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

Кафедра «МОЭВМ»

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студент гр. 7381 _____ Дорох С. В.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург

2018

Цель работы.

Ознакомиться с основными понятиями и приемами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций.

Основные теоретические положения.

Рекурсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя.

Ход работы:

Задание:

Требуется построить синтаксический анализатор для понятия *простое выражение.*

простое_выражение::=простой_идентификатор |
(*простое_выражение знак_операции простое_выражение*)
простой_идентификатор::= буква
знак_операции:: = – | + | *

В работе используется язык программирования C.

Выполнение:

Был создан файл с исходным кодом: lab1.c.

В нем созданы:

Функция is_Valid(char* buffer). Данная функция анализирует исходное выражение при помощи рекурсии и выводит ответ, который зависит от вида выражения.

Файл компиляции программы — compile.sh:

компилирует исходный код в исполняемый файл lab1.

Файл содержащий результат выполнения программы: runtests.sh:

содержит скрипт, запускающий ряд тестов.

Тестирование программы:

Тестирование данной программы состоит в том, чтобы ее проверить работоспособность путем запуска со всеми верными входными данными, а также убедится в корректном выводе при неверных введенных данных. В качестве корректных входных данных были выбраны следующие примеры записи вещественных чисел:

(a*(c+d));

((a+b)*(m-k));

a

В качестве некорректных входных данных были выбраны следующие строки:

((((((a+b)));

((m-k)/d);

a+.

Вывод.

В ходе выполнения лабораторной работы получены знания по теме «рекурсия» и закреплены знания синтаксиса языка С;

ИСХОДНЫЙ КОД:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

int is_Valid(char *buffer) {
   if (strlen(buffer) == 1)
       return isalpha(buffer[0]);
   else if (!strlen(buffer))
      return 0;

   char *first_brackets;                        //Указатели, которые будут строками после
разделения исходного выражения
   char *second_brackets;                        //

   int bracket_count = 0;
   int i = 1;
   do {                                    //Начало блока
      if (buffer[i] == '(')                    //по нахождению закрывающейся скобки
         bracket_count++;                        //для соответствующей открывающейся
      if (buffer[i] == ')')
         bracket_count--;
      i++;
      if(buffer[i] == '\0' && bracket_count != 0){
         printf("Different number of brackets!\n");
         return 0;
      }                                       //
   } while (bracket_count != 0 && i <= strlen(buffer));    //

   if(buffer[i] != '+' && buffer[i] != '-' && buffer[i] != '*' && !isalpha(buffer[i])){
//Проверка знака в выражении
      printf("%d element is an invalid character!\n", i);                    //на его
валидность
      return 0;
   }

   buffer[strlen(buffer)-1] = '\0';                //Разделение строк
   first_brackets = buffer + 1;                    //на строку до знака выражения
   second_brackets = buffer + i + 1;                    //и строку после знака
   buffer[i] = '\0';

   if (is_Valid(first_brackets) && is_Valid(second_brackets)) {    //Одновременный
вызов исходной функции для получившихся строк
```

```
        return 1;
    }
    return 0;

}

int main(){
    char *buffer=malloc(sizeof(char)*20);
    scanf("%s", buffer);
    printf(is_Valid(buffer) ? "This expression is correct!\n" : "This expression is
incorrect!!\n");
    free(buffer);
}
```