Programovací jazyk C
piata časť

Miroslav Melicherčík

Štruktúra

- heterogénny dátový typ
- záznam je zložený z dátových prvkov rôznych typov
- príklad:
  informácie o človeku - meno
  - priezvisko
  - dátum narodenia
  - miesto narodenia
  - rodné číslo
  - adresa
  - tel. číslo
  - hrubá mzda

Štruktúra

- štruktúry je možné do seba vnárať
- zadefinovanú štruktúru ako nový dátový typ je možné použiť aj v inej štruktúre
- štruktúru je možné použiť ako návratový typ funkcie
- informácie o človeku
  - meno
  - priezvisko
  - dátum narodenia
  - miesto narodenia
  - rodné číslo
  - adresa
    - ulica
    - číslo
    - mesto
    - PSČ
  - tel. číslo
  - hrubá mzda

Definícia štruktúry

- 5 spôsobov definície štruktúry
- 1 spôsob

```
struct {
   int   vyska;
   float vaha;
} Milan, Pavol, Peter;
```

Definícia štruktúry

- 2 spôsob
- štruktúra je pomenovaná a dá sa neskôr použiť v programe

```
struct miery{
   int   vyska;
   float vaha;
} Milan, Pavol, Peter;
```

Definícia štruktúry

- 3 spôsob

```
struct miery{
   int   vyska;
   float vaha;
};

struct miery Milan, Pavol, Peter;
```

POZOR NIE: `miery Milan, Pavol, Peter;`

Definícia štruktúry

- 4 spôsob
- máme vytvorený dátový typ
- je možné ho použiť aj pre pretypovani

```
typedef struct {
   int   vyska;
   float vaha;
} MIERY;

MIERY Milan, Pavol, Peter;
```

POZOR NIE: `struct MIERY Milan, Pavol, Peter;`

Definícia štruktúry

- 5 spôsob
- je pomenovaná aj štruktúra aj nový dátový typ
- tento spôsob sa používa keď sa štruktúra odkazuje na samu seba

```
typedef struct miery {
   int   vyska;
   float vaha;
} MIERY;

MIERY Milan, Pavol, Peter;
```

Použitie

- pri statickej definícií

```
MIERY Milan, Peter;
Milan.vyska = 180;
Milan.vaha  = 75.8;


Peter = Milan;   //priradenie funguje


MIERY zamestnanec[100];
zamestnanec[0].vyska = 180;
zamestnanec[99].vyska = 165;
```

Pointer na štruktúru

```
typedef struct {
   int   vyska;
   float vaha;
} MIERY;


MIERY Peter, *p_Peter;

p_Peter = &Peter;
```

Pointer na štruktúru

```
typedef struct {
   int   vyska;
   float vaha;
} MIERY, *P_MIERY;


MIERY Peter;
P_MIERY p_Peter;
p_Peter = (P_MIERY) malloc(sizeof(MIERY));
```

Pointer na štruktúru

- prístup ku štruktúre

```
MIERY m;
  m.vyska = 180;
```

```
MIERY *m;
  (*m).vyska = 180;
```

```
P_MIERY m;
```

```
MIERY *m;
  m->vyska = 180;
```

```
P_MIERY m;
```

Inicializácia štruktúry

```
typedef struct {
  int   vyska;
  float vaha;
} MIERY;


MIERY Milan = {180, 75.8};
```

Typ vymenovaním

```
typedef enum {
   Pondelok,
   Utorok,
   Streda,
   Stvrtok,
   Piatok,
   Sobota,
   Nedela
} DNI;


DNI t;

t = Pondelok;
```

Typ vymenovaním

```
typedef enum {
   Pondelok = 2,
   Utorok = 3,
   Streda = 4,
   Stvrtok = 5,
   Piatok = 6,
   Sobota = 7,
   Nedela = 1
} DNI;

DNI d;

printf("Dnes je %s\n", d);              ZLE
printf("Dnes je %d\n", (int) d);        DOBRE
```

Union

- pracuje podobne ako štruktúra
- v unione môže byť súčasne uložená len jedna hodnota
- v pamäti sa vyhradí miesto podľa najdlhšej položky

Union

- pracuje podobne ako štruktúra
- v unione môže byť súčasne uložená len jedna hodnota
- v pamäti sa vyhradí miesto podľa najdlhšej položky

```
typedef union {
  int   vyska;
  float vaha;
} MIERY;
```