

**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 системы данных»**  
**ТЕМА: РЕКУРСИЯ**

Студент гр. 9381 \_\_\_\_\_

Нагин Р.В.

Преподаватель \_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2020

## Цель работы.

Ознакомится с понятием рекурсии и способами её использования.  
Написать программу на языке Си, решающую поставленную задачу с помощью рекурсии.

## Задание.

11. Написать программу, которая по заданному (см. предыдущее задание) *константному\_выражению* вычисляет его значение либо сообщает о переполнении (превышении заданного значения) в процессе вычислений.

Константное выражение из 10 задания:

$$\begin{aligned} \text{константное\_выражение} &::= \text{ряд\_цифр} | \\ &\quad \text{константное\_выражение} \text{ знак\_операции } \text{константное\_выражение} \\ \text{знак\_операции} &::= + \mid - \mid * \\ \text{ряд\_цифр} &::= \text{цифра} \quad | \quad \text{цифра} \quad \text{ряд\_цифр} \end{aligned}$$

## Основные теоретические положения.

Рекурсия - определение части функции через саму себя, то есть это функция, которая вызывает саму себя непосредственно (в своём теле) или косвенно (через другую функцию). Всё решение сводится к решению основного или, как ещё его называют, базового случая. Существует такое понятие как шаг рекурсии или рекурсивный вызов. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов с целью сведения задачи к более простой. И так до тех пор, пока не получится базовое решение.

## Описание алгоритма.

Цель алгоритма - посчитать полученное выражение соблюдая при этом порядок действий (сначала умножение, потом сложение и вычитание). При этом программа должна следить за тем, чтобы в процессе вычисления не происходило переполнение заданной величины.

Алгоритм забирает из строки первое число и проверяет не ответ ли это. Если нет, то проверяет знак, число и знак за числом:

- Если знак “\*”, то последовательно перемножает все множители, пока знак за числом не станет “+” или “-”. В процессе перемножения удаляет знаки перед числами и сами числа, так чтобы впоследствии строка выглядела как плюс или минус и остальное выражение.
- Если “+” или “-”, то идет проверка знака после числа. Если это “+” или “-”, то просто суммирует два слагаемых или считает разность. Если “\*”, то сначала считается произведение за первым знаком и только потом сложение или вычитание. В процессе счета также удаляются использованный знак и число, чтобы обработчик на следующей итерации получил выражение вида: плюс или минус и остальное выражение.

Алгоритм заканчивает свою работу если:

- Встреченный знак — это знак конца строки (“\n”). В файле-ответе записывается ответ арифметического выражения, и программа завершается.
- В процессе счета случилось переполнение. Программа выведет переполнение и значение, на котором возникло переполнение, и завершится.

## **Выполнение работы.**

Описание структуры data:

- `int sum` - сумма всего выражения.
- `int mul` - переменная хранит промежуточные значения произведений.
- `int max` - порог переполнения.
- `int tabs` - количество табуляций.
- `char* s` - строка-выражение.
- `FILE* f` - файл-ответ.

Вспомогательные функции:

- `char* readsent() :`

Функция предназначена для считывания данных из файла “data.txt” в строку `sent`. Функция открывает нужный файл и посимвольно считывает данные из него. Память выделяется динамически, так что не стоит беспокоиться о длине выражения. Функция возвращает указатель на строку `sent`.

- `void shift(char* sent, int ind) :`

Функция предназначена для смещения символов строки `sent` влево на `ind` элементов. Функция удаляет использованные числа и арифметические знаки, чтобы рекурсия могла продолжить свою работу.

- `int numlen(int num) :`

Функция принимает число `num` и возвращает количество цифр в нем. Эта функция используется в основном как аргумент функции `shift()`, чтобы определить сколько символов нужно удалить.

- `void printtabs(FILE* f, int n) :`

Функция принимает указатель на файл `f` и вписывает в него количество табуляций `n`, а также символ “|\_\_\_\_” для визуализации углубления рекурсии.

Рекурсивные функции:

- `void mult(data *dt) :`

Рекурсивная функция умножения. Принимает указатель на данные `dt`. Функция заранее получает первый множитель, записанный в `dt.mul`. Затем она считывает из строки множитель и знак после множителя (для определения следующего действия) с помощью `sscanf()` и вычисляет значение произведения в тот же `dt.mul`. Потом идет проверка на переполнение (если переполнение, то выходим из функции), а также проверка следующего действия (если умножение, то рекурсия продолжается). Визуализация умножения реализована как цепочка промежуточных и основного ответа произведения. Н-р: для выражения `2*2*2*2` визуализация будет выглядеть как “2 --> 4 --> 8 --> 16”.

- `void calc(data *dt) :`

Основная рекурсивная функция. Принимает указатель на данные `dt`. Функция заранее получает первое число, записанное в `dt.sum`. Затем она считывает из строки арифметический знак до числа, число и знак после множителя (для определения следующего действия) с помощью `sscanf()`. Идет проверка числа на переполнение и проверка на первый знак (если это знак переноса строки, то завершить вычисления). Затем идет разделение на арифметические действия:

- Сложение: где также идет разделение на сложение с числом или произведением. Если с произведением, то сначала считается произведение с помощью рекурсивной функции `mult()` со своими проверками на переполнение и только потом считается сумма, после которой идет ещё одна проверка на переполнение. В процессе счета обработанные знаки и числа удаляются функцией `shift()`. Если с числом, то все тоже самое, но без подсчета произведения. Затем переход к следующему действию вызовом `calc()`.
- Вычитание: где также идет разделение на вычитание числа или произведения. Если вычитаемое - произведение, то сначала считается произведение с помощью рекурсивной функции `mult()` со своими проверками на переполнение и только потом считается разность. В процессе счета обработанные знаки и числа удаляются функцией `shift()`. Если вычитаемое - число, то все тоже самое, но без подсчета произведения. Затем переход к следующему действию вызовом `calc()`.
- Умножение: идет умножение первых двух множителей, проверка на переполнение. Если знак после второго множителя “\*”, то дальнейшее вычисление переходит рекурсивной функции `mult()`. После проверки на переполнение результат умножения записывается в всего выражения. Затем переход к следующему действию вызовом `calc()`.

Функция `main()` :

Функция инициализирует структуру `data dt`. Затем заполняет её поля: все целочисленные переменные - нули, строка заполняется функцией `readsent()` и открывается файл на запись “`answer.txt`”. Порог переполнения

— это первое число в файле “data.txt”, так что оно записывается в dt.max, а затем удаляется из строки dt.s функцией shift(), чтобы в строке осталось только выражение.

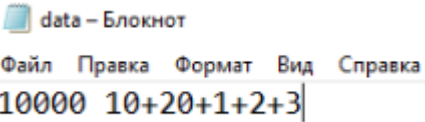
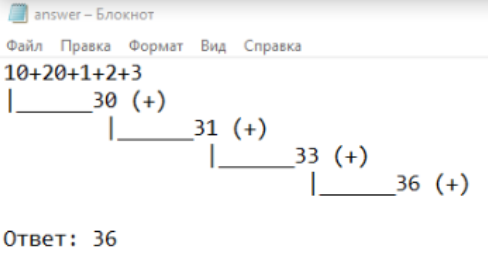
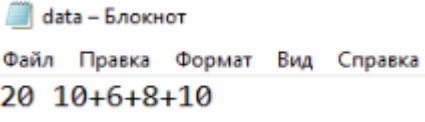
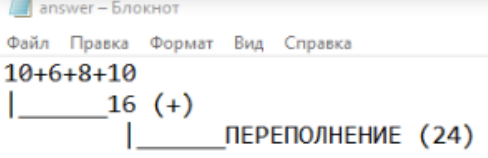
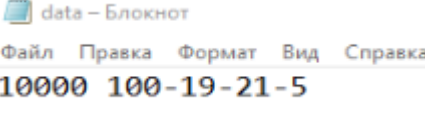
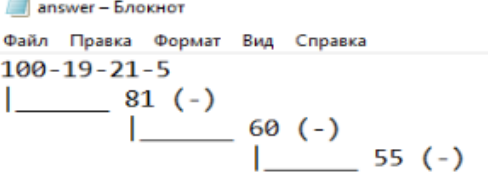
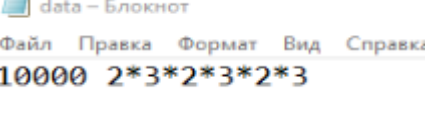
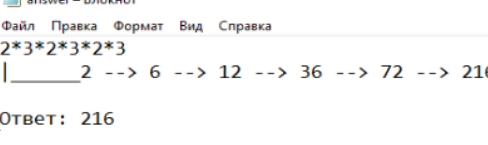
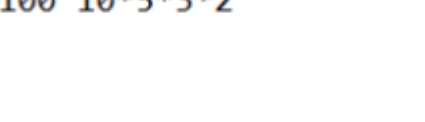
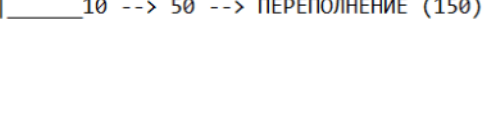
Далее считывается первое число и знак. Если знак - знак конца строки, то пишется ответ или переполнение. Если какой-либо арифметический знак, то после проверки на переполнение вход в рекурсивную функцию calc() и ответ.

Заккрытие файла dt.f и очищение памяти строки dt.s.





## Тестирование.

Тестирование программы представлено в таблице 1.

Таблица 1.

Номер теста:	Входные данные: (файл: "data.txt")	Выходные данные: (файл: "answer.txt")
1		
2		
3		
4		
5		



6	<p>10000 20*10*20-10*10+24+10-34*2+1*5</p>	<p>20*10*20-10*10+24+10-34*2+1*5</p> <pre>  _____20 --&gt; 200 --&gt; 4000        _____10 --&gt; 100        _____3900 (-)              _____3924 (+)                    _____3934 (+)                          _____34 --&gt; 68                          _____3866 (-)                                _____1 --&gt; 5                                _____3871 (+) </pre> <p>Ответ: 3871</p>
7	<p> data – Блокнот</p> <p>Файл Правка Формат Вид Справка</p> <p>100 10-24*2+7*7*7-100*20+1</p>	<p> answer – Блокнот</p> <p>Файл Правка Формат Вид Справка</p> <p>10-24*2+7*7*7-100*20+1</p> <pre>  _____24 --&gt; 48  _____38 (-)        _____7 --&gt; 49 --&gt; ПЕРЕПОЛНЕНИЕ (343) </pre>
8	<p> data – Блокнот</p> <p>Файл Правка Формат Вид Справка</p> <p>100 99</p>	<p> answer – Блокнот</p> <p>Файл Правка Формат Вид Справка</p> <p>Ответ: 99</p>

## **Выводы.**

Были изучены основные понятия рекурсии и был получен опыт работы с рекурсивными функциями на языке программирования Си. Была разработана программа, считающая арифметическое выражение с помощью рекурсивных функций.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файлов: main.c

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

typedef struct data {                                //структура для хранения
всех необходимых данных
    int sum;    //сумма всего выражения
    int mul;    //переменная хранит промежуточные значения
произведений
    int max;    //порог переполнения
    int tabs;   //количество табуляций
    char* s;    //строка-выражение
    FILE* f;    //файл-ответ
} data;

char* readsent() {                                    //функция посимвольного
считывания строки из файла
    int size = 10;
    int n = 0;
    FILE* f = fopen("data.txt", "r");
    char* sent = malloc(size * sizeof(char));
    while ((sent[n] = fgetc(f)) != EOF ) {
        if (++n >= size) {
            size += 10;
            sent = realloc (sent, size * sizeof(char));
        }
    }
    sent[n]='\n';
    sent[n+1]='\0';
    fclose(f);
    return sent;
}

void shift(char* sent, int ind) {                      //функция сдвига
строки-выражения влево (удаление)
    int i = 0;
    for(i = ind; i<strlen(sent); i++)
        sent[i-ind] = sent[i];
    sent[strlen(sent)-ind]='\0';
}

int numlen(int num) {                                  //функция измерения длины
числа
    int n = 0;
    while (num>0) {
        num = num / 10;
        n++;
    }
}
```

```

        return n;
    }

void printtabs(FILE* f, int n) {           //функция отвечающая за
отступы в файле-ответе
    int i;
    for(i=0; i<n; i++)
        fprintf(f, "\t");
    fprintf(f, "|_____");
}

void mult(data *dt) {                     //рекурсивная функция
умножения
    int num=0;
    char sign;
    char sign2;
    sscanf(dt->s, "%c%d%c", &sign, &num, &sign2);
    dt->mul = dt->mul * num;               //подсчет произведения
    if (dt->mul > dt->max)                  //переполнение
        return;
    fprintf(dt->f, "%d", dt->mul);
    shift(dt->s, numlen(num)+1);
    if(strchr("*", sign2)) {
        fprintf(dt->f, " --> ");
        mult(dt);                       //следующий множитель
    }
}

void calc(data *dt) {                     //рекурсивная функция
счета
    int num = 0;
    char sign;
    char sign2;
    int mul = 0;
    sscanf(dt->s, "%c%d%c", &sign, &num, &sign2);
    if (num > dt->max) {                   //переполнение числа
        printtabs(dt->f, dt->tabs++);
        fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", num);
        dt->sum = num;
        return;
    }
    if (strchr("\n", sign)) {
        return;
    }
    if (strchr("+", sign)) {              //сложение
        if(strchr("*", sign2)) {         //сумма с произведение
            shift(dt->s, numlen(num)+1);
            printtabs(dt->f, dt->tabs);
            fprintf(dt->f, "%d --> ", num);
            dt->mul = num;
            mult(dt);                     //подсчет произведения
            if (dt->mul > dt->max) {         //переполнение произведения
                fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", dt->mul);
                dt->sum = dt->mul;
                return;
            }
        }
    }
}

```

```

    }
    dt->sum = dt->sum + dt->mul;
    if (dt->sum > dt->max) { //переполнение суммы
        fprintf(dt->f, "\n");
        printtabs(dt->f, dt->tabs++);
        fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", dt->sum);
        return;
    }
    fprintf(dt->f, "\n");
    printtabs(dt->f, dt->tabs++);
    fprintf(dt->f, "%d (+)\n", dt->sum);
}
else { //сумма двух слагаемых
    dt->sum= dt->sum + num;
    if (dt->sum > dt->max) { //переполнение суммы
        printtabs(dt->f, dt->tabs++);
        fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", dt->sum);
        return;
    }
    shift(dt->s, numlen(num)+1);
    printtabs(dt->f, dt->tabs++);
    fprintf(dt->f, "%d (+)\n", dt->sum);
}
calc(dt); //переход к следующему действию
return;
}
if (strchr("-", sign)) { //вычитание
    if(strchr("*", sign2)) { //разность с произведением
        shift(dt->s, numlen(num)+1);
        printtabs(dt->f, dt->tabs);
        fprintf(dt->f, "%d --> ", num);
        dt->mul = num;
        mult(dt); //подсчет произведения
        if (dt->mul > dt->max) { //переполнение произведения
            fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", dt->mul);
            dt->sum = dt->mul;
        }
        return;
    }
    dt->sum = dt->sum - dt->mul; //подсчет разности с
произведением
    fprintf(dt->f, "\n");
    printtabs(dt->f, dt->tabs++);
    fprintf(dt->f, "%d (-)\n", dt->sum);
}
else { //разность с простым вычитаемым
    dt->sum= dt->sum - num;
    shift(dt->s, numlen(num)+1);
    printtabs(dt->f, dt->tabs++);
    fprintf(dt->f, " %d (-)\n", dt->sum);
}
calc(dt); //переход к следующему действию
return;
}
if (strchr("*", sign)) { //умножение

```

```

        shift(dt->s, numlen(num)+1);
        printtabs(dt->f, dt->tabs++);
        fprintf(dt->f, "%d --> ", dt->sum);
        dt->mul = dt->sum * num;          //произведение из двух
множителей
        if (dt->sum > dt->max) {          //переполнение произведения
            fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", dt->sum);
            return;
        }

        if (strchr("?", sign2)){
            fprintf(dt->f, "%d --> ", dt->mul);
            mult(dt);                    //произведение из 3+ множителей
        }
        else
            fprintf(dt->f, "%d", dt->mul);
        dt->sum = dt->mul;
        if (dt->sum > dt->max) {          //переполнение произведения
            fprintf(dt->f, "ПЕРЕПОЛНЕНИЕ (%d)\n", dt->sum);
            return;
        }
        fprintf(dt->f, "\n");
        calc(dt);                      //переход к следующему действию
        return;
    }
}

```

```

}

int main() {
    data dt;
    dt.sum = 0;
    dt.tabs = 0;
    dt.s = readsent();
    dt.f = fopen("answer.txt", "w");
    char sign;

    sscanf(dt.s, "%d", &dt.max);
    shift(dt.s, numlen(dt.max)+1);

    sscanf(dt.s, "%d%c", &dt.sum, &sign);
    shift(dt.s, numlen(dt.sum));
    if (strchr("\n", sign))            //если было введено только одно
число
        if (dt.sum < dt.max)
            fprintf(ft.f, "Ответ: %d\n", dt.sum);
        else
            fprintf(ft.f, "ПЕРЕПОЛНЕНИЕ");
    else {
        if (dt.sum < dt.max) {
            fprintf(f1, "%d%s", dt.sum, dt.s);
            calc(&dt);                //вход в рекурсивную функцию подсчета
            if (dt.sum < dt.max)
                fprintf(dt.f, "\nОтвет: %d\n", dt.sum);
        }
        else

```

```
        fprintf(dt.f, "ПЕРЕПОЛНЕНИЕ");  
    }  
  
    fclose(dt.f);  
    free(dt.s);  
    return 0;  
}
```