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

кафедра ТС и ВС

Лабораторная работа 7

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 Стек

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

Задание:

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

Вариант:

Вариант	Условие задачи
5	**Составить программу, которая:** • обеспечивает первоначальный ввод строки символов и формирует из символов стек; • затем, используя стек, позволяет провести проверку строки на симметричность относительно символа ‘*’; • если этот символ отсутствует, вывести об этом сообщение;

Схема алгоритма

Начало
↓
Ввести строку
↓
Создать пустой стек
↓
Читать строку по символам:
- **Если символ ≠ '*' → добавить в стек**
- **Если символ = '*' → остановиться**

↓
Была ли '*'?
|----- **Нет → Вывести "Символ '*' отсутствует" → Конец**
|-----**Да → Продолжить**
↓
Читать оставшиеся символы после '*':
- **Брать символ из строки и верхний из стека**
- **Если не равны → не симметрично**
- **Если равны → убрать верхний из стека**

↓
Проверить:
|----- **Все символы совпали И стек пуст → "Симметрично"**
|----- **Иначе → "Не симметрично"**
↓
Конец

Текст программы

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

int main() {
    string stroka;
    stack<char> stack;
    bool star=false;
    int starPol=0;
    bool symetrichno=true;

    cout<<"Введите строку: ";
    getline(cin,stroka);

    for (int i=0;i<stroka.length();i++) {
        char c=stroka[i];
        if (c=='*'){starPol=i;star=true;break;}
        stack.push(c);}

    if (star==false){cout<<"Символ * отсутствует"<<endl;return 0;}

    for (int i=starPol+1;i<stroka.length();i++){
        if (stack.empty()||stroka[i]!=stack.top()){
            symetrichno=false;
            break;}
        stack.pop();}

    if (symetrichno&&stack.empty()){cout<<"Строка симметрична"<<endl;
    }else{cout<<"Строка не симметрична"<<endl;}}
```

Результаты:

Введите строку: demin*nimed
Строка симметрична

Введите строку: demin*demin
Строка НЕ симметрична

Введите строку: deminnimed
Символ '*' отсутствует

Описание результатов:

Программа успешно выполняет проверку симметричности введенной строки символов относительно символа <*>. В ходе тестирования было подтверждено, что алгоритм корректно обрабатывает различные сценарии. При вводе строки "demin*nimed" программа правильно идентифицирует ее как симметричную, поскольку последовательность символов до звездочки полностью совпадает с обратной последовательностью после звездочки. Для

строки "demin*demin" программа обоснованно определяет отсутствие симметрии из-за несовпадения соответствующих последовательностей. В случае строки без звездочки "deminnimed" программа адекватно реагирует выводом сообщения об отсутствии ключевого символа.

Выводы:

Стек из стандартной библиотеки C++ удобно использовать для таких задач. Он хорошо подходит для проверки симметрии, так работает по принципу "последний зашел - первый вышел". Программа получилась простой и понятной. STL контейнеры экономят время, так как не нужно писать свои реализации структур данных. Алгоритм работает быстро даже для длинных строк. Такую проверку симметрии можно использовать в реальных задачах, например, при анализе текста или проверке правильности расстановки символов.

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 стеками.

Основными операциями со стеком, реализующими принцип LIFO (последним пришел — первым ушел), являются push (добавление) и pop (удаление) элемента с вершины. При push указатель вершины смещается, и новый элемент помещается в стек, а при pop элемент на вершине извлекается, и указатель смещается обратно, обеспечивая доступ только к самому верхнему элементу для манипуляций.