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. 0384 _____ Дзаппала Д.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург
2021

Цель работы.

Научиться азам ЯП C++, ознакомиться и написать реализацию структуры данных стек с помощью класса.

Задание.

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.

Примечания:

1. Указатель на массив должен быть protected.
2. Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено
3. Предполагается, что пространство имен std уже доступно
4. Использование ключевого слова using также не требуется
5. Методы не должны выводить ничего в консоль

Выполнение.

«Закрытая» (private) область класса содержит 2 поля: count (переменная, хранящая в себе число элементов в стеке) и _size (кол-во выделенной памяти под целочисленный массив). В области protected хранится указатель на целочисленный массив. В конструкторе по умолчанию count инициализируется нулем, _size инициализируется трем. Деструктор освобождает память, выделенную под массив.

В методе push(int val) вначале идет проверка, а есть ли вообще место в массиве. Если нет, выделяется память, в которую копируется массив, после чего мы добавляем элемент в стек. Если места в массиве достаточно, просто добавляем эл-т в конец массива (стек). Также инкрементируем поле count.

Метод pop() просто декрементирует поле count.

Метод top() возвращает последний эл-т массива (верхушка стека).

Метод size() возвращает кол-во эл-ов в стеке.

Метод empty() возвращает пустой стек или нет.

Метод extend(int n) вначале прибавляет к полю _size значение n, после чего идет выделение памяти под новый массив размером _size, копируется в него все эл-ты предыдущего массива, освобождается память предыдущего массива и указатель на новый массив приравнивается указателю на старый массив.

В ф-ии main() объявляется переменная класса, переменная std::string («классовой строки») и целочисленная переменная, в которую будет записываться число, которое будет добавляться в стек. Далее запускается бесконечный цикл, вначале которого идет ввод строки, которая потом будет сравниваться с условиями из условия задачи. В зависимости от строки, выполняется тот или иной метод.

Выводы.

Были изучены классы, как реализовать свой собственный стек с помощью классов.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab.cpp

```
#include<iostream>

#include<cstdio>
#include<cstring>
#include<string>

class CustomStack {
public:
CustomStack()
{
count = 0;
_size = 3;
mData = new int[_size];
}
~CustomStack()
{
delete [] mData;
}

void push(int val)
{
if (count == _size)
{
// _size += 5;
extend(5);
}
mData[count] = val;
++count;
}

void pop()
{
--count;
}

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

size_t size() const { return count; }

bool empty() const { return (count > 0 ? false : true) ;}

void extend(int n)
{
_size += n;
int *cpy = new int[_size];
```

```
for (int i = 0; i < count; i++)
cpy[i] = mData[i];
delete [] mData;
mData = cpy;
}

private:
int count;
size_t _size;

protected:
int* mData;
};

int main() {

CustomStack st;
std::string a;
int d;
// bool flag = 1;


while (1)
{
// scanf("%s", a);
std::cin >> a;
if (a == "cmd_push")
{
// scanf("%d", &d);
std::cin >> d;
st.push(d);
std::cout << "ok" << std::endl;
}
else if (a == "cmd_pop")
{
if (st.empty()){
printf("error");
return 0;
}
std::cout << st.top() << std::endl;
st.pop();
}
else if (a == "cmd_top")
{
if (st.empty()){
printf("error");
return 0;
}
printf("%d\n", st.top());
}
```

```
else if (a == "cmd_size")
{
printf("%ld\n", st.size());
}
else if (a == "cmd_exit")
{
printf("bye\n");
return 0;
}

}

return 0;
}
```