**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

**Тема: «Динамические структуры данных»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Жучков О.Д. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

## Цель работы

Изучить различные динамические структуры данных и их особенности. На языке C++ реализовать стек на базе списка с использованием ООП.

## Задание

Требуется написать программу, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе **списка.**

**1)**Реализовать **класс** CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных ***int.***

Структура класса узла списка:

struct ListNode {

ListNode\* mNext;

int mData;

};

Объявление класса стека:

class CustomStack {

public:

// методы push, pop, size, empty, top + конструкторы, деструктор

private:

// поля класса, к которым не должно быть доступа извне

protected: // в этом блоке должен быть указатель на голову

ListNode\* mHead;

};

Перечень методов класса стека, которые должны быть реализованы:

* **void push(int val)** -  добавляет новый элемент в стек
* **void pop()** - удаляет из стека последний элемент
* **int top()** - доступ к верхнему элементу
* **size\_t size()**- возвращает количество элементов в стеке
* **bool empty()** - проверяет отсутствие элементов в стеке

**2)** Обеспечить в программе считывание из потока ***stdin***последовательности (не более 100 элементов) из чисел и арифметических операций (+, -, \*, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

* Если очередной элемент входной последовательности - число, то положить его в стек,
* Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже),
* Если входная последовательность закончилась, то вывести результат (число в стеке).

Если в процессе вычисления возникает ошибка:

* например вызов метода **pop**или **top** при пустом стеке (для операции в стеке не хватает аргументов),
* по завершении работы программы в стеке более одного элемента,

программа должна вывести "**error**" и завершиться.

**Примечания:**

1. Указатель на голову должен быть protected.
2. Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.
3. Предполагается, что пространство имен std уже доступно.
4. Использование ключевого слова using также не требуется.
5. Структуру **ListNode** реализовывать самому не надо, она уже реализована.

## Выполнение работы

## Класс CustomStack:

## В области private находится mSize, хранящий размер стека.

## В области protected находится ссылка mHead на голову стека.

## В public находятся методы:

## CustomStack() – конструктор, присваивает нулевые значения полям.

* push(int val) - добавляет новый элемент в стек.
* pop() - удаляет из стека последний элемент, если возможно.
* top() - доступ к верхнему элементу, если возможно.
* size() - возвращает количество элементов в стеке.
* empty() - проверяет отсутствие элементов в стеке.

В основной части программы происходит считывание данных через пробел до символа конца строки. Если введено число, оно добавляется в стек. Если введён один из символов “+”, “-“, “\*”, “/”, то над двумя верхними элементами стека производится соответствующая операция; получившееся число добавляется в стек. При завершении ввода выводится значение в голове стека или ошибка, если в стеке более одного элемента.

## Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарии |
| 1. | 1 2 + 3 4 - 5 \* + | -2 | Программа работает корректно. |
|  |  |  |
| 2. | 1 + 5 3 - | error |
| 3. | -12 -1 2 10 5 -14 17 17 \* - - + - \* + | 304 |

## Выводы

В ходе выполнения лабораторной работы был изучен синтаксис языка C++ для работы с классами, с использованием чего реализована динамическая структура стек на основе списка и методы работы с ней.

# Приложение А Исходный код программы

Название файла: main.cpp

class CustomStack {

public:

CustomStack(){

mSize = 0;

mHead = nullptr;

}

void push(int val){

ListNode\* temp = new ListNode;

temp -> mData = val;

temp -> mNext = mHead;

mSize++;

mHead = temp;

}

void pop(){

if (empty()){

cout << "error";

exit(0);

}

mHead = mHead -> mNext;

mSize--;

}

int top(){

if (empty()){

cout << "error";

exit(0);

}

return mHead -> mData;

}

size\_t size(){

return mSize;

}

bool empty(){

return mSize == 0;

}

private:

size\_t mSize;

protected:

ListNode\* mHead;

};

int main(){

CustomStack stk;

string InputElem;

int a, b;

while (cin.peek() != '\n')

{

cin >> InputElem;

if (InputElem == "+" || InputElem == "-" || InputElem == "\*" || InputElem == "/"){

b = stk.top();

stk.pop();

a = stk.top();

stk.pop();

if (InputElem == "+")

stk.push(a + b);

else if (InputElem == "-")

stk.push(a - b);

else if (InputElem == "\*")

stk.push(a \* b);

else if (InputElem == "/")

stk.push(a / b);

}

else stk.push(stoi(InputElem));

}

if (stk.size() > 1){

cout << "error";

exit(0);

}

cout << stk.top();

return 0;

}