**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 |  | Мурдасов М.К. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Изучение динамических структур данных. Моделирование динамических структур данных в языке 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.

**Примечания:**

1. Указатель на голову 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. Структуру **ListNode** реализовывать самому не надо, она уже реализована.

## Выполнение работы

Был реализован *class CustomStack*, содержащий в себе следующие поля и методы:

* public:
  + *CustomStack()* – конструктор, создающий голову стека.
  + *~CustomStack()* – деструктор, очищающий память, выделенную под стек.
  + ***bool*** *empty()* – возвращает *true*, если стек пуст и *false* в противном случае.
  + ***void*** *push(int val)* – добавляет новый элемент со значением *val* в стек и выводит *“ok”* .
  + ***void*** *pop()* – удаляет из стека последний элемент и выводит его значение на экран.
  + ***int*** *top()* – выводит на экран верхний элемент стека.
  + ***size\_t*** *size()* – возвращает количество элементов в стеке.
* private:
  + ***size\_t*** *count* – хранит количество элементов в стеке.
* protected:
  + ***ListNode\**** *mHead* – указатель на голову.

В функции *main()* объявляется объект класса *CustomStack* и переменная *command* типа *string* для записи в нее вводимых команд. Далее из стандартного потока ввода считываются команды, в зависимости от которых вызываются соответствующие методы для объявленного ранее стека. Также учитываются возможные ошибки и, при встрече таковых, на экран выводится *“error”*, а программа завершает работу с помощью *exit(0).* По завершении работы программы корректным образом, на экран выводится *“bye”*.

**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 | Корректно |

## Выводы

Были изучены динамические структуры данных. Был реализован стек на базе списка в языке С++ и написан API к нему. Также был реализован ввод команд из стандартного потока ввода.

# Приложение А Исходный код программы

Название файла: Murdasov\_Mikhail\_lb4.cpp

class CustomStack{

public:

CustomStack(){

mHead = new ListNode;

mHead->mNext = nullptr;

}

~CustomStack(){

while(mHead){

ListNode\* next = mHead->mNext;

delete mHead;

mHead = next;

}

}

bool empty() {

return count == 0;

}

void push(int val){

ListNode\* newNode = new ListNode{mHead, val};

mHead = newNode;

count++;

cout << "ok" << endl;

}

void pop(){

ListNode\* next = mHead->mNext;

cout << mHead->mData << endl;

delete mHead;

mHead = next;

count--;

}

int top(){

return mHead->mData;

}

size\_t size(){

return count;

}

private:

size\_t count = 0;

protected:

ListNode\* mHead;

};

int main(){

CustomStack stack;

string command;

while(cin >> command){

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

int val;

cin >> val;

stack.push(val);

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

if(stack.empty()){

cout << "error" << endl;

exit(0);

}

stack.pop();

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

if(stack.empty()){

cout << "error" << endl;

exit(0);

}

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

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

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

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

cout << "bye" << endl;

exit(0);

}

}

return 0;

}