**Лабораторна робота №7**

# Стандартна бібліотека STL і сереалізація в C#.

**МЕТА:** Освоїти технологію узагальненого програмування з використанням

бібліотеки стандартних шаблонів (STL) та навчитись сереалізувати об’єкти в С# .

**Постановка задачі:**

**В-13**

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | priority\_queue | stack | int |

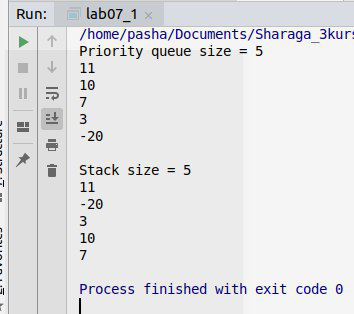
**Задача №1**

Використати контейнери із типом int.

**Код пограми C++:**

#include **<iostream>**#include **<bits/stdc++.h>  
  
using namespace** std;  
  
**int** main() {priority\_queue <**int**> q;  
 q.push(7);  
 q.push(10);  
 q.push(3);  
 q.push(-20);  
 q.push(11);  
  
 cout << **"Priority queue size = "** << q.size() << endl;  
 **while** (!q.empty()){  
 cout << q.top() << endl;  
 q.pop();  
 }  
  
 cout << endl;  
  
 stack <**int**> s;  
 s.push(7);  
 s.push(10);  
 s.push(3);  
 s.push(-20);  
 s.push(11);  
  
 cout << **"Stack size = "** << s.size() << endl;  
 **while** (!s.empty()){  
 cout << s.top() << endl;  
 s.pop();  
 }  
  
  
  
  
 **return** 0;  
}

**Результат:**



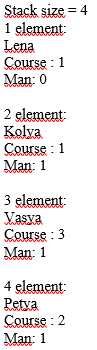
**Задача №2**

Використати контейнери із типом власного класу.

**Код пограми C++:**

#include **<bits/stdc++.h>  
  
using namespace** std;  
  
**class** Student  
{  
 **private**:  
 **char** \*name = **"defaultName"**;  
 **int** course = -1;  
 **bool** man = 0;  
  
 **public**:  
 Student();  
 Student(**char** \*n, **int** c, **bool** m);  
 *//Student(const Student &obj);* ~Student();  
  
 **void** input();  
 **void** output();  
 **void** setName(**char** \*n);  
 **char**\* getName() **const**;  
 **void** setCourse(**int** c);  
 **int** getCourse() **const**;  
 **void** setGender(**bool** m);  
 **bool** getGender() **const**;  
};  
  
Student::Student(){};  
  
Student::Student(**char** \*n, **int** c, **bool** m)  
{  
 setName(n);  
 setCourse(c);  
 setGender(m);  
}  
  
Student::~Student(){*/\*cout << "Object " << name << " deleted" << endl;\*/*};  
**void** Student::input()  
{  
 **char** \*n = **new char**[20];  
 **int** c;  
 **bool** m;  
 **int** i = 0;  
  
 **while** (i < 1)  
 {  
 **try** {  
 cout << **"Enter name: "**;  
 cin >> n;  
 setName(n);  
  
 cout << **"Enter course: "**;  
 cin >> c;  
 setCourse(c);  
  
 cout << **"Is it man (1) or woman (0)"**;  
 cin >> m;  
 setGender(m);  
 }  
 **catch**(exception e)  
 {  
 cout << **"Please, again.."**;  
 **continue**;  
 }  
 ++i;  
 }  
}  
  
**void** Student::output()  
{  
 cout << **"Name : "** << **this**->name << endl;  
 cout << **"Course : "** << **this**->course << endl;  
 cout << **"Man : "** << **this**->man << endl;  
}  
  
**void** Student::setName(**char** \*n)  
{  
 string str = n;  
 **if** (str.length() > 20 || find\_if(str.begin(), str.end(), ::isdigit) != str.end())  
 {  
 cout << **"Incorrect name"** << endl;  
 }  
 **else** name = n;  
}  
  
**char**\* Student::getName() **const**{  
 **return** name;  
}  
  
**void** Student::setCourse(**int** c)  
{  
 c > 0 && c < 5 ? course = c : course = -1;  
 **if** (course == -1) cout << **"Incorrect course"** << endl;  
}  
  
**int** Student::getCourse() **const**{  
 **return** course;  
}  
  
**void** Student::setGender(**bool** m)  
{  
 **if** (m == 1 || m == 0) man = m; **else** cout << **"Incorrect gender"** << endl;  
}  
  
**bool** Student::getGender() **const**{  
 **return** man;  
}  
  
**bool operator**<(**const** Student& lhs, **const** Student& rhs)  
{  
 **return** lhs.getCourse() < rhs.getCourse();  
}  
  
**int** main()  
{  
 Student s1(**"Petya"**, 2, **true**);  
 Student s2(**"Vasya"**, 3, **true**);  
 Student s3(**"Kolya"**, 1, **true**);  
 Student s4(**"Lena"**, 1, **false**);  
  
 priority\_queue <Student> q;  
 q.push(s1);  
 q.push(s2);  
 q.push(s3);  
 q.push(s4);  
  
 cout << **"Priority queue size = "** << q.size() << endl;  
 **int** i = 1;  
 **while** (!q.empty()){  
 cout << i++ << **" element:"** << endl;  
 cout << q.top().getName() << endl;  
 cout << **"Course : "** << q.top().getCourse() << endl;  
 cout << **"Man: "** << q.top().getGender() << endl << endl;  
 q.pop();  
 }  
  
 cout << endl;  
  
 stack <Student> s;  
 s.push(s1);  
 s.push(s2);  
 s.push(s3);  
 s.push(s4);  
  
 cout << **"Stack size = "** << s.size() << endl;  
 i = 1;  
 **while** (!s.empty()){  
 cout << i++ << **" element:"** << endl;  
 cout << s.top().getName() << endl;  
 cout << **"Course : "** << s.top().getCourse() << endl;  
 cout << **"Man: "** << s.top().getGender() << endl << endl;  
 s.pop();  
 }  
  
**return** 0;  
}

**Результат:**

Priority queue size = 4

1 element:

Vasya

Course : 3

Man: 1

2 element:

Petya

Course : 2

Man: 1

3 element:

Lena

Course : 1

Man: 0

4 element:

Kolya

Course : 1

Man: 1

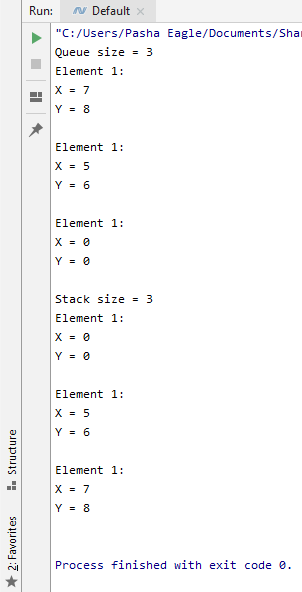
**Задача №3**

Використати контейнери із типом власного класу у C#.

**Код пограми C#:**

using System;  
using System.Collections.Generic;  
  
namespace lab07\_1  
{  
 class Point  
 {  
 private int x;  
 private int y;  
 public int X { set { x = value; } get { return x; } }  
 public int Y { set { y = value; } get { return y; } }  
 public Point() { }  
 public Point(int x, int y)  
 {  
 this.x = x;  
 this.y = y;  
 }  
  
 public void show()  
 {  
 Console.WriteLine( "X: " + this.x );  
 Console.WriteLine( "Y: " + this.y );  
 }  
  
 public double distance()  
 {  
 return (Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2)));  
 }  
  
 public void onVector(int a, int b)  
 {  
 Console.WriteLine( "X: " + (a + x));  
 Console.WriteLine( "Y: " + b + y );  
 }  
  
 public void setPointXY(int x, int y)  
 {  
 this.x = x;  
 this.y = y;  
 }  
  
 public void XYMultiplyBy(int z)  
 {  
 x \*= z;  
 y \*= z;  
 }  
  
 };  
  
  
 internal class Program  
 {  
 static void Main(string[] args)  
 {  
 Point p1 = new Point(7, 8);  
 Point p2 = new Point(5, 6);  
 Point p3 = new Point(0, 0);  
  
 Queue<Point> q = new Queue<Point>();  
   
 q.Enqueue(p1);  
 q.Enqueue(p2);  
 q.Enqueue(p3);  
  
 Console.WriteLine("Queue size = " + q.Count);  
 int i = 1;  
 while (q.Count > 0)  
 {  
 Console.WriteLine("Element {0}:", i);  
 Console.WriteLine("X = {0}", q.Peek().X);  
 Console.WriteLine("Y = {0}", q.Peek().Y);  
 Console.WriteLine();  
 q.Dequeue();  
 }  
  
   
   
 Stack<Point> s = new Stack<Point>();  
   
 s.Push(p1);  
 s.Push(p2);  
 s.Push(p3);  
  
 Console.WriteLine("Stack size = " + s.Count);  
 i = 1;  
 while (s.Count > 0)  
 {  
 Console.WriteLine("Element {0}:", i);  
 Console.WriteLine("X = {0}", s.Peek().X);  
 Console.WriteLine("Y = {0}", s.Peek().Y);  
 Console.WriteLine();  
 s.Pop();  
 }  
  
 }  
 }  
}

**Результат:**



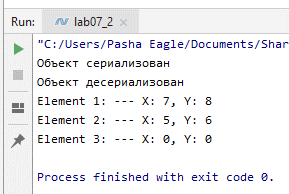
**Задача №4**

Зробити серіалізацію у файл у форматі SOAP, та десеріалізувати (C#)

**Код пограми C#:**

using System;  
using System.Collections.Generic;  
using System.IO;  
using System.Runtime.Serialization.Formatters.Soap;  
  
  
namespace lab07\_2  
{  
 [Serializable]  
 class Point  
 {  
 private int x;  
 private int y;  
 public int X { set { x = value; } get { return x; } }  
 public int Y { set { y = value; } get { return y; } }  
 public Point() { }  
 public Point(int x, int y)  
 {  
 this.x = x;  
 this.y = y;  
 }  
  
 public void show()  
 {  
 Console.WriteLine( "X: " + this.x );  
 Console.WriteLine( "Y: " + this.y );  
 }  
  
 public double distance()  
 {  
 return (Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2)));  
 }  
  
 public void onVector(int a, int b)  
 {  
 Console.WriteLine( "X: " + (a + x));  
 Console.WriteLine( "Y: " + b + y );  
 }  
  
 public void setPointXY(int x, int y)  
 {  
 this.x = x;  
 this.y = y;  
 }  
  
 public void XYMultiplyBy(int z)  
 {  
 x \*= z;  
 y \*= z;  
 }  
 };  
  
 internal class Program  
 {  
 static void Main(string[] args)  
 {  
 Point p1 = new Point(7, 8);  
 Point p2 = new Point(5, 6);  
 Point p3 = new Point(0, 0);  
  
 Point[] points = new Point[]{p1, p2, p3};  
 SoapFormatter formatter = new SoapFormatter();  
 // получаем поток, куда будем записывать сериализованный объект  
 using (FileStream fs = new FileStream("points.soap", FileMode.**OpenOrCreate**))  
 {  
 formatter.Serialize(fs, points);  
   
 Console.WriteLine("Объект сериализован");  
 }  
  
 int i = 1;  
 // десериализация  
 using (FileStream fs = new FileStream("points.soap", FileMode.**OpenOrCreate**))  
 {  
 Point[] newPoints = (Point[]) formatter.Deserialize(fs);  
  
 Console.WriteLine("Объект десериализован");  
 foreach (Point p in newPoints)  
 {  
 Console.WriteLine("Element {0}: --- X: {1}, Y: {2}", i++, p.X, p.Y);  
 }  
 }  
 }  
 }  
}

**Результат:**



**Висновки**: отже, на цій лабораторній роботі я навчився, одержав практичні навички створення контейнерів і використання їх у програмах С#/C++. Виконав Серіалізацію та десеріалізацію (C#).