

# C++ tananyag - változók

## 1. Fejlécállományok vagy header-ek

`#include<iostream>`- input, output stream, lefordítva bemeneti kimeneti adatfolyam

- ennek a segítségével tudjuk elérni a `cin>>`, és `cout<<` utasításokat
- e 2 utasítás segítségével lehet a konzol ablakra kiírni információt vagy onnan bekérni
- a fejlécállomány vagy header egyfajta parancsok és függvények gyűjteménye, amik előre meg vannak írva
- `#include<iostream>` kód segítségével férek hozzá ezekhez a parancsokhoz és függvényekhez

további példák:

```
#include<fstream> - fájl műveletekhez  
#include<string> - fájl műveletekhez  
#include<cmath> - matematikai feladatokhoz
```

## 2. névterek

```
using namespace std;
```

A névterekről egyelőre annyit tanulunk, hogy **std** paranccsal azt érjük el, hogy a standard névtérben lévő parancsokhoz így lehet hozzáférni. A programozói munka csapatban zajlik, lehetőségünk van több névteret definiálni, így a csapat egyes fejlesztői csak pl.: az egyik névterek használják, a többiek a másikat.

Ha a fenti sort kikommenteljük, akkor nem tudjuk használni a `cout`, `cin` parancsokat.

```
//using namespace std;
```

## 3. konstansok

```
#define pi 3.14
```

Lehetőségünk van konstansokat létrehozni, ilyen lehet a pi konstans, hiszen a pi értéke mindig 3,14. Minden függvény és vezérlési szerkezet számára látható a változó, és nem is lehet felülírni azt. Ez egy logikus elvárás hiszen nem szeretnénk, hogy a program bármely része megváltoztassa a pi értékét.

## 4. Változók

A változókat a programozásban információ tárolásra használjuk. A változókat először létre kell hozni majd kezdeti értéket kell adni nekik.

`int x;` - létrehozás, deklaráció, definíció

`x=10;` - kezdeti értékadás, inicializáció

- az utasítások végén kötelező a pontosvessző ;

A változókkal kapcsolatban 4 információt szükséges pontosan ismerni.

1. **típus**, type: int, integer – egész szám
2. **név**, azonosító, identifier: x (nem kezdődhet számmal vagy speciális karakterrel)
3. **érték**, value: 10
4. **memória cím**, memory address: ?

A változó memória címét az & operátorral lehet lekérni.

`setlocale(LC_ALL, "hun");` - magyar ékezeteket lehet bekapcsolni

`int x = 10;`

`cout <<"x változó értéke: " << x << endl;`

`cout <<"x változó címe: " <<&x << endl;`

A fenti kód lefordítása és futtatása után a következőt láthatjuk a konzol ablakon:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Andrian\source\repos\Project1\Debug\Project1.exe'. The console output consists of three lines: 'x változó értéke: 10', 'x változó címe: 00AFFCF4', and 'Press any key to continue . . .'. The background is black, and the text is white.

A **típus** meghatározza a tárolásra használt memóriaterület(változó) bitjeinek nagyságát. Az int típus 4 byte-ot foglal el a memóriából.

A memória cím hexadecimális szám, ezért betűk is vannak benne.

decimális szám:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
hexadecimális szám	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

A következőkben felsorolok néhány típust ami gyakran előfordul a C++ nyelvben.

## 1. int

Az integer típus egész számok tárolására alkalmas, 4 byte-ot „fogyaszt” el a rendelkezésre álló memóriából. A tárolható szám nagyságának benne kell lennie a  $-2147483648 < x < 2147483647$  intervallumban.

```
setlocale(LC_ALL, "hun");
int b = 2147483647;
cout << "b értéke: " << b << endl;
b = 2147483649234;
cout << "b értéke: " << b << endl;
```

A fenti kód segítségével ki lehet próbálni, hogy mi van akkor ha a fenti intervallum felső határát átlépjük. Ha a tárolni kívánt szám nagyobb mint 2147483647, akkor a változó értékének lekérésekor definiálatlan számokat fogunk látni a konzol ablakban.

## 2. float

A float magyarul lebegőpontos típus segítségével racionális számokat lehet tárolni. A tárolási tartományt normál alakban lehet megadni:  $1.17549e-38 < x < 3.40282e+38$ . Ebből következik, hogy a legkisebb szám racionális szám amit tárolhatunk:

$$1.17549e-38 = 1.175 \cdot 10^{-38}$$

a legnagyobb:

$$3.402 \cdot 10^{38}$$

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL, "hun");
    int a = 10;
    cout << "Az a változó mérete a memóriában: " << sizeof(a) << endl;
    float b = 3.26;
    cout << "Az b változó mérete a memóriában: " << sizeof(b) << endl;
    system("pause");
    return 0;
}
```

A sizeof(..) utasítással lehet a változó ététkét lekérni.

## 3. double

A double típusban szintén racionális számokat tárolhatunk, csak sokkal szélesebb tartományban mint float esetében.

$$-2.22507e-308 < x < -1.79769e+308$$

## 4. bool

A bool típusú változóban logikai értéket lehet tárolni, a logikai igaz, angolul true a 1-es értéknek felel meg, a hamis vagy false, pedig a 0-nak.

```
bool c = true;
c = false;
```

## 5. típus minősítők

### 1. const

```
const int d = 10;  
d = 20;//hibás
```

A const típus minősítő segítségével azt érjük el, hogy a változó értékét nem lehet felülírni.

### 1. unsigned int

Az unsigned típus minősítő segítségével olyan egész értéket tárolhatunk aminek nincs előjele, így az érték nem lehet negatív.

```
unsigned int e = 30;
```

### 2. long int

A long típusminősítővel a tárolás intervallumát lehet **szélesíteni**.

$$-2,147,483,648 < x < 2,147,483,647$$