

Теоретические сведения:

см. методичку К.Е. Климентьева и лекцию 1.

Общее задание:

Реализовать заданный в варианте алгоритм в 3 вариантах:

1. на языке C;
2. на языке C с упрощением;
3. на языке ассемблера NASM (<https://www.nasm.us/pub/nasm/releasebuilds/2.15.05/win64/>).

Под упрощением программы понимается приближение программы на C к ассемблерному коду. Для этого необходимы:

1. замена составных операций на последовательность простых ($a = b + c + d$ на $a = b$;
 $a += c$; $a += d$)
2. замена блоков
if(условие) { <код ветки 1> }
else { <код ветки 2> }
на
if (!условие) goto .<метка else>
<код ветки 1>
goto .<метка end>
.<метка else>:
<код ветки 2>
.<метка end>:
3. замена циклов на if + goto

Пример упрощения программы приведен в приложении А.

Ввод/вывод программы осуществлять с помощью printf/scanf/gets/puts для языка C и с помощью соответствующих макросов из стандартного комплекта SASM (<http://dman95.github.io/SASM/> , список макросов – внизу страницы) для языка ассемблера.

Массивы реализовывать, как глобальные переменные (для ассемблера - в секции .data или .bss).

При работе со строками максимальную длину строки установить не менее 255.

При работе со строками считать, что допустимый алфавит состоит из цифр, английских букв обоих регистров и символов `[]()+-*/=&?!/\'";:,@.`, ([ASCII](#)-символы, при желании можно добавить кириллицу)

При работе с числовыми массивами максимальную длину массива установить не менее 100

Если в задании предполагается ввод массива пользователем, то пользователь сначала вводит размер массива, а затем сам массив.

Для массивов чисел, если не указано иное, тип элемента - int.

Проверку ввода пользователя можно не делать.

Задания на лабораторную:

Легкий уровень:

1. Для введенной пользователем строки вывести инвертированную строку и строку, состоящую только из четных элементов исходной строки.
2. Для введенной пользователем строки и числа N вывести 2 строки: циклически сдвинутую на N вправо и на N влево.
3. Для введенной пользователем строки проверить, является ли она палиндромом
4. Для введенной пользователем строки проверить, сколько раз в ней встречаются буквы, с которых начинаются ваша фамилия и имя (регистр букв игнорировать)
5. Для введенного пользователем массива рассчитать минимум и максимум.
6. Проверить введенное пользователем число на простоту, вывести ответ.
7. Для введенного пользователем вектора (представляется в виде массива чисел) вывести квадрат его нормы.
8. Для введенного пользователем массива чисел подсчитать количество *битов* со значением 1.

Средний уровень

1. Зашифровать введенную пользователем строку шифром Цезаря.
2. Реализовать аналог функции `strcmp`. Продемонстрировать результат работы алгоритма на 2-х введенных строках.
3. Для введенных пользователем строк проверить, содержит ли первая строка вторую.
4. Для пользовательской строки вывести частоту встречаемости каждого символа. (помните, что символы - это целые числа в диапазоне 0-255).
5. Для пользовательского массива чисел рассчитать и вывести массив с результатом расчета скользящего среднего в окне размером 5.
6. Посчитать количество уникальных элементов в массиве чисел типа `short`.

Сложный уровень:

1. Отсортировать введенный пользователем массив пузырьком
2. Отсортировать введенный пользователем массив вставками
3. Отсортировать введенный пользователем массив выбором
4. Реализовать [линейный конгруэнтный генератор](#) псевдослучайных чисел. Параметры генератора считать постоянными. Используя введенное пользователем число в качестве начальной точки, вывести 100 сгенерированных чисел.
5. Реализовать генератор псевдослучайных чисел [xorshift128](#). Используя введенное пользователем число в качестве начальной точки, вывести 100 сгенерированных чисел.
6. Реализовать генератор псевдослучайных чисел [BlumBlumShub](#). Параметры p и q выбрать самостоятельно. В качестве результата использовать 1 или 2 младших байта полученного числа. Используя введенное пользователем число в качестве начальной точки, вывести 100 сгенерированных чисел.
7. Реализовать вычисление контрольной суммы [CRC32](#) для введенной строки. Вывести полученное значение.
8. Отсортировать введенный пользователем массив шейкерной сортировкой

Приложение А. Пример выполнения л/р

Задача: написать программу, которая считывает 2 базисных числа X и Y и выводит запись числа X в системе счисления с основанием Y.

Код на C:

```
#include <stdio.h>

char buffer[65]; // макс. длина - 64 символа в двоичной записи

void main(){
    unsigned int x;
    unsigned int radix;
    scanf("%u %u", &x, &radix);
    buffer[64] = 0; // end of line
    int count = 63;
    do{
        unsigned int digit = x % radix;
        x /= radix;
        if (digit < 10)
            digit = '0'+digit;
        else
            digit = 'a'+digit-10;
        buffer[count--] = digit;
    }while(x > 0);
    puts(buffer+count+1);
}
```

Упрощенный код на C:

```
#include <stdio.h>

char buffer[65];

void main(){
    unsigned int x;
    unsigned int radix;
    scanf("%u", &x);
    scanf("%u", &radix);
    buffer[64] = 0; // end of line
    int count = 63;
    unsigned int digit;
cycle_start:
    digit = x % radix;
    x /= radix;
    if (digit >= 10) goto else_label;
    digit += '0';
    goto if_end;
else_label:
    digit += 'a' - 10;
    goto if_end;
if_end:
    buffer[count--] = digit;
}
```

```

else_label:
    digit += 'a';
    digit -= 10;
if_end:
    buffer[count] = digit;
    count-=1;
    if(x > 0) goto cycle_start;

    char* address = &buffer[count+1];
    puts(address);
}

```

Код NASM:

```

%include "io64.inc"

section .bss
buffer: resb 65

section .text
global main

main:
    ; Переменные:
    ; EAX - x
    ; EDI - radix
    ; EDX - digit (обновляется при делении)
    ; ECX - count
    GET_UEDEC 4, eax
    GET_UEDEC 4, edi
    mov byte[buffer+64], 0
    mov ecx, 63
.cycle_start:
    xor edx, edx
    div edi
    cmp edx, 10
    jae .else_label
    add edx, '0'
    jmp .if_end
.else_label:
    lea edx, [edx+'a'-10]
.if_end:
    mov [buffer+ecx], dl
    sub ecx, 1
    test eax, eax
    jnz .cycle_start
    lea rdx, [buffer + rcx + 1]
    PRINT_STRING [rdx]

```

```
xor eax, eax  
ret
```