Základy programování v C++ ... 10. cvičení
Výčtový datový typ
Struktura

Zuzana Petříčková

23. října 2019

Přehled

1. Neobjektové datové typy definované uživatelem
   - Zavedení nového jména pro existující typ
   - Výčtový typ
   - Struktura

Neobjektové datové typy definované uživatelem

Zavedení nového jména pro existující typ

```
typedef typ nazev; // starsi zpusob
using nazev = typ; // novejsi zpusob od C++11
```

- podobný význam jako deklarace pojmenované konstanty → zpřehlednění programu, později stačí změnit typ na jednom místě (např. na požádání změnit přesnost z **int** na **long long**)

```
typedef int cislo;
// using cislo = int;
typedef cislo malePole[10];
// using malePole = cislo[10];
typedef unsigned int UInt;
typedef const UInt CUInt;
...
cislo x = 7;              // deklarace promenne
malePole p = {1, 2};
...
cislo minimum(cislo a, cislo b) // funkce
```

Výčtový typ

- usnadňuje práci s veličinami, které mohou mít malou množinu hodnot (dny v týdnu, barvy, chyby programu,...)

Deklarace

```
enum nazev {seznam_polozek};
enum nazev {seznam_polozek} seznam_deklaratoru;
```

- **nazev** je název (identifikátor) nového typu
- položky jsou identifikátory (popř. doplněné o hodnotu, kterou představují)
- **seznam_deklaratoru** je seznam proměnných daného typu

```
enum Den{pondeli,utery,streda,ctvrtek,patek,sobota,nedele};
enum SvetovaStrana {sever,jih,vychod,zapad} smer;
...
Den dnes = pondeli;
```

Výčtový typ

```
enum Den{pondeli,utery,streda,ctvrtek,patek,sobota,nedele};
enum SvetovaStrana {sever,jih,vychod,zapad} smer;
```

- položky jsou reprezentovány celými čísli (počínaje od 0, hodnota každé položky je vždy o 1 větší než je hodnota předchozí)
- hodnoty položek lze předefinovat konstantním výrazem:

```
enum Den {pondeli,utery,streda,ctvrtek=8,patek,sobota,ne
enum SvetovaStrana {sever=1,jih,vychod=4,zapad=8};
```

- **cout** nevypíše jméno položky, ale odpovídající celočíselnou hodnotu.

Výčtový typ

- jedná se o celočíselný datový typ a lze jej použít v podmínce příkazu **switch**:

```
SvetovaStrana kam;
...
switch (kam)
{
case sever:
    cout << "Jdu na sever";
    break;
case jih:
    cout << "Jdu na jih";
    break;
    ...
}
```

Výčtový typ ... příklady

Pro výčtové typy **Den** (pro 7 dní v týdnu) a **SvetovaStrana** (pro 4 světové strany) definujeme následující funkce:

- Funkce **bool vikend(Den den)** vrátí **true**, pokud je **den** sobota nebo nedele.
- Funkce **void vypisDen(Den den)** vypíše na konzoli, jaký je **den**, např. "Dnes je streda".
- Funkce **bool muzuJit(SvetovaStrana kam)** vrátí **true**, pokud je hodnota **kam** sever nebo jih
  (lze řešit s využitím bitových operací & a |)

Výčtový typ - řešení příkladů

```
enum Den {pondeli, utery, streda, ctvrtek=8, patek, sobota, nedele};
enum SvetovaStrana {sever=1, jih, vychod=4, zapad=8};
bool vikend(Den den)
{
    return (den == sobota || den == nedele);
}
void vypisDen(Den den)
{
    const static string jmena[] = {"pondeli","utery",
                "streda","ctvrtek","patek","sobota","nedele"};
    if (den < 7 && den >= 0)
        cout << "Dnes je " << jmena[den] << endl;
}
bool muzuJit(SvetovaStrana kam)
{
    int muzu = sever | jih;
    return ((int)kam & muzu);
}
```

Struktura

- skupina proměnných různých typů, se kterou se pracuje jako s celkem

Deklarace

```
struct identifikator
{
    deklarace_slozek
};
struct identifikator
{
    deklarace_slozek
} seznam_deklaratoru;
```

- **identifikator** je název (identifikátor) struktury
- složka může být proměnná libovolného (jiného) typu
- **seznam_deklaratoru** je seznam proměnných daného typu

Struktura

```
struct Datum
{
    unsigned short den;
    unsigned short mesic;
    int rok;
};

struct KomplexniCislo
{
    float re;
    float im;
};
```

- je konvence umísťovat definice struktur (a definice datových typů obecně) na začátek zdrojového souboru (nad definice funkcí), popřípadě do **hlavičkového souboru**

Struktura

Deklarace proměnné a její inicializace

```
Datum datum; // bez inicializace
Datum narozen = {1,1,1998};
Datum dnes = {29,10,}; // rok bude 0
Datum terminyZkousek[10]; // staticke pole
```

Struktura

Přístup ke složkám struktury

- pomocí operátoru **.** (tečka)

```
Datum dnes = {29,10,2018};

Datum datum;
datum.den = 28;
datum.mesic = 10;
datum.rok = 1918;

int uplynuloLet = dnes.rok - datum.rok;

Datum vcera = dnes;
vcera.den--;
```

Struktura

Kopírování struktury

- při přiřazení se kopírují všechny složky struktury včetně obsahu statických polí
- pokud je složkou statické pole, nemusím ho kopírovat prvek po prvku

```
struct Predmet
{
    string nazev;
    Datum terminyZkousek[5];
    ...
};
...
Predmet matalyza = {"Matematicka analyza", {{1,2,2019},{10,2
Predmet lingebra = matalyza;
/* netreba:
for (int i = 0; i < 5; i++)
    lingebra.terminyZkousek[i] = matalyza.terminyZkousek[i];
*/
```

Struktury - příklady

- Definujte výčtový typ **Pozice** (pro několik pracovních pozic) a struktury **Datum** (se složkami den, mesic a rok) a **Zamestnanec** se složkami jmeno (string), prijmeni (string), nastup (Datum), plat (int) a pozice (Pozice).
- Deklarujte lokální proměnné typu Zamestnanec a naplňte je hodnotami.
- Deklarujte pole zaměstnanců **pobockaPraha** a naplňte ho hodnotami.
- Napište ,,uživatelsky přívětivou" funkci **void vypisZamestnance(Zamestnanec pobocka[], int n)**, která za využití cyklu vypíše obsah pole **pobocka** délky **n** na konzoli.
- (na procvičení) Napište ,,uživatelsky přívětivou" funkci **void nactiZamestnance(Zamestnanec pobocka[], int n)**, která za využití cyklu načte obsah pole **pobocka** z konzole.

Struktury - část řešení příkladů

```
struct Datum
{
        unsigned short den;
        unsigned short mesic;
        int rok;
};
enum Pozice  {vratny, programator, ucetni};
const string nazevPozice[] = {"vratny",
                "programator", "ucetni"};
struct Zamestnanec
{
        string jmeno;
        string prijmeni;
        Datum nastup;
        int plat;
        Pozice pozice;
};
```