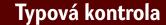
O datových typech a jejich kontrole

Programovací techniky

doc. Ing. Jiří Rybička, Dr. ústav informatiky PEF MENDELU v Brně rybicka@mendelu.cz



• Co to je

- Co to je
- Jaké jsou vlastnosti

- Co to je
- Jaké jsou vlastnosti
- Kdy ji potřebujeme a kdy ne

- Co to je
- Jaké jsou vlastnosti
- Kdy ji potřebujeme a kdy ne
- Jak ji ovlivnit beztypové struktury

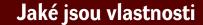
• Souvisí s pojmem kompatibilita typů, identita typů

- Souvisí s pojmem kompatibilita typů, identita typů
- Sémantická kontrola zjišťující shodu v rámci kompatibility

- Souvisí s pojmem kompatibilita typů, identita typů
- Sémantická kontrola zjišťující shodu v rámci kompatibility
- Okamžik přiřazení

- Souvisí s pojmem kompatibilita typů, identita typů
- Sémantická kontrola zjišťující shodu v rámci kompatibility
- Okamžik přiřazení
- Okamžik volání podprogramu, parametr odkazem

- Souvisí s pojmem kompatibilita typů, identita typů
- Sémantická kontrola zjišťující shodu v rámci kompatibility
- Okamžik přiřazení
- Okamžik volání podprogramu, parametr odkazem
- Vyčíslování výrazu



• V každém jazyce implementována jinak

- V každém jazyce implementována jinak
- Striktní je implementována všude (téměř)

- V každém jazyce implementována jinak
- Striktní je implementována všude (téměř)
- Volná (žádná) je implementována částečně

- V každém jazyce implementována jinak
- Striktní je implementována všude (téměř)
- Volná (žádná) je implementována částečně
- Primární vlastnost ochrana před chybami

- V každém jazyce implementována jinak
- Striktní je implementována všude (téměř)
- Volná (žádná) je implementována částečně
- Primární vlastnost ochrana před chybami
- Někdy komplikuje zápis textu programu



• Ano: Čistota jazyka

- Ano: Čistota jazyka
- Ano: Jazyk pro výuku

- Ano: Čistota jazyka
- Ano: Jazyk pro výuku
- Ano: Tvorba bezpečných aplikací

- Ano: Čistota jazyka
- Ano: Jazyk pro výuku
- Ano: Tvorba bezpečných aplikací
- Ne: Zjednodušení zápisu programu

- Ano: Čistota jazyka
- · Ano: Jazyk pro výuku
- Ano: Tvorba bezpečných aplikací
- Ne: Zjednodušení zápisu programu
- Ne: Umožnění zápisu obecných struktur

- Ano: Čistota jazyka
- Ano: Jazyk pro výuku
- Ano: Tvorba bezpečných aplikací
- Ne: Zjednodušení zápisu programu
- Ne: Umožnění zápisu obecných struktur
- Ne: Tlak praxe



Volná kontrola se nedá zostřit

- Volná kontrola se nedá zostřit
- Striktní se dá uvolnit obejít

- Volná kontrola se nedá zostřit
- Striktní se dá uvolnit obejít
- Přetypování

- Volná kontrola se nedá zostřit
- Striktní se dá uvolnit obejít
- Přetypování
- Použití konverzních podprogramů

- Volná kontrola se nedá zostřit
- Striktní se dá uvolnit obejít
- Přetypování
- Použití konverzních podprogramů
- Použití beztypových struktur

- Volná kontrola se nedá zostřit
- Striktní se dá uvolnit obejít
- Přetypování
- Použití konverzních podprogramů
- Použití beztypových struktur
- Přetěžování operací

• Změna datového typu, přepočet hodnoty v paměti

- Změna datového typu, přepočet hodnoty v paměti
- Příklady explicitní konverze: Str, Val

- Změna datového typu, přepočet hodnoty v paměti
- Příklady explicitní konverze: Str, Val
- Příklady implicitní konverze:

```
R:=B; - konverze celočíselné hodnoty na reálnou read(S); - konverze posloupnosti znaků na řetězec write(R) - konverze reálné hodnoty na posloupnost znaků
```

Konverze

- Změna datového typu, přepočet hodnoty v paměti
- Příklady explicitní konverze: Str, Val
- Příklady implicitní konverze:
 R:=B; konverze celočíselné hodnoty na reálnou
 read(S); konverze posloupnosti znaků na řetězec
 write(R) konverze reálné hodnoty na posloupnost
 znaků
- Uživatelské konverze vždy s tvorbou uživatelských typů

Konverze

- Změna datového typu, přepočet hodnoty v paměti
- Příklady explicitní konverze: Str, Val
- Příklady implicitní konverze:
 R:=B; konverze celočíselné hodnoty na reálnou
 read(S); konverze posloupnosti znaků na řetězec
 write(R) konverze reálné hodnoty na posloupnost
 znaků
- Uživatelské konverze vždy s tvorbou uživatelských typů
- Příklad: textová reprezentace hodnot výčtových typů

• Změna datového typu BEZ přepočtu hodnoty v paměti

- Změna datového typu BEZ přepočtu hodnoty v paměti
- Podmínka: oba typy musí zabírat v paměti stejně velký prostor

- Změna datového typu BEZ přepočtu hodnoty v paměti
- Podmínka: oba typy musí zabírat v paměti stejně velký prostor
- Identifikátor typu je zároveň v roli přetypovací funkce

- Změna datového typu BEZ přepočtu hodnoty v paměti
- Podmínka: oba typy musí zabírat v paměti stejně velký prostor
- Identifikátor typu je zároveň v roli přetypovací funkce
- Příklad:

- Změna datového typu BEZ přepočtu hodnoty v paměti
- Podmínka: oba typy musí zabírat v paměti stejně velký prostor
- Identifikátor typu je zároveň v roli přetypovací funkce
- Příklad:

 Nekontroluje se, zda hodnota v paměti odpovídá povolené hodnotě cílového typu



Beztypové struktury

Variantní záznam

Beztypové struktury

- Variantní záznam
- Obecný ukazatel

Beztypové struktury

- Variantní záznam
- Obecný ukazatel
- Soubor bez udání typu



• Schéma záznamu je rozšířeno:

```
record
A: byte; {pevná složka}
  case B: Boolean of {rozlišovací složka}
  false: (L: longint); {variantní složky}
  true: (M: mnozina)
end;
```

• Schéma záznamu je rozšířeno:

```
record
A: byte; {pevná složka}
  case B: Boolean of {rozlišovací složka}
  false: (L: longint); {variantní složky}
  true: (M: mnozina)
end;
```

• Původní účel: úspora paměti

• Schéma záznamu je rozšířeno:

```
record
A: byte; {pevná složka}
  case B: Boolean of {rozlišovací složka}
  false: (L: longint); {variantní složky}
  true: (M: mnozina)
end;
```

- Původní účel: úspora paměti
- Původní implementace: rozlišovací složka určuje viditelnost variantních složek

Schéma záznamu je rozšířeno:

```
record
A: byte; {pevná složka}
  case B: Boolean of {rozlišovací složka}
  false: (L: longint); {variantní složky}
  true: (M: mnozina)
end;
```

- Původní účel: úspora paměti
- Původní implementace: rozlišovací složka určuje viditelnost variantních složek
- Novější implementace: Rozlišovací složka nemusí mít identifikátor, kontrola viditelnosti se neprovádí



 Příklad využití variantního záznamu bez rozlišovací složky:

```
type zaznam = record
      case boolean of
       false: (L: longint);
       true: (M: mnozina)
       {definice typu množina je stejná}
     end:
var Z: zaznam;
begin readln(Z.L);
      for I:=0 to 31 do
         if I in Z.M then write('1')
                      else write('0')
end.
```



• (podrobněji ukazatele v další přednášce)

- (podrobněji ukazatele v další přednášce)
- Deklarace standardním identifikátorem pointer

- (podrobněji ukazatele v další přednášce)
- Deklarace standardním identifikátorem pointer
- Manipulace s obecným ukazatelem: Vždy je potřebné dodat překladači informaci související s datovým typem

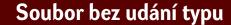
- (podrobněji ukazatele v další přednášce)
- Deklarace standardním identifikátorem pointer
- Manipulace s obecným ukazatelem: Vždy je potřebné dodat překladači informaci související s datovým typem
- Alokace paměti: GetMem (P, N) dodána informace o velikosti dynamické proměnné

- (podrobněji ukazatele v další přednášce)
- Deklarace standardním identifikátorem pointer
- Manipulace s obecným ukazatelem: Vždy je potřebné dodat překladači informaci související s datovým typem
- Alokace paměti: GetMem(P, N) dodána informace o velikosti dynamické proměnné
- Dealokace paměti: FreeMem(P, N) jako u alokace, pozor na stejnou velikost uvolňované paměti

- (podrobněji ukazatele v další přednášce)
- Deklarace standardním identifikátorem pointer
- Manipulace s obecným ukazatelem: Vždy je potřebné dodat překladači informaci související s datovým typem
- Alokace paměti: GetMem (P, N) dodána informace o velikosti dynamické proměnné
- Dealokace paměti: FreeMem(P, N) jako u alokace, pozor na stejnou velikost uvolňované paměti
- Práce s dynamickou proměnnou: VŽDY je potřebné přetypování.

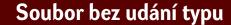
- (podrobněji ukazatele v další přednášce)
- Deklarace standardním identifikátorem pointer
- Manipulace s obecným ukazatelem: Vždy je potřebné dodat překladači informaci související s datovým typem
- Alokace paměti: GetMem(P, N) dodána informace o velikosti dynamické proměnné
- Dealokace paměti: FreeMem(P, N) jako u alokace, pozor na stejnou velikost uvolňované paměti
- Práce s dynamickou proměnnou: VŽDY je potřebné přetypování.
- Příklad:

```
var P: pointer;
begin GetMem(P, SizeOf(real));
    readln(real(P^));
```



Příklad ukládání řetězce na minimálním paměťovém prostoru:

```
var P: pointer;
   S: string;
begin readln(S); {běžné přečtení}
   GetMem(P, Length(S)+1);
   {přesná alokace podle dat}
   string(P^):=S;
   {přiřazení do získaného prostoru}
   ...
```

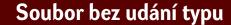


 Netextový soubor, vyjadřuje obecný pohled na soubor jako na posloupnost nespecifikovaných bytů

- Netextový soubor, vyjadřuje obecný pohled na soubor jako na posloupnost nespecifikovaných bytů
- Soubor s udaným typem nese informaci o velikosti a typu složky, ve všech operacích se tato informace využívá

- Netextový soubor, vyjadřuje obecný pohled na soubor jako na posloupnost nespecifikovaných bytů
- Soubor s udaným typem nese informaci o velikosti a typu složky, ve všech operacích se tato informace využívá
- Příklad manipulace s běžným netextovým souborem:

```
var F: file of char;
B: string;
begin assign(F,'neco.txt');
    reset(F); read(F,B[4]); close(F);
    reset(F); seek(F,6); write(F,B[5]);
    rewrite(F); write(...);
    seek(F,1); read(F,...)
```



• Soubor bez udaného typu – deklarace:

```
var F: file;
```

• Soubor bez udaného typu – deklarace:

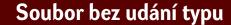
```
var F: file;
```

 Ve všech operacích se dodávají informace související s požadovaným typem:

• Soubor bez udaného typu – deklarace:

```
var F: file;
```

- Ve všech operacích se dodávají informace související s požadovaným typem:
- Příklady: assign(F,'bin.bin'); reset(F,1); - dodána informace o velikosti složky blockread(F,V,200,N); - dodána informace o počtu složek, není prováděna typová kontrola rewrite(F,N); - analogie operace reset blockwrite(F,V,100); - analogie operace blockread seek(F,10); - odvolání na číslo bloku, číslování v souboru je od nuly



• Příklady manipulací:

```
var G: file;
begin assign(G,...);
    reset(G,1);
    seek(G,10);
    reset(G,2);
    seek(G,10);
```

• Příklady manipulací:

```
var G: file;
begin assign(G,...);
    reset(G,1);
    seek(G,10);
    reset(G,2);
    seek(G,10);
```

• Aplikace – drtivá většina netextových souborů v praxi

• Příklady manipulací:

```
var G: file;
begin assign(G,...);
    reset(G,1);
    seek(G,10);
    reset(G,2);
    seek(G,10);
```

- Aplikace drtivá většina netextových souborů v praxi
- Příklad "databázového" souboru:

```
var DB: file;
  nazev: string[8];
  delka: byte;
  pocet: byte;
```

pokračování příkladu

```
begin assign(DB,'data.db');
      rewrite (DB, 1);
      pocet:=0; blockwrite(DB, Pocet, 1);
      while not eof do begin
         inc (pocet);
         readln(nazev, delka);
        blockwrite(DB, nazev[1], 8);
        blockwrite(DB, delka, 1);
      end;
      seek(DB, 1);
      blockwrite (DB, pocet, 1);
      close (DB);
end.
```