2. fejezet

Típus paraméterek
Magasabb rendű függvények

Haladó Alkalmazott Programozás
ELTE, 2019

Típusparaméterek

A múltkor volt a négyzetre emelés függvény:

```
double sq(double x){ return x*x; }
```

Típusparaméterek

A múltkor volt a négyzetre emelés függvény,
de ez több típusra is kéne:

```
double sq(double x){ return x*x; }
float  sq(float  x){ return x*x; }
int    sq(int    x){ return x*x; }
```

Típusparaméterek

A múltkor volt a négyzetre emelés függvény,
de ez több típusra is kéne:

```
double sq(double x){ return x*x; }
float  sq(float  x){ return x*x; }
int    sq(int    x){ return x*x; }
```

```
std::complex<double> sq(std::complex<double> x){ return x*x; }
std::complex<float>  sq(std::complex<float>  x){ return x*x; }
```

Ez nyilván nem skálázik, kéne jobb megoldás...

Típusparaméterek

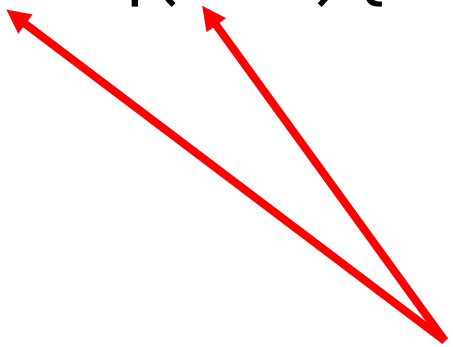
Legyen a típus paraméter!

```
template<typename T>
T sq(T x){ return x*x; }
```

Típusparaméterek

Legyen a típus paraméter!

```
template<typename T>
T sq(T x){ return x*x; }
```

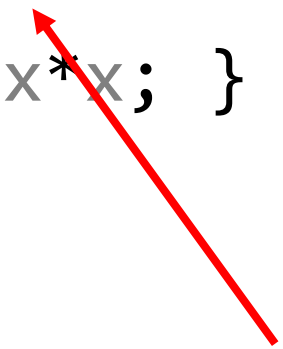


Ahol eddig konkrét típus szerepelt (pl.: `double`),
oda most egy szimbólumot (`T`) írunk...

Típusparaméterek

Legyen a típus paraméter!

```
template<typename T>
T sq(T x){ return x*x; }
```



A függvény előtt pedig meg mondjuk, hogy `T` micsoda:
`typename T` azt jelenti, hogy `T` egy típust fog megnevezni,
A `template` kulcsszó pedig azt jelenti, hogy az utána definiált nyelvi elem (`sq`) a `< >` közötti paraméterektől függő sablon, tehát ez most egy függvény sablon (function template).

Típusparaméterek

Legyen a típus paraméter!

```
template<typename T>
T sq(T x){ return x*x; }
```

Amikor generikus lambdát használunk, az `auto` argumentumok mögött szintén template-k dolgoznak:

```
auto sq = [](auto x){ return x*x; };
```

Típusparaméterek

A sablonból úgy kapunk konkrét nyelvi elemet, hogy megadjuk a paramétereit:

```
template<typename T> T sq(T x){ return x*x; }

int main()
{
  double b = sq<double>(2.5); //6.25
  int    a = sq<int>(4); //16
  return a;
}
```

Típusparaméterek

A sablonból úgy kapunk konkrét nyelvi elemet, hogy megadjuk a paramétereit:

```
template<typename T> T sq(T x){ return x*x; }

int main()
{
  double b = sq<double>(2.5); //6.25
  int    a = sq<int>(4); //16
  return a;
}
```

A kód úgy viselkedik, mintha megírtuk volna külön az `int` és a `double` esetet.

Típusparaméterek

Azonban függvényeknél, ha minden template paraméter kitalálható az argumentumokból, akkor nem kell kiírnunk expliciten a típus paramétereket!

```
template<typename T> T sq(T x){ return x*x; }


int main()
{
  double b = sq(2.5); //6.25
  int    a = sq(4); //16
  return a;
}
```

Típusparaméterek

Azonban függvényeknél, ha minden template paraméter kitalálható az argumentumokból, akkor nem kell kiírnunk expliciten a típus paramétereket!

```
template<typename T> T sq(T x){ return x*x; }


int main()
{
  double b = sq(2.5); //6.25
  int    a = sq(4); //16
  return a;
}
```

Függvényparaméterek

Függvényparaméterek

Sokszor lesz szükségünk arra, hogy függvényt adjunk át függvénynek.

Például a Newton iterátor, vagy a numerikus integrátor úgy elegáns, ha bármilyen függvényt odaadhatunk neki paraméterként!

Függvényparaméterek

C++-ban az egyik mód függvények átadására a függvény sablon

```
template<typename F, typename T>
T integrate(F f, int n, T x0, T x1);
```

Nincs különbség `F` és `T` között: a `typename` bármilyen típust helyettesíthet, akár értéket, akár függvényt, akár objektumot.

Függvényparaméterek

Függvény átadása függvény sablonnal:

```
template<typename F, typename T>
T integrate(F f, int n, T x0, T x1)
{
  double dx = (x1-x0)/n, sum = 0.0;
  for(int i=1; i<=n-1; i++)
  {
    sum += f(x0+i*dx);
  }
  return dx/2.0*(f(x0) + 2.0*sum + f(x1));
}
```

Függvényparaméterek

Függvény átadása függvény sablonnal:

Az átadott függvény típusa bármi lehet (mert paraméter)

```
template<typename F, typename T>
T integrate(F f, int n, T x0, T x1)
{
  double dx = (x1-x0)/n, sum = 0.0;
  for(int i=1; i<=n-1; i++)
  {
    sum += f(x0+i*dx);
  }
  return dx/2.0*(f(x0) + 2.0*sum + f(x1));
}
```

Ugyan úgy használjuk, mintha `f` bárhol másutt lett volna definiálva normál függvényként

Függvényparaméterek

Használat:

```
integrate([](double x){ return x*x; }, 100, 0.0, 1.0);

//Ha sq már be volt vezetve (nem template-s) függvényként:
integrate(sq, 100, 0.0, 1.0);

//Ha a template-s változatot akarjuk megadni,
akkor ki kell írni a típus paramétert:
integrate(sq<double>, 100, 0.0, 1.0);
```

Függvényparaméterek

Feladat / Házi feladat:

Írjuk meg a Newton-iterátort most általánosan úgy, hogy a függvény és a deriváltja is az előbb bemutatott módon legyen átadva és a kezdő paraméter típusa is általános `T` legyen.

Ha jól csináltuk, akkor a következő hívásnak működnie kell:

```
Newton([](double x){ return x*x - 612.0;   },
       [](double x){ return 2.0*x; }, 10.0);
//~24.738633753705963298928
```

Figyelem: két függvényhez két külön típus paraméter kell!

Függvényparaméterek

Szorgalmi feladat:

Írjuk meg úgy a Newton iterátort, hogy a megállási feltétel is egy függvény paraméter legyen!

Ez a függvény az előző és az aktuális `x`-et kapja meg és `bool`-t ad vissza. Ha `true`, folytatódik az iteráció, ha `false`, akkor vége, és az aktuális `x` érték az eredmény.