**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 |  | Коршунов П.И. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Изучение и получение навыков для работы с ООП на языке C++, реализация собственного стека.

## Задание.

Вариант 3

Моделирование стека. Требуется написать программу, моделирующую работу стека на базе массива. Для этого необходимо:

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

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 массив данных

int\* mData;

};

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() - возвращает верхний элемент

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 команд (каждая команда с новой строки), в зависимости от которых программа выполняет ту или иную операцию и выводит результат ее выполнения с новой строки.

Перечень команд, которые подаются на вход программе в stdin:

cmd\_push n - добавляет целое число n в стек. Программа должна вывести "ok"

cmd\_pop - удаляет из стека последний элемент и выводит его значение на экран

cmd\_top - программа должна вывести верхний элемент стека на экран не удаляя его из стека

cmd\_size - программа должна вывести количество элементов в стеке

cmd\_exit - программа должна вывести "bye" и завершить работу

Если в процессе вычисления возникает ошибка (например вызов метода pop или top при пустом стеке), программа должна вывести "error" и завершиться.

## Выполнение работы

Был реализован стек CustomStack c членами данных mData(protected), capacity и len(private) (массив данных, объем массива и количество символов). Создан конструктор, который инициализирует массив и объем массива(в данном случае, 20) и устанавливает начальное значение длины и деструктор, который освобождает память, выделенную для массива. Реализованы такие методы стека как: push, который добавляет элемент в стек, pop, который удаляет из стека последний элемент, top, который возвращает верхний элемент стека, empty, который проверяет на заполненность стека и extend, который увеличивает стек на n элементов. В главной функции создается экземпляр класса и строка, в которую записывается опция. Следом, бесконечным циклом, проверяется на соответствие доступным командам. Если все корректно, команда выполняется, иначе выводится error. Предусмотрены обработки ошибок при пустом стеке.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | cmd\_push 1 cmd\_top cmd\_push 2 cmd\_top cmd\_pop cmd\_size cmd\_pop cmd\_size cmd\_exit | ok 1 ok 2 2 1 1 0 bye |  |

## Выводы

Были изучены и получены навыки для работы с ООП на языке C++, была реализована собственная структура - стек.

# Приложение А Исходный код программы

Название файла: main.cpp

class CustomStack {

public:

CustomStack() : mData (new int[20])

, capacity (20)

, len (0)

{}

~CustomStack() {delete[] mData;}

void push(int val)

{

if (len == capacity) extend(1);

mData[len++] = val;

}

void pop() {len--;}

int top() {return mData[len - 1];}

size\_t size() {return len;}

bool empty() {return len == 0;}

void extend(size\_t n)

{

capacity += n;

int\* temp = new int[capacity];

memcpy(temp, mData, (capacity \* sizeof(int)));

delete[] mData;

mData = temp;

}

protected:

int\* mData;

private:

size\_t capacity;

size\_t len;

};

int main() {

CustomStack stack;

string option;

while (true) {

cin >> option;

if (option == "cmd\_push") {

int n;

cin >> n;

stack.push(n);

cout << "ok" << '\n';

}

else if (option == "cmd\_pop") {

if(stack.empty()) {

cout << "error\n";

break;

}

cout << stack.top() << '\n';

stack.pop();

}

else if (option == "cmd\_top") {

if(stack.empty()) {

cout << "error\n";

break;

}

cout << stack.top() << '\n';

}

else if (option == "cmd\_size") {

cout << stack.size() << '\n';

}

else if (option == "cmd\_exit") {

cout << "bye";

break;

} else cout << "error";

}

return 0;

}