázevTypu1 = Definice1;*

 *NázevTypu2 = Definice2;*

 ...

**var** *NázevProměnné1 : TypProměnné1;*

 *NázevProměnné2 : TypProměnné2;*

 ...

**procedure** *NázevProcedury1*(parametry1);

DefiniceProcedury1;

**procedure** *NázevProcedury2*(parametry2);

DefiniceProcedury2;

 ...

**function** *NázevFunkce1*(parametry):*VýslednýTypFunkce1*;

DefiniceFunkce1;

**function** *NázevFunkce2*(parametry):*VýslednýTypFunkce2*;

DefiniceFunkce2;

 ...

{Zde končí definice s definice s deklaracemi a začíná vlastní tělo programu}

**Begin**

 *Příkaz1;*

 *Příkaz2;*

 *...*

 *PříkazN;*

**End.**

{Zde končí tělo programu}

Za end. se sice psát může, ovšem je to zcela ignorováno (asi tak jako komentář)

**Hlavička, jednotlivé deklarace i příkazy se oddělují středníkem (znakem ; ).**

**Klíčová (vyhrazená ) slova** (v tisku zpravidla tučně,) = speciální víceznakové symboly s vyhrazeným významem. Např. var, type, if, then, else, while, do, for, to, repeat, until, array, of atd.

Přiřazení **A:=5**; {Do proměnné A jsme uložili hodnotu 5}

 **C:= B-D {**Do proměnné C jsme uložili rozdíl hodnot v proměnných B a D **}**

**Proměnná** = paměťové místo pro uložení hodnoty

 - jméno … jednoznačné pojmenování pomocí identifikátoru

 - typ … jaké hodnoty může uchovávat (určuje velikost přidělené paměti)

Deklarace proměnných - pomocí klíčového slova var – v Pascalu povinná

 syntaxe: var Jméno: Typ;

 Jm1, Jm2, Jm3: Typ; …{ více proměnných téhož typu}

 Např. var i: integer; { celá čísla }

 r, p: real; { reálná čísla }

**Typ vyjmenovaný a interval - můžeme nadefinovat svůj vlastní typ**

 type Den= ( Pondeli, Utery, Streda, Ctvrtek, Patek, Sobota, Nedele); {vyjmenovaný typ}

 type Pracden=Pondeli**..**Patek; { typ interval}

 type Vikend= Sobota**..**Nedele { typ interval}

 var D1, D2, D3: Pracden; {proměnné typu Pracden}

**Při deklaraci více proměnných v programu se nemusí (ale může) klíčové slovo var opakovat, jdou-li deklarace proměnných hned po sobě.** **Pokud je mezi nimi deklarován nějaký jiný prostředek (např. typ, konstanta, procedura), musí se var znovu uvést na začátku každého úseku deklarací proměnných.**

**Identifikátor** = způsob pojmenování jakýchkoliv objektů používaných v programu

 (**název programu, jména proměnných i jiných deklarovaných
 prostředků**)

 - tvořen písmeny **anglické abecedy a číslicemi**

 - **nesmí začínat číslicí**

 např.: Soucet, x1, alfa

 - v TP navíc znak \_ („podtržítko“)

 umožňuje používat „víceslovné názvy“, např.:
 Pocet**\_**Prvku

 - nerozlišuje malá a velká písmena
 (tzn. v identifikátoru a = A,

 na rozdíl od jazyků C, Java apod., kde **a**, **A** jsou různé znaky),

 alfa = ALFA = Alfa nebo Promenna=PROMENNA=promenna

lze používat „velbloudí notaci“ PocetKamenuNaSachovnici

Doporučení: - používat mnemotechnické identifikátory – název proměnné napovídá

 o jejím významu a účelu →Strana\_c\_trojuhelnika

 - zavést si pevný styl používání malých/velkých písmen.

**Typ** – určuje, kolik se pro proměnnou vyhradí místa v paměti

 a jakým způsobem se proměnná může používat v programu

 - **jednoduchý** – uložení jedné hodnoty

 v Pascalu (resp. TP) připraveny standardní datové typy

 celé číslo integer (byte, word, shortint, longint)

 desetinné číslo real (single, double, extended)

 logická hodnota boolean

 znak char

 - **strukturovaný** – sestaven z více složek (např. pole)

 nutno předem specifikovat počet a typ složek pomocí definice typu
 využitím vyhrazených klíčových slov

 pole array

 záznam record

 znakový řetězec string

 množina set

**const N=100;**

**type** pole = **array** [1..N] **of** real; {Typ jménem pole, n-tice reálných čísel }

**var** A, B, C: pole; { tři pole }

**Celočíselné typy** v TP

 shortint 1B se znaménkem -128..127

 **integer 2B se znaménkem -32768..32767**

 longint 4B se znaménkem -2147483648..2147483647

 byte 1B bez znaménka 0..255

 word 2B bez znaménka 0..65535

**Základní reálný typ v Pascalu**

 **real 6B 11-12 platných cifer rozsah hodnot 10-39..1038**

 **Dodatečné reálné typy v TP**

 single 4B 7-8 platných cifer rozsah hodnot 10-48..1038

 double 8B 15-16 platných cifer rozsah 10-324..10308

 extended 10B 19-20 platných cifer rozsah 10-4932..104932

**Čísla ve světě**

1 AU (astronomická jednotka) = střední vzdálenost Země – Slunce 1,495 . 1011 m

1 l.y. (světelný rok) 9,46 . 1015 m

vzdálenost hvězd – cca 10-15 l.y. (Proxima Centauri 4,27 l.y.)

hmotnost Slunce 1,993 . 1020 kg

velikost atomu řádově 10-10 m

hmotnost protonu 1,672 . 10-27 kg

hmotnost elektronu 9,109 . 10-31 kg

elementární náboj 1,602 . 10-19 C

Planckova konstanta (E=h.ν) h = 6,625 . 10-34 Js

Avogadrova konstanta (počet částic v 1 molu) NA = 6,022 . 1023 mol-1

**Rozsah hodnot standardního datového typu real plně postačuje.**

Doporučení:

 - jednotlivé deklarace a příkazy psát na samostatné řádky

 - používat indentaci (odsazování řádků od levého okraje podle logického zanoření příkazových struktur)

 **begin**

 read(X);

 Y := 0;

 **while** X <> 0 **do**

 **begin**

Y := Y + X **mod** 10;

 X := X **div** 10

  **end;**

 writeln(Y)

 **end.**

 - používat komentáře (význam jednotlivých proměnných, co dělá

 která procedura a funkce, co dělá která část programu).

**program** PlochaKruhu;

**var** R, P: real; {poloměr, výsledná plocha}

 **begin**

 read(R); {čtení poloměru}

 P := 3.14159265 \* R \* R;

 write(P) {tisk vypočtené plochy}

 **end.**

Je dobré zavést si vlastní styl zápisu programu a dodržovat ho.

**Nic z toho není povinné, překladač to ignoruje.** Významné zvýšení čitelnosti zdrojového kódu pro programátora (pro autora samotného i pro případné další „čtenáře“).

**Definice konstanty**

 - pojmenování konstanty identifikátorem

 - uvádí se v oblasti deklarací

 - syntaxe: const Jméno = Hodnota;

 - **typ konstanty je automaticky odvozen z uvedené hodnoty**

 - **hodnotu konstanty nelze při výpočtu měnit (nelze do ní dosazovat)**

 - **důvody: opakovaný výskyt téže „ošklivé“ hodnoty v programu**

 **možnost snadné změny na jediném místě v programu**

**const N=3;**

**type** vektor = **array** [1..N] **of** real; {Typ jménem vektor, trojice reálných čísel }

**var** A, B, C: vektor; { tři vektory }

**program** *PlochaKruhu;*

**const Pi = 3.14159265;** {DEFINICE KONSTANTY}

**var** R, P: real; {poloměr, výsledná plocha}

 **begin**

 read(R);

 P := Pi \* R \* R;

 write(P)

 **end.**

**Výraz typu real**

 - konstanty a proměnné typu real (příp. i celočíselné – provádí se

 automatická konverze do typu real)

 - aritmetické operátory +, -, \*, /

 - standardní funkce

 abs(X) absolutní hodnota z X

 sqr(X) druhá mocnina X

 sqrt(X) druhá odmocnina z X

 ln(X) přirozený logaritmus z X

 exp(X) eX

 sin(X) cos(X) arctan(X) goniometrické funkce

Výraz typu integer lze dosadit do proměnné typu real (provádí se automatická konverze), obráceně nikoliv.

**Výraz 10/5 má sice hodnotu 2, ale je typu real (reálné dělení).**

Převod výrazu typu real na integer: konverzní funkce

 round(X) zaokrouhlení round(3.8) = 4

 trunc(X) oříznutí desetinné části trunc(3.8) = 3

**Podmíněný příkaz**

**if** Podmínka **then** Příkaz

**if** Podmínka **then** Příkaz1 **else** Příkaz2

 **if** podmínka **then** příkaz1 **else** příkaz2

***Jestliže*** platí podmínka ***pak*** proveď příkaz 1  ***jinak*** proveď příkaz 2

 **if** A > 0 **then** writeln(‘ Cislo je kladne!’) **else** writeln(‘ Cislo neni kladne!’)

Tisk textu pomocí write nebo writeln: Text musí být umístěn mezi apostrofy

Write (‘text’); { text }

Jednoduchá podmínka = relace:

 Výraz1 relační operátor Výraz2

 = < > <= >= <>

Příklad:  **if** A > 0 **then** A := A-1 **else** A := A+1

Volba z více možností:

 **if** A=1 **then** …

 **else if** A=2 **then …**

 **else if** A=3 **then …**

 **else …**

**Sémantika vnořených if**

**if** A > 0 **then** i**f** B = C **then** A := A-1 **else** A := 0  **else** A := 1

pro lepší pochopení píšeme raději:

**if** A > 0 **then** i**f** B = C

 **then** A := A-1

 **else** A := 0

 **else** A := 1

**Složený příkaz** = „příkazové závorky“

**begin** Příkaz1; Příkaz2; …; PříkazN **end**

**if** A > 0 **then**

 **begin** A := A-1; B := 100 **end**

 **else {**před else není středník!}

 **begin** A := A+1; B := 200 **end**

**Cyklus**

**while** Podmínka **do** Příkaz

***Dokud*** platí podmínka ***provádě***j příkaz

 - dokud je podmínka splněna, příkaz v těle cyklu se opakuje

 - cyklus s podmínkou na začátku → tělo se nemusí provést ani jednou, když
 podmínka od začátku neplatí

i:=1 - více akcí v těle cyklu → složený příkaz

while i<10 do

 begin

 write(i) {1 2 3 4 5 6 7 8 9 }

 i:=i+1;

 end;

**repeat** Příkaz1; Příkaz2; …; PříkazN **until** Podmínka

***opakuj*** provádění příkazů ***dokud***  neplatí podmínka

 - příkazy v těle cyklu se opakují až do splnění podmínky

 - cyklus s podmínkou na konci →tělo se vždy provede aspoň jednou (i když podmínka platí)

 - více akcí v těle cyklu lze přímo zapsat (bez begin a end)

 podmínka od začátku neplatí

i:=1 - více akcí v těle cyklu → složený příkaz

repeat

 write(i) { 1 2 3 4 5 6 7 8 9 }

 i:=i+1;

until I >= 10

 end;

**Cyklus FOR**

**for** Řídicí\_Proměnná := Dolní\_Mez **to** Horní\_Mez **do** Příkaz

***pro*** proměnou od její dolní meze ***po*** horní mez ***prováděj***  příkaz

- řídicí proměnná musí být normálně deklarovaná proměnná celočíselného typu, dolní o horní mez jsou celočíselné výrazy

for i:=1 to 9 do

 write(i) {1 2 3 4 5 6 7 8 9 }

**Sémantika for-cyklu**

 - řídicí proměnná nabývá všech celočíselných hodnot počínaje dolní mezí a
 konče horní mezí (včetně), pro každou z nich se vykoná příkaz v těle cyklu

 - v příkazu lze využívat hodnoty řídicí proměnné(výhodné zejména pro indexování polí )
 - výrazy určující meze se vyhodnocují jen jednou při vstupu do cyklu → cyklus „s pevným počtem opakování“ (případné změny horní meze v průběhu výpočtu cyklu již neovlivní počet opakování)

 - pokud je dolní mez větší než horní mez, příkaz se nevykoná ani jednou

 - pokud se horní mez rovná dolní mezi, příkaz se vykoná právě jednou

 - řídicí proměnná má po skončení cyklu poslední hodnotu, kterou nabyla, tzn. horní mez
 (podle normy není její hodnota definována)

 - změna řídicí proměnné uvnitř cyklu je možná a ovlivní počet opakování (tzn. Je
 nebezpečná, může vést i k zacyklení,
 podle normy nebyla vůbec povolena – raději nepoužívat)

**Varianta for-cyklu pro opačný směr průchodu**

**for** Řídicí\_Proměnná := Horní\_Mez **downto** Dolní\_Mez **do** Příkaz

***pro*** proměnou od její horní meze ***dolů k***  dolní mezi ***prováděj***  příkaz

for i:=9 downto 1 do

 write(i) { 9 8 7 6 5 4 3 2 1}

- hodnota řídicí proměnné klesá po 1 od horní meze dolů k dolní

mezi (jinak totéž)

- použití není příliš časté, hodí se někdy při indexování polí

**Předčasné ukončení cyklu**

Občas se hodí mít možnost za určitých podmínek výpočet cyklu předčasně ukončit → standardní procedura  **break** použitá v těle cyklu (for, while, repeat), zpravidla v podmíněném příkazu.

**for** I:=1 **to** N **do**

**begin**

…; …; …;

**if** PodmínkaPředčasnéhoUkončení **then** break;

…; …; …;

**end**;

**Výpočet pokračuje příkazem následujícím bezprostředně za cyklem.**