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: Объектно-ориентированное программирование. Стандартные обобщенные алгоритмы библиотеки STL

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

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

**Постановка задачи**

Задача 1.

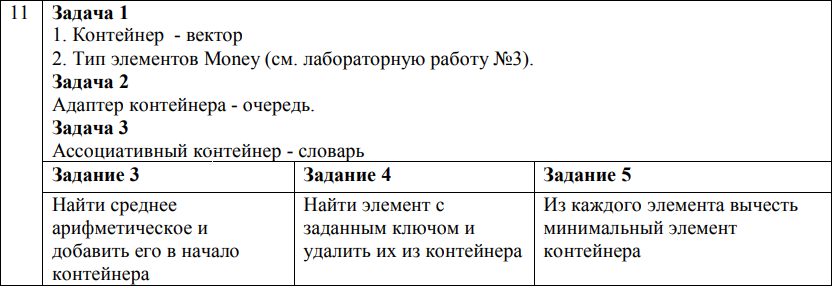
1. Создать последовательный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).
4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).
7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 2.

1. Создать адаптер контейнера.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).
4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

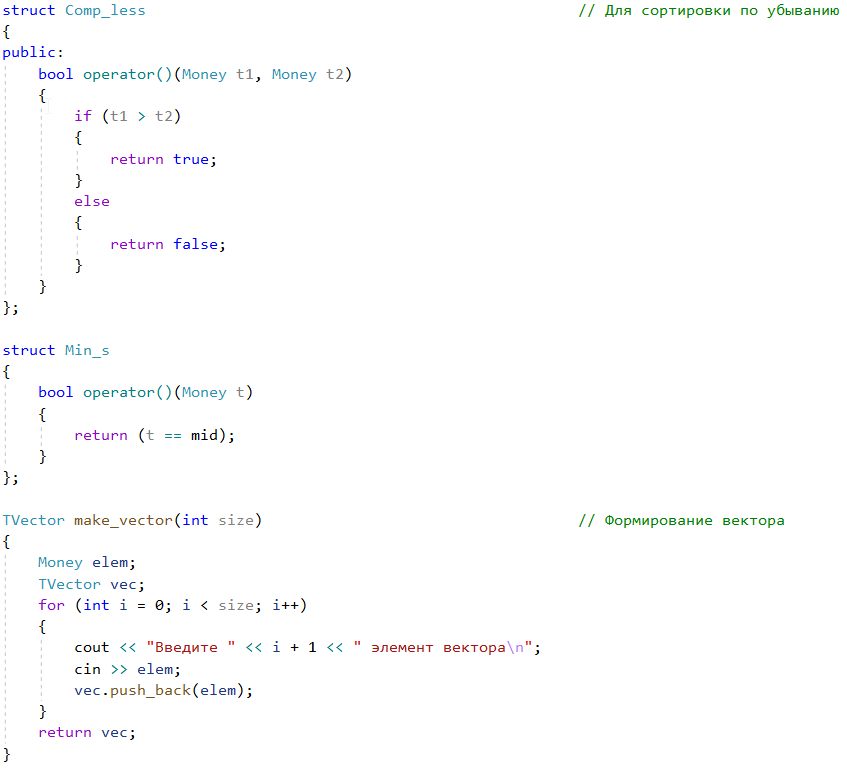
Задача 3

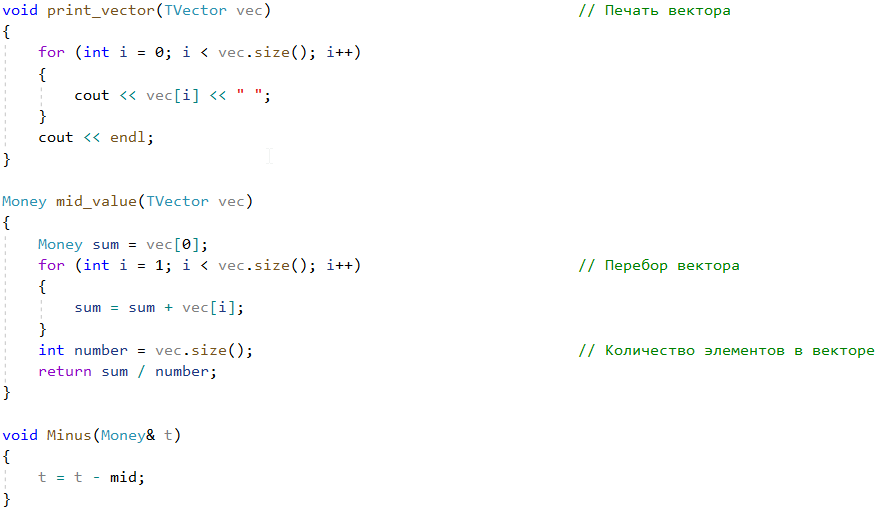
1. Создать ассоциативный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).
4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()).
8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.



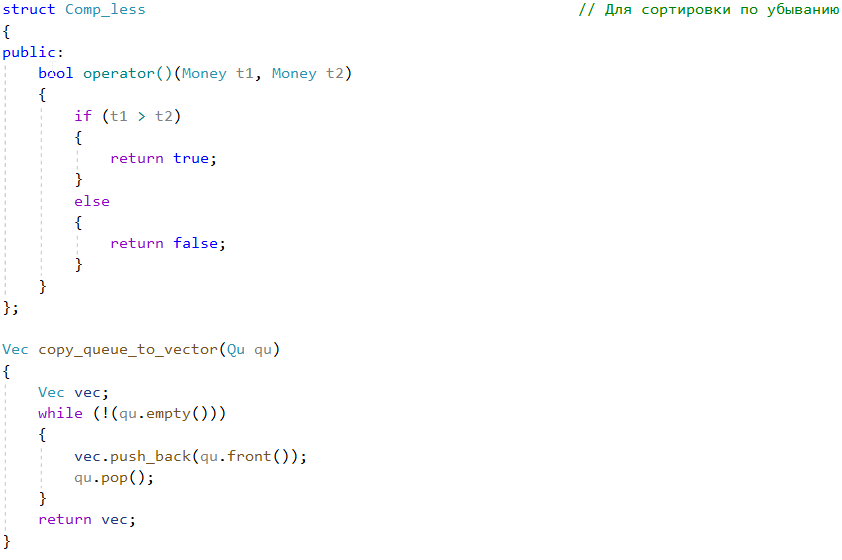
**Анализ задачи**

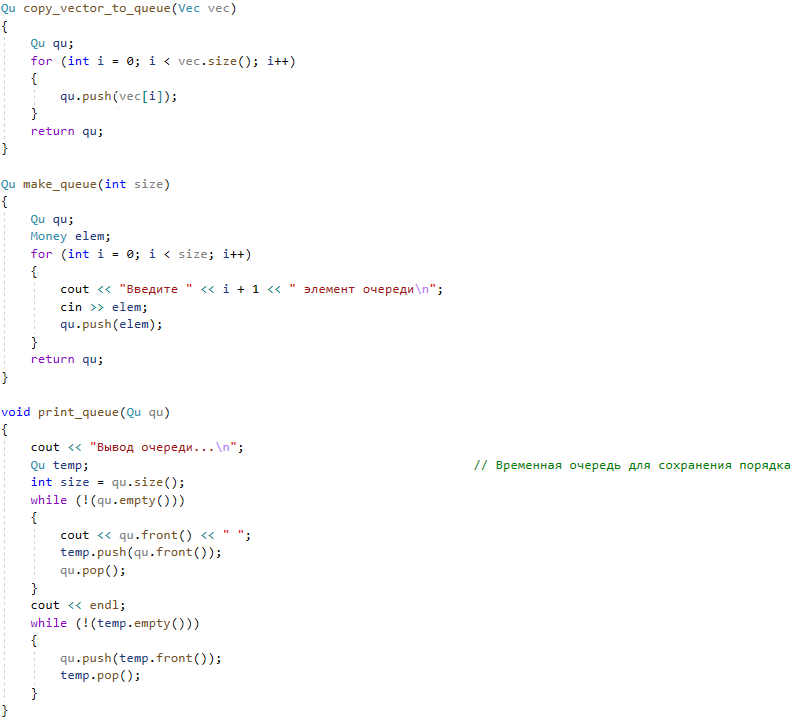
1. Для решения задачи необходимо:
   1. Реализовать определение функций для работы с вектором типа Money: создание вектора, вычисление среднего арифметического, вычитание минимального элемента вектора, добавление среднего арифметического в начало вектора, вывод вектора.

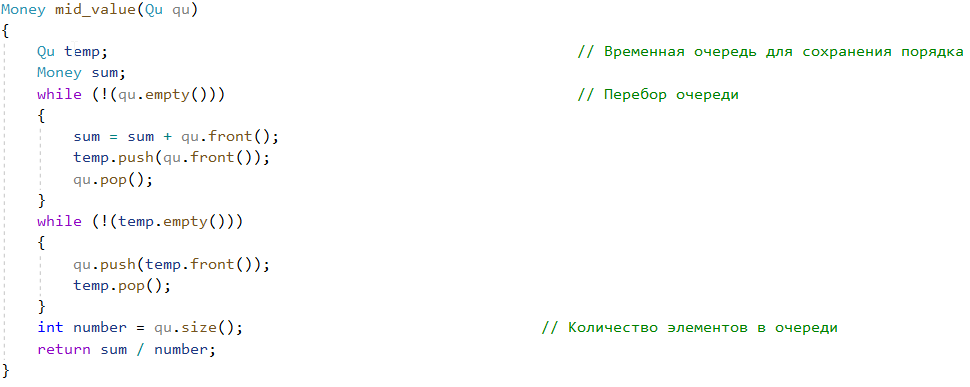


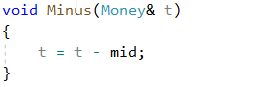


* 1. Реализовать определение функций для работы с адаптером контейнера queue: создание очереди, вычисление среднего арифметического, вычитание минимального элемента очереди, добавление среднего арифметического в начало очереди, вывод очереди.

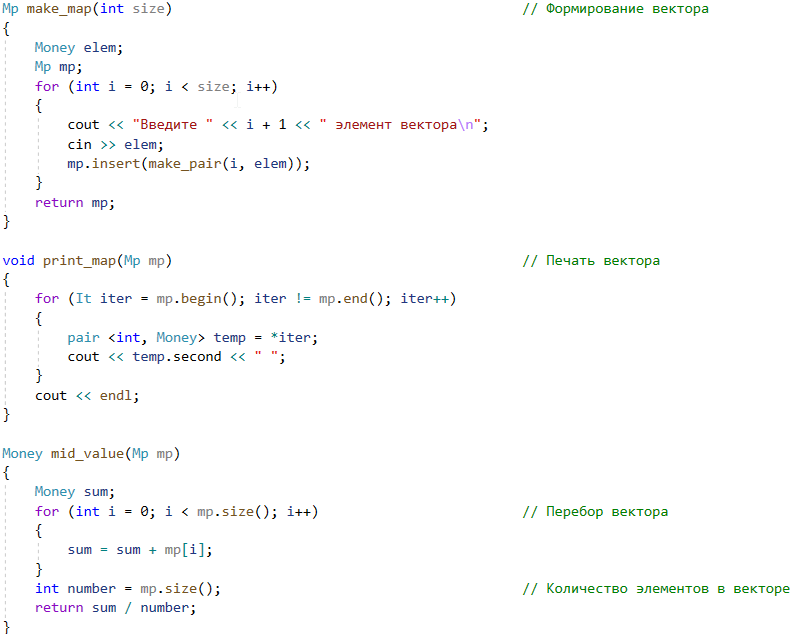


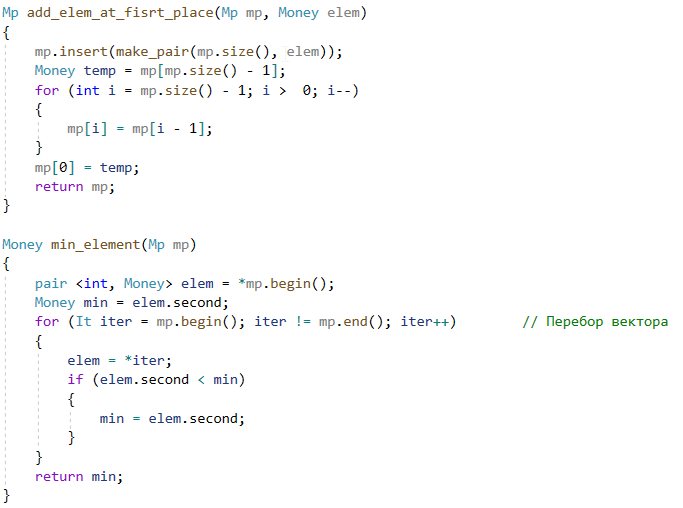




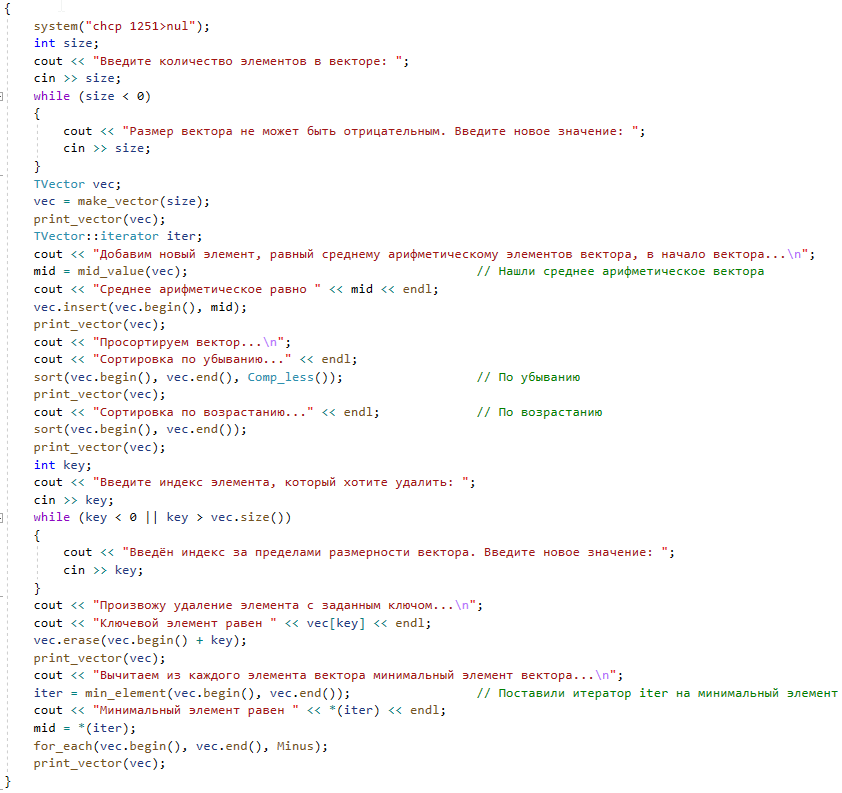


* 1. Реализовать определение функций для работы с адаптером контейнера map: создание словаря, вычисление среднего арифметического, вычитание минимального элемента словаря, добавление среднего арифметического в начало словаря, вывод словаря.

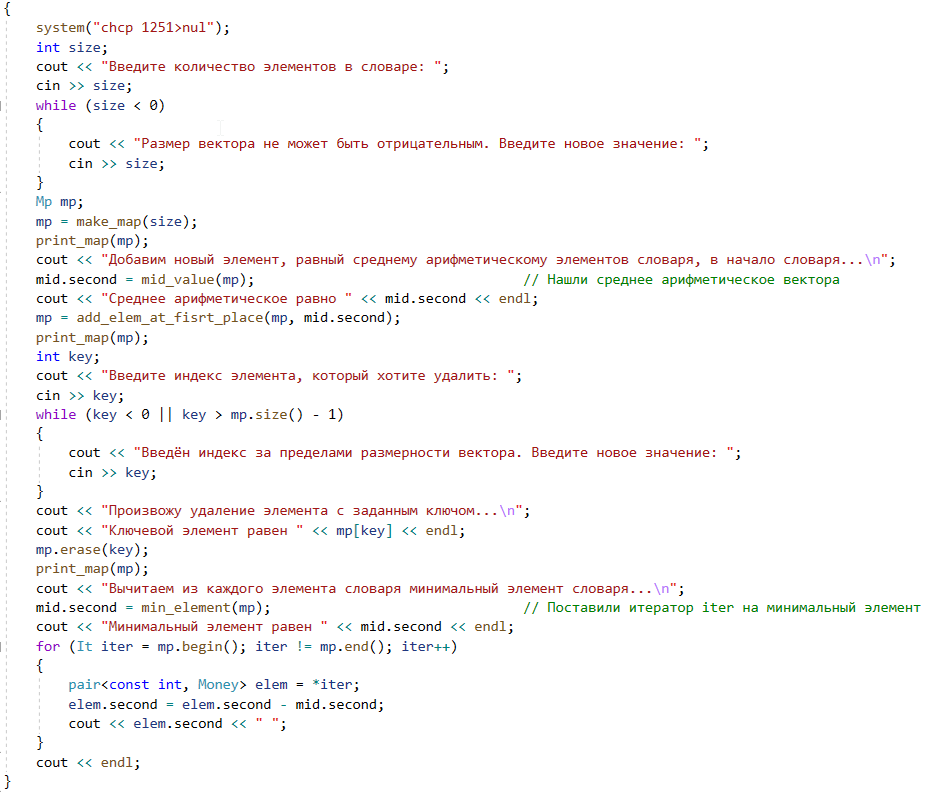




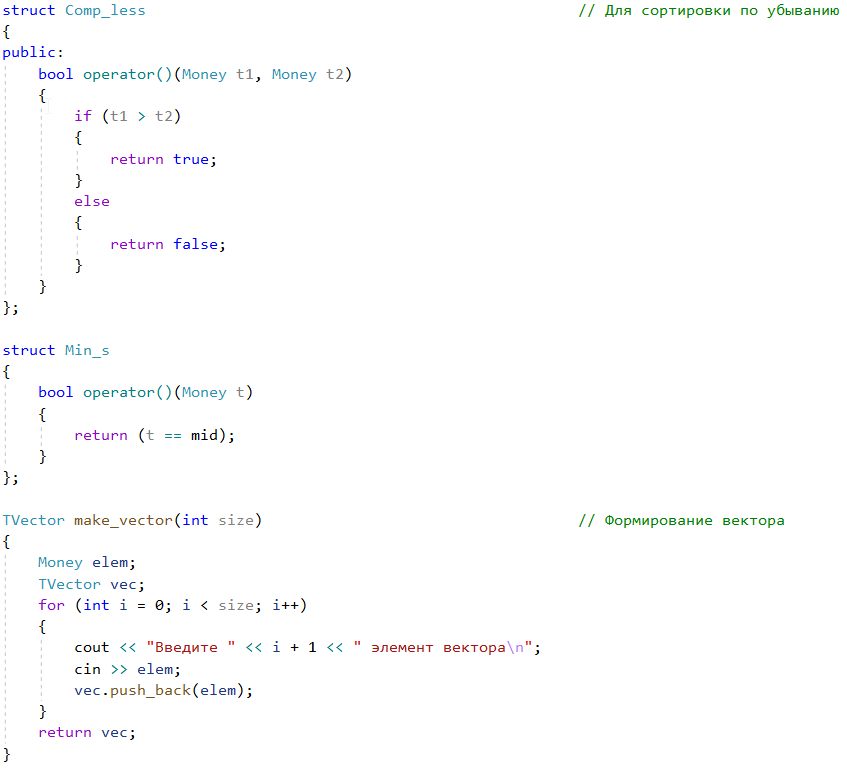
* 1. Реализовать применение этих функций в главной функции.

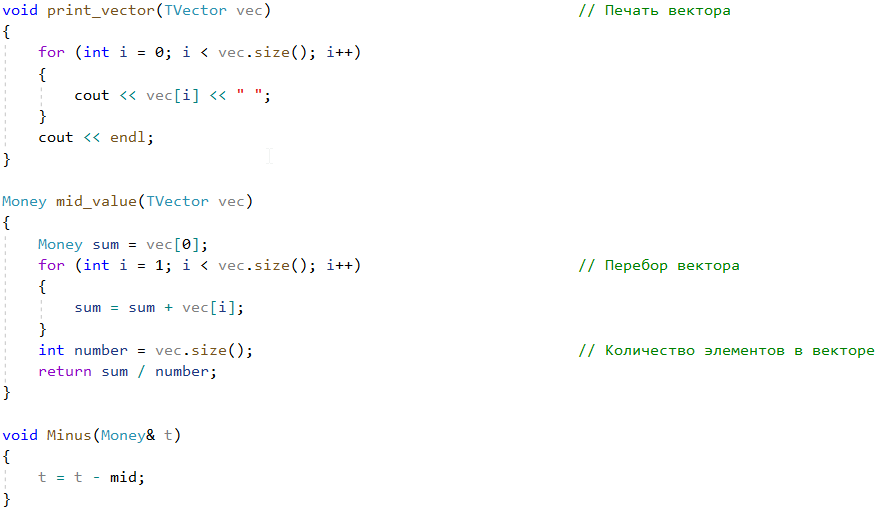




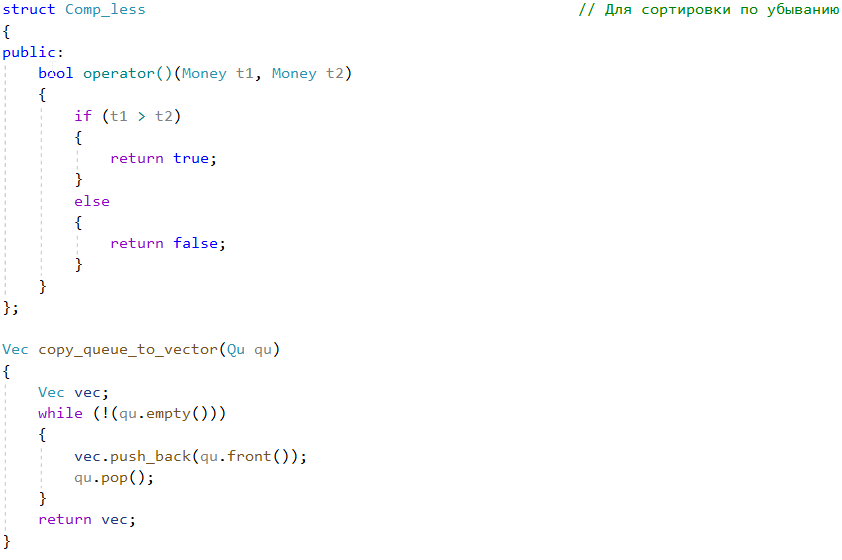


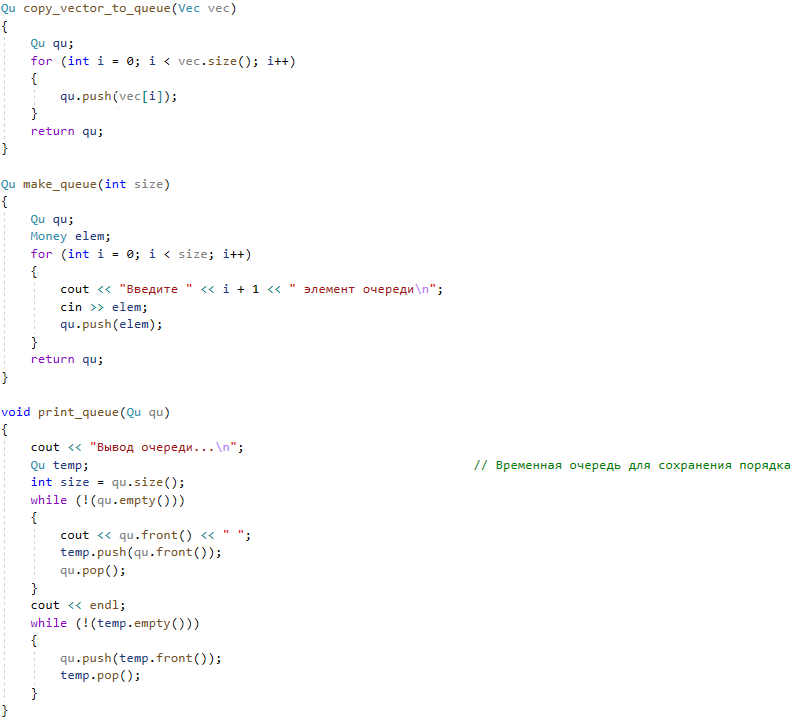
1. Поставленные задачи решены следующими действиями:
   1. В задании 1 реализованы функции для работы с вектором типа Money: создание вектора, вычисление среднего арифметического, вычитание минимального элемента вектора, добавление среднего арифметического в начало вектора, вывод вектора.

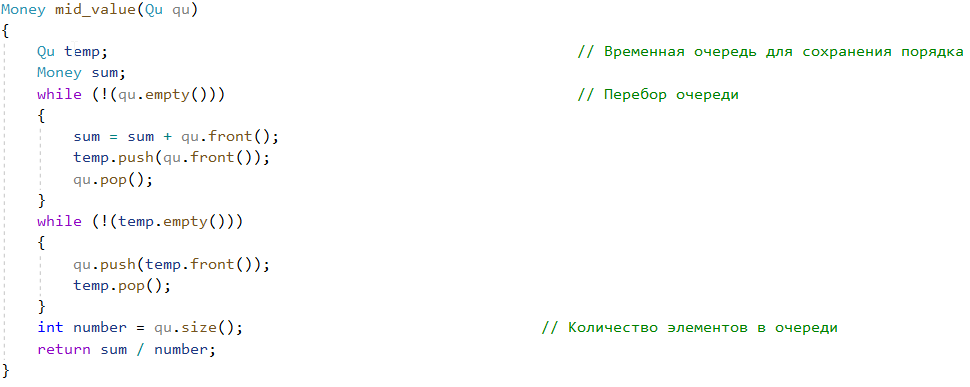


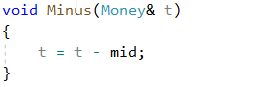


* 1. В задании 2 реализованы функции для работы с адаптером контейнера queue: создание очереди, вычисление среднего арифметического, вычитание минимального элемента очереди, добавление среднего арифметического в начало очереди, вывод очереди.

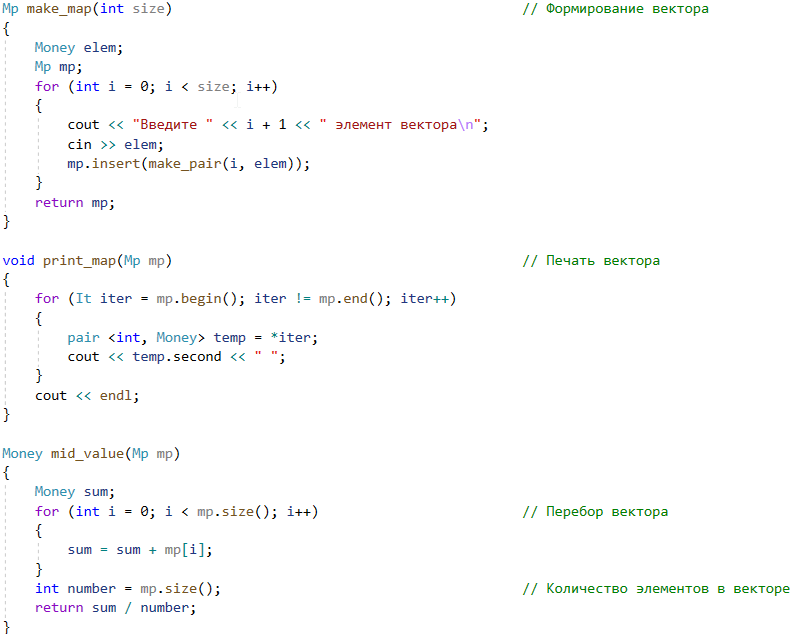


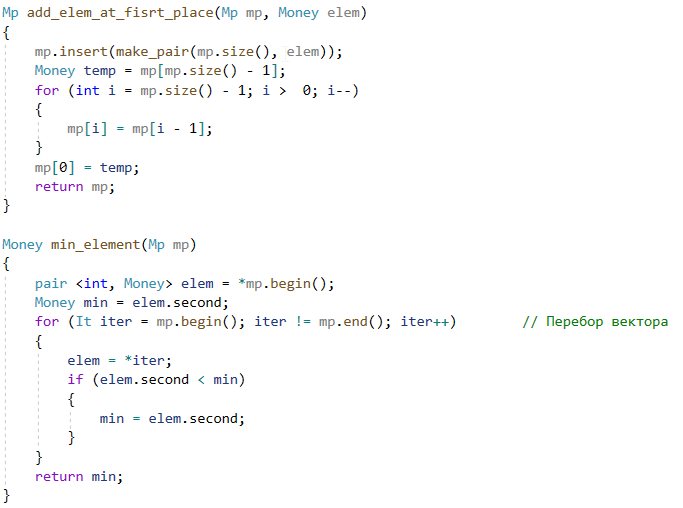






* 1. В задании 3 реализованы функции для работы с адаптером контейнера map: создание словаря, вычисление среднего арифметического, вычитание минимального элемента словаря, добавление среднего арифметического в начало словаря, вывод словаря.





**Реализация задачи на языке С++**

**(1)**

**Файл с главной программой Лабораторная работа № 18.13(1).cpp**

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.h"

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.cpp"

#include <vector>

#include <iostream>

#include <functional>

#include <algorithm>

using namespace std;

typedef vector<Money> TVector;

Money mid;

struct Comp\_less

{

public:

bool operator()(Money t1, Money t2)

{

if (t1 > t2)

{

return true;

}

else

{

return false;

}

}

};

struct Min\_s

{

bool operator()(Money t)

{

return (t == mid);

}

};

TVector make\_vector(int size)

{

Money elem;

TVector vec;

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

{

cout << "Введите " << i + 1 << " элемент вектора\n";

cin >> elem;

vec.push\_back(elem);

}

return vec;

}

void print\_vector(TVector vec)

{

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

{

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

}

cout << endl;

}

Money mid\_value(TVector vec)

{

Money sum = vec[0];

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

{

sum = sum + vec[i];

}

int number = vec.size();

return sum / number;

}

void Minus(Money& t)

{

t = t - mid;

}

int main()

{

system("chcp 1251>nul");

int size;

cout << "Введите количество элементов в векторе: ";

cin >> size;

while (size < 0)

{

cout << "Размер вектора не может быть отрицательным. Введите новое значение: ";

cin >> size;

}

TVector vec;

vec = make\_vector(size);

print\_vector(vec);

TVector::iterator iter;

cout << "Добавим новый элемент, равный среднему арифметическому элементов вектора, в начало вектора...\n";

mid = mid\_value(vec);

cout << "Среднее арифметическое равно " << mid << endl;

vec.insert(vec.begin(), mid);

print\_vector(vec);

cout << "Просортируем вектор...\n";

cout << "Сортировка по убыванию..." << endl;

sort(vec.begin(), vec.end(), Comp\_less());

print\_vector(vec);

cout << "Сортировка по возрастанию..." << endl;

sort(vec.begin(), vec.end());

print\_vector(vec);

int key;

cout << "Введите индекс элемента, который хотите удалить: ";

cin >> key;

while (key < 0 || key > vec.size())

{

cout << "Введён индекс за пределами размерности вектора. Введите новое значение: ";

cin >> key;

}

cout << "Произвожу удаление элемента с заданным ключом...\n";

cout << "Ключевой элемент равен " << vec[key] << endl;

vec.erase(vec.begin() + key);

print\_vector(vec);

cout << "Вычитаем из каждого элемента вектора минимальный элемент вектора...\n";

iter = min\_element(vec.begin(), vec.end());

cout << "Минимальный элемент равен " << \*(iter) << endl;

mid = \*(iter);

for\_each(vec.begin(), vec.end(), Minus);

print\_vector(vec);

}

**(2)**

**Файл с главной программой Лабораторная работа № 18.13(2).cpp**

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.h"

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.cpp"

#include <iostream>

#include <queue>

#include <vector>

#include <algorithm>

using namespace std;

typedef queue<Money> Qu;

typedef vector<Money> Vec;

Money mid;

struct Comp\_less

{

public:

bool operator()(Money t1, Money t2)

{

if (t1 > t2)

{

return true;

}

else

{

return false;

}

}

};

Vec copy\_queue\_to\_vector(Qu qu)

{

Vec vec;

while (!(qu.empty()))

{

vec.push\_back(qu.front());

qu.pop();

}

return vec;

}

Qu copy\_vector\_to\_queue(Vec vec)

{

Qu qu;

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

{

qu.push(vec[i]);

}

return qu;

}

Qu make\_queue(int size)

{

Qu qu;

Money elem;

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

{

cout << "Введите " << i + 1 << " элемент очереди\n";

cin >> elem;

qu.push(elem);

}

return qu;

}

void print\_queue(Qu qu)

{

cout << "Вывод очереди...\n";

Qu temp;

int size = qu.size();

while (!(qu.empty()))

{

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

temp.push(qu.front());

qu.pop();

}

cout << endl;

while (!(temp.empty()))

{

qu.push(temp.front());

temp.pop();

}

}

Money mid\_value(Qu qu)

{

Qu temp;

Money sum;

while (!(qu.empty()))

{

sum = sum + qu.front();

temp.push(qu.front());

qu.pop();

}

while (!(temp.empty()))

{

qu.push(temp.front());

temp.pop();

}

int number = qu.size();

return sum / number;

}

void Minus(Money& t)

{

t = t - mid;

}

int main()

{

system("chcp 1251>nul");

Qu qu;

int size;

cout << "Введите количество элементов в очереди: ";

cin >> size;

while (size < 0)

{

cout << "Размер очереди не может быть отрицательным. Введите новое значение: ";

cin >> size;

}

qu = make\_queue(size);

print\_queue(qu);

Vec vec;

cout << "Добавим новый элемент, равный среднему арифметическому элементов вектора, в начало вектора...\n";

mid = mid\_value(qu);

cout << "Среднее арифметическое равно " << mid << endl;

vec = copy\_queue\_to\_vector(qu);

vec.insert(vec.begin(), mid);

qu = copy\_vector\_to\_queue(vec);

print\_queue(qu);

cout << "Просортируем очередь...\n";

cout << "Сортировка по убыванию..." << endl;

vec = copy\_queue\_to\_vector(qu);

sort(vec.begin(), vec.end(), Comp\_less());

qu = copy\_vector\_to\_queue(vec);

print\_queue(qu);

cout << "Сортировка по возрастанию..." << endl;

sort(vec.begin(), vec.end());

qu = copy\_vector\_to\_queue(vec);

print\_queue(qu);

int key;

cout << "Введите индекс элемента, который хотите удалить: ";

cin >> key;

while (key < 0 || key > vec.size() - 1)

{

cout << "Введён индекс за пределами размерности вектора. Введите новое значение: ";

cin >> key;

}

cout << "Произвожу удаление элемента с заданным ключом...\n";

cout << "Ключевой элемент равен " << vec[key] << endl;

vec.erase(vec.begin() + key);

qu = copy\_vector\_to\_queue(vec);

print\_queue(qu);

cout << "Вычитаем из каждого элемента вектора минимальный элемент вектора...\n";

vector<Money>::iterator iter = min\_element(vec.begin(), vec.end());

cout << "Минимальный элемент равен " << \*(iter) << endl;

mid = \*(iter);

for\_each(vec.begin(), vec.end(), Minus);

qu = copy\_vector\_to\_queue(vec);

print\_queue(qu);

}

**(3)**

**Файл с главной программой Лабораторная работа № 18.13(3).cpp**

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.h"

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.cpp"

#include <map>

#include <iostream>

using namespace std;

typedef map<int, Money> Mp;

typedef map<int, Money>::iterator It;

pair<int, Money> mid;

Mp make\_map(int size)

{

Money elem;

Mp mp;

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

{

cout << "Введите " << i + 1 << " элемент вектора\n";

cin >> elem;

mp.insert(make\_pair(i, elem));

}

return mp;

}

void print\_map(Mp mp)

{

for (It iter = mp.begin(); iter != mp.end(); iter++)

{

pair <int, Money> temp = \*iter;

cout << temp.second << " ";

}

cout << endl;

}

Money mid\_value(Mp mp)

{

Money sum;

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

{

sum = sum + mp[i];

}

int number = mp.size();

return sum / number;

}

Mp add\_elem\_at\_fisrt\_place(Mp mp, Money elem)

{

mp.insert(make\_pair(mp.size(), elem));

Money temp = mp[mp.size() - 1];

for (int i = mp.size() - 1; i > 0; i--)

{

mp[i] = mp[i - 1];

}

mp[0] = temp;

return mp;

}

Money min\_element(Mp mp)

{

pair <int, Money> elem = \*mp.begin();

Money min = elem.second;

for (It iter = mp.begin(); iter != mp.end(); iter++)

{

elem = \*iter;

if (elem.second < min)

{

min = elem.second;

}

}

return min;

}

int main()

{

system("chcp 1251>nul");

int size;

cout << "Введите количество элементов в словаре: ";

cin >> size;

while (size < 0)

{

cout << "Размер вектора не может быть отрицательным. Введите новое значение: ";

cin >> size;

}

Mp mp;

mp = make\_map(size);

print\_map(mp);

cout << "Добавим новый элемент, равный среднему арифметическому элементов словаря, в начало словаря...\n";

mid.second = mid\_value(mp);

cout << "Среднее арифметическое равно " << mid.second << endl;

mp = add\_elem\_at\_fisrt\_place(mp, mid.second);

print\_map(mp);

int key;

cout << "Введите индекс элемента, который хотите удалить: ";

cin >> key;

while (key < 0 || key > mp.size() - 1)

{

cout << "Введён индекс за пределами размерности вектора. Введите новое значение: ";

cin >> key;

}

cout << "Произвожу удаление элемента с заданным ключом...\n";

cout << "Ключевой элемент равен " << mp[key] << endl;

mp.erase(key);

print\_map(mp);

cout << "Вычитаем из каждого элемента словаря минимальный элемент словаря...\n";

mid.second = min\_element(mp);

cout << "Минимальный элемент равен " << mid.second << endl;

for (It iter = mp.begin(); iter != mp.end(); iter++)

{

pair<const int, Money> elem = \*iter;

elem.second = elem.second - mid.second;

cout << elem.second << " ";

}

cout << endl;

}