**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. 3342 |  | Львов А.В. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью данной работы является ознакомление с такой структурой данных, как стек и его реализация на языке С++.

## Задание

Стековая машина.

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.

## Выполнение работы

Класс CustomStack содержит следующие public функции:

* Конструктор класса присваивает полям класса значения: mHead – nullptr, mSize – 0.
* Функция push (void push(int value)) создает новый элемент списка, инициализируя значение поля mData, равным value, переданным в функцию и присваивает полю mHead класса указатель на созданный элемент.
* Функция pop (void pop()) в случае, если стек не пустой, присваивает полю mHead стека указатель на следующий за ним элемент и удаляет старую «голову», а иначе выводит ошибку и завершает программу.
* Функция top (int top() const) возвращает значение головного элемента стека, если он не пустой, а иначе выводит ошибку и завершает программу.
* Функция size (size\_t size() const) возвращает количество элементов стека.
* Функция empty (bool empty() const) возвращает true, если список не пустой, иначе – false.
* Деструктор класса проходит по всем элементам стека и освобождает память, выделенную для них.

Функция compute (void compute(CustomStack \* s, const char \* op)) получает на вход указатель на объект класса CustomStack и операцию, которую необходимо применить к двум последним элементам стека, а также добавляет полученный результат в стек.

В функции main создается объект класса CustomStack, строка str, в которую будет передана последовательность чисел и операций. Затем, разбив строку по пробелу и переносу строки с помощью цикла while происходят вычисления согласно заданию. После завершения цикла происходит проверка на количество элементов в стеке и вывод результата. Затем, память, выделенная с помощью new для объекта класса CustomStack и str освобождается с помощью delete.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 1 2 + 3 4 - 5 \* + | -2 |
|  | 1 + 5 3 - | error |
|  | -12 -1 2 10 5 -14 17 17 \* - - + - \* + | 304 |

## Выводы

Было проведено ознакомление с стеком, были реализованы функции для взаимодействия с ним на языке С++.

# Приложение А Исходный код программы

Название файла: main.cpp

#define BUF\_SIZE 1024

void printError() {

    printf("error");

    exit(0);

}

class CustomStack {

public:

    CustomStack(){

        mHead = nullptr;

        mSize = 0;

    }

    void push(int value) {

        auto lNode = new ListNode;

        lNode -> mData = value;

        lNode -> mNext = mHead;

        mHead = lNode;

        mSize++;

    }

    void pop() {

        if ( !(empty()) ) {

            ListNode \*tmp = mHead -> mNext;

            delete mHead;

            mHead = tmp;

        } else {

           printError();

        }

        mSize--;

    }

    int top() const {

        if (empty()) {

            printError();

        } else {

            return mHead -> mData;

        }

    }

    size\_t size() const {

        return mSize;

    }

    bool empty() const {

        return mSize == 0;

    }

    ~CustomStack() {

        ListNode \*tmp = mHead;

        while (tmp != nullptr) {

            tmp = tmp -> mNext;

            delete mHead;

            mHead = tmp;

        }

    }

private:

    size\_t mSize;

protected:

    ListNode\* mHead{};

};

void compute(CustomStack \* s, const char \* op) {

    int num2 = s -> top();

    s -> pop();

    int num1 = s -> top();

    s -> pop();

    int res;

    switch (\*op) {

        case '+':

            res = num1 + num2;

            break;

        case '-':

            res = num1 - num2;

            break;

        case '\*':

            res = num1 \* num2;

            break;

        case '/':

            res = num1 / num2;

            break;

        default:

            return;

    }

    s -> push(res);

}

int main() {

    auto s = new CustomStack;

    auto str = new char[BUF\_SIZE];

    fgets(str, BUF\_SIZE, stdin);

    char \* p;

    p = strtok(str, " \n");

    while (p != nullptr) {

        if (strlen(p) == 1) {

            if (!isdigit(\*p) && (\*p != '\n')) {

                compute(s, p);

            } else {

                s -> push(stoi(p));

            }

        } else {

            if (isdigit(\*p) || \*p == '-') {

                s -> push(stoi(p));

            }

        }

        p = strtok(nullptr, " \n");

    }

    if (s -> size() != 1) {

        printError();

    } else {

        printf("%d", s->top());

    }

    delete s;

    delete[] str;

}