Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа №11

" Последовательные контейнеры библиотеки STL "

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 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.

Вариант:

Задача 1
1. Контейнер - список
2. Тип элементов - int

Задача 2
Тип элементов Pair (см. лабораторную работу №3).

Задача 3
Параметризированный класс – Список (см. лабораторную работу №7)

Задача 4
Адаптер контейнера – очередь с приоритетами.

Задача 5
Параметризированный класс – Список
Адаптер контейнера – очередь с приоритетами.

Задание 3	**Задание 4**	**Задание 5**
Найти среднее арифметическое и добавить его на заданную позицию контейнера	Найти элементы ключами из заданного диапазона и удалить их из контейнера	Из каждого элемента вычесть среднее арифметическое контейнера.

2 Код на C++

money.h | money.cpp | mylist.h | queuelist.h | tasks.h | tasks.cpp | main.cpp

11лабапеределка | (Глобальная область)

```cpp
#include <windows.h>
#include <clocale>
#include "tasks.h"

int main() {
    SetConsoleOutputCP(65001);
    setlocale(LC_ALL, "ru");

    task1();
    task2();
    task3();
    task4();
    task5();

    return 0;
}
```

money.h | money.cpp | mylist.h | queuelist.h | tasks.h | tasks.cpp | main.cpp

11лабапеределка | (Глобальная область)

```cpp
#include "tasks.h"
#include "money.h"
#include "mylist.h"
#include "queuelist.h"
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>

using namespace std;

void task1() {
    cout << "Задача 1 (float): ";
    vector<float> v = { 3.5, 2.0, 7.1, 4.9 };

    float sum = 0;
    for (float x : v) sum += x;
    float avg = sum / v.size();
    v.insert(v.begin(), avg);

    v.erase(remove(v.begin(), v.end(), 2.0f), v.end());

    float min_val = v[0];
    for (float x : v) if (x < min_val) min_val = x;
    for (float& x : v) x -= min_val;

    for (float x : v) cout << x << " ";
    cout << endl;
}

void task2() {
    cout << "Задача 2 (Money): ";
    vector<Money> v = { Money(1, 50), Money(3, 20), Money(2, 0), Money(1, 50)

    float sum = 0;
    for (const auto& m : v) sum += m.toFloat();
    float avg = sum / v.size();
    v.insert(v.begin(), Money(int(avg), int((avg - int(avg)) * 100)));

    Money key(1, 50);
    v.erase(remove(v.begin(), v.end(), key), v.end());

    float min_f = v[0].toFloat();
    for (const auto& m : v) if (m.toFloat() < min_f) min_f = m.toFloat();

    for (auto& m : v) {
        float diff = m.toFloat() - min_f;
        int r = int(diff);
        int k = int((diff - r) * 100);
        cout << Money(r, k) << " ";
    }
    cout << endl;
}

void task3() {
    cout << "Задача 3 (MyList<float>): ";
    MyList<float> l;
    l.add(3.5f); l.add(1.0f); l.add(4.0f);
    l.insertAverageAtBegin();
    l.removeByKey(1.0f);
    l.subtractMin();
    l.print();
}

void task4() {
    cout << "Задача 4 (queue<Money>): ";
    queue<Money> q;
    q.push(Money(1, 10));
    q.push(Money(3, 20));
    q.push(Money(2, 50));

    vector<Money> temp;
    float sum = 0;
    while (!q.empty()) {
        temp.push_back(q.front());
        sum += q.front().toFloat();
        q.pop();
    }

    float avg = sum / temp.size();
    temp.insert(temp.begin(), Money(int(avg), int((avg - int(avg)) * 100)));

    Money key(2, 50);
    vector<Money> filtered;
    for (auto& m : temp) if (!(m == key)) filtered.push_back(m);

    float min_f = filtered[0].toFloat();
    for (auto& m : filtered) if (m.toFloat() < min_f) min_f = m.toFloat();

    for (auto& m : filtered) {
        float diff = m.toFloat() - min_f;
        int r = int(diff);
        int k = int((diff - r) * 100);
        cout << Money(r, k) << " ";
    }
    cout << endl;
}

void task5() {
```

```
    QueueList<float> q;
    q.add(5.0f); q.add(2.0f); q.add(4.0f);
    q.insertAverage();
    q.removeByKey(2.0f);
    q.subtractMin();
}
```

money.h | money.cpp | mylist.h | queuelist.h | **tasks.h** | tasks.cpp | main.cpp

11лабапеределка — (Глобальная область)

```
#pragma once

void task1();
void task2();
void task3();
void task4();
void task5();
```

money.h | money.cpp | mylist.h | **queuelist.h** | tasks.h | tasks.cpp | main.cpp

11лабапеределка — (Глобальная область)

```
#pragma once
#include <queue>
#include <vector>
#include <iostream>
using namespace std;

template <typename T>
class QueueList {
public:
    queue<T> q;

    void add(T val) { q.push(val); }

    void insertAverage() {
        vector<T> temp;
        float sum = 0;
        while (!q.empty()) {
            temp.push_back(q.front());
            sum += q.front();
            q.pop();
        }
        float avg = sum / temp.size();
        temp.insert(temp.begin(), avg);
        for (T val : temp) q.push(val);
    }

    void removeByKey(T key) {
        vector<T> temp;
        while (!q.empty()) {
            if (q.front() != key) temp.push_back(q.front());
            q.pop();
        }
        for (T val : temp) q.push(val);
    }

    void subtractMin() {
        vector<T> temp;
        while (!q.empty()) {
            temp.push_back(q.front());
            q.pop();
        }
        T min_val = temp[0];
        for (T x : temp) if (x < min_val) min_val = x;
        cout << "Задача 5 (QueueList<float>): ";
        for (T& x : temp) {
            x -= min_val;
            cout << x << " ";
        }
        cout << endl;
        for (T val : temp) q.push(val);
    }
};
```

money.h | money.cpp | mylist.h | queuelist.h | tasks.h | tasks.cpp | main.cpp

11лабапеределка | (Глобальная область)

```cpp
#pragma once
#include <vector>
#include <iostream>
using namespace std;

template <typename T>
class MyList {
public:
    vector<T> data;

    void add(T val) { data.push_back(val); }

    void insertAverageAtBegin() {
        float sum = 0;
        for (T x : data) sum += x;
        float avg = sum / data.size();
        data.insert(data.begin(), avg);
    }

    void removeByKey(T key) {
        data.erase(remove(data.begin(), data.end(), key), data.end());
    }

    void subtractMin() {
        T min_val = data[0];
        for (T x : data) if (x < min_val) min_val = x;
        for (T& x : data) x -= min_val;
    }

    void print() const {
        for (T x : data) cout << x << " ";
        cout << endl;
    }
};
```

money.h | money.cpp | mylist.h | queuelist.h | tasks.h | tasks.cpp | main.cpp

11лабапеределка | (Глобальная облас

```cpp
#include "money.h"

Money::Money() : rub(0), kop(0) {}

Money::Money(int r, int k) {
    rub = r + k / 100;
    kop = k % 100;
}

float Money::toFloat() const {
    return rub + kop / 100.0f;
}

bool Money::operator==(const Money& m) const {
    return rub == m.rub && kop == m.kop;
}

Money Money::operator-(const Money& m) const {
    float diff = toFloat() - m.toFloat();
    int r = int(diff);
    int k = int((diff - r) * 100);
    return Money(r, k);
}

ostream& operator<<(ostream& os, const Money& m) {
    os << m.rub << "." << (m.kop < 10 ? "0" : "") << m.kop;
    return os;
}
```

money.h | money.cpp | mylist.h | queuelist.h | tasks.h | tasks.cpp | main.cpp

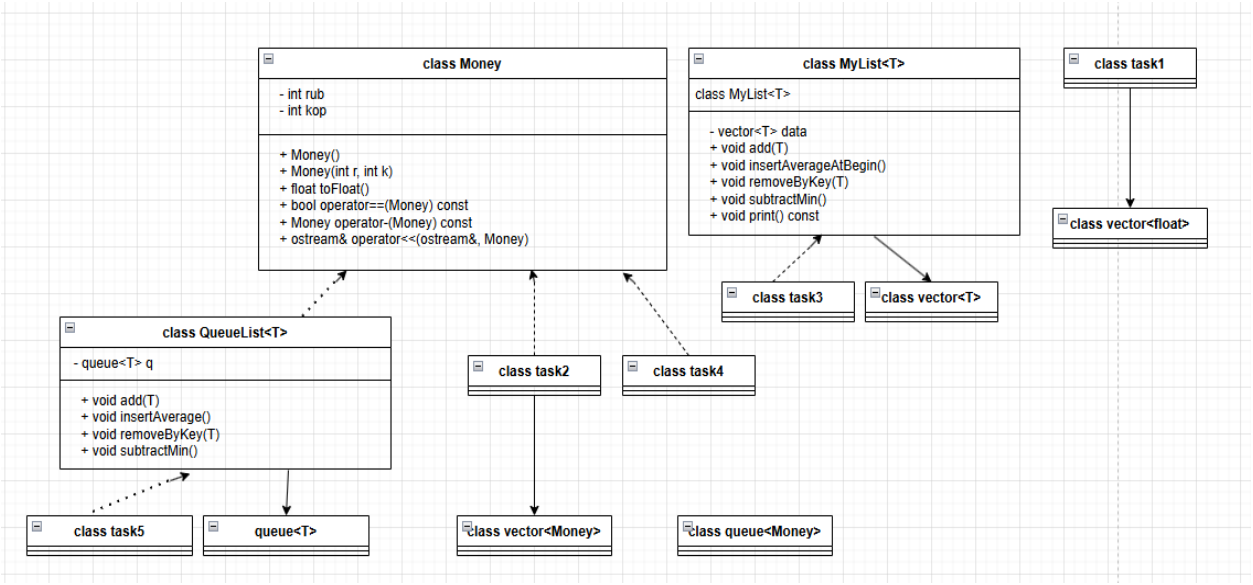
11лабапеределка (Глобальная область

```cpp
#pragma once
#include <iostream>
using namespace std;

class Money {
public:
    int rub;
    int kop;

    Money();
    Money(int r, int k);

    float toFloat() const;
    bool operator==(const Money& m) const;
    Money operator-(const Money& m) const;

    friend ostream& operator<<(ostream& os, const Money& m);
};
```

3 Результаты работы

```
Задача 1 (float): 0.875 0 3.6 1.4
Задача 2 (Money): 0.03 1.20 0.00
Задача 3 (MyList<float>): 0 0.666667 1.16667
Задача 4 (queue<Money>): 1.15 0.00 2.09
Задача 5 (QueueList<float>): 0 1.33333 0.333333
```

4 UML-диаграмма классов



5 Ответы на контрольные вопросы

```
// Контрольные вопросы и ответы по STL

// 1
// Из каких частей состоит библиотека STL?
// Ответ: Контейнеры, алгоритмы, итераторы.

// 2
// Какие типы контейнеров существуют в STL?
// Ответ: Последовательные, ассоциативные, неупорядоченные, адаптеры.

// 3
// Что нужно сделать для использования контейнера STL в своей программе?
// Ответ: Подключить заголовок и использовать std::.

// 4
// Что представляет собой итератор?
// Ответ: Объект для доступа и перемещения по элементам контейнера.

// 5
// Какие операции можно выполнять над итераторами?
// Ответ: Разыменование, инкремент/декремент, сравнение, арифметика.

// 6
// Каким образом можно организовать цикл для перебора контейнера с использованием итератора?
// Ответ:
// for (auto it = c.begin(); it != c.end(); ++it) { ... }

// 7
// Какие типы итераторов существуют?
// Ответ: Input, Output, Forward, Bidirectional, Random Access.

// 8
// Перечислить операции и методы общие для всех контейнеров.
// Ответ: begin(), end(), size(), empty(), clear(), insert(), erase().

// 9
// Какие операции являются эффективными для контейнера vector? Почему?
// Ответ: Доступ по индексу и push_back. Элементы в памяти подряд.

// 10
// Какие операции являются эффективными для контейнера list? Почему?
// Ответ: Вставка/удаление в любом месте. Двусвязный список.

// 11
// Какие операции являются эффективными для контейнера deque? Почему?
// Ответ: Вставка/удаление в начало и конец. Массив блоков.

// 12
// Методы vector: push_back, pop_back, insert, erase, clear, at, front, back, data.

// 13
// Методы list: push_front, push_back, pop_front, pop_back, insert, erase, remove, unique, sort, merge, reverse.

// 14
// Методы deque: push_front, push_back, pop_front, pop_back, insert, erase, clear, at, front, back.

// 15
// Удаление из vector со 2 по 5:
// v.erase(v.begin() + 1, v.begin() + 5);

// 16
// Удаление последнего из vector:
// v.pop_back();

// 17
// Удаление из list со 2 по 5:
// auto it1 = next(l.begin(), 1);
// auto it2 = next(l.begin(), 5);
// l.erase(it1, it2);

// 18
// Удаление последнего из list:
// l.pop_back();

// 19
// Удаление из deque со 2 по 5:
// d.erase(d.begin() + 1, d.begin() + 5);

// 20
// Удаление последнего из deque:
// d.pop_back();

// 21
// Печать контейнера с итератором:
// template<typename T>
// void print(const T& c) {
//     for (auto it = c.begin(); it != c.end(); ++it) std::cout << *it << " ";
// }
```

```cpp
// 21
// Печать контейнера с итератором:
// template<typename T>
// void print(const T& c) {
//     for (auto it = c.begin(); it != c.end(); ++it) std::cout << *it << " ";
// }

// 22
// Что такое адаптеры контейнеров?
// Ответ: Обёртки над контейнерами с ограниченным интерфейсом.

// 23
// Разница между stack<int> s и stack<int, list<int>> s?
// Ответ: Первый использует deque, второй list.

// 24
// Методы stack: push, pop, top, empty, size.

// 25
// Методы queue: push, pop, front, back, empty, size.

// 26
// Разница между queue и priority_queue?
// Ответ: queue – FIFO, priority_queue – по приоритету (heap).

// 27
// Удаление по номеру из stack:
// Переложить во временный стек, пропуская нужный.

// 28
// Удаление по номеру из queue:
// Переложить во временную очередь, пропуская нужный.

// 29
// Печать stack:
// template<typename T> void printStack(stack<T> s) {
//     while (!s.empty()) { std::cout << s.top() << " "; s.pop(); }
// }

// 30
// Печать queue:
// template<typename T> void printQueue(queue<T> q) {
//     while (!q.empty()) { std::cout << q.front() << " "; q.pop(); }
// }
```

6 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1