

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

Студентка гр. 3344

Якимова Ю.А.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Изучение основных механизмов языка C++ путем разработки структур данных стека на основе динамической памяти.

Задание.

Вариант 2. Стековая машина.

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 (не более 100 элементов) из чисел и арифметических операций (+, -, *, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

Если очередной элемент входной последовательности - число, то положить его в стек,

Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже),

Если входная последовательность закончилась, то вывести результат (число в стеке).

Если в процессе вычисления возникает ошибка:

например вызов метода `pop` или `top` при пустом стеке (для операции в стеке не хватает аргументов),

по завершении работы программы в стеке более одного элемента,

программа должна вывести "error" и завершиться.

Примечания:

Указатель на голову 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.

Структуру `ListNode` реализовывать самому не надо, она уже реализована.

Выполнение работы

С помощью класса *CustomStack* был реализован стек на основе односвязного списка. Класс *CustomStack* включает в себя:

1. Конструктор *CustomStack()*, который создает новый пустой стек с головой, указывающей на нулевой указатель.

2. Функцию *void push(int val)*, в которой создается узел для новой головы списка. Новая голова указывает на предыдущую и содержит в поле *mData* значение нового элемента.

3. Функцию *void pop()*, в которой происходит проверка на наличие элементов в списке. Если список пуст, программа выводит сообщение «error» и завершается. В противном случае указатель на голову списка сохраняется в переменную *delHead*, после чего голова заменяется на следующий элемент списка, а память, выделенная под старую голову, очищается.

4. Функцию *int top()*, в которой происходит проверка на наличие элементов в списке. Если список пуст, программа выводит сообщение «error» и завершается. В противном случае функция возвращает значение поля *mData* головы списка.

5. Функцию *size_t size()*, которая возвращает количество элементов в стеке.

6. Функцию *bool empty()*, которая проверяет отсутствие элементов в стеке.

7. Деконструктор *~CustomStack()*, который удаляет первый элемент списка до тех пор, пока он не опустеет.

8. Поле *size_t mSize* (private)

9. Указатель на голову

Далее в функции *main* был создан экземпляр класса *CustomStack p*, строка ввода была считана в переменную *input*. Строка ввода была разбита на токены с использованием *stringstream* и *getline*. Каждый токен проверяется на то, является ли он числом. Если является, то добавляется в стек с помощью метода *push*. В противном случае извлекаются 2 верхних элемента, над которыми выполняется соответствующая операция, после чего ее результат добавляется в стек.

Если после всех операций в стеке находится более 1 элемента, программа выводит сообщение «error» и завершается. В противном случае программа выводит значение головы.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 -10 - 2 *	error	-
2.	1 -10 - 2 *	22	-

Выводы

Были изучены основные механизмы языка C++ путем разработки структур данных стека на основе динамической памяти.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Yakimova_Yuliya_lb4.cpp

```

class CustomStack
{
public:
    CustomStack()
    {
        mHead = nullptr;
        mSize = 0;
    }

    void push(int val)
    {
        ListNode* newHead = new ListNode;

        newHead->mData = val;
        newHead->mNext = mHead;
        mHead = newHead;

        mSize++;
    }

    void pop()
    {
        if (empty())
        {
            cout << "error" << endl;
            exit(0);
        }

        ListNode* delHead = mHead; //???
        mHead = mHead->mNext;
        delete delHead;

        mSize--;
    }

    int top()
    {
        if (empty())
        {
            cout << "error" << endl;
            exit(0);
        }
        return mHead->mData;
    }

    size_t size()
    {
        return mSize;
    }

    bool empty()
    {
        return mHead == nullptr;
    }

    ~CustomStack()
    {
        while (!empty())
            pop();
    }
}

```

```

    }
}

private:
    size_t mSize;

protected:
    ListNode* mHead;
};

int main()
{
    CustomStack p = CustomStack();
    string input;
    getline(cin, input);

    stringstream ss(input);
    string token;

    while (getline(ss, token, ' ')) {
        if (isdigit(token[0]) || (token.size() > 1 && isdigit(token[1]))) {
            int num = stoi(token);
            p.push(num);
        }
        else {
            int num1 = p.top();
            p.pop();
            int num2 = p.top();
            p.pop();

            char oper = token[0];
            int res;
            switch (oper)
            {
                case '+':
                    res = num2 + num1;
                    break;
                case '-':
                    res = num2 - num1;
                    break;
                case '*':
                    res = num2 * num1;
                    break;
                case '/':
                    res = num2 / num1;
                    break;
            }

            p.push(res);
        }
    }

    if (p.size() > 1) {
        cout << "error" << endl;
        exit(0);
    }
    cout << p.top();

    return 0;
}

```