**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. 3344 |  | Тукалкин.В.А |
| Преподаватель |  | Глазунов.С.А |

Санкт-Петербург

2024

## Цель работы

Изучить принцип работы с динамическими структурами данных на языке программирования С++.

## Задание.

Вариант 4.

Моделирование стека.

Требуется написать программу, моделирующую работу стека на базе списка. Для этого необходимо:

1) Реализовать класс CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных int.

Структура класса узла списка:

struct ListNode {

ListNode\* mNext;

int mData;

};

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 голову

ListNode\* mHead;

};

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

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:

* сmd\_push n - добавляет целое число n в стек. Программа должна вывести "ok"
* сmd\_pop - удаляет из стека последний элемент и выводит его значение на экран
* сmd\_top - программа должна вывести верхний элемент стека на экран не удаляя его из стека
* сmd\_size - программа должна вывести количество элементов в стеке
* сmd\_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.

## Выполнение работы

Выполнение работы будет расписано по шагам:

1. Написать класс CustomStack, в котором будут типы методов public, private и protected.
2. Написать конструктор и деструктор для класса.
3. В public написать методы:

* 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

1. В private создать переменную size\_stack типа int и присвоить 0, она будет отображать длину списка.
2. Написать функцию main, в которой создать stack для работы со списком и command для считывания команд.
3. Написать работу со списком с помощью if и else if.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 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 | Верный ответ |

## Выводы

Были изучены принципы работы с динамическими структурами данных на языке программирования С++.

# Приложение А Исходный код программы

Название файла: main.cpp

class CustomStack {

public:

//конструкторы

CustomStack() {

mHead = new ListNode;

mHead->mNext = nullptr;

}

//деструктор

~CustomStack() {

delete mHead;

}

// методы push, pop, size, empty, top

void push(int val){

ListNode\* mNext = new ListNode;

mNext->mData = val;

mNext->mNext = mHead;

mHead = mNext;

size\_stack++;

}

void pop(){

ListNode\* newHead = mHead->mNext;

delete mHead;

mHead = newHead;

--size\_stack;

}

int top(){

return mHead->mData;

}

size\_t size(){

return size\_stack;

}

bool empty(){

return size\_stack==0;

}

private:

int size\_stack=0;

protected: // в этом блоке должен быть указатель на голову

ListNode\* mHead;

};

int main() {

CustomStack stack = CustomStack();

string command;

while (cin >> command) {

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

int val;

cin >> val;

stack.push(val);

cout << "ok" << endl;

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

if (stack.empty()) {

cout << "error";

break;

} else {

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

stack.pop();

}

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

if (stack.empty()) {

cout << "error";

break;

} else {

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

}

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

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

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

cout << "bye";

break;

}

}

return 0;

}