**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»

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Гельман П.Е. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

# Цель работы

Целью работы является изучение основных механизмов языка С++ путем разработки структур данных стека на основе динамической памяти. Для достижения поставленной цели требуется решить следующие задачи:

– ознакомиться со структурами данных стека, особенностями их реализации;

– изучить и использовать базовые механизмы языка С++, необходимые для реализации стека;

– реализовать индивидуальный вариант стека в виде С++ класса, его операции в виде функций этого класса, ввод и вывод данных программы.

# Задание

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

Класс CustomStack моделирует работу стека с типом int на базе списка. Метод push добавляет новый элемент в стек и при добавлении увеличивает размерность стека на единицу. Метод empty обеспечивает проверку на пустоту и возвращает значение типа bool. Void pop() – удаляет «верхний» элемент из стека, если он непустой. Int top() – выводит последний добавленный элемент. Метод size возвращает количество элементов в стеке. CustomStack() и ~CustomStack() – конструктор и деструктор класса соответственно. Функция int pushing(string s) необходима для того, чтобы выделить из входящей строки число, которое нужно добавить в стек (вызывается при вызове метода cmd\_push). В функции main() реализован ввод нужных команд до тех пор, пока не поступит “cmd\_exit”, после этого на экран выводится “bye” и осуществляется выход из программы.

## Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1.

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Выходные данные |
| 1. | cmd\_push 1  cmd\_push 2  cmd\_top  cmd\_push 12  cmd\_size  cmd\_pop  cmd\_exit | Ok  Ok  2  Ok  3  12  bye |
| 2. | 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 |
| 3. | cmd\_pop | error |

# Выводы

Были изучены основные механизмы языка С++ путем разработки структур данных стека на основе списка с помощью ООП и реализована программа, решающая поставленную в лабораторной задачу.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

class CustomStack{

public:

CustomStack(){

mHead = nullptr;

mSize = 0;

}

void push(int val){

ListNode\* newElement = new ListNode;

newElement->mData = val;

newElement->mNext = mHead;

mHead = newElement;

mSize++;

}

bool empty(){

return mHead == nullptr;

}

void pop(){

if (empty()){

return;

}

ListNode \*deletingElement = mHead;

//cout << deletingElement->mData << endl;

mHead = mHead->mNext;

delete deletingElement;

mSize--;

}

int top(){

if (empty()){

return 0;}

return mHead->mData;

}

size\_t size(){

if (empty()){

return 0;

}

return mSize;

}

~CustomStack() {

while (mHead) {

ListNode\* temp = mHead;

mHead = mHead->mNext;

delete temp;

}

}

private:

size\_t mSize;

protected:

ListNode\* mHead;

};

int pushing(string s){

int n;

string cmd;

istringstream iss(s);

iss >> cmd >> n;

return n;

}

int main(){

string s;

CustomStack Stack;

while (s != "cmd\_exit"){

getline(cin, s);

if (s.find("cmd\_push") == 0){

int n = pushing(s);

Stack.push(n);

cout << "ok\n";

}

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

if (Stack.top() == 0){

cout << "error" << endl;

return 0;

}

else{

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

}

Stack.pop();

}

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

if (Stack.top() == 0){

cout << "error\n";

return 0;

}else{

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

}

}

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

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

}

}

cout << "bye\n";

return 0;

}