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 kod pro obecný typ a překladač si potom vygeneruje podle něj kod 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 – klíčové slovo říkající, že se jedná o předpis
- použitelné pro každý typ, který je “schopen” prováděných operací (v příkladu typ, který má definován operátor > a kopykonstruktor)
- <class T> starý zápis, nově <typename T>, určuje název typu, pro který se bude vytvářet. T v dalším zastupuje typ
- konkrétní typ T se zjistí při použití, zde 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);
- lze i více obecných typů, lze i template pro třídu

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

template <class 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 >
class 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;
bude c,d třídy A, kde a,b jsou double
A <int> g, h;
bude g,h třídy A, kde a,b jsou int
```

- specifikace jména typu získaného z parametru šablony

```
template<class 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>
```