Laboratorium 11 Pomoc. Szablony klas. Biblioteka STL

Cel laboratorium:
Zapoznanie z ideą tworzenia szablonów funkcji i szablonów klas i wykorzystania biblioteki STL

- Szablony funkcji: mechanizm tworzenia rodziny identycznych w działaniu funkcji różniących się typem argumentów

```
template <class typ>
void funkcja (typ arg1, typ arg2) {//……}
```

lub

```
template <typename typ>
void funkcja (typ arg1, typ arg2) {//……}
```

np.

```
template<class typ>
typ maximum(typ a, typ b)
{return (a>b)?a:b;}
```

Szablony klas: mechanizm automatycznego tworzenia klas

```
template <class typ>        //typ - parametrem szablonu
class klasa
{private:
   typ pole ;
 public:
  klasa(typ p): pole(p){}
   void setPole(typ x) { pole = x ; }
   typ getPole() { return pole ; }
};
```

np.

```
template <class typ>
class schowek
{private:
   typ wartosc ;
 public:
   void schowaj(typ x) { wartosc = x ; }
   typ oddaj() { return wartosc ; }
};
```

Wykorzystanie szablonu do tworzenia klasy:

```
nazwa_klasy<parametr_aktualny>
```

np.

```
schowek<int> sejf1; sejf1.schowaj(1000);cout<<sejf1.oddaj();
```

Specjalizacja szablonu - definicja szablonu dla konkretnego typu:

```
template <>
class Klasa<konkretnyTyp> {....};
```

Biblioteka STL (*Standard Template Library*) – standardowa biblioteka szablonów zawierająca:

- Kontenery (pojemniki):
  - sekwencyjne (`deque, list, queue, priority_queue, stack, vector`): elementy uporządkowane liniowo
  - asocjacyjne (`map, multimap, set, multiset`): łączy wartość z kluczem i używa klucza do wyszukiwania tej wartości
- Iteratory - wskaźniki poruszające się po kontenerach
- Funktory - obiekt używany jak funkcja z ()
- Algorytmy - zestawy funkcji (nie metod) używanych dla wszystkich klas kontenerowych

- Przykłady kontenerów (obiekty klas szablonowych):

```
#include <vector>
#include <list>
#include <queue>
#include <stack>

void show(int x){cout<<x<<endl;

int main(){
vector<double> wektor(5);
list<double> lista(10);
queue<string> kolejka;
stack<char> stos;

int n;
   cout<<"Podaj rozmiar wektora ";
   cin>>n;
   vector<int> wek2(n);//
for(int i=0;i<n;i++)
   {wek2[i]=i*2;
   cout<<wek2[i]<<endl;

vector<int>::iterator it;
for(it=wek2.begin();it!=wek2.end();it++)
   {cout<<*it<<",";}

wek2.push_back(13);
cout<<wek2.size();

for_each(wek1.begin(), wek1.end(), show);

return 0;
}
```