```
// Iterator

#include<iostream>
#include<string>
using namespace std;

void main()
{
	int k;
	string lancuch="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
	string::iterator iter=lancuch.begin();
	cout<<"Poczatek lancucha to : "<<*iter;
	cout<<"\nJedziemy w prawo o ";
	cin>>k;
	iter+=k;
	cout<<"pozycji i jestesmy w "<<*iter<<endl;
	iter=lancuch.end()-1;
	cout<<"Koniec lancucha to : "<<*iter;
	cout<<"\nJedziemy w lewo o ";
	cin>>k;
	iter-=k;
	cout<<"pozycji i jestesmy w "<<*iter<<endl;
	system("pause");
}
```

```
Poczatek lancucha to : A
Jedziemy w prawo o 10
pozycji i jestesmy w K
Koniec lancucha to : Z
Jedziemy w lewo o 12
pozycji i jestesmy w N
Press any key to continue . . .
```

```
// Użycie std::find

#include<string>
#include<iostream>
using namespace std;

void main()
{
	string lancuch ("abcd efgh ijkl mnop rst abcdefgh ");
	cout << "Przykladowy ciag tekstowy to:" << endl;
	cout << lancuch <<endl << endl;
	size_t pozycja = lancuch.find ("efgh",0);
	if (pozycja != string::npos)
	//npos przyjmuje wartosc -1
		cout <<"Pierwsze wystapienie 'efgh' nastapilo w
pozycji "
		<< pozycja;
	else
		cout <<"Nie znaleziono slowa 'efgh'";
	cout << endl <<endl;
```

```
        // Lokalizacja wszystkich wystapien
        size_t pozycja_1 = lancuch.find("abcd", 0);
        while (pozycja_1 != string::npos)
        {cout <<"Slowo abcd znaleziono w polozeniu "
        << pozycja_1<<endl;
        //Funkcja find wyszukuje dalej
        size_t pozycja_2 = pozycja_1 +1;
        pozycja_1 = lancuch.find("abcd", pozycja_2);
        }
        cout <<endl;
        system("pause");
}
```

```
 Przykladowy ciag tekstowy to:
 abcd efgh ijkl mnop rst abcdefgh

 Pierwsze wystapienie 'efgh' nastapilo w pozycji 5

 Slowo abcd znaleziono w polozeniu 0
 Slowo abcd znaleziono w polozeniu 24

 Press any key to continue . . .
```

```
// Dzialania na zawartosci ciągu tekstowego

#include<string>
#include<algorithm>
#include<iostream>
using namespace std;

void main()
{
	string lancuch
("abcdefghijklmnoqprstuvwxyzABCEEFGHIJKLMNOPQRSTUVWXYZ0123456789"
);
	cout << "Poczatkowa wartosc ciagu to: " <<endl;
	cout << lancuch <<endl<<endl;
	//Odwracanie ciagu tekstowego
	string lancuch_1(lancuch);
	reverse(lancuch.begin(),lancuch.end());
	cout<<"Ciag odwrocony :"<<endl;
	cout << lancuch <<endl<<endl;
	//Usuniecie 10 znaków od 15
	cout << "10 znakow od 15 zostalo usunietych" <<endl;
	lancuch_1.erase(14,10);
	cout <<lancuch_1<<endl<<endl;
```

```
// Usuniecie P
cout <<"Usuniecie P"<<endl<<endl;
string::iterator iter =
find(lancuch.begin(),lancuch.end(),'P');
       if(iter != lancuch.end())
              lancuch.erase(iter);
       cout <<lancuch <<endl<<endl;

//Usuniecie znakow za pomoca przeciazonej wersji erase()

cout<<"Usuniecie zakresu znakow od begin do end"<<endl<<endl;
lancuch.erase(lancuch.begin(),lancuch.end());
if (lancuch.length() ==0)
cout<<"Ciag tekstowy jest pusty"<<endl;

// Konwersja wiekości znaków
transform(lancuch_1.begin(),lancuch_1.end(),lancuch_1.begin(),
toupper);
cout << "Ciag w ktorym znaki skonwertowano na duze: " <<endl;
cout << lancuch_1 <<endl<<endl;
transform(lancuch_1.begin(),lancuch_1.end(),lancuch_1.begin(),
tolower);
```

```
cout << "Ciag w ktorym znaki skonwertowano na male: "
<<endl;
	cout << lancuch_1 <<lancuch_1<<endl;
	system("pause");
}
```

```
Poczatkowa wartosc ciagu to:
abcdefghijklmnoqprstuvwxyzABCEEFGHIJKLMNOPQRSTUVWXYZ0123456789

Ciag odwrocony :
9876543210ZYXWVUTSRQPONMLKJIHGFEECBAzyxwvutsrpqonmlkjihgfedcba

10 znakow od 15 zostalo usunietych
abcdefghijklmnyzABCEEFGHIJKLMNOPQRSTUVWXYZ0123456789

Usuniecie P

9876543210ZYXWVUTSRQONMLKJIHGFEECBAzyxwvutsrpqonmlkjihgfedcba

Usuniecie zakresu znakow od begin do end

Ciag tekstowy jest pusty
Ciag w ktorym znaki skonwertowano na duze:
ABCDEFGHIJKLMNYZABCEEFGHIJKLMNOPQRSTUVWXYZ0123456789

Ciag w ktorym znaki skonwertowano na male:
abcdefghijklmnyzabceefghijklmnopqrstuvwxyz0123456789abcdefghij
klmnyzabceefghijkl
mnopqrstuvwxyz0123456789
Press any key to continue . . .
```

```
// Dzialanie iteratorow

#include<string>
#include<algorithm>
#include<iostream>
using namespace std;

void main()
{
	string lancuch ("abcdefghijklmnopqrstuvwxyz");
	string::iterator iter=lancuch.begin()+5;
	cout << "Poczatkowa wartosc ciagu to: " <<endl;
	cout << lancuch <<endl<<endl;
	cout<<"Wartosc iteratora : "<<*iter<<endl<<endl;
	reverse(iter,iter+6);
	cout<<"Ciag odwrocony :"<<endl;
	cout << lancuch <<endl;
	cout<<"Wartosc iteratora : "<<*iter<<endl<<endl;
	iter+=6;
	cout<<"Wartosc iteratora po przesunieciu:
	"<<*iter<<endl<<endl;
```

```
lancuch.erase(iter);
	cout<<"Po usunieciu 1 znaku :"<<endl;
	cout << lancuch <<endl;
	cout<<"Wartosc iteratora : "<<*iter<<endl<<endl;
	lancuch.erase(iter,iter+4);
	cout<<"Po usunieciu 4 znakow :"<<endl;
	cout << lancuch <<endl;
	cout<<"Wartosc iteratora : "<<*iter<<endl<<endl;
	lancuch.insert(iter,'0');
	cout<<"Po wstawieniu :"<<endl;
	cout << lancuch <<endl;
	cout<<"Wartosc iteratora : "<<*iter<<endl<<endl;
	lancuch.insert(iter,4,'1');
	cout<<"Po wstawieniu 4 znakow :"<<endl;
	cout << lancuch <<endl;
	iter+=4;
	cout<<"Wartosc iteratora : "<<*iter<<endl<<endl;

	system("pause");
}
```

```
Poczatkowa wartosc ciagu to:
abcdefghijklmnopqrstuvwxyz

Wartosc iteratora : f

Ciag odwrocony :
abcdekjihgflmnopqrstuvwxyz
Wartosc iteratora : k

Wartosc iteratora po przesunieciu: l

Po usunieciu 1 znaku :
abcdekjihgfmnopqrstuvwxyz
Wartosc iteratora : m

Po usunieciu 4 znakow :
abcdekjihgfqrstuvwxyz
Wartosc iteratora : q

Po wstawieniu :
abcdekjihgf0qrstuvwxyz
Wartosc iteratora : 0

Po wstawieniu 4 znakow :
abcdekjihgf11110qrstuvwxyz
Wartosc iteratora : 0
Press any key to continue . . .
```

```
// Wektory inicjalizacja

#include<iostream>
#include<vector>
using namespace std;

template<typename T> void Drukuj(vector<T> wektor)
{
	vector<T>::iterator it;
	cout<<"[";
	for( it=wektor.begin(); it!=wektor.end(); it++)
	{cout<<*it<<" ";}
	cout<<"]"<<endl;
}

void main()
{
	vector <int> wektor(6,4);
	vector<int>::iterator iter;
	Drukuj(wektor);
	iter=wektor.begin();
	iter+=2;
```

```
        wektor.insert(iter,3,6);
        Drukuj(wektor);
        iter=wektor.begin();
        iter+=5;
        wektor.erase(iter,wektor.end());
        Drukuj(wektor);
        wektor.pop_back();
        Drukuj(wektor);
        vector <int> wektor1(wektor);
        Drukuj(wektor1);
        vector <int> wektor2(8,0);
        for (int k=0; k<7;k++)
        {wektor2[k]=2*k+1;}
        Drukuj(wektor2);

        system("pause");
}
```

```
[4 4 4 4 4 4 ]
[4 4 6 6 6 4 4 4 4 ]
[4 4 6 6 6 ]
[4 4 6 6 ]
[4 4 6 6 ]
[1 3 5 7 9 11 13 0 ]
Press any key to continue . . .
```

```
//find, distance

#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;

template<typename T> void Drukuj(vector<T> wektor)
{
	vector<T>::iterator it=wektor.begin();
	cout<<"[ ";
	while (it!=wektor.end())
	{
		cout<<*it<<" ";
		++it;
	}
	cout<<"]"<<endl;
}
```

```
template<typename T> int Pozycja(vector<T> wektor, T elem)
{
	vector<T>::iterator
	iter=find(wektor.begin(),wektor.end(),elem);
	int odleglosc=distance(wektor.begin(),iter);
	return odleglosc;
}

void main()
{
	vector<char> wektor;
	for (int i=80; i<90; i++)
	{wektor.push_back(char(i));}
	Drukuj(wektor);
	wektor.pop_back();
	Drukuj(wektor);
	cout<<"T znajduje sie na pozycji :
	"<<Pozycja(wektor,'T')<<endl;
	system("pause");
}
```

```
[ P Q R S T U V W X Y ]
[ P Q R S T U V W X ]
T znajduje sie na pozycji : 4
Press any key to continue . . .
```

```
//size capacity

#include<iostream>
#include<vector>
using namespace std;

template<typename T> void Drukuj(vector<T> wektor)
{
	vector<T>::iterator it=wektor.begin();
	cout<<"[ ";
	while (it!=wektor.end())
	{
		cout<<*it<<" ";
		++it;
	}
	cout<<"]\n";
}
```

```
void main()
{
	vector<char> wektor(4,'c');
	Drukuj(wektor);
	cout<<"rozmiar  : "<<wektor.size()<<endl;
	cout<<"pojemnosc  : "<<wektor.capacity()<<endl;
	wektor.push_back('d');
	wektor.push_back('e');
	Drukuj(wektor);
	cout<<"rozmiar  : "<<wektor.size()<<endl;
	cout<<"pojemnosc  : "<<wektor.capacity()<<endl;
	wektor.insert(wektor.begin(),2,'a');
	Drukuj(wektor);
	cout<<"rozmiar  : "<<wektor.size()<<endl;
	cout<<"pojemnosc  : "<<wektor.capacity()<<endl;
	vector<char> wektor1(3,'f');
	Drukuj(wektor1);
	cout<<"rozmiar  : "<<wektor1.size()<<endl;
	cout<<"pojemnosc  : "<<wektor1.capacity()<<endl;
	wektor.insert(wektor.begin()+4,wektor1.begin(),wektor1
	.end());
```

```
        Drukuj(wektor);
        cout<<"rozmiar  : "<<wektor.size()<<endl;
        cout<<"pojemnosc  : "<<wektor.capacity()<<endl;
        wektor.pop_back();
        wektor.pop_back();
        Drukuj(wektor);
        cout<<"rozmiar  : "<<wektor.size()<<endl;
        cout<<"pojemnosc  : "<<wektor.capacity()<<endl;

        system("pause");
}
```

```
[ c c c c ]
rozmiar  : 4
pojemnosc  : 4
[ c c c c d e ]
rozmiar  : 6
pojemnosc  : 6
[ a a c c c c d e ]
rozmiar  : 8
pojemnosc  : 9
[ f f f ]
rozmiar  : 3
pojemnosc  : 3
[ a a c c f f f c c d e ]
rozmiar  : 11
pojemnosc  : 13
[ a a c c f f f c c ]
rozmiar  : 9
pojemnosc  : 13
Press any key to continue . . .
```

```
// Mechanika ruchu iteratora

#include<iostream>
#include<vector>
using namespace std;

template<typename T> void Drukuj(vector<T> wektor)
{
	vector<T>::iterator it;
	cout<<"[ ";
	for( it=wektor.begin(); it!=wektor.end(); it++)
	{cout<<*it<<" ";}
	cout<<"]"<<endl;
}

void main()
{
	vector<int> wektor;
	int k;
	for (k=0; k<8 ; k++)
	{wektor.push_back(3*k+2);}
	Drukuj(wektor);
```

```
        vector<int>::iterator it;
        it=wektor.begin()+2;
        cout<<"Iterator wskazuje :"<<*it<<endl;
        wektor.push_back(30+k);
        Drukuj(wektor);
        cout<<"Iterator wskazuje :"<<*it<<endl;
        wektor.insert(wektor.begin(),0);
        Drukuj(wektor);
        /*cout<<"Iterator wskazuje :"<<*it<<endl;
        Program kompiluje się ale błąd wykonania*/
        it=wektor.end()-4;
        cout<<"Iterator wskazuje :"<<*it<<endl;
        system("pause");
}
```

```
[ 2 5 8 11 14 17 20 23 ]
Iterator wskazuje :8
[ 2 5 8 11 14 17 20 23 38 ]
Iterator wskazuje :8
[ 0 2 5 8 11 14 17 20 23 38 ]
Iterator wskazuje :17
Press any key to continue . . .
```

Klasa STL deque

Podobna do klasy STL vector, ale umożliwia wstawianie elementów na początku i na końcu

```
// Używanie klasy STL deque

#include<deque>
#include<iostream>
#include <algorithm>
using namespace std;

template<typename T> void Drukuj(deque<T> wektor)
{
	deque<T>::iterator it=wektor.begin();
	cout<<"Zawartosc tablicy : \n[ ";
	while (it!=wektor.end())
	{
		cout<<*it<<" ";
		++it;
	}
	cout<<"]\n";
}
```

```
void main()
{
	deque <char> obiekt;
	//Wstawianie liczb calkowitych na koncu tablicy
	int i;
	for (i=80; i<85; i++)
	{obiekt.push_back(char(i));}
	Drukuj(obiekt);
	//Wstawianie liczb calkowitych na poczatku
	for (i=79; i>74; i--)
		{obiekt.push_front(char(i));}
	Drukuj(obiekt);
	// Usuniecie elementu z konca tablicy
	obiekt.pop_back();
	//Usuniecie elementu z poczatku
	obiekt.pop_front();
	//Wyswietlenie elementow za pomoca iteratorow
	cout <<"Po usunieciu elementow :\n";
	Drukuj(obiekt);
	system("pause");
}
```

```
Zawartosc tablicy :
[ P Q R S T ]
Zawartosc tablicy :
[ K L M N O P Q R S T ]
Po usunieciu elementow :
Zawartosc tablicy :
[ L M N O P Q R S ]
Press any key to continue . . .
```

```
// Podstawowe operacje klasy STL list

#include <list>
#include <iostream>
using namespace std;

template<typename typ> void Drukuj(list<typ> lista);

int main()
{
	list<char> lista_znakow;
	int i;
	// Wstawianie elementow na poczatku
	for (i=6 ; i>0; i--)
	{lista_znakow.push_front(char(100+i));}
	cout <<"Lista 1 :"<<endl;
	Drukuj(lista_znakow);
	// Wstawianie elementow na koncu
	for (i=0 ; i<6; i++)
	{lista_znakow.push_back(char(110+i));}
	cout <<"Lista 2 :"<<endl;
	Drukuj(lista_znakow);
```

```
//Wstawianie znakow
        lista_znakow.insert(lista_znakow.begin(),'A');
        lista_znakow.insert(lista_znakow.end(),'Z');
        //lista_znakow.insert(lista_znakow.begin()+3,'C');
        //lista_znakow.insert(lista_znakow.end()-1,'Y');
        cout <<"Lista 3 :"<<endl;
        Drukuj(lista_znakow);
        list<char>::iterator iter;
        iter=lista_znakow.begin();
        //iter+=3;
        for ( i=0;i<3; i++)
        {iter++;}
        lista_znakow.insert(iter,4,'B');
        iter=lista_znakow.end();
        for ( i=0;i<3; i++)
        {iter--;}
        lista_znakow.insert(iter,4,'Y');
        cout <<"Lista 4 :"<<endl;
        Drukuj(lista_znakow);
        list<char> lista;
```

```
for ( i=0;i<4; i++)
        {lista.push_front('A');}
        //lista_znakow.insert(iter,lista);
        lista_znakow.insert(iter,lista.begin(),lista.end());
        cout <<"Lista 5 :"<<endl;
        Drukuj(lista_znakow);

        system("pause");
}
        template<typename typ> void Drukuj(list<typ> lista)
        {
                if (lista.size() > 0)
                {
                        cout << "{ ";
                        list<typ>::iterator iter;
                        for (iter = lista.begin();
                                iter != lista.end();
                                ++ iter)
                                cout << *iter << " ";
                        cout <<"}" << endl;}
                else
                cout <<"Pusta lista" << endl;
        }
```

```
Lista 1 :
{ e f g h i j }
Lista 2 :
{ e f g h i j n o p q r s }
Lista 3 :
{ A e f g h i j n o p q r s Z }
Lista 4 :
{ A e f B B B B g h i j n o p q Y Y Y Y r s Z }
Lista 5 :
{ A e f B B B B g h i j n o p q Y Y Y Y A A A A r s Z }
Press any key to continue . . .
```

```
// Podstawowe operacje klasy STL list

#include <list>
#include <iostream>
using namespace std;

template<typename typ> void Drukuj(list<typ> lista);

int main()
{
	list<int> lista_liczb;
	int i;
	// Wstawianie elementow na koncu
	for (i=0 ; i<15; i++)
	{lista_liczb.push_back(2*i+1);}
	cout <<"Lista 1 :"<<endl;
	Drukuj(lista_liczb);
	list<int>::iterator it;
	it=lista_liczb.begin();
	it++;
```

```
//Usuwanie liczb
for(i=0; i<3; ++i)
{lista_liczb.pop_front();}
lista_liczb.pop_back();
cout <<"Lista 2 :"<<endl;
Drukuj(lista_liczb);
//cout<<"Iterator wskazuje : "<<*it<<endl;
it=lista_liczb.begin();
for(i=0; i<4; ++i)
{it++;}
//lista_liczb.erase(it,3);
list<int>::iterator it1;
it1=lista_liczb.begin();
for(i=0; i<7; ++i)
{it1++;}
cout <<"Lista 3 :"<<endl;
lista_liczb.erase(it,it1);
Drukuj(lista_liczb);
//cout<<"Iterator 1 wskazuje : "<<*it<<endl;
cout<<"Iterator 2 wskazuje : "<<*it1<<endl;
```

```
cout <<"Lista 4 :"<<endl;
	lista_liczb.erase(lista_liczb.begin(),it1);
	Drukuj(lista_liczb);
	cout<<"Iterator 2 wskazuje : "<<*it1<<endl;
	cout <<"Lista 5 :"<<endl;
	lista_liczb.erase(lista_liczb.begin(),lista_liczb.end
());
	Drukuj(lista_liczb);

	system("pause");
}
```

```
template<typename typ> void Drukuj(list<typ> lista)
	{
		if (lista.size() > 0)
		{
			cout << "{ ";
			list<typ>::iterator iter;
			for (iter = lista.begin();
				iter != lista.end();
				++ iter)
				cout << *iter << " ";
			cout <<"}" << endl;}
		else
		cout <<"Pusta lista" << endl;
	}
```

```
Lista 1 :
{ 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 }
Lista 2 :
{ 7 9 11 13 15 17 19 21 23 25 27 }
Lista 3 :
{ 7 9 11 13 21 23 25 27 }
Iterator 2 wskazuje : 21
Lista 4 :
{ 21 23 25 27 }
Iterator 2 wskazuje : 21
Lista 5 :
Pusta lista
Press any key to continue . . .
```

```
// Sortowanie list

#include <list>
#include <iostream>
using namespace std;

template<typename typ> void Drukuj(list<typ> lista);
bool Sort(int &elem_1,int &elem_2);


int main()
{
	list<char> lista;
	int i;
	for (i=0 ; i<15; i++)
	{lista.push_back(char(120-i));}
	Drukuj(lista);
	lista.sort();
	Drukuj(lista);
	lista.reverse();
	Drukuj(lista);
```

```
list<int>lista_1;
	for (i=0 ; i<15; i++)
	{lista_1.push_front(3*i);}
	Drukuj(lista_1);
	lista_1.sort();
	Drukuj(lista_1);
	lista_1.sort(Sort);
	Drukuj(lista_1);

	system("pause");
}
```

```
template<typename typ> void Drukuj(list<typ> lista)
	{
		if (lista.size() > 0)
		{
			cout << "{ ";
			list<typ>::iterator iter;
			for (iter = lista.begin();
				iter != lista.end();
				++ iter)
				cout << *iter << " ";
			cout <<"}" << endl;}
		else
		cout <<"Pusta lista" << endl;
	}

bool Sort(int &elem_1,int &elem_2)
{	//zamiana miejsc
	return (elem_2 < elem_1);
}
```

```
{ x w v u t s r q p o n m l k j }
{ j k l m n o p q r s t u v w x }
{ x w v u t s r q p o n m l k j }
{ 42 39 36 33 30 27 24 21 18 15 12 9 6 3 0 }
{ 0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 }
{ 42 39 36 33 30 27 24 21 18 15 12 9 6 3 0 }
Press any key to continue . . .
```

Klasy set i multiset

Są to kontenery asocjacyjne ułatwiające szybkie wyszukiwanie elementów.

1. multiset pozwala na przechowywanie wartości powtarzających się, set – unikalnych.
2. Elementy wstawiane do obiektów są sortowane w kolejności ich wstawiania
3. W obiekcie set element w danym położeniu nie może zostać zastąpiony przez nowy element o innej wartości.

set<typ obiektu, opcjonalna_funkcja_bool>
nazwa_obiektu;

```
// W_13_01
// Podstawowe operacje klasy set

#include<set>
#include<iostream>
using namespace std;

typedef set<char, greater<char>> CharMalejacy;
//Sortowanie w kolejności malejącej
typedef set<char> CharRosnacy;
//Sortowanie w kolejności rosnacej

template<typename T> void Drukuj (T obiekt);

void main()
{
	set <int> setInt;
	int k;
	for (k=47; k>20 ;k-=3)
	{setInt.insert(k);}

	cout << "Obiekt set :" <<endl;
	Drukuj(setInt);
```

```
setInt.insert(47);
setInt.insert(33);
Drukuj(setInt);
CharMalejacy setChar;
for (k=75; k<82;k++)
{setChar.insert(char(k));}
Drukuj(setChar);
setChar.insert('a');
Drukuj(setChar);
CharRosnacy setCharRosn;
for (k=102; k>96;k--)
{setCharRosn.insert(char(k));}
Drukuj(setCharRosn);
setChar.insert(setCharRosn.begin(),setCharRosn.end());
Drukuj(setChar);
// Wyszukiwanie elementu

CharMalejacy::iterator iZnajdz = setChar.find('a');
if (iZnajdz !=setChar.end())
       cout << "Znaleziono element " <<*iZnajdz<<endl;
else
       cout << "Nie znaleziono elementu " <<endl;
```

```
	iZnajdz = setChar.find('k');
	if (iZnajdz !=setChar.end())
		cout << "Znaleziono element " <<*iZnajdz<<endl;
	else
		cout << "Nie znaleziono elementu " <<endl;

	//Usuwanie elementów set
	cout<<"Podaj element do usuniecia :"<<endl;
	char znak;
	cin>>znak;
	cout << "Usuwanie "<< setChar.count(znak)<<" elementow
"<<znak<<endl;
	setChar.erase(znak);
	cout<<"Po usunieciu :"<<endl;
	Drukuj(setChar);
	//Opróżnienie setInt
	setChar.clear();
	cout<<"Obiekt setChar"<<endl;
	Drukuj(setChar);
	system("pause");
}
```

```
template<typename T> void Drukuj(T obiektSTL)
{
	if (obiektSTL.size()!=0)
	{
		T::iterator iter = obiektSTL.begin();
		while (iter != obiektSTL.end())
		{
			cout << *iter <<"  ";
			++ iter;
		}
		cout <<endl;
	}
	else
		cout<<"Pusty obiekt\n";
}
```

```
Obiekt set :
23  26  29  32  35  38  41  44  47
23  26  29  32  33  35  38  41  44  47
Q  P  O  N  M  L  K
a  Q  P  O  N  M  L  K
a  b  c  d  e  f
f  e  d  c  b  a  Q  P  O  N  M  L  K
Znaleziono element a
Nie znaleziono elementu
Podaj element do usuniecia :
K
Usuwanie 1 elementow K
Po usunieciu :
f  e  d  c  b  a  Q  P  O  N  M  L
Obiekt setChar
Pusty obiekt
Press any key to continue . . .
```

```
// W_13_02
// Podstawowe operacje klasy multiset

#include<set>
#include<iostream>
using namespace std;

typedef multiset<char, greater<char>> CharMalejacy;
//Sortowanie w kolejności malejącej
typedef multiset<char> CharRosnacy;
//Sortowanie w kolejności rosnacej

template<typename T> void Drukuj (T obiekt);

void main()
{
	multiset <int> msetInt;
	int k;
	for (k=47; k>20 ;k-=3)
	{msetInt.insert(k);}
```

```
        cout << "Obiekt multiset :" <<endl;
        Drukuj(msetInt);
        msetInt.insert(47);
        msetInt.insert(33);
        Drukuj(msetInt);
        CharMalejacy msetChar;
        for (k=75; k<82;k++)
        {msetChar.insert(char(k));}
        Drukuj(msetChar);
        msetChar.insert('a');
        Drukuj(msetChar);
        CharRosnacy msetCharRosn;
        for (k=102; k>96;k--)
        {msetCharRosn.insert(char(k));}
        Drukuj(msetCharRosn);
        msetChar.insert(msetCharRosn.begin(),msetCharRosn.end()
);
        Drukuj(msetChar);
        // Wyszukiwanie elementu
        CharMalejacy::iterator iZnajdz = msetChar.find('a');
        if (iZnajdz !=msetChar.end())
               cout << "Znaleziono element " <<*iZnajdz<<endl;
        else
               cout << "Nie znaleziono elementu " <<endl;
```

```
        iZnajdz = msetChar.find('a');
        if (iZnajdz !=msetChar.end())
                cout << "Znaleziono element " <<*iZnajdz<<endl;
        else
                cout << "Nie znaleziono elementu " <<endl;
        //Usuwanie elementów set
        cout<<"Podaj element do usuniecia :"<<endl;
        char znak;
        cin>>znak;
        cout << "Usuwanie "<< msetChar.count(znak)<<" elementow
"<<znak<<endl;
        msetChar.erase(znak);
        cout<<"Po usunieciu :"<<endl;
        Drukuj(msetChar);

        //Opróżnienie setInt
        msetChar.clear();
        cout<<"Obiekt msetChar"<<endl;
        Drukuj(msetChar);
        system("pause");

}
```

```
template<typename T> void Drukuj(T obiektSTL)
{
	if (obiektSTL.size()!=0)
	{T::iterator iter = obiektSTL.begin();
	while (iter != obiektSTL.end())
	{
		cout << *iter <<"  ";
		++ iter;
	}
	cout <<endl;}
	else
		cout<<"Pusty obiekt\n";

}
```

```
Obiekt multiset :
23  26  29  32  35  38  41  44  47
23  26  29  32  33  35  38  41  44  47  47
Q  P  O  N  M  L  K
a  Q  P  O  N  M  L  K
a  b  c  d  e  f
f  e  d  c  b  a  a  Q  P  O  N  M  L  K
Znaleziono element a
Znaleziono element a
Podaj element do usuniecia :
a
Usuwanie 2 elementow a
Po usunieciu :
f  e  d  c  b  Q  P  O  N  M  L  K
Obiekt msetChar
Pusty obiekt
Press any key to continue . . .
```

Klasy map i multimap

Kontenery asocjacyjne par klucz - wartość, służące do wyszukiwania na podstawie klucza.

1. multimap pozwala na stosowanie duplikatów, map przechowuje wartości unikalne
2. Elementy wstawiane do obiektów map (multimap) są sortowane w trakcie wstawiania
3. Element klasy map w danym położeniu nie może(?) być zastąpiony przez element o innej wartości

map <typ_klucza, typ_wartości, Predicate::less<typ_klucza>> nazwa obiektu;

// Predykat pomaga w sortowaniu

```
// W_13_03
// Podstawowe operacje klasy map

#include <map>
#include<iostream>
using namespace std;

typedef map<int,char> mapa_rosnaca;
typedef map<int,char,greater<int>> mapa_malejaca;

template<typename T> void Drukuj (T obiekt);

void main()
{
	mapa_rosnaca map1;
	cout<<"Mapa rosnaca :\n";
	int k;
	// Wstawienie pary klucz - wartosc
	// 1. value_type
	for (k=97; k<100; k++)
	{map1.insert(mapa_rosnaca::value_type(k,char(k)));}
```

```
for (k=100; k<103; k++)
       // 2. make_pair
       map1.insert(make_pair(k,char(k)));
       // 3. Bezposrednie wstawienie obiektu pary
       map1.insert(pair<int, char>(104,char(104)));
       //4. Skladnia podobna do tablicy
       map1[105]= char(105);
       map1[106]= char(105);
       Drukuj(map1);
       map1[105]= char(106);
       Drukuj(map1);

       cout<<"Klucz elementu do usuniecia : ";
       cin>>k;
       mapa_rosnaca::iterator iter_rosn=map1.find(k);
       cout<<"\nElement o kluczu "<<k<<" to "<<iter_rosn-
>second<<endl;
       map1.erase(k);
       Drukuj(map1);
```

```
cout<<"Wartosc elementu do usuniecia : ";
	char znak;
	cin>>znak;
	map1.erase(znak);
	Drukuj(map1);
	map1.clear();
	Drukuj(map1);

	cout<<"Mapa malejaca :\n";
	mapa_malejaca map2;
	for (k=0; k<10; k++)
	map2.insert(make_pair(k,char(107-k)));
	Drukuj(map2);
	map2[1]= char(100);
	Drukuj(map2);

	system("pause");
}
```

```
template<typename T> void Drukuj (T obiekt)
{
	T::iterator iter;
	cout<<"Obiekt map zawiera "<<obiekt.size()<<" par
	klucz-wartosc\n";
	if (obiekt.size()!=0)
	{
		for (iter=obiekt.begin();
		iter != obiekt.end();
		iter++)
		{
		cout << "Klucz: " <<iter->first<<"\tWartosc :
		"<<iter->second;
		cout << endl;
		}
	}
	else
		cout << "Pusty obiekt\n";
}
```

```
Mapa rosnaca :
Obiekt map zawiera 9 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 100      Wartosc : d
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 104      Wartosc : h
Klucz: 105      Wartosc : i
Klucz: 106      Wartosc : i
Obiekt map zawiera 9 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 100      Wartosc : d
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 104      Wartosc : h
Klucz: 105      Wartosc : j
Klucz: 106      Wartosc : i
Klucz elementu do usuniecia : 100
```

```
Element o kluczu 100 to d
Obiekt map zawiera 8 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 104      Wartosc : h
Klucz: 105      Wartosc : j
Klucz: 106      Wartosc : i
Wartosc elementu do usuniecia : h
Obiekt map zawiera 7 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 105      Wartosc : j
Klucz: 106      Wartosc : i
Obiekt map zawiera 0 par klucz-wartosc
Pusty obiekt
```

```
Mapa malejaca :
Obiekt map zawiera 10 par klucz-wartosc
Klucz: 9        Wartosc : b
Klucz: 8        Wartosc : c
Klucz: 7        Wartosc : d
Klucz: 6        Wartosc : e
Klucz: 5        Wartosc : f
Klucz: 4        Wartosc : g
Klucz: 3        Wartosc : h
Klucz: 2        Wartosc : i
Klucz: 1        Wartosc : j
Klucz: 0        Wartosc : k
Obiekt map zawiera 10 par klucz-wartosc
Klucz: 9        Wartosc : b
Klucz: 8        Wartosc : c
Klucz: 7        Wartosc : d
Klucz: 6        Wartosc : e
Klucz: 5        Wartosc : f
Klucz: 4        Wartosc : g
Klucz: 3        Wartosc : h
Klucz: 2        Wartosc : i
Klucz: 1        Wartosc : d
Klucz: 0        Wartosc : k
Press any key to continue . . .
```

```
// W_13_04
// Podstawowe operacje klasy multimap

#include <map>
#include<iostream>
using namespace std;

typedef multimap<int,char> mapa_rosnaca;
typedef multimap<int,char,greater<int>> mapa_malejaca;

template<typename T> void Drukuj (T obiekt);

void main()
{
	mapa_rosnaca multimap1;
	cout<<"Mapa rosnaca :\n";
	int k;
	// Wstawienie pary klucz - wartosc
	// 1. value_type
	for (k=97; k<100; k++)
	{multimap1.insert(mapa_rosnaca::value_type(k,char(k)));}
	// 2. make_pair
	for (k=100; k<103; k++)
	multimap1.insert(make_pair(k,char(k)));
```

```
// 3. Bezposrednie wstawienie obiektu pary
for (k=103; k<106; k++)
multimap1.insert(pair<int, char>(k,char(k)));
//4. Skladnia podobna do tablicy nie kompiluje się
//multimap1[105]= char(105);
Drukuj(multimap1);
//Wstawienie duplikatów
for (k=103; k<106; k++)
multimap1.insert(make_pair(k,char(k-32)));

Drukuj(multimap1);

cout<<"Klucz elementu do usuniecia : ";
cin>>k;
mapa_rosnaca::iterator iter_rosn=multimap1.find(k);
cout<<"\nElement o kluczu "<<k<<" to "<<iter_rosn
->second<<endl;
multimap1.erase(k);
Drukuj(multimap1);
```

```
for (k=0; k<2; k++)
        {
        cout<<"Wartosc elementu do usuniecia : ";
        char znak;
        cin>>znak;
        multimap1.erase(znak);
        }
        Drukuj(multimap1);
        multimap1.clear();
        Drukuj(multimap1);

        cout<<"Mapa malejaca :\n";
        mapa_malejaca multimap2;
        for (k=0; k<10; k++)
        multimap2.insert(make_pair(k,char(107-k)));
        Drukuj(multimap2);
        multimap2.insert(make_pair(4,char(107)));
        Drukuj(multimap2);

        system("pause");
}
```

```
template<typename T> void Drukuj (T obiekt)
{
	T::iterator iter;
	cout<<"Obiekt map zawiera "<<obiekt.size()<<" par klucz-
wartosc\n";
	if (obiekt.size()!=0)
	{
		for (iter=obiekt.begin(); iter != obiekt.end();
iter++)
		{
		cout << "Klucz: " <<iter->first<<"\tWartosc :
"<<iter->second;
		cout << endl;
		}
	}
	else
		cout << "Pusty obiekt\n";
}
```

```
Mapa rosnaca :
Obiekt map zawiera 9 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 100      Wartosc : d
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 103      Wartosc : g
Klucz: 104      Wartosc : h
Klucz: 105      Wartosc : i
Obiekt map zawiera 12 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 100      Wartosc : d
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 103      Wartosc : g
Klucz: 103      Wartosc : G
Klucz: 104      Wartosc : h
Klucz: 104      Wartosc : H
Klucz: 105      Wartosc : i
Klucz: 105      Wartosc : I
```

```
Klucz elementu do usuniecia : 104
Element o kluczu 104 to h
Obiekt map zawiera 10 par klucz-wartosc
Klucz: 97       Wartosc : a
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 100      Wartosc : d
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 103      Wartosc : g
Klucz: 103      Wartosc : G
Klucz: 105      Wartosc : i
Klucz: 105      Wartosc : I
Wartosc elementu do usuniecia : a
Wartosc elementu do usuniecia : G
Obiekt map zawiera 9 par klucz-wartosc
Klucz: 98       Wartosc : b
Klucz: 99       Wartosc : c
Klucz: 100      Wartosc : d
Klucz: 101      Wartosc : e
Klucz: 102      Wartosc : f
Klucz: 103      Wartosc : g
Klucz: 103      Wartosc : G
Klucz: 105      Wartosc : i
Klucz: 105      Wartosc : I
```

```
Obiekt map zawiera 0 par klucz-wartosc
Pusty obiekt
Mapa malejaca :
Obiekt map zawiera 10 par klucz-wartosc
Klucz: 9        Wartosc : b
Klucz: 8        Wartosc : c
Klucz: 7        Wartosc : d
Klucz: 6        Wartosc : e
Klucz: 5        Wartosc : f
Klucz: 4        Wartosc : g
Klucz: 3        Wartosc : h
Klucz: 2        Wartosc : i
Klucz: 1        Wartosc : j
Klucz: 0        Wartosc : k
Obiekt map zawiera 11 par klucz-wartosc
Klucz: 9        Wartosc : b
Klucz: 8        Wartosc : c
Klucz: 7        Wartosc : d
Klucz: 6        Wartosc : e
Klucz: 5        Wartosc : f
Klucz: 4        Wartosc : g
Klucz: 4        Wartosc : k
Klucz: 3        Wartosc : h
Klucz: 2        Wartosc : i
Klucz: 1        Wartosc : j
Klucz: 0        Wartosc : k
Press any key to continue . . .
```

Algorytmy STL

Pełne zestawienie:
J. Liberty, S. Rao, B. L. Jones: C++ dla każdego, wyd. 2, Helion 2011

Obiekty funkcji

Są to obiekty działające jak funkcje – jedno lub dwuargumentowe. Jeśli funkcja zwraca wartość typu bool – nazywa się ją predykatem

```
// W_13_05
// Wyświetlenie na ekranie adresów kolekcji
// za pomocą funkcji jednoargumentowej

#include<iostream>
#include<algorithm>
#include<vector>
#include<string>
#include<list>
using namespace std;

string
napisy[5]={"napis_0","napis_1","napis_2","napis_3","napis_4"};

// klasa, która zachowuje się jak funkcja jednoargumentowa
template <typename typ>  class PokazAdres
{
public:
       void operator()      (typ element)
       {cout << element <<"\t"<<&element<<endl;}
};
```

```
void main()
{
	vector <string> wektor;
	for (int m = 0; m < 5; ++m)
	wektor.push_back(napisy[m]);

	list <int> listaInteger;
	for (int k = 4; k < 14; ++ k)
	listaInteger.push_back(k);
	cout << "Wyswietlenie adresow obiektu przechowujacego
lancuchy" << endl;
	for_each (wektor.begin(), wektor.end(),
	PokazAdres <string>());
	cout << endl <<endl;

	cout << "Wyswietlenie adresow obiektu przechowujacego
liczby" << endl;
	for_each (listaInteger.begin(),
	listaInteger.end(),PokazAdres<int>());
	cout << endl <<endl;
	system("pause");
}
```

```
Wyswietlenie adresow obiektu przechowujacego lancuchy
napis_0 00F8F558
napis_1 00F8F558
napis_2 00F8F558
napis_3 00F8F558
napis_4 00F8F558


Wyswietlenie adresow obiektu przechowujacego liczby
4       00F8F580
5       00F8F580
6       00F8F580
7       00F8F580
8       00F8F580
9       00F8F580
10      00F8F580
11      00F8F580
12      00F8F580
13      00F8F580


Press any key to continue . . .
```