

# OO Programozás

## Operátor felültöltés, másolás és átalakítás

Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

[https://github.com/wajzy/GKxB\\_INTM085](https://github.com/wajzy/GKxB_INTM085)

2023. október 27.

## Sablonok (template)

- Különböző típusokat kezelő függvények / osztályok csoportjának egyszerű létrehozása
- *Típusparaméter* (fordítási időben eldől) vs. függvény paraméter (futási időben adják át)
- Kulcsszó: `class` vagy `typename` (*majdnem* egyenértékűek)
- *Sablon példányosítás*: adott típusparaméterrel új változat előállítása → egyszerűbb, mint sok felültöltött függvényt kézzel létrehozni

## Feladat:

Készítsünk `max` függvényt, ami két paramétere közül visszaadja a nagyobbát!

## max1.cpp

```
3  // Overloaded max functions
4  int max(int a, int b) {
5      return a > b ? a : b;
6  }
7
8  double max(double a, double b) {
9      return a > b ? a : b;
10 }
```

## max1.cpp

```
12 int main() {  
13     const int x = 1;  
14     const int y = 2;  
15     std::cout << x << " es " << y << " kozul "  
16             << max(x, y) << " a nagyobb.\n";  
17  
18     const double i = 1.5;  
19     const double j = 2.5;  
20     std::cout << i << " es " << j << " kozul "  
21             << max(i, j) << " a nagyobb.\n";  
22 }
```

## Kimenet

```
1 es 2 kozul 2 a nagyobb.  
1.5 es 2.5 kozul 2.5 a nagyobb.
```

## max2.cpp

```

3  template<class T> T max(T a, T b) {
4      return a > b ? a : b;
5  }
6
7  int main() {
8      const int x = 1;
9      const int y = 2;
10     std::cout << x << " és " << y << " közül "
11               << max(x, y) << " a nagyobb.\n";
12
13     const double i = 1.5;
14     const double j = 2.5;
15     std::cout << i << " és " << j << " közül "
16               << max(i, j) << " a nagyobb.\n";

```

- Egy sablon több típusparamétert is fogadhat, de akkor ezeket mind használni is kell!
- Csak akkor állít elő kódot a fordító, amikor a függvény konkrét hívásával találkozik → a példányosításig a hibás sablon nem vált ki hibaüzenetet
- Pontosan olyan változatok jönnek létre, amikre szükség is van
- Típusonként pontosan egy változatot hoz létre, akkor is, ha létezik a típusok között implicit konverzió (pl. `int` → `long`)
- Mi történik, ha két *eltérő* típusú paraméterrel hívják a függvényünket?

## max2.cpp

```

18 //const int k = 3;
19 //const double l = 4.5;
20 //std::cout << k << " es " << l << " kozul "
21 //      << max(k, l) << " a nagyobb.\n";
22 // error: no matching function for call to
23 // 'max(const int&, const double&)'
24 // note: candidate: template<class T> T max(T, T)
25 // note: template argument deduction/substitution failed:
26 // note: deduced conflicting types for parameter 'T' ('int' and 'double')
27 }

```

### Probléma:

A két paraméter típusának mindenképpen egyeznie kell!

### Félmegoldás:

Két típusparaméter használata (nem befolyásolható futásidőben, melyikben keletkezzen az eredmény)



## max3.cpp

```

3  template<class T1, class T2> T1 max(T1 a, T2 b) {
4      return a > b ? a : b;
5  }
6
7  int main() {
8      const int i = 3;
9      const double d = 4.5;
10     std::cout << i << " es " << d << " kozul "
11               << max(i, d) << " a nagyobb.\n";
12     std::cout << d << " es " << i << " kozul "
13               << max(d, i) << " a nagyobb.\n";
14 }

```

# Kimenet

3 es 4.5 kozul 4 a nagyobb.  
4.5 es 3 kozul 4.5 a nagyobb.

Ha létezik függvény a szükséges paraméterekkel, nem jön létre példány a sablonból, és a korábbi hiba orvosolható.

### max4.cpp

```
3  template<class T1, class T2> T1 max(T1 a, T2 b) {  
4      return a > b ? a : b;  
5  }  
6  
7  double max(int a, double b) {  
8      return a > b ? a : b;  
9  }
```

### Kimenet

```
3 es 4.5 kozul 4.5 a nagyobb.  
4.5 es 3 kozul 4.5 a nagyobb.
```