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 №18.12**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Ассоциативные контейнеры

Вариант 8

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

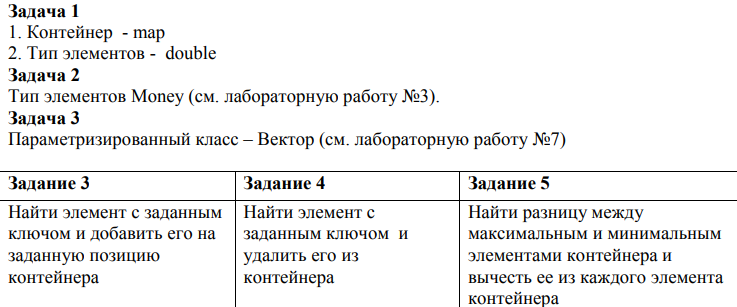
# **Цель работы**

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

# **Постановка задачи**

1. Создать ассоциативный контейнер.
2. Создать параметризированный класс, используя в качестве контейнера ассоциативный 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. В пример возьмем глобальные функции из 1 задачи

TMap make\_map(int quantity) {

TMap map;

cout << endl;

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

double el;

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

map.insert(make\_pair(i, el));

}

return map;

}

void print\_map(TMap map) {

cout << "\nСловарь:\n";

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

cout << i + 1 << " : " << map[i] << " " << endl;

}

cout << endl;

}

int find\_element(TMap map, double key) {

int i = 0;

bool flag = true;

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

if (map[i] == key) {

flag = false;

}

i++;

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

i++;

}

}

return i;

}

void add\_element(TMap& map, int pos, double key) {

TMap temp\_map;

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

temp\_map.insert(make\_pair(i, map[i]));

}

temp\_map.insert(make\_pair(pos, key));

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

temp\_map.insert(make\_pair(i, map[i - 1]));

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

void del\_element(TMap& map, int pos) {

TMap temp\_map;

int lenght = map.size();

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

temp\_map.insert(make\_pair(i, map[i]));

}

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

temp\_map.insert(make\_pair(i-1, map[i]));

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

double max\_map(TMap map) {

double max = map[0];

int lenght = map.size();

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

if (map[i] > max) {

max = map[i];

}

}

return max;

}

double min\_map(TMap map) {

double min = map[0];

int lenght = map.size();

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

if (map[i] < min) {

min = map[i];

}

}

return min;

}

void decrease\_map(TMap& map, double value) {

int lenght = map.size();

TMap temp\_map;

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

double res = map[i] - value;

temp\_map[i] = res;

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

* Разработать шаблон контейнер на основе ассоциативного контейнера словарь c полями m и lenght для решения задачи 3.

template <class T>

class Container {

private:

map<int, T> m;

int lenght;

public:

Container();

Container(int n);

int Find(T value);

int Size();

void Add(T value, int pos);

void Dell(int pos);

void Print();

T Min();

T Max();

void Decrease(T value);

~Container();

};

template <class T>

Container<T>::Container(){

lenght = 0;

}

template <class T>

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

template <class T>

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

T el;

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

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

m.insert(make\_pair(i, el));

}

lenght = m.size();

}

template <class T>

void Container<T>::Print() {

cout << "\nСловарь:\n"; cout << endl;

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

cout << i + 1 << " : " << m[i] << " " << endl;

}

}

template <class T>

int Container<T>::Find(T value) {

int i = 0;

bool flag = true;

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

if (m[i] == value) {

flag = false;

}

i++;

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

i++;

}

}

return i;

}

template <class T>

int Container<T>::Size() {

return lenght;

}

template <class T>

void Container<T>::Add(T value, int pos) {

map<int, T> temp\_map;

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

temp\_map.insert(make\_pair(i, m[i]));

}

temp\_map.insert(make\_pair(pos, value));

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

temp\_map.insert(make\_pair(i, m[i - 1]));

}

m.clear();

m = temp\_map;

temp\_map.clear();

}

template <class T>

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

map<int, T> temp\_map;

int lenght = m.size();

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

temp\_map.insert(make\_pair(i, m[i]));

}

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

temp\_map.insert(make\_pair(i - 1, m[i]));

}

m.clear();

m = temp\_map;

temp\_map.clear();

}

template <class T>

T Container<T>::Min() {

map<int, Money>::iterator it = m.begin();

T min = (\*it).second;

while (it != m.end()) {

if (min > (\*it).second) {

min = (\*it).second;

}

it++;

}

return min;

}

template <class T>

T Container<T>::Max() {

map<int, Money>::iterator it = m.begin();

T max = (\*it).second;

while (it != m.end()) {

if (max < (\*it).second) {

max = (\*it).second;

}

it++;

}

return max;

}

template <class T>

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

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

m[i] - value;

}

}

1. C какими типами данных предстояло работать:

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

long int rubles;

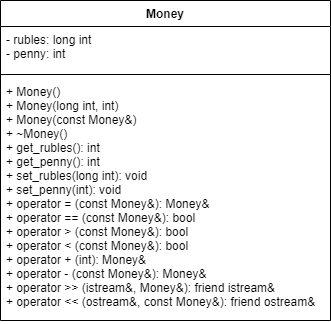
* Для хранения кол-ва копеек реализована переменная penny типа int.

int penny;

* Для хранения длины контейнера реализована переменная lenght типа int.

int lenght;

# **UML диаграмма**



# **Код программы на языке C++**

Задача №1

#include <iostream>

#include <map>

using namespace std;

typedef map<int, double> TMap;

typedef TMap::iterator it;

TMap make\_map(int quantity) {

TMap map;

cout << endl;

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

double el;

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

map.insert(make\_pair(i, el));

}

return map;

}

void print\_map(TMap map) {

cout << "\nСловарь:\n";

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

cout << i + 1 << " : " << map[i] << " " << endl;

}

cout << endl;

}

int find\_element(TMap map, double key) {

int i = 0;

bool flag = true;

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

if (map[i] == key) {

flag = false;

}

i++;

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

i++;

}

}

return i;

}

void add\_element(TMap& map, int pos, double key) {

TMap temp\_map;

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

temp\_map.insert(make\_pair(i, map[i]));

}

temp\_map.insert(make\_pair(pos, key));

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

temp\_map.insert(make\_pair(i, map[i - 1]));

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

void del\_element(TMap& map, int pos) {

TMap temp\_map;

int lenght = map.size();

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

temp\_map.insert(make\_pair(i, map[i]));

}

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

temp\_map.insert(make\_pair(i-1, map[i]));

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

double max\_map(TMap map) {

double max = map[0];

int lenght = map.size();

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

if (map[i] > max) {

max = map[i];

}

}

return max;

}

double min\_map(TMap map) {

double min = map[0];

int lenght = map.size();

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

if (map[i] < min) {

min = map[i];

}

}

return min;

}

void decrease\_map(TMap& map, double value) {

int lenght = map.size();

TMap temp\_map;

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

double res = map[i] - value;

temp\_map[i] = res;

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

void main() {

system("chcp 1251");

int quantity;

cout << "\nCоздание и заполнение словаря\n\nКол-во элементов в словаре > ";

do {

cin >> quantity;

if (quantity < 1) {

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

}

} while (quantity < 1);

TMap map = make\_map(quantity);

print\_map(map);

double key;

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

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

try {

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

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

}

else {

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

if (pos > map.size()) {

throw 1.0;

}

add\_element(map, pos - 1, key);

}

}

catch (double) {

cout << "\nОшибка индекса\n";

}

print\_map(map);

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

pos = find\_element(map, key);

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

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

}

else {

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

del\_element(map, pos - 1);

}

print\_map(map);

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

double max = max\_map(map), min = min\_map(map), res = max - min;

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

decrease\_map(map, res);

print\_map(map);

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 <iostream>

#include <map>

#include "Money.h"

using namespace std;

typedef map<int, Money> TMap;

typedef TMap::iterator it;

TMap make\_map(int quantity) {

TMap map;

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

Money el;

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

map.insert(make\_pair(i, el));

}

return map;

}

void print\_map(TMap map) {

cout << "\nСловарь:\n"; cout << endl;

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

cout << i + 1 << " : " << map[i] << " " << endl;

}

}

int find\_element(TMap map, Money key) {

int i = 0;

bool flag = true;

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

if (map[i] == key) {

flag = false;

}

i++;

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

i++;

}

}

return i;

}

void add\_element(TMap& map, int pos, Money key) {

TMap temp\_map;

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

temp\_map.insert(make\_pair(i, map[i]));

}

temp\_map.insert(make\_pair(pos, key));

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

temp\_map.insert(make\_pair(i, map[i - 1]));

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

void del\_element(TMap& map, int pos) {

TMap temp\_map;

int lenght = map.size();

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

temp\_map.insert(make\_pair(i, map[i]));

}

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

temp\_map.insert(make\_pair(i - 1, map[i]));

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

Money max\_map(TMap map) {

it id = map.begin();

Money max = (\*id).second;

while (id != map.end()) {

if (max < (\*id).second) {

max = (\*id).second;

}

id++;

}

return max;

}

Money min\_map(TMap map) {

it id = map.begin();

Money min = (\*id).second;

while (id != map.end()) {

if (min > (\*id).second) {

min = (\*id).second;

}

id++;

}

return min;

}

void decrease\_map(TMap& map, Money value) {

int lenght = map.size();

TMap temp\_map;

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

Money res = map[i] - value;

temp\_map[i] = res;

}

map.clear();

map = temp\_map;

temp\_map.clear();

}

void main() {

system("chcp 1251");

int quantity;

cout << "\nCоздание и заполнение словаря\n\nКол-во элементов в словаре > ";

do {

cin >> quantity;

if (quantity < 1) {

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

}

} while (quantity < 1);

TMap map = make\_map(quantity);

print\_map(map);

Money key;

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

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

try {

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

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

}

else {

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

if (pos > map.size()) {

throw 1.0;

}

add\_element(map, pos - 1, key);

}

}

catch (double) {

cout << "\nОшибка индекса\n";

}

print\_map(map);

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

pos = find\_element(map, key);

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

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

}

else {

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

del\_element(map, pos - 1);

}

print\_map(map);

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

Money max = max\_map(map), min = min\_map(map), res = max;

res - min;

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

decrease\_map(map, res);

print\_map(map);

system("pause");

}

Задача №3

**Container.h**

#pragma once

#include <C:\Users\apicd\OneDrive\Рабочий стол\Labs\Lab 18 (Class)\Lab 18.12 (Associative STL)\Lab 18.12 (Task 2)\Money.cpp>

#include <iostream>

#include <map>

using namespace std;

template <class T>

class Container {

private:

map<int, T> m;

int lenght;

public:

Container();

Container(int n);

int Find(T value);

int Size();

void Add(T value, int pos);

void Dell(int pos);

void Print();

T Min();

T Max();

void Decrease(T value);

~Container();

};

template <class T>

Container<T>::Container(){

lenght = 0;

}

template <class T>

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

template <class T>

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

T el;

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

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

m.insert(make\_pair(i, el));

}

lenght = m.size();

}

template <class T>

void Container<T>::Print() {

cout << "\nСловарь:\n"; cout << endl;

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

cout << i + 1 << " : " << m[i] << " " << endl;

}

}

template <class T>

int Container<T>::Find(T value) {

int i = 0;

bool flag = true;

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

if (m[i] == value) {

flag = false;

}

i++;

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

i++;

}

}

return i;

}

template <class T>

int Container<T>::Size() {

return lenght;

}

template <class T>

void Container<T>::Add(T value, int pos) {

map<int, T> temp\_map;

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

temp\_map.insert(make\_pair(i, m[i]));

}

temp\_map.insert(make\_pair(pos, value));

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

temp\_map.insert(make\_pair(i, m[i - 1]));

}

m.clear();

m = temp\_map;

temp\_map.clear();

}

template <class T>

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

map<int, T> temp\_map;

int lenght = m.size();

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

temp\_map.insert(make\_pair(i, m[i]));

}

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

temp\_map.insert(make\_pair(i - 1, m[i]));

}

m.clear();

m = temp\_map;

temp\_map.clear();

}

template <class T>

T Container<T>::Min() {

map<int, Money>::iterator it = m.begin();

T min = (\*it).second;

while (it != m.end()) {

if (min > (\*it).second) {

min = (\*it).second;

}

it++;

}

return min;

}

template <class T>

T Container<T>::Max() {

map<int, Money>::iterator it = m.begin();

T max = (\*it).second;

while (it != m.end()) {

if (max < (\*it).second) {

max = (\*it).second;

}

it++;

}

return max;

}

template <class T>

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

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

m[i] - value;

}

}

**Main**

#include <iostream>

#include "Container.h"

using namespace std;

void main() {

system("chcp 1251");

int quantity;

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

Container<Money> vec(quantity);

vec.Print();

system("pause");

Money element;

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

int pos = vec.Find(element);

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

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

}

else {

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

try {

if (pos > vec.Size()) {

throw 1.0;

}

vec.Add(element, pos - 1);

vec.Print();

}

catch (double) {

cout << "\nОшибка индекса\n";

}

}

system("pause");

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

pos = vec.Find(element);

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

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

}

else {

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

vec.Dell(pos-1);

vec.Print();

}

system("pause");

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

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

res - min;

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

vec.Decrease(res);

vec.Print();

system("pause");

}

# **Скриншоты тестов**

