Politechnika Świętokrzyska

Laboratorium

Programowanie w języku C++ 2

Ćwiczenie 1

Wstęp do standardowej biblioteki szablonów (STL)
Kontener vector

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie studentów z biblioteką STL języka C++.

dr inż Robert Kazała

Kontener vector

Metody kontenera vector

Category	Method	
		vector
construction	constructor	x
	destructor	x
	operator=	x
iterators	begin	x
	end	x
	rbegin	x
	rend	x
capacity	size	x
	max_size	x
	empty	x
	resize	x
	capacity	x
	reserve	x
access	front	x
	back	x
	operator[]	x
	at	x
modifiers	assign	x
	insert	x
	erase	x
	swap	x
	clear	x
	push_back	x
	pop_back	x

Przykład tworzenia kontenera vector

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

int main()
{
	vector <int> v1(10);
	vector <int> v2 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
	for(unsigned i = 0; i < v1.size(); ++i)
	{
		v1[i] = i;
	}
	for(unsigned i = 0; i < v1.size(); ++i)
	{
		cout << v1[i] << " ";
		cout << v2[i] << " ";
	}
	cout << endl;

	cout << v1.size() << endl;
	v1.push_back(100);
	cout << v1.size() << endl;
```

```
        v1.pop_back();
        cout << v1.size() << endl;
        return 0;
}

Output of the program:
0 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9
10
11
10
```

Tworzenie kontenera vector z konstruktorem inicjującym

```
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    vector<int> v1(10, 0);
    cout<<"Size: "<<v1.size()<<endl;
    for(unsigned i = 0; i < v1.size(); ++i)
    {
        cout<< v1[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

Tworzenie kontenera vector z konstruktorem kopiującym

```
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    //first one
    vector<int> v1(10, 0);
    for(unsigned i = 0; i < v1.size(); ++i)
    {
        v1[i]=i+1;
    }
    cout<<"Size (v1):  "<<v1.size()<<endl;
    for(unsigned i = 0; i < v1.size(); ++i)
    {
```

```
        cout<< v1[i]<<" ";
    }
    cout<<endl;
    //second one;
    vector<int> v2(v1.begin(), v1.begin()+5);
    cout<<"Size (v2):  "<<v2.size()<<endl;
    for(unsigned i = 0; i < v2.size(); ++i)
    {
        cout<< v2[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

Tworzenie kontenera vector z konstruktorem iterującym

```
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    int a1[]={1,2,3,4,5,6,7,8,9,10};
    //first one
    vector<int> v1(a1, a1+10);
    cout<<"Size (v1):  "<<v1.size()<<endl;
    for(unsigned i = 0; i < v1.size(); ++i)
    {
        cout<< v1[i]<<" ";
    }
    cout<<endl;
    //second one;
    vector<int> v2(a1+5,a1+10);
    cout<<"Size (v2):  "<<v2.size()<<endl;
    for(unsigned i = 0; i < v2.size(); ++i)
    {
        cout<< v2[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

Tworzenie kontenera vector z konstruktorem kopiującym

```
#include <vector>
#include <iostream>

using namespace std;

int main()
```

```
{
    int a1[]={1,2,3,4,5,6,7,8,9,10};
    //first one
    vector<int> v1(a1, a1+10);
    cout<<"Size (v1):  "<<v1.size()<<endl;
    for(unsigned i = 0; i < v1.size(); ++i)
    {
        cout<< v1[i]<<" ";
    }
    cout<<endl;
    //second one;
    vector<int> v2(v1);
    cout<<"Size (v2):  "<<v2.size()<<endl;
    for(unsigned i = 0; i < v2.size(); ++i)
    {
        cout<< v2[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

Literatura

Zadania

1. Uruchomić i przeanalizować działanie wszystkich przykładów z instrukcji.
2. Utworzyć zmodyfikowane wersje przykładów z instrukcji.
3. Zaprezentować na własnych przykładach tworzenie kontenerów vector dla różnych typów danych.
4. Na własnych przykładach pokazać działanie różnych rodzajów konstruktorów.
5. Przeanalizować i omówić działanie poniższych programów, wyjaśnić różnice między między metodami back(), begin(), end().

```
// vector::back
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;

  myvector.push_back(10);
```

```
  while (myvector.back() != 0)
  {
    myvector.push_back ( myvector.back() -1 );
  }

  std::cout << "myvector contains:";
  for (unsigned i=0; i<myvector.size() ; i++)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}



// vector::begin/end
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;
  for (int i=1; i<=5; i++) myvector.push_back(i);

  std::cout << "myvector contains:";
  for (std::vector<int>::iterator it = myvector.begin() ; it !=
myvector.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

6. Przeanalizować i omówić działanie poniższego programu i metody resize(). Pokazać na własnym przykładzie działanie metody resize().

```
// resizing vector
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;

  // set some initial content:
  for (int i=1;i<10;i++) myvector.push_back(i);

  myvector.resize(5);
  myvector.resize(8,100);
  myvector.resize(12);
```

```
  std::cout << "myvector contains:";
  for (int i=0;i<myvector.size();i++)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}
```

7. Przeanalizować i omówić działanie poniższego programu i metody erase(). Pokazać na własnym przykładzie działanie metody erase().

```
// erasing from vector
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;

  // set some values (from 1 to 10)
  for (int i=1; i<=10; i++) myvector.push_back(i);

  // erase the 6th element
  myvector.erase (myvector.begin()+5);

  // erase the first 3 elements:
  myvector.erase (myvector.begin(),myvector.begin()+3);

  std::cout << "myvector contains:";
  for (unsigned i=0; i<myvector.size(); ++i)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}
```

8. Przeanalizować i omówić działanie poniższego programu i metody assign(). Pokazać na własnym przykładzie działanie metody assign().

```
// vector assign
#include <iostream>
#include <vector>

int main ()
{
```

```
  std::vector<int> first;
  std::vector<int> second;
  std::vector<int> third;

  first.assign (7,100);

  std::vector<int>::iterator it;
  it=first.begin()+1;

  second.assign (it,first.end()-1);

  int myints[] = {1776,7,4};
  third.assign (myints,myints+3);

  std::cout << "Size of first: " << int (first.size()) << '\n';
  std::cout << "Size of second: " << int (second.size()) << '\n';
  std::cout << "Size of third: " << int (third.size()) << '\n';
  return 0;
}
```