**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. 3344 |  | Волков А.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Познакомиться с основами языка С++.

Изучить основные динамические структуры данных.

Рассмотреть парадигму ООП в языке С++.

Реализовать класс стека и методы для работы с его объектами.

Решить поставленную задачу, применив созданную структуры данных.

## Задание

Вариант 1.

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

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

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

• extend(int n) - расширяет исходный массив на n ячеек

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: программа должна вывести "error" и завершиться.

Примечания:

• Указатель на массив должен быть protected.

•Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.

• Предполагается, что пространство имен std уже доступно.

• Использование ключевого слова using также не требуется.

## Выполнение работы

Рассматривая реализацию класса CustomStack, стоит обратить внимание на некоторые детали.

Среди приватных атрибутов находится поле, которое хранит текущий размер стека. Это сделано с целью скрыть от пользователя возможность изменять важные для работы стека как структуры данных детали реализации.

Публичные методы, требуемые в задании, реализуются тривиально, стоит отметить только проверку на крайние случаи в методах push и pop, где может подаваться элемент для добавления в стек без головного элемента, а также запрашиваться удаление элемента пустого стека соответственно.

В функции main() создаётся объект класса CustomStack. Происходит считывание строки с помощью функции fgets(). Далее с помощью функции strtok() строка разбивается на токены (разделителем является пробел или символ новой строки). В цикле while происходит обработка каждого токена, который был получен из входной строки. Если токен является одной из операций (+, -, \*, /), то выполняется соответствующее арифметическое действие над двумя верхними числами из стека, и результат помещается обратно в стек. Если же токен является числом, то оно добавляется в стек.

**Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | 1 2 + 3 4 - 5 \* + | -2 | Данные корректно обрабатываются. |
| 2. | 1 -10 - 2 \* | 22 | Данные корректно обрабатываются. |
| 3. | 1 5 \* \* | error | Данные корректно обрабатываются. |
| 4. | 5 5 5 \* \* | 125 | Данные корректно обрабатываются. |

## Выводы

Были изучены основы языка программирования С++ и новые динамические структуры данных.

Была рассмотрена парадигма ООП в языке C++.

Реализован класс CustomStack и методы для работы с его объектами, которые представляют собой сте.

Решена задача с использованием созданного класса.

Расширение языка Си в виде C++ кажется очень логичным, так как парадигма ООП значительно увеличивает возможности для реализации новых типов данных, а также методов для работы с ним, имея при этом гораздо более безопасную систему взаимодействия пользователя с созданной сущностью.

# Приложение А Исходный код программы

Название файла: lb\_4.c

class CustomStack

{

public:

CustomStack()

{

mSize = 0;

mCapacity = 100;

mData = new int[mCapacity];

}

~CustomStack()

{

delete []mData;

}

void push(int val)

{

if (mSize == mCapacity)

extend(100);

mData[mSize++] = val;

}

void pop()

{

if (empty()) {

cout << "error" << endl;

return;

}

--mSize;

}

int top()

{

if (empty()) {

cout << "error" << endl;

exit(1);

}

return mData[mSize - 1];

}

size\_t size()

{

return mSize;

}

bool empty()

{

return mSize == 0;

}

void extend(int n)

{

int \*newData = new int[mCapacity + n];

for (int i = 0; i < mSize; i++) {

newData[i] = mData[i];

}

delete []mData;

mData = newData;

mCapacity += n;

}

protected:

int \*mData;

private:

size\_t mSize;

size\_t mCapacity;

};

int main() {

CustomStack stack;

char str[101];

fgets(str, 100, stdin);

char \*token = strtok(str, " \n");

while (token) {

string tokens(token, strlen(token));

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

if (stack.size() < 2) {

cout << "error" << endl;

return 0;

}

int num1 = stack.top();

stack.pop();

int num2 = stack.top();

stack.pop();

int result;

switch (tokens[0]) {

case '+':

result = num2 + num1;

break;

case '-':

result = num2 - num1;

break;

case '\*':

result = num2 \* num1;

break;

case '/':

if (num1 == 0) {

cout << "error" << endl;

return 0;

}

result = num2 / num1;

break;

}

stack.push(result);

} else {

stack.push(stoi(tokens));

}

token = strtok(NULL, " \n");

}

if (stack.size() != 1) {

cout << "error" << endl;

return 0;

}

cout << stack.top() << endl;

return 0;

}