Obsah

1 Soubory v Pascalu **1**
1.1 Netypové soubory 1

2 Dynamické datové struktury **1**
2.1 Typ ukazatel 2

1 Soubory v Pascalu

1.1 Netypové soubory

Použití netypového souboru

```
var f:file;
    i:integer;
begin
  Assign(f, 'soubor.dta');
  Rewrite(f,1);        Velikost 1 bloku je 1B
  i:=1200;
  BlockWrite(f,i,1);   Do souboru zapíše 1 blok po 1B
  Reset(f,1);
  i:=0;
  BlockRead(f,i,1);    V proměnné i bude číslo 176
                       (1200=0x04B0, 0xB0=176)
  Close(f);
end.
```

Příklad použití netypového souboru

Čtení hlavičky souboru .PCX. Popis je na `http://www.geocad.ru/new/site/Formats/Graphics/pcx/pcx.txt`.

2 Dynamické datové struktury

Dynamické datové struktury

- Dosud jsme pracovali s proměnnými (datovými strukturami), které mají předem danou velikost a umísnění v paměti. Takovéto datové struktury se nazývají *statické datové struktury*.
- Opakem jsou *dynamické datové struktury*, jejichž velikost (a tím také umístění) se určuje až za běhu programu.

- *Statické* struktury se v programu vznikají a a zanikají automaticky dle deklarace.
- *Dynamické* struktury se nazavádějí deklarací, ale vznikají v průběhu programu pomocí speciálních příkazů. Nemohou mít své jméno (identifikátor). Abychom se k těmto strukturám dostali, máme k dispozici datový typ *Ukazatel* (většinou statická proměnná), který obsahuje odkaz na nějaký kus paměti.

2.1 Typ ukazatel

Datový typ ukazatel

- Ukazatel obsahuje pouze adresu paměti, ve které jsou (nebo mohou být) nějaká data.
- Speciální hodnota `nil`, která znamená „neukazuji nikam“ – ukazatelová nula.
- Nastavení hodnoty ukazatele lze provést pomocí:
  - Přiřazení z jiné proměnné typu ukazatel
  - Operátoru `@` – vytvoří ukazatel na proměnnou
  - Funkce `Ptr` – vytvoří ukazatel do specifikovaného místa v paměti
  - Procedur `New` nebo `GetMem` – alokuje novou paměť pro ukazatel – většinový přístup
- Paměť alokovaná pomocí `New` (nebo `GetMem`) je nutné uvolnit pomocí procedury `Dispose` (nebo `FreeMem`)

Příklady ukazatelů

```
type PInteger=^integer;   Ukazatel na číslo
     PString=^string;     Ukazatel na řetězec
     PUkazatel=^PString;  Ukazatel na ukazatel
     TPole=array[1..10] of integer;
     PPole=^TPole;        Ukazatel na pole
     PZaznam=^TZaznam;    Ukazatel na záznam
     TZaznam=record       deklarovaný níže
               ...
             end;
```

Použití ukazatelů

```
var p:PInteger;
    i:integer;
begin
```

```
  i:=10;       i=10, p=???(nil?)
  p:=@i;       i=10, p ukazuje na i, p^=10
  p^:=100;     i=100, p^=100
  i:=0;        i=0, p^=0
  New(p);      i=0, p ukazuje do paměti, p^=???
  p^:=10;      i=0, p^=10
  Dispose(p); p ukazuje na uvolněnou paměť
  GetMem(p, SizeOf(integer));
  ...
  FreeMem(p, SizeOf(integer));
end.
```

Příklad využití ukazatelů

Jeden z příkladů využití ukazatelů je zřetězený seznam:

```
type PSeznam=^TSeznam;
     TSeznam=record
               i:integer;
               Dalsi:PSeznam;
             end;
procedure PridejDoSeznamu(Cislo:integer;
                          var Seznam:PSeznam);
var NovyZacatek:PSeznam;
begin
  New(NovyZacatek);
  NovyZacatek^.Dalsi:=Seznam;
  NovyZacatek^.i:=Cislo;
  Seznam:=NovyZacatek;
end;
```

Příklad využití ukazatelů – pokračování

```
type PSeznam=^TSeznam;
     TSeznam=record
               i:integer;
               Dalsi:PSeznam;
             end;
procedure VypisSeznam(Seznam:PSeznam);
begin
  while Seznam<>nil do
  begin
    WriteLn(Seznam^.Cislo);
    Seznam:=Seznam^.Dalsi;
  end;
end;
```

Příklad využití ukazatelů – pokračování

```
type PSeznam=^TSeznam;
     TSeznam=record
               i:integer;
               Dalsi:PSeznam;
             end;
procedure ZrusSeznam(var Seznam:PSeznam);
begin
  if Seznam<>nil then
  begin
    ZrusSeznam(Seznam^.Dalsi);
    Dispose(Seznam);
    Seznam:=nil;
  end;
end;
```