

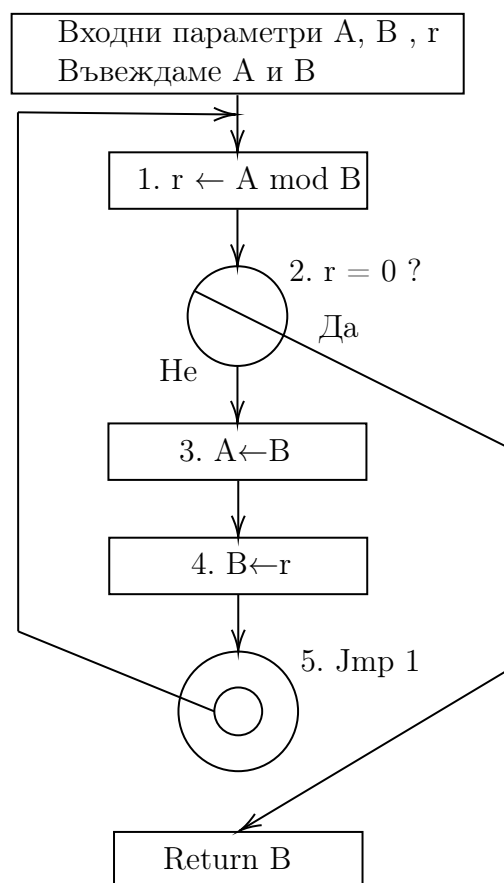
# Увод в алгоритмите и програмирането

## Домашна работа №2

Никола Николов, F106501

**Задача 1:** Съставете програма за алгоритъма на Евклид, в която няма оператор за цикъл. Ползвайте схемата от предходното домашно. Предайте снимка на екран с кода и с изхода от изпълнение.

Решение:



```
#include <iostream>

int GCD(int a, int b)
{
    int r = a%b;

    if(r==0){
        return b;
    }
    else{
        a = b;
        b = r;

        return GCD(a,b);
    }
}

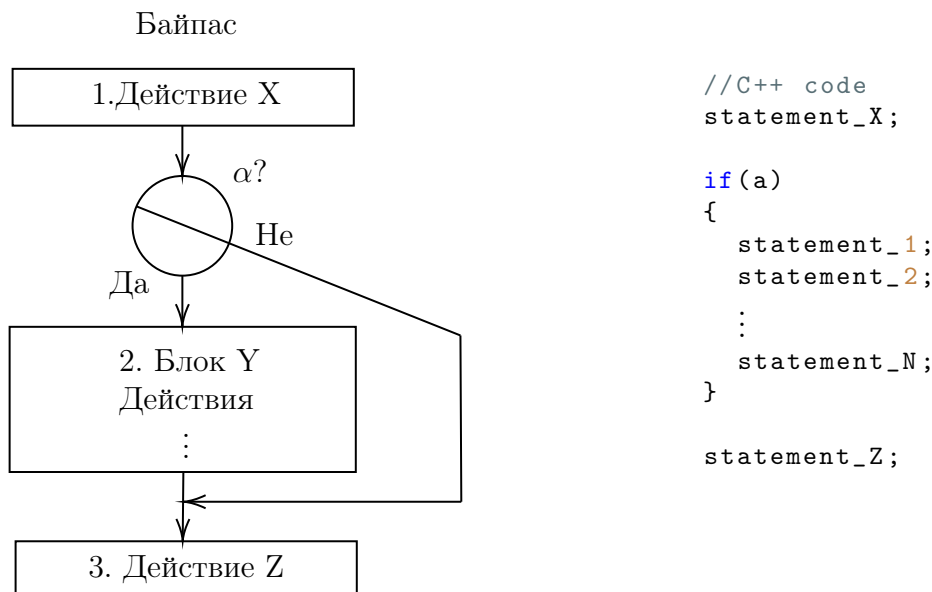
int main()
{
    int a,b;
    std::cin >> a >> b;
    std::cout << GCD(a,b);
    return 0;
}
```

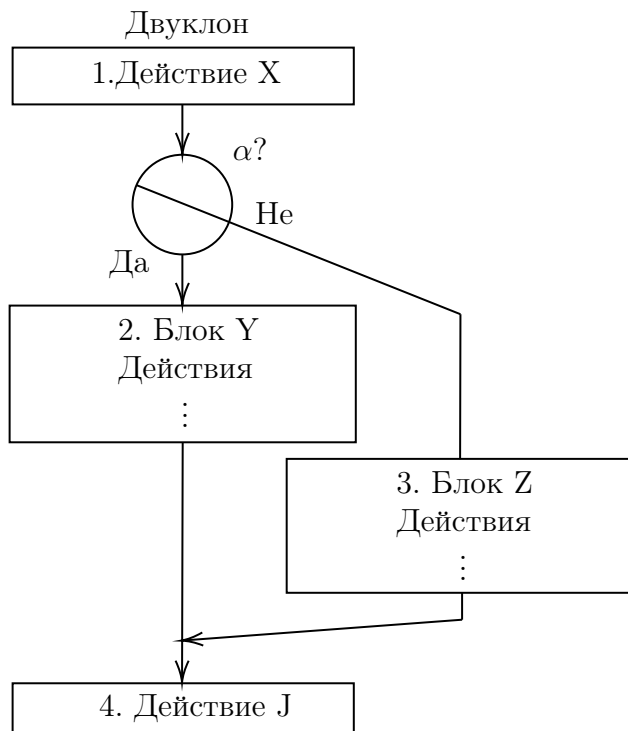
```
nn@NNPC:~/Programming/NBU/Algoritmi/Week2/HW2$ ./zad1 ; echo
49 42
7
```

Фигура 1: Изход от кода на задача 1.

**Задача 2:** Като се базирате на основните елементи на управление, дадени в презентациите към тази тема, съставете петте схеми на основните Конструкции на Управление - Клоновете (двуклон и байпас) и на Повторенията (цикли с предусловие, със следусловие и по брояч). Може на ръка, може да копирате схемите, както ви е по-удобно. Тези конструкции са изучавани в уводните курсове по програмиране, сега задачата е да видите мястото на условните и безусловни преходи като елементи на управление. Прегледайте анимациите към тази тема, за да си припомните изученото по програмиране. Успоредно на схемите, маркирайте програмни оператори на познат език, както е показано в презентациите и тук:

Решение:

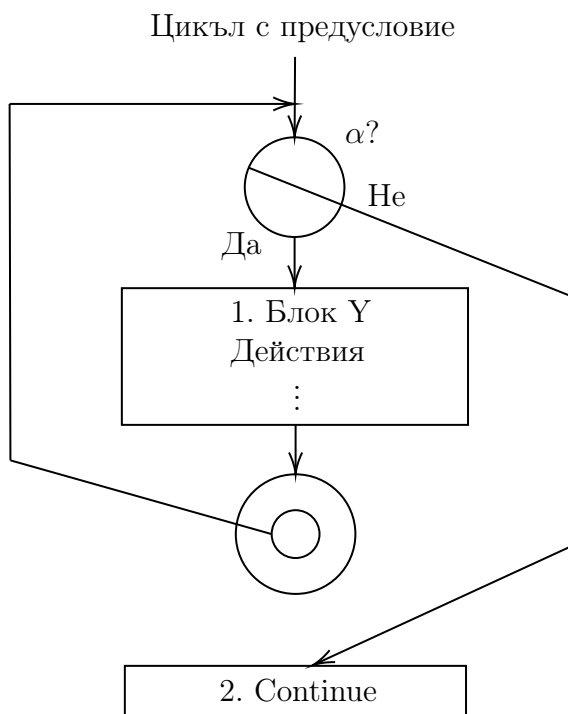




```
//C++ code
statement_X;

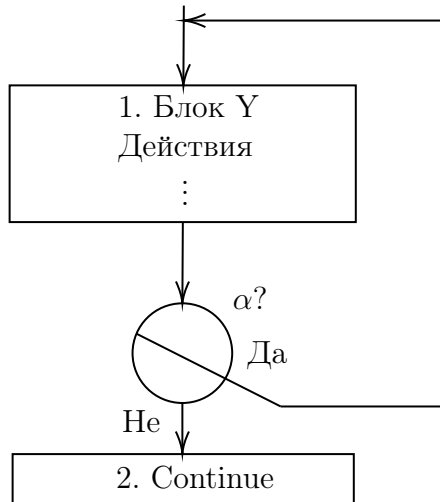
if(a)
{
    statement_Y1;
    statement_Y2;
    ⋮
    statement_YN;
}
else
{
    statement_Z1;
    statement_Z2;
    ⋮
    statement_ZN;
}

statement_J;
```



```
//C++ code
⋮
while(a)
{
    statement_1;
    ⋮
    statement_N;
}
⋮
```

### Цикъл със следусловие

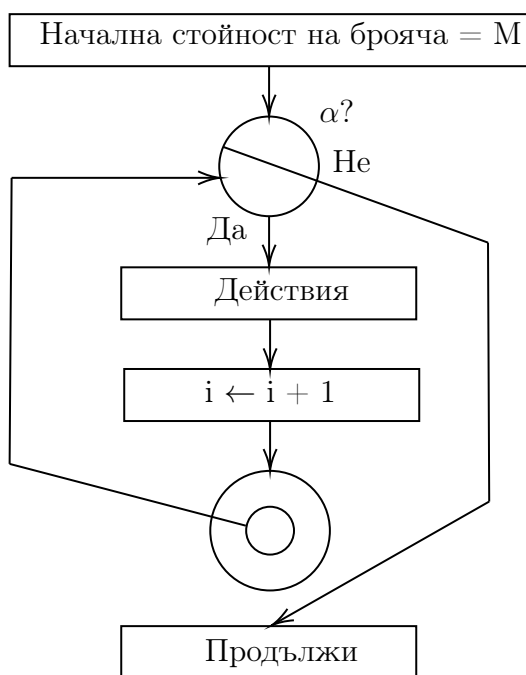


//C++ code

```

⋮
do{
    statement_1;
    ⋮
    statement_N;
}while(a);
⋮
  
```

### Цикъл с вграден брояч



//C++ code

```

⋮
for(int i = M; i < N ; i++){
    statement_1;
    ⋮
    statement_N;
}
⋮
  
```

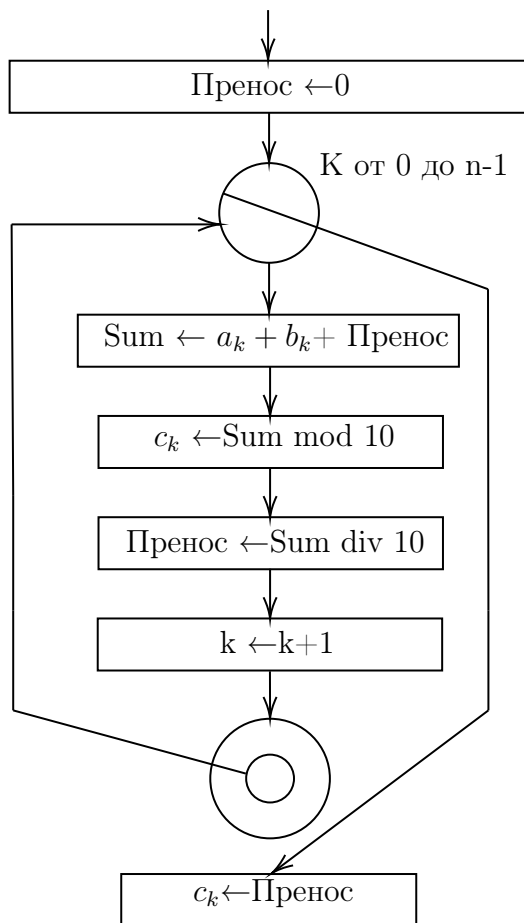
*\*Забележка: Цикълът с вграден брояч може да работи както за '<', така и за '≤', '>' и '≥' според нуждите на програмиста. При случая '≥' действието спрямо брояча в цикъла би следвало да е 'i ← i - 1', като това трябва да бъде отразено в програмния код:*

```
for(int i = ... ; i ≥ ... ; i-- )
```

**Задача 3:** Съставете схемата на управление за събиране на две числа (зададени с два вектора) в позиционна бройна система (използване на операциите с цели числа, по-точно - делене с остатък). Схемата е дадена в презентация към тази тема, а алгоритъмът е подробно разработван в час. Встрани от схемата напишете програмен текст на предпочетан от вас език.

Решение:

Схема за събиране на две числа в десетична позиционна бройна система



```
#include <iostream>

void sumOfTwoNumbers(int *a, int *b,
int *c, int n);

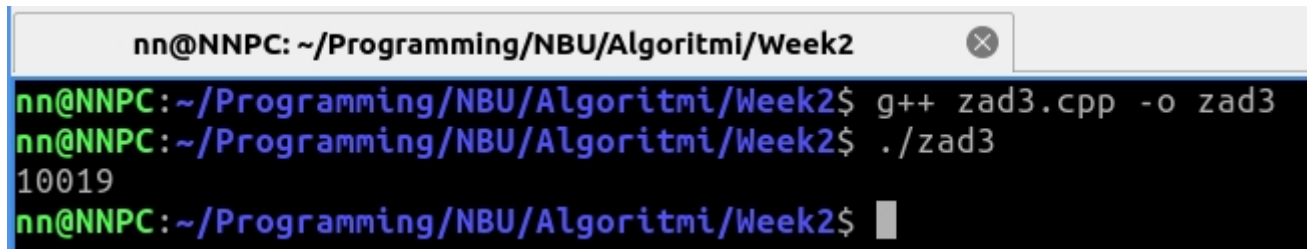
int main()
{
    int a[] = {5,3,2,0};
    int b[] = {4,8,7,9};
    int n = sizeof(a)/sizeof(*a);
    int c[n+1] = {0,};

    sumOfTwoNumbers(a,b,c,n);
    //Print summed number:
    for(int i=n;i>=0;i--){
        std::cout << c[i];
    }
    std::cout << '\n';

    return 0;
}

void sumOfTwoNumbers(int *a, int *b,
int *c, int n)
{
    int p = 0;
    int sum = 0;
    for(int k=0;k<n;k++){
        sum = a[k] + b[k] + p;
        c[k] = sum % 10;
        p = sum / 10;
    }
    c[n] = p;
}
```

Задача 4:



```
nn@NNPC: ~/Programming/NBU/Algoritmi/Week2
nn@NNPC:~/Programming/NBU/Algoritmi/Week2$ g++ zad3.cpp -o zad3
nn@NNPC:~/Programming/NBU/Algoritmi/Week2$ ./zad3
10019
nn@NNPC:~/Programming/NBU/Algoritmi/Week2$
```