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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Стек, очередь, дек

Студент гр. 8304 _____ Порывай П.А.

Преподаватель _____ Фиалковский М.С.

Санкт-Петербург

2019

Цель работы.

Получить опыт работы с классом «стек», выполнить задачу посредствам данной структуры данных.

Постановка задачи.

Определить, имеет ли заданная в файле F символьная строка следующую структуру: a D b D c D d..., где каждая строка a,b,c,d,..., в свою очередь, имеет вид x1 C x2. Здесь x1 есть строка, состоящая из символов A и B, а x2 строка, обратная строке x1(т.е. если x1= ABABB, то x2= BBABA). Таким образом, исходная строка состоит только из символов A,B,C и D. Исходная строка может читаться только последовательно (посимвольно) слева направо.

Описание алгоритма

Сначала программа должна считать получить команду на ввод из файла или из строки. После счета строки, если это не пустая строка(« », \n), вызывается функция analizator, которая и выполняет проверку последовательности символов a D b D c D d…, в ней используется стек в который помещаются начальные элементы последовательности из символов A, B пока не встретиться символ C, далее элементы последовательно «достаются» из стека и сравниваются с элементами строки по которой движемся

Описание функций и структур данных

int analizator(std::string& str)

Функция принимает входную строку и вызывает конструктор класса стек STACK<char> save(str.length()) , который создает его с некоторыми методами , str.length() - длина стека(а точнее динамического массива который выделяется для данной АСД). Во внешнем цикле while идет прохождение строки и если встретились символы A или B они

помещаются в стек, как только встретился символ C читаем «отраженную» строку и сравниваем с элементом из стека, в случае несовпадения исходной последовательности «отраженной» flag = 0 и происходит выход из while. Также есть проверка на несовпадение буквам C И D после которых происходит выход из циклов

template <class Item> class STACK

Шаблонный класс стек позволяет использовать стек для загрузки в него элементов разных типов. Элементы к которым мы не можем обратиться вне описания класса Item* s; int N; int N1; В public функции к которым мы можем обратиться «извне»

bool empty()

Возвращает true если стек пуст

void push(Item elem)

Добавлет элемент в вершину стека,если хватает места

Item pop()

Достает элемент из стека

~STACK()

Деструктор - свобождает память выделенную под стек через оператор delete после всех операций с объектами

void inp(...) не допускает дубликации кода при вводе стоки из файла или из консоли

int main(int argc,char *argv[])

Получает в качестве строки для открытия файла argv[1], вызывает inp, далее вызывается analizator если строка не пуста.

Выводы.

Получены навыки работы с «классовой» реализацией стека.

Приложение

Исходный код

```
#INCLUDE <IOSTREAM>
#INCLUDE <FSTREAM>
#INCLUDE <CSTDLIB>
#INCLUDE <STRING>
#INCLUDE<ASSERT.H>
// ИНТЕРФЕЙС АТД "СТЕК" (СМ. СЕДЖВИК - SEDGEWICK)
// ШАБЛОННЫЙ КЛАСС. БЕЗ НЕКОТОРЫХ ПРОВЕРОК...
// И ИНТЕРФЕЙС, И РЕАЛИЗАЦИЯ В ЭТОМ ЗАГОЛОВОЧНОМ ФАЙЛЕ
VOID INP(STD::STRING STR);



TEMPLATE <CLASS ITEM>
CLASS STACK
{
PRIVATE:
      ITEM* S; INT N; INT N1; // MY N1
PUBLIC:
      STACK(INT MAXN)
      {
            S = NEW ITEM[MAXN];
            N = 0;
            N1 = MAXN;
      }  // MY N1
      BOOL EMPTY() CONST
      {
            RETURN N == 0;
      }
      VOID PUSH(ITEM ELEM)
      {
            S[N++] = ELEM;
            IF(N>N1)
                  ASSERT(1);


      }
```

```
	~STACK(){
		DELETE [] S;
		N1 = 0;
		N = 0;
	}
	ITEM POP()
	{
		IF(N!=0)
			RETURN S[--N];
		ELSE
			ASSERT(1);
	}

};



INT ANALIZATOR(STD::STRING& STR){

	STACK<CHAR> SAVE(STR.LENGTH());

	INT FLAG = 1;
	INT I = 0;

	WHILE(I < STR.LENGTH()) {

		IF (STR[I] == 'C') {
			I++;
			WHILE (SAVE.EMPTY() == FALSE) {

				CHAR A = SAVE.POP();
				IF (STR[I] != A) {
					FLAG = 0;

					BREAK;//ВЫХОД ИЗ ВНУТР WHILE
				}

				I++;
```

```
                }
                //BREAK ДАЛЕЕ ВЫХОД ИЗ ВНЕШНЕГО WHILE
                IF(I == STR.LENGTH())
                      BREAK;

                IF(STR[I] !='D')
                      FLAG = 0;

                IF (FLAG == 0)
                      BREAK;

          }
          ELSE IF(STR[I] == 'A' || STR[I] == 'B' ) {
                SAVE.PUSH(STR[I]);
          }
          ELSE {
                FLAG = 0;
                BREAK;
          }

          I++;
     }

     RETURN FLAG;



}

INT MAIN(INT ARGC,CHAR *ARGV[])
{
     SETLOCALE(LC_ALL, "RUSSIAN");
     STD::STRING STR;
//   INT FLAG;
     STD::COUT<<"ВВОД ИЗ КОНСОЛИ ИЛИ ИЗ ФАЙЛА?(C,F)"<<"\N";
     STD::STRING ARG;
     GETLINE(STD::CIN,ARG);
//   STD::OFSTREAM FOUT("OUT_WRONG.TXT");
     IF(ARG == "F"){
```

```
                STD::IFSTREAM FIN(ARGV[1]);
                STD::COUT<<"ВВЕДИТЕ   СТРОКУ   ИЛИ   КОМБИНАЦИЮ   ДЛЯ
ВЫХОДА\N";

                WHILE (GETLINE(FIN, STR)) {
                          STD::COUT<<"ВВЕДИТЕ  СТРОКУ  ИЛИ  КОМБИНАЦИЮ
ДЛЯ ВЫХОДА\N";
                          INP(STR);

                }
           }
           ELSE IF( ARG == "C"){
                STD::COUT<<"ВВЕДИТЕ   СТРОКУ   ИЛИ   КОМБИНАЦИЮ   ДЛЯ
ВЫХОДА\N";
                WHILE(GETLINE(STD::CIN, STR)){
                          STD::COUT<<"ВВЕДИТЕ  СТРОКУ  ИЛИ  КОМБИНАЦИЮ
ДЛЯ ВЫХОДА\N";
                               INP(STR);
                }



             }



     }

     VOID INP( STD::STRING STR ){

           STD::OFSTREAM FOUT("OUT_WRONG.TXT");
           INT FLAG;
            IF (STR != " " && STR != "") {

                                    FLAG = ANALIZATOR(STR);

                                    IF (FLAG == 1){
                                           STD::COUT<<STR<<"\N";
```

```
                                                    STD::COUT << "СТРОКА
ЗАДАНА ВЕРНО" << "\N";
                                                STD::COUT<<"\N";
                                                FOUT<<STR<<"\N";
                                                    FOUT << "СТРОКА ЗАДАНА
ВЕРНО" << "\N";
                                                FOUT<<"\N";
                                            }
                                            ELSE{
                                                STD::COUT<<STR<<"\N";
                                                  STD::COUT << "СТРОКА
ЗАДАНА НЕВЕРНО" << "\N";
                                                STD::COUT<<"\N";
                                                FOUT<<STR<<"\N";
                                                    FOUT << "СТРОКА ЗАДАНА
НЕВЕРНО" << "\N";
                                                FOUT<<"\N";


                                            }



            }
               ELSE{
               STD::COUT << "ПУСТАЯ СТРОКА" << "\N";
                    STD::COUT<<"\N";
                    FOUT<<"ПУСТАЯ СТРОКА"<<"\N";
                    FOUT<<"\N";
            }


     }
```

Тесты

Ввод

ABACABADABBCBBA

ABCBADABBCBBA

BBACABBDAAABCBAAA

BBBBACABBBBDABCBA

ABBACABBADBAABCBAAB

ABCBADABCBADABCBA

AABCBAADABCBA

AABACABAADABCBADABCBA

ABCBADABCBA

BACABDABCBA

Вывод

ABACABADABBCBBA

```
Строка задана верно
```

ABCBADABBCBBA

```
Строка задана верно
```

BBACABBDAAABCBAAA

```
Строка задана верно
```

BBBBACABBBBDABCBA

```
Строка задана верно
```

ABBACABBADBAABCBAAB

```
Строка задана верно
```

ABCBADABCBADABCBA

```
Строка задана верно
```

AABCBAADABCBA

```
Строка задана верно
```

AABACABAADABCBADABCBA

```
Строка задана верно
```

```
ABCBADABCBA
Строка задана верно

BACABDABCBA
Строка задана верно
```

Ввод

```
ABwCABADABBCBBA
aBCBADABBCBBA
wBACABBDAAABCBAAA
BwBBACABBBBDABCBA
ABBACABwADBAABCBAAB
ABCBADAeCBADABCBA
AABCBAeDABCBA
AABACABAADrBCBADABCBA
ABCBADABCBa
BACABDABCB11
ABCBAqABACABA
```

Вывод

```
ABwCABADABBCBBA
Строка задана неверно

aBCBADABBCBBA
Строка задана неверно

wBACABBDAAABCBAAA
Строка задана неверно

BwBBACABBBBDABCBA
Строка задана неверно

ABBACABwADBAABCBAAB
Строка задана неверно

ABCBADAeCBADABCBA
Строка задана неверно

AABCBAeDABCBA
Строка задана неверно

AABACABAADrBCBADABCBA
```

```
Строка задана неверно

ABCBADABCBa
Строка задана неверно

BACABDABCB11
Строка задана неверно

ABCBAqABACABA
Строка задана неверно

Пустая строка

Пустая строка
```