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.

Студентка гр. 0382 _____ Кривенцова Л.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучить работу классов,методов и динамических структур (стека) на языке программирования Си.

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.

Примечания:

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

Основные теоретические положения.

New && Delete

В языке С память можно выделять с помощью библиотечной функции malloc (). Язык С++ предоставляет альтернативный способ — оператор new.

Он обеспечивает выделение динамической памяти в куче. Для освобождения выделенной памяти используется оператор delete.

Пример использования new и delete:

```
#include<iostream>
using namespace std;
int main(){
   int* number = new int;
   *number = 1;
   cout << *number; // 1
   delete number;
return 0;
}
```

Создать динамический массив на C++ легко; вы сообщаете оператору new тип элементов массива и требуемое количество элементов. Например, если необходим массив из 5 элементов int, следует записать так:

```
int * something = new int [5] ; // получение блока памяти из 5 элементов типа int
```

Оператор new возвращает адрес первого элемента в блоке. В данном примере это значение присваивается указателю something.

Теперь чтобы освободить память, выделенную раннее будет недостаточно написать:

```
delete something;
```

Потому что something указывает только на начальный элемент массива. Для таких случаев, когда требуется удалить массив данных, нужно написать так:

```
delete[] something;
```

Для успешной работы с new и delete надо помнить несколько простых правил:

Не использовать delete для освобождения той памяти, которая не была выделена new.

Не использовать delete для освобождения одного и того же блока памяти дважды.

Использовать delete [], если применялась операция new[] для размещения массива.

Использовать delete без скобок, если применялась операция new для размещения отдельного элемента.

Помнить о том, что применение delete к нулевому указателю является безопасным (при этом ничего не происходит).

Классы

Предисловие. Тема классов в C++, как, где их применять, тонкости работы с ними - тема очень обширная и не уместится на одном степе, поэтому рекомендуем воспользоваться ссылками на литературу в конце степа и разобраться более подробно.

Проблема

В языке C есть возможность определять структуры, т.е. новые типы данных, которые являются композицией из уже существующих типов. Однако структура в C определяется только данными, например:

```
struct Point {  // Структура в C это объединение различных типов данных в новый, единый тип данных
  int x;
  int y;
}
```

Язык C++ реализует объектно-ориентированную парадигму программирования, которая включает в себя реализацию механизма инкапсуляции данных . Инкапсуляция в C++ подразумевает, что:

В одной языковой конструкции размещаются как данные, так и функции для обработки этих данных

Доступ к данным извне этой конструкции ограничен, иными словами, напрямую редактировать данные как в структурах C нельзя. Пользователю предоставляется интерфейс из методов (API) с помощью которого он может влиять на состояние данных.

Структуры из C не подходят по обоим параметрам, язык C не поддерживает объектно-ориентированную парадигму.

P.S Причина ввода классов описанная выше - не единственная, но в рамках этого степа и курса её достаточно.

Решение

Для того, чтобы обеспечить такую инкапсуляцию данных, в C++ ввели классы. Класс - это шаблон, по которому определяется форма объекта. В нем указываются данные и код, который будет оперировать этими данными.

По-другому, класс - это абстрактный тип данных, который может включать в себя не только данные, но и программный код в виде функций. Они реализуют в себе оба принципа, описанных выше следующим образом:

В классе могут размещаться как данные (их называют полями), так и функции (их называют методы) для обработки этих данных.

Любой метод или поле класса имеет свой спецификатор доступа: public, private или protected (его мы не будем рассматривать).

Виртуальный метод — метод класса, который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения.

Особенности виртуальных методов:

Модификатор virtual располагается перед типом возвращаемого значения.

При определении базового класса следует объявить виртуальными те методы класса, которые могут быть переопределены в производных классах.

Выполнение работы.

Ход решения:

Реализован класс стека “CustomStack”.

Перечень методов класса стека:

- Метод *void push(int val)* - вызывает метод *extend*, выделяя память для нового элемента, который добавляетв стек.

- Метод *void pop()* - проверяет, если стек пуст – то выводит сообщение об ошибке, в противном случае выводит на экран последний элемент стека, а затем удаляет его.
- Метод *int top()* - проверяет, если стек пуст – то выводит сообщение об ошибке, в противном случае возвращает верхний элемент.
- Метод *size_t size()* - возвращает количество элементов в стеке.
- Метод *bool empty()* - проверяет отсутствие элементов в стеке – возвращает значение *true*, если стек пуст и значение *false*, если нет.
- Метод *void extend(int n)* - расширяет исходный массив на n ячеек (перевыделяет память массива с помощью *realloc*).
- Конструктор – инициализирует переменные (динамически выделяется память для массива, на основе которого моделируется стек).

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	Программа выводит верный ответ.
2.	cmd_top	error	Программа выводит верный ответ.

	cmd_push 5 cmd_push 7 cmd_top cmd_pop cmd_size cmd_pop cmd_size cmd_exit	ok ok 7 7 1 5 0 bye	Программа выводит верный ответ.

Выводы.

Были освоены навыки работы с классами, методами и динамическими структурами на языке программирования Си.

Разработана программа, моделирующая стек на основе динамического массива.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb4.c

```
class CustomStack {
public:
CustomStack(){
    size_stack = 0;
    mData = new int[0];
}

void pop(){
    if (size()<=0) {
        cout << "error";
        exit(0);
    }
    else {
        cout << *(mData + size_stack - 1) << endl;
        *(mData + size_stack - 1) = 0;
        size_stack--;
    }

}


void push (int val)
{  extend(1);
    size_stack++;
    *(mData+size_stack-1) = val;
    //cout << "ok" << endl;
}

int top(){
    if (size()<=0) {
        cout << "error";
        exit(0);
    }
    else
    return *(mData+size_stack-1);
}

size_t size(){
    return size_stack;
}

bool empty(){
    if(size_stack == 0) return true;
    else return false;
}
// методы size, empty + конструкторы, деструктор

private:
```

```
int size_stack = 0;
// поля класса, к которым не должно быть доступа извне
void extend(int n){
        count+=n;
        mData=(int*)realloc(mData,count*sizeof(int));
}
protected:  // в этом блоке должен быть указатель на массив данных

int* mData;
    int count=100;
};


int main (){
    CustomStack Stack;
    char comands[5][10] = {"cmd_push", "cmd_pop", "cmd_top",
                                  "cmd_size", "cmd_exit"};
    int count;
    char el_push[20];
    char input_str[10];
    while (1) {
        cin >> input_str;
        if (!strcmp(input_str, comands[0])) {
            cin >> el_push;
            count = 1;
        }
        for (int k = 1; k < 5; k++) {
            if (!strcmp(input_str, comands[k])) {
                count = k + 1;
            }
        }
        switch (count) {
            case 1:
                Stack.push(atoi(el_push));
                cout << "ok\n";
                break;
            case 2:
                Stack.pop();
                break;
            case 3:
                cout << Stack.top() << endl;
                break;
            case 4:
                cout << Stack.size() << endl;
                break;
            case 5:
                cout << "bye";
                exit(0);
            default:;
        }
        count = 0;
    }
    return 0;
}
```