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

кафедра ТС и ВС

Лабораторная работа 5

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 Список

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

Задание:

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

Вариант:

Вариант	Условие задачи
5	**Для создания списка со структурированной информацией разработать класс для создания объектов с такой структурой.** На междугородной телефонной станции картотека абонентов содержит сведения в следующем виде: • Номер телефона • ФИО владельца • Паспортные данные (номер); **Составить программу, которая:** • обеспечивает начальное формирование картотеки в виде линейного списка, упорядоченного по алфавиту; • производит вывод всей картотеки; • вводит номер телефона и выдает имя владельца;

Текст программы

```
#include <iostream>
#include <string>
#include <list>

using namespace std;

class Abonent {
public:
   string phone;
   string name;
   string passport;

   Abonent(string p,string n,string pass){
      phone=p;
```

```
        name=n;
        passport=pass;}

    void display(){
        cout<<"Телефон: "<<phone<<endl;
        cout<<"ФИО: "<<name<<endl;
        cout<<"Паспорт: "<<passport<<endl;
        cout<<"------------------------"<<endl;}};

int main() {
    list<Abonent> abonents;
    int n;

    cout<<"Введите количество абонентов: ";
    cin>>n;
    cin.ignore();

    for (int i=0;i<n;i++){
            string phone, name, passport;
            cout<<"\nАбонент "<<i+1<<":"<<endl;
            cout<<"Номер телефона: ";
            getline(cin, phone);
            cout<<"ФИО: ";
            getline(cin, name);
            cout<<"Паспорт: ";
            getline(cin, passport);

            Abonent newAbonent(phone, name, passport);

            auto it=abonents.begin();
            while (it!=abonents.end()&&it->name<name){it++;}
            abonents.insert(it, newAbonent);}

    cout<<"\nОТСОРТИРОВАННЫЙ СПИСОК"<<endl;
    for (Abonent a:abonents){a.display();}

    int searchCount;
    cout<<"\nВведите количество номеров для поиска: ";
    cin>>searchCount;
    cin.ignore();

    for (int i=0;i<searchCount;i++){
        string Phone;
        cout<<"Введите номер телефона для поиска "<<i+1<<": ";
        getline(cin,Phone);

        bool found=false;
        for (Abonent a:abonents){
            if (a.phone==Phone){
                cout<<"Владелец: "<<a.name<<endl;
                found=true;
                break;}}

        if (!found){cout<<"Абонент не найден!"<<endl;}}}
```

Результаты:

Введите количество абонентов: 5

Абонент 1:
Номер телефона: 123
ФИО: First
Паспорт: 666

Абонент 2:
Номер телефона: 321
ФИО: Thirt
Паспорт: 111

Абонент 3:
Номер телефона: 678
ФИО: Forth
Паспорт: 148

Абонент 4:
Номер телефона: 999
ФИО: Fifth
Паспорт: 777

Абонент 5:
Номер телефона: 888
ФИО: Thirt
Паспорт: 123

ОТСОРТИРОВАННЫЙ СПИСОК
Телефон: 999
ФИО: Fifth
Паспорт: 777

Телефон: 123
ФИО: First
Паспорт: 666

Телефон: 678
ФИО: Forth
Паспорт: 148

Телефон: 888
ФИО: Thirt
Паспорт: 123

Телефон: 321
ФИО: Thirt

Паспорт: 111

Введите количество номеров для поиска: 3
Введите номер телефона для поиска 1: 666
Абонент не найден!
Введите номер телефона для поиска 2: 678
Владелец: Forth
Введите номер телефона для поиска 3: 321
Владелец: Thirt

Описание результатов:

Программа продемонстрировала работу со списком абонентов, содержащим записи о номере телефона, ФИО и паспортных данных. Было введено 5 записей, после чего список был отсортирован по полю ФИО в алфавитном порядке. Сортировка выполнена корректно - записи расположены от "Fifth" до "Thirt", при этом для одинаковых ФИО "Thirt" сохранен исходный порядок следования записей. Затем выполнен поиск по номеру телефона для трех различных случаев: номер 666 не найден (что корректно), номера 678 и 321 найдены успешно с выводом соответствующих владельцев. Целостность данных после сортировки сохранена - связи между полями каждого абонента не нарушены.

Выводы:

Программа эффективно реализует основные операции работы со списками: хранение структурированных данных, сортировку по текстовому полю и поиск по ключевому полю. Использование номера телефона в качестве уникального идентификатора для поиска является логичным решением, поскольку в реальных системах номер телефона обычно уникален. Сортировка по ФИО обеспечивает удобный алфавитный просмотр данных. Обработка случая отсутствия данных (номер 666) демонстрирует надежность алгоритма поиска.

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 списками.

Основные операции, производимые со списками, включают: добавление элемента в конец или в определенную позицию, удаление элемента по индексу или значению, поиск элемента, получение элемента по индексу, изменение элемента по индексу, проверку наличия элемента в списке, определение длины списка, сортировку элементов, очистку всего списка, а также обход всех элементов списка для выполнения операций над ними.