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 №18.12**

Дисциплина: «Информатика»

Тема: Ассоциативные контейнеры библиотеки STL

Вариант 7

Выполнил:

Студент группы РИС-20-1б

Коваленко Никита Алексеевич

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 - multiset
2. Тип элементов – float

**Задача 2**

Тип элементов Money (см. лабораторную работу №3).

**Задача 3**

Параметризированный класс – Вектор (см. лабораторную работу №7)

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 <vector>

#include <queue>

#include "CLASS.h"

#include <list>

#include <deque>

#include <set>

using namespace std;

void Multi()

{

float check = 1, menu = 1;

float min = 32000, max = -32000;

multiset<float> mSet;

cout << "Вводи элементы дерева. Окончание ввода - 0" << endl;

while (check != 0)

{

cin >> check;

if (check != 0)

mSet.insert(check);

else break;

}

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

{

cout << "Меню: \n\t1. Вывести структуру\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 (auto& item : mSet)

{

cout << item << " ";

}

cout << endl;

}

if (menu == 2)

{

for (auto& item : mSet)

{

if (item < min)

min = item;

}

mSet.insert(min);

cout << "Элемент добавлен" << endl;

}

if (menu == 3)

{

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

if (mSet.find(check) != mSet.end())

{

mSet.erase(check);

cout << "Элемент удален" << endl;

}

else

cout << "Элемент не найден" << endl;

}

if (menu == 4)

{

int i = 0, s = mSet.size();

float \*mas = new float[mSet.size()];

min = 32000;

for (auto& item : mSet)

{

if (item > max) max = item;

if (item < min) min = item;

}

for (auto& item : mSet)

{

mas[i] = item + min + max;

}

mSet.clear();

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

{

check = mas[i];

mSet.insert(check);

}

delete[] mas;

cout << "\nИзменения применены" << endl;

}

}

}

void Moneyy()

{

int s = 0, r = 0, menu = 1, index, k = 0;

double zn, min = 32000, max = -32000;

long int c = 0;

while (s <= 0)

{

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

}

Money\* data;

data = new Money[s];

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

{

cout << "Введи количество рублей: "; cin >> r;

cout << "Введи количество копеек: "; cin >> c; cout << endl;

data[i] = Money(r, c);

}

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

{

cout << "Меню: \n\t1. Вывести все суммы\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 = 0; i < s; i++)

{

cout << data[i].Get() << " ";

}

cout << endl;

}

if (menu == 2)

{

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

{

if (data[i].sum < min)

min = data[i].sum;

}

Money\* data1;

data1 = new Money[s + 1];

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

{

data1[i] = data[i];

}

data1[s].sum = min;

delete[] data;

data = data1;

cout << "Элемент добавлен" << endl;

s++;

}

if (menu == 3)

{

cout << "Введи значение: "; cin >> zn;

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

{

if (data[i].sum == zn)

{

data[i].sum = -1;

k++;

}

}

Money\* data1;

data1 = new Money[s - k];

int j = 0;

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

{

if (data[i].sum != -1)

{

data1[j] = data[i];

j++;

}

}

delete[] data;

data = data1;

s = s - k;

k = 0;

cout << "Элементы удалены" << endl;

}

if (menu == 4)

{

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

{

if (data[i].sum > max) max = data[i].sum;

if (data[i].sum < min) min = data[i].sum;

}

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

data[i].sum = data[i].sum + max + min;

cout << "\nИзменения применены" << endl;

}

}

}

void Vectorrr()

{

int s, in, menu = 4;

cout << "Введи количество сумм "; cin >> s;

Vector a(s);

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

{

cout << "Меню: \n\t1. Вывести вектор\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 = 0; i < a.givesize(); i++)

cout << a.give(i) << " ";

cout << endl;

}

if (menu == 2)

{

a.pushback(a.min());

cout << "Элемент добавлен" << endl;

}

if (menu == 3)

{

int el;

cout << "Введи элемент: "; cin >> el;

for (int i = 0; i < a.givesize(); i++)

{

if (a.give(i) == el)

a.del(i);

}

cout << "Элемент удален" << endl;

}

if (menu == 4)

{

for (int i = 0; i < a.givesize(); i++)

{

a.bred(a.max(), a.min(), i);

}

cout << "Изменения внесены" << endl;

}

if (menu == 0)

{

a.end();

break;

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int menu = 1;

while (menu == 1 || menu == 2 || menu == 3 || menu == 4 || menu == 5)

{

cout << "Меню: \n\t1. multiset\n\t2. Money \n\t3. Параметризированный вектор \n\n\t0. Выход";

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

if (menu == 1)

Multi();

if (menu == 2)

Moneyy();

if (menu == 3)

Vectorrr();

}

}

MHETODS.cpp:

#include <iostream>

#include <sstream>

#include <fstream>

using namespace std;

#include "CLASS.h"

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

{

rub = r;

cop = c;

double co = c;

sum = r + co / 100;

}

double Money::Get()

{

return sum;

}

void Money::Print()

{

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

}

Vector::Vector(int s)

{

size = s;

data = new int[size];

cout << "Введи числа" << endl;

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

{

cin >> data[i];

}

}

int Vector::give(int index)

{

return data[index];

}

int Vector::givesize()

{

return size;

}

void Vector::end()

{

delete[] data;

data = 0;

}

void Vector::pushback(int el)

{

int\* data1;

data1 = new int[size + 1];

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

data1[i] = data[i];

data1[size] = el;

delete[] data;

data = data1;

size++;

}

void Vector::del(int index)

{

int\* data1;

data1 = new int[size - 1];

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

data1[i] = data[i];

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

data1[i-1] = data[i];

delete[] data;

data = data1;

size--;

}

int Vector::min()

{

int min = 32000;

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

{

if (data[i] < min)

min = data[i];

}

return min;

}

int Vector::max()

{

int max = -32000;

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

{

if (data[i] > max)

max = data[i];

}

return max;

}

void Vector::bred(int max, int min, int index)

{

data[index] = data[index] + min + max;

}

MONEY.h:

#include <iostream>

#include <sstream>

#include <fstream>

using namespace std;

class Money

{

public:

long int rub;

int cop;

double sum;

Money()

{

rub = cop = sum = 0;

}

Money(int r, long int c);

double Get();

void Print();

};

class Vector

{

public:

Vector(int s);

int give(int index);

int givesize();

void end();

void pushback(int el);

void del(int index);

int min();

int max();

void bred(int max, int min, int index);

private:

int size;

int\* data;

};

Блок-схема:

Main.cpp:



MHETODS.cpp:



CLASS.h:



Работа кода



