**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. 3383 |  | Логинова А. Ю. |
| Преподаватель |  | Гаврилов А. В. |

Санкт-Петербург

2024

**Цель работы**

Целью данной лабораторной работы является разработка программы, моделирующая работу стека на базе массива.

**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.

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.

Пример:

| Test | Input | Result |
| --- | --- | --- |
| #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 |

# Выполнение работы

В данной лабораторной работе была реализована система управления стеком с использованием пользовательских массивов и структур данных. Рассмотрим подробнее каждую часть кода.rray

Макрос Array(type) используется для создания типизированных структур данных, представляющих собой динамические массивы. Он определяет структуру, которая содержит указатель на массив, размер и вместимость массива, а также указатель на функцию удаления элементов массива. Также макрос предоставляет функции для создания, удаления, очистки и изменения размера массива.

* create\_##type##array: Создает новый массив заданного типа с начальной вместимостью
* delete\_##type##array: Удаляет все элементы массива, вызывая указанную функцию удаления для каждого элемента, а затем освобождает память под массив.
* clear\_##type##array: Очищает массив, удаляя все его элементы.
* resize\_##type##array: Дважды увеличивает вместимость массива, если это необходимо. d\_next\_char

Функция read\_next\_char считывает следующий символ из файла, добавляет его в массив Array\_char, увеличивая размер массива при необходимости. Если достигнут конец файла или встречается символ новой строки, функция возвращает 0.d\_line

Функция read\_line считывает строку из файла, используя read\_next\_char, до тех пор, пока не встретит символ новой строки.

#### Класс CustomStack CustomStack

Класс CustomStack представляет собой реализацию стека с динамическим изменением размера. Класс реализован на основе массива объектов типа int. Он содержит методы для добавления (push), удаления (pop), получения размера (size), проверки на пустоту (empty) и получения верхнего элемента (top) стека. Также предусмотрена возможность расширения стека (extend).

* Конструктор и деструктор управляют выделением и освобождением памяти под стек.
* Метод push добавляет элемент в стек, увеличивая его размер при необходимости. Выводит сообщение «ok» в стандартный поток вывода std::cout, если функция отработала корректно.
* Метод pop удаляет верхний элемент стека. Выводит сообщение «error» в стандартный поток вывода, если функция отработала некорректно и завершает работу программы.
* Метод size возвращает текущий размер стека, выводя его значение.
* Метод empty проверяет, пуст ли стек, возвращает значение типа bool.
* Метод top возвращает верхний элемент стека без его удаления. Если элементов в стеке нет, выводит сообщение «error» в стандартный поток вывода и завершает работу программы.
* Метод extend увеличивает вместимость стека с помощью функции стандартной библиотеки realloc().

Класс CustomStack хранит в себе 3 поля: указатель на массив данных mData с модификатором доступа protected, по умолчанию он инициализируется как nullptr, размер стека stack\_size с модификатором доступа private, емкость стека stack\_capacity с модификатором доступа private.

В функции main() представлена работа с функциями, описанными выше. В работе были созданы 5 глобальных переменных cmd\_push, cmd\_pop, cmd\_top, cmd\_size, cmd\_exit. Функция сравнивает поступившую на вход строку с командой и вызывает ее.

**Тестирование**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| cmd\_push  ok  cmd\_push 1  ok  cmd\_push -1  ok  cmd\_push 000  ok  cmd\_top  0  cmd\_pop  0  cmd\_top  -1  cmd\_pop  -1  cmd\_top  1  cmd\_push 7  ok  cmd\_pop  7  cmd\_exit  bye | cmd\_push  ok  cmd\_push 1  ok  cmd\_push -1  ok  cmd\_push 000  ok  cmd\_top  0  cmd\_pop  0  cmd\_top  -1  cmd\_pop  -1  cmd\_top  1  cmd\_push 7  ok  cmd\_pop  7  cmd\_exit  bye |

**Выводы**

В ходе выполнения данной лабораторной работы было успешно реализовано управление динамическими массивами и стеком с использованием механизмов языка С++. Было продемонстрировано управление памятью через использование функций calloc, realloc и free. Это позволило эффективно создавать, изменять размер и удалять динамические массивы, что является важным аспектом разработки на языке C. Реализация стека с динамическим изменением размера показала, как можно эффективно использовать стековые структуры данных в программировании. Использование стандартных библиотек: в работе активно использовались стандартные библиотеки C++, такие как <cstdlib>, <iostream>, <cstring>, <string>, что позволяет упростить работу с памятью, вводом-выводом и строками, делая код более читаемым и поддерживаемым.

В целом, данная работа продемонстрировала умение работать с низкоуровневыми механизмами языка C++ для создания абстрактных структур данных и эффективного управления памятью.

## Исходный код программы

Название файла: main.py

#include <cstdlib>

#include <iostream>

#include <cstring>

#include <string>

#include "string"

#define SIZE 2

#define Array(type) \

typedef struct array\_##type { \

type \*arr; \

size\_t size, capacity; \

void (\*deleter)(type \*p); \

} Array\_##type; \

\

Array\_##type create\_##type##\_array(void (\*deleter)(type \*)) { \

Array\_##type arr = { static\_cast<type\*>(calloc(SIZE, sizeof(type))), 0, SIZE, deleter }; \

if(arr.arr) \

return arr; \

else { \

printf("[create\_" #type "\_array()]: memory allocation error!\n"); \

exit(1); \

} \

} \

\

void delete\_##type##\_array(Array\_##type \*arr) { \

if (arr->deleter) { \

for (size\_t i = 0; i < arr->size; ++i) \

arr->deleter(&arr->arr[i]); \

} \

free(arr->arr); \

} \

\

void clear\_##type##\_array(Array\_##type \*arr) { \

if (arr->deleter) { \

for (size\_t i = 0; i < arr->size; ++i) \

arr->deleter(&arr->arr[i]); \

} \

arr->size = 0; \

} \

\

void resize\_##type##\_array(Array\_##type \*arr){ \

arr->capacity \*= 2; \

type \*temp = static\_cast<type\*>(realloc(arr->arr, arr->capacity \* sizeof(type))); \

if(temp) \

arr->arr = temp; \

else { \

printf("[resize\_" #type "\_array()]: memory allocation error!\n"); \

delete\_##type##\_array(arr); \

exit(1); \

} \

} \

\

Array(char)

char read\_next\_char(Array\_char \*str, FILE \*f) {

char c = (char)getc(f);

if(c != EOF && c != '\n') {

if(str->size + 1 == str->capacity)

resize\_char\_array(str);

str->arr[str->size++] = c;

str->arr[str->size] = '\0';

return c;

}

else

return 0;

}

void read\_line(Array\_char \*s, FILE \*f) {

while(read\_next\_char(s, f));

}

class CustomStack {

public:

CustomStack() {

}

~CustomStack() {

free(mData);

}

void push(int obj) {

if(stack\_capacity == 0)

mData = static\_cast<int\*>(malloc(++stack\_capacity \* sizeof(int)));

else if (stack\_size == stack\_capacity){

resize();

}

mData[stack\_size++] = obj;

std::cout << "ok\n";

}

void pop() {

if(stack\_size == 0) {

std::cout << "error";

exit(0);

}

std::cout << mData[--stack\_size] << '\n';

}

size\_t size() {

std::cout << stack\_size << '\n';

return stack\_size;

}

bool empty() {

return stack\_size == 0;

}

int top() {

if(stack\_size == 0) {

std::cout << "error";

exit(0);

}

std::cout << mData[stack\_size - 1] << '\n';

return mData[stack\_size - 1];

}

void extend(int n) {

if(n > 0)

mData = static\_cast<int\*>(realloc(mData, (stack\_capacity + n) \* sizeof(int)));

}

private:

size\_t stack\_size = 0, stack\_capacity = 0;

void resize() {

extend(stack\_capacity);

}

protected: // в этом блоке должен быть указатель на массив данных

int\* mData = nullptr;

};strcmp(cmd.arr, cmd\_pop) == 0)

s.pop();

else if(strcmp(cmd.arr, cmd\_size) == 0)

s.size();

else if(strcmp(cmd.arr, cmd\_exit) == 0) {

std::cout << "bye";

exit(0);

}

}

}

const char cmd\_push[] = "cmd\_push";

const char cmd\_pop[] = "cmd\_pop";

const char cmd\_top[] = "cmd\_top";

const char cmd\_size[] = "cmd\_size";

const char cmd\_exit[] = "cmd\_exit";

int main() {

CustomStack s;

Array\_char cmd = create\_char\_array(NULL);

long n;

char \*endptr;

while(true) {

clear\_char\_array(&cmd);

read\_line(&cmd, stdin);

if(strncmp(cmd.arr, cmd\_push, sizeof(cmd\_push) - 1) == 0) {

n = std::strtol(&cmd.arr[sizeof(cmd\_push) - 1], &endptr, 10);

s.push(n);

}

else if(strcmp(cmd.arr, cmd\_top) == 0)

s.top();

else if(strcmp(cmd.arr, cmd\_pop) == 0)

s.pop();

else if(strcmp(cmd.arr, cmd\_size) == 0)

s.size();

else if(strcmp(cmd.arr, cmd\_exit) == 0) {

std::cout << "bye";

exit(0);

}

}

}