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 _____ Белаид Фарук

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Изучить основы объектно-ориентированного программирования на языке C++. Освоить принципы работы с динамическими структурами данных. Реализовать в виде класса динамическую структуру данных – стек на базе массива - в C++.

Задание

Требуется написать программу, моделирующую работу 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 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.
-

Выполнение работы

1) Класс **CustomStack** имеет следующие поля:

- **int* mData** – массив целых чисел, основа для стека.
- **int mSize** – количество элементов в стеке.
- **int topIndex** – индекс верхнего элемента стека.

А также следующие методы:

- **CustomStack()** – конструктор класса. Инициализирует переменные *mSize* = 0, topIndex = 1, mData = nullptr.
- **void push(int val)** – добавляет переданный аргумент *val* в стек. В случае, если под новый элемент недостаточно места, вызывает метод *extend()* для выделения дополнительной памяти.
- **void pop()** – удаляет последний добавленный элемент стека путем декрементации переменных mSize и topIndex. Если в стеке нет ни одного элемента (empty() == true), то выводит в консоль сообщение об ошибке и завершает работу программы.
- **bool empty()** – возвращает *true*, если стек пуст, и *false* в противном случае.
- **int top()** – возвращает верхний элемент стека. Если стек пуст, выводит в консоль сообщение об ошибке и завершает работу программы.
- **size_t size()** – возвращает количество элементов в стеке.
- **void extend(int n)** – приватный метод, позволяющий выделить в массиве еще n ячеек для элементов стека.

2) В функции **main():**

- Создается объект класса *CustomStack*.
- В строку *str* с помощью функции *scanf* стандартной библиотеки ввода записывается введенная пользователем последовательность операторов чисел.

- В цикле while с помощью функции стандартной библиотеки Си *strcmp* сопоставляются введённые строки с доступными командами. В случае совпадения к объекту класса *CustomStack* применяется тот или иной метод, реализованный внутри этого класса.
- Затем, после обработки соответствующей введённым строке и числу команде, с помощью функции memset стандартной библиотеки Си строка *str* “обнуляется” с целью корректной записи последующих строк-команд.
- Затем, при условии корректного завершения цикла, в консоль выводится сообщение *“bye”* и программа завершает свою работу.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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	Ответ верный.
2.	cmd_push 200 cmd_top cmd_pop cmd_pop	Ok 200 200 error	Ответ верный.

Выводы

В ходе выполнения лабораторной работы был изучен принцип работы такой динамической структуры данных, как стек. Была создана программа на языке C++, реализующая класс стека на базе массива, включая все необходимые функции для работы с ним, а также способная по введенной пользователем последовательности операторов производить добавление, удаление, проверку размера, печать верхнего элемента и размера стека.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *main.cpp*

```
class CustomStack {
public:
    CustomStack() : mData(nullptr), mSize(0), topIndex(-1) {}

    ~CustomStack() {
        delete[] mData;
    }

    void push(int val) {
        topIndex++;
        if ((!mData) || (topIndex == mSize)) {
            if (!extend(1)) {
                // Memory allocation failed
                return;
            }
        }
        mData[topIndex] = val;
    }


    void pop(){
        topIndex--;
        mSize--;
    }

    size_t size(){
        return mSize;
    }

    bool empty(){
        if (topIndex == -1) return true;
        return false;
    }



    int top(){
        return mData[topIndex];

    }

private:
    int topIndex;
    size_t mSize;

    bool extend(int n) {
        int* new_arr = new (std::nothrow) int[mSize + n];
        if (new_arr == nullptr) {
            // Memory allocation failed
            return false;
```

```
        }

        if (mData) {
            for (size_t i = 0; i < mSize; i++) {
                new_arr[i] = mData[i];
            }
            delete[] mData;
        }
        mData = new_arr;
        mSize += n;
        return true;
}



protected:
    int* mData;
};

int main() {
    char str[256];
    int num;
    CustomStack stack;
    scanf("%s", str);
    while(strcmp(str, "cmd_exit") != 0){
        if(strcmp(str, "cmd_push") == 0){
            scanf("%d", &num);
            stack.push(num);
            if (!stack.empty()) {
                cout << "ok" << endl;
            } else {
                cout << "error" << endl;
                exit(0);
            }
        }

        else if(strcmp(str, "cmd_pop") == 0){
            if(stack.empty()){
                cout << "error" << endl;
                exit(0);
            }
            cout << stack.top() << endl;
            stack.pop();
        }
        else if(strcmp(str, "cmd_top") == 0) {
            if(stack.empty()){
                cout << "error" << endl;
                exit(0);
            }
            cout << stack.top() << endl;
        }

        else if(strcmp(str, "cmd_size") == 0) cout << stack.size() <<
endl;

        memset(str, 0, 256);
        scanf("%s", str);
```

```
    }
    cout << "bye" << endl;

    return 0;
}
```