Лабораторна робота №7
Шаблони функцій і класів.Контейнери. STL. Колекції в C#
Серіалізація і Десеріалізація

Мета. Одержати практичні навички створення шаблонів і використання їх у програмах C++, роботи з серіалізацією та десеріалізацією.Робота з колекціями в C#.

Хід роботи

Завдання 1

У програмі №1 виконати наступне:

1.Створити об'єкт-контейнер відповідно до варіанта завдання і заповнити його даними, тип яких визначається варіантом завдання.

2.Переглянути контейнер.

3.Змінити контейнер, видаливши з нього одні елементи і замінивши інші.

4.Переглянути контейнер, використовуючи для доступу до його елементів ітератори.

Код програми

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


//multiset   vector int


int main()
{
	std::multiset<int> mySet = { 5, 3, 8, 1, 3, 7, 9, 5 };
	std::vector<int> myVector = { 5, 2, 7, 4, 6, 2, 4, 7, 9, 0, 4, 2 };

	std::cout << "------------------ Show containers ----------------------" <<
std::endl;
	std::cout << "Multiset elements: ";
	for (const auto& elem : mySet)
		std::cout << elem << " ";

	std::cout << std::endl;
	std::cout << "Vector elements: ";
	for (const auto& elem : myVector)
		std::cout << elem << " ";

	std::cout << std::endl;
	std::cout << "-----------------------------------------------------------" <<
std::endl;

	std::cout << "------------------ Insert elements ----------------------" <<
std::endl;
	mySet.insert(4);
	myVector.push_back(6);
```

					ЛР.ОК.19.ПІ231.20.07	
Змін.	Аркуш	№ докум.	Підпис	Дата		
Розробив		Дар'єв Д.О.			Літ	Аркуш 1 / Аркушів 1
Перевірив						
Н.контр.					ХПК	
Затвер.						

```
	std::cout << "Multiset elements: ";
	for (const auto& elem : mySet)
		std::cout << elem << " ";

	std::cout << std::endl;
	std::cout << "Vector elements: ";
	for (const auto& elem : myVector)
		std::cout << elem << " ";

	std::cout << std::endl;

	std::cout << "----------------- Delete elements ----------------------" <<
std::endl;

	mySet.erase(3);
	myVector.erase(std::remove(myVector.begin(), myVector.end(), 2));

	std::cout << "Multiset elements: ";
	for (auto it = mySet.cbegin(); it != mySet.cend(); ++it)
		std::cout << *it << " ";

	std::cout << std::endl;
	std::cout << "Vector elements: ";
	for (auto it = myVector.cbegin(); it != myVector.cend(); ++it)
		std::cout << *it << " ";

	std::cout << std::endl;

}
```

Результат

```
----------------- Show containers ----------------------
Multiset elements: 1 3 3 5 5 7 8 9
Vector elements: 5 2 7 4 6 2 4 7 9 0 4 2
--------------------------------------------------------
----------------- Insert elements ----------------------
Multiset elements: 1 3 3 4 5 5 7 8 9
Vector elements: 5 2 7 4 6 2 4 7 9 0 4 2 6
----------------- Delete elements ----------------------
Multiset elements: 1 4 5 5 7 8 9
Vector elements: 5 7 4 6 4 7 9 0 4 6 2 6
```

Завдання 2

У програмі №2 виконати наступне:

1.Створити контейнер, що містить об'єкти типу користувача. Тип 1 контейнера вибирається відповідно до варіанта завдання.

2.Відсортувати його по спаданню елементів.

3.Переглянути контейнер.

4.Використовуючи придатний алгоритм, знайти в контейнері елемент, що задовольняє заданій умові.

5.Перемістити елементи, що задовольняють заданій умові в інший (попередньо порожній) контейнер. Тип другого контейнера визначається варіантом завдання.

6.Переглянути другий контейнер.

7.Відсортувати перший і другий контейнери по зростанню елементів.

8.Переглянути їх.

9.ЗАПИСАТИ ОБЄКТИ У ФАЙЛ ТА ПРОЧИТАТИ З НЬОГО.

Код програми

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
#include <list>
#include <string>
#include <fstream>

class Product {
	std::string name;
	std::string code;
	int count;
public:
	Product(const std::string& name, const std::string& code, int count)
		: name(name), code(code), count(count) {
	}
	const std::string& getName() const { return name; }

	const std::string& getCode() const { return code; }

	int getCount() const { return count; }

	bool operator<(const Product& other) const {
		return code < other.code;
	}
};


int main()
{
	std::cout << "---------------------------------- Create multiset and show -
------------------------------" << std::endl;
	std::multiset<Product> productsSet;
	productsSet.insert(Product{ "Yay", "Y232", 50 });
	productsSet.insert(Product{ "Banana", "B456", 30 });
	productsSet.insert(Product{ "Orange", "O789", 20 });
	productsSet.insert(Product{ "Cocos", "C555", 25 });
	productsSet.insert(Product{ "Apple", "A123", 15 });

	std::cout << "Products in multiset (sorted by code):" << std::endl;
	for (auto it : productsSet) {
		std::cout << "Name: " << it.getName() << ", Code: " << it.getCode() <<
", Count: " << it.getCount() << std::endl;
	}

	std::cout << "---------------------------------- Sort product by descending
--------------------------------" << std::endl;
```

```
	std::vector<Product> productsVector(productsSet.begin(), productsSet.end());

	std::sort(productsVector.begin(), productsVector.end(), [](const Product& a,
const Product& b) { return a.getName() > b.getName(); });

	std::cout << "Sorted products in vector (descending by name) :" << std::endl;
	for (auto it : productsVector) {
		std::cout << "Name: " << it.getName() << ", Code: " << it.getCode() <<
", Count: " << it.getCount() << std::endl;
	}

	std::cout << "---------------------------------- Find product and move to
list --------------------------------" << std::endl;

	std::list<Product> productsList;


	auto found_it = std::find_first_of(productsVector.begin(),
productsVector.end(), productsSet.begin(), productsSet.end(),
		[](const Product& a, const Product& b) { return a.getName() ==
b.getName(); });

	if (found_it != productsVector.end())
	{
		std::cout << "Found product to move: " << found_it->getName() <<
std::endl;

		productsList.insert(productsList.begin(), *found_it);

		productsVector.erase(found_it);
	}

	std::cout << "Products in list after move:" << std::endl;
	for (auto it : productsList) {
		std::cout << "Name: " << it.getName() << ", Code: " << it.getCode() <<
", Count: " << it.getCount() << std::endl;
	}

	std::cout << "\nProducts left in vector after move:" << std::endl;
	for (auto it : productsVector) {
		std::cout << "Name: " << it.getName() << ", Code: " << it.getCode() <<
", Count: " << it.getCount() << std::endl;
	}
	std::cout << "------------------------------- Write and read from file ----
----------------------------" << std::endl;

	std::string filename = "products.txt";
	std::ofstream outFile(filename);

	if (!outFile.is_open()) {
		std::cerr << "ERROR: Cannot open file" << std::endl;
		return 1;
	}
	std::cout << "Writing products to file: " << filename << std::endl;
	for (auto it : productsSet) {
		outFile << it.getName() << "," << it.getCode() << "," << it.getCount()
<< std::endl;
	}
```

```
	std::cout << "Writing completed." << std::endl;
	outFile.close();
	std::ifstream inFile(filename);
	if (!inFile.is_open()) {
		std::cerr << "ERROR: Cannot open file" << std::endl;
		return 1;
	}
	std::multiset<Product> productsFromFile;
	std::string line;

	std::cout << "Reading products from file: " << filename << std::endl;
	while (std::getline(inFile, line)) {
		size_t pos1 = line.find(',');
		size_t pos2 = line.find(',', pos1 + 1);
		std::string name = line.substr(0, pos1);
		std::string code = line.substr(pos1 + 1, pos2 - pos1 - 1);
		int count = std::stoi(line.substr(pos2 + 1));
		productsFromFile.insert(Product{ name, code, count });
	}
	std::for_each(productsFromFile.begin(), productsFromFile.end(), [](const
Product& it) {
		std::cout << "Name: " << it.getName() << ", Code: " << it.getCode() <<
", Count: " << it.getCount() << std::endl;
		});


	std::cout << "----------------------------------- End ----------------------
----------" << std::endl;



	return 0;
}
```

Результат

```
----------------------------------- Create multiset and show ------------
Products in multiset (sorted by code):
Name: Apple, Code: A123, Count: 15
Name: Banana, Code: B456, Count: 30
Name: Cocos, Code: C555, Count: 25
Name: Orange, Code: O789, Count: 20
Name: Yay, Code: Y232, Count: 50
----------------------------------- Sort product by descending ----------
Sorted products in vector (descending by name) :
Name: Yay, Code: Y232, Count: 50
Name: Orange, Code: O789, Count: 20
Name: Cocos, Code: C555, Count: 25
Name: Banana, Code: B456, Count: 30
Name: Apple, Code: A123, Count: 15
----------------------------------- Find product and move to list -------
Found product to move: Yay
Products in list after move:
Name: Yay, Code: Y232, Count: 50

Products left in vector after move:
Name: Orange, Code: O789, Count: 20
Name: Cocos, Code: C555, Count: 25
Name: Banana, Code: B456, Count: 30
Name: Apple, Code: A123, Count: 15
---------------------------------- Write and read from file ---------------
Writing products to file: products.txt
Writing completed.
Reading products from file: products.txt
Name: Apple, Code: A123, Count: 15
Name: Banana, Code: B456, Count: 30
Name: Cocos, Code: C555, Count: 25
Name: Orange, Code: O789, Count: 20
Name: Yay, Code: Y232, Count: 50
----------------------------------- End ----------------------------------
```

Завдання 3

Використати колекції в проекті WinForm C#
Код програми(код не весь а там де є колекції)
HistoryManager.cs

```
using LW_Daryev_WinForm_NewEdition.Brush;
using LW_Daryev_WinForm_NewEdition.Shape;
using System.Collections.Generic;

namespace LW_Daryev_WinForm_NewEdition.HisotyManagment
{
    public class HistoryManager
    {
        private readonly Stack<HistoryRecord> undoStack = new();
        private readonly Stack<HistoryRecord> redoStack = new();
        public void SaveState(List<IShapeDraw> shapes, List<IBrushDraw> brushes)
        {
            undoStack.Push(new HistoryRecord(shapes, brushes));
            redoStack.Clear();
        }
        public HistoryRecord Undo()
        {
            if (undoStack.Count <= 1)
                return null;

            var current = undoStack.Pop();
            redoStack.Push(current);

            return undoStack.Peek();
        }
        public HistoryRecord Redo()
        {
            if (redoStack.Count == 0)
                return null;

            var state = redoStack.Pop();
            undoStack.Push(state);

            return state;
        }
    }
}
```

HistoryRecord.cs

```
using LW_Daryev_WinForm_NewEdition.Brush;
using LW_Daryev_WinForm_NewEdition.Shape;

namespace LW_Daryev_WinForm_NewEdition.HisotyManagment
{
    public class HistoryRecord
    {
        public List<IShapeDraw> ShapesSnapshot { get; }
        public List<IBrushDraw> BrushSnapshot { get; }

        public HistoryRecord(List<IShapeDraw> shapes, List<IBrushDraw> brushes)
        {
            ShapesSnapshot = new List<IShapeDraw>(shapes);
            BrushSnapshot = new List<IBrushDraw>(brushes);
        }
    }
```

```
}
```

Висновок

На лабораторній роботі було одержано практичні навички створення шаблонів і використання їх у програмах C++, також робота з серіалізацією та десеріалізацією та робота з колекціями в C#.