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: Введение в язык C++

Студентка гр. 3343 _____ Волох И.О.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Целью работы является изучение основных механизмов языка C++ путем разработки структур данных стека и очереди на основе динамической памяти.

Для достижения поставленной цели требуется решить следующие задачи:

– ознакомиться со структурами данных стека и очереди, особенностями их реализации;

– изучить и использовать базовые механизмы языка C++, необходимые для реализации стека и очереди;

– реализовать индивидуальный вариант стека в виде C++ класса, его операции в виде функций этого класса, ввод и вывод данных программы.

Задание

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

В ходе выполнения лабораторной работы был написан класс CustomStack с такими методами, как

- конструктор CustomStack(), который присваивает полю size_(0), mData(NULL).
- конструктор CustomStack(), который вызывает другой конструктор.
- деструктор ~CustomStack(), который очищает выделенную под массив память.
- метод void push(int val) проверяет, что размера стека достаточно для нового элемента, увеличивает значение поля, отвечающего за индекс, и помещает по номеру индекса элемент в массив.
- метод void pop() проверяет, что массив не пуст, иначе выводится сообщение об ошибке и завершается программа, а так же уменьшает значение поля mIndex на один, как бы удаляя последний элемент.
- метод int top() проверяет, что массив не пуст, иначе выводится сообщение об ошибке и завершается программа, а так же возвращает значение последнего элемента массива.
- метод size_t size() const возвращает размер стека, то есть индекс последнего элемента, увеличенный на один.
- метод void extend(int n), который расширяет размер стека на n.
- метод bool empty() проверяет пустой ли стек.

В main’е создается экземпляр класса CustomStack и с помощью цикла while считываются команды пользователя, в зависимости от которых вызываются определенные методы для стека, пока не поступит команда "cmd_exit". После цикла в консоль выводится сообщение "bye".

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	Выходные данные корректны.
2.	cmd_push 1 cmd_top cmd_pop cmd_size cmd_pop	ok 1 1 0 error	Случай с некорректным удалением обрабатывается корректно.

Выводы

Были подробно изучены такие структуры данных, как стек и очередь и требующиеся для их реализации механизмы языка C++. В соответствии с вариантом лабораторной работы был написан класс CustomStack, моделирующий поведение стека на основе массива.

Приложение А

Исходный код программы

Название файла: main.c

```
class CustomStack {
public:
    CustomStack();
    ~CustomStack();
    void push(int n);
    void pop();
    int top();
    size_t size();
    bool empty();

protected:
    size_t size_;
    int* mData;
};

CustomStack::CustomStack() : size_(0), mData(NULL) {}

CustomStack::~CustomStack() {
    while (!empty()) {
        pop();
    }
}

void CustomStack::push(int n) {
    int* newData = new int[size_ + 1];
    for (size_t i = 0; i < size_; ++i) {
        newData[i] = mData[i];
    }
    newData[size_] = n;
    delete[] mData;
    mData = newData;
    size_++;
}

void CustomStack::pop() {
    if (empty()) {
        cout << "Error: pop, but stack is empty" << endl;
        return;
    }
    size_--;
    int* newData = new int[size_];
    for (size_t i = 0; i < size_; ++i) {
        newData[i] = mData[i];
    }
    delete[] mData;
    mData = newData;
}

int CustomStack::top() {
    if (empty()) {
        cout << "Error: top, but stack is empty" << endl;
    }
```

```
    return mData[size_ - 1];
}

size_t CustomStack::size() {
    return size_;
}

bool CustomStack::empty() {
    return size_ == 0;
}


int main() {
    CustomStack Newstack;
    string input;
    int value;

    while (cin >> input) {
        if (input == "cmd_push") {
            cin >> value;
            Newstack.push(value);
            cout << "ok" << endl;
        } else if (input == "cmd_pop") {
            if (Newstack.empty()) {
                cout << "error" << endl;
                return 1;
            }
            int poppedValue = Newstack.top();
            Newstack.pop();
            cout << poppedValue << endl;
        } else if (input == "cmd_top") {
            if (Newstack.empty()) {
                cout << "error" << endl;
                return 1;
            }
            cout << Newstack.top() << endl;
        } else if (input == "cmd_size") {
            cout << Newstack.size() << endl;
        } else if (input == "cmd_exit") {
            cout << "bye" << endl;
            return 0;
        }
    }
}
```