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: Линейные структуры данных: стек, очередь и дек

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

СОДЕРЖАНИЕ

Цель работы 3
Реализация задачи 3
Тестирование 4
Выводы 4
ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ 5
ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ 6

Цель работы

Ознакомится со стеком и очередью на практике, написать программу с реализацией стека на основе списка на языке программирования C++.

Формулировка варианта 5-д:

Правильная скобочная конструкция с тремя видами скобок определяется как

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

< элемент > ::= < символ > | (< текст >) | [< текст >] | { < текст > }

где < символ > – любой символ, кроме (,) , [,] , { , }. Проверить, является ли текст, содержащийся в заданном файле *F*, правильной скобочной конструкцией; если нет, то указать номер ошибочной позиции.

Реализация задачи

В данной работе были разработан класс Stack для работы со стеком:

```
class Stack {
private:
  char br;
  Stack *prev;
  Stack *head;
public:
  Stack() {
    br = '\0';
    head = NULL;
    prev = NULL;
  };
  void pop();
  void push(char a);
  char top();
  bool stempty();
  ~Stack();
};
```

В поле `char br` хранятся открывающие скобки, в поле `Stack *prev` – указатель на предыдущий элемент списка, в поле `Stack *head` – указатель на вершину стека. Эти поля закрыты для доступа из других частей программы. Далее идут функции для работы со стеком:

`Stack()` – конструктор класса, создает пустой стек;

`void pop()` – удаляет из стека последний элемент;

`void push(char a)` – добавляет скобку в стек: создает элемент списка и делает его головой;

`char top()` – возвращает из стека последний элемент, не удаляет его;

`bool stempty()` – проверяет пуст ли стек;

`~Stack()` – деструктор класса, удаляет все элементы списка;

Также была реализована функция `int Check(string str, Stack s)` проверяющая, является ли введенный текст правильной скобочной конструкцией. Функция посимвольно проходит по строке и добавляет в стек открывающие скобки: (, [, {. Если в строке встречается закрывающая скобка, то она сравнивается со скобкой в стеке. В том случае, когда скобки соответствуют, вызывается `pop()` и переходим к следующему символу. Иначе возвращается место ошибки (индекс символа в строке). Также место ошибки выводится, если в стеке остались открывающие скобки или, наоборот, в строке есть скобки, которым не предшествует открывающая.

Тестирование

Программа была собрана в компиляторе G++ в OS Linux Ubuntu 16.04 при помощи Makefile. В других системах тестирование не проводилось. Результаты тестирования приведены в приложении А.

Выводы

В ходе лабораторной работы была освоена работа с классами, а также были получены практические навыки работы со стеком для решения задач на языке C++.

Была написана программа для определения правильной скобочной конструкции.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

В табл. 1 приведены примеры работы программы.

Таблица 1 – Тестовые случаи

asdfghjykuli	Это текст!
(((((	Это не текст! Место ошибки: 4, (
()()(){{[]}}	Это текст!
Asdfgfb}hgvjhk	Это не текст! Место ошибки: 7, }
]]]]]]]]]]	Это не текст! Место ошибки: 10,]
	Это текст!
(ytfgyuhk)gjhb[jgkj{utgjvk}]	Это текст!
ohbjsk[agd(ghvjmb]hgc)	Это не текст! Место ошибки: 17,]

ПРИЛОЖЕНИЕ Б.

КОД ПРОГРАММЫ

main.cpp:

```
#include "stack.hpp"

int main() {

  string str;
  char k = 'a';
  Stack S;

  while (k != 'q'){

    cout<<"\nВыберите действие:"<<endl;
    cout<<"\t1 - Считать данные из файла."<<endl;
    cout<<"\t2 - Ввести данные вручную."<<endl;
    cout<<"\tq - Выход из программы."<<endl;

    cin>>k;
    switch (k) {
      case '1': {
        ifstream ifile("test.txt");
        getline(ifile, str);
        str.push_back('\0');
        len = str.size();
        ans = Check(str, S);
        if (ans == -1 && S.stempty()) cout<<"Это текст!" <<endl;
        else {
          cout<<"Это не текст!" <<endl;
          cout<<"Место ошибки: "<<ans<<endl;
        }
        break;
      }
      case '2': {
        getchar();
        getline(cin, str);
        str.push_back('\0');
        len = str.size();
        cout<<"len: "<<len<<endl;
        ans = Check(str, S);
```

```
        cout<<"ans: "<<ans<<endl;
        if (ans == -1 && S.stempty()) cout<<"Это текст!" <<endl;
        else {
          cout<<"Это не текст!" <<endl;
          cout<<"Место ошибки: "<<ans<<", "<<str[i]<<endl;
        }
        break;
      }
    }
  }
  return 0;
}
```

stack.cpp:

```
#include "stack.hpp"

void Stack::pop() { // удаляем верхний элемент
  if (head == NULL) {
    cerr<<"Stack is empty"<<endl;
    return;
  }
  Stack *temp = head;
  head = head->prev;
  delete temp;
}

void Stack::push(char b) { // добавляем новый элемент
  Stack *temp;
  temp = new Stack;
  temp->br = b;
  temp->prev = head;
  head = temp;
}

char Stack::top() { // возвращаем значение верхнего элемента
  if (head == NULL) {
    cerr<<"Stack is empty"<<endl;
    return '\0';
  }
  return head->br;
}

bool Stack::stempty() { // проверка на пустой стек
```

```
  if (head == NULL)
    return true;
  else
    return false;
}

Stack::~Stack() { // деструктор
  while (head != NULL)
    pop();
}

int Check(string str, Stack s) {
  int flag = 0;
  for (int i=0; str[i]!='\0'; i++){
    if (str[i] == '[' || str[i] == '{' || str[i] == '(') {
      s.push(str[i]);
      flag = i;
    }
    if (str[i] == ']' || str[i] == '}' || str[i] == ')') {
      if (s.top() == '[' && str[i] == ']')
        s.pop();
      else if (s.top() == '{' && str[i] == '}')
        s.pop();
      else if(s.top() == '(' && str[i] == ')')
        s.pop();
      else return i;
    }
  }
  if (s.stempty()) return -1;
  else return flag;
}
```

stack.hpp:

```
#include <iostream>
#include <cstring>
#include <fstream>
using namespace std;

class Stack { // определение класса
private:
  char br;
  Stack *prev;
  Stack *head;
```

```
public:
  Stack() {
    br = '\0';
    head = NULL;
    prev = NULL;
  };
  void pop();
  void push(char a);
  char top();
  bool stempty();
  void clear();
  ~Stack();
};

int Check(string str, Stack s);
```