

1. Gyakorlat

Rövid elméleti összefoglaló

- A C++ nyelv hatékony, általános célú programozási nyelv, amely hagyományos fejlesztőeszközként és objektum-orientált programozási nyelvként egyaránt használható. Erősen típusos nyelv. Újdonságai: kivételek (*exception*) kezelése, valamint a paraméterezett típusok, osztályok és függvények használata (*templates*, sablonok).
- **A C++ nyelv alapelemei:**
Az ANSI C++ szabvány a nyelv alapelemeit (nevek, számok, karakterek) *token*nek nevezi. A C++ nyelv betűérzékeny, megkülönbözteti a kis- és a nagybetűket.
- **A nyelv jelkészlete:** Az angol ábécé kis- és nagybetűi s egyéb jelek (! " # % & ' () * + , - / : ; < = > ? [\] ^ _ { | } ~) s a nem látható jelek (*whitespace*) (szóköz, vízszintes és függőleges tabulátor, új sor ('\n') stb.), valamint a UniCode karakterek, amelyek csak megjegyzésekben szerepelhetnek.
- **A C++ nyelv azonosítói:** A legtöbb fordító 32-karakteres neveket használ, az első karakter betű vagy _ (aláhúzásjel), melyet betű, szám, illetve aláhúzásjel követhet. A legtöbb fordító kibővíti a szabványos kulcsszavakat saját jelentéssel: `__try`, `__property` stb.
- **Az Ansi C++ nyelv kulcsszavait** nem lehet átdefiniálni, új jelentéssel ellátni.
Konstansok: C++ nyelvben: karakteres, logikai, egész, felsorolt és lebegőpontos konstansokat használhatunk:
 - A C++ nyelv *logikai konstansai* **true** és **false** lehetnek.
 - *Egyetlen mutatókonstanssal* rendelkezik ez a 0 (nulla), melyet a NULL szimbólummal jelölünk.
 - *Egész konstansok:* 12, 012, 0x20 (decimális, oktális, hexadecimális), 32U, 43L (U, u - **unsigned**, L, l - **long**)
 - *Felsorolt konstansok:* **enum** minoseg{jo, kozepes, rossz};
A minoseg típusú konstansok értékei: jo 0, kozepes 1, rossz 2
 - *Lebegőpontos konstansok:* 100.3, 2.567F, 1.78f (F, f - egyszeres pontosság), 3.456L (L, l **long double** nagy pontosság) 2e-3, 5.3e+10, 1E2, 1E-2 (E, e - exponens, a kitevő).
 - *Sztringkonstansok:* "ez egy szövegkonstans\nkét sorban"
 - *Karakterkonstansok:* 'a' egybájtos, L'2*' (két karaktert tartalmazó UniCode karakter- konstans)
- **Megjegyzések:** /* */ többsoros megjegyzés, // egysoros megjegyzés
- **A C++ nyelv alaptípusai**
 - *Adattípusok:* **bool**, **char**, **int**, **float**, **double**
 - *Típusmódosítók:* **short**, **long** – tárolási hosszat, **singed**, **unsigned** – előjel értelmezését szabályozzák.
 - *Típusminősítők:* **const** (csak olvasható), **volatile** (másik futó folyamat is megváltoztathatja).
- **Egyszerű változók definiálása:**
`<tárolási osztály> típus <típus...> változónév <= kezdőérték><,...>;`

Például: `unsigned int i; // előjel nélküli egész`
`long double x = 1.523674e12; // nagypontosságú valós`
Tárolási osztály: **auto**, **register**, **static**, **extern**;
 - A függvényeken belül deklarált változók alaphelyzetben automatikus (**auto**) változók.
 - A **register** változót, ha lehetséges a fordító a számítógép regiszterébe helyezi (ha nem tudja, akkor a változó a memóriában marad). A **register** változó a függvény a program futási sebességét növeli.
 - A **static** változót a fordító úgy helyezi el, hogy benne van a kezdőérték, tehát a kezdőértékadás nem igényel futási időt.
 - A függvényen kívül definiált változók alapértelmezés szerint globális (más modulból elérhető - **extern**) tárolási osztállyal rendelkeznek.
- **Operátorok és kifejezések:**
 - *Aritmetikai operátorok:* / , % , * , + , - (|osztás, maradék, szorzás|, |összeadás, kivonás|. Az operátorok felsorolása a műveletek precedenciája szerint történt, az azonos szintű precedenciákat a | vonal fogja össze.)
 - *Precedencia-szabály:* Műveletek precedenciája, zárójelezés.
 - *Értékadó utasítás :* jobb oldal és a bal oldal fogalma. Például: a = 12.5; a = a+4; a +=4; -=, *=, /=, %=,
 - *Csoportosítási szabály :* a = b = c = 0; (jobbról balra szabály)
- **A C++ program szerkezete**
A C++ nyelvű program egy vagy több forrásfájlban helyezkedik el, melyek kiterjesztése .CPP. A forrásprogramhoz az ún. deklarációs (include, header, fej-) állományok csatlakoznak, melyeket a #include előfordító utasítás segítségével építünk be a forrásállományba.

- A C++ szabvány minden könyvtári elemet a közös *std* névterületen definiál:

```
#include <iostream.h>    // helyett
használjuk a
#include <iostream>
using namespace std;
```
- A C++ programhoz egy *main()* függvény tartozik, amely kijelöli a program belépési pontját – azaz a program futása a *main()* függvény indításával kezdődik.
- Az adatok beolvasására a szabványos input (*cin*), a kiírására pedig a szabványos output (*cout*) adatfolyam-objektumot használjuk. A kiírás után a soremelés kétféleképpen is biztosítható:

```
cout << "\n";
```

 vagy

```
cout << endl;
```
- Létezik még egy szabványos hibastream:

```
cerr << "Hibas adat" << endl;
```
- A különféle formátum kialakításához szükséges az alábbi fejlánc beszerkesztése:

```
#include <iomanip>
```

Jelző	Jelentése
<code>ios::fixed</code>	fixpontos kiírás: <code>cout.setf(ios::fixed)</code>
<code>ios::scientific</code>	E hatványkitevővel, normál alakban Ha sem a <i>fixed</i> , sem a <i>scientific</i> nincs bekapcsolva, akkor az eredmény nagyságrendjétől függ, hogy fix- vagy lebegőpontos lesz a megjelenő érték,
<code>ios::showpoint</code>	A tizedespont és a tizedespont utáni vezető nullák is kiírásra kerülnek,
<code>ios::showpos</code>	A + előjel is kiíródik a pozitív egész szám előtt,
<code>ios::right</code>	jobbra tömörít,
<code>ios::left</code>	balra tömörít,
<code>ios::adjustfield</code>	A <i>right</i> , <i>left</i> váltásánál használjuk: <code>cout.setf(ios::left, ios::adjustfield)</code>
<code>cout.precision(n)</code>	<i>n</i> az értékes jegyek száma
<code>setw(n)</code>	<i>n</i> mezőszélességnek megfelelően jobbra tömörítve: <code>cout<<setw(4)<<x;</code> <code>cout<<setw(6)<<y;</code>
<code>cout << dec << x;</code>	decimális kiírás,
<code>cout << hex << x;</code>	hexadecimális kiírás,
<code>cout << oct << x;</code>	oktális kiírás,
<code>cout.fill('*')</code>	karakterrel töltjük fel az üres helyeket jobbra, illetve balra,
<code>cout.width(n)</code>	<i>n</i> a mező szélessége.

```
//MINTA1_01
#include <iostream>
using namespace std;
int main()
{
    //konstansok
    const bool log1 = true;
    const bool log2 = false;
    const int egesz = 7;
    const int oktalis = 012;
    const int hexa = 0x20;
    const double a1 = 100.3;
    const double a2= 2.56F;
    const double a3= 1.78f;
    const double a4= 2e-3;
    const double a5= 5e-6;
    const double a6= 3e+10;
    const float a7= 142.67;
    const long double
        pi = 3.1415926535897932385;
    const char
        *s = "ez egy szovegkonstans\n"
            "ket sorban";
    const char c = 'a';
    int i; // deklaráció
    unsigned int j;
    short int k;
    long int t;
    double sugar = 1;
    long double ter;
```

```
//utasítások
cout <<"egesz: " << egesz << endl;
cout << oct << "oktalis: " << oktalis << endl;
cout << hex << "hexadecimalis: " << hexa << endl;
cout << "a1: " << a1; cout << " a2: " << a2;
cout << " a3: " << a3 << endl;
cout << "a4: " << a4; cout << " a5: " << a5;
cout << " a6: " << a6 << endl;
cout << "a7: " << a7 << endl;
cout << "szoveg: " << s << endl;
cout << "karakter: " << c << endl;
ter = sugar * sugar * pi; cout <<"terulet: " << ter <<endl;
cout.precision(16);
cout <<"terulet: " << ter << endl;
cin.get();
return 0;
}
```

A program futásának eredménye:

```
egesz: 7
oktalis: 12
hexadecimalis: 20
a1: 100.3 a2: 2.56 a3: 1.78
a4: 0.002 a5: 5e-06 a6: 3e+10
a7: 142.67
szoveg: ez egy szovegkonstans
ket sorban
karakter: a
terulet: 3.14159
terulet: 3.141592653589793
```

```
//MINTA1_02
#include <iostream>
using namespace std;
int main()
{
    double a=2, b=4, c, d, e, f, g;
    c = a * b - 2;
    d = a * (b - 2);
    e = a * b / 2;
    f = (a + b)/2;
    g = a + b/2;
    cout << "c: " << c << " d: " << d << " e: " << e
        << " f: " << f << " g: " << g << endl;
    cin.get();
    return 0;
}
```

A program futásának eredménye:

c: 6 d: 4 e: 4 f: 3 g: 4

```
//MINTA1_03
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
    double a = 0.00005, b = 123.4142;
    cout << "a: " << a << " b: " << b << endl;
    //fixpontos kiíratás
    cout.setf(ios::fixed);
    cout << "a: " << a << " b: " << b << endl;
    cin.get();
    return 0;
}
```

Megjegyzés: Ha sem a *fixed*, sem a *scientific* nincs bekapcsolva, akkor a kiírás eredménye a nagyságrendtől függően fix- vagy lebegőpontos alakú lesz (lásd az első futtatási eredményt).

A program futásának eredménye:

a: 5e-05 b: 123.414

a: 0.000050 b: 123.414200

```
//MINTA1_04
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
    double a = 0.00005, b = 123.4142;
    //hatványkitevős kiíratás
    cout.setf(ios::scientific);
    cout << "a: " << a << " b: " << b << endl;
    cin.get();
    return 0;
}
```

A program futásának eredménye:

a: 5.000000e-05 b: 1.234142e+02

```
//MINTA1_05
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
    double a = 1.5, b = 123.4142;
    // tizedespont utáni nullák kiírása
    cout.setf(ios::showpoint);
    // kiírás 5 értékes jegyre
    cout.precision(5);
    cout << a << endl;
    cout << b << endl;
    cin.get();
    return 0;
}
```

A program futásának eredménye:

1.5000

123.41

1. Mintafeladat

A mintafeladat két egész változó beolvasását, szorzatának és hányadosának számítását kétféle módon való képzését mutatja be. (Két egész szám hányadosa egész eredményt ad!)

<pre>//MINTA1_06 #include <iostream> using namespace std; int main() { int a, b, szorzat; double hanyad1, hanyad2; cout << "a: "; cin >> a; cout << "b: "; cin >> b; szorzat = a * b; hanyad1 = a/b; cout << "szorzat: " << szorzat << endl; cout << "a/b : " << hanyad1 << endl; hanyad2 = (double)a/b; cout << "(double)a/b: " << hanyad2; cin.get(); cin.get(); return 0; }</pre>	<p>A program futásának eredménye:</p> <pre>a: 2 b: 5 szorzat: 10 a/b : 0 (double)a/b: 0.4</pre>
---	--

2. Mintafeladat

Olvassunk be két valós számot (a, b)! Számítsuk ki az a változó b -edik hatványát, valamint az a változó gyökét!

A *cmath* könyvtár beszerkesztése szükséges a *pow()*, és az *sqrt()* függvények használata miatt!

<pre>//MINTA1_07 #include <iostream> #include <cmath> using namespace std; int main() { double a, b, hatvany, gyok; cout << "a: "; cin >> a; cout << "b: "; cin >> b; hatvany = pow(a, b); gyok = sqrt(a); cout.precision(4); cout << "hatvanya: " << hatvany << endl; cout << "negyzet gyoke: " << gyok << endl; cin.get(); cin.get(); return 0; }</pre> <p>A program futásának eredménye:</p> <pre>a: 4 b: 3 hatvanya: 64 negyzet gyoke: 2</pre>	<p>Ha matematikai függvényeket használunk, akkor az alábbi fejléc-állomány szükséges:</p> <pre>#include <cmath></pre> <p>A <i>cin</i>>> sorban adatot olvasunk egy változóba. (Nem kell konverzió, sem a változó címe.) <i>cout</i><< írunk a képernyőre. Kiírható:</p> <ul style="list-style-type: none"> • "" között a szöveges információ, • változó, • kifejezés, • függvényhívás, • endl soremelés. <p>A <i>cin.get()</i>; karakter leütésére vár.</p> <p>Ha van olvasás a függvényben, akkor azért kell két <i>cin.get()</i>; utasítás, mert az első az <Enter> karaktert nyeli le.</p>
--	---

Feladatok

- Olvassunk be két egész számot! Számítsuk ki a számok átlagát kétféle módon, valamint a mértani közepét! Alkalmazzuk az alábbi definíciókat (GYAK1_1):

```
int a, b;
double atlag1, atlag2, mkozep;
```

Az átlag számítását kétféle módon hajtsuk végre, mivel két egész szám hányadosa egész eredményt adna:

- A számláló típusát alakítsuk át valóssá (**double**):
`atlag1 = (double)(a+b)/2;`
- Mivel a nevező konstans, ezért oszthatunk a 2. valós számmal, így az eredmény valós lesz:
`atlag2 = (a+b)/2.;`

A program futásának eredménye:

```
a: 1
b: 2
1. atlag: 1.5
2. atlag: 1.5
Mertani kozep: 1.41421
```

2. Olvassuk be egy téglalap két oldalának hosszát, és számítsuk ki a téglalap területét és kerületét! (GYAK1_2)

```
double a,b,ter,ker;
```

A program futásának eredménye:

```
A teglalap a oldala: 1
A teglalap b oldala: 2

A teglalap terulete: 2
A teglalap kerulete: 6
```

3. Olvassuk be a sugár adatát! Írjunk programot, amely kiszámítja a kör kerületét, területét, a gömb felszínét és térfogatát! (GYAK1_3)

```
const double pi = 3.141592654;
double r, korker, korter, gombfelsz, gombterf;
```

$$\begin{aligned} \text{körkerület} &= 2 \cdot r \cdot \pi & \text{körterület} &= r^2 \cdot \pi \\ \text{gömbfelszín} &= 4 \cdot r^2 \cdot \pi & \text{gömbtérfogat} &= \frac{4r^3 \cdot \pi}{3} \end{aligned}$$

A program futásának eredménye:

```
Sugar: 1
Kor kerulete : 6.28319
Kor terulete : 3.14159
Gomb felszine : 12.5664
Gomb terfogata: 4.18879
```

4. Olvassuk be a kocka élének hosszát Írjunk programot, amely kiszámítja a kocka felszínét és térfogatát! (GYAK1_4)

```
double kocka_el, kocka_felsz, kocka_terf;
```

A program futásának eredménye:

```
Kocka ele: 1
Kocka felszine : 6
Kocka terfogata: 1
```

5. Olvassuk be egy derékszögű háromszög két befogóját! Számítsuk ki az átfogót, a háromszög területét és a kerületét! (GYAK1_5)

A program futásának eredménye:

```
Befogo1: 3
Befogo2: 4

Atfogo: 5
Derekszogu haromszog kerulete: 12
Derekszogu haromszog terulete: 6
```