**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 |  | Щербак М.С. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы изучить и использовать базовые механизмы языка С++, необходимые для реализации стека и очереди.

## Задание

Вариант 1. Стековая машина.

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: 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.

## Выполнение работы

### Класс CustomStack:

1. \*\*Определение класса:\*\*

- В классе CustomStack определены приватные члены std::stack<int> stack\_, которые представляют стек целых чисел.

2. \*\*Методы класса:\*\*

- push(int num): Добавляет целое число num в вершину стека.

- pop(): Удаляет элемент из вершины стека.

- top(): Возвращает значение элемента на вершине стека.

- size(): Возвращает количество элементов в стеке.

### Функция main:

1. \*\*Инициализация объекта:\*\*

- Создается объект CustomStack stack, который представляет стек для хранения чисел.

2. \*\*Чтение входных данных:\*\*

- Считывается строка str с помощью fgets, содержащая выражение с операндами и операторами.

- Строка разбивается на токены с помощью strtok.

3. \*\*Обработка токенов:\*\*

- Проверяется каждый токен на оператор (+, -, \*, /) или операнд (целое число).

- Если токен является оператором, то извлекаются два числа из стека, выполняется соответствующая операция и результат помещается обратно в стек.

- Если токен является операндом, он добавляется в стек.

4. \*\*Проверка корректности выражения:\*\*

- Проверяется, что в конце выполнения программы в стеке осталось только одно число.

- Если стек пуст или содержит больше одного элемента, выводится сообщение "error".

5. \*\*Вывод результата:\*\*

- Если выражение было корректным, выводится результат вычислений, находящийся на вершине стека.

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 1 2 + 3 4 - 5 \* + | -2 | Данные обработаны корректно. |
|  | 2 3 4 + + + | error | Данные обработаны корректно. |
|  | 3 4 5 + + | 12 | Данные обработаны корректно. |

## Выводы

Были изучена работа с базовыми механизмами языка C++. Была создана программа, в которой реализован стек на основе динамического массива.

# Приложение А Исходный код программы

Название файла: lb4.c

#include <iostream>

#include <string>

#include <cstring> // Для функции strtok

#include <cstdlib> // Для функции exit

#include <sstream>

class CustomStack {

public:

CustomStack(){

mSize=0; //кол-во эл-тов в стеке

mCapacity=100; //размер массива данных стека

mData=new int[mCapacity]; // указатель на массив целых чисел

}

~CustomStack(){

delete[] mData;

}

void push(int val) {

if (mSize == mCapacity) {

extend(100); // Расширяем массив при необходимости

}

mData[mSize++] = val;

}

void pop() {

if (mSize == 0) {

std::cout << "error" << std::endl;

exit(1); // Завершить программу с кодом ошибки

}

if (mSize > 0) {

--mSize;

}

}

int top(){

if (empty()) {

std::cout << "error" << std::endl;

exit(1);

}

return mData[mSize - 1];

}

bool empty(){

return mSize == 0;

}

size\_t size() {

return mSize;

}

void extend(int n) {

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

for (size\_t 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) {

std::string tokens(token, strlen(token));

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

if (stack.size() < 2) {

std::cout << "error" << std::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) {

std::cout << "error" << std::endl;

return 0;

}

result = num2 / num1;

break;

}

stack.push(result);

} else {

stack.push(std::stoi(tokens));

}

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

}

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

std::cout << "error" << std::endl;

return 0;

}

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

return 0;

}