Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №7**

По дисциплине: “ООТПиСП”

Тема: “Последовательные контейнеры библиотеки STL”

**Вариант №8**

**Выполнил**: студент 2 курса группы ПО-7 Крупенков Михаил Дмитриевич

**Проверил:** Хацкевич М.В.

Брест 2022

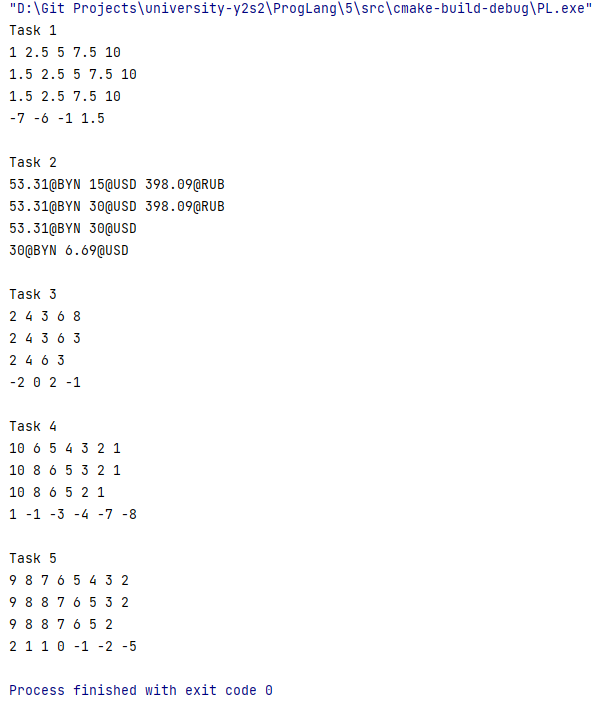
**Цель:**

Изучение контейнеров в языке С++.

**Постановка задачи:**

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | Задача 1   1. Контейнер - список 2. Тип элементов - double   Задача 2  Тип элементов Money (см. лабораторную работу №3).  Задача 3  Параметризированный класс – Вектор (см. лабораторную работу №7)  Задача 4  Адаптер контейнера – очередь c приоритетами.  Задача 5  Параметризированный класс – Вектор  Адаптер контейнера – очередь с приоритетами. | | |
| Задание 3 | Задание 4 | Задание 5 |
| Найти элемент с заданным ключом и добавить его на заданную позицию контейнера | Найти элемент с заданным ключом и удалить его из контейнера | Найти разницу между максимальным и минимальным элементами контейнера и вычесть ее из каждого элемента контейнера |

**Результат**



**Код программы**

**main.cpp**

#include **"list\_functions.cpp"**#include **"Money.h"**#include **"Vector.h"**#include **"priority\_queue\_functions.cpp"  
  
  
int** main() {  
 cout << **"Task 1\n"**;  
 list<**double**> values{1.0, 2.5, 5.0, 7.5, 10.0};  
 print(values);  
 replaceElement(values, 1.0, 1.5);  
 print(values);  
 deleteElement(values, 5.0);  
 print(values);  
 subDiff(values);  
 print(values);  
  
 cout << **"\nTask 2\n"**;  
 list<Money> wallet{Money(53.31, **"BYN"**), Money(15, **"USD"**), Money(398.09, **"RUB"**)};  
 print(wallet);  
 replaceElement(wallet, Money(15, **"USD"**), Money(30, **"USD"**));  
 print(wallet);  
 deleteElement(wallet, wallet.back());  
 print(wallet);  
 subDiff(wallet);  
 print(wallet);  
  
 cout << **"\nTask 3\n"**;  
 Vector<**int**> numbers = {2, 4, 3, 6, 8};  
 numbers.print();  
 numbers[4] = numbers[2];  
 numbers.print();  
 numbers.pop(2);  
 numbers.print();  
 numbers.subDiff();  
 numbers.print();  
  
 cout << **"\nTask 4\n"**;  
 priority\_queue<**int**> pq;  
 vector<**int**> data{3, 5, 1, 4, 6, 2, 10};  
 **for** (**auto** el: data) pq.push(el);  
 print(pq);  
 pq = replaceElement(pq, 4, 8);  
 print(pq);  
 deleteElement(pq, 3);  
 print(pq);  
 subDiff(pq);  
 print(pq);  
  
 cout << **"\nTask 5\n"**;  
 priority\_queue<**int**, Vector<**int**>> pq2;  
 Vector<**int**> data2{5, 6, 4, 7, 3, 8, 2, 9};  
 **for** (**auto** el: data2) pq2.push(el);  
 print(pq2);  
 pq2 = replaceElement(pq2, 4, 8);  
 print(pq2);  
 deleteElement(pq2, 3);  
 print(pq2);  
 subDiff(pq2);  
 print(pq2);  
}

**List\_functions.cpp**

#include **<list>**#include **<iostream>  
  
using namespace** std;  
  
**template**<**class** T>  
**void** print(**const** list<T> &values) {  
 **for** (**auto** el: values) {  
 cout << el << **" "**;  
 }  
 cout << **"\n"**;  
}  
  
**template**<**class** T>  
**int** replaceElement(list<T> &values, **const** T &oldValue, **const** T &newValue) {  
 **for**(**auto** iter = values.begin(); iter != values.end(); iter++) {  
 **if** (\*iter == oldValue) {  
 \*iter = newValue;  
 **return** 0;  
 }  
 }  
 **return** -1;  
}  
  
**template**<**class** T>  
**int** deleteElement(list<T> &values, **const** T &oldValue) {  
 **for**(**auto** iter = values.begin(); iter != values.end(); iter++) {  
 **if** (\*iter == oldValue) {  
 values.erase(iter);  
 **return** 0;  
 }  
 }  
 **return** -1;  
}  
  
**template**<**class** T>  
**void** subDiff(list<T> &values) {  
 **if** (values.empty()) **return**;  
 **auto** iter = values.begin();  
 **auto** minValue = \*iter, maxValue = \*iter;  
 **for** (iter++; iter != values.end(); iter++) {  
 minValue = min(minValue, \*iter);  
 maxValue = max(maxValue, \*iter);  
 }  
 **auto** diff = maxValue - minValue;  
 **for** (iter = values.begin(); iter != values.end(); iter++)  
 \*iter -= diff;  
}

**Money.h**

#ifndef **MONEY\_H**#define **MONEY\_H**#include **<string>**#include **<iostream>  
  
using namespace** std;  
  
**class** Money {  
 **double** value;  
 string currency;  
**public**:  
 Money() : value(0), currency(**"BYN"**) {}  
  
 Money(**double** value, string currency) : value(value), currency(move(currency)) {}  
  
 Money(**const** Money &money) : value(money.value), currency(move(money.currency)) {}  
  
 Money &**operator**=(**const** Money &money) {  
 value = money.value;  
 currency = move(money.currency);  
 **return** \***this**;  
 }  
  
 **bool operator**==(**const** Money &other) **const** {  
 **return** ((value == other.value) && (currency == other.currency));  
 }  
  
  
 Money &**operator**-=(Money &other) {  
 value -= other.value;  
 **return** \***this**;  
 }  
  
 Money &**operator**-(**const** Money &other) **const** {  
 **auto** money = **new** Money(\***this**);  
 money->value -= other.value;  
 **return** \*money;  
 }  
  
 **friend** ostream &**operator**<<(ostream &out, **const** Money &money);  
  
  
 **friend** Money &min(Money &m1, Money &m2);  
  
 **friend** Money &max(Money &m1, Money &m2);  
  
};  
  
ostream &**operator**<<(ostream &out, **const** Money &money) {  
 out << money.value << **"@"** << money.currency;  
 **return** out;  
}  
  
Money &min(Money &m1, Money &m2) {  
 **return** (m1.value < m2.value) ? m1 : m2;  
}  
  
Money &max(Money &m1, Money &m2) {  
 **return** (m1.value > m2.value) ? m1 : m2;  
}  
  
  
#endif *//MONEY\_H*

**Vector.h**

#ifndef **VECTOR\_H**#define **VECTOR\_H**#include **<vector>**#include **<iostream>  
  
using namespace** std;  
  
  
**template**<**class** T>  
**class** Vector {  
 vector<T> \_vector;  
**public**:  
 Vector() {}  
  
 Vector(Vector &v) : \_vector(v.\_vector) {}  
  
 Vector(initializer\_list<T> v) : \_vector(v) {}  
  
 **void** print() {  
 **for** (**auto** el: \_vector) {  
 cout << el << **" "**;  
 }  
 cout << **"\n"**;  
 }  
  
 T &**operator**[](**unsigned** index) {  
 **return** \_vector[index];  
 }  
  
 T &pop(**unsigned** index) {  
 **auto** el = **new** T(\_vector.at(index));  
 \_vector.erase(\_vector.begin() + index);  
 **return** \*el;  
 }  
  
 **void** subDiff() {  
 **if** (\_vector.empty()) **return**;  
 **auto** iter = \_vector.begin();  
 **auto** minValue = \*iter, maxValue = \*iter;  
 **for** (iter++; iter != \_vector.end(); iter++) {  
 minValue = min(minValue, \*iter);  
 maxValue = max(maxValue, \*iter);  
 }  
 **auto** diff = maxValue - minValue;  
 **for** (iter = \_vector.begin(); iter != \_vector.end(); iter++)  
 \*iter -= diff;  
 }  
};  
  
  
#endif *//VECTOR\_H*

**priority\_queue\_functions.cpp**

#ifndef **VECTOR\_H**#define **VECTOR\_H**#include **<vector>**#include **<iostream>  
  
using namespace** std;  
  
  
**template**<**class** T>  
**class** Vector {  
 vector<T> \_vector;  
**public**:  
 Vector() {}  
  
 Vector(Vector &v) : \_vector(v.\_vector) {}  
  
 Vector(initializer\_list<T> v) : \_vector(v) {}  
  
 **void** print() {  
 **for** (**auto** el: \_vector) {  
 cout << el << **" "**;  
 }  
 cout << **"\n"**;  
 }  
  
 T &**operator**[](**unsigned** index) {  
 **return** \_vector[index];  
 }  
  
 T &pop(**unsigned** index) {  
 **auto** el = **new** T(\_vector.at(index));  
 \_vector.erase(\_vector.begin() + index);  
 **return** \*el;  
 }  
  
 **void** subDiff() {  
 **if** (\_vector.empty()) **return**;  
 **auto** iter = \_vector.begin();  
 **auto** minValue = \*iter, maxValue = \*iter;  
 **for** (iter++; iter != \_vector.end(); iter++) {  
 minValue = min(minValue, \*iter);  
 maxValue = max(maxValue, \*iter);  
 }  
 **auto** diff = maxValue - minValue;  
 **for** (iter = \_vector.begin(); iter != \_vector.end(); iter++)  
 \*iter -= diff;  
 }  
};  
  
  
#endif *//VECTOR\_H*

**Вывод:** было получено преисполнение при работе с контейнерами в С++