Anotace

- Direktivy překladače
- Soubory (textové)
- Struktury (record)

Direktivy překladače

- Překladač kontroluje plno věcí, například:
- zda nekoukáme za konec pole,
- zda nám nepřetekl zásobník,
- anebo zda nenastala chyba na vstupu/výstupu...
- Většinou je užitečné mít kontroly zapnuté, někdy však "víme, co děláme".
- V tom případě můžeme na nezbytnou dobu chování překladače změnit pomocí tzv. *direktiv překladače.*
- Direktivy vypadají jako komentář, tedy jsou ve složených závorkách, ovšem začínají znakem string ($), jméno je zpravidla 1znakové a následuje přepínač $+/-$.

Direktivy překladače:

- Příklad: $\{\$R-\}$ – vypni *range-checking*.
- Nejdůležitější:
  - $\$Q$ – overflow-checking,
  - $\$R$ – range-checking,
  - $\$I$ – test vstupu a výstupu,
  - Úplný seznam najdete v helpu (některé direktivy se liší podle překladačů).

Soubory a práce s nimi

- V tomto semestru budou pouze soubory textové. Ačkoliv existují i soubory binární, ty budou předmětem výuky až v létě!
- Textový soubor ovládáme pomocí proměnné typu `Text`.
- Příslušnou proměnnou "napojíme" na daný soubor pomocí funkce `Assign`,
- otevřeme pomocí funkce `Reset`, `Rewrite` nebo `Append`,
- soubor čteme pomocí funkce `Read` (resp. `Readln`), které jako první argument předáme příslušnou proměnnou typu `Text`, zapisujeme analogicky pomocí funkcí `Write` a `Writeln`.
- Nakonec soubor uzavřeme pomocí funkce `Close`.

Práce se souborem – syntax (1)

- `var f:Text;`
- `Assign(f,'soubor.txt');` – asociuj proměnnou `f` se souborem `soubor.txt`.
- `Reset(f);` – otevři soubor `f` (pro čtení).
- `Rewrite(f);` – otevři soubor `f` a jeho dosavadní obsah znič.
- `Append(f);` – otevři soubor `f` pro zápis za jeho dosavadní konec.

Práce se souborem – syntax (2)

- `Writeln(f,'Zapiseme text do souboru');` – zapiš do souboru příslušný text.
- `Read(f,a);` – načti ze souboru proměnnou `a`.
- `Close(f);` – uzavři soubor (už s ním nebudeme pracovat).
- `eof(f);` – funkce, která sdělí, zda jsme na konci souboru.
- `eof;` – funkce oznamující konec standardního vstupu (z klávesnice).
- Existuje mnoho dalších funkcí jako `Rename`, `Erase`, ...

Potíže se soubory

- Často se stane, že otevíraný soubor neexistuje.
- Tato událost vyvolá input/output error.
- Nechceme-li při zapnuté této direktivě překladače soubor zničit (pomocí `Rewrite`, které sice neexistující soubor založí, ale existující přemaže), použijeme direktivu překladače a zda nastala chyba zjistíme pomocí funkce `IOResult`.

Příklad

```
Assign(f,'soubor.txt');
{$I-} {Vypni test na vstupně-výstupní chyby}
Reset(f);
{$I+} {Zapni test vstupně-výstupních chyb}
if IOResult<>0 then
begin writeln('Chyba!'); halt;
end;
while not eof(f) do begin
      readln(f,s);
      writeln(s);
end;
```

Pozor, `IOResult` je funkce a po zavolání ztratí hodnotu, nelze ji tedy číst opakovaně a její výsledek je případně třeba uložit do proměnné!

Využití souborů

- Pokud máme vstupy několika typů (například popis kódu a text),
- konfigurace (například zápočtového programu),
- ladění: Vyrobíme několik vzorových vstupů a zkusíme, co program udělá.
- Ladit můžeme kupříkladu hned třídící programy (algoritmy).

Datový typ `record`

- Typicky míváme objekt popsaný několika hodnotami (bod v rovině: dvojice hodnot).
- Je nešikovné mít každou hodnotu někde úplně jinde (je vhodnější mít data co nejvíce pohromadě).
- Definujeme tedy strukturu, kde budeme mít všechny údaje pohromadě.
- Definujeme pomocí klíčového slova `record`.

Typ record – příklad

```
type  bod=record
            x,y:integer;
      end;
var   a,b:bod;
      kn:record
            autor: string[100];
            nazev: string[100];
            rok: integer;
      end;
```

Přístup do struktury

Do struktur přistupujeme pomocí operátoru tečky:
`a.x` – prvek `x` struktury `a`
Jednotlivé prvky se chovají jako běžné proměnné, můžeme tedy číst jejich hodnoty, zapisovat do nich...

Typ record – příklad použití

```
var a,b:bod; kn:kniha;
begin
      a.x:=1;
      a.y:=2;
      b.x:=10;
      b.y:=10;
      kn.autor:='Topfer, P.';
      kn.nazev:='Algoritmy a programovaci techniky';
      kn.rok:=1995;
end.
```

Pole struktur

```
var knihovna:array [1..100] of record
            autor:string[25];
            nazev:string[45];
            rok:integer;
end;
begin
      for i:=1 to 100 do begin
            readln(knihovna[i].autor);
            readln(knihovna[i].nazev);
            readln(knihovna[i].rok);
      end; ...
```

Klíčové slovo `with`

Chceme-li pracovat se strukturami, je otravné stále říkat, ve které struktuře se pohybujeme (`struktura_s_dlouhym_nazvem_kterou_jsme_vymysleli_v_opilosti`) Proto můžeme říct:

`with` struktura `do` příkaz nebo blok;

a v sekci `with` můžeme přistupovat k jednotlivým prvkům struktury rovnou.

Konstrukce `with` – příklad:

```
procedure vypis(kniha_s_dlouhym_nazvem:kniha);
begin
      with kniha_s_dlouhym_nazvem do
            writeln(autor:25,nazev:46,rok:5);
end;
...
      for i:=1 to 100 do
            vypis(knihovna[i]);
...
```