Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.13**

Дисциплина: «Информатика»

Тема: Стандартные обобщенные алгоритмы библиотеки STL

Вариант 7

Выполнил:

Студент группы РИС-20-1б

Нечаев Д.А.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель задачи**

* 1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
  2. Использование последовательных контейнеров библиотеки STL в ОО программе.

Для достижения поставленной цели необходимо решить следующие задачи:

* Проанализировать теоретические сведения о ООП
* Абстрактные типы данных. Контейнеры
* Реализовать алгоритм поставленной задачи
* Учесть все исключительные ситуации на ввод элементов
* Создать отдельные файлы: .cpp для описания методов класса, .cpp для описания главной функции, .h для описания класса

Постановка задачи

**Задача 1**

1. Контейнер - вектор
2. Тип элементов Money (см. лабораторную работу №3).

**Задача 2**

Адаптер контейнера - очередь.

**Задача 3**

Ассоциативный контейнер - словарь

Анализ задачи

1. Определить какие операции должны быть выполнены по заданию:

* Создание класса

class Money

class Vector

* Создание группы сумм, заполнение класса
* Создание методов получения элемента по номеру, печати, конструкторов
* Отчистка памяти

1. Для решения задачи используются переменные:

Несколько переменных целочисленного значения для ввода чисел и реализации меню

int s, in, menu = 4;

Указатель для работы с динамической памятью

1. Ввод данных осуществляется посредством функции

cin >> rub;

Вывод данных осуществляется посредством функции

cout << "Искомый элемент = " << a.give(in);

1. Создание метода класса , который печатает сумму:

void Money::Print()

{

cout << "\nРубли - " << rub << ", копейки - " << cop;

1. Создание методов класса для нахождения размера и доступа по индексу

int give(int index);

int givesize();

Создание контейнера

Money::Money(int r, long int c)

{

rub = r;

cop = c;

double co = c;

sum = r + co / 100;

}

1. Создание инструмента обработки исключительных ситуаций. Для этого была использована логика try:

try

{

cout << "Искомый элемент: " << a.give(in);

}

catch (const int in)

{

cout << "Ошибка ввода";

}

}

Код

Main.cpp:

#include <iostream>

#include "CLASS.h"

#include <algorithm>

#include <functional>

#include <vector>

#include <queue>

#include <list>

#include <deque>

#include <map>

#include <windows.h>

#include <string>

using namespace std;

void Money()

{

int s;

double el;

cout << "Введи количество элементов: "; cin >> s;

VectorOfMoney Vec(s);

int menu = 1;

while (menu == 1 || menu == 2 || menu == 3 || menu == 4)

{

cout << "Меню: \n\t1. Найти минимум и добавить его в конец\n\t2. Удалить по ключу \n\t3. К каждому элементу добавить сумму максимума и минимума \n\t4. Вывести вектор\n\n\t0. Выход";

cout << endl << "Выбери пункт меню: "; cin >> menu;

if (menu == 1)

{

Vec.pushback(Vec.FindMin());

cout << "Элемент добавлен" << endl;

}

if (menu == 2)

{

cout << "Введи элемент: "; cin >> el;

Vec.del(el);

cout << "Элемент удален" << endl;

}

if (menu == 3)

{

Vec.add2(Vec.FindMax(), Vec.FindMin());

cout << "Изменения применены";

}

if (menu == 4)

{

Vec.Print();

}

}

}

void Queue()

{

int min = 32000; int max = -32000;

list<int> l;

int check = 1, menu = 1;

cout << "Вводи элементы очереди. 0 - конец ввода" << endl;

while (check != 0)

{

cin >> check;

if (check != 0)

l.push\_back(check);

else break;

}

list<int>::iterator iter = l.begin();

while (menu == 1 || menu == 2 || menu == 3 || menu == 4)

{

cout << "Меню: \n\t1. Вывести очередь\n\t2. Найти минимум и добавить в конец \n\t3. Найти элемент и удалить \n\t4. К каждому элементу добавить сумму максимума и минимума \n\n\t0. Выход";

cout << endl << "Выбери пункт меню: "; cin >> menu;

if (menu == 1)

{

for (int i : l)

cout << i << " ";

cout << endl;

}

if (menu == 2)

{

for (iter = l.begin(); iter != l.end(); iter++)

{

if (\*iter < min)

min = \*iter;

}

l.push\_back(min);

cout << "Элемент добавлен" << endl;

}

if (menu == 3)

{

int k = 0;

cout << "Введи элемент: "; cin >> check;

for (iter = l.begin(); iter != l.end(); iter++)

{

if (check == \*iter)

{

l.erase(iter);

break;

}

}

cout << "Элемент удален" << endl;

}

if (menu == 4)

{

min = 32000; max = -32000;

for (iter = l.begin(); iter != l.end(); iter++)

{

if (\*iter < min)

min = \*iter;

if (\*iter > max)

max = \*iter;

}

list<int> l1;

list<int>::iterator iter1 = l1.begin();

for (iter = l.begin(); iter != l.end(); iter++)

{

l1.push\_back(\*iter + max + min);

}

l.clear();

for (iter1 = l1.begin(); iter1 != l1.end(); iter1++)

{

l.push\_back(\*iter1);

}

cout << "Изменения применены " << endl;

}

}

}

void Dictionary()

{

int s, k = 0, menu = 1; string ad;

multimap<int, string> Map;

cout << "Введи количество слов: "; cin >> s;

for (int i = 0; i < s; i++)

{

cout << "Введи слово: " << endl;

cin >> ad;

Map.emplace(i, ad);

}

while (menu == 1 || menu == 2 || menu == 3)

{

cout << "Меню: \n\t1. Вывести все слова \n\t2. Удалить слово по номеру \n\t3. Добавить слово \n\n\t0. Выход";

cout << endl << "Выбери пункт меню: "; cin >> menu;

if (menu == 1)

{

for (auto iter = Map.begin(); iter != Map.end(); iter++)

{

cout << iter->first << ") " << iter->second << "\t";

}

cout << endl;

}

if (menu == 2)

{

cout << "Введи ключ: "; cin >> k;

Map.erase(k);

cout << "Элемент удален" << endl;

s--;

}

if (menu == 3)

{

cout << "Введи слово: "; cin >> ad;

Map.emplace(s + 1, ad);

s++;

cout << "Слово добавлено" << endl;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

int menu = 1;

while (menu == 1 || menu == 2 || menu == 3)

{

cout << "Меню: \n\t1. Money\n\t2. Queue \n\t3. Map \n\n\t0. Выход";

cout << endl << "Выбери пункт меню: "; cin >> menu;

if (menu == 1)

Money();

if (menu == 2)

{

Queue();

}

if (menu == 3)

{

Dictionary();

}

}

}

MHETODS.cpp:

#include <iostream>

#include "CLASS.h"

#include <algorithm>

#include <functional>

using namespace std;

VectorOfMoney::VectorOfMoney(int s)

{

size = s;

data = new double[size];

cout << "Вводи суммы" << endl;

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

{

cin >> data[i];

}

}

void VectorOfMoney::Print()

{

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

cout << data[i] << " ";

cout << endl;

}

double VectorOfMoney::FindMin()

{

double min = 32000;

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

{

if (data[i] < min)

min = data[i];

}

return min;

}

double VectorOfMoney::FindMax()

{

double max = -32000;

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

{

if (data[i] > max)

max = data[i];

}

return max;

}

void VectorOfMoney::pushback(double el)

{

double\* data1;

data1 = new double[size+1];

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

{

data1[i] = data[i];

}

data1[size] = el;

delete[] data;

data = data1;

size++;

}

void VectorOfMoney::del(double el)

{

int k = 0;

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

{

if (data[i] == el)

k++;

}

double\* data1;

data1 = new double[size - k];

int j = 0;

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

{

if (data[i] != el)

data1[j] = data[i];

else

j--;

j++;

}

delete[] data;

data = data1;

size = size - k;

}

void VectorOfMoney::add2(double el1, double el2)

{

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

{

data[i] = data[i] + el1 + el2;

}

}

MONEY.h:

class VectorOfMoney

{

public:

double \*data;

int size;

VectorOfMoney(int s);

void Print();

double FindMin();

double FindMax();

void pushback(double el);

void del(double el);

void add2(double el1, double el2);

};

Блок-схема:

Main.cpp:



MHETODS.cpp:



CLASS.h:



Работа кода



