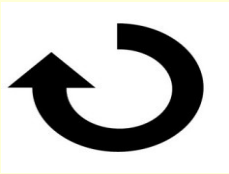
Bevezetés a programozásba

7. Előadás

C++ függvény

C++ „Hello world!”



```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

C++ „Hello world!”

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

C++ példa program: osztóösszeg

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

int main() {
    int a, b;
    cout << "Add meg a két számot: ";
    cin >> a >> b;

    int ooa=0;
    for( int i=1; i<=a; i++ ) {
        if( a%i==0 ) ooa+=i;
    }
    int oob=0;
    for( int j=1; j<=b; j++ ) {
        if( b%j==0 ) oob+=j;
    }
    cout << "Az osztóösszegek: " << ooa <<", " << oob <<endl;
    return 0;
}
```

Ismétlődő kódrészlet!

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

int main() {
    int a, b;
    cout << "Add meg a két számot: ";
    cin >> a >> b;

    int ooa=0;
    for( int i=1; i<=a; i++ ) {
        if( a%i==0 ) ooa+=i;
    }
    int oob=0;
    for( int j=1; j<=b; j++ ) {
        if( b%j==0 ) oob+=j;
    }
    cout << "Az osztóösszegek: " << ooa <<", " << oob <<endl;
    return 0;
}
```

Alprogram bevezetése

```
alprogram{
    int ooX=0;
    for( int i=1; i<=X; i++ ) {
        if( X%i==0 ) ooX+=i;
    }

}

int main() {
    int a, b;
    cout << "Add meg a két számot: ";
    cin >> a >> b;

     alprogram X <- a
     alprogram X <- b

    cout << "Az osztóösszegek: " << ooa <<", " << oob <<endl;
    return 0;
}
```

Visszatérési érték bevezetése

```
int alprogram{
    int eredmeny = 0;
    for( int i=1; i<=X; i++ ) {
        if( X%i==0 ) eredmeny += i;
    }
     return eredmeny;
}

int main() {
    int a, b;
    cout << "Add meg a két számot: ";
    cin >> a >> b;

     int ooa = alprogram X <- a
     int oob = alprogram X <- b

    cout << "Az osztóösszegek: " << ooa <<", " << oob <<endl;
    return 0;
}
```

Paraméter átadás bevezetése

```
int alprogram( int X ) {
    int eredmeny = 0;
    for( int i=1; i<=X; i++ ) {
        if( X%i==0 ) eredmeny += i;
    }
     return eredmeny;
}

int main() {
    int a, b;
    cout << "Add meg a két számot: ";
    cin >> a >> b;

     int ooa = alprogram( a );
     int oob = alprogram( b );

    cout << "Az osztóösszegek: " << ooa <<", " << oob <<endl;
    return 0;
}
```

Alprogramok kommunikációja

- Az alprogramok egymásnak átadhatnak értékeket a következő eszközökkel:
  - **globális változók/konstansok:** olyan változók, amelyek a teljes programkódban érvényben vannak, nem csak egy adott programrészen belül
  - **visszatérési érték:** az alprogram által visszaszolgáltatott érték, amely visszakerül a hívás helyére, egy alprogramnak csak egy visszatérési értéke lehet
  - **paraméterek:** olyan változók, amelyek az alprogram meghívásakor kapnak értéket, és a teljes alprogramban érvényesek, paraméterből bármennyit adhatunk egy alprogramnak, de a meghíváskor mindegyiknek értéket kell adnunk

Visszatérési értékek

- Minden függvényben, amely nem visszatérési érték nélküli, kell szerepelnie egy érték visszaadásnak
  - erre a célra a ***return*** utasítás szolgál, amelyet bárhol elhelyezhetünk a függvény kódjában, de gondoskodnunk kell arról, hogy a függvény minden lefutása hozzávezessen
  - a ***return*** utáni utasítások nem kerülnek végrehajtásra, ezért általában a függvény utolsó sorában szerepel (de elágazások használatával előbbre is tehető, illetve több is elhelyezhető egy függvényben)
  - a ***return*** után szerepelhet érték, változó, kifejezés, de a típusának meg kell egyeznie a függvény deklarációjában megadott típussal
  - a programkód lefutásakor mindig pontosan egy ***return*** hajtódik végre, tehát mindig egy érték adódik vissza

Függvények írásmódja

- Ha egy C++ kódban a következőt látjuk:

```
típusnév függvénynév(tetszőleges deklarációk)
{
    …

    return valami;
}
```

- akkor az egy függvény

Függvények írásmódja

- Ha egy C++ kódban a következőt látjuk:

```
típusnév függvénynév(tetszőleges deklarációk)
{
    …

    return valami;
}
```

- akkor az egy függvény

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!”;
    return 0;
}
```

Függvények szerkezete C++ -ban

```
double terulet(double a, double b, double c) { … }
```

- A függvény szignatúrája
  - típusa: ami a visszatérési érték típusa *(típusnév)*
  - név: a függvény neve *(függvénynév)*
  - paraméter lista: változók deklarációját jelenti (ugyanúgy, mint bárhol máshol a kódban), melyek a függvényhez tartoznak *(tetszőleges deklarációk)*
- függvény törzse: a végrehajtandó utasítássorozat

A **void** típus

- Ez a „semmilyen” típus
- Ilyen típusú változót nem lehet készíteni
- Használata: Ezzel jelezzük, ha egy függvénynek nincs visszatérési értéke
  - tehát a visszatérési típus „semmilyen”

„Eljárások” C++ -ban

- A visszatérési érték nélküli függvények típusa ***void***, és ekkor a ***return*** után nem szerepel érték
  - Pl: egy skip nevű függvény, amely nem csinál semmit, és nem ad vissza értéket:

```
void skip(){
return;
}
```

  - ezekben a függvényekben nem is kötelező kiírni a ***return*** utasítást, ezért csak akkor írjuk ki őket, ha a függvény működését előbb meg akarjuk szakítani, minthogy a teljes függvénytörzs kódja lefusson
  - tehát az előző függvény így is írható:

```
void skip(){}
```

Függvények a C++-ban

- A függvény belsejében egy program van
- Minden C/C++ programban van egy fő függvény, ennek a neve „main”
- Amikor a teljes program elindul, akkor a main függvény indul el, és ha ez a függvény befejeződik, akkor a teljes program is befejeződik
- A lényeg: Függvények más függvényeket hívnak meg.
- Ez lényegében egy új programkonstrukció

Függvényhívás

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;
    double t = terulet(ha, hb, hc);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Függvényhívás

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;
    double t = terulet(ha, hb, hc);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

A főprogram végrehajtása megáll, a vezérlést átadja a meghívott függvénynek. A függvény törzs végrehajtásával folytatódik a program.

Paraméterátadás

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;
    double t = terulet(ha, hb, hc);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Paraméterátadás

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;
    double t = terulet(ha, hb, hc);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

A függvény törzsben az átvett paraméterekkel dolgozik a program.

Visszatérésiérték

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;
    double t = terulet(ha, hb, hc);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

A függvény visszaadja a megadott típusú értéket a hívási pontba.

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(ha, hb, hc);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

A változók értéket kapnak.

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = (a+b+c)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Az aktuális paraméterekkel meghívjuk a függvényt.

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = (3+4+5)/2.0;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

A formális paraméterek átveszik az aktuális paraméterek értékét.

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = 6;
    return sqrt((s-a)*(s-b)*(s-c)*s);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = 6;
    return sqrt((6-3)*(6-4)*(6-5)*6);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = 6;
    return sqrt(3*2*1*6);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = 6;
    return sqrt(36);
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = 6;
    return 6;
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = terulet(3, 4, 5);
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Függvényhívás példa

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

double terulet(double a, double b, double c)
{
    double s = 6;
    return 6;
}

int main() {
    double ha, hb, hc;
    cin >> ha >> hb >> hc;          // ha=3, hb=4, hc=5
    double t = 6;
    cout << "Terulet: " << t <<endl;
    return 0;
}
```

Egyebek

- Mi a teendő, ha nem egy változót szeretnék visszaadni, hanem többet?
  - Türelmesen várj a következő előadásig
- Lehet-e függvényhíváskor a paraméter egy másik függvény visszatérési értéke?
  - Természetesen, a visszatérési értékkel rendelkező függvény tetszőleges kifejezés része lehet
- Mi történik, ha kifejezésben/értékadásban void típusú függvényt hívok?
  - Nem használható kifejezésben, vagy értékadásban a függvényhívás, mert nem ad vissza értéket („semmilyen a típusa”)

Folytatás következik

- C++ függvények második rész a következő előadáson