**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. 3341 |  | Лодыгин И.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с динамическими структурами данных C++.

## Задание

3 вариант.

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

Динамические структуры данных в C++ являются структурами, размер и форма которых могут динамически изменяться во время выполнения программы. Они обеспечивают гибкое управление памятью и эффективную работу с данными. Основные типы динамических структур данных включают динамические массивы, связанные списки, стеки, очереди и деревья. Для работы с динамическими структурами данных в C++ используются указатели и операторы new/delete для выделения и освобождения памяти. Динамические структуры данных позволяют управлять памятью более эффективно, оптимизировать использование ресурсов системы и реализовывать различные алгоритмы. Однако при использовании динамических структур данных важно следить за правильным управлением памятью, чтобы избежать утечек памяти и ошибок выполнения программы.

## Выполнение работы

Для начала в программе создается экземпляр класса CustomStack, который представляет собой стек, способный хранить целочисленные значения. После этого программа переходит к циклу, в котором ожидает ввода пользовательских команд.

Каждая команда обрабатывается в соответствии с ее типом. Например, если пользователь вводит команду "cmd\_push", программа считывает целочисленное значение, которое следует добавить в стек, используя метод push. После успешного добавления элемента в стек программа выводит сообщение "ok".

Если пользователь вводит команду "cmd\_top", программа проверяет, содержит ли стек элементы. Если стек не пуст, то программа выводит значение верхнего элемента, не удаляя его, с помощью метода top.

При вводе команды "cmd\_pop" программа также проверяет, не пуст ли стек. Если стек содержит элементы, программа выводит и удаляет верхний элемент с помощью метода pop.

Команда "cmd\_size" вызывает метод size, который возвращает размер стека. Программа выводит полученное значение.

Если пользователь вводит команду "cmd\_exit", программа выводит сообщение "bye" и завершает свою работу. При вводе некорректной команды программа выводит "error" и также завершает работу.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | cmd\_push 864  cmd\_push 998  cmd\_push 833  cmd\_pop  cmd\_push -7  cmd\_push 854  cmd\_push 654  cmd\_size  cmd\_pop  cmd\_pop  cmd\_exit | ok  ok  ok  833  ok  ok  ok  5  654  854  bye | Демонстрация работоспособности программы |

## Выводы

Была освоена работы с динамическими структурами данных в C++, также была написана программа, которая моделирует собой Stack.

# Приложение А Исходный код программы

Название файла: main.c

#include <iostream>

class CustomStack {

public:

CustomStack(){

mData = new int[0];

length = 0;

}

~CustomStack(){

delete[] mData;

}

void push(int val){

extend(1);

mData[length-1] = val;

}

void pop(){

extend(-1);

}

int top(){

return mData[length-1];

}

size\_t size(){

return length;

}

void extend(int n) {

int\* newData = new int[length + n];

for (int i = 0; i < length; ++i) {

newData[i] = mData[i];

}

delete[] mData;

mData = newData;

length += n;

}

bool empty(){

if(length == 0) return true;

return false;

}

protected:

int\* mData;

int length;

};

void processing() {

CustomStack stack;

std::string cmd;

int value;

while(true){

std::cin >> cmd;

if(cmd == "cmd\_push"){

std::cin >> value;

stack.push(value);

std::cout << "ok" << std::endl;

} else if(cmd == "cmd\_top"){

if(stack.empty()){

std::cout << "error" << std::endl;

break;

} else{

std::cout << stack.top() << std::endl;

}

} else if(cmd == "cmd\_pop"){

if(stack.empty()){

std::cout << "error" << std::endl;

break;

} else{

std::cout << stack.top() << std::endl;

stack.pop();

}

} else if(cmd == "cmd\_size"){

std::cout << stack.size() << std::endl;

} else if(cmd == "cmd\_exit"){

std::cout << "bye" << std::endl;

break;

} else{

std::cout << "error" << std::endl;

break;

}

}

}

int main() {

processing();

return 0;

}