

## Programovanie v C++ - 3. diel

Seriál o programovaní v programovacom jazyku C++. V treťom diele si bližšie opíšeme funkcie a zoznámime sa s premennými.

V predchádzajúcej časti sme si opísali, čo je to funkcia, zoznámili sme sa so základnými dátovými typmi jazykov C a C++. V tejto časti sa pozrieme na zúbok veľmi dôležitej súčasti každého programovacieho jazyka, premenným, a bližšie si vysvetlíme funkcie.

### Premenné

Premenné sú oblasti v pamäti určitého typu, ktoré môžu nadobúdať nejakú hodnotu. Typ premennej môže byť akýkoľvek základný dátový typ, s ktorým sme sa oboznámili v ostatnom diele, alebo, ako sa neskôr dozvieme, aj typ nami vytvorený.

Deklarácia premennej vyzerá nasledovne: `__typ __názov_premennej ;`

`int premenna1; // v pamäti sa vyhradí 4B`

`bool premenna2; // v pamäti sa vyhradí 1B` Keď deklarujeme premennú, tak kompilátoru oznámime, že existuje nejaká premenná nejakého typu, ale pamäť sa alokovať nebude. Na to, aby nám kompilátor požadovanú pamäť vyhradil, musíme premennú aj definovať, čiže jej priradiť hodnotu z jej oboru hodnôt. `int premenna1; // deklarácia`

`int premenna2 = 1; // definícia`

Premenna1 je iba deklarovaná, pretože nie je inicializovaná nijakou hodnotou. Zato premenna2 je inicializovaná hodnotou 1, teda je definovaná. Na uvedenom príklade vydlíme, že premenné možno v C/C++ aj automaticky inicializovať (priradiť im hodnotu). Ešte pripomeniem, že znamienko "=" (rovná sa) je operátorom priradenia. `int premenna1 = 0; // celočíselná 0`

`float premenna2 = 0.0f; // reálna 0`

`bool premenna3 = true;` V prvom prípade priraďujeme premennej premenna1, ktorá je typu int celočíselnú hodnotu 0. V druhom máme premennú typu float, pričom jej priradíme hodnotu 0.0f, čiže tiež nula, ale reálneho charakteru. I keď to vyzerá tak, že medzi týmito dvoma nulami nie je prakticky žiadny rozdiel, je tam! Keby ste chceli priradiť 0.0f premennej typu int, niektoré kompilátory by mohli vyhadzovať varovanie o nekompatibilitu dátových typov, poprípade o možnej strate dát, alebo nejaké podobné hlásenia. `int premenna1 = 0.0f; // bude obsahovať hodnotu 0`

`int premenna2 = 0.4f; // bude obsahovať hodnotu 0!`

`int premenna3 = 0.6f; // bude obsahovať hodnotu 1!` Všimnite si, že keď premennej celočíselného typu priradíte reálnu hodnotu, do celočíselnej premennej sa uloží jej zaokrúhlená verzia (zaokrúhľuje sa podľa normálnych pravidiel matematiky). V opačnom prípade, čiže keby ste priraďovali premennej reálneho typu celočíselnú hodnotu sa nič podobné stať nemôže, celé čísla sú totiž podmnožinou reálnych čísel.

Premenné sa dajú rozdeliť nielen podľa typu, ale aj podľa toho, či sú globálne, alebo lokálne. Globálne premenné sú také, ktoré deklarujeme/definujeme mimo akejkoľvek funkcie, teda v tele programu. Existujú počas celého behu programu, od jeho začiatku až po jeho koniec. Lokálne sú také, ktoré sú deklarované/definované v rámci funkcie. Existujú preto len po dobu vykonávania danej funkcie. `void globalna;`

```
void main() {
void lokalna;
}
Funkcie
```

Už sme sa naučili, že každý program napísaný v C/C++ musí obsahovať funkciu `main()` (poprípade jej ekvivalent). Táto funkcia sa v 99.9% prípadov vyskytuje len v definovanej podobe. To však pre ostatné funkcie platí nemusí. V C/C++ preto môžeme ako premenné, tak aj funkcie deklarovať a definovať. Deklarácia funkcie je vlastne len uvedenie jej typu a názvu bez toho, aby sme uvádzali telo: `int func(); // telo nie je uvedené` Definícia vyzerá takmer rovnako, len prídajú zátvorky a telo funkcie: `int func() {`

`// typ a názov return 0;`

`// telo }` Ak sa vám zdá, že funkcie a premenná zdieľajú mnoho spoločných vlastností, tak máte pravdu. Potvrďuje to aj to, že funkcie môžu byť takisto lokálne a globálne. `void globalna() {} // globalna funkcie`

```
void main() {
void lokalna() {} // lokalna funkcia
}
```

V tomto príklade definujeme funkciu `lokalna()` v rámci funkcie `main()`. To znamená, že bude prístupná iba a len z funkcie `main()`. Funkcia `globalna()` bude dostupná aj z funkcie `main()`, aj z funkcie `lokalna()`.

### Rozsah platnosti

Premenné aj funkcie majú svoj rozsah platnosti, čiže oblasť, v ktorej fungujú a v ktorej nie. Kompilátor C/C++ kompiluje zhora nadol. Keby sme teda vo funkcii `globalna()` zavolali funkciu `lokalna()`, kompilátor by nám oznámil chybu.

Situáciu vám bližšie objasní tento výpis: `/* voláme funkciu lokalna(), no tá je definovaná až pod touto, preto je neznáma */`

```
void globalna() {
lokalna();
}
```

```

/* v poriadku, pretože funkcia globalna() je definovaná nad funkciou lokalna() */
void main() {
    lokalna() {
        globalna();
    }
}

```

V niektorých prípadoch by bolo zdâhavé obchádzať toto "obmedzenie", preto je v C/C++ možné funkcie predefinovať. Predefinovať funkciu znamená deklarovať ju nad funkciou, v ktorej budeme danú funkciu volať, pričom jej definícia bude uvedená pod funkciou, z ktorej bude volaná. Možno to znie zložito, ale príklad vám všetko objasní. /\* tu funkciu lokalna() predefinujeme \*/

```

void lokalna();
/* OK, voláme funkciu lokalna(), ktorá je predefinovaná */
void globalna() {
    lokalna();
}

```

```

/* a tu je definícia funkcie lokalna() */
void main() {
    lokalna() {}
}

```

V prípade premenných prebieha všetko presne tak isto - premenná, ktorá je deklarovaná/definovaná pod premennou, ktorá ju volá pre ňu neexistuje. Hovorí o predefinovaní premenných asi ani nemá zmysel... int horna = dolna; // chyba, dolna nie je dostupná

```

int dolna = horna; // OK
...
int dolna; // deklarujeme premenuu dolna

int horna = dolna; // teraz je už všetko OK
dolna = horna; Záver

```

Oboznámili sme sa s mnohými faktami a keďže nechcem, aby ste si k programovaniu vytvorili odpor, tak som sa rozhodol, že toto by na dnes aj mohlo stačiť ;-). Možno sa vám niektoré veci budú zdať zložité na pochopenie, ale vedzte, že je to len a len zdanie. Akonáhle si to vyskúšate v praxi, hneď všetko pochopíte. V nasledujúcom diele za budeme zaoberať operátormi a urobíme si aj prvý jednoduchý programček. Teším sa na ďalšie stretnutie s vami a nezabúdajte, ak máte nejaké pripomienky k seriálu, napíšte mi email, alebo sa vyjadrite vo fóre. Do programovania.