**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». Вариант

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Калиберов Н.И |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

# Цель работы

Изучить особенности реализации классов на языке C++ и освоить работу с ними. Реализовать на основе списка динамическую структуру данных стек, с использованием ООП.

# Задание

Требуется написать программу, которая последовательно выполняет подаваемые ей на вход арифметические операции над числами с помощью стека на базе **списка.**

**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()** - доступ к верхнему элементу
* **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

Описание класса *CustomStack*:

*public* методы:

* *CustomStack()* – конструктор класса, заполняющий поля нулевыми данными.
* *empty()* – проверка наличия элементов в стеке.
* *top()* – возвращает данные в верхнем элементе стека, если это возможно.
* *size()* – возвращает размер стека.
* *push(int value)* – добавляет новый элемент в стек.
* *pop()* – удаляет элемент из стека, если это возможно.
* *change(string value)* – удаляет два элемента из стека и в зависимости от полученного значения *value* добавляет сумму, разность, произведение или частное от деления удалённых элементов в стек.
* *~CustomStack()* – деконструктор класса, очищающий стек.

В области *private* находится размер стека *mSize*.

В области *protected* находится ссылка на голову стека *mHead*.

Описание основной части:

Сначала происходит считывание элементов и добавление в вектор. Для отслеживания символа перехода к новой строке, заканчивающего ввод, используется *cin.peek()*, который смотрит следующий символ из потока ввода, не удаляя его. Затем идёт обработка полученных элементов: числа добавляются в стек, а для операций вызывается метод *change*. По итогу должен остаться только один элемент в стеке, который выводится.

## Тестирование

Результаты тестирования содержатся в таблице 1.

Таблица 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарии |
| 1. | cmd\_push 1  cmd\_top  cmd\_push 2  cmd\_top  cmd\_pop  cmd\_size  cmd\_pop  cmd\_size  cmd\_exi | ok  1  ok  2  2  1  1  0  bye | Вывод соответствует ожиданиям. |
|  |  |  |
| 3. |  |  |

# Выводы

Во время выполнения лабораторной работы мы ознакомились с синтаксисом языка C++ по работе с классами, а также написали программу с использованием стека на основе списка.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

class CustomStack {

public:

CustomStack(){

mData = new int [100];

len = 0;

}

void push(int val){

mData[len++] = val;

}

void pop(){

len--;

}

int top(){

return mData[len - 1];

}

size\_t size(){

return len;

}

bool empty(){

return len == 0;

}

void extend(int n){

len += n;

}

protected:

int\* mData;

size\_t len;

};

int main() {

char str[200];

int num;

CustomStack stack = CustomStack();

scanf("%s %d", str, &num);

while (strcmp(str, "cmd\_exit") != 0){

if (strcmp(str, "cmd\_push") == 0){

stack.push(num);

cout<<"ok\n";

}

else if (strcmp(str, "cmd\_pop") == 0){

if (stack.size() > 0){

num = stack.top();

stack.pop();

cout<<num<<endl;

}

else{

cout<<"error\n";

return(0);

}

}

else if (strcmp(str, "cmd\_top") == 0){

if (stack.size() > 0) {

num = stack.top();

cout << num << endl;

}

else{

cout<<"error\n";

return(0);

}

}

else if (strcmp(str, "cmd\_size") == 0){

num = stack.size();

cout<<num<<endl;

}

scanf("%s %d", str, &num);

}

cout<<"bye\n";

return 0;

}