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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студентка гр. 9383 Ноздрин В.Я.

Преподаватель Попова Е.В.

Санкт-Петербург
2020

Цель работы.

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

Задание.

Написать программу, которая по заданному константному выражению, где, **константное_выражение::= ряд_цифр |**
константное_выражение знак_операции константное_выражение,
знак_операции::=+ | − | *,
ряд_цифр::=цифра | цифра ряд_цифр
вычисляет его значение либо сообщает о переполнении (превышении заданного значения) в процессе вычислений.

Теория.

Рекурсия - способ организации вычислительного процесса, при котором процедура или функция в ходе выполнения составляющих ее операторов обращается сама к себе.

Ход работы:

1. Проанализировать данные задачи, выделив рекурсивные действия с данными.
2. Понять, какие действия лучше выполнять рекурсивно, а какие — циклично.
3. Разработать программу с использованием рекурсии.
4. Протестировать программу

Выполнение работы:

В данной задаче нам дается правильное **константное_выражение** и необходимо его соответствующим образом считать и вычислить, отслеживая переполнение. Для отслеживания переполнения были написаны функции **addOvf** и **mulOvf**, которые вычисляют соответственно сумму и произведение и сообщают о переполнении, если оно происходит. Для распознавания **ряда_цифр** написана функция **numbers**, которая использует цикл, чтобы преобразовать строку символов цифр в число. А основной ход программы выполняет рекурсивная функция **eval**, которая согласно рекурсивному определению **константного_выражения**, соответствующим образом производит необходимые вычисления. Важно отметить, что функция **eval** формально разделяет строку на подстроки соблюдая приоритет операций, что в результате дает на выходе арифметически правильно вычисленное **константное_выражение.**

Пример работы программы:

Входные данные (в файле input.txt):

1111111+1111*5

12*9+23*8-45*7+67*6-89*5

5*1111+1111111

Выходные данные:

1111111+1111*5 = 1116666

12*9+23*8-45*7+67*6-89*5 = 33

5*1111+1111111 = 1116666

Разработанный программный код см. в приложении А.

Вывод.

Было произведено знакомство с основными понятиями и приемами рекурсивного программирования, были получены навыки программирования рекурсивных процедур и функций на языке программирования С.

Приложение А
Исходный код программы

Название файла: main.c

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

long long int addOvf(int a, int b) {
    long long int result = a + b;
    if ((a > 0 && b > 0 && result < 0)
    ||(a < 0 && b < 0 && result > 0))
        perror ("Int overflow");
    return result;
}

long long int mulOvf(int a, int b) {
    if (a == 0 || b == 0)
        return 0;

    long long int result = a * b;
    if (a == result / b)
        return result;
    else
        perror ("Int overflow");
}

long long int number(char* str) {
    int len = strlen(str);
    long long int result = 0;
    while (len) {
        result = mulOvf(result, 10);
        result = addOvf(result, (int) str[--len] - '0');
    }
    return result;
}

long long int eval(char* str) {
    char* operator = NULL;

    operator = strchr(str, '+');
    if (operator) {
        *(operator++) = '\0';
        return addOvf(eval(str), eval(operator));
    }

    operator = strchr(str, '-');
    if (operator) {
        *(operator++) = '\0';
        return addOvf(eval(str), -eval(operator));
    }

    operator = strchr(str, '*');
    if (operator) {
        *(operator++) = '\0';
        return mulOvf(eval(str), eval(operator));
    }
```

```
    return number(str);
}

int main () {
    FILE *pFile;
    char* str = malloc(sizeof(char)*201);

    pFile = fopen("input.txt", "r");
    if (pFile == NULL) {
        perror ("Error opening file");
    } else {
        while (fgets(str, 200, pFile)) {
            char* tmp = strchr(str, '\n');
            if (tmp) {*tmp='\0';};
            printf("%s", str);
            printf(" = %lld\n", eval(str));
        };
    };
    fclose(pFile);
    return 0;
}
```