МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек. Очередь. Дек

Студент гр. 7383 _____ Корякин М.П.

Преподаватель _____ Размочева Н.В.

Санкт-Петербург

2018

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель работы: познакомиться со структурой и реализаей стека и использованием их в практических задачах на языке программирования C++.

Формулировка задачи: В заданном текстовом файле *F* записан текст, сбалансированный по круглым скобкам:

< текст > ::= < пусто > | < элемент > < текст >

< элемент > ::= < символ > | (< текст >)

где < символ > – любой символ, кроме (,). Для каждой пары соответствующих открывающей и закрывающей скобок вывести номера их позиций в тексте, упорядочив пары в порядке возрастания номеров позиций:

а) закрывающих скобок; б) открывающих скобок.

Вариант 9-аб-в.

2. РЕАЛИЗАЦИЯ ЗАДАЧИ

В функции main было реализовано меню для пользователя, где можно выбрать способ ввода входных данных. Данные можно ввести либо из файла, либо из терминала.

Функция except принимает на вход введенную строку. Пока строка не закончится, из нее считываются символы. Если мы встретили символ «,», то прекращаем программу и выводим «Недопустимый символ ‘,’».

Функция forstack принимает на вход введенную строку, далее считывает ее посимвольно и заносит в стэк открывающие скобки. Если встречается закрывающая скобка, то из стэка удаляется предыдущая скобка, и в массив структур заносятся номера расположения скобок в исходной строке.

Функция written осуществляет вывод «координат» скобок в нужной последовательности, для этого было сделано дополнительное меню для выбора: в какой именно последовательности нужно вывести «координаты» скобок. За сортировку данных местоположения скобок в строке отвечает стандартная функция qsort. В зависимости от нужной нам сортировки выполняются функции-компораторы comp1 и comp2.

Шаблонный класс Stack с шаблонным параметром T (тип хранимых элементов) представляет из себя стек на базе динамического массива. Класс содержит в себе следующие поля:

- T * arr.
- int size – базовый размер массива.
- int len – текущая длина

массива. Методы класса Stack:

- Конструктор, устанавливающий первоначальные значение полей.
- Push – принимает объект типа T, который добавляется в стек, при этом увеличивается индекс массива на единицу и проверяется, достигли ли мы конца массива, если да, то вызывается функция new_size, которая созздает массив нового размера и перезаписывает данные.
- Pop – возвращает нам последний элемент типа T.
- IsEmpty – проверяет стек на наличие в нем элементов.
- Clear – отчищает весь стэк.

3. ТЕСТИРОВАНИЕ

Программа была собрана в компиляторе g++ с ключом -std=c++14 в OS Linux Mint.

В ходе тестирования ошибки не были найдены. Тестовые случаи представлены в приложении А.

4. ВЫВОД

В ходе работы были получены навыки работы со стеком. Был реализован шаблонный класс, представляющий из себя стек на базе массива. Стек является удобной структурой данных для решения разных видов задач, в том числе и для данной.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 2 — Корректные тестовые случаи

Входные данные	Результат
Adsgsdfg()dfd() 1	8;9 13;14
Sdfsf()df,dfa	Недопустимый символ ‘,’
D(df(df)df(d)) 1	1;13 4;7 10;12
D(df(df)df(d)) 2	4;7 10;12 1;13
cxbdfdsgf	Скобок нет

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

Main.cpp:

```
#include "func.hpp"
    int main ()
    {
        string a;
        int run = 1;
        int m;
        while(run){
            cout<<"Введите 1, если хотите ввести выражение из консоли, введите 2, если хотите ввести выражение из файла, 3 - выход из программы."<<endl;
            cin>>m;
            cin.ignore();
            switch(m){
                case 1:{
                    cout << "Введите выражение:" << endl;
                    getline(cin, a);
                    a=except(a);
                    if (a=="null")
                        cout<<"Недопустимый символ ',' "<<endl;
                    else
                        forstack(a);
                break;
                }
                case 2:{
                        ifstream infile("input.txt");
                        if(!infile){
                            cout<<"File not open for reading"<<endl;
                            break;
                        }
```

```
                        getline(infile, a);
                        a=except(a);
                        forstack(a);
                        break;
                        }
                case 3:
                    cout<<"End!"<<endl;
                    return 0;
                default:
                    break;
            }
        }
        return 0;
}
```

Func.cpp:

```
#include "func.hpp"
   void forstack(string a){
            int i=0;
            int k=0;
            bracket* obj = new bracket[k];
            bool check=false;
            Stack<int> s;
            int n = a.length();
            for(i=0; i<n; i++){
                if(a[i]=='(')
                    s.Push(i);
                if(a[i]==')'){
                    check=true;
                    obj[k].open=s.Pop();
                    obj[k].close=i;
                    k++;
                    bracket* obj = new bracket[k];
                }
```

```
            }
            if(s.IsEmpty()){
                if(!check)
                    cout<<"Скобок нет"<<endl;
                else
                    written(k, obj);
            }
            else
                cout<<"Некоректное расположение скобок!"<<endl;
}

        int comp1(const void* a, const void* b){
            const bracket* k = (const bracket*)a;
            const bracket* m = (const bracket*)b;
            return ((k->open) - (m->open));
        }
        int comp2(const void* a, const void* b){
            const bracket* k = (const bracket*)a;
            const bracket* m = (const bracket*)b;
            return ((k->close) - (m->close));
        }

        void written(int k, struct bracket* obj){
            cout<<"Введите 1 для вывода в порядке возрастания для
открывающих скобок, 2 - для закрывающих"<<endl;
            int m;
            int i;
            cin>>m;
            switch (m) {
                case 1:
                        qsort(obj, k, sizeof(bracket), comp1);
                break;
                case 2:
                        qsort(obj, k, sizeof(bracket), comp2);
```

```
                break;
            }
            for(i=0; i<k; i++){
                cout<<obj[i].open<<';'<<obj[i].close<<endl;
            }
            delete [] obj;
        }

        string except(string str0){
            int i = 0;
            string nuli = "null";
            while(str0[i]!='\0'){
                if(str0[i]==',')
                    return nuli;
                i++;
            }
            return str0;
        }
```

Func.hpp:

```
#include <iostream>
#include <sstream>
#include <cstdlib>
#include <fstream>
#include <cstring>
#include <cmath>
#include <exception>
using namespace std;
using base = int;
struct bracket{
    int open;
    int close;
};

template <class T>
```

```
class Stack
{
private:
        T * arr;
        int size;
        int len;
public:
        Stack();
        ~Stack();
        int Size();
        void Push(T x);
        T Pop();
        T Top();
        void Clear();
        bool IsEmpty();
        void new_size(T s);
};
        template <class T>
        Stack<T>::Stack()
        {
                len = 0;
                arr = new T[size = 5];
        }
        template <class T>
        Stack<T>::~Stack()
        {
                delete[] arr;
        }
        template <class T>
        int Stack<T>::Size()
        {
                return len;
        }
        template <class T>
```

```
void Stack<T>:: Push(T x)
{
        if (len == size)
           new_size(size<<1);
        arr[len++] = x;
}
template <class T>
T Stack<T>::Pop()
{
    if (len == 0)
    return arr[len];
    return arr[--len];
}
template <class T>
T Stack<T>::Top()
{
    return arr[len];
}
template <class T>
void Stack<T>::Clear()
{
    delete arr;
}

template <class T>
bool Stack<T>::IsEmpty()
{
    return len == 0;
}
template <class T>
void Stack<T>::new_size(T s)
{
        if (s<=0)
    return;
```

```
                T * new_arr = new T[s];
                for (int i = 0; i<s; i++)
            new_arr[i] = arr[i];
                delete[] arr;
                size = s;
                arr = new_arr;
        cout<<"NEW SIZE "<<size<<endl;
        }
void written(int k, struct bracket* obj);
void forstack(string a);
int comp1(const void* a, const void* b);
int comp2(const void* a, const void* b);
string except(string str0);
```

Makefile:

```
all: run

run: main.o func.o func.hpp
 g++ main.o func.o -o run
main.o: main.cpp func.hpp
 g++ -c  main.cpp
func.o: func.cpp func.hpp
 g++ -c func.cpp
clean: rm -rf *.o
```