Základy programovacího jazyka Turbo Pascal

- Programovací jazyk Pascal byl navržen začátkem 70. let
  - profesor **Niklaus Wirth** z Vysoké školy technické v Curychu

- **Cíle návrhu**
  - vytvořit jazyk vhodný pro výuku programování založený na omezeném počtu srozumitelných konstrukcí
  - navrhnout strukturu jazyka tak, aby bylo snadné implementovat Pascal na většině tehdejších počítačů

- **V roce 1981 byla vydána norma ISO**

- Největšího úspěchu dosáhla implementace firmy **Borland** pod názvem **Turbo Pascal**

Struktura a základní prvky programu

- Podpora strukturovaného programování

- Základním stavebním kamenem je **blok**
  - **Klíčová slova** jsou stanovena definicí jazyka a nelze je žádným způsobem měnit - vyznačují jednotlivé konstrukce jazyka
  - **Identifikátory** označují objekty, které program používá (proměnné, typy, …), a akce, které s nimi provádí (procedury, funkce, …)
  - **Komentář** slouží ke zvýšení srozumitelnosti programu – **{…}**

Identifikátory

- Jako jména konstant, proměnných, procedur a funkcí lze použít **jakoukoliv posloupnost písmen a číslic začínající písmenem**

- Jména nesmí obsahovat **mezeru** a speciální znaky

- **Pascal nerozlišuje velká a malá písmena**

- Celá řada předem připravených identifikátorů - **standardní identifikátory**
  - předdefinované **datové typy**
  - předdefinované **procedury a funkce**

http://kii.fei.czu.cz/FIVYS

Zdrojový text programu

- Pascal používá **volný formát zápisu** - nevyžaduje pevný tvar zdrojového textu

- **Používaná pravidla**
  - každý příkaz na samostatném řádku
  - na levý okraj řádků se vkládají mezery tak, aby příkazy vnořených částí byly odsazeny více než jejich obalující konstrukce
  - dvojice k sobě patřících klíčových slov (např. `begin` a `end`) jsou pod sebou
  - významnější části programu bývají navzájem odděleny prázdnými řádky a opatřeny komentářem

Části programu

- **hlavička - jméno programu**
- **použité moduly** (*unit*)
- **definice konstant**
- **definice datových typů**
- **definice proměnných**
- **definice procedur a funkcí**
- **hlavní program**

```
program Jméno_programu;
uses Jedn1,Jedn2;
const Jméno = hodnota;
type datový_typ = ...;
var proměnná:datový_typ;
procedure ...;
function ...;
begin
  {hlavní program}
end.
```

Deklarace

- **Deklarace**
  - obsahují výčet **konstant**, **proměnných**, **datových typů**, **procedur** a **funkcí**
  - veškeré prvky, které se kdekoliv v programu objeví, musí být před svým použitím definovány

- **Konstanty** jsou prvky, které v průběhu provádění programu nemění svou hodnotu
  - `const konstanta = hodnota`

- **Proměnná** je datový objekt, jehož hodnota se v průběhu programu může měnit
  - `var seznam_proměnných : datový_typ`

http://kii.pef.czu.cz/Elvys

Příkazy a operátory

- **Příkazy**
  - definují, co konkrétně má blok programu udělat
  - realizují vlastní algoritmus
  - příkazy jsou odděleny **středníkem** `;`
  - **složený příkaz** = **programové závorky** - je vymezen klíčovými slovy `begin` a `end`

- **Přiřazovací příkaz**
  - pomocí něho proměnná nabývá hodnotu, případně se nahradí aktuální hodnota hodnotou novou
  - `proměnná := výraz(hodnota)`

- **Operátory**
  - S proměnnými ve výrazech pracujeme pomocí operátorů
    - **logické** `and`, `or`, `not`
    - **aritmetické** `+`, `-`, `*`, `/`, `div`, `mod`
    - **porovnávací** `=`, `>`, `<`, `>=`, `<=`, `<>`
  - Pro **priority operátorů** (pořadí vyhodnocování) platí běžná matematická pravidla
    - pořadí lze změnit závorkami `( )`

Předdefinované datové typy

- **Celočíselné**
  - `shortint` -128 .. 127
  - `integer` -32768..32767
  - `longint` -2147483648..2147483647
  - `byte` 0..255
  - `word` 0..65535

- **Reálné**
  - `real` 2.9E-39..1.7E38
  - `single` 1.5E-45..3.4E38
  - `double` 5.0E-324..1.7E308
  - `extended` 3.4E-4932..1.1E4932
  - `comp` -9.2E18..9.2E18

- **Znakové**
  - `char` jeden znak
  - `string` řetězec max. 255 znaků
  - `pchar` řetězec max. 65535 znaků

- **Logické**
  - `boolean` true, false

http://mipef.czu.cz/Elvys

- **Datový typ interval**
  - `proměnná : dolní_mez .. horní_mez`

- **Výčtový datový typ**
  - `jméno_typu = (hodnota1, hodnota2, … , hodnotaN)`

- **Datový typ pole**
  - `array [typ_indexu] of typ_položky`
  - datový typ `string` = pole znaků
  - odkaz na jednotlivé položky pole prostřednictvím indexu: `proměnná [index]`

- **Datový typ množina**
  - `set of bázový_typ`
  - **speciální operace** `+` - sjednocení, `*` - průnik, `<=` - podmnožina, `in` - prvek množiny, …

- **Datový typ záznam**
  - ```
    record
          seznam_identifikátorů1 : datový_typ1;
          …
          seznam_identifikátorůN : datový_typN
    end
    ```
  - Odkazy na jednotlivé položky
    - **tečková notace**
      `proměnná.jméno_položky`
    - **příkaz** `with`
      `with proměnná do příkaz`

- **Datový typ soubor**
  - `SouborTyp = file of typ_položka`
    - typový soubor
  - `SouborNetyp = file`
    - soubor bez udání typu
  - `SouborText = text`
    - textový soubor

- **Datový typ ukazatel**
  - `Ukazatel = ^ doménový_typ`
  - dynamicky alokované proměnné

http://kti.pef.czu.cz/EIVYS
    ```

Procedury a funkce

- Program bývá rozdělen na menší části – **podprogramy**
  - **procedury** a **funkce**

- **Struktura procedury**
  - jméno procedury, parametry
  - seznam konstant a proměnných
  - vnořené procedury a funkce
  - tělo procedury

```
procedure Jméno_procedury
  (definice_parametrů);
const …;
var …;
procedure …;
function …;

begin
  {tělo procedury}
end;
```

- **Struktura funkce**
  - jméno funkce, parametry, typ výsledku
  - seznam konstant a proměnných
  - vnořené procedury a funkce
  - tělo funkce, výsledek funkce

```
function Jméno_funkce
  (definice_parametrů):
  typ_výsledku;
const …;
var …;
procedure …;
function …;

begin
  {tělo procedury}
  Jméno_funkce := výsledek
end;
```

- **Volání procedur a funkcí v programu**
  - **procedury:**
    `Jmeno_procedury(parametry);`
  - **funkce:**
    `Jmeno_promenne:=Jmeno_funkce(parametry);`

- **Parametry procedur a funkcí**
  - **volané hodnotou**
    - lokální proměnné inicializované danou počáteční hodnotou, skutečný parametr se nemění
  - **volané odkazem**
    - cokoli se provádí s formální proměnnou, dělá se ve skutečnosti s odpovídajícím skutečným parametrem

http://kti.pef.czu.cz/Elvys

Větvení programu

- Větvení programu může být nepodmíněné nebo podmíněné
  - **nepodmíněné** – příkaz `goto`
    - porušení strukturovaného návrhu
  - **podmíněné**
    - realizace struktury **selekce** - větvení na základě splnění či nesplnění podmínky

```
if (podmínka) then příkaz1 else příkaz2;
```

    - **výběr variant**

```
case výraz of
  hodnota1: příkaz1;
  …
  hodnotaN: příkazN;
  else příkaz_ostatní
end;
```

Cykly

- **Cyklus s podmínkou na začátku**

```
while (podmínka) do příkaz;
```

- **Cyklus s podmínkou na konci**

```
repeat příkaz until (podmínka);
```

- **Cyklus s pevným počtem opakování**

```
for proměnná := počáteční to koncová do příkaz;
for proměnná := počáteční downto koncová do příkaz;
```

Procedury vstupu a výstupu

- **Standardní** zařízení počítače pro **vstup je klávesnice**, pro **výstup obrazovka**
  - **Procedury vstupu**
    - `Read(proměnná);`

    čte vstup z klávesnice a uloží jej do proměnné
  - **Procedury výstupu**
    - `Write('Text…',proměnná,konstanta,…);`

    výstup na obrazovku, jedné nebo více proměnných či konstant
    - `WriteLn('Text…',proměnná,konstanta,…)`

    po výpisu dojde navíc k odřádkování

http://kip.pef.czu.cz/Elvys

Další procedury a funkce

- Další procedury a funkce jsou **definovány v jednotkách** (modulech)
  - použití jednotky – po klíčovém slovu **uses**

- **Vybrané jednotky**
  - **SYSTEM**
    - matematické funkce
    - operace s řetězci
    - operace se soubory
  - **CRT**
    - operace pro obsluhu klávesnice a obrazovky

Matematické funkce

- **Abs**(x:*číslo*):*číslo*;
  - Absolutní hodnota čísla *x*. Parametr i vrácená hodnota mohou být libovolné číslo.

- **Exp**(x:real):real;
  - Exponenciála čísla *x* ve tvaru e^x.

- **Ln**(x:real):real;
  - Přirozený logaritmus čísla *x* .

- **Sqr**(x:*číslo*):*číslo*;
  - Druhá mocnina čísla *x* . Parametr může být libovolného číselného typu, vrácená hodnota je stejného typu.

- **Sqrt**(x:real):real;
  - Druhá odmocnina čísla *x* . Argument musí být nezáporný.

- **Frac**(x:real):real;
  - Desetinná část čísla *x* .

- **Int**(x:real):real;
  - Celá část čísla *x* vrácená jako typ `real`.

- **Trunc**(x:real):longint;
  - Celá část čísla *x* vrácená jako typ `longint`.

- **Round**(x:real):longint;
  - Běžné zaokrouhlení čísla *x* .

- **Random**(w:word):word;
  - Vrací pseudonáhodné číslo v rozsahu **0 - *w***. Před použitím této funkce je vhodné inicializovat generátor náhodných čísel a to procedurou `Randomize`.

http://kti.pef.czu.cz/Elvys

- **Pi**:real;
  - Vrací konstantu 3.1415926535897932385 jako datový typ `real`. Přesnost se může měnit podle použití matematického koprocesoru.

- **ArcTan**(x:real):real;
  - Arcus tangens čísla *x* vrací úhel v radiánech.

- **Cos**(x:real):real;
  - Kosinus úhlu *x* v radiánech.

- **Sin**(x:real):real;
  - Sinus úhlu *x* v radiánech .

Operace s řetězci

- **Length**(S:string):integer;
  - Vrací počet znaků v řetězci.

- **Chr**(B:byte):char;
  - Vrací znak z tabulky ASCII podle osmi-bitové hodnoty parametru.

- **UpCase**(C:char):char;
  - Převede malé písmeno na velké.

- **Copy**(S:string;Index,Count:integer):string;
  - Vrací `Count` znaků řetězce S počínaje znakem na pozici `Index` .

- **Delete**(var S:string;Index,Count:integer);
  - Odstraní `Count` znaků z řetězce S počínaje znakem na pozici `Index`.

- **Insert**(source:string;var S:string;Index:integer);
  - Do řetězce *S* od pozice `Index` vloží řetězec `source`.

- **Str**(X:číslo;var S:string);
  - Převede číslo *x* na řetězec *S*.

- **Val**(S:string;var V:číslo;var Code:integer);
  - Převede řetězec *S* na číslo *V*. V případě chyby nastaví v `Code` číslo pozice znaku, který nelze převést.

http://kti.pef.czu.cz/Elvys

Práce se soubory

- `Read(var f:file,seznam_proměnných);`
- `Readln(var f:file,seznam_proměnných);`
  - Čte hodnoty ze souboru *f* a ukládá je do seznamu proměnných.

- `Write(var f:file,seznam_výrazů);`
- `Writeln(var f:file,seznam_výrazů);`
  - Zapisuje hodnoty dané seznamem výrazů do souboru *f*.

- `IOResult:integer;`
  - Vrací kód chyby poslední operace vstupu - výstupu. Byla-li operace úspěšná, vrací nulu.

- `Assign(var f:file; 'jméno_souboru');`
  - Přiřadí jméno souboru proměnné *f*. Veškeré operace s proměnnou se ve skutečnosti provádějí se souborem. Jméno souboru je zadáváno jako znakový řetězec.

- `Append(var f:text);`
  - Otevření textového souboru pro připojení dalšího textu na konec souboru.

- `Reset(var f:file);`
  - Otevření existujícího souboru. Je-li to textový soubor, bude otevřen pouze pro čtení.

- `Rewrite(var f:file);`
  - Vytvoření a otevření souboru. Pokud již existoval, bude jeho obsah smazán.

- `Close(var f:file);`
  - Uzavření souboru.

- `Eof(var f:file):boolean;`
  - Jestliže znak, který se má právě přečíst je konec souboru, vrací hodnotu `true`, jinak `false`.

- `EoLn(var f:file):boolean;`
  - Jestliže znak, který se má právě přečíst je konec řádku, vrací hodnotu `true`, jinak `false`.

- `FilePos(var f:file):longint;`
  - Vrací aktuální polohu ukazatele souboru (pořadové číslo složky). Když je aktuální poloha ukazatele na začátku souboru, funkce vrací hodnotu 0. Soubor musí být před voláním funkce otevřený.

- `FileSize(var f:file):longint;`
  - Funkce vrací počet složek uložených do souboru (aktuální velikost souboru). Soubor musí být před voláním funkce otevřený.

- `Seek(var f:file; U:longint);`
  - Procedura přemístí polohu ukazatele v souboru na složku s pořadovým číslem *U*. Číslo první složky v souboru je 0.

http://kj.pef.czu.cz/elvys

- **`Seek(f,FileSize(f));`**
  - Přemístí ukazatel na konec souboru.

- **`Erase(var f:file);`**
  - Odstraní soubor z disku. Soubor nesmí být otevřen.

- **`Rename(var f:file; 'nové_jméno_souboru');`**
  - Přejmenuje soubor.

Operace pro obsluhu klávesnice a obrazovky

- **`KeyPressed:boolean;`**
  - Testuje, zda byla stisknuta některá klávesa na klávesnici. Jestliže ano, vrací hodnotu true. Funkce `Keypressed` je oblíbena v kombinaci s cyklem `repeat-until`.

- **`repeat until keypressed;`**
  - Tento speciální `repeat` cyklus doslova zastaví běh programu a čeká na stisk klávesy. Použít tento sled příkazů se vyplatí např. na konci programu, kdy jsou vypsány nějaké hodnoty, které by uživatel jinak těžko zachytil (Platí to zejména pokud spouštíte programy z Windows a máte nastaveno "Při skončení zavřít").

- **`ReadKey:char;`**
  - Vrací jeden znak z vyrovnávací paměti klávesnice. Pokud tam žádný znak není, čeká na stisknutí klávesy.

- **`ClrScr;`**
  - Vymaže obrazovku a nastaví textový kurzor do levého horního rohu.

- **`Delay(time:word);`**
  - Pozastaví provádění programu na `time` milisekund.

http://kti.pef.czu.cz/EMVS