šablony (template)

- dost často napíšeme kód a následně zjistíme, že bychom ho potřebovali několikrát, přičemž jediné čím se liší, je typ proměnné, se kterou pracuje (například funkce max, lineární seznam…). Toto může řešit princip šablon, kdy se napíše kód pro obecný typ a překladač si potom vygeneruje podle něj kód pro typ, který potřebuje.
- umožňují psát kód pro obecný typ
- vytvoří se tak návod (předpis, šablona) na základě které se konkrétní kód vytvoří až v případě potřeby pro typ, se kterým se má použít
- umisťuje se do hlavičkového souboru (předpis, netvoří kód). Do samostatného souboru lze za použití klíčového slova export template …

```
template <typename T> T max ( T &h1, T &h2 )
{
return (h1>h2) ? h1 : h2 ;
}

double d,e,f;
int i;
d = max(e,f);
d = max(e,i);//nelze, parametry jsou různého typu
d = max<float>(e,i);// typ T stanoven explicitně
```

```
template <typename T>
T max ( T &h1, T &h2 )
{
return (h1>h2) ? h1 : h2 ;
}

double a, b, c;
c = max(b,a);
```

- template – klíčové slovo říkající, že se jedná o předpis
- použitelné pro každý typ, který je “schopen” prováděných operací (v příkladu výše typ, který má definován operátor > a kopykonstruktor pro vytvoření návratové hodnoty)
- Zápis <typename T> (původní zápis byl <class T>) určuje název zástupného typu = T. Tento typ je použit při psaní šablony. Při realizaci bude nahrazen reálným typem.
- konkrétní typ T se zjistí při použití. V příkladu výše double, proto je vytvořeno max, kde na místě T se objeví double
- díky přetížení je možné na základě template vytvoření funkce (či třídy) pro různé typy
- vytvoření je možné i ”silou”.Např. deklarací int max(int, int); nebo int max<int>(int,int)
- lze i template pro třídu
- lze uvést i více obecných typů,

```
template < typename T, typename S >
double max ( T h1, S h2 )
```

template <typename T, int nn=10> …

- výrazový parametr nn, pokud použijeme nn, pak se nahradí zadaným číslem (nebude-li zadání, potom hodnotou 10) <int, 22>

```
template < class T >  // starý zápis s class místo typename
struct A {
T a , b ;
T fce ( double, T, int )
}

T A<class T>:: fce (double a, T b,int c) {}
```

potom
A <double>c, d;
budou c,d typu A<double>, proměnné a,b ve struktuře jsou double
A <int> g, h;
budou g,h A<int>, kde proměnné a,b ve struktuře jsou int

- specifikace jména typu získaného z parametru šablony

```
template<typename T>
class X {
typedef typename T::InnerType Ti;
// synonymum pro typ uvnitř T
int m(Ti o) { ..... }
}
```

- šablony lze i přetěžovat – vytvořit specializaci pro daný typ (pro ostatní typy se volá šablona původní)

```
template <> TSpec <int>
```