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 № 18.11**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Последовательные контейнеры 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 стандартного типа (тип указан в варианте).
3. Добавить элементы в соответствии с заданием.
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 2.

1. Создать последовательный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 3

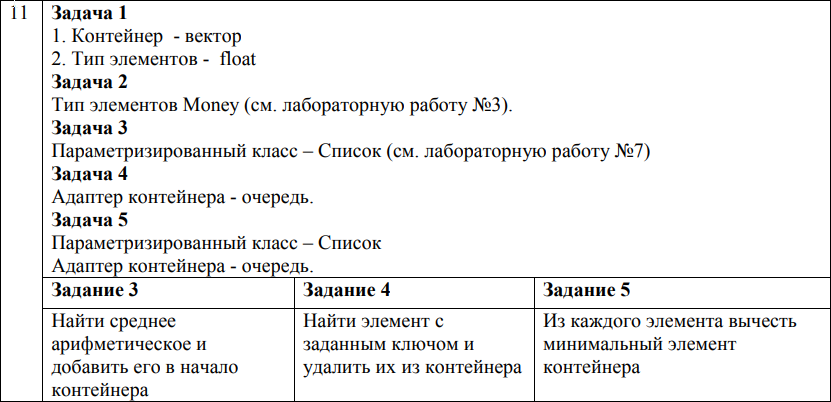
1. Создать параметризированный класс, используя в качестве контейнера последовательный контейнер.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

Задача 4

1. Создать адаптер контейнера.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

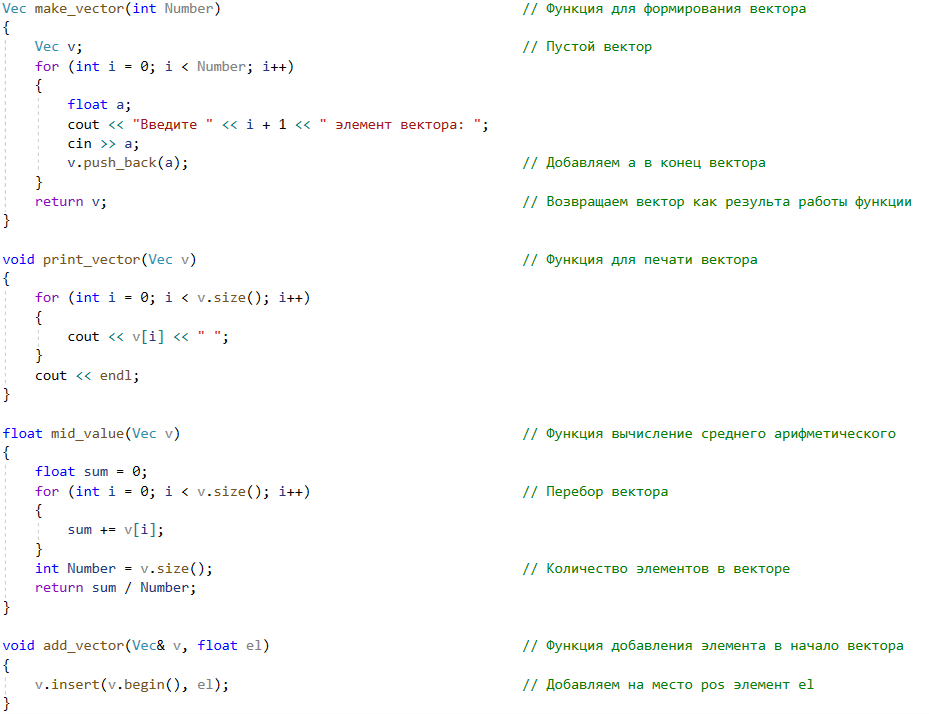
Задача 5

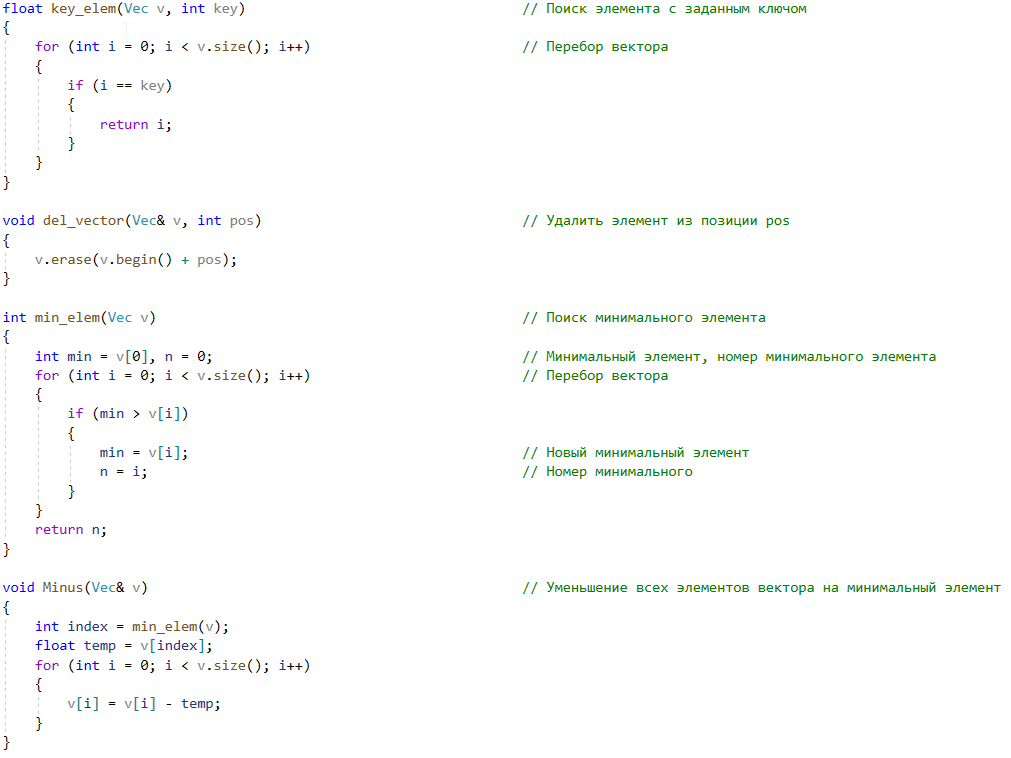
1. Создать параметризированный класс, используя в качестве контейнера адаптер контейнера.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.



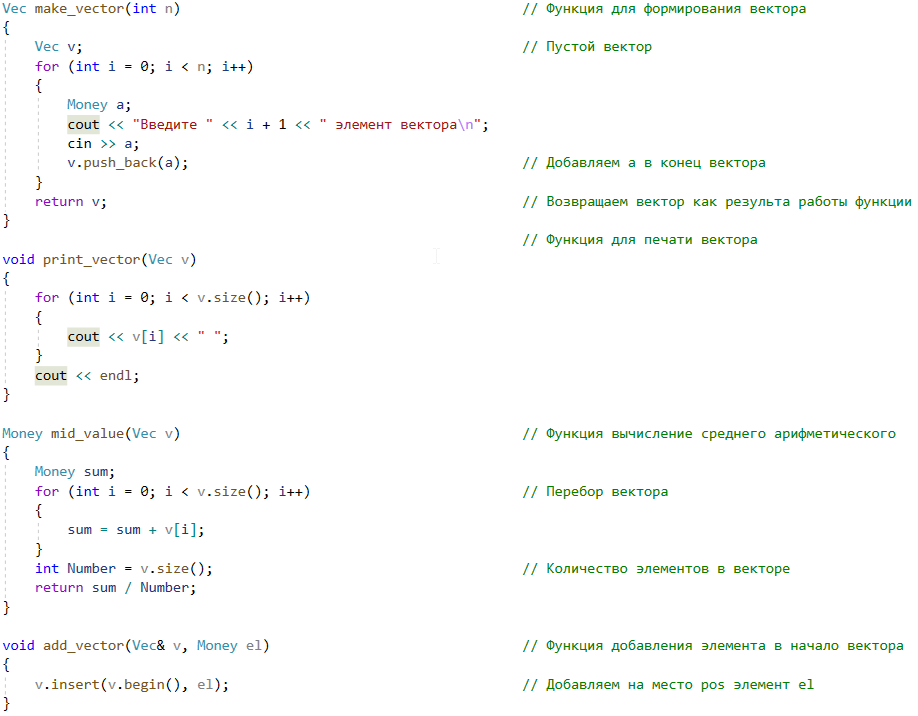
**Анализ задачи**

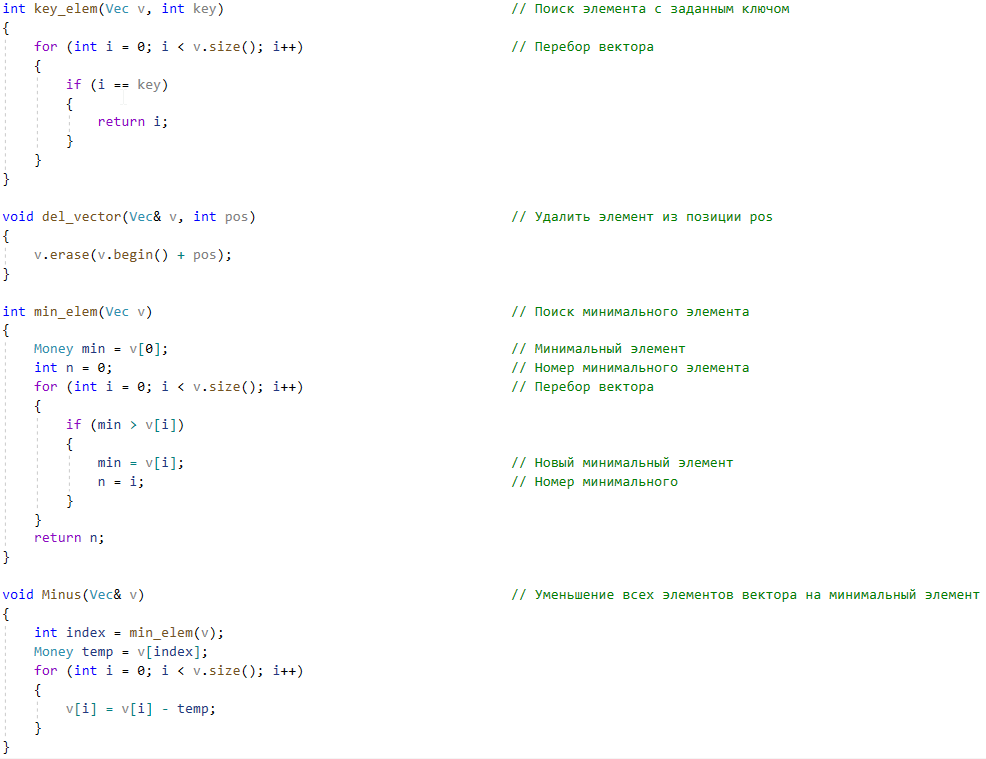
1. Для решения задачи необходимо:
   1. Реализовать определение функций для работы с вектором типа float: создание вектора, вычисление среднего арифметического, вычитание минимального элемента вектора, добавление среднего арифметического в начало вектора, вывод вектора.



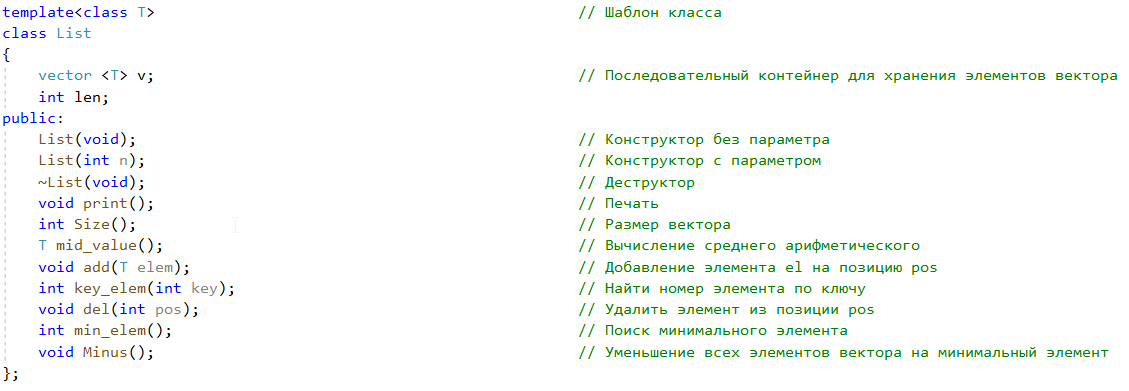


* 1. Реализовать определение функций для работы вектора с типом Money: создание вектора, вычисление среднего арифметического, вычитание минимального элемента вектора, добавление среднего арифметического в начало вектора, вывод вектора.

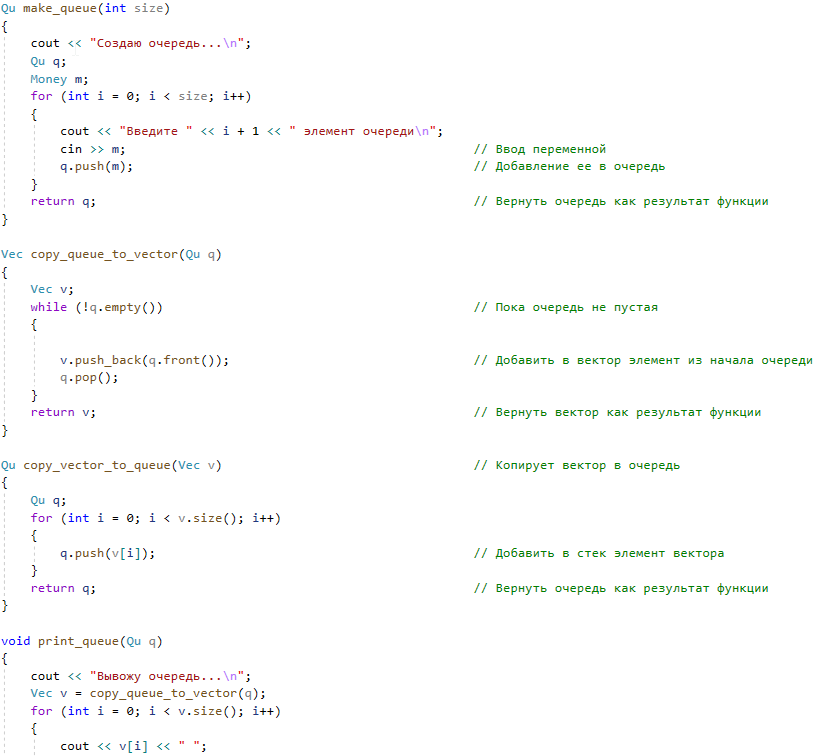


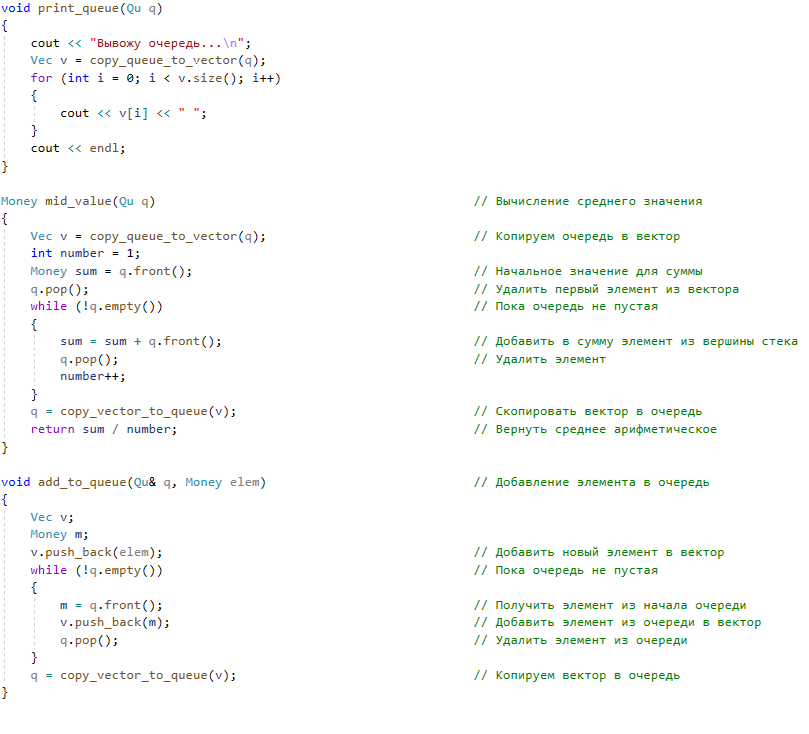


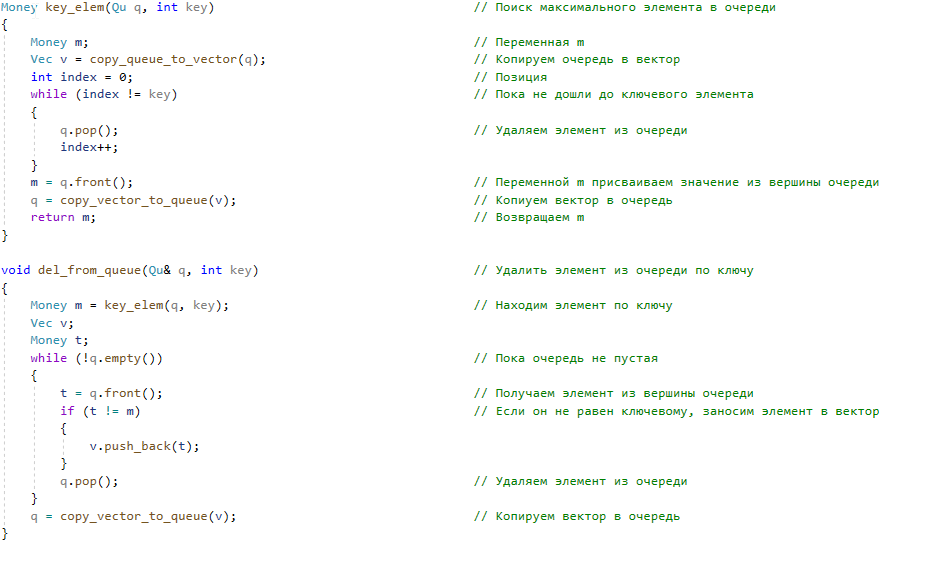
* 1. Реализовать определение параметризированного класса List и методов данного класса.

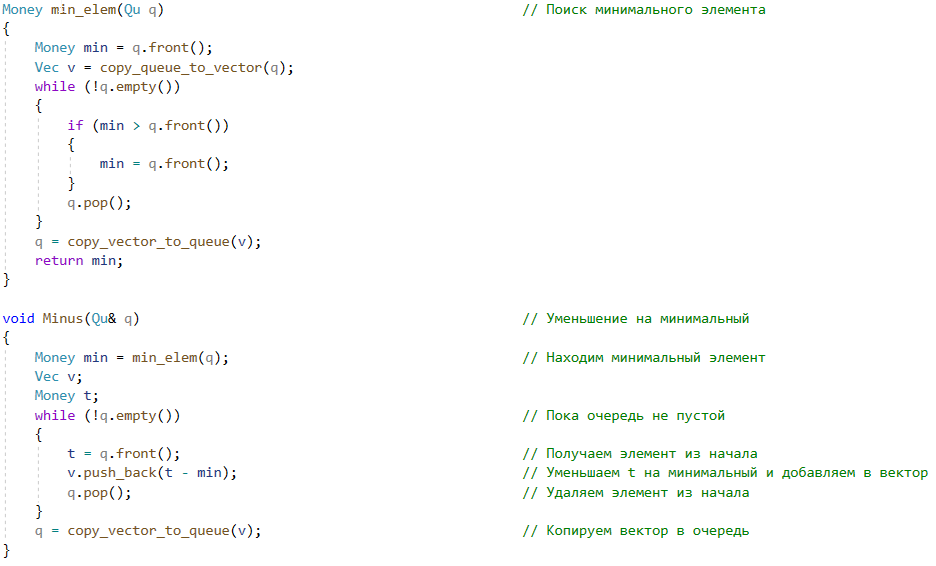


* 1. Реализовать определение функций для работы с адаптером контейнера queue.

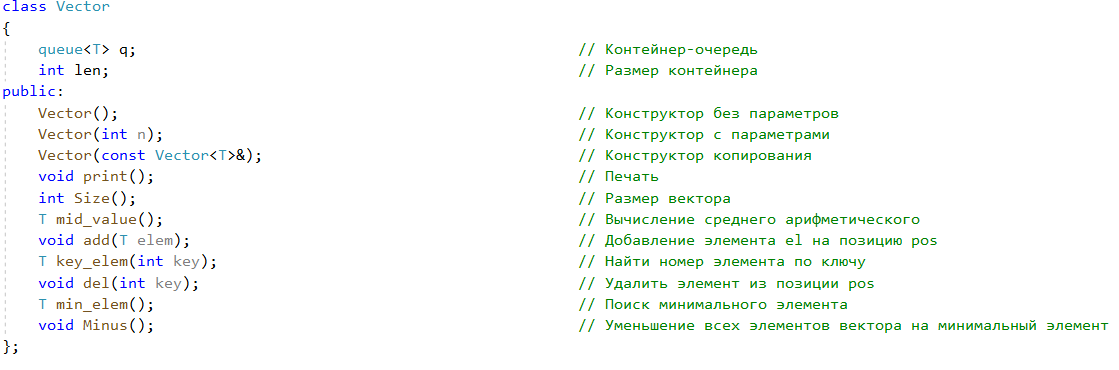








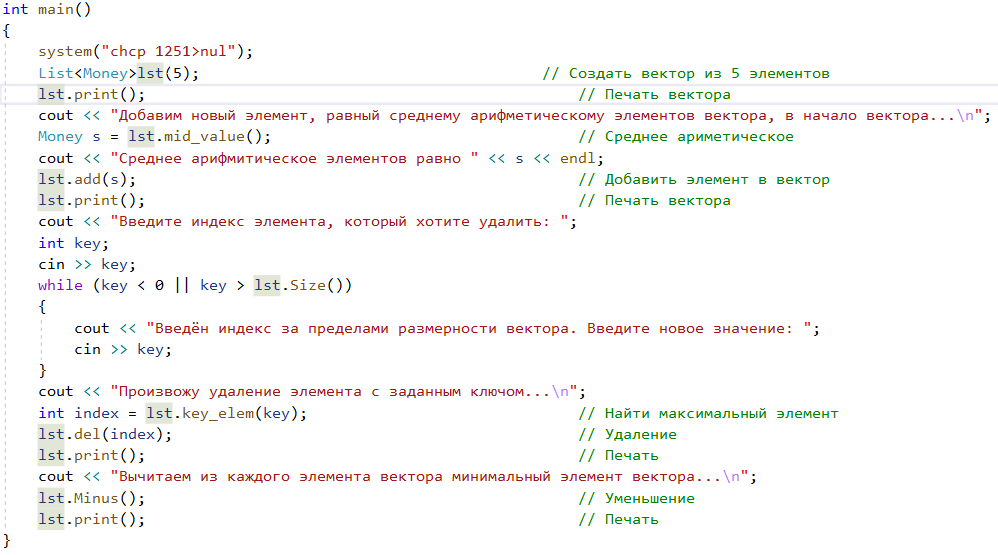
* 1. Реализовать определение параметризированного класса List и его методов для работы с адаптером шаблонного контейнера queue.



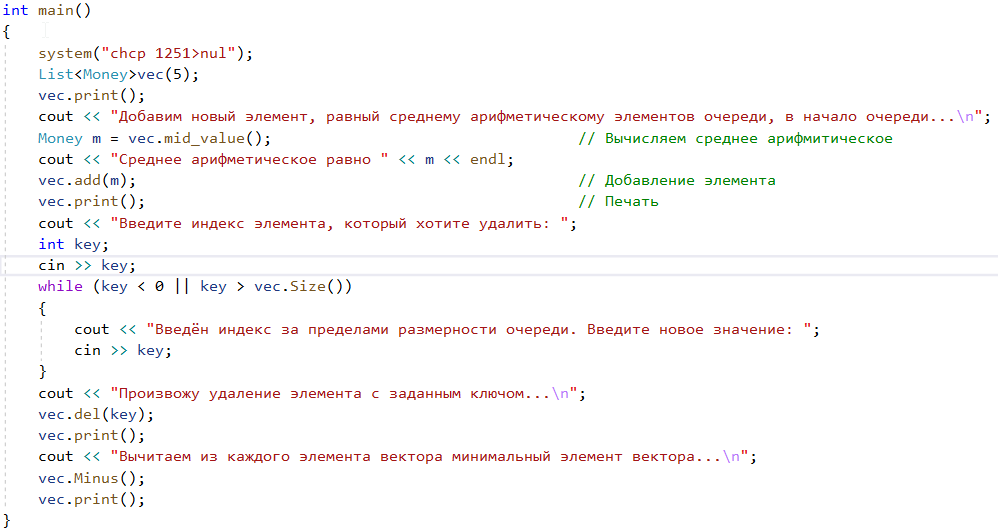
* 1. Реализовать применение этих функций в главной функции.



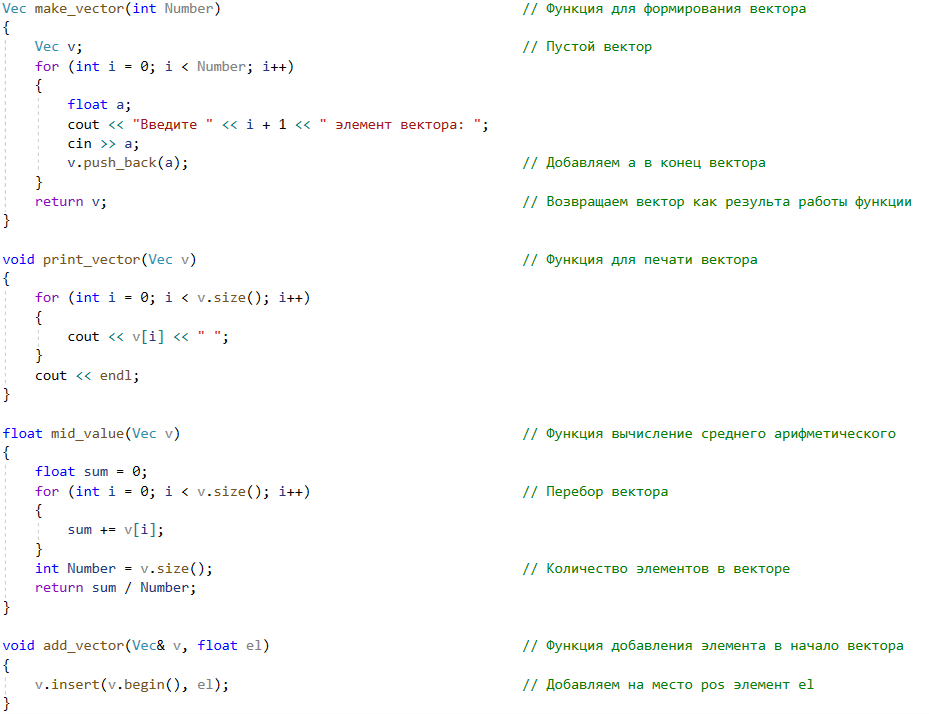


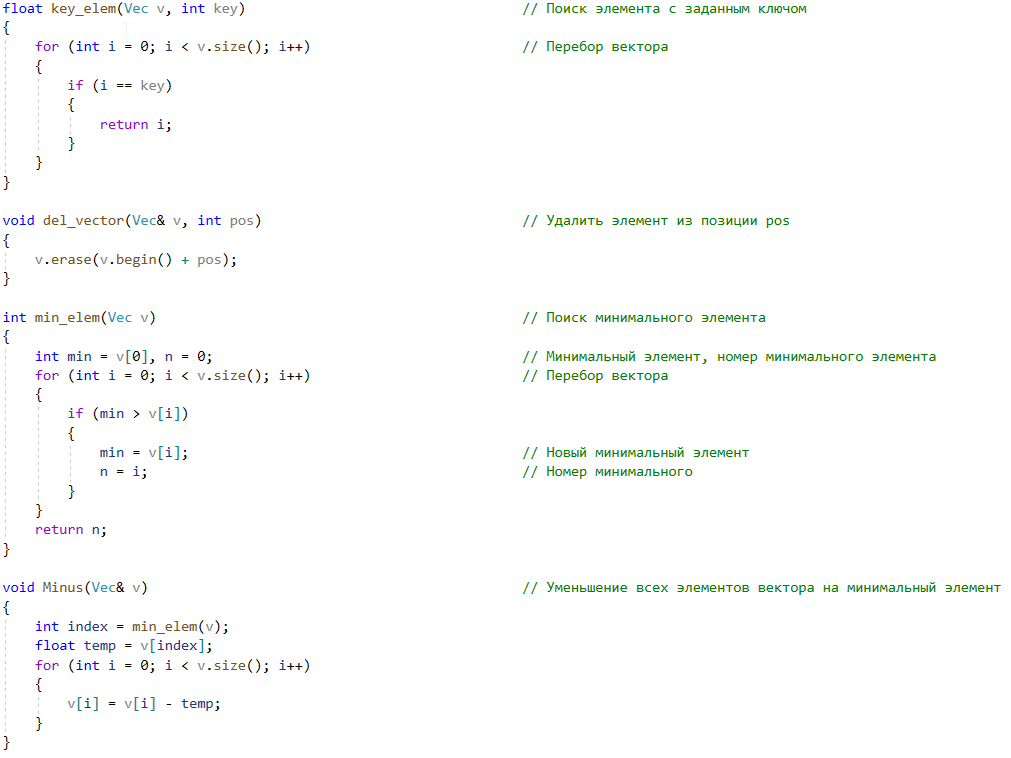




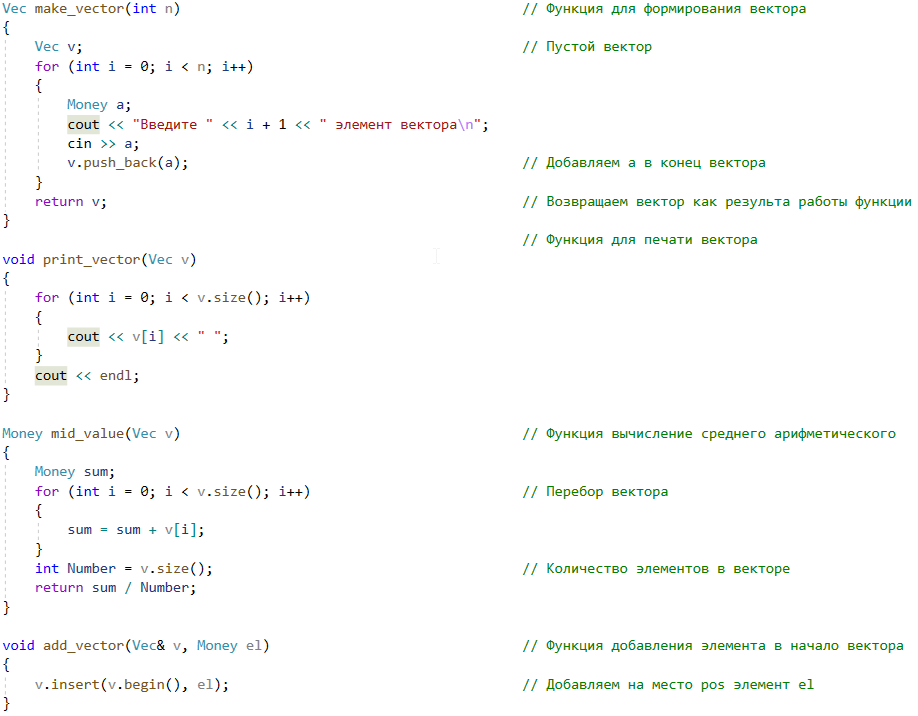


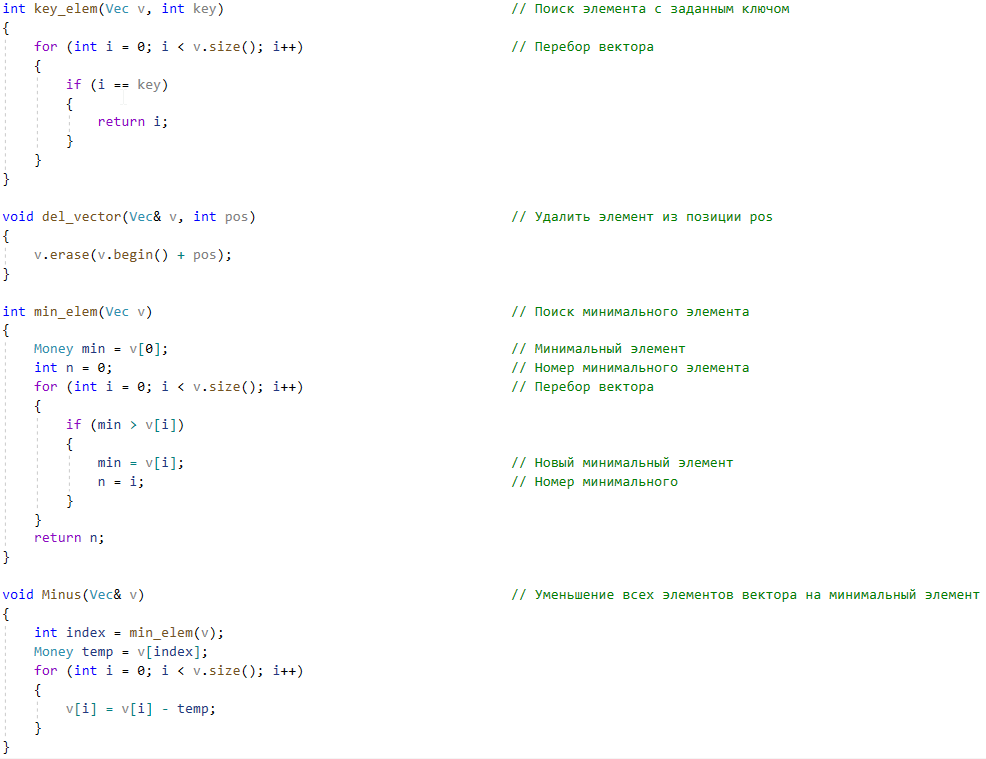
1. Поставленные задачи решены следующими действиями:
   1. Для задачи 1 были реализованы функции создания вектора, вычисления среднего арифметического, вычитания минимального элемента вектора, добавления среднего арифметического в начало вектора и вывода вектора.



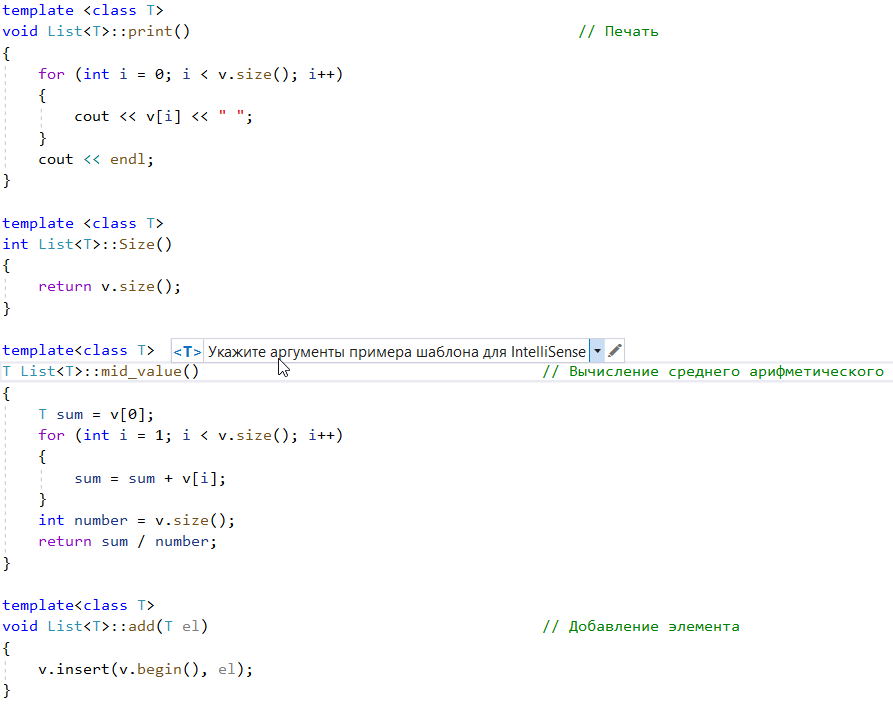


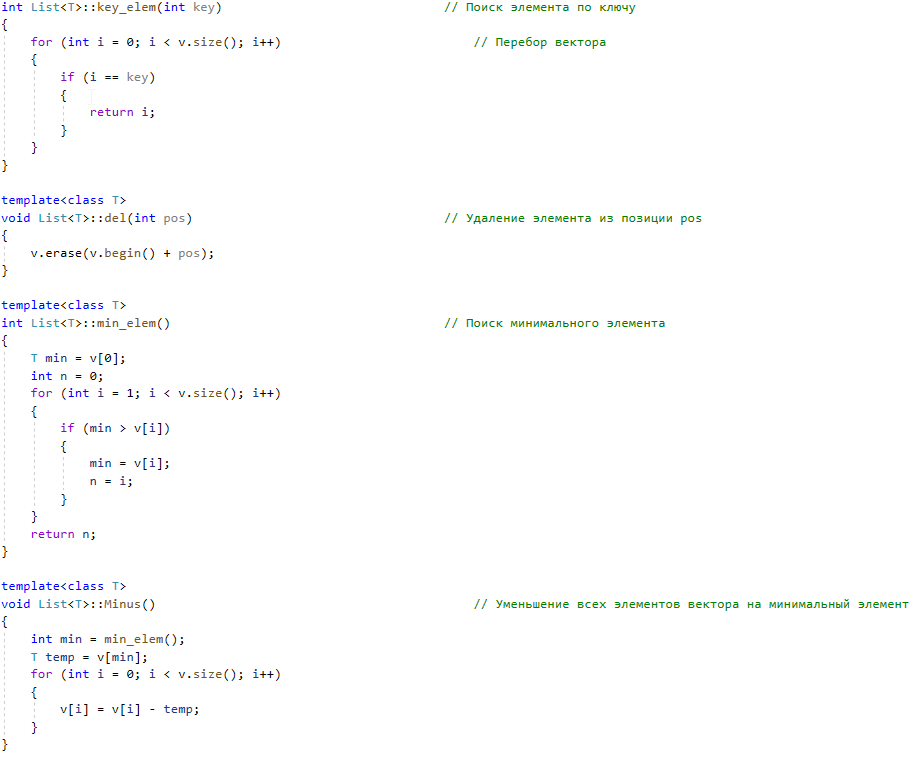
* 1. Для задачи 2 были реализованы функции создания вектора, вычисления среднего арифметического, вычитания минимального элемента вектора, добавления среднего арифметического в начало вектора и вывода вектора.



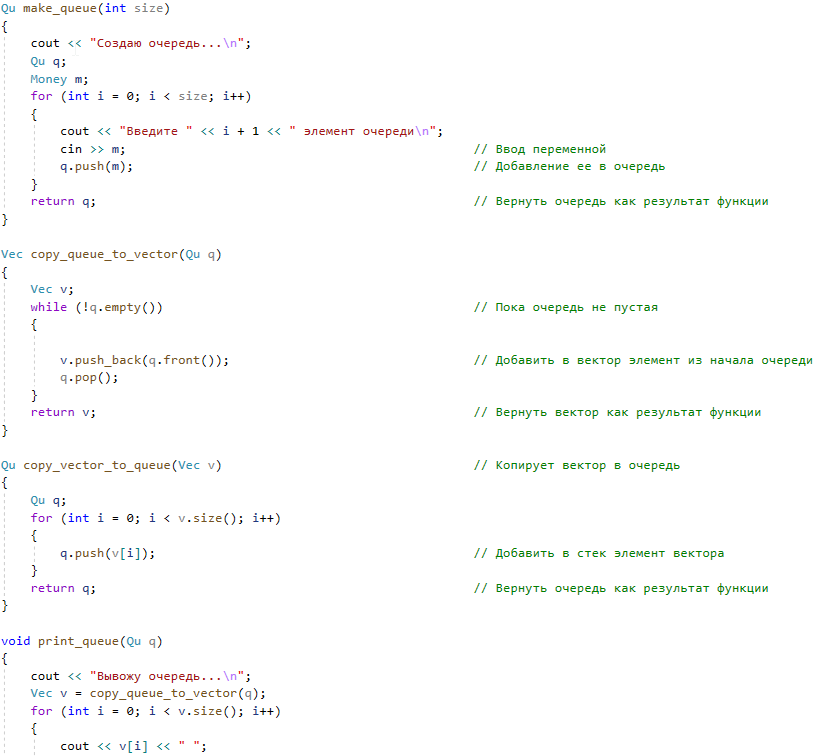


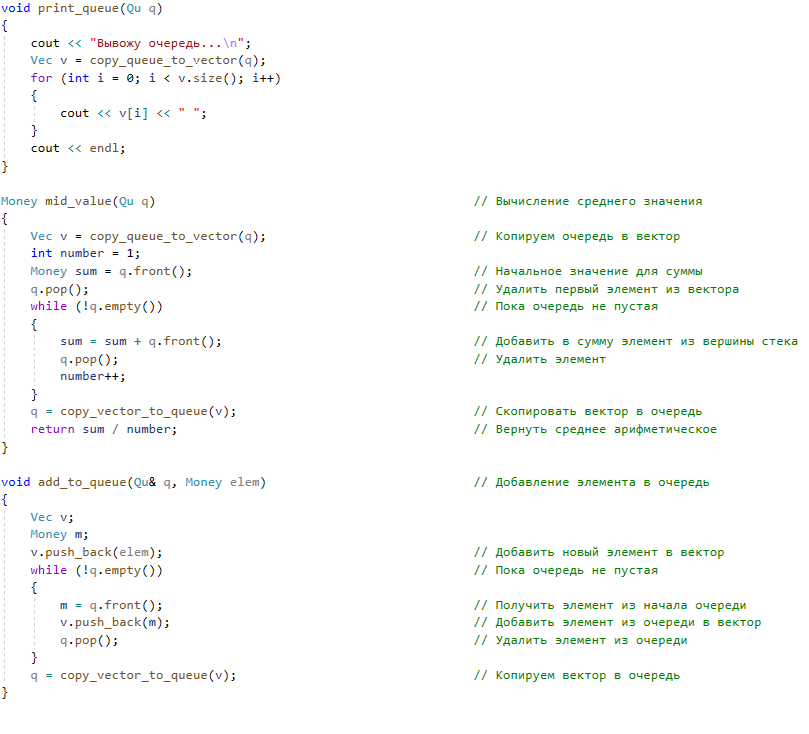
* 1. Для задачи 3 были реализованы методы создания списка, вычисления среднего арифметического, вычитания минимального элемента списка, добавления среднего арифметического в начало списка и вывода списка.

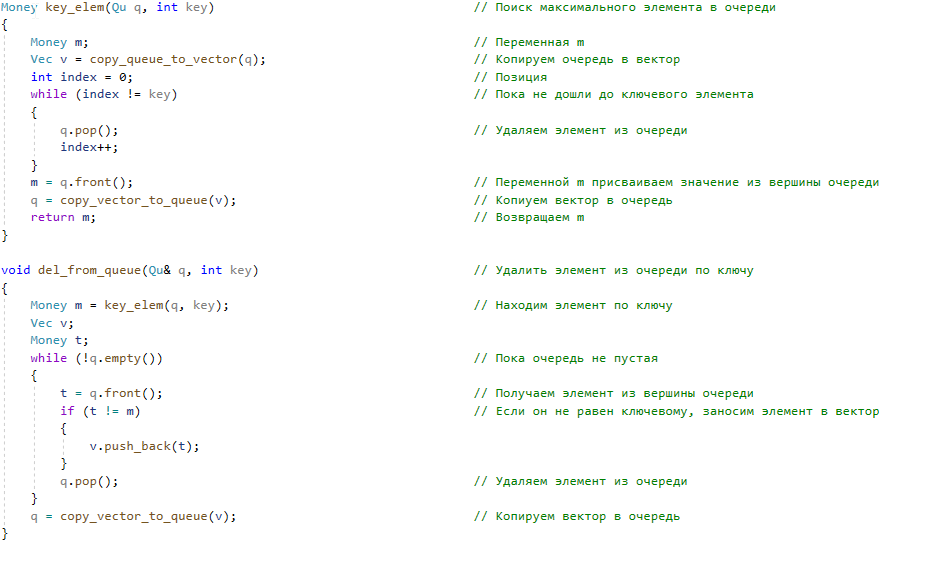


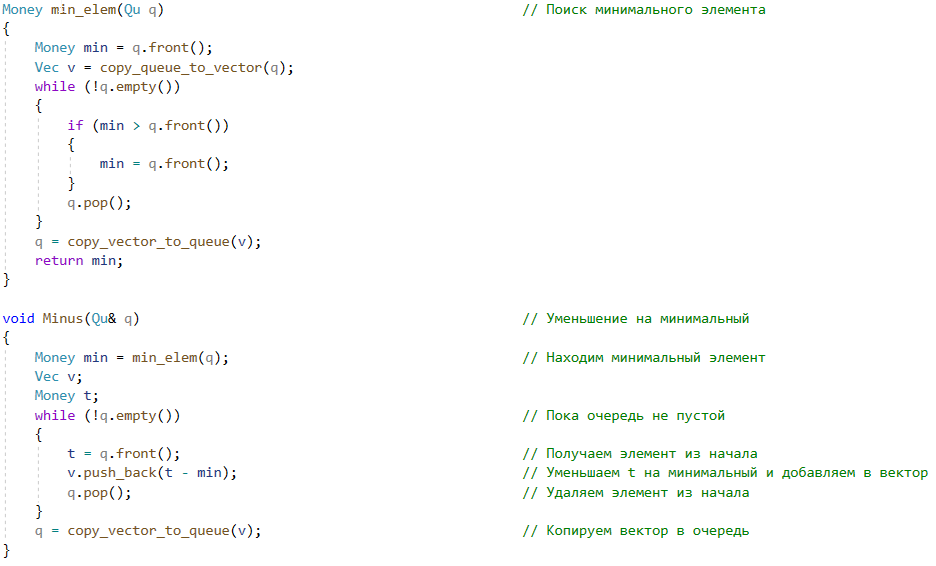


* 1. Для задачи 4 были реализованы функции создания очереди, перезаписи очереди в вектор и обратно, вывод очереди, вычисления среднего арифметического, вычитания минимального элемента очереди, добавления среднего арифметического в начало очереди.

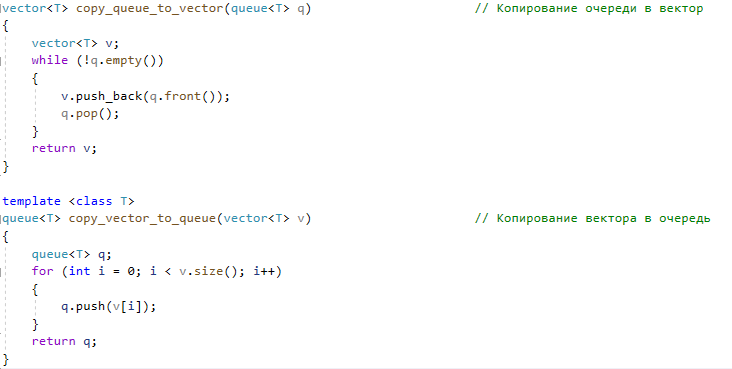


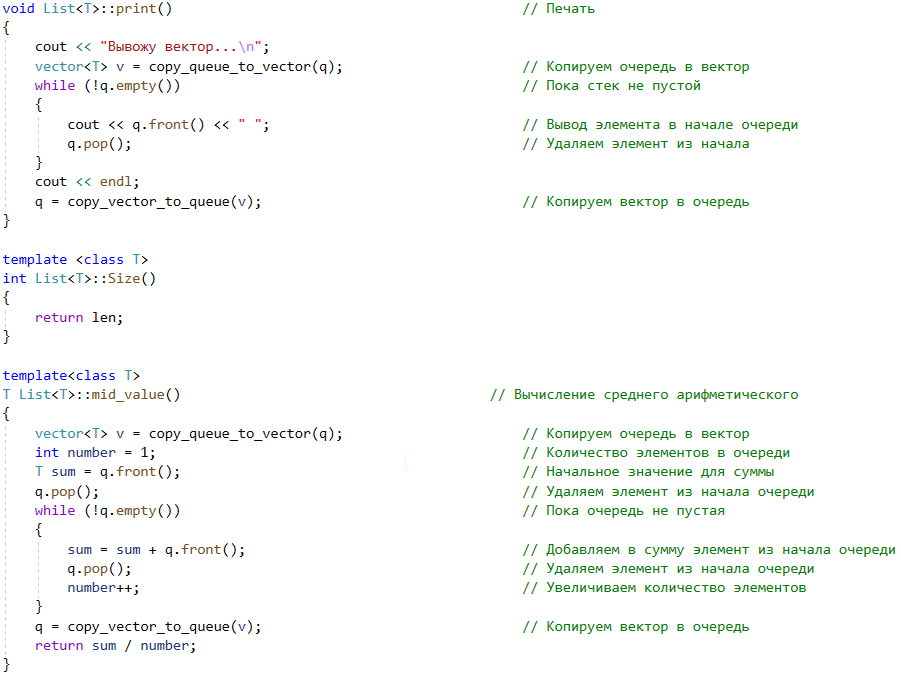




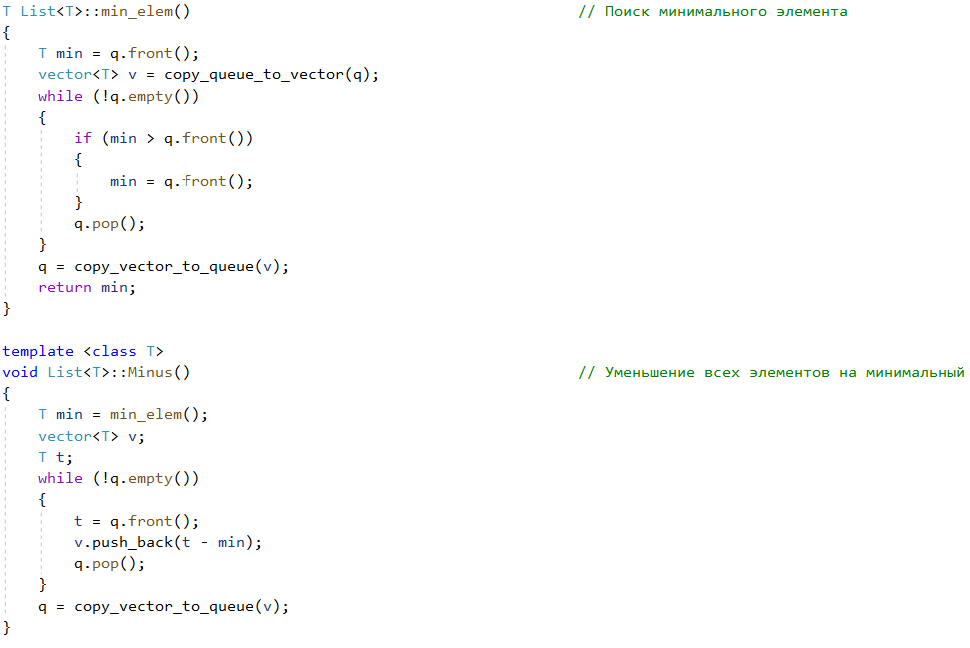


* 1. Для задачи 5 были реализованы методы создания очереди, перезаписи очереди в вектор и обратно, вывод очереди, вычисления среднего арифметического, вычитания минимального элемента очереди, добавления среднего арифметического в начало очереди.

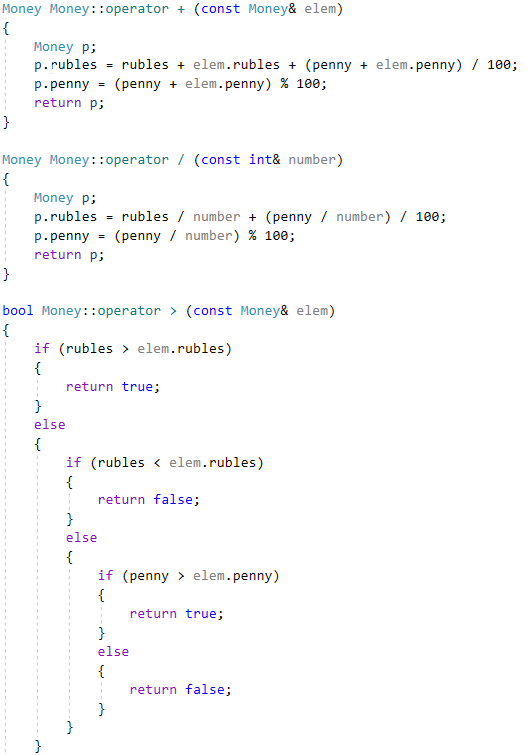


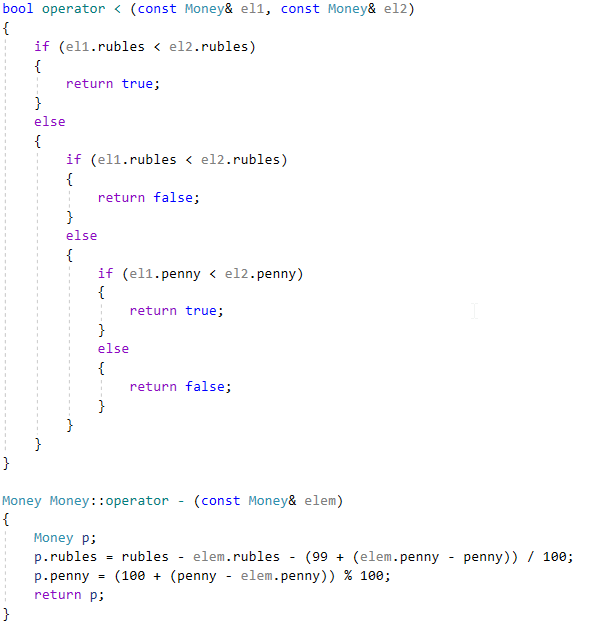


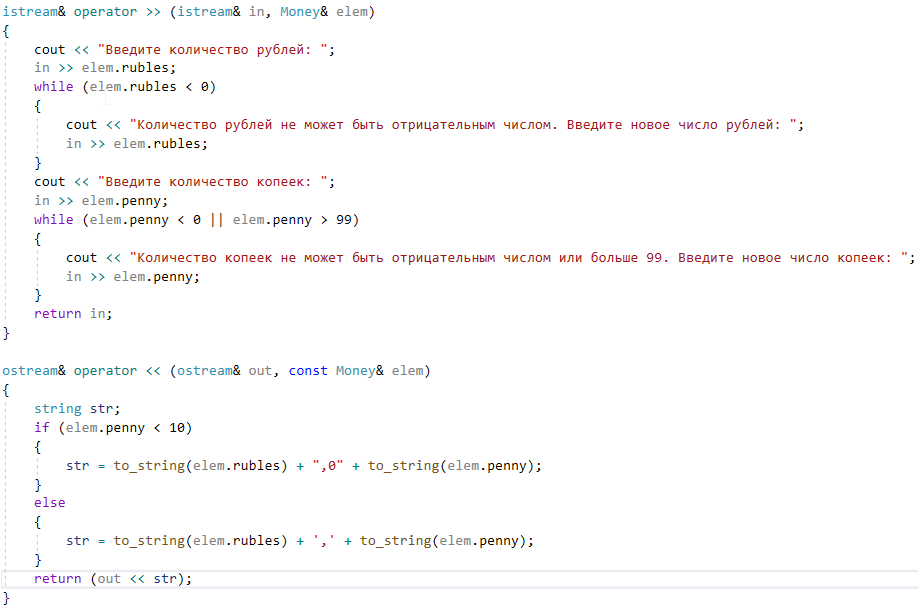




* 1. В классе Money находится реализация ввода-вывода и работы с деньгами.







**Реализация задачи на языке С++**

**(1)**

**Файл с главной программой Лабораторная работа № 18.11(1).cpp**

#include <iostream>

#include <vector>

#include <cstdlib>

using namespace std;

typedef vector<float>Vec;

Vec make\_vector(int Number)

{

Vec v;

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

{

float a;

cout << "Введите " << i + 1 << " элемент вектора: ";

cin >> a;

v.push\_back(a);

}

return v;

}

void print\_vector(Vec v)

{

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

{

cout << v[i] << " ";

}

cout << endl;

}

float mid\_value(Vec v)

{

float sum = 0;

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

{

sum += v[i];

}

int Number = v.size();

return sum / Number;

}

void add\_vector(Vec& v, float el)

{

v.insert(v.begin(), el);

}

float key\_elem(Vec v, int key)

{

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

{

if (i == key)

{

return i;

}

}

}

void del\_vector(Vec& v, int pos)

{

v.erase(v.begin() + pos);

}

int min\_elem(Vec v)

{

int min = v[0], n = 0;

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

{

if (min > v[i])

{

min = v[i];

n = i;

}

}

return n;

}

void Minus(Vec& v)

{

int index = min\_elem(v);

float temp = v[index];

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

{

v[i] = v[i] - temp;

}

}

int main()

{

system("chcp 1251>nul");

try

{

Vec vec;

Vec::iterator vi = vec.begin();

int Number;

cout << "Введите количество элементов: ";

cin >> Number;

while (Number < 0)

{

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

cin >> Number;

}

vec = make\_vector(Number);

print\_vector(vec);

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

float el = mid\_value(vec);

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

add\_vector(vec, el);

print\_vector(vec);

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

int key;

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";

int index = key\_elem(vec, key);

del\_vector(vec, index);

print\_vector(vec);

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

Minus(vec);

print\_vector(vec);

}

catch (int)

{

cout << "Ошибка!";

}

}

**(2)**

**Заголовочный файл Money.h**

#pragma once

#include <iostream>

using namespace std;

class Money

{

long rubles;

int penny;

public:

Money();

Money(long rubs, int pens);

Money(const Money& elem);

~Money();

long get\_rubles();

int get\_penny();

void set\_rubles(long rubs);

void set\_penny(int pens);

Money& operator = (const Money& elem);

Money operator + (const Money& elem);

Money operator / (const int& number);

bool operator > (const Money& elem);

//bool operator < (const Money& elem);

friend bool operator < (const Money& el1, const Money& el2);

Money operator - (const Money& elem);

bool operator != (const Money& elem) { if (rubles != elem.rubles && penny != elem.penny) return true; else return false; }

bool operator == (const Money& elem) { if (rubles == elem.rubles && penny == elem.penny) return true; else return false; }

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

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

};

**Файл с описанием методов класса Money.cpp**

#include "Money.h"

#include <string>

using namespace std;

Money::Money()

{

rubles = 0;

penny = 0;

}

Money::Money(long rubs, int pens)

{

rubles = rubs;

penny = pens;

}

Money::Money(const Money& elem)

{

rubles = elem.rubles;

penny = elem.penny;

}

Money::~Money()

{

}

long Money::get\_rubles()

{

return rubles;

}

int Money::get\_penny()

{

return penny;

}

void Money::set\_rubles(long rubs)

{

rubles = rubs;

}

void Money::set\_penny(int pens)

{

penny = pens;

}

Money& Money::operator = (const Money& elem)

{

if (this != &elem)

{

rubles = elem.rubles;

penny = elem.penny;

}

return \*this;

}

Money Money::operator + (const Money& elem)

{

Money p;

p.rubles = rubles + elem.rubles + (penny + elem.penny) / 100;

p.penny = (penny + elem.penny) % 100;

return p;

}

Money Money::operator / (const int& number)

{

Money p;

p.rubles = rubles / number + (penny / number) / 100;

p.penny = (penny / number) % 100;

return p;

}

bool Money::operator > (const Money& elem)

{

if (rubles > elem.rubles)

{

return true;

}

else

{

if (rubles < elem.rubles)

{

return false;

}

else

{

if (penny > elem.penny)

{

return true;

}

else

{

return false;

}

}

}

}

bool operator < (const Money& el1, const Money& el2)

{

if (el1.rubles < el2.rubles)

{

return true;

}

else

{

if (el1.rubles < el2.rubles)

{

return false;

}

else

{

if (el1.penny < el2.penny)

{

return true;

}

else

{

return false;

}

}

}

}

Money Money::operator - (const Money& elem)

{

Money p;

p.rubles = rubles - elem.rubles - (99 + (elem.penny - penny)) / 100;

p.penny = (100 + (penny - elem.penny)) % 100;

return p;

}

istream& operator >> (istream& in, Money& elem)

{

cout << "Введите количество рублей: ";

in >> elem.rubles;

while (elem.rubles < 0)

{

cout << "Количество рублей не может быть отрицательным числом. Введите новое число рублей: ";

in >> elem.rubles;

}

cout << "Введите количество копеек: ";

in >> elem.penny;

while (elem.penny < 0 || elem.penny > 99)

{

cout << "Количество копеек не может быть отрицательным числом или больше 99. Введите новое число копеек: ";

in >> elem.penny;

}

return in;

}

ostream& operator << (ostream& out, const Money& elem)

{

string str;

if (elem.penny < 10)

{

str = to\_string(elem.rubles) + ",0" + to\_string(elem.penny);

}

else

{

str = to\_string(elem.rubles) + ',' + to\_string(elem.penny);

}

return (out << str);

}

**Файл с главной программой Лабораторная работа № 18.11(2).cpp**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <stdlib.h>

#include "Money.h"

using namespace std;

typedef vector<Money>Vec;

Vec make\_vector(int n)

{

Vec v;

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

{

Money a;

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

cin >> a;

v.push\_back(a);

}

return v;

}

void print\_vector(Vec v)

{

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

{

cout << v[i] << " ";

}

cout << endl;

}

Money mid\_value(Vec v)

{

Money sum;

for (int i = 0; i < v.size(); i++) // Перебор вектора

{

sum = sum + v[i];

}

int Number = v.size();

return sum / Number;

}

void add\_vector(Vec& v, Money el)

{

v.insert(v.begin(), el);

}

int key\_elem(Vec v, int key)

{

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

{

if (i == key)

{

return i;

}

}

}

void del\_vector(Vec& v, int pos)

{

v.erase(v.begin() + pos);

}

int min\_elem(Vec v)

{

Money min = v[0];

int n = 0;

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

{

if (min > v[i])

{

min = v[i];

n = i;

}

}

return n;

}

void Minus(Vec& v)

{

int index = min\_elem(v);

Money temp = v[index];

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

{

v[i] = v[i] - temp;

}

}

int main()

{

system("chcp 1251>nul");

try

{

Vec vec;

Vec::iterator vi = vec.begin();

int Number;

cout << "Введите количество элементов: ";

cin >> Number;

while (Number < 0)

{

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

cin >> Number;

}

vec = make\_vector(Number);

print\_vector(vec);

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

Money el = mid\_value(vec);

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

add\_vector(vec, el);

print\_vector(vec);

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

int key;

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";

int index = key\_elem(vec, key);

del\_vector(vec, index);

print\_vector(vec);

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

Minus(vec);

print\_vector(vec);

}

catch (int)

{

cout << "error!";

}

}

**(3)**

**Заголовочный файл Vector.h**

#pragma once

#include <vector>

#include <iostream>

using namespace std;

template<class T>

class List

{

vector <T> v;

int len;

public:

List(void);

List(int n);

~List(void);

void print();

int Size();

T mid\_value();

void add(T elem);

int key\_elem(int key);

void del(int pos);

int min\_elem();

void Minus();

};

template <class T>

List<T>::List()

{

len = 0;

}

template <class T>

List<T>::~List(void)

{

}

template <class T>

List<T>::List(int n)

{

T a;

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

{

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

cin >> a;

v.push\_back(a);

}

len = v.size();

}

template <class T>

void List<T>::print()

{

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

{

cout << v[i] << " ";

}

cout << endl;

}

template <class T>

int List<T>::Size()

{

return v.size();

}

template<class T>

T List<T>::mid\_value()

{

T sum = v[0];

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

{

sum = sum + v[i];

}

int number = v.size();

return sum / number;

}

template<class T>

void List<T>::add(T el)

{

v.insert(v.begin(), el);

}

template <class T>

int List<T>::key\_elem(int key)

{

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

{

if (i == key)

{

return i;

}

}

}

template<class T>

void List<T>::del(int pos)

{

v.erase(v.begin() + pos);

}

template<class T>

int List<T>::min\_elem()

{

T min = v[0];

int n = 0;

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

{

if (min > v[i])

{

min = v[i];

n = i;

}

}

return n;

}

template<class T>

void List<T>::Minus(){

int min = min\_elem();

T temp = v[min];

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

{

v[i] = v[i] - temp;

}

}

**Файл с главной программой Лабораторная работа № 18.11(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 "Vector.h"

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251>nul");

List<Money>lst(5);

lst.print();

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

Money s = lst.mid\_value();

cout << "Среднее арифмитическое элементов равно " << s << endl;

lst.add(s);

lst.print();

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

int key;

cin >> key;

while (key < 0 || key > lst.Size())

{

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

cin >> key;

}

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

int index = lst.key\_elem(key);

lst.del(index);

lst.print();

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

lst.Minus();

lst.print();

}

**(4)**

**Файл с главной программой Лабораторная работа № 18.11(4).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>

using namespace std;

typedef queue<Money>Qu;

typedef vector<Money>Vec;

Qu make\_queue(int size)

{

cout << "Создаю очередь...\n";

Qu q;

Money m;

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

{

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

cin >> m;

q.push(m);

}

return q;

}

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

{

Vec v;

while (!q.empty())

{

v.push\_back(q.front());

q.pop();

}

return v;

}

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

{

Qu q;

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

{

q.push(v[i]);

}

return q;

}

void print\_queue(Qu q)

{

cout << "Вывожу очередь...\n";

Vec v = copy\_queue\_to\_vector(q);

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

{

cout << v[i] << " ";

}

cout << endl;

}

Money mid\_value(Qu q)

{

Vec v = copy\_queue\_to\_vector(q);

int number = 1;

Money sum = q.front();

q.pop();

while (!q.empty())

{

sum = sum + q.front();

q.pop();

number++;

}

q = copy\_vector\_to\_queue(v);

return sum / number;

}

void add\_to\_queue(Qu& q, Money elem)

{

Vec v;

Money m;

v.push\_back(elem);

while (!q.empty())

{

m = q.front();

v.push\_back(m);

q.pop();

}

q = copy\_vector\_to\_queue(v);

}

Money key\_elem(Qu q, int key)

{

Money m;

Vec v = copy\_queue\_to\_vector(q);

int index = 0;

while (index != key)

{

q.pop();

index++;

}

m = q.front();

q = copy\_vector\_to\_queue(v);

return m;

}

void del\_from\_queue(Qu& q, int key)

{

Money m = key\_elem(q, key);

Vec v;

Money t;

while (!q.empty())

{

t = q.front();

if (t != m)

{

v.push\_back(t);

}

q.pop();

}

q = copy\_vector\_to\_queue(v);

}

Money min\_elem(Qu q)

{

Money min = q.front();

Vec v = copy\_queue\_to\_vector(q);

while (!q.empty())

{

if (min > q.front())

{

min = q.front();

}

q.pop();

}

q = copy\_vector\_to\_queue(v);

return min;

}

void Minus(Qu& q)

{

Money min = min\_elem(q);

Vec v;

Money t;

while (!q.empty())

{

t = q.front();

v.push\_back(t - min);

q.pop();

}

q = copy\_vector\_to\_queue(v);

}

int main()

{

system("chcp 1251>nul");

Money m;

Qu q;

int number;

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

cin >> number;

while (number < 0)

{

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

cin >> number;

}

q = make\_queue(number);

print\_queue(q);

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

m = mid\_value(q);

cout << "Среднее арифметическое равно " << mid\_value(q) << endl;

add\_to\_queue(q, m);

print\_queue(q); cout << "Введите индекс элемента, который хотите удалить: ";

int key;

cin >> key;

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

{

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

cin >> key;

}

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

del\_from\_queue(q, key);

print\_queue(q);

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

Minus(q);

print\_queue(q);

}

**(5)**

**Заголовочный файл Vector.h**

#pragma once

#include <iostream>

#include <queue>

#include <vector>

using namespace std;

template<class T>

class List

{

queue<T> q;

int len;

public:

List();

List(int n);

List(const List<T>&);

void print();

int Size();

T mid\_value();

void add(T elem);

T key\_elem(int key);

void del(int key);

T min\_elem();

void Minus();

};

template <class T>

vector<T> copy\_queue\_to\_vector(queue<T> q)

{

vector<T> v;

while (!q.empty())

{

v.push\_back(q.front());

q.pop();

}

return v;

}

template <class T>

queue<T> copy\_vector\_to\_queue(vector<T> v)

{

queue<T> q;

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

{

q.push(v[i]);

}

return q;

}

template <class T>

List<T>::List()

{

len = 0;

}

template <class T>

List<T>::List(int n)

{

T a;

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

{

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

cin >> a;

q.push(a);

}

len = q.size();

}

template <class T>

List<T>::List(const List<T>& Vec)

{

vector<T> v = copy\_queue\_to\_vector(Vec.q);

len = v.len;

q = copy\_vector\_to\_queue(v);

}

template <class T>

void List<T>::print()

{

cout << "Вывожу вектор...\n";

vector<T> v = copy\_queue\_to\_vector(q);

while (!q.empty())

{

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

q.pop();

}

cout << endl;

q = copy\_vector\_to\_queue(v);

}

template <class T>

int List<T>::Size()

{

return len;

}

template<class T>

T List<T>::mid\_value()

{

vector<T> v = copy\_queue\_to\_vector(q);

int number = 1;

T sum = q.front();

q.pop();

while (!q.empty())

{

sum = sum + q.front();

q.pop();

number++;

}

q = copy\_vector\_to\_queue(v);

return sum / number;

}

template<class T>

void List<T>::add(T elem)

{

vector<T> v;

T t;

v.push\_back(elem);

while (!q.empty())

{

t = q.front();

v.push\_back(t);

q.pop();

}

q = copy\_vector\_to\_queue(v);

}

template <class T>

T List<T>::key\_elem(int key)

{

T min;

vector<T> v = copy\_queue\_to\_vector(q);

int index = 0;

while (index != key)

{

q.pop();

index++;

}

min = q.front();

q = copy\_vector\_to\_queue(v);

return min;

}

template <class T>

void List<T>::del(int key)

{

T min = key\_elem(key);

vector<T> v;

T t;

while (!q.empty())

{

t = q.front();

if (t != min)

{

v.push\_back(t);

}

q.pop();

}

q = copy\_vector\_to\_queue(v);

}

template <class T>

T List<T>::min\_elem()

{

T min = q.front();

vector<T> v = copy\_queue\_to\_vector(q);

while (!q.empty())

{

if (min > q.front())

{

min = q.front();

}

q.pop();

}

q = copy\_vector\_to\_queue(v);

return min;

}

template <class T>

void List<T>::Minus()

{

T min = min\_elem();

vector<T> v;

T t;

while (!q.empty())

{

t = q.front();

v.push\_back(t - min);

q.pop();

}

q = copy\_vector\_to\_queue(v);

}

**Файл с главной программой Лабораторная работа № 18.11(5).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 <stack>

#include <vector>

#include "Vector.h"

using namespace std;

int main()

{

system("chcp 1251>nul");

List<Money>vec(5);

vec.print();

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

Money m = vec.mid\_value();

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

vec.add(m);

vec.print();

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

int key;

cin >> key;

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

{

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

cin >> key;

}

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

vec.del(key);

vec.print();

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

vec.Minus();

vec.print();

}

**Ответы на вопросы**

1. Классы и алгоритмы
2. Списки стеки очереди
3. Подключить библиотеку
4. Указатель
5. Инкремент, декремент, разыменование
6. От begin() до end()
7. Входные выходные прямые
8. Добавление и удаление элементов, вычисление длины, проверка на пустоту
9. Вставка элемента в конец
10. Вставка элемента в конец
11. Вставка элемента в конец
12. Доступ по индексу, вставка элемента
13. Добавление элемента, удаление, вставка
14. Вставка элементов в начало и в конец
15. Emplace
16. Emplace\_back
17. Erase
18. Pop
19. Erase
20. Pop
21. For (auto i=a.begin();i!=a.end();i++) cout << i<< “ “;
22. Контейнер, основанный на другом контейнере
23. Стек чисел и стек списков чисел
24. Добавить элемент в начало, удалить первый элемент
25. Добавить элемент в конец, удалить элемент сначала
26. Последний расставляет элементы по порядку
27. Переложить в другой стек все кроме нужного
28. Перекладывать из начала в конец, нужный пропустить
29. For (auto i=a.begin();i!=a.end();i++) cout << i<< “ “;
30. For (auto i=a.begin();i!=a.end();i++) cout << i<< “ “;