

# Тема 4

## Циклические алгоритмы

*Задача:* выучить ответы на вопросы.

*Входные данные:* количество строк в тексте, текст ответов.

Алгоритм решения:

Если текст ответов прочитан столько раз, сколько в нем строк, то отдыхать.

Иначе:

    прочитать текст ответов; запомнить, что число прочтений  
    увеличилось;

Если текст ответов прочитан столько раз, сколько в нем строк, то отдыхать.

Иначе:

    прочитать текст ответов; запомнить, что число прочтений  
    увеличилось;

...

Альтернативный алгоритм:

    повторить следующее действие столько раз, сколько строк в тексте:  
    прочитать текст ответов.

# Синтаксические разновидности циклов

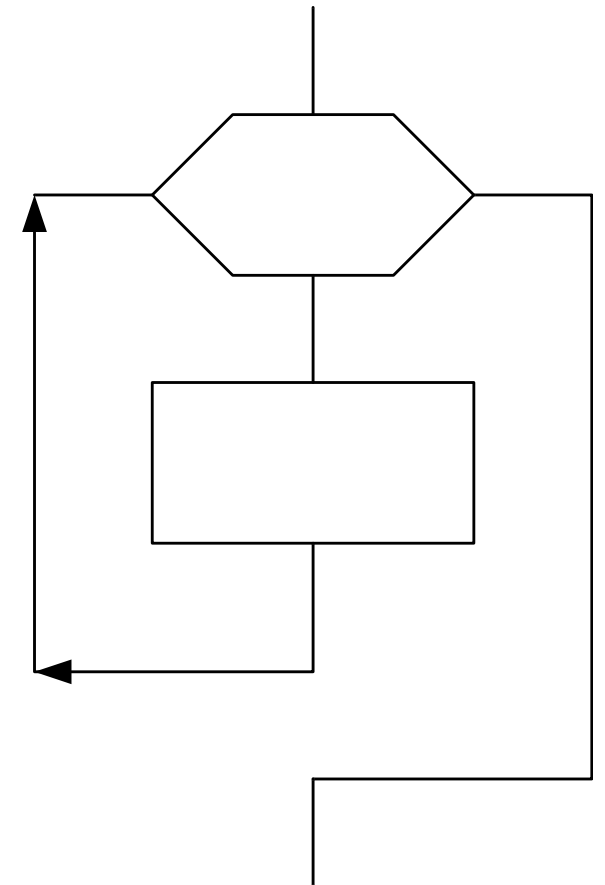
- Цикл со счетчиком
- Цикл с предусловием
- Цикл с постусловием
- Range-based циклы со счетчиком

# Цикл со счетчиком

Обычно используется когда известно число повторений тела цикла  
for (выражение1; выражение2; выражение3)

тело\_цикла

При первом заходе в цикл вычисляется выражение1, затем в случае истинности условия, записанного в выражение2 выполняется тело\_цикла. После завершения тела цикла вычисляется выражение3 и снова проверяется выражение2. Цикл заканчивается, когда выражение2 принимает значение false.



# Простейшие примеры цикла со счетчиком

```
/*бесконечный цикл*/
```

```
for( ; ; )
```

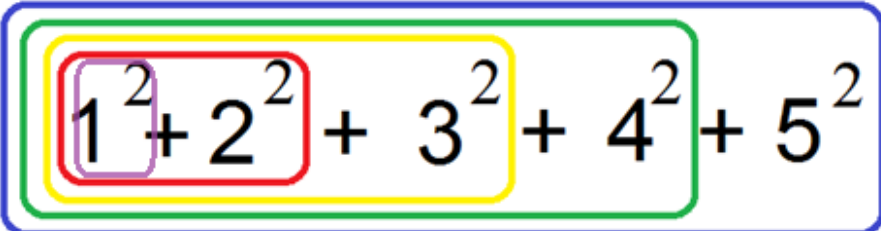
```
    cout << "Бесконечный цикл" << endl;
```

```
/*ВЫВОДИТ В СТОЛБИК значения от 1 до 10*/
```

```
for(int i = 1; i <= 10; i++)
```

```
    cout << i << endl;
```

*Задача:* вычислить сумму ряда:

$$s = \sum_{i=1}^n i^2$$


*Входные данные:*  $n$  – натуральное число.

Алгоритм решения:

1. Записать в переменную результата ( $s$ ) значение 0.
2. Выполнять следующую последовательность действий пока  $i$  не превышает  $n$  (т.е.  $n$  раз):

    Вычислить  $i^2$ ;

    Прибавить  $i^2$  к текущему  $s$ ;

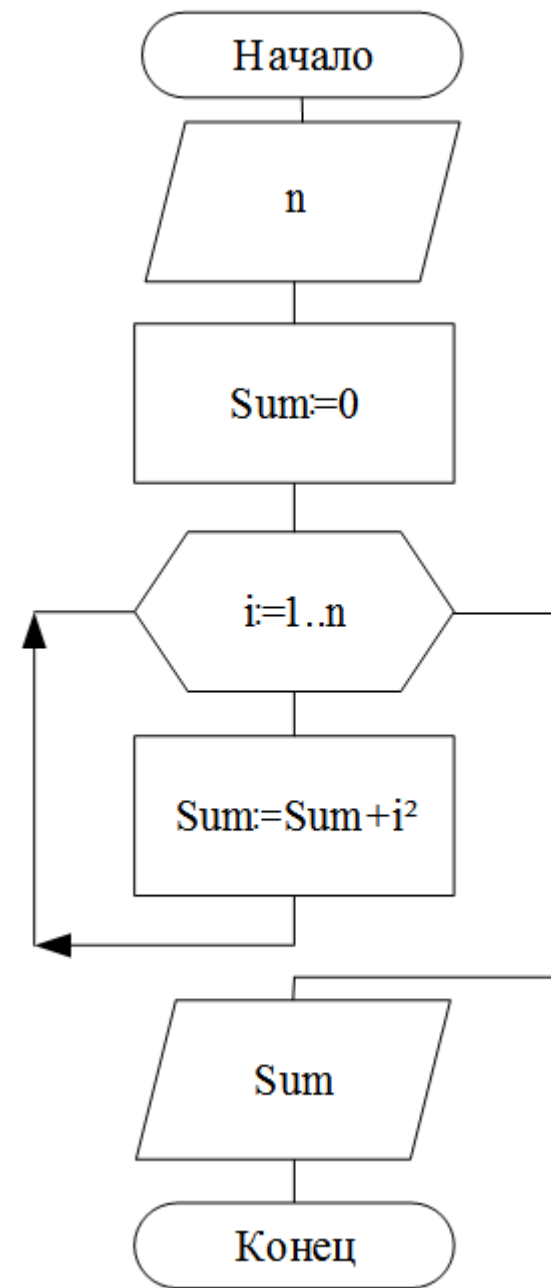
    Перезаписать текущее  $s$ ;

    Увеличить  $i$  на 1.

# Сумма ряда

```
int n;  
cin >> n;  
long Sum = 0;  
for (int i = 1; i <= n; i++)  
    Sum = Sum + i * i; //Sum += i * i;  
cout << Sum;
```

```
/* another way  
for (int i = n; i >= 1; i--)  
    Sum = Sum + i * i;  
*/
```



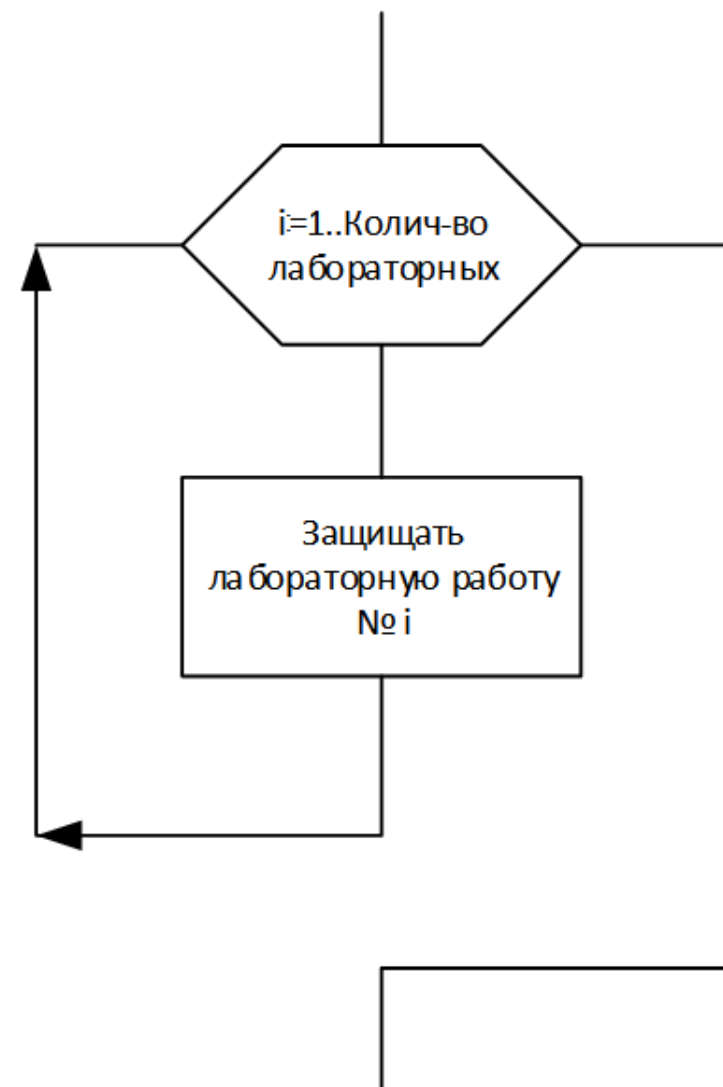
## Вычисление факториала (счетчик)

```
typedef unsigned long long Ull;  
Ull n;  
cin >> n;  
Ull fact = 1;  
for (Ull i = 1; i <= n; i++)  
    fact *= i;  
cout << fact;
```



# Цикл со счетчиком в дисциплине ОАиП

```
for (unsigned i = 1; i <= labsCount; i++)  
    protectLab(i);
```



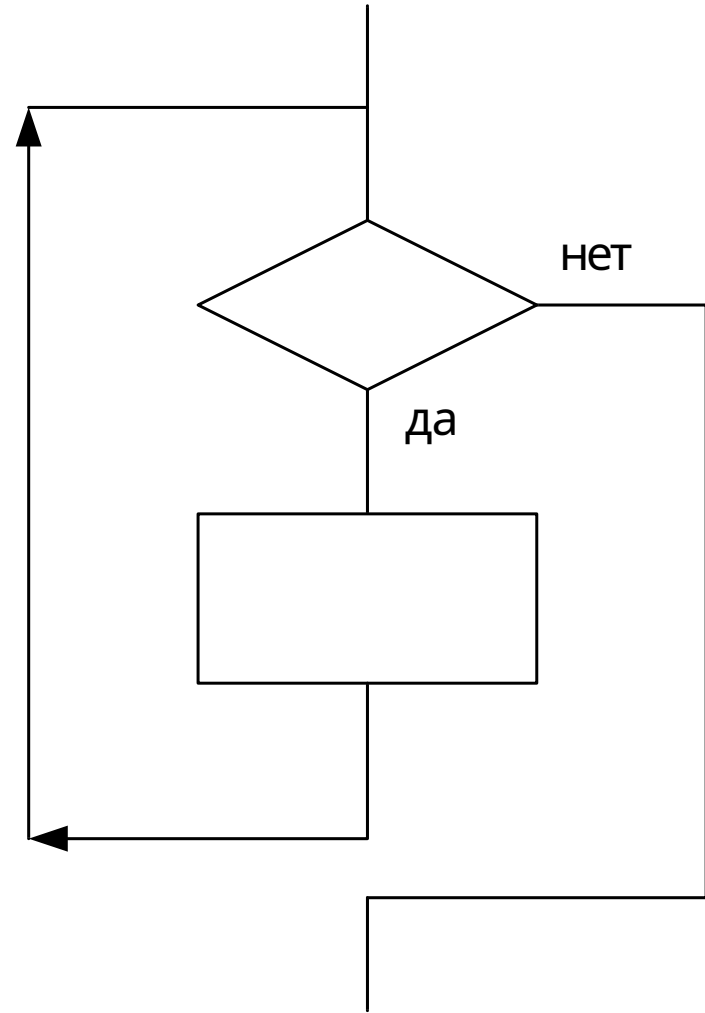
# Цикл с предусловием

Обычно применяется, когда число повторений тела цикла зависит от истинности некоторого условия

```
while (выражение)  
    тело_цикла
```

Если выражение в `while` дает значение `true`, выполняется тело\_цикла, иначе цикл прекращается.

Конструкция цикла данного типа позволяет ни разу не выполнить тело цикла, если условие изначально ложно.



# Примеры циклов с предусловием

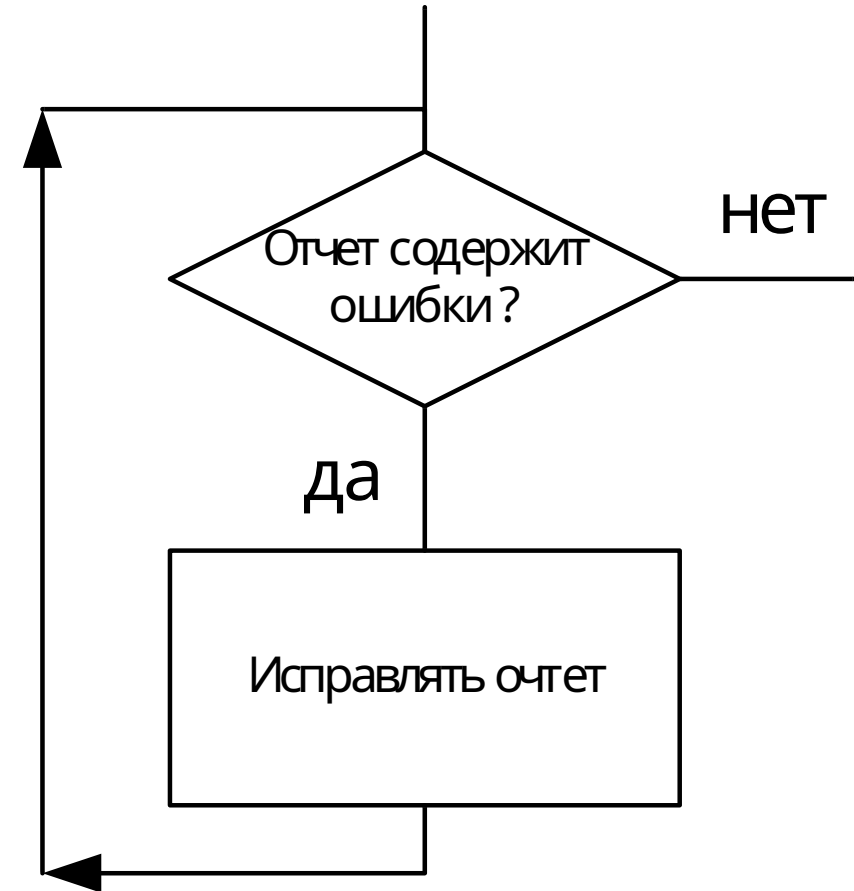
```
while (true) //бесконечный  
    cout << "Бесконечный цикл";  
  
int i, a = 5;  
while (a > 0) //бесконечный  
    i++;  
while (!a) //не выполнится ни разу  
    ++a;
```

# Вычисление факториала (предусловие)

```
int i = 1, fact = 1, n;  
cin >> n;  
while (++i <= n)  
    fact *= i;  
cout << fact;
```

# Цикл с предусловием в дисциплине ОАиП

```
while (errCount)
    errCount = CorrectErr();
```



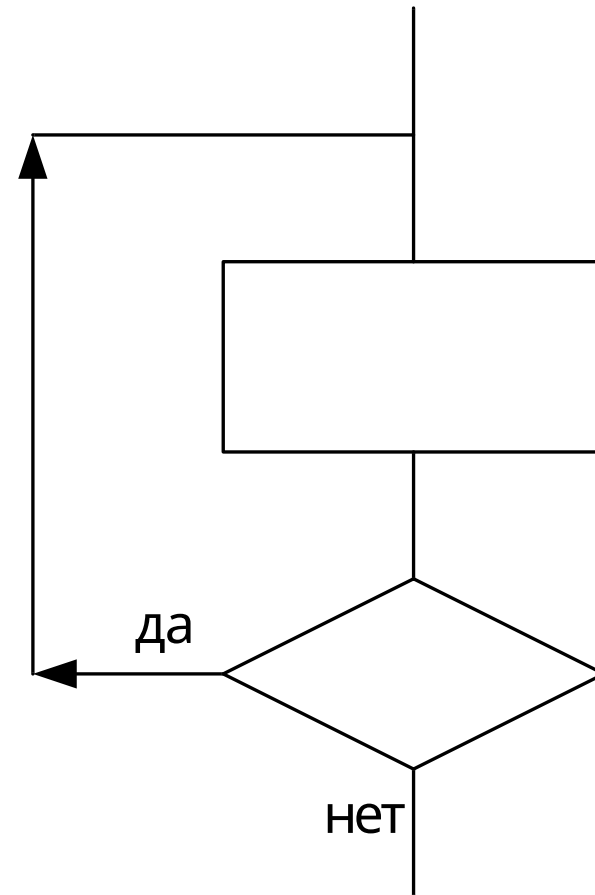
# Цикл с постусловием

Обычно применяется, когда число повторений тела цикла зависит от истинности некоторого условия.

```
do  
    тело_цикла  
while (выражение);
```

тело\_цикла выполняется, пока выражение в `while` дает значение `true`.

Конструкция цикла данного типа позволяет выполнить тело цикла по крайней мере один раз.



# Примеры циклов с постусловием

```
do
```

```
    cout << "Бесконечный цикл";
```

```
while (true); //бесконечный
```

```
int i, a = 5;
```

```
do
```

```
    i++;
```

```
while (a > 0); //бесконечный
```

```
do
```

```
    ++a;
```

```
while (!a); //выполнится 1 раз
```

# Вычисление факториала (постусловие)

```
//факториал n  
int i = 1, fact = 1, n;  
cin >> n;  
do  
    fact *= i;  
while (++i <= n);  
cout << fact;
```

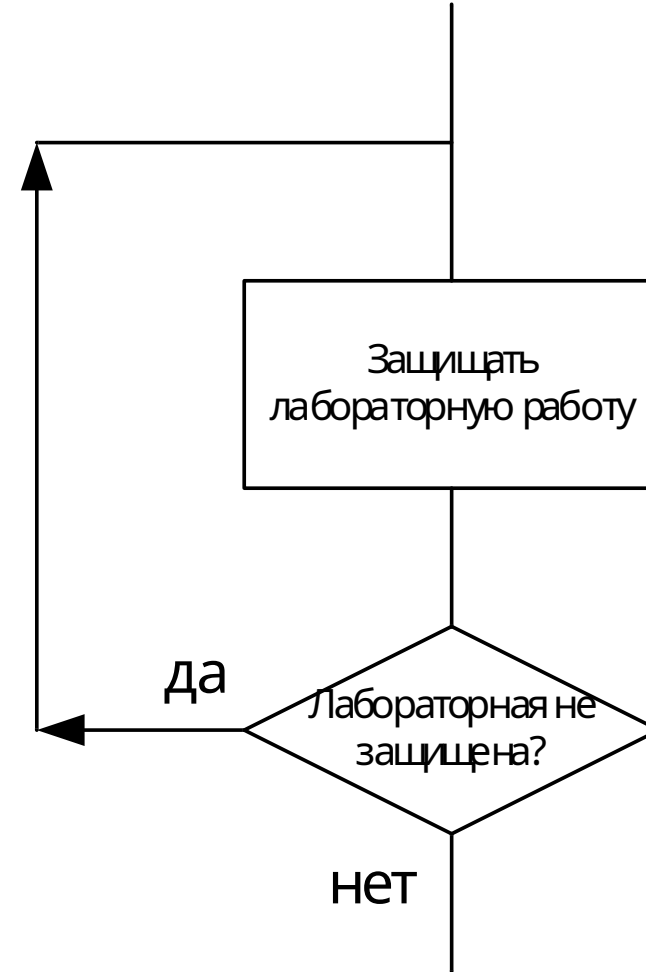


# Цикл с постусловием в дисциплине ОАиП

do

```
    flag = protect_lab();
```

```
while (!flag);
```



## Сравнение 3-х типов циклов: вывести квадраты чисел [0,100)

```
for (int i = 0; i < 100; ++i)
```

```
    cout << i * i << " ";
```

```
//---
```

```
int i = 0;
```

```
while (i < 100)
```

```
    cout << i * i++ << " ";
```

```
//---
```

```
int i = 0;
```

```
do
```

```
    cout << i * i << " ";
```

```
while (++i < 100);
```

## range-based циклы со счетчиком

Применяется, когда необходимо перебрать все элементы некоторого конечного контейнера, не заботясь об индексах

```
for (объявление_элемента : контейнер)
    инструкция;
```

```
string s = "I love programming!";
for (char ch : s)
    cout << ch << endl;
```

```
for (auto ch : s)
    cout << ch << endl;
```

# Оператор прерывания цикла break

Досрочное завершение цикла с последующим выходом из него.  
Обычно используется в сочетании с условным оператором.

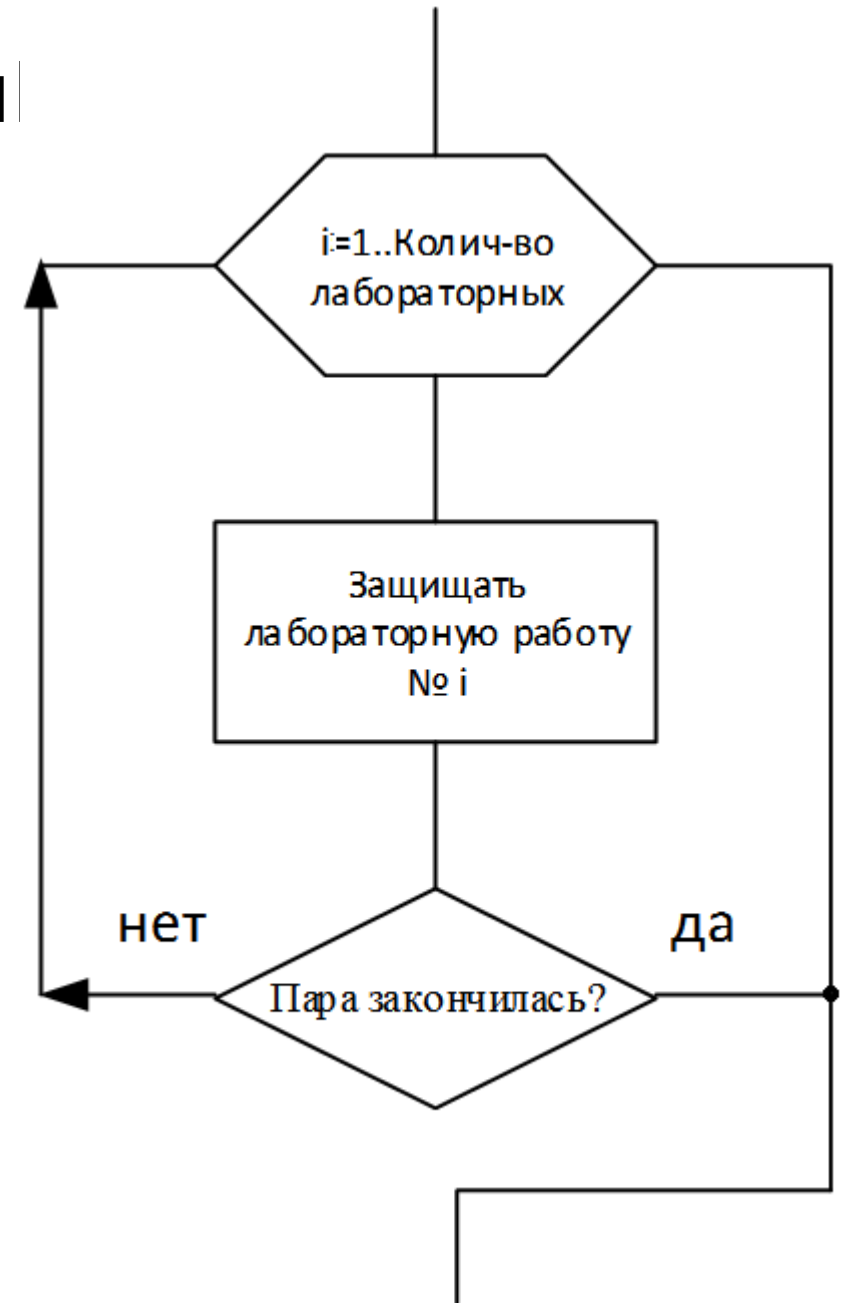
```
while (условие_цикла) {  
    действие1;  
    ...  
    действиеN;  
    if (условие_выхода) /* если истинно, то выполняется  
действие_после_цикла*/  
        break;  
    действиеN1; // иначе цикл продолжается  
    ...}  
действие_после_цикла; /*сюда попадем когда условие_цикла  
станет ложным или условие_выхода станет истинным*/
```

# Использование оператора break в цикле с несколькими условиями выхода

```
for( ; ; ){  
    ...  
    if(условие1)  
        break;  
    if(условие2)  
        break;  
    ...  
}
```

# Оператор break в дисциплине ОАи

```
unsigned i;  
for (i = 1; i <= labsCount; i++){  
    protectLab(i);  
    if(lesson_is_over)  
        break;  
}
```



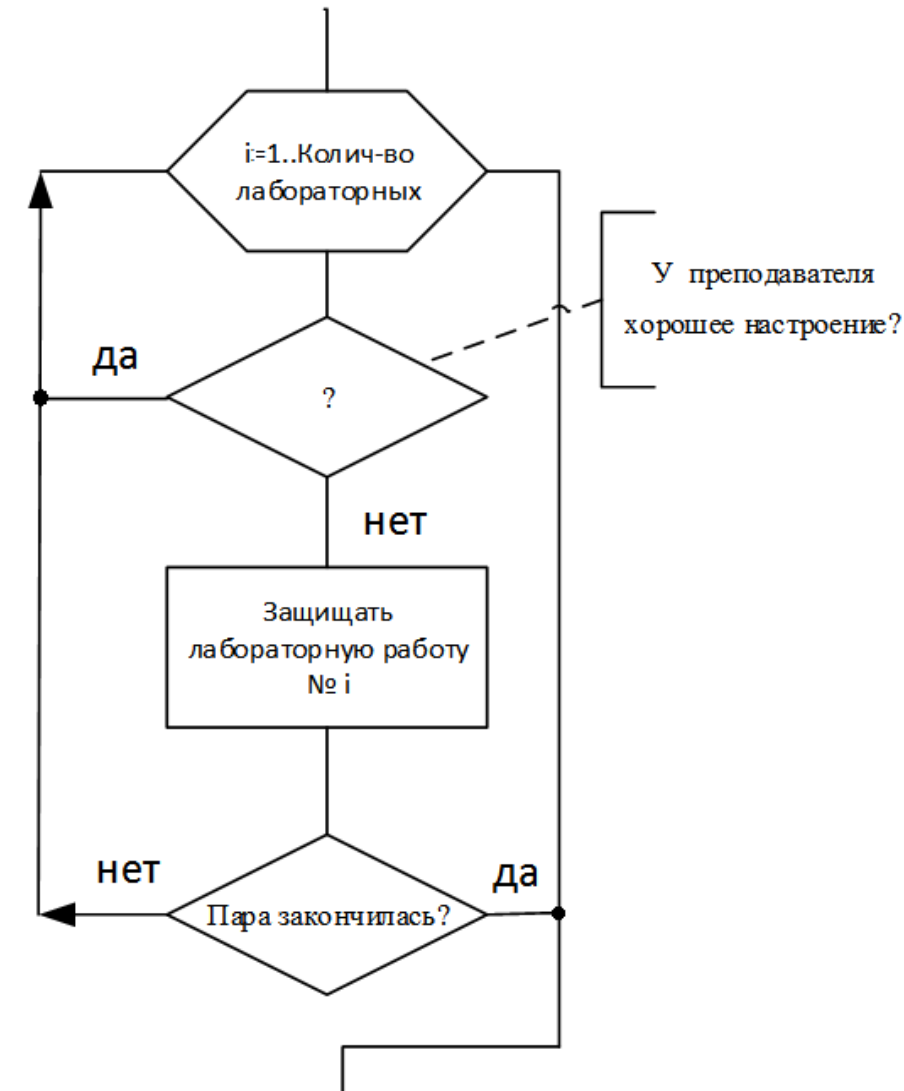
# Оператор прерывания continue

Досрочное завершение цикла с последующим переходом к проверке условия цикла. Обычно используется в сочетании с условным оператором.

```
while (условие_цикла){  
    действие1;  
    ...  
    действиеN;  
    if (условие_выхода) /* если истинно, то проверяем  
условие_цикла*/  
        continue;  
    действиеN1; // иначе цикл продолжается  
    ...  
}  
действие_после_цикла; /*сюда попадем только когда условие  
цикла станет ложным*/
```

# Оператор continue в дисциплине ОАиП

```
for (unsigned i = 1; i <= labsCount; i++){  
    if (t_good_mood)  
        continue;  
    protectLab(i);  
    if(lesson_is_over)  
        break;  
}
```





# Примеры применения операторов прерывания цикла

Задача 1. Студент в общежитии проводит перепись своих вещей. Он до смерти боится тараканов, поэтому, только увидев их, бросает все свои дела и убегает. Необходимо вывести на экран вещи из списка студента, но если среди вещей появится 'таракан' — немедленно выйти из цикла.

```
string thing;
int count = 0;
while (cin >> thing){ /* учебник тетрадь грязная кружка таракан
    степлер носки*/
    if (thing == "таракан")
        break;
    cout << ++count << " : " << thing << "\n";
}
```

Задача 2. Убежавший студент из предыдущего упражнения позвал на помощь с переписью вещей своего друга, который тараканов просто игнорирует. Необходимо вывести на экран вещи из списка студента, а если среди вещей появится 'таракан' — на экран его не выводить, но цикл продолжить.

# Текущий контроль

Чему будет равно значение переменной count после выполнения кода:

```
int val = 25;  
int count = 0;  
while (val--, val > 10)  
    count++;
```

# Текущий контроль

Чему будет равно значение переменной count после выполнения кода:

```
int val = 25;  
int count = 0;  
while (val--, val > 10)  
    count++;
```

**Ответ: 14**

# Текущий контроль

Укажите для каждого цикла количество исполнений его тела

```
for (int i = 2; i < 4; i++) x+=5;
```

```
for (int i = 2; i < 2; i++) x++;
```

```
for (int i = 10; i < 2; i++) x-=5;
```

```
for (int i = 10; i > 9; i--) x--;
```

```
int n = 1;
```

```
for (int i = 1; i < n; i++) n = 10;
```

# Текущий контроль

Укажите для каждого цикла количество исполнений его тела

for (int i = 2; i < 4; i++) x+=5;                   **//2**

for (int i = 2; i < 2; i++) x++;                   **//0**

for (int i = 10; i < 2; i++) x-=5;                   **//0**

for (int i = 10; i > 9; i--) x--;                   **//1**

int n = 1;

for (int i = 1; i < n; i++) n = 10;                   **//0**

# Текущий контроль

Укажите для каждого цикла сколько раз будет выполнена строка, помеченная комментарием `//?`

```
int x = 10;  
while (x != 1)  
    x--; //?
```

```
int x = 10;  
do  
    x = 5; //?  
while (x != 5);
```

```
float x = 1.01, s = 0;  
while (x < 5.6)  
    x += 0.6;  
s += x; //?
```

# Текущий контроль

Укажите для каждого цикла сколько раз будет выполнена строка, помеченная комментарием `//?`

```
int x = 10;  
while (x != 1)  
    x--; //9
```

```
int x = 10;  
do  
    x = 1; //1  
while (x != 1);
```

```
float x = 1.01, s = 0;  
while (x < 5.6)  
    x += 0.6;  
s += x; //1
```

Разные задачи на циклы,  
синтаксические особенности разных  
ТИПОВ ЦИКЛОВ



# Пример изменения счетчика на произвольное значение

*Задача:* найти сумму:

$\lg(n) + \lg(n+2) + \lg(n+4) + \dots + \lg(m-2) + \lg(m)$ ,

где  $n$  и  $m$  – целые числа одинаковой четности,  $n < m$ .

Входные данные:  $n, m$  – целые числа.

```
int n, m, i;  
double s = 0;  
cin >> n >> m;  
for (i = n; i <= m; i += 2)  
    s += log10(i);  
cout << s;
```

# Нахождение чисел Фибоначчи

*/\*находим числа не превышающие максимальное 32-х битное беззнаковое целое\*/*

```
const unsigned long limit = UINT_MAX / 2;
```

```
unsigned long prev = 0;
```

```
unsigned long last = 1;
```

```
while (prev < limit / 2) {
```

```
    cout << last << " ";
```

```
    unsigned long sum = prev + last;
```

```
    prev = last;
```

```
    last = sum;
```

```
}
```

## Пример использования составного оператора в теле цикла

*Задача:* найти сумму первых  $n$  членов геометрической прогрессии, если заданы первый член  $b$  и знаменатель  $q$  (не используя формулу). Вывести значения всех членов данной прогрессии и промежуточные суммы.

```
double b, q;
unsigned i, n;
cin >> n >> b >> q;
double sum = b;
cout << 1 << " " << b << " " << sum << endl;
for (i = 2; i <= n; i++){
    b *= q;           // или аналогичное тело
    sum += b;         // sum += b *= q;
    cout << i << " " << b << " " << sum << endl;
}
```

# Сравнение 3-х типов циклов: количество цифр в числе

```
int t;  
cin >> t;  
//---for---  
for (int k = 1; t != 0; t /= 10)  
    k++;  
//---while-  
int k = 1;  
while (t /= 10)  
    k++;  
//---do---  
int k = 0;  
do  
    k++;  
while (t /= 10);
```

# Пример использования действительного типа для счетчика

*Задача:* вычислить значение суммы ряда:

$$tg1.1+tg1.3+tg1.5+tg1.7+tg1.9+tg2.1$$

```
double rez, i;  
double sum = 0;  
for (i = 1.1; i < 2.2; i += 0.2)  
    sum += tan(i);  
cout << sum;
```

\* в качестве условия окончания цикла стоит условие ( $i < 2.2$ ), а не ( $i \leq 2.1$ ) в связи с особенностью представления действительных чисел в памяти компьютера и применения к ним операций сравнения

# Пример использования символьного типа для счетчика

*Задача:* вывести символы и соответствующие им коды в заданном диапазоне от  $a$  до  $b$ .

```
int a, b;  
cin >> a >> b;  
for (char i = a; i <= b; i++)  
    cout << i << " " << static_cast<int>(i) << endl;  
    //cout << i << " " << +i << endl;
```

# Оператор запятая в цикле for

```
//переменные-счетчики меняются одновременно  
for (int i = 0, j = n; i < n && j > 0; i += 2, j--)  
    cout << i + j << endl;
```

```
//значение одного счетчика влияет на значение другого  
for (int i = 0, j = n; i < n && j > 0; i += 2, j -= i)  
    cout << i + j << endl;
```

# Оператор запятая в цикле for

//попарно суммируются члены арифметической и геометрической прогрессии

```
int i, a0 = 1, b0 = 1, d = 2, q = 2, s = 0;
for (i = a0 + n * d, j = b0 * pow(q, n);
     i >= a0 && j >= b0;
     i -= d, j /= q){
    s += i + j;
    cout << s << endl;
}
```



Количество цифр в числе t, отличных от цифры d

```
int k = 0, m, t, d;  
cin >> t >> d;  
do  
    if (t % 10 != d)  
        k++;  
while (t /= 10);  
cout << k;
```

# Произведение всех четных цифр заданного натурального числа

```
unsigned n, prod = 1;
cin >> n;
do {
    if (n % 2 == 0)
        prod *= n % 10;
}
while (n /= 10);
cout << "prod = " << prod;
}
```

# Проверка натурального числа на простоту

```
unsigned long numb, delim;  
cout << "input the number" << endl;  
cin >> numb;  
for (delim = 2; delim <= numb / 2; delim++)  
    if (numb % delim == 0){  
        cout << "not prime, the delimiter is "  
            << delim;  
        return 0;  
    }  
cout << "is prime";
```

\* при определении простоты числа достаточно перебирать не  $\text{numb}/2$  делителей, а  $\text{sqrt}(\text{numb})$

Порядковый номер крайней правой цифры 5 в заданном натуральном числе, если такая цифра есть

```
unsigned number, k = 0;
cin >> number;
bool flag = false;
if (number){//если number не 0
    do {
        k++;
        if (number % 10 == 5){
            flag = true;
            break;
        }
    }
    while (number /= 10);
}
if (flag) ... //вывод k
else ... //такой цифры нет
```

Среди  $n$  членов ряда вычислить сумму тех элементов, для которых разность между соседними из них превышает заданное значение  $\text{eps}$  (если условие выполняется для всех  $n$  членов ряда – вычислить сумму всех  $n$  членов):  $1 + 1/2 + 1/3 + \dots + 1/(n-1) + 1/n$

```
double eps;  
int n;  
cin >> eps >> n;  
double sum = 0, t = 2;  
for (unsigned i = 1; i <= n; i++){  
    if (t - 1. / i < eps)  
        break;  
    t = 1. / i;  
    sum += t;  
    cout << t;  
}  
cout << sum;
```

## Произведение ненулевых чисел, введенных пользователем

```
char ans = 'n';
float number, prod = 1;
while (ans != 'y'){
    cout << "input some number except 0\n";
    cin >> number;
    if (!number) {
        cout << "oops, it cant be 0\n";
        continue;
    }
    prod *= number;
    cout << "input \'y\' to end entering\n";
    cin >> ans;
}
cout << "the product is " << prod;
```

# Практическое применения циклов while

## 1) Загрузка данных с ограничениями

Например, Microsoft Excel имеет ограничение на количество строк, хранящихся на одном листе, — примерно 1 млн записей. При заливке данных в Excel разработчики часто используют цикл while: пока количество записей на листе меньше 1 млн, данные загружаются, а в случае превышения лимита загрузка данных будет продолжаться на следующем листе.

Ограничение может возникнуть в количестве записей, файлов, секунд, объеме информации и т.д.

## 2) Технологические процессы с ограничениями

Процесс изготовления какого-либо продукта имеет достаточно много ограничений, а на больших производствах их количество и вовсе может достигать нескольких сотен. Очевидно, что в таких условиях польза от вычисления искомых значений, а также контроль изготовления продукции с помощью компьютерных программ неоценимы.

## 3) Программы мониторинга

Программы мониторинга служат примером программ с использованием бесконечных циклов. Они в случайный момент времени «оживают», проверяют, что вокруг них происходит, выполняют свою работу и снова «засыпают». Такие системы работают все время: если для какой-то из них есть работа, она ее

# Метка и оператор goto

Осуществляет безусловный переход к строчке кода, на которой стоит метка. Обычно используется в сочетании с условными операторами.

```
имя_метки1:  
...  
if (условие1)  
    goto имя_метки1;  
...  
if(условие2)  
    goto имя_метки2;  
...  
имя_метки2:  
...
```

\* использование goto считается плохим стилем разработки и не рекомендуется применять при разработке прикладных программ



# Пример вычисления факториала с помощью метки и goto

```
unsigned int n ;  
cin >> n;  
unsigned long fact = 1;  
int i = 1;  
start:  
    fact *= i;  
    i++;  
if (i <= n)  
    goto start;
```

goto: порядковый номер крайней правой цифры  
"5" в заданном натуральном числе

```
unsigned number, k = 0;
cin >> number;
bool flag = false;
if (number){
    start:
    k++;
    if (number % 10 == 5){
        flag = true;
        goto end;
    }
    if (number /= 10)
        goto start;
}
end:
if (flag) ... //вывод k
else ... //такой цифры нет
```

# Вложенные циклы

# Определения

Внешний цикл – цикл, в теле которого содержится один или несколько других циклов.

Внутренний цикл – цикл, который является частью тела другого цикла.

По типу вложенности выделяют *иерархическое* и *последовательное* расположение внутренних циклов.

Переход к следующей итерации внешнего цикла происходит тогда, когда полностью завершится выполнение его тела, а именно:

- выполнятся все действия тела внешнего цикла, стоящие до внутреннего цикла,
- выполнятся все итерации внутреннего цикла,
- выполнятся все действия тела внешнего цикла, стоящие после внутреннего цикла.

# Последовательное расположение циклов

```
for (i = 1; i <= n; i++){ //внешний цикл
```

```
    ...  
    // 1-й внутренний цикл
```

```
    for (j = 1; j <= m; j++){
```

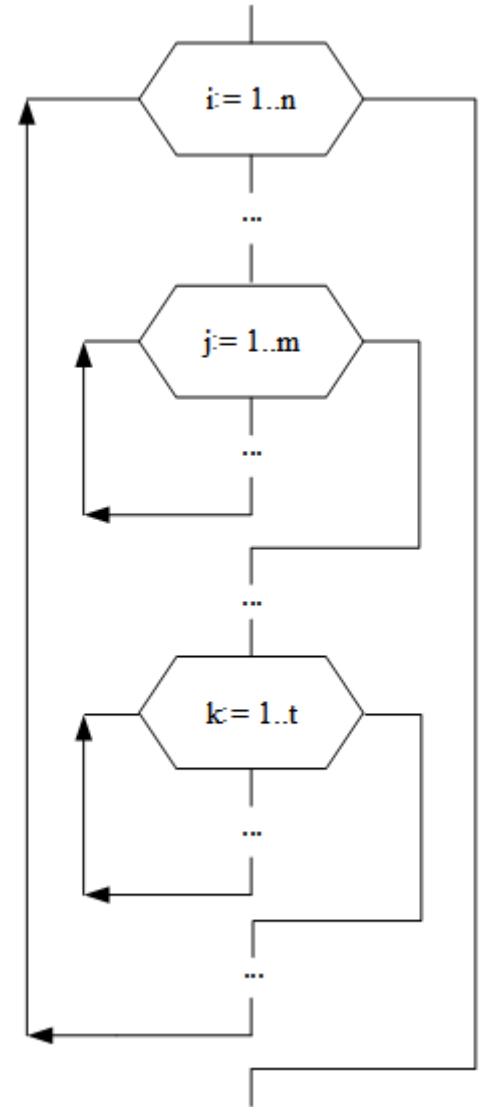
```
        ...  
    } // конец 1-го внутреннего цикла
```

```
    ...  
    // 2-й внутренний цикл
```

```
    for (k = 1; k <= t; k++){
```

```
        ...  
    } // конец 2-го внутреннего цикла
```

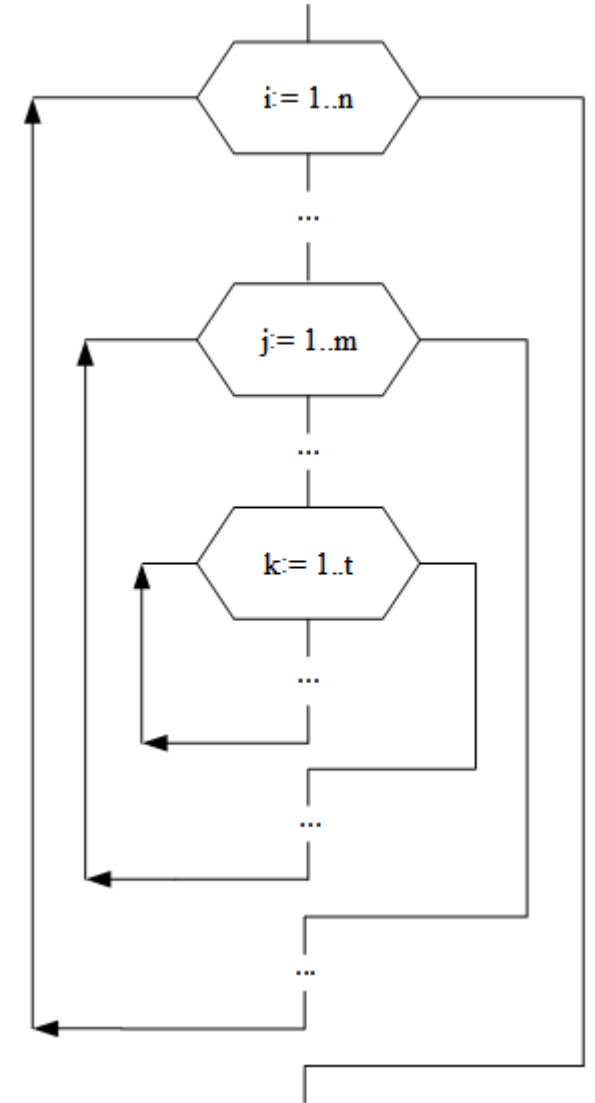
```
    ...  
} //конец внешнего цикла
```



# Иерархическое расположение циклов

```
for (i = 1; i <= n; i++){ //внешний цикл
    ...
    // 1-й внутренний цикл
    for (j = 1; j <= m; j++){
        ...
        // 2-й внутренний цикл
        for (k = 1; k <= t; k++){
            ...
        } // конец 2-го внутреннего цикла
    } // конец 1-го внутреннего цикла
} //конец внешнего цикла
```

\* тело цикла со счетчиком k будет выполнено  $t*m*n$  раз



# Последовательность изменения значений счетчиков ЦИКЛОВ

$i = 1, j = 1, k = 1$

$i = 1, j = 1, k = 2$

...

$i = 1, j = 1, k = t$

$i = 1, j = 2, k = 1$

$i = 1, j = 2, k = 2$

...

$i = 1, j = 2, k = t$

...

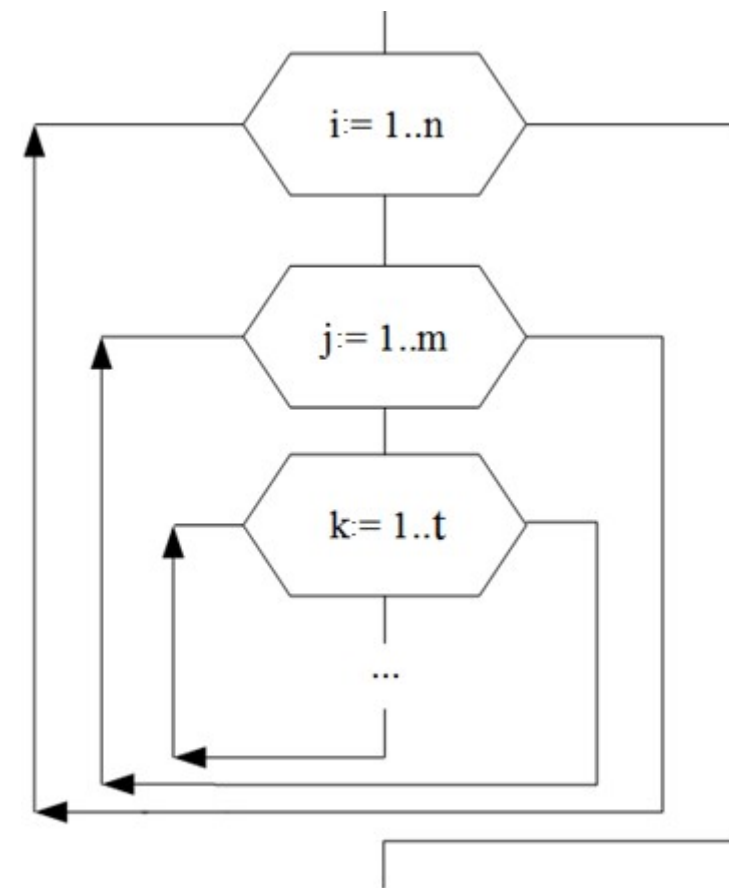
$i = 1, j = m, k = t$

$i = 2, j = 1, k = 1$

...

$i = n, j = m, k = t$

\*/



# Допустимые и недопустимые операции передачи управления при работе с вложенными циклами

При решении некоторых задач может возникнуть необходимость передачи управления выполнением программы посредством использования оператора `goto`, хотя это и не отвечает принципам структурного программирования

- *Допустимо* передавать управление из внутреннего цикла во внешний (частный случай `break` и `continue`).
- *Опасно* передавать управление из внешнего цикла во внутренний, минуя вход во внутренний цикл, т.к. в этом случае отсутствует инициализация переменных цикла, что может привести к его бесконечности.



Пример, когда порядок вложения циклов **не имеет значения**

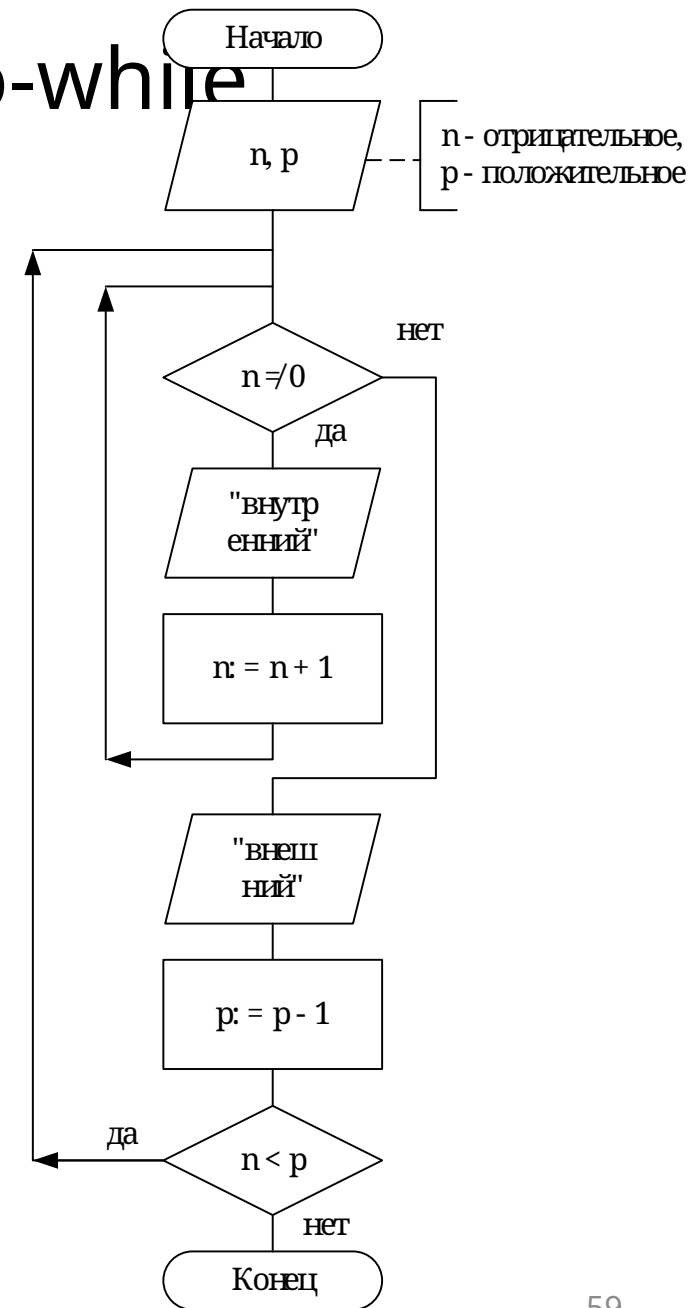
```
int i, j, s = 0, n = 3, m = 5;
for (i = 1; i <= n; i++){
    s = 0;
    for (j = 1; j <= m; j++)
        s += i + j; // сумма коммутативна
    cout << s << " ";
}
```

Пример, когда порядок вложения циклов **имеет значение**

```
int i, j, s, n = 3, m = 5;
for (i = 1; i <= n; i++){
    s = 0;
    for (j = 1; j <= m; j++)
        s += i + j * j;
    cout << s << " ";
}
```

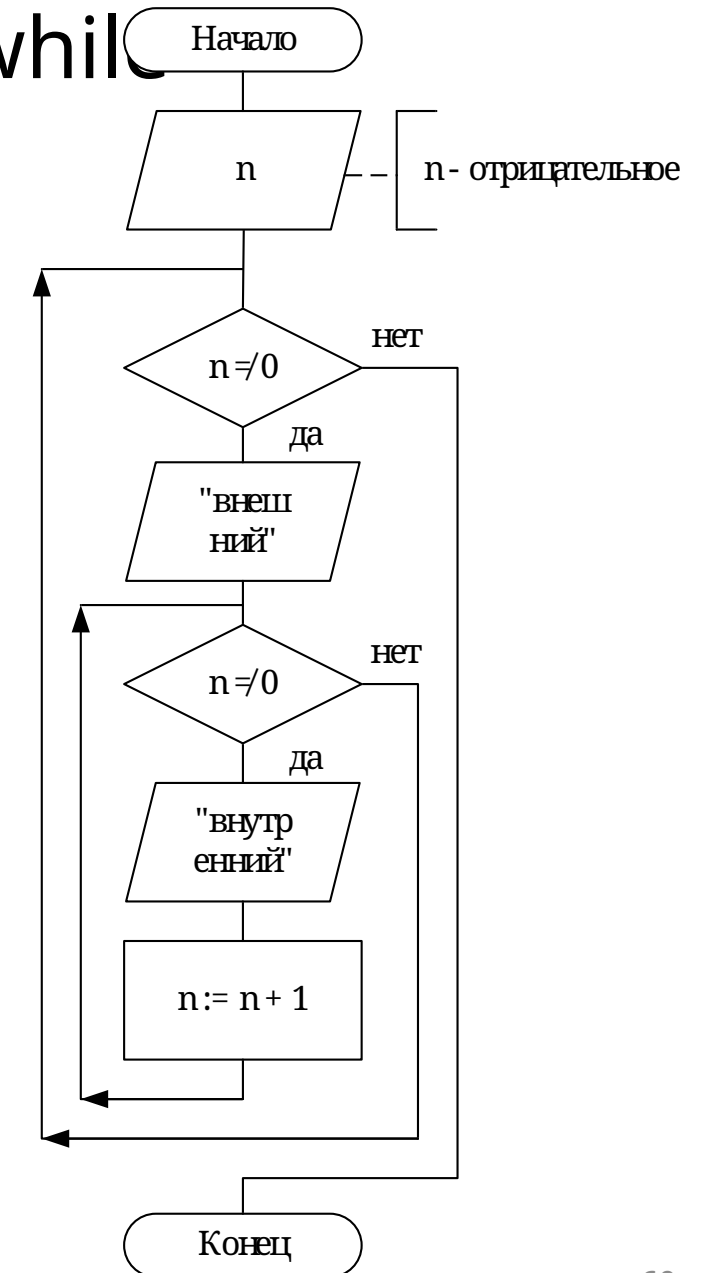
# Пример вложенности while в цикл do-while

```
int n, p;  
cin >> n >> p;  
do{  
    while (n){  
        cout << "inside";  
        n++;  
    }  
    cout << "outside";  
    p--;  
}  
while (n < p);
```



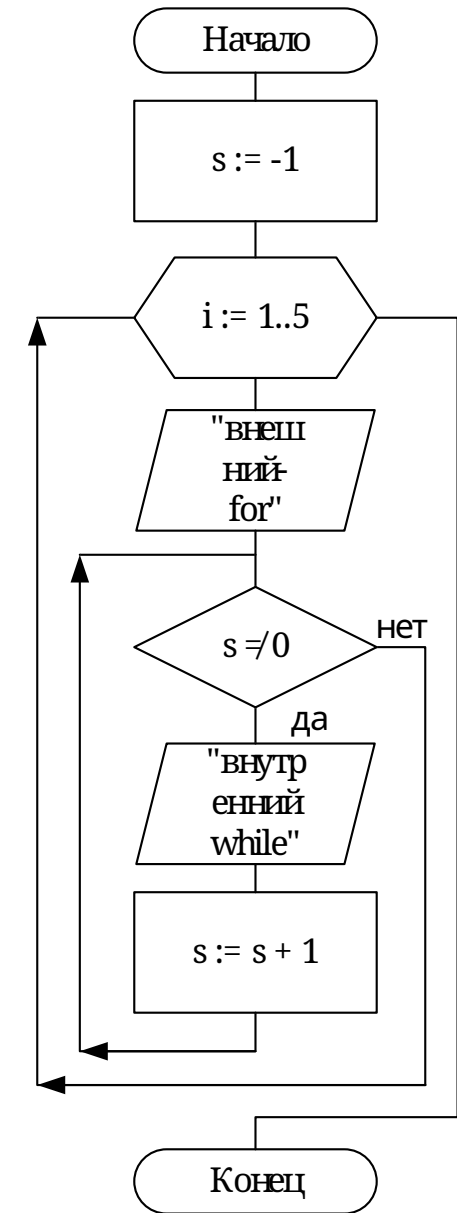
# Пример вложенности while в цикл while

```
int n;  
cin >> n; //negative  
while (n){  
    cout << "внешний";  
    while (n){  
        cout << "внутренний";  
        n++;  
    }  
}
```



# Пример вложенности while в цикл for

```
int i, s = -3;  
for (i = 1; i <= 5; i++){  
    cout << "внешний-for";  
    while (s){  
        cout << "внутренний-while";  
        s++;  
    }  
}
```



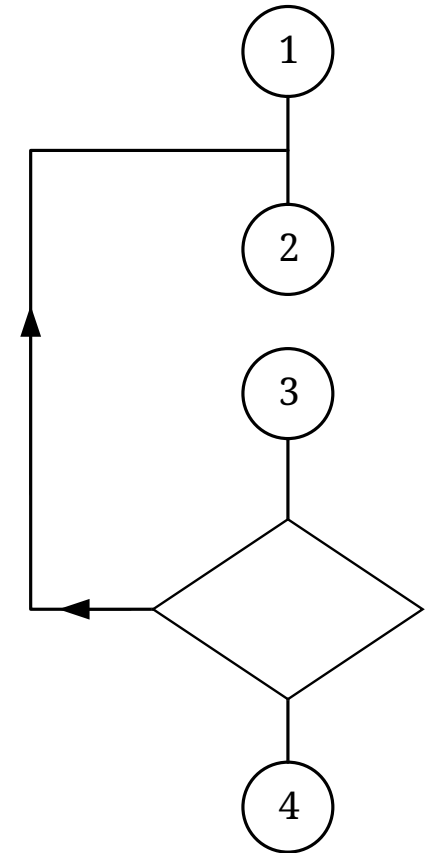
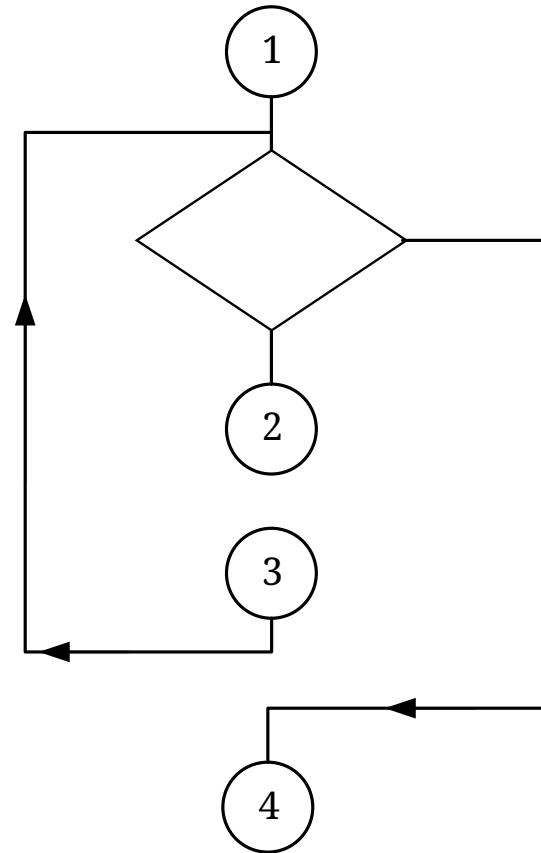
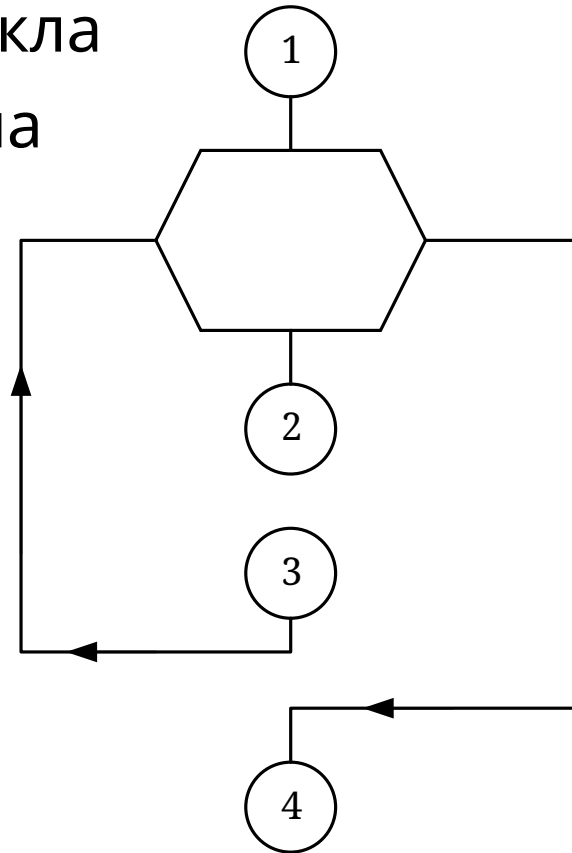
# Как рисовать схемы со вложенными циклами

1 – вход в цикл

2 – начало тела цикла

3 – конец тела цикла

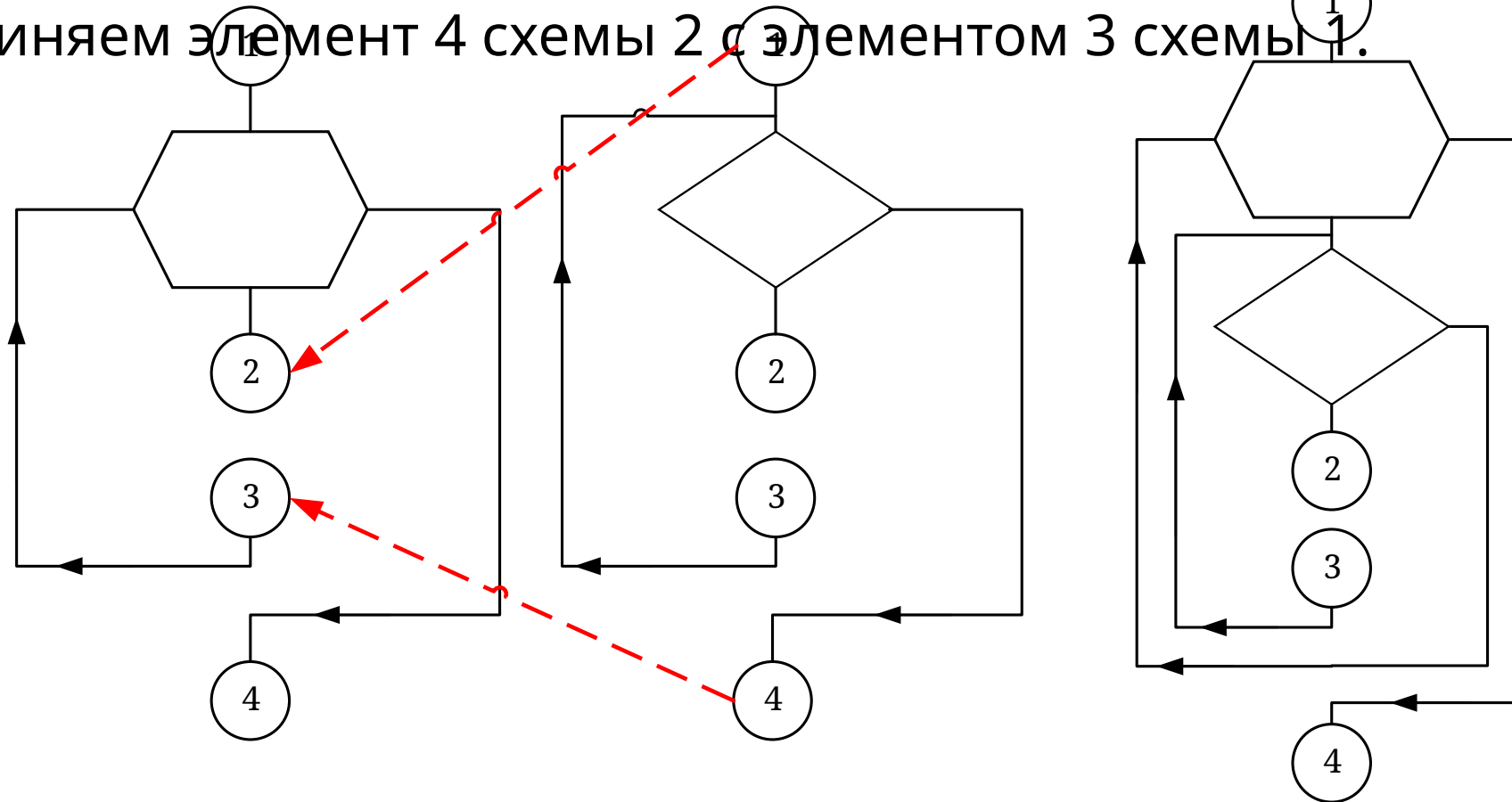
4 – выход из цикла



# Как рисовать схемы со вложенными циклами

Вкладываем цикл с предусловием (схема 2) в цикл со счетчиком (схема 1):

- соединяем элемент 1 схемы 2 с элементом 2 схемы 1;
- соединяем элемент 4 схемы 2 с элементом 3 схемы 1.



Задача: вывести значения в табличном виде. Решение

1

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

```
int main(){
int k = 1;
for(int i = 1; i <= 3; i++){
    for(int j = 1; j <= 5; j++){
        cout << k++ << " ";
    }
    cout << endl;
}
```



Задача: вывести значения в табличном виде.  
Решение 2 – **без дополнительной переменной**

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

```
int main(){
for(int i = 1; i <= 3; i++){
    for(int j = 1; j <= 5; j++)
        cout << (i - 1) * 5 + j << " ";
    cout << endl;
}
}
```

Вычислить сумму делителей всех чисел промежутка от a до b

```
int a, b, sum = 0;
cin >> a >> b;
    for (int i = a; i <= b; sum += i, i++)
        for (int j = 1; j <= i / 2; j++)
            if (i % j == 0)    // if (!(i % j))
                sum += j;
cout << sum;
```

Вычислить сумму делителей для каждого из чисел промежутка от a до b

```
int a, b, sum = 0;
cin >> a >> b;
for (int i = a; i <= b; i++){
    sum = i;
    for (int j = 1; j <= i / 2; j++)
        if (i % j == 0) // if(!(i % j))
            sum += j;
    cout << "for number: " << i
        << "sum is: " << sum << endl;
}
```

Вывести простые числа, не превышающие заданного n

```
int n;
cin >> n;
for (int i = 1; i <= n; i++){
    bool simple = true;
    for (int j = 2; j <= sqrt(i); j++)
        //аналогично проверке if (!(i % j))
        if (i % j == 0){
            simple = false;
            break;
        }
    if (simple)
        cout << i << endl;
}
```

## Отображение в консоли простых чисел в графическом виде

```
const unsigned char white = 219; // for simple numbers
const unsigned char gray = 176; // for other numbers
unsigned char ch;
for(int count = 0; count < 100; count++){ //check 100 numb
    ch = white; // suppose current number is simple
    for(int j = 2; j <= sqrt(count); j++)
        if(count % j == 0){
            ch = gray; // is not simple
            break;
        }
    cout << ch; //print symbol
}
```

\* кодам 219 и 176 в таблице ASCII соответствуют символы, представляющие белый и серый прямоугольник

# Пример. Защита лабораторной работы

```
unsigned short i(1), j , labsCount, wrongAnswer;
bool badLab;
cin >> labsCount;
while (i <= labsCount){
    wrongAnswer = 0;
    badLab = false;
    for (j = 1; j <= questionsCount; j++){
        if (answerTheQuestion() != true)
            wrongAnswer++;
        if (wrongAnswer > 3){
            badLab = true;
            break;
        }
    }
    if (!badLab) i++;
}
```

# Организация вложенных циклов при помощи меток и безусловных переходов

```
int i, j, n, m;
for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
        cout << i + j;
// ---
i = 0;
outside:
j = 0;
inside:
cout << i + j;
j++;
if (j < m) goto inside;
i++;
if (i < n) goto outside;
```

## Задача Шахматная доска

- Нарисуйте шахматную доску размера  $n \times n$  (натуральный параметр  $n$  вводится пользователем), состоящую из нулей и единиц. Верхний левый символ должен быть единицей.



# Задача Шахматная доска

```
int n;  
cin >> n;  
int c = n / 2;  
for (int i = 0; i < n; i++){  
    bool odd = i % 2 == 0;  
    for(int j = 0; j < c; j++){  
        cout << (odd ? "10" : "01");  
    }  
    if (n % 2 == 1)  
        cout << (!odd ? '0' : '1');  
    cout << endl;  
}
```

# Задача Пирамида

Выведите пирамиду заданной высоты, состоящую из символов \*. Значение высоты вводится пользователем. Важно соблюдать увеличение количества звезд в зависимости от высоты, а также правильно центрировать звездочки. Важно: требуется именно центрированная пирамида-елочка в привычном понимании. Обратите внимание, что справа от пирамиды не должно быть пробелов.

# Задача Пирамида

```
int h;  
cin >> h;  
int base = 2 * h + 1;  
for(int i = 0; i < h; i++){  
    for(int j = 0; j < h - i - 1; j++)  
        cout << ' '  
    for(int j = 0; j < 2 * i + 1; j++)  
        cout << '*'  
    cout << endl;  
}
```

# Задача Цикл проверки логина и пароля

- Создайте цикл проверки введенных пользователем логина и пароля на соответствие следующим установленным правилам:
- Логин должен содержать ровно две заглавные буквы и состоять минимум из 6 символов.
- Пароль должен содержать ровно одну заглавную букву, хотя бы одну цифру и состоять минимум из 8 символов.
- Если логин / пароль соответствует требованиям, должно выводиться сообщение: "Логин соответствует условиям регистрации" / "Пароль соответствует условиям регистрации". В противном случае выводится сообщение "Логин не соответствует условиям регистрации" или "Пароль не соответствует условиям регистрации" соответственно для логина и пароля.
- Если и логин, и пароль правильные, то выводится фраза "Вы успешно зарегистрированы! Добро пожаловать в MangoFM". Цикл завершается. В ином случае на экране пишется "К сожалению, условия не выполнены. попробуйте еще раз" и у пользователя

# Примеры решения задач

# Написать программу для решения следующих задач

1. Используя цикл со счетчиком решить задачу

Вычислить значение выражения  $\cos 1 + \cos 1.1 + \dots + \cos n$  ( $n$  – целое)

2. Используя цикл с предусловием решить задачу

Определить количество цифр в целом числе

3. Используя цикл с постусловием решить вторую задачу

4. Используя вложенные циклы решить задачу

Найти 100 первых простых чисел

5. Используя вложенные циклы решить задачу

Дано действительное число  $x$ , вычислить выражение:

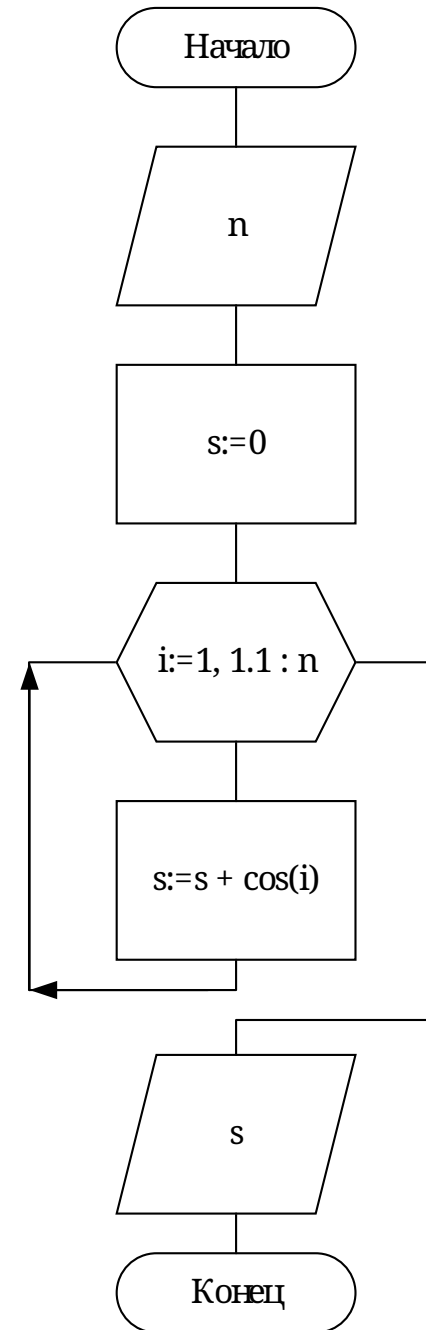
$$\prod_i \sum_j f(x) \quad f(x) = \frac{x}{i \cdot j} \quad i = 1, 2, \dots, n \quad j = 1, 2, \dots, n - 1$$

# Решение задачи 1

```
int main(){
long double s = 0;
unsigned n;
cout << "input value > 1\n";
cin >> n;
for (float i = 1; i < n + 0.1; i += 0.1)
    s += cos(i);
cout << "sum is " << s;
}
```

Входные данные:  $n$  – целое число.

Выходные данные:  $s$  – действительное число.

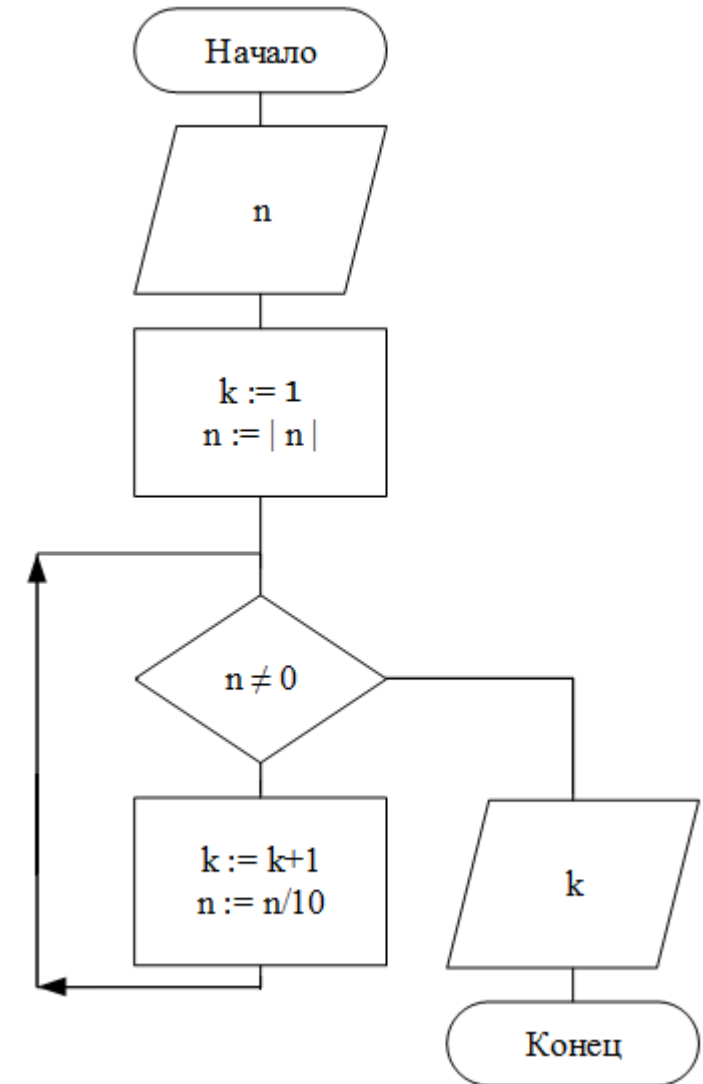


## Решение задачи 2

```
int main(){
long n;
short k = 1;
cout << "input integer value\n";
cin >> n;
n = abs(n);
while (n /= 10);
    k++;
cout << "result is " << k;
}
```

Входные данные:  $n$  – целое число.

Выходные данные:  $k$  – целое число.



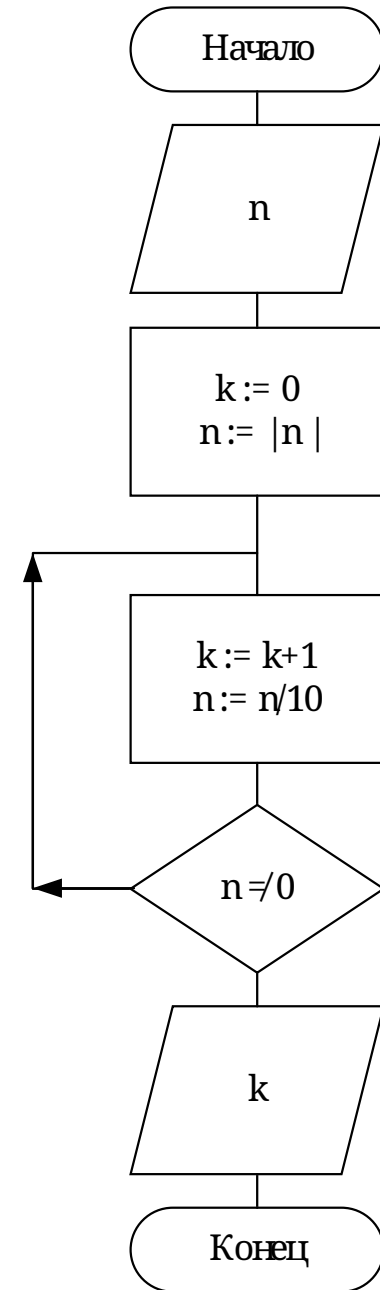


# Решение задачи 3

```
int main(){
long n;
short k = 0;
cout << "input integer value\n";
cin >> n;
n = abs(n);
do
    k++;
while (n /= 10);
cout << "result is " << k;
}
```

Входные данные:  $n$  – целое число.

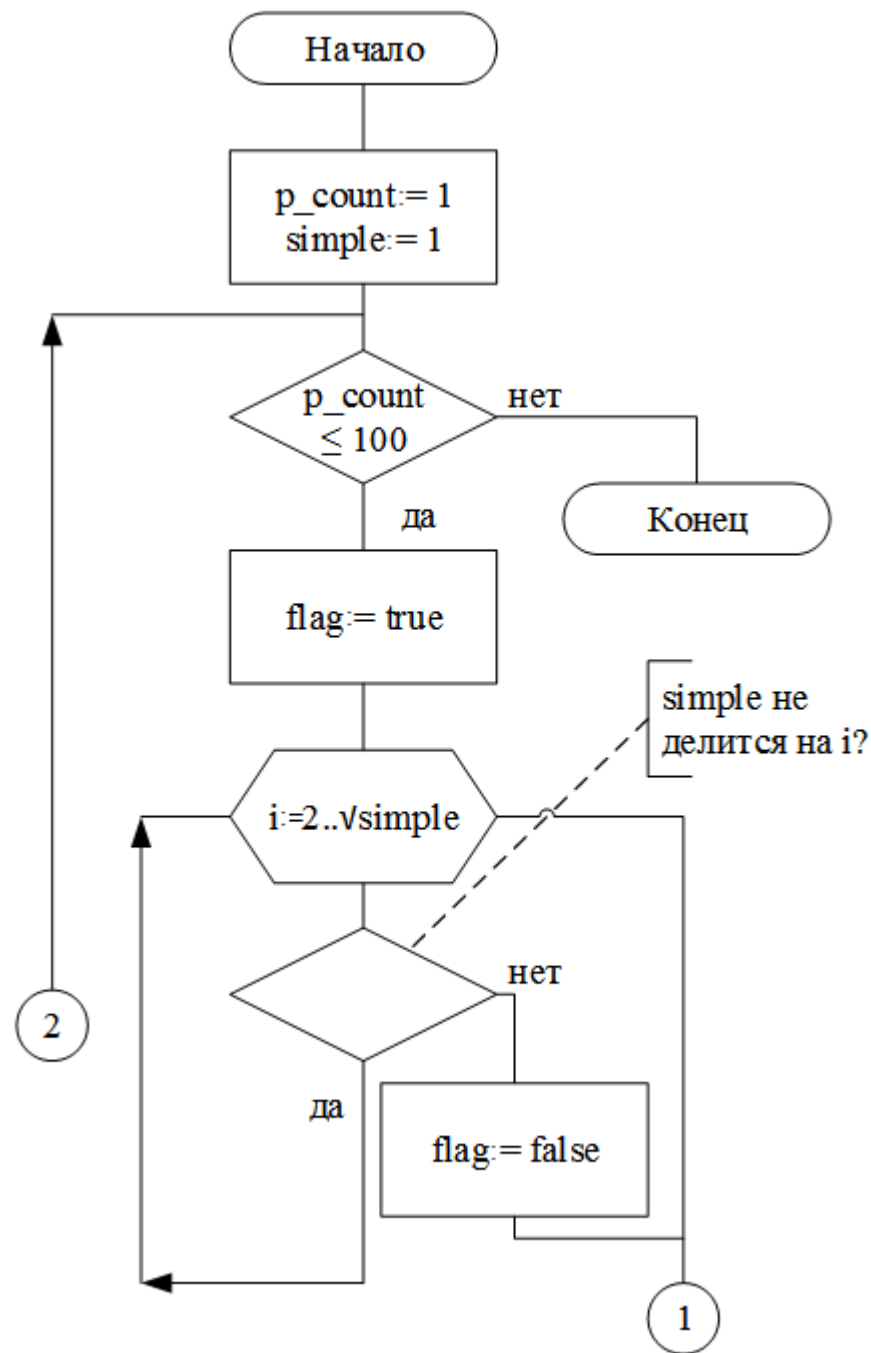
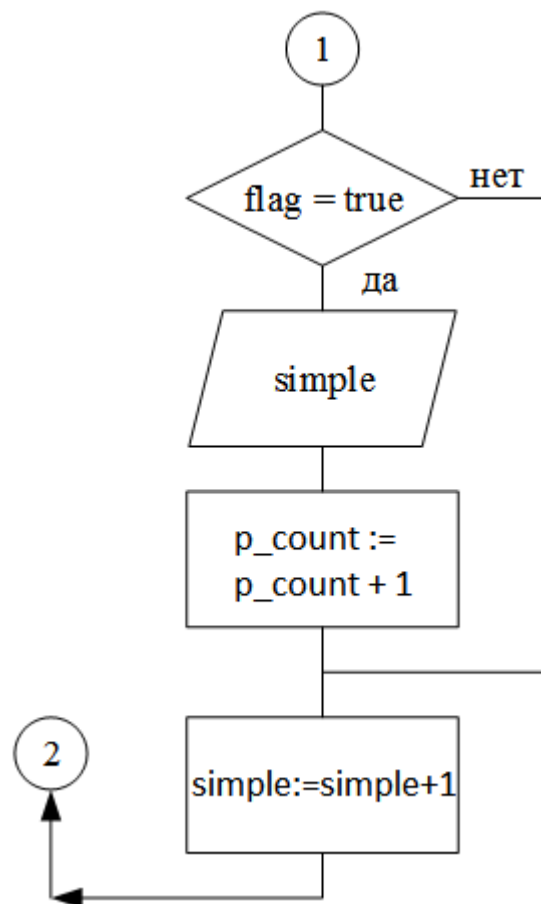
Выходные данные:  $k$  – целое число.



# Решение задачи 4

Входные данные: отсутствуют.

Выходные данные: *simple* – целые числа.



# Решение задачи 4

```
typedef unsigned short us;
```

```
int main(){  
for (us sCount = 1, mbSimple = 1; sCount <= 100; mbSimple++){  
//пока количество найденных чисел не превосходит 100  
    bool hasNoDiv = true;  
    double sq = sqrt(MbSimple);  
    for (int i = 2; hasNoDiv && i <= sq; i++)  
        if (mbSimple % i == 0)  
            hasNoDiv = false;  
    if (hasNoDiv){ //если делителей нет, то simple - простое  
        cout << sCount << ": " << mbSimple << endl;  
        sCount++;  
    }  
}  
}
```

## Решение задачи 5

```
int main(){
double x, sum(0), prod(1);
unsigned short i, j, n;
cin >> x >> n;
for (i = 1; i <= n; i++){
    sum = 0;
    for (j = 1; j < n; j++){
        sum += x / (i * j);
    }
    prod *= sum;
}
cout << prod;
}
```

Входные данные:  $x$  – действительное число,  $n$  – целое число.  
Выходные данные:  $prod$  – действительное число.

