Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.11**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Последовательные контейнеры STL

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

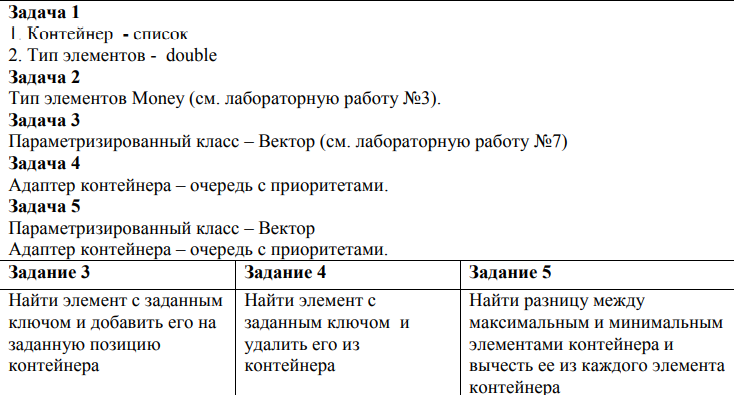
# **Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, использующая последовательные контейнеры библиотеки STL в ОО проограмме.

# **Постановка задачи**

1. Создать последовательный контейнер.
2. Создать параметризированный класс, используя в качестве контейнера последовательный контейнер.
3. Создать адаптер контейнер.
4. Создать параметризированный класс, используя в качестве контейнера адаптер контейнер.
5. Заполнить контейнеры элементами (стандартными, пользовательский тип данных)
6. Добавить элементы в соответствии с заданием.
7. Удалить элементы в соответствии с заданием.
8. Написать демонстрационную программу, иллюстрирующую выполнение задачи.

(8 вариант)



# **Анализ задачи**

1. Определить какие действия предстоит выполнить:

* Разработать класс Money который хранит в себе значения рублей и копеек.
* Организовать методы для ввода данных, селекторы и модификатор, конструкторы и деструкторы, перегрузки операторов.

class Money {

long int rubles;

int penny;

public:

Money();

Money(long int, int);

Money(const Money&);

Money& operator = (const Money&);

int get\_rubles();

int get\_penny();

void set\_rubles(long int);

void set\_penny(int);

bool operator == (const Money&);

bool operator > (const Money&);

bool operator < (const Money&);

Money& operator + (int);

friend istream& operator>> (istream& in, Money&);

friend ostream& operator<< (ostream& out, const Money&);

friend fstream& operator>> (fstream& fin, Money&);

friend fstream& operator<< (fstream& fout, const Money&);

~Money();

};

* Разработать глобальные функции для решения задач 1, 2, 4. В пример возьмем глобальные функции из 1 задачи

List make\_list(int quantity) {

List L;

for (int i = 0; i < quantity; i++) {

double number;

cout << i + 1 << ") "; cin >> number;

L.push\_back(number);

}

return L;

}

void print\_list(List L) {

int lenght = L.size();

cout << "\nСписок:\n";

for (int i = 0; i < lenght; i++) {

cout << L.front() << " ";

L.pop\_front();

}

cout << endl;

}

int find\_element(List L, double element) {

int lenght = L.size(), i = 0;

bool flag = true;

while (i < lenght && flag) {

if (L.front() == element) {

flag = false;

}

L.pop\_front();

i++;

if (i == lenght && flag == true) {

i++;

}

}

return i;

}

void add\_list(List& L, int pos, double element) {

List::iterator it;

it = L.begin();

advance(it, pos - 1);

L.insert(it, element);

}

void del\_list(List& L, int pos) {

List::iterator it;

it = L.begin();

advance(it, pos - 1);

L.erase(it);

}

double max\_list(List L) {

double max = L.front();

int lenght = L.size();

for (int i = 0; i < lenght - 1; i++) {

L.pop\_front();

if (L.front() > max) {

max = L.front();

}

}

return max;

}

double min\_list(List L) {

double min = L.front();

int lenght = L.size();

for (int i = 0; i < lenght - 1; i++) {

L.pop\_front();

if (L.front() < min) {

min = L.front();

}

}

return min;

}

void decrease\_list(List& L, double value) {

int lenght = L.size();

List temp\_L;

for (int i = 0; i < lenght; i++) {

double res = L.front() - value;

temp\_L.push\_back(res);

L.pop\_front();

}

L = temp\_L;

}

* Разработать шаблон контейнер Vector c полями vec и size для решения задачи 3.

template <class T>

class Vector {

private:

vector <T> vec;

int size;

public:

Vector();

Vector(int);

~Vector();

int Size();

void Show();

int Find(T);

T Max();

T Min();

void Add(T, int);

void Dell(int);

void Decrease(T);

};

template <class T>

Vector <T>::Vector() {

size = 0;

}

template <class T>

Vector <T>::~Vector() { }

template <class T>

Vector <T>::Vector(int quantity) {

T value;

for (int i = 0; i < quantity; i++) {

cout << "\n" << i + 1 << ")\n";

cin >> value;

vec.push\_back(value);

}

size = vec.size();

}

template <class T>

int Vector<T>::Size() {

return size;

}

template <class T>

void Vector<T>::Show() {

cout << "\nВектор:\n";

for (int i = 0; i < vec.size(); i++) {

cout << vec[i] << " ";

}

cout << endl;

}

template <class T>

int Vector<T>::Find(T element) {

int i = 0;

bool flag = true;

while (i < vec.size() && flag) {

if (vec[i] == element) {

flag = false;

}

i++;

if (i == vec.size() && flag == true) {

i++;

}

}

return i;

}

template <class T>

void Vector<T>::Add(T element, int pos) {

vec.insert(vec.begin() + pos, element);

size++;

}

template <class T>

void Vector<T>::Dell(int pos) {

vec.erase(vec.begin() + pos - 1);

size--;

}

template <class T>

T Vector<T>::Max() {

T max = vec[0];

for (int i = 1; i < vec.size(); i++) {

if (vec[i] > max) {

max = vec[i];

}

}

return max;

}

template <class T>

T Vector<T>::Min() {

T min = vec[0];

for (int i = 1; i < size; i++) {

if (vec[i] < min) {

min = vec[i];

}

}

return min;

}

template <class T>

void Vector<T>::Decrease(T value) {

for (int i = 0; i < size; i++) {

vec[i] - value;

}

}

* Разработать шаблон контейнер на основе STL библиотеки вектор Pririty\_queque c полями pque и size для решения задачи 5.

template <class T>

class Vector {

private:

priority\_queue <T> pque;

int size;

public:

Vector();

Vector(int);

~Vector();

int Size();

void Show();

int Find(T);

T Max();

T Min();

void Add(T);

void Dell(int);

void Decrease(T);

};

template <class T>

Vector <T>::Vector() {

size = 0;

}

template <class T>

Vector <T>::~Vector() { }

template <class T>

Vector <T>::Vector(int quantity) {

T value;

for (int i = 0; i < quantity; i++) {

cout << "\n" << i + 1 << ")\n";

cin >> value;

pque.push(value);

}

size = pque.size();

}

template <class T>

int Vector<T>::Size() {

return size;

}

template <class T>

void Vector<T>::Show() {

priority\_queue<T> temp\_pque;

cout << "\nВектор:\n";

while (!pque.empty()) {

cout << pque.top() << " ";

temp\_pque.push(pque.top());

pque.pop();

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

cout << endl;

}

template <class T>

int Vector<T>::Find(T element) {

priority\_queue<T> temp\_pque;

int i = 0;

bool flag = true;

while (!pque.empty() && flag) {

if (pque.top() == element) {

flag = false;

}

temp\_pque.push(pque.top());

pque.pop();

i++;

if (i == size && flag == true) {

i++;

}

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

return i;

}

template <class T>

void Vector<T>::Add(T element) {

pque.push(element);

size++;

}

template <class T>

void Vector<T>::Dell(int pos) {

priority\_queue<T> temp\_pque;

int i = 0;

while (!pque.empty()) {

if (i == pos - 1) {

pque.pop();

}

else {

temp\_pque.push(pque.top());

pque.pop();

}

i++;

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

size--;

}

template <class T>

T Vector<T>::Max() {

return pque.top();

}

template <class T>

T Vector<T>::Min() {

priority\_queue<T> temp\_pque;

for (int i = 0; i < size - 1; i++) {

temp\_pque.push(pque.top());

pque.pop();

}

T min = pque.top();

temp\_pque.push(pque.top());

pque.pop();

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

return min;

}

template <class T>

void Vector<T>::Decrease(T value) {

priority\_queue<T> temp\_pque;

int i = 0;

while (!pque.empty()) {

T res = pque.top();

res - value;

temp\_pque.push(res);

pque.pop();

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

}

1. C какими типами данных предстояло работать:

* Для хранения кол-ва рублей реализована переменная rubles типа long int.

long int rubles;

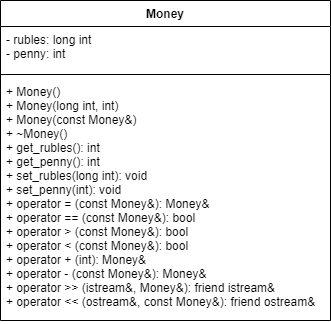
* Для хранения кол-ва копеек реализована переменная penny типа int.

int penny;

* Для хранения длин векторов реализована переменная size типа int.

int size;

# **UML диаграмма**



# **Код программы на языке C++**

Задача №1

#include <list>

#include <iostream>

using namespace std;

typedef list<double> List;

List make\_list(int quantity) {

List L;

for (int i = 0; i < quantity; i++) {

double number;

cout << i + 1 << ") "; cin >> number;

L.push\_back(number);

}

return L;

}

void print\_list(List L) {

int lenght = L.size();

cout << "\nСписок:\n";

for (int i = 0; i < lenght; i++) {

cout << L.front() << " ";

L.pop\_front();

}

cout << endl;

}

int find\_element(List L, double element) {

int lenght = L.size(), i = 0;

bool flag = true;

while (i < lenght && flag) {

if (L.front() == element) {

flag = false;

}

L.pop\_front();

i++;

if (i == lenght && flag == true) {

i++;

}

}

return i;

}

void add\_list(List& L, int pos, double element) {

List::iterator it;

it = L.begin();

advance(it, pos - 1);

L.insert(it, element);

}

void del\_list(List& L, int pos) {

List::iterator it;

it = L.begin();

advance(it, pos - 1);

L.erase(it);

}

double max\_list(List L) {

double max = L.front();

int lenght = L.size();

for (int i = 0; i < lenght - 1; i++) {

L.pop\_front();

if (L.front() > max) {

max = L.front();

}

}

return max;

}

double min\_list(List L) {

double min = L.front();

int lenght = L.size();

for (int i = 0; i < lenght - 1; i++) {

L.pop\_front();

if (L.front() < min) {

min = L.front();

}

}

return min;

}

void decrease\_list(List& L, double value) {

int lenght = L.size();

List temp\_L;

for (int i = 0; i < lenght; i++) {

double res = L.front() - value;

temp\_L.push\_back(res);

L.pop\_front();

}

L = temp\_L;

}

void main() {

system("chcp 1251");

List list;

List::iterator it\_list = list.begin();

int quantity;

cout << "\nCоздание и заполнение списка\n\nКол-во элементов в списке > ";

do {

cin >> quantity;

if (quantity < 1) {

cout << "\nВведите положительное кол-во элемнетов > ";

}

} while (quantity < 1);

list = make\_list(quantity);

print\_list(list);

double key;

cout << "\nДобавление существующего элемента списка\n\nИскомый элемент > "; cin >> key;

int pos = find\_element(list, key);

try{

if (pos == list.size() + 1) {

cout << "\nЭлемент в списке не найден!\n";

}

else {

cout << "\nЭлеменет в списке найден!\n\nПозиция нового элемента в списке > "; cin >> pos;

if (pos > list.size()) {

throw 1.0;

}

add\_list(list, pos, key);

}

}

catch (double) {

cout << "\nОшибка индекса\n";

}

print\_list(list);

cout << "\nУдаление элементов из списка\n\nИскомый элемент > "; cin >> key;

pos = find\_element(list, key);

if (pos == list.size() + 1) {

cout << "\nЭлемент в списке не найден!\n";

}

else {

cout << "\nЭлеменет в списке найден!\n";

del\_list(list, pos);

}

print\_list(list);

cout << "\nУменьшение всех элеменетов списка\n";

double max = max\_list(list), min = min\_list(list), res = max - min;

cout << "\nMax элемент списка = " << max << "\nMin элемент списка = " << min << "\nРазность Max и Min = " << res << endl;

decrease\_list(list, res);

print\_list(list);

system("pause");

}

Задача №2

**Money.h**

#pragma once

#include <iostream>

using namespace std;

class Money {

private:

long int rubles;

int penny;

public:

Money();

Money(long int, int);

Money(const Money&);

~Money();

Money& operator=(const Money&);

long int get\_rubles();

int get\_penny();

void set\_rubles(long int);

void set\_penny(int);

bool operator==(const Money&) const;

bool operator>(const Money&) const;

bool operator<(const Money&) const;

Money& operator-(const Money&);

friend istream& operator>>(istream&, Money&);

friend ostream& operator<<(ostream&, const Money&);

};

**Money.cpp**

#include "Money.h"

Money::Money() {

rubles = 0;

penny = 0;

}

Money::Money(long int r, int p) {

rubles = r;

penny = p;

}

Money::Money(const Money& temp\_Money) {

rubles = temp\_Money.rubles;

penny = temp\_Money.penny;

}

Money::~Money() { }

Money& Money::operator=(const Money& temp\_Money) {

if (\*this == temp\_Money) {

return \*this;

}

rubles = temp\_Money.rubles;

penny = temp\_Money.penny;

return \*this;

}

long int Money::get\_rubles() {

return rubles;

}

int Money::get\_penny() {

return penny;

}

void Money::set\_rubles(long int r) {

rubles = r;

}

void Money::set\_penny(int p) {

penny = p;

}

bool Money::operator==(const Money& temp\_Money) const{

return (rubles == temp\_Money.rubles && penny == temp\_Money.penny);

}

bool Money::operator>(const Money& temp\_Money) const{

return (rubles \* 100 + penny > temp\_Money.rubles \* 100 + temp\_Money.penny);

}

bool Money::operator<(const Money& temp\_Money) const{

return (rubles \* 100 + penny < temp\_Money.rubles \* 100 + temp\_Money.penny);

}

Money& Money::operator-(const Money& temp\_Money) {

long int sum1 = rubles \* 100 + penny;

long int sum2 = temp\_Money.rubles \* 100 + temp\_Money.penny;

sum1 -= sum2;

rubles = sum1 / 100;

if (sum1 < 0) {

sum1 \*= -1;

}

penny = sum1 % 100;

return \*this;

}

istream& operator>> (istream& in, Money& temp\_Money) {

do {

cout << "Рубли: "; in >> temp\_Money.rubles;

} while (temp\_Money.rubles < 0);

do {

cout << "Копейки: "; in >> temp\_Money.penny;

} while (temp\_Money.penny > 99 || temp\_Money.penny < 0);

return in;

}

ostream& operator << (ostream& out, const Money& temp\_Money) {

if (temp\_Money.penny < 10) {

out << temp\_Money.rubles << ",0" << temp\_Money.penny;

}

else {

out << temp\_Money.rubles << "," << temp\_Money.penny;

}

return out;

}

**Main**

#include <list>

#include <iostream>

#include "Money.h"

using namespace std;

typedef list<Money> List;

List make\_list(int quantity) {

List L;

for (int i = 0; i < quantity; i++) {

Money number;

cout << i + 1 << ")\n"; cin >> number;

L.push\_back(number);

}

return L;

}

void print\_list(List L) {

int lenght = L.size();

cout << "\nСписок:\n";

for (int i = 0; i < lenght; i++) {

cout << L.front() << " ";

L.pop\_front();

}

cout << endl;

}

int find\_element(List L, const Money element) {

int lenght = L.size(), i = 0;

bool flag = true;

while (i < lenght && flag) {

if (L.front() == element) {

flag = false;

}

L.pop\_front();

i++;

if (i == lenght && flag == true) {

i++;

}

}

return i;

}

void add\_list(List& L, int pos, const Money element) {

List::iterator it;

it = L.begin();

advance(it, pos - 1);

L.insert(it, element);

}

void del\_list(List& L, int pos) {

List::iterator it;

it = L.begin();

advance(it, pos - 1);

L.erase(it);

}

Money max\_list(List L) {

Money max = L.front();

int lenght = L.size();

for (int i = 0; i < lenght - 1; i++) {

L.pop\_front();

if (L.front() > max) {

max = L.front();

}

}

return max;

}

Money min\_list(List L) {

Money min = L.front();

int lenght = L.size();

for (int i = 0; i < lenght - 1; i++) {

L.pop\_front();

if (L.front() < min) {

min = L.front();

}

}

return min;

}

void decrease\_list(List& L, Money value) {

int lenght = L.size();

List temp\_L;

for (int i = 0; i < lenght; i++) {

Money res = L.front() - value;

temp\_L.push\_back(res);

L.pop\_front();

}

L = temp\_L;

}

void main() {

system("chcp 1251");

List list;

List::iterator it\_list = list.begin();

int quantity;

cout << "\nCоздание и заполнение списка\n\nКол-во элементов в списке > ";

do {

cin >> quantity;

if (quantity < 1) {

cout << "\nВведите положительное кол-во элемнетов > ";

}

} while (quantity < 1);

list = make\_list(quantity);

print\_list(list);

Money key;

cout << "\nДобавление существующего элемента списка\n\nИскомый элемент >\n"; cin >> key;

int pos = find\_element(list, key);

try {

if (pos == list.size() + 1) {

cout << "\nЭлемент в списке не найден!\n";

}

else {

cout << "\nЭлеменет в списке найден!\n\nПозиция нового элемента в списке > "; cin >> pos;

if (pos > list.size()) {

throw 1.0;

}

add\_list(list, pos, key);

print\_list(list);

}

}

catch (double) {

cout << "\nОшибка индекса\n";

}

cout << "\nУдаление элементов из списка\n\nИскомый элемент >\n"; cin >> key;

pos = find\_element(list, key);

if (pos == list.size() + 1) {

cout << "\nЭлемент в списке не найден!\n";

}

else {

cout << "\nЭлеменет в списке найден!\n";

del\_list(list, pos);

print\_list(list);

}

cout << "\nУменьшение всех элеменетов списка\n";

Money max = max\_list(list), min = min\_list(list), res = max;

res - min;

cout << "\nMax элемент списка = " << max << "\nMin элемент списка = " << min << "\nРазность Max и Min = " << res << endl;

decrease\_list(list, res);

print\_list(list);

system("pause");

}

Задача №3

**Vector.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

template <class T>

class Vector {

private:

vector <T> vec;

int size;

public:

Vector();

Vector(int);

~Vector();

int Size();

void Show();

int Find(T);

T Max();

T Min();

void Add(T, int);

void Dell(int);

void Decrease(T);

};

template <class T>

Vector <T>::Vector() {

size = 0;

}

template <class T>

Vector <T>::~Vector() { }

template <class T>

Vector <T>::Vector(int quantity) {

T value;

for (int i = 0; i < quantity; i++) {

cout << "\n" << i + 1 << ")\n";

cin >> value;

vec.push\_back(value);

}

size = vec.size();

}

template <class T>

int Vector<T>::Size() {

return size;

}

template <class T>

void Vector<T>::Show() {

cout << "\nВектор:\n";

for (int i = 0; i < vec.size(); i++) {

cout << vec[i] << " ";

}

cout << endl;

}

template <class T>

int Vector<T>::Find(T element) {

int i = 0;

bool flag = true;

while (i < vec.size() && flag) {

if (vec[i] == element) {

flag = false;

}

i++;

if (i == vec.size() && flag == true) {

i++;

}

}

return i;

}

template <class T>

void Vector<T>::Add(T element, int pos) {

vec.insert(vec.begin() + pos, element);

size++;

}

template <class T>

void Vector<T>::Dell(int pos) {

vec.erase(vec.begin() + pos - 1);

size--;

}

template <class T>

T Vector<T>::Max() {

T max = vec[0];

for (int i = 1; i < vec.size(); i++) {

if (vec[i] > max) {

max = vec[i];

}

}

return max;

}

template <class T>

T Vector<T>::Min() {

T min = vec[0];

for (int i = 1; i < size; i++) {

if (vec[i] < min) {

min = vec[i];

}

}

return min;

}

template <class T>

void Vector<T>::Decrease(T value) {

for (int i = 0; i < size; i++) {

vec[i] - value;

}

}

**Main**

#include <C:\Users\apicd\OneDrive\Рабочий стол\Labs\Lab 18 (Class)\Lab 18.11 (Sequential STL)\Lab 18.11 (Task 2)\Money.cpp>

#include <iostream>

#include "Vector.h"

using namespace std;

void main() {

system("chcp 1251");

Vector<Money> vec;

int quantity;

Money element;

cout << "\nCоздание и заполнение вектора\n\nКол-во элементов в списке > "; cin >> quantity;

for (int i = 0; i < quantity; i++) {

cout << i + 1 << ")\n"; cin >> element;

vec.Add(element, i);

}

vec.Show();

system("pause");

cout << "\nДобавление существующего элемента вектора\n\nИскомый элемент >\n"; cin >> element;

int pos = vec.Find(element);

if (pos == vec.Size() + 1) {

cout << "\nЭлемент в векторе не найден!\n";

}

else {

cout << "\nЭлеменет в векторе найден!\n\nПозиция нового элемента в векторе > "; cin >> pos;

try {

if (pos > vec.Size()) {

throw 1.0;

}

vec.Add(element, pos - 1);

vec.Show();

}

catch (double) {

cout << "\nОшибка индекса\n";

}

}

system("pause");

cout << "\nУдаление элемента из вектора\n\nИскомый элемент >\n"; cin >> element;

pos = vec.Find(element);

if (pos == vec.Size() + 1) {

cout << "\nЭлемент в векторе не найден!\n";

}

else {

cout << "\nЭлеменет в векторе найден!\n";

vec.Dell(pos);

vec.Show();

}

system("pause");

cout << "\nУменьшение всех элеменетов вектора\n";

Money max = vec.Max(), min = vec.Min(), res = max;

res - min;

cout << "\nMax элемент вектора = " << max << "\nMin элемент вектора = " << min << "\nРазность Max и Min = " << res << endl;

vec.Decrease(res);

vec.Show();

system("pause");

}

Задача №4

#include <C:\Users\apicd\OneDrive\Рабочий стол\Labs\Lab 18 (Class)\Lab 18.11 (Sequential STL)\Lab 18.11 (Task 2)\Money.cpp>

#include <iostream>

#include <queue>

using namespace std;

typedef priority\_queue<Money, vector<Money>> Pque;

typedef vector<Money> Vec;

Pque make\_pque(int quantity) {

Pque p;

Money m;

for (int i = 0; i < quantity; i++) {

cout << i + 1 << ")\n";

cin >> m;

p.push(m);

}

return p;

}

void print\_pque(Pque p) {

cout << "\nОчередь с приоритетом:\n";

while (!p.empty()) {

cout << p.top() << " ";

p.pop();

}

cout << endl;

}

int find\_element(Pque p, const Money element) {

int lenght = p.size(), i = 0;

bool flag = true;

while (!p.empty() && flag) {

if (p.top() == element) {

flag = false;

}

p.pop();

i++;

if (i == lenght && flag == true) {

i++;

}

}

return i;

}

void delete\_pque(Pque& p, int pos) {

Vec vec;

int i = 0;

while (!p.empty()) {

if (i == pos) {

p.pop();

}

else {

vec.push\_back(p.top());

p.pop();

}

i++;

}

for (i = 0; i < vec.size(); i++) {

p.push(vec[i]);

}

}

Money min\_pque(Pque p) {

int lenght = p.size();

for (int i = 0; i < lenght - 1; i++) {

p.pop();

}

return p.top();

}

void decrease\_pque(Pque& p, Money value) {

Vec vec;

while (!p.empty()) {

Money res = p.top();

res - value;

p.pop();

vec.push\_back(res);

}

for (int i = 0; i < vec.size(); i++) {

p.push(vec[i]);

}

}

void main() {

system("chcp 1251");

int quantity;

cout << "\nCоздание и заполнение очереди\n\nКол-во элементов в очереди > "; cin >> quantity;

Pque p;

p = make\_pque(quantity);

print\_pque(p);

system("pause");

Money key;

cout << "\nДобавление существующего элемента очереди\n\nИскомый элемент >\n"; cin >> key;

int pos = find\_element(p, key);

if (pos == p.size() + 1) {

cout << "\nЭлемент в списке не найден!\n";

}

else {

cout << "\nЭлеменет в списке найден!\n";

p.push(key);

print\_pque(p);

}

system("pause");

cout << "\nУдаление элементов из списка\n\nИскомый элемент >\n"; cin >> key;

pos = find\_element(p, key);

if (pos == p.size() + 1) {

cout << "\nЭлемент в списке не найден!\n";

}

else {

cout << "\nЭлеменет в списке найден!\n";

delete\_pque(p, pos - 1);

print\_pque(p);

}

system("pause");

cout << "\nУменьшение всех элеменетов списка\n";

Money max = p.top(), min = min\_pque(p), res = max;

res - min;

cout << "\nMax элемент списка = " << max << "\nMin элемент списка = " << min << "\nРазность Max и Min = " << res << endl;

decrease\_pque(p, res);

print\_pque(p);

system("pause");

}

Задача №5

**Vector.h**

#pragma once

#include <vector>

#include <queue>

#include <iostream>

using namespace std;

template <class T>

class Vector {

private:

priority\_queue <T> pque;

int size;

public:

Vector();

Vector(int);

~Vector();

int Size();

void Show();

int Find(T);

T Max();

T Min();

void Add(T);

void Dell(int);

void Decrease(T);

};

template <class T>

Vector <T>::Vector() {

size = 0;

}

template <class T>

Vector <T>::~Vector() { }

template <class T>

Vector <T>::Vector(int quantity) {

T value;

for (int i = 0; i < quantity; i++) {

cout << "\n" << i + 1 << ")\n";

cin >> value;

pque.push(value);

}

size = pque.size();

}

template <class T>

int Vector<T>::Size() {

return size;

}

template <class T>

void Vector<T>::Show() {

priority\_queue<T> temp\_pque;

cout << "\nВектор:\n";

while (!pque.empty()) {

cout << pque.top() << " ";

temp\_pque.push(pque.top());

pque.pop();

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

cout << endl;

}

template <class T>

int Vector<T>::Find(T element) {

priority\_queue<T> temp\_pque;

int i = 0;

bool flag = true;

while (!pque.empty() && flag) {

if (pque.top() == element) {

flag = false;

}

temp\_pque.push(pque.top());

pque.pop();

i++;

if (i == size && flag == true) {

i++;

}

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

return i;

}

template <class T>

void Vector<T>::Add(T element) {

pque.push(element);

size++;

}

template <class T>

void Vector<T>::Dell(int pos) {

priority\_queue<T> temp\_pque;

int i = 0;

while (!pque.empty()) {

if (i == pos - 1) {

pque.pop();

}

else {

temp\_pque.push(pque.top());

pque.pop();

}

i++;

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

size--;

}

template <class T>

T Vector<T>::Max() {

return pque.top();

}

template <class T>

T Vector<T>::Min() {

priority\_queue<T> temp\_pque;

for (int i = 0; i < size - 1; i++) {

temp\_pque.push(pque.top());

pque.pop();

}

T min = pque.top();

temp\_pque.push(pque.top());

pque.pop();

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

return min;

}

template <class T>

void Vector<T>::Decrease(T value) {

priority\_queue<T> temp\_pque;

int i = 0;

while (!pque.empty()) {

T res = pque.top();

res - value;

temp\_pque.push(res);

pque.pop();

}

while (!temp\_pque.empty()) {

pque.push(temp\_pque.top());

temp\_pque.pop();

}

}

**Main**

#include <C:\Users\apicd\OneDrive\Рабочий стол\Labs\Lab 18 (Class)\Lab 18.11 (Sequential STL)\Lab 18.11 (Task 2)\Money.cpp>

#include <iostream>

#include "Vector.h"

using namespace std;

void main() {

system("chcp 1251");

Vector<Money> vec;

int quantity;

Money element;

cout << "\nCоздание и заполнение вектора\n\nКол-во элементов в списке > "; cin >> quantity;

for (int i = 0; i < quantity; i++) {

cout << "\n" << i + 1 << ")\n"; cin >> element;

vec.Add(element);

}

vec.Show();

system("pause");

cout << "\nДобавление существующего элемента вектора\n\nИскомый элемент >\n"; cin >> element;

int pos = vec.Find(element);

if (pos == vec.Size() + 1) {

cout << "\nЭлемент в векторе не найден!\n";

}

else {

cout << "\nЭлеменет в векторе найден!\n";

vec.Add(element);

vec.Show();

}

system("pause");

cout << "\nУдаление элемента из вектора\n\nИскомый элемент >\n"; cin >> element;

pos = vec.Find(element);

if (pos == vec.Size() + 1) {

cout << "\nЭлемент в векторе не найден!\n";

}

else {

cout << "\nЭлеменет в векторе найден!\n";

vec.Dell(pos);

vec.Show();

}

system("pause");

cout << "\nУменьшение всех элеменетов вектора\n";

Money max = vec.Max(), min = vec.Min(), res = max;

res - min;

cout << "\nMax элемент вектора = " << max << "\nMin элемент вектора = " << min << "\nРазность Max и Min = " << res << endl;

vec.Decrease(res);

vec.Show();

system("pause");

}

# **Скриншоты тестов**

