**C++**

**Varijable**

**Što su varijable?**

Varijable postoje u svakom programskom jeziku. U varijablama pohranjujete nekakve vrednosti, brojeve, slova, i sve druge znakove.
Evo primjera kako rečenicu iz prošlog tutorijala upisati u varijablu i zatim pomoć varijable ispisati istu na ekran.

*#include<iostream>
using namespace std;
main()
{****string recenica;*** *recenica = "Hello World!";
cout << recenica;
char a;
cin >> a;
return 0;
}*

Dakle sa

***string recenica***

deklariramo da će varijabla ''recenica'' biti string odnosno skup nekakvih znakova. String je tip podatka. O tome više u idućem tutorijalu. Zapamtite samo da se svaka varijabla mora deklarirati prije pridruživanja vrijedosti.

recenica = ''Hello World'';

Sa ovime varijabli ''recenica'' pridružujemo vrijednost ''Hello World!''

cout << recenica;

Sa ovim ispisujemo varijablu. Primjetite da nema navodnika. Ako želite nešto pored toga napisati što nije sadržano u varijabli možete to napisati ovako.

cout << recenica << " What's up?";

Dakle prvo ide varijabla ''recenica'' bez navodnika, zatim opet stavljamo strelice za ispis i u navodnike pišemo nastavak rečenice. Primjetite razmak između navodnika i slova W u drugoj rečenici. To smo napravili tako da riječi ne budu spojene jer će se ispisati u isti red. Prisjetite se da ako želite u novi red ispisati nešto možete to napraviti sa ''<<endl'' Također u novi red možete ići ako napišete negdje u navodnicima **\n**, npr.

cout << recenica << "\n What's up?";

Idemo sada napraviti mali program koji će izračunavati dva broja.

#include<iostream>
using namespace std;

main()
{
int a, b, rezultat;
a = 7;
b = 12;
rezultat = a + b;
cout << "Zbroj ta dva broja je " << rezultat;

char x;
cin >> x;
return 0;
}

Kao što string označava niz znakova, tako int označava *integer* odnosno cijeli broj. Znači sve što će biti pridodano varijablama a, b i rezultat će biti cijeli broj (nikakvi decimalni brojevi, slova, drugi znakovi nego samo cijeli broj).

Primjetite da smo ovdje protiv nestajanja prozora koristili drugu varijablu da ne bi došlo do konflikta.

Deklarirati varijable možete ovako kao u primjeru a možete i svaku posebno na ovaj način:

int a;
int b;
int rezultat;

Nakon deklariranja varijabli slijedi naravno pridodavanje vrijednosti tim varijablama. Pridodajemo varijabli a broj 7, varijabli b broj 12 a varijabla rezultat će pohraniti onaj zbroj koji daju varijable a i b. U ovom slučaju 7 + 12 = 19. Znači varijabli "rezultat" se pridružuje vrijednost 19.

Varijable možete također deklarirati i pridružiti im vrijednost odjednom. Dakle ovako:

int a = 7;
int b = 12;
int rezultat = a + b;

Probajte sada napisati ovakav program

#include<iostream>
using namespace std;

main()
{
int a, b, rezultat;
cout << "Unesite prvi broj: ";
cin >> a;
cout << "Unesite drugi broj: ";
cin >> b;
rezultat = a + b;
cout << "Zbroj ta dva broja je " << rezultat;

char x
cin >> x;
return 0;
}

Dakle, ovo je skoro isto kao i prethodni primjer samo što ovdje imamo

cin >> a;

Program očekuje da ćete napisati neki broj i kada vi napišete taj broj on će ga spremiti u varijablu "a". Isto tako i dvije linije poslje za varijablu "b".

**Signed i Unsigned**
Ovo koristimo za varijable sa predznakom i bez predznaka.
Ako stavite

unsigned int a;

onda varijabli "a" nećete moći pridružiti broj sa predznakom (npr. -12) a ako umjesto unsigned napišete signed ili ostavite bez toga onda ćete moći upisati predznak.

**Doseg varijabli**

Varijable mogu biti globalne ili lokalne. Globalne varijable su one varijable koje su deklarirane u glavnom tijelu source koda, izvan svih funkcija dok su lokalne varijable one varijable koje su deklarirane u funkciji ili bloku.

Vjerujem da će vam biti jasnije ako vam to objasnim primjerom.

**Primjer lokalnih varijabla**

#include<iostream>
using namespace std;

main()
{
int a, b, rezultat;
a = 2;
b = 5;
rezultat = a + b;
cout << rezultat;

char x;
cin >> x;
return 0;
}

**Primjer globalnih varijabli**

#include<iostream>
using namespace std;

int a, b, rezultat;

main()
{ a = 2;
b = 5;
rezultat = a + b;
cout << rezultat;

char x;
cin >> x;
return 0;
}

Globalnim varijablama se može pristupiti bilo gdje u kodu, čak i u funkcijama dok je lokalnim varijablama moguće pristupiti samo unutar vitičastih zagrada { i }

NAPOMENA: sve varijable **moraju** početi sa slovom ili donjom crticom **\_**. Ne smiju početi brojevima niti ičim drugim. Osim toga, sve varijable u svom nazivu smiju imati samo slova, brojeve (od drugog mjesta na dalje), i crtice (**\_**), a ne smiju sadržavati razmake, navodnike i slične simbole.

Također bitno je to da vam se varijable ne smiju zvati sljedećim imenima:

asm, auto, bool, break, case, catch, char, class, const, const\_cast, continue, default, delete, do, double, dynamic\_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, operator, private, protected, public, register, reinterpret\_cast, return, short, signed, sizeof, static, static\_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar\_t, while, and, and\_eq, bitand, bitor, compl, not, not\_eq, or, or\_eq, xor, xor\_eq

Jer su to ključne riječi i operatori jezika C++.

Još jedna napomena: C++ je case sensitive jezik, odnosno osjetljiv na velika i mala slova. Tako da recenica i RecEnica nije ista varijabla.

Kao što smo već rekli, u sljedećem tutorijalu ćemo se pozabaviti tipovima podataka (to vam je ono int, char, string i ostali... )

**Tipovi podataka**

Ovaj tutorijal se nadovezuje na varijable, jer kako smo u prošlom tutorijalu napisali da int, string i char označavaju tip podakta koji će biti pridružen varijabli. Pa ćemo sada detaljnije objasniti te i ostele tipove podataka.

Kada programiramo, spremamo varijable u memoriju kompjutera, ali kompjuter mora znati što ćemo (kakav tip podatka) spremiti u varijablu. Jedan jednostavan broj neće zauzeti istu količinu memorije kao jedan veliki tekst ili veliki broj, i neće biti interpretirano na isti način.

Memorija kompjutera je organizirana po bajtovima (bytes). Jedan bajt je minimalna količina memorije koju možemo sačuvati u C++. Jedan bajt može pohraniti relativno mali broj podataka. Jedno slovo ili jedan cijeli broj od 0 do 255.

Osim toga, C++ može upravljati mnogo kompleksnijim tipovima podataka koje dolaze grupiranjem bajtova. Kao što su dugački brojevi (long numbers) i slično.

Evo jedne tablice sa osnovnim tj. bitnim tipovima podataka.



*Floating Point Numbers* su brojevi sa pomičnim zarezom.

**Konstante (nepromjenjive)**

Varijable su promjenjive dok su konstante su izrazi sa fiksnim, nepromjenjivim vrijednostima.

Morate inicirati konstantu kada ju napravite, i ne možete jej prirdružiti novu vrijednost kasnije. Poslje kada je konstanta inicirana njezina vrijednost je nepromjenjiva.

**Literal Constants (konkretne konstante)**

C++ ima dva tipa konstanti: konkretne i slimbolične (literal i symbolic)

Literal konstanta je vrijednost upisana direktno u program kada god je to potrebno. Npr.

int godine = 24;

godine je varijabla tipa integer, a 24 je literal konstanta. Ne možete pridružiti vrijednost na 24, i ta vrijednost ne može biti promijenjena.

**Symbolic constants (simbolične konstante)**
Simbolična konstanta je konstanta koja je pretstavljena po imenu, baš kao i varijabla. Ali nakon što je konstanta inicirana, njezina vrijednost ne može biti promjenjena. Ako imate jednu integer varijablu imenovanu "studenti" i drugu imenovanu "ucionica", možete procijeniti koliko studenata imate, i broj učionica, te ako znate da je 15 studenata po razredu.

sudenti = ucionice \* 15;

U ovom primjeru, broj 15 je Literal konstanta. Kod bi bio puno jednostavniji za čitanje i jednostavniji za održavati ako biste tu vrijednost zamijenili simboličnom konstantom.

studenti = ucionice \* ucenika\_po\_ucionici;

Ako kasnije odlučite promijeniti broj studenata po ucionice, mozete to uciniti tako da definirate konstantu ucenika\_po\_ucionici bez potrebe za izmjenama u svakom dijelu koda gdje ste koristili tu vrijednost.

**Definiranje konstanti sa #define**

Da biste definirali konstantu na sa staromodnim, i lošim načinom unest ćete

#define ucenika\_po\_razredu 15;

Primjetite da ucenika\_po\_ucionici nema konkretni tip (int, char itd.). Svaki puta kad preprocesor vidi riječ ucenika\_po\_razredu upisat će broj 15 u tekst.

Zbog toga što se preprocesor pokreće prije kompajlera, kompjuter nikad ne vidi vašu konstantu, ono vidi broj 15.

**Definiranje konstanti sa const**
Iako #define radi, postoji bolji, ukusniji način definiranja konstanti u C++

const unsigned int ucenika\_por\_ucionici = 15;

U ovom primjeru se isto deklarira simbolična konstanta imenovana ucenika\_po\_ucionici, ali ovaj puta ucenika\_po\_ucionici je napisano kao unsigned int (bez predznaka, cijeli broj)

S ovime imate više za tipkati ali nudi nekoliko prednosti.Najveća razlika je u tome što ova konstanta ima tip i kompajler može prisliti da to bude korišteno po svom tipu.

**Odnosni operatori**

U programskom jeziku C++ kao i u mnogim drugim programskim jezicima postoje odnosni operatori. Takvi operatori se npr. koriste ako želite napraviti program koji će iz baze podataka odrediti osobe sa više od 30 godina i slično.

Evo tih odnosnih operatora:



**Odnosne naredbe**

Kao i aritmetički operatori, odnosni operatori su binarni tj. uspoređuju dva operanda. Naredba sa dva operanda i odnosnim operatorom zove se odnosna naredba (eng. *relational expression*.)
Rezultat odnosne naredbe je Boolean vrijednost odnosno istinu ili laž (*true*ili *false*). S ovom tablicom možete vidjeti kako to funkcionira



U ovoj tablici se koriste konkretne (*literal*) vrijednosti koje ne mogu biti promijenjene. 4 je konkretna vrijednost (konstanta), i ona se ne može mijenjati (moglo bi se mijenjati da je umjesto konstanta koristimo varijable)

Isprobajmo sada ovaj kod koji umjesto konstanti koristi varijable

#include <iostream>
using namespace std;
int main()
{
int a = 4, b = 5;
cout << (a > b) << endl;
cout << (a >= b) << endl;
cout << (a == b) << endl;
cout << (a <= b) << endl;
cout << (a < b) << endl;

char x;
cin >> x;
return 0;
}

Ovaj program će ispisati:
0
0
0
1
1

S time da 0 predstalja laž (*false*) a 1 istinu (*true*).