МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Динамические структуры данных

Студент гр. 1304 _____ Кардаш Я.Е

Преподаватель _____ Чайка К.В

Санкт-Петербург

2022

Цель работы.

Изучение работы динамических структур данных в языке C++.

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.

Экспериментальные результаты

Входные данные	Вывод	Комментарий
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	Корректная работа программы

Выводы.

Были изучены основы языка C++, рассмотрена работа динамических структур данных. В качестве практического задания был написан стек на основе массива и продемонстрирована его работа.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#DEFINE N 20
#DEFINE M 100
CLASS CUSTOMSTACK
{
PUBLIC:
     CUSTOMSTACK()
     {
          BUFFER_SIZE = 0;
          ELEMENTS_COUNT = 0;
          MDATA = NULL;
     }

     ~CUSTOMSTACK()
     {
          DELETE[] MDATA;
     }

     VOID PUSH(INT VAL)
     {
          IF (ELEMENTS_COUNT == BUFFER_SIZE)
               EXTEND(100);
          MDATA[ELEMENTS_COUNT] = VAL;
          ELEMENTS_COUNT++;
     }

     VOID POP()
     {
          /*IF (EMPTY())
          {
               THROW 1;
               RETURN;
          }*/
          ELEMENTS_COUNT--;
```

```
    }

    INT TOP()
    {
        /*IF (EMPTY())
        {
            THROW 1;
            RETURN 0;
        }*/
        RETURN MDATA[ELEMENTS_COUNT-1];
    }

    SIZE_T SIZE()
    {
        RETURN ELEMENTS_COUNT;
    }

    BOOL EMPTY()
    {
        IF (!ELEMENTS_COUNT) RETURN TRUE;
        RETURN FALSE;
    }

PRIVATE:
    VOID EXTEND(INT N)
    {
        INT *TMP = NEW INT [BUFFER_SIZE + N];
        MEMCPY(TMP, MDATA, SIZEOF(INT)*ELEMENTS_COUNT);
        DELETE[] MDATA;
        MDATA = TMP;
        BUFFER_SIZE += N;
    }

    INT BUFFER_SIZE;
```

```
    INT ELEMENTS_COUNT;

PROTECTED:
    INT *MDATA;

};
VOID FREEALL(CHAR** ARR,INT N)
{
    FOR(INT I=0;I<N;I++)
        FREE(ARR[I]);
    FREE(ARR);
}
INT MAIN()
{
    CUSTOMSTACK();
    CUSTOMSTACK STACK;
    CHAR** CMD_ARR= (CHAR**)MALLOC(N*SIZEOF(CHAR*));
    CHAR* INP_S = (CHAR*)MALLOC(M*SIZEOF(CHAR));
    FGETS(INP_S,M,STDIN);
    INT N=0;
    WHILE(STRCMP(INP_S,"CMD_EXIT\N\0"))
    {
        CMD_ARR[N]=(CHAR*)MALLOC(M*SIZEOF(CHAR));
        STRCPY(CMD_ARR[N],INP_S);
        N++;
        FGETS(INP_S,M,STDIN);
    }
    CMD_ARR[N]=(CHAR*)MALLOC(M*SIZEOF(CHAR));
    STRCPY(CMD_ARR[N],INP_S);
    N++;
    FREE(INP_S);
    FOR(INT I=0;I<N;I++)
    {

        CHAR* P= STRTOK(CMD_ARR[I]," \N\0");
```

```
IF(!STRCMP(P,"CMD_PUSH"))
{
    CHAR* P=STRTOK(NULL," \N\0");
    INT NUMB = ATOI(P);
    STACK.PUSH(NUMB);
    COUT<<"OK\N";
    CONTINUE;
}
IF(!STRCMP(CMD_ARR[I],"CMD_POP"))
{
    IF(STACK.EMPTY())
    {
        COUT<<"ERROR\N";
        RETURN 0;
    }

    COUT<<STACK.TOP()<<"\N";
    STACK.POP();
    CONTINUE;
}
IF(!STRCMP(CMD_ARR[I],"CMD_TOP"))
{
     IF(STACK.EMPTY())
    {
        COUT<<"ERROR\N";
        RETURN 0;
    }
    COUT<<STACK.TOP()<<"\N";
    CONTINUE;
}
IF(!STRCMP(CMD_ARR[I],"CMD_SIZE"))
{
    COUT<<STACK.SIZE()<<"\N";
    CONTINUE;
}
```

```
        IF(!STRCMP(CMD_ARR[I],"CMD_EXIT"))
        {
            COUT<<"BYE\N";
            BREAK;
        }
    }
    FREEALL(CMD_ARR,N);
    RETURN 0;
}
```