matek, fizika programozás könnyedén

C++ programnyelv változói és típusai

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> - szöveges műveletekhez
#include<cmath> - matematikai feladatokhoz

2. Névterek

usingnamespace std;

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

Kipróbálható, hogy 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 **π** (pí) konstans, hiszen a **π** 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 **π** 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
x = 10; - kezdeti értékadás, inicializáció

- az utasítások végén kötelező a pontosvessző beírása ;

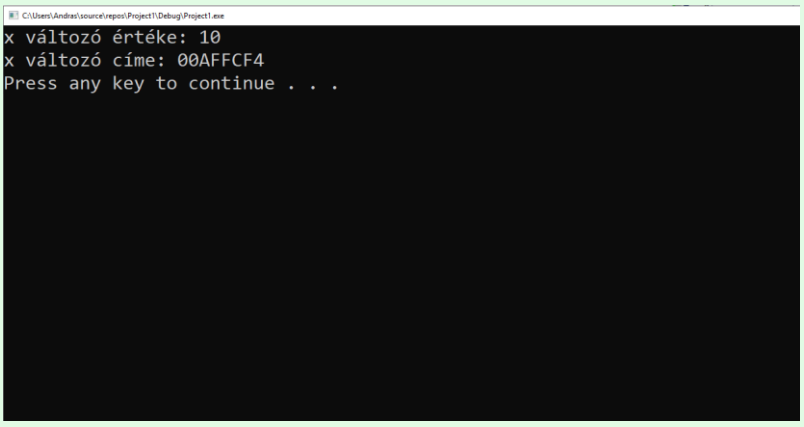
A változókkal kapcsolatban 4 információt szükséges pontosan ismerni.

1. **típus**, type: int, integer, egész szám
2. **név**, azonosító, identifier: x (nem kezdődhez 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 **típus** meghatározza a tárolásra használt memóriaterület (változó) bitjeinek mennyiségét vagy memória fogyasztását. 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.

5. 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.

6. 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 racionális szám amit tárolhatunk:

$$1.17549e-38 = 1{,}17 \cdot 10^{-38}$$

a legnagyobb:

$$1.17549e-38 = 3{,}4 \cdot 10^{38}$$

```
#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 által elfogyasztott memória nagyságát lekérdezni.

7. double

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

$$-2{,}22507\mathrm{e}-308 \ < \ \mathrm{x} \ < \ 1{,}79769\mathrm{e}+308$$

8. bool

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

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

9. típus minősítők

9.1 const

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

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

9.2 unsigned int

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

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

9.3 long int

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

$$-2{,}147{,}483{,}648 \ < \ \mathrm{x} \ < \ 2{,}147{,}483{,}647$$