Výčtové a strukturované datové typy

Základy programování 1

Martin Kauer

Motivace

- Se základními datovými typy si sice vystačíme
- Někdy to ale může být nepříjemně nepřehledné
- Zejména když popisujeme sadu složitějších objektů
- Příklady:

```
int dnes[3]; // datum
dnes[0] = 5; dnes[1] = 11; dnes[2] = 2014;

char jmeno_osoba[] = "Petr";
int narozen_osoba[3] = {30, 4, 1995};
```

Vlastní datové typy

- Definujeme pomocí konstrukce `typedef`
- Definice typu obecně:
  `typedef` *`zápis_ve_tvaru_definice_proměnné`*`;`
- Příklady definice typu:

```
typedef int cele_cislo;
typedef int pole_deseti_cisel[10];
typedef unsigned long ULong;
typedef const ULong CULong;
typedef char string[];
```

- Příklady použití typu (definice proměnné):

```
pole_deseti_cisel moje_cisla;
ULong velke_cislo;
string jmeno = "Ales";
```

Výčtové datové typy

- Definujeme obvykle pomocí `typedef` a `enum`
- Používá se pro definici spolu souvisejících celočíselných konstant
- Definice typu obecně:

```
typedef enum {
    jmeno1 = hodnota1,
    ...
    jmenoN = hodnotaN
} jmeno_typu;
```

- Příklad definice:

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

- Příklad použití typu:

```
boolean vysledek = TRUE;
```

Příklady – výčtové typy

```
typedef enum {
    FALSE, TRUE
} boolean;

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

boolean splneno;
boolean splneno = 1;
boolean splneno = TRUE;
den ZP1 = St;

splneno = ZP1 * 5 - TRUE + Ut * So;
```

Příklady – výčtové typy

```
typedef enum {
    FALSE, TRUE
} boolean;

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

boolean splneno;
boolean splneno = 1;
boolean splneno = TRUE;
den ZP1 = St;

splneno = ZP1 * 5 - TRUE + Ut * So; // ZLO
```

Strukturované datové typy

- Definujeme obvykle pomocí `typedef` a `struct`
- Uložení více souvisejících hodnot do jedné proměnné
- Hodnoty mohou být různých typů (rozdíl od pole)
- Definice strukturovaného typu obecně:

```
typedef struct {
    typ1 jmeno_clenu1;
    ...
    typN jmeno_clenu1;
} jmeno_typu;
```

- Příklad definice:

```
typedef struct {
    char den;
    char mesic;
    short rok;
} datum;
```

Příklady – strukturované typy

- Strukturované proměnné definujeme obvyklým způsobem:

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

- Přístup ke členům proměnné pomocí operátoru tečka:

```
datum muj_den, narozen, dnes;
int vek;
…
muj_den.den = 24;
muj_den.mesic = 1 + (muj_den.mesic + 6) % 13;
vek = dnes.rok - narozen.rok;
```

Struktury vs. pole

- **Struktury** mohou obsahovat položky různých datových typů
- Používáme je pro uložení různých informací o nějakém jednom celku
- Struktury lze kopírovat přiřazením:

```
narozen = dnes;
```

- **Pole** mohou obsahovat pouze položky jediného typu
- Používáme je pro uložení více datových položek, se kterými chceme pracovat jendotným způsobem
- Pole **nelze** kopírovat přiřazením:

```
int cisla[] = { 2, 5, 11, 17, 23 };
int data[5];

data = cisla;
```

Složitější struktury

- Členem struktury může být jakýkoli datový typ
- Včetně polí, výčtových a strukturovaných datových typů:

```
typedef struct {
    char jmeno[20];
    datum narozen;
} osoba;
osoba teta = { "Eva", { 12, 3, 1989 } };
teta.jmeno[1] = 'm';
teta.narozen.den = 11;
```

- Členem struktury **nemůže** být ta samá struktura:

```
typedef struct {
    char jmeno[20]; osoba partner;
} osoba;
```

- Lze obejít použitím ukazatelů

Cvičení – Práce se zlomky

Vytvořte strukturovaný datový typ pro reprezentaci zlomků. Naprogramujte funkce pro součet, rozdíl, součin a podíl dvou zlomků, které vrací výsledný zlomek jako svou návratovou hodnotu. Vrácený zlomek musí být upraven do základního tvaru. Vytvořenou strukturu i funkce náležitě otestujte.

Požadavky na odevzdanou práci:

- Jeden soubor s příponou .c, který bude obsahovat, definici daného strukturovaného typu, všechny zmíněné funkce a nebude obsahovat funkci `main`. Datová struktura se musí jmenovat **`zlomek`**, funkce se musí jmenovat **`soucet, rozdil, soucin, podil`**.
- Žádná z funkcí nesmí nic vypisovat na obrazovku ani obsahovat žádná blokující volání typu scanf(…); getch(); system(“pause”); apod.
- Všechny funkce musí vracet výsledný zlomek jako svou návratovou hodnotu.
- Všechny funkce musí být robustní, musí si poradit s nekorektními vstupy.
- Pokud z nějakého důvodu nelze operaci provést funkce musí vrátit zlomek 0/0.