C++ ťahák

Každý program napísaný v C++ má obsahovať jednu hlavnú funkciu `main()`. Preto najjednoduchší program vyzerá nasledovne.

```
int main(){
}
```

Nezabudnite, že každý program musíte pred spustením skompilovať (v DevC++ `F9`) – vytvoriť spustiteľnú binárku a až potom ho spustiť (v DevC++ `F10`). (V DevC++ môžete použiť aj `F11` – skompiluje a rovno spustí.)

Premenné

Premenná je krabička (miesto v pamäti), kde si program ukladá dáta. Každá premenná má svoj `typ` (určuje, aké dáta sa tam dávajú), `meno` (jednoznačný identifikátor) a samozrejme obsahuje `hodnotu` príslušného typu.

Typ	Slovný popis	Počet bitov	Rozsah	Poznámka
int	celé čísla	32	$-2^{31} .. 2^{31}-1$	
long long	celé čísla	64	$-2^{63} .. 2^{63}-1$	
char	znak	8		čísla z ASCII tabuľky
bool	pravda/nepravda	8 (občas aj 32 alebo 64)	`true/false`	
string	reťazec znakov			knižnica `string`

Premenné treba pred použitím zadeklarovať (povedať akého typu je a ako sa volá). Následne pomocou znaku `=` vieme do premennej priraďovať hodnotu alebo výraz zložený z matematických operátorov a iných premenných.

```
int main() {
    int a, b=4; //tu som si zadeklaroval dve premenne typu int, v b je teraz 4
    a = 4; //do a priradim tiez 4
    b = 2*b+a-3; //do b priradim hodnotu vyrazu napravo
}
```

Načítavanie vstupu a výpis výstupu – knižnica iostream

Na načítavanie a vypisovanie používame funkcie `cin` a `cout`, ktoré patria do knižnice `iostream`. Nezabudnite na magický riadok `using namespace std;`

```
#include <iostream>
using namespace std;

int main() {
    int a,b;
    //nacitam dve cisla na vstupe, prve do premennej a, druhe do b
    cin >> a >> b;
    //vypisem na vystup cislo b, medzeru, a, text " sucet ", ich sucet a+b a koniec riadka (enter)
    cout << b << "␣" << a << "␣sucet␣" << a+b << endl;
}
```

Načítavanie vstupu a výpis výstupu – knižnica cstdio

Na načítavanie a vypisovanie používame funkcie `scanf()` a `printf()`, ktoré patria do knižnice `cstdio`.

```
#include <cstdio>

int main() {
    int a,b;
    //nacitam dve cisla na vstupe, prve do premennej a, druhe do b
    scanf("%d␣%d",&a,&b);
    //vypisem na vystup cislo b, medzeru, a, text " sucet ", ich sucet a+b a koniec riadka (enter)
    printf("%d␣%d␣sucet␣%d\n",b,a,a+b);
}
```

Podmienky

Ak chceme aby sa program správal inak za splnení istej podmienky, použijeme príkaz `if`. Ten zjednodušene vyzerá `if(<podmienka>) <prikaz>;`. To znamená, `Ak platí <podmienka> vykonaj <príkaz>`. **Nezabudnite**, že porovnávanie dvoch vecí sa vykonáva pomocou **dvoch** rovná sa `==`. Ak chcem vykonať po splnení `if`u viac príkazov, uzavriem ich do kučeravých zátvoriek. Takto vyzerá použitie:

```
int main() {
    int x,p;
    //jednoducha podmienka if
    if(x == 4) p=8;
    //podmienka if s vetvou else
```

```
    if(p<10) x=x+2;
    else x=x+14;
    //vetvena podmienka if
    if(x%3 == 0) x=4;
    else if(x%3 == 1) {x=13; p=8;}
    else x=6;
}
```

Podmienka vie byť ľubovoľný vyraz, ktorý vieme vyhodnotiť ako `true` alebo `false`. Takisto však vieme skladať viac výrazov pomocou logických spojek `and` – `&&` a `or` – `||`.

Priraď do premennej x číslo 5, ak x sa rovná 4 **alebo** je p rovné 8 **a** x rovné 7.

```
if(x==4 || (p==8 && x==7)) x=5;
```

Cykly

Ak chceme aby sa nám nejaká časť programu opakovala viac krát, môžeme použiť cykly. Základný cyklus je príkaz `for`.

Na vypísanie čísiel od 1 po 10 môžeme použiť nasledovný kus programu.

```
for(int i=1; i<=10; i++)
    cout << i << endl;
```

Význam toho: vytvor premennú cyklu i. Kým platí podmienka ($i <= 10$) vykonávaj príkaz za `for`om a po každom vykonaní zmeň premennú i podľa posledného príkazu ($i++$ – pričítaj 1). Ak chcem vykonať viac príkazov v cykle, použijem **kučeravé zátvorky**.

Vypíš párne čísla medzi 20 až 40 by sa zapísalo:

```
for(int j=20; j<=40; j=j+2)
    cout << j << endl;
```

Všestrannejší spôsob cyklu v programe je použiť cyklus `while`, ktorý funguje `while(<podmienka>) <príkazy>;` – kým platí `<podmienka>` vykonávaj `<príkazy>`.

```
int i=1;
while(i<=10) {
    cout << i << endl;
    i++;
}
```

Spraví presne to, čo prvý `for` cyklus – vypíše čísla od 1 po 10.

Polia

Ak chceme naraz vytvoriť a používať množstvo premenných rovnakého typu, môžeme použiť `pole`. Pole deklaruje naraz množstvo premenných uložených za sebou, ku ktorým sa dá pristupovať pomocou ich pozície. Pozor, ak máme pole veľkosti 100, pozície sú číslované od 0, takže sa viem pozerať len na pozície 0 až 99 a **neviem sa pozerať** na pozíciu 100.

Pri deklarácií musí byť veľkosť poľa určená konštantou, nie premennou.

Tento program vypočíta prvých 30 Fibonacciho čísiel a uloží ho do pola a. Na pozícii i je i-te Fibonacciho číslo.

```
int main() {
    //vytvori pole intov s nazvom a, obsahujucim 47 intov
    int A[47];
    A[0]=0; A[1]=1;
    for(int i=2; i<30; i++)
        A[i] = A[i-1] + A[i-2];
}
```

A takisto, keďže aj pole je len typ premennej, viem robiť polia viacrozmerné – polia polí.

```
int A[15][20];
A[5][4] = 1;
A[3][8] = A[5][4];
int B[100][10][10];
```