Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации Сибирский государственный университет телекоммуникаций и информатики

кафедра ТС и ВС

Лабораторная работа 6

по дисциплине

«Объектно-ориентированное программирование»

Выполнил: Демин С. А.
Группа: ИКС-433
Вариант: 5

Проверил: Нейдорф П. Я.

Новосибирск, 2025

Стандартная библиотека STL языка C++. Контейнерные классы стандартной библиотеки. Класс Очередь

Цель работы: Знакомство с контейнерными классами в C++ . Изучение класса deque в языке C++ . Знакомство с методами класса, Итераторами и функциями стандартной библиотеки STL для работы с контейнером deque.

Задание:

В соответствии с вариантом задания разработать программу создания и обработки динамической структуры данных очередь. Программу разработать с использованием методов стандартной библиотеки языка C++.

Вариант:

Вариант	Условие задачи
5	**Разработать класс для создания объектов, закладываемых в контейнер Очередь.** Составить программу, которая содержит текущую информацию о заявках на авиабилеты. Каждая заявка содержит: • пункт назначения; • номер рейса; • фамилию и инициалы пассажира; • желаемую дату вылета. **Программа должна обеспечивать:** - хранение всех заявок в виде очереди; - добавление заявок в очередь; - удаление заявок; - вывод всех заявок.

Текст программы

```
#include <iostream>
#include <deque>
#include <string>
using namespace std;

class bilet{
private:
   string nazv,number,fio,data;
public:
   bilet(string n,string num,string f, string d){nazv=n;number=num;fio=f;data=d;}
   void setNazv(string n){nazv=n;}
   void setNumber(string num){number=num;}
   void setFio(string f){fio=f;}
```

```
    void setData(string d){data=d;}

    string getNazv(){return nazv;}
    string getNumber(){return number;}
    string getFio(){return fio;}
    string getData(){return data;}};

int main() {
    deque<bilet> queue;
    int vibor;

    while (true){
        cout<<"\n1. Добавить заявку"<<endl;
        cout<<"2. Удалить заявку"<<endl;
        cout<<"3. Показать все"<<endl;
        cout<<"4. Выход"<<endl;
        cout<<"Выберите: ";
        cin>>vibor;
        cin.ignore();

        switch (vibor){
            case 1:{
                string nazv,number,fio,data;
                cout<<"Пункт назначения: ";getline(cin,nazv);
                cout<<"Номер рейса: ";getline(cin,number);
                cout<<"Пассажир: ";getline(cin,fio);
                cout<<"Дата вылета: ";getline(cin,data);
                queue.push_back(bilet(nazv,number,fio,data));
                cout<<"Заявка добавлена!\n";
                break;}

            case 2: {
                if (queue.empty()){cout<<"Очередь пуста!\n";break;}
                cout<<"Удалена заявка: "<<queue.front().getFio()<<endl;
                queue.pop_front();
                break;}

            case 3: {
                if (queue.empty()){cout<<"Очередь пуста!\n";break;}
                cout<<"\nВсе заявки:\n";
                int i=1;
                for (bilet t:queue){
                    cout<<i<<". ";
                    cout<<t.getNazv()<<" "<<t.getNumber()<<" "<<t.getFio()<<"
"<<t.getData()<<endl;
                    i++;}
                break;}

            case 4:return 0;
            default: cout << "Такого варианта НЕТ\n";}}}
```

Результаты:

```
1. Добавить заявку
2. Удалить заявку
```

3. Показать все
4. Выход
Выберите: 1
Пункт назначения: Новосибирск
Номер рейса: 666
Пассажир: Демин С.А.
Дата вылета: 06.06.06
Заявка добавлена!

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 1
Пункт назначения: Усть-Каменогорск
Номер рейса: 123
Пассажир: Криволапов Н.А.
Дата вылета: 01.01.2000
Заявка добавлена!

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 1
Пункт назначения: Улан-Батор
Номер рейса: 777
Пассажир: Синица М.А.
Дата вылета: 01.02.2030
Заявка добавлена!

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 3

Все заявки:
1. Новосибирск 666 Демин С. А. 06.06.06
2. Усть-Каменогорск 123 Криволапов Н.А. 01.01.2000
3. Улан-Батор 777 Синица М.А. 01.02.2030

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 3

Все заявки:
1. Новосибирск 666 Демин С. А. 06.06.06
2. Усть-Каменогорск 123 Криволапов Н.А. 01.01.2000
3. Улан-Батор 777 Синица М.А. 01.02.2030

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 2
Удалена заявка: Демин С. А.

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 3

Все заявки:
1. Усть-Каменогорск 123 Криволапов Н.А. 01.01.2000
2. Улан-Батор 777 Синица М.А. 01.02.2030

1. Добавить заявку
2. Удалить заявку
3. Показать все
4. Выход
Выберите: 4

Описание результатов:

В ходе выполнения лабораторной работы была разработана программа, реализующая очередь заявок на авиабилеты с использованием контейнера deque. Программа успешно выполняет основные операции: добавление заявок в конец очереди, удаление из начала очереди и отображение всех элементов. Тестирование показало корректную работу программы — данные сохраняются в порядке их добавления, а удаление происходит строго в соответствии с принципом FIFO. Ввод данных осуществляется с помощью getline, что позволяет работать со строками, содержащими пробелы.

Выводы:

Использование контейнера deque эффективно для задач, требующих организации очереди. Его преимущества включают быструю вставку и удаление элементов на обоих концах, а также простоту реализации. Программа продемонстрировала практическую применимость стандартной библиотеки STL для работы с динамическими структурами данных. Принцип FIFO был соблюден, что подтверждает корректность выбранного подхода.

Контрольные вопросы:

1. Понятие динамической структуры.

Динамическая структура данных — это структура, память для которой выделяется и освобождается во время выполнения программы, что позволяет её размеру и форме гибко меняться в отличие от статического массива. Элементы таких структур создаются в специальной области памяти (куче) и связываются между собой с помощью указателей, что обеспечивает эффективное использование ресурсов и удобство для реализации таких структур, как списки, деревья и графы.

2. Описание структурного типа.

Структурный тип (например, struct в языке C) — это составной тип данных, позволяющий объединить несколько переменных, возможно

разных типов, под одним именем для удобства представления составного объекта. Каждая переменная внутри структуры называется полем и описывает определенное свойство объекта, что делает код более организованным и читаемым, например, для создания элемента списка можно объединить в структуру поля data и next.

3. Описание указателя.

Указатель — это переменная, значением которой является адрес ячейки памяти, где хранится другая переменная или структура данных. Он не содержит сами данные, а лишь ссылается на них, что позволяет косвенно манипулировать данными, динамически выделять память и строить связи между элементами динамических структур, как в стеке, где указатель на вершину используется для отслеживания текущего положения.

4. Основные операции, производимые со очередью.

Очередь работает по простому принципу «первый пришел — первый ушел», как в обычной жизни. С ней можно выполнять основные действия: добавить новый элемент в конец, забрать элемент из начала, посмотреть на первый элемент без его удаления, проверить, пуста ли очередь, и узнать ее текущий размер. При этом нельзя влезать в середину или удалять элементы оттуда — все операции происходят только с начала или конца, что сохраняет строгий порядок обработки элементов.