Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №7**

По дисциплине: “ООТПиСП”

Тема: “Шаблоны в C++”

Вариант №6

**Выполнил**:

студент 2 курса

группы ПО-7

Комиссаров А.Е.

**Проверилa:**

Хацкевич М.В.

Брест, 2022

**Цель:** создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio и реализация шаблона класса-контейнера.

**Задание:**

**Вариант №6**

Класс – контейнер СПИСОК с ключевыми значениями типа int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора;

+ число – добавляет костанту ко всем элемента вектора;

Пользовательский класс Pair (пара чисел). Пара должна быть представлена двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть определено от второго числа двоеточием.

**Ход работы**

**Содержимое файла list.h:**

#include <iostream>  
using namespace std;  
template <class T>  
class List {  
public: int size; T \*values;  
 List(): size(0), values(nullptr) {};  
 List(int s, T k); List(const List<T>&a);  
 ~List() { delete[] values; size = 0; };  
 int operator()(); List operator+(const T k);  
 List&operator=(const List<T>& a);  
 T& operator[](int i);};

// Constructor with parameters  
template <class T>List<T>::List(int s, T k) {  
 size = s; values = new T[size];  
 for (int i = 0; i < size; i++) {  
 values[i] = k; }}  
// Copying constructor  
template <class T>List<T>::List(const List<T>&a) {  
 size = a.size; values = size > 0 ? new T[size] : nullptr;  
 for (int i = 0; i < size; i++) {  
 values[i] = a.values[i]; }}  
template <class T>List<T>&List<T>:: operator=(const List<T>& a) {  
 if (this == &a) return \*this;  
 size = a.size; if (values != 0) delete[] values;  
 values = size > 0? new T[size]: nullptr;  
 for (int i = 0; i < size; i++) { values[i] = a.values[i]; }  
 return \*this;}  
// Operation of access by index  
template <class T>T&List<T>::operator [](int i) {  
 if (i < 0 || i >= size) throw range\_error("Index out of range!");  
 return values[i];}  
// Operation of adding constant to each list's element  
template <class T>List<T> List<T>::operator+(const T k) {  
 List<T> temp(size, k);//инициализируем временный вектор любым значением  
 for (int i = 0; i < size; i++) {  
 temp.values[i] = values[i] + k;  
 }  
 return temp;  
}  
// Operation for getting size of list  
template <class T>int List<T>::operator ()()  
{ return size;}

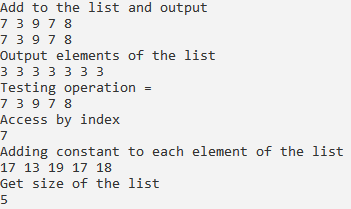
// Operation of output  
template <class T>ostream & operator<< (ostream & out, const List<T>& a)  
{ for (int i = 0; i < a.size ; i++) {  
 out << a.values[i] << " "; }  
 return out;}  
// Operation of input  
template <class T>istream & operator>> (istream&in, List<T>&a)  
{ for (int i = 0; i < a.size; i++) {  
 in >> a.values[i];  
 } return in;}

**Содержимое файла Main.cpp:**

#include "List.h"

int main() {  
 // Add to the list and output  
 cout << "Add to the list and output" << endl;  
 List<int> A(5, 0); cin >> A; cout << A << endl;  
 // Output elements of the list  
 cout << "Output elements of the list" << endl;  
 List<int> B(7, 3); cout << B << endl;  
 // Testing operation =  
 cout << "Testing operation =" << endl;  
 B = A; cout << B << endl;  
 // Access by index  
 cout << "Access by index" << endl; cout << A[3] << endl;  
 // Adding constant to each element of the list  
 cout << "Adding constant to each element of the list" << endl;  
 B = A + 10; cout << B << endl;  
 // Get size of the list  
 cout << "Get size of the list" << endl; cout << B() << endl;}

**Результат работы программы:**



Добавим в проект класс Pair с минимальной функциональностью (конструкторы, деструктор, операция присваивания, операции ввода-вывода). Для этого в файл Pair.h добавляем описание класса:

**Содержимое файла pair.h:**

#include <iostream>

using namespace std;

class Pair {

public:

int val1; double val2;

Pair(); Pair(int, double);

Pair(const Pair&);

Pair& operator=(const Pair&);

virtual ~Pair() {};};

Pair::Pair() { val1 = 0; val2 = 0.0;}

Pair::Pair(int v1, double v2) {

val1 = v1; val2 = v2;}

Pair::Pair(const Pair& t) {

val1 = t.val1; val2 = t.val2;}

Pair& Pair::operator=(const Pair& t) {

val1 = t.val1; val2 = t.val2; return \*this;}

ostream& operator<< (ostream& out, const Pair& t) {

out << t.val1 << ": " << t.val2 << endl; return out;}

istream& operator>> (istream& in, Pair& t) {

cout << "Value 1: "; in >> t.val1; cout << "Value 2: "; in >> t.val2;

return in;}

Протестируем сложение для класса Pair. Для этого фукцию main() добавим операторы:

#include “Pair.h”

int main() {cout << "Object 1: " << endl;  
Pair p1(10, 3.33);cout << p1;  
cout << "Object 2: " << endl;  
Pair p2;cin >> p2;cout << p2;}

**Результат работы программы:**



Выполним тестирование параметризированного класса List для пользовательского типа данных Pair:

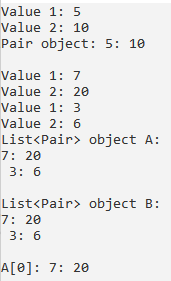
#include “Pair.h”

#include “List.h”

int main() {Pair p1;cin >> p1;cout << "Pair object: " << p1 << endl;  
List<Pair> A(2, p1);cin >> A;cout << "List<Pair> object A:\n" << A << endl;  
List<Pair> B(10, p1);B = A;  
cout << "List<Pair> object B:\n" << B << endl;  
cout << "A[0]: " << A[0];

}

**Результат работы программы:**

****

**Вывод:** В ходе лабораторной работы освоил способы работы с шаблонами, параметризированными классами, реализовал шаблон класса-контейнера.