KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO V OLOMOUCI

Strukturované datové typy

7. cvičení

Jiří Zacpal

KMI/ZP1 – Základy programování 1

Odvozené datové typy

- definujeme pomocí konstrukce `typedef`
- nevytváří nové datové typy, pouze vytvoří **jméno** pro existující typ
- syntaxe definice typu:
  ```
  typedef zápis_ve_tvaru_deklarace;
  ```
- příklady definice typu:
  ```
  typedef int cele_cislo;
  typedef int male_pole[10];
  typedef unsigned long ULong;
  typedef const ULong CULong;
  ```
- příklady deklarace proměnné:
  ```
  male_pole moje_cisla;
  ULong velke_cislo;
  ```

Výčtové datové typy

- definujeme pomocí konstrukce `enum`
- používá se pro definici většího počtu spolu souvisejících celočíselných konstant
- syntaxe definice typu:

```
enum identifikátor_typu{
   id_clenu_1 =
konstantní_výraz_1,
   …
   id_clenu_N = konstantní_výraz_N
} id_prom_1=hod_1, …, id_prom_M
=hod_M;
```

Příklady k výčtovému typy

- definice výčtového typu:
```
enum Bool{TRUE=1, FALSE=0};
enum Bool{TRUE=1, FALSE=0} splnen=TRUE;
enum {FALSE, TRUE} splnen=1;
enum Den{Po,Ut,St,Ct,Pa,So=10,Ne}
muj_den;
```
- deklarace proměnných výčtového typu:
```
enum Bool splneno;
enum Bool splneno = 0;
enum Den muj_den = Po;
```
- použití proměnných a konstant výčtového typu:
```
splneno = TRUE;
muj_den = 3;
cislo = muj_den * So + splneno;
```

Výčtový typ pomocí typedef

- nejpřehlednější a nejpoužívanější způsob
- příklady definice typu:

```
typedef enum{
   TRUE = 1, FALSE = 0
} Bool;

typedef enum{
   Po, Ut, St, Ct, Pa, So=10, Ne
} Den;
```

- příklady deklarace proměnných:

```
Bool splneno;
Bool splneno = 0;
Den muj_den = Po;
```

Strukturovaný datový typ

- definujeme pomocí konstrukce `struct`
- používá se pro uložení více souvisejících hodnot, které mohou být různého typu (na rozdíl od pole)
- syntaxe definice typu:

```
struct identifikátor_typu{
   typ_clenu_1 id_clenu_1;
   …
   typ_clenu_N id_clenu_N;
} id_pro1={inic_1}, …,
id_pro_M={inic_M};
```

Příklady ke strukturovanému typu

- definice strukturovaného typu:

```
struct Datum{char Den, Mesic; short
Rok;};
struct Datum{char Den, Mesic; short
Rok;}
   narozen = {24, 3, 1972};
struct {char Den, Mesic; short Rok;}
   narozen = {24, 3, 1972};
```

- deklarace proměnných strukturovaného typu:

```
struct Datum narozen;
struct Datum narozen = {24, 3, 1972};
```

Struktura pomocí typedef

- příklad definice typu:

```
typedef struct {
   char Den, Mesic; short Rok;
} Datum;
```

- příklad deklarace proměnných:

```
Datum narozen;
Datum narozen = {24, 3, 1972};
```

Přístup ke členům struktury

- pomocí operátoru tečka
- příklady:

```
Datum muj_den, narozen, dnes;
int vek;
…
muj_den.Den = 24;
muj_den.Mesic =
1+(muj_den.Mesic+6)%12;
vek = dnes.Rok - narozen.Rok;
```

Pár poznámek ke strukturám

- člen struktury může být libovolného typu, pouze nesmí být stejného typu jako právě definovaná struktura (lze obejít užitím ukazatelů, viz příští seminář)
- je možné přiřadit hodnoty všech členů jedné struktury (proměnné strukturovaného typu) druhé struktuře, pokud jsou stejného typu
- operace se strukturami:
  - kopírování
  - přiřazení struktuře
  - získání adresy
  - přístup k jejím složkám
- příklady:

```
struct {char pole[10]; int i;} s;
s.pole[0]='A';

muj_den = dnes;
```

Příklad 1

```
#define RAD 20
#define SLO 30

typedef enum{FALSE,TRUE}bol;
typedef struct{int x;int y;} bod;
typedef struct{bod b1;bod b2;} obdelnik;

bod vytvor_bod(int x, int y)
{
    bod pom;
    pom.x=x;
    pom.y=y;
    return pom;
}

obdelnik vytvor_obdelnik(bod b1, bod b2)
{
    obdelnik pom;
    pom.b1=b1;
    pom.b2=b2;
    return pom;
}
```

Příklad 1

```
bol je_v_obdelniku(bod b, obdelnik o)
{
            if((b.x>=o.b1.x)&&(b.x<=o.b2.x)&&(b.y>=o.b1.y)&&(b.y<=o.b2.y)) return TRUE; else return
FALSE;
}

void nakresli_obdelnik(obdelnik o)
{
     int i,j;
     bod pom;
     for(i=0;i<=RAD;i++)
     {
          for(j=0;j<=SLO;j++)
          {
                 pom=vytvor_bod(j,i);
                 if(je_v_obdelniku(pom,o)) printf("x"); else printf(".");
          }
     printf("\n");
     }
}

int main()
{
     bod A=vytvor_bod(4,2);
     bod B=vytvor_bod(10,8);
     obdelnik O=vytvor_obdelnik(A,B);
     nakresli_obdelnik(O);
}
```

Typ union

- používá se v případech, kdy je třeba se stejnými daty pracovat „různými způsoby“
- definice má obdobnou syntaxi jako definice strukturovaného typu, pouze klíčové slovo `struct` nahradíme slovem `union`
- na rozdíl od struktur využívají jednotlivé členy stejnou paměť (existují jedna data s více možnostmi přístupu)
- inicializací lze přiřadit hodnotu „prvnímu členu“
- příklad definice:

```
union {
    unsigned char c[4];
    int i;
} u = {0,1,0,0};
```