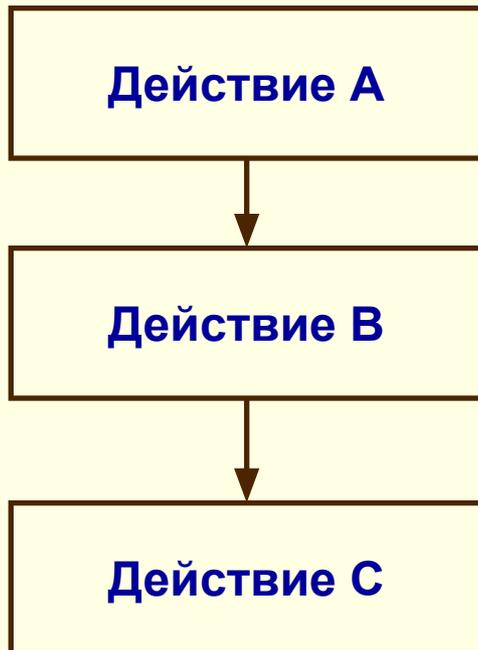


# АЛГОРИТМЫ

## 3 базовые управляющие алгоритмические структуры

### Последовательность

Непосредственное выполнение одного действия за другим



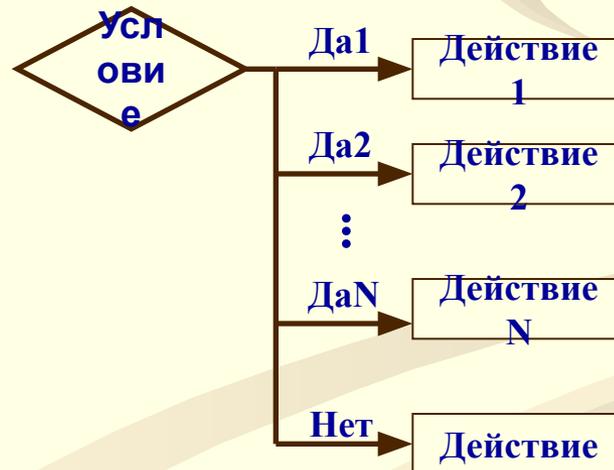
### Выбор решения

Проверка выполнения условия и выбор одного из альтернативных действий

#### 1. Бинарный выбор - ветвление

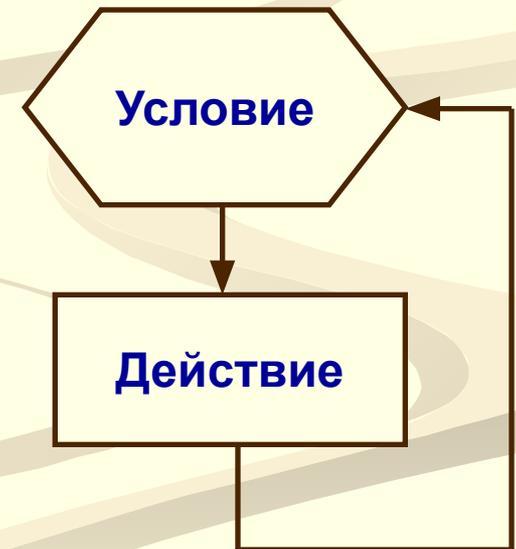


#### 2. Множественный выбор



### Цикл

Организация повторяющихся действий в соответствии с заданным условием



C / C++

## Операторы цикла

**Цикл for**  
(с параметром)



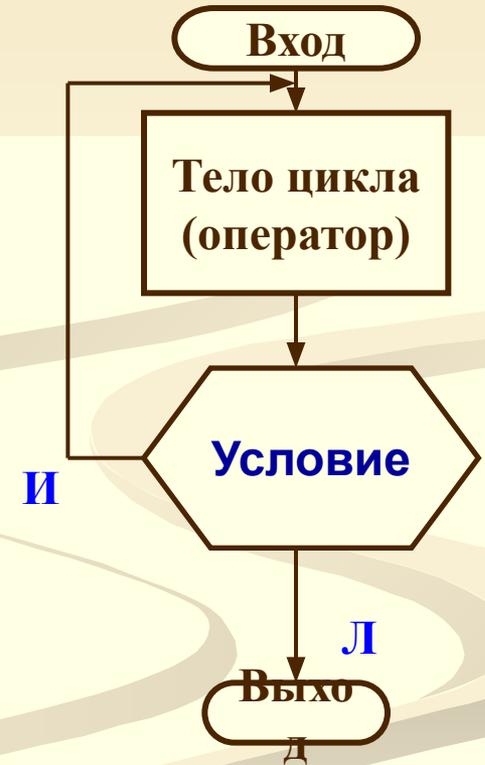
Тело цикла выполняется  $n-f/s$  раз

**Цикл while**  
(с предусловием)



Тело цикла может не выполниться ни разу

**Цикл: do-while – C**  
(с постусловием)



Тело цикла обязательно выполниться хотя бы один раз

## Цикл с параметром

Эта конструкция цикла используется в тех случаях, когда заранее известно точное количество повторов (итераций) цикла, требующееся для выполнения действия.

В псевдокоде для описания цикла с параметром используется следующая конструкция:

**ДЛЯ** `loop_index = initial_value` **ДО** `final_value`

**Тело цикла**

**ДЛЯ ВСЁ**

- `loop_index` – это переменная цикла – счётчик номера итерации (повтора) цикла,
- `initial_value` – начальное значение переменной цикла, номер первой итерации,
- `final_value` – конечное значение переменной цикла, номер последней итерации.

Количество итераций цикла равно разности `final_value` и `initial_value`.

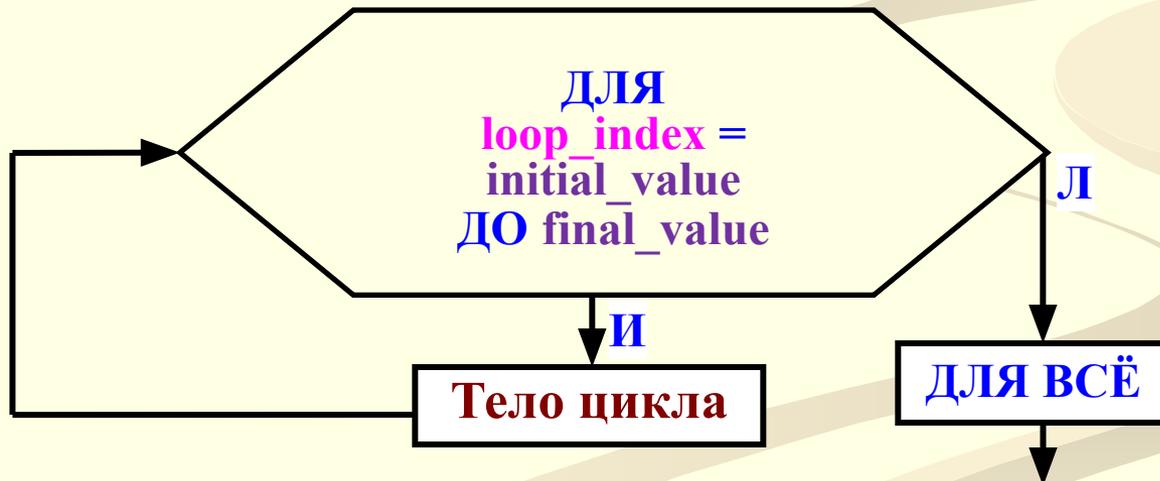
Итерации цикла повторяются, пока параметр цикла `loop_index` находится в диапазоне от `initial_value` до `final_value`, можно считать, что при этом условие продолжения цикла – **Истинно (И)**, когда параметр цикла за пределами диапазона, условие – **Ложно (Л)**.

## Цикл с параметром

Работа цикла **ДЛЯ**:

- Переменная **loop\_index** устанавливается в заданное начальное значение **initial\_value**,
- При каждом прохождении (итерации) цикла переменная цикла автоматически увеличивается (уменьшается) на 1,
- В начале новой итерации переменная **loop\_index** проверяется на соответствие верхнему (нижнему) пределу (**final\_value**),
- При достижении переменной **loop\_index** заданного верхнего (нижнего) предела (**final\_value**) цикл завершается и алгоритм переходит к выполнению следующего за **ДЛЯ ВСЁ** действия.

В виде блок-схемы эта конструкция выглядит так:



В ЯП С эта конструкция реализуется с помощью оператора **for**

## Оператор цикла **for**

С

```
for (<инициализация>;  
    <условие>; <приращение>)  
    <оператор>;
```

где **for** – ключевое слово для,

- **инициализация** – присваивание начального значения параметру(-ам) цикла,
- **условие** – выражение, цикл выполняется пока оно истинно,
- **приращение** – изменение параметра цикла при каждой итерации,
- **оператор** – одиночный или составной оператор.

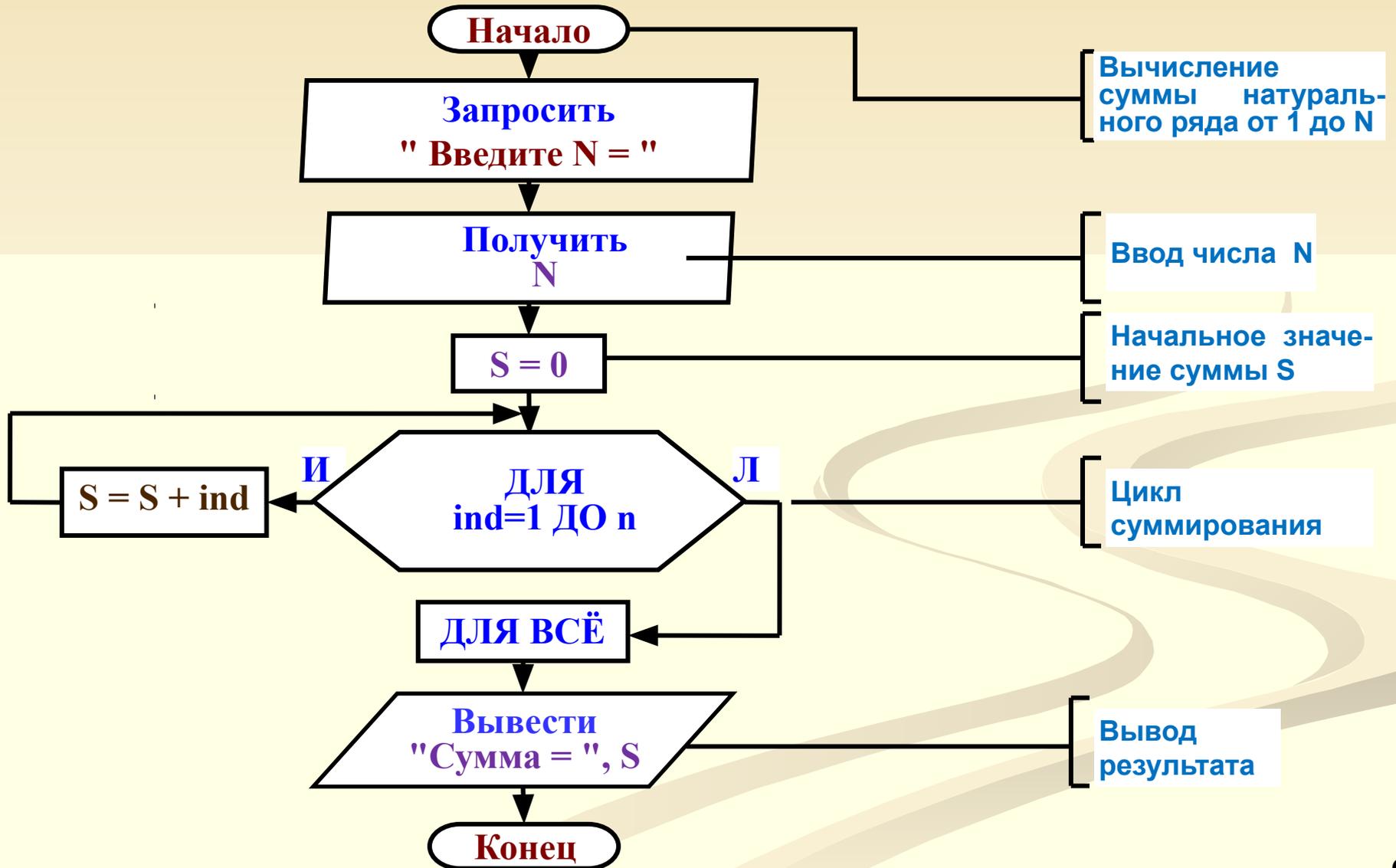
Принудительное завершение всего цикла или текущей итерации – операторы: **break**, **continue**, **return**, **goto**.

Пример: **for (i = 1; i <= n; i++)**

# Элементы ЯПВУ

## Примеры цикла for

Блок-схема алгоритма: Вычислить сумму целых положительных чисел от 1 до N



# Элементы ЯПВУ

C / C++

## Примеры цикла for

Вычислить сумму целых положительных чисел от 1 до N

```
#include <stdio.h>
int main ()
{
int i, n, s=0;
    printf ("Введите n = ");
    scanf ("%d",&n); /* Ввод числа */
/* Цикл подсчета суммы */
    for (i = 1; i <= n; i++)
        s = s + i;
/* Вывод результата */
    printf("Сумма = %d\n", s);
    return 0;
}
```

## Задание на дом на цикл for:

1. Найти все делители целого положительного числа.
2. Напечатать таблицу значений функции  $Y=X^2+1$  во введенном диапазоне.
3. Ввести 5 дробных чисел и после ввода каждого числа вывести среднее арифметическое введенной части последовательности.

Нарисовать блок-схему алгоритма и написать программы на C

## Цикл с предусловием

Эта конструкция используется для выполнения цикла, условие завершения которого описывается в заголовке (в начале) цикла.

В псевдокоде для описания цикла с предусловием используется следующая конструкция:

**ПОКА** условие **Р** истинно  
**Тело цикла**  
**ПОКА ВСЁ**

- условие **Р** – логическое условие продолжения цикла (терминальное условие).

Конструкция **ПОКА** – это цикл с предусловием, т.е. условие проверяется *до* выполнения действий тела цикла.

### Замечания:

а) поскольку условие проверяется в начале цикла, то чтобы задать корректное условие необходимо выполнить определённую логическую обработку данных до проверки условия,

б) стандартный способ прервать цикл **ПОКА** – сделать условие ложным; это означает, что в теле цикла должны выполняться какие-то операции изменяющие условие цикла, иначе цикл может стать бесконечным.

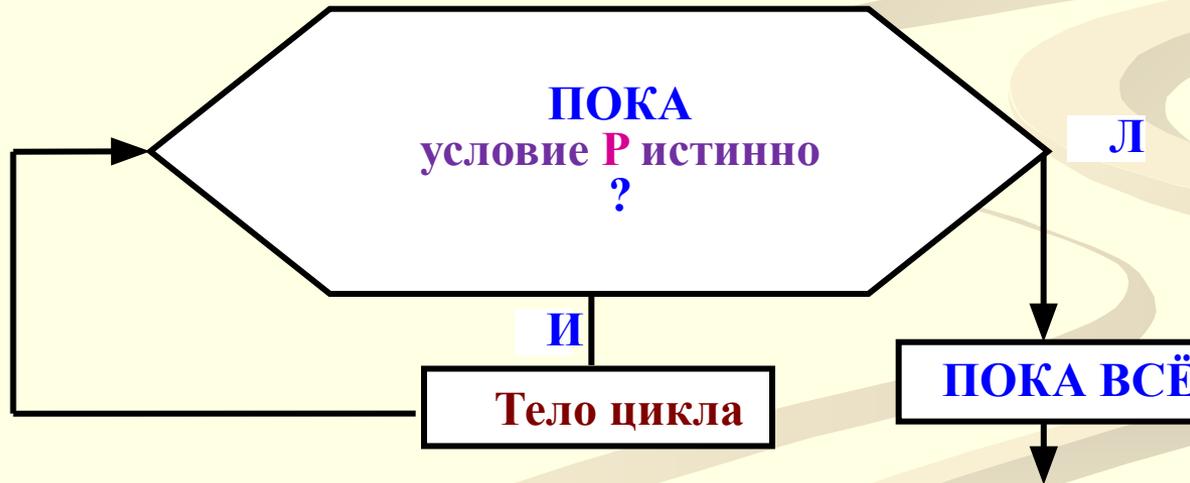
**Принудительное завершение всего цикла или текущей итерации – операторы безусловного перехода: break, continue, return, goto.**

## Цикл с предусловием

Работа цикла **ПОКА**:

1. Проверяется логическое условие **Р**,
2. Если условие **Р** истинно, один раз выполняются действия (операции) заданные в теле цикла, затем выполняется переход к началу цикла и повторно проверяется условие,
3. Если условие **Р** по-прежнему истинно, снова повторяется тело цикла,
4. Если условие **Р** ложно, управление передаётся к действию, следующему за ключевыми словами **ПОКА ВСЁ**, и тело цикла больше не выполняется.

В виде блок-схемы эта конструкция выглядит так:



В ЯП **С** эта конструкция реализуется с помощью оператора **while**

## Цикл с предусловием

### Использование счётчика итераций как условия цикла ПОКА

При необходимости выполнить тело цикла определённое количество раз организуется **счётчик итераций (повторов) цикла** – переменная, значение которой увеличивается на единицу после каждого выполнения тела цикла. **Это переменная используется в логическом выражении условия цикла Р.** Перед началом цикла задаётся начальное значение переменной-счётчика, а в теле цикла это значение увеличивается на единицу (до оператора ПОКА ВСЁ) при каждой итерации цикла.

**Использование в качестве условия цикла ПОКА заключительной записи (сигнальной метки) или признака конца файла.**

Если необходимо обработать в цикле неизвестное заранее количество элементов (например, список, количество записей в котором неизвестно), то счётчик итераций цикла использовать не получится.

Часто в конце данных находится **заключительная запись** или **сигнальная метка** – это особая запись или значение, размещённое в конце данных, она означает конец данных и должна содержать значение, которое чётко отличается от других обрабатываемых данных.

Возможен также случай, когда идёт обработка данных размещённых в файле на внешнем устройстве (магнитном диске, флешке и др.). При это сигнальная метка не требуется, так как **в каждом файле при его создании или изменении последним символом добавляется маркёр конца файла – EOF – End of File.** В качестве условия цикла тогда можно использовать одно из равнозначных выражений:

**ПОКА ещё данные**

**ПОКА ещё записи**

**ПОКА есть записи**

**ПОКА не EOF**

С такими условиями цикла все действия между операторами **ПОКА** и **ПОКА ВСЁ** будут повторяться, пока не будет сделана попытка прочесть данные после символа **EOF**. Когда это произойдёт, программа получит сигнал, обозначающий что данных в файле больше нет и **условие ПОКА** – ложно.

# Элементы ЯПВУ

C / C++

## Оператор цикла **while**

**while** (<условие>)

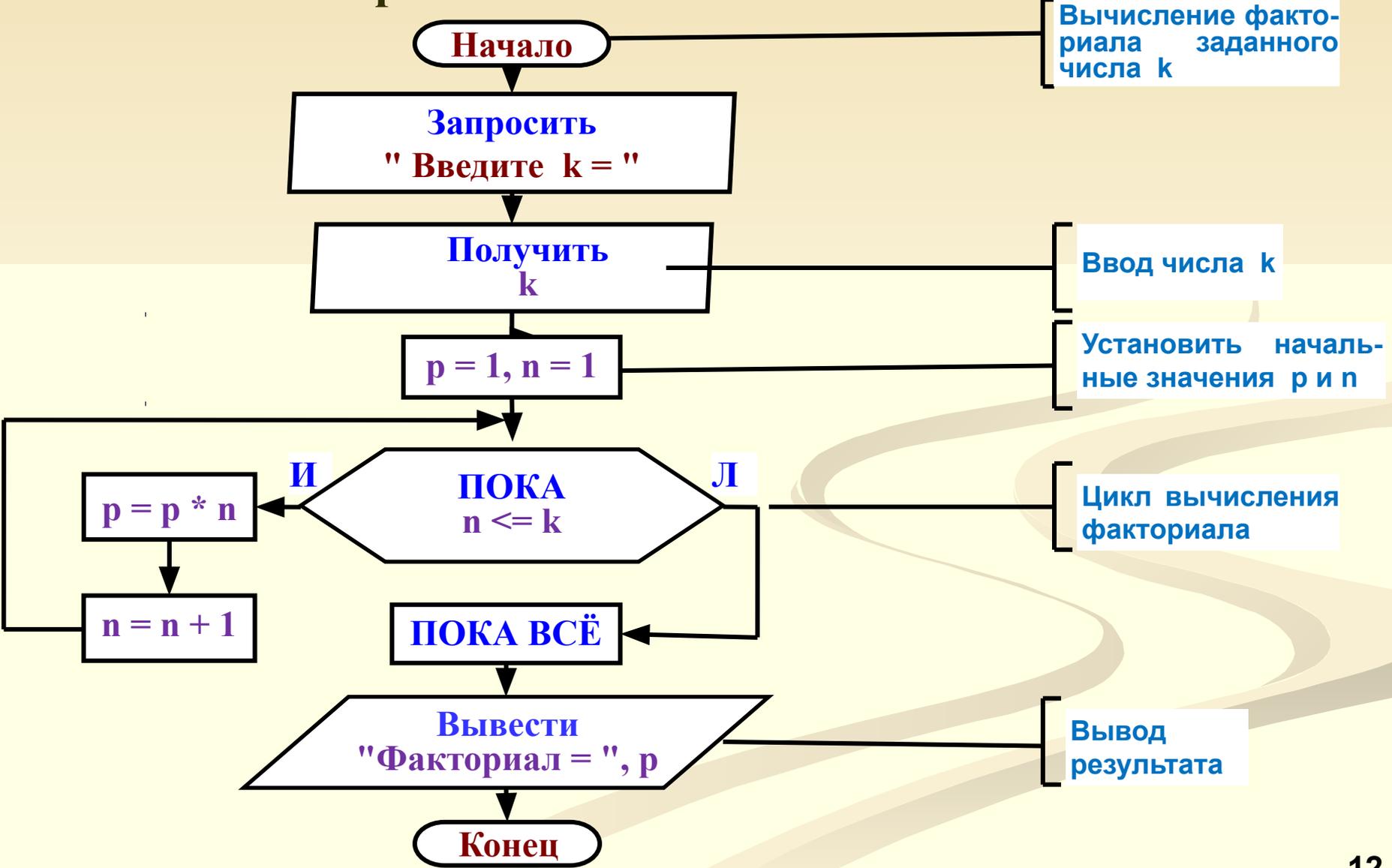
<оператор>; *где*

- **while** – ключевое слово пока,
- **условие** – выражение,
- **оператор** – одиночный или составной оператор.

# Элементы ЯПВУ

## Примеры цикла while

### Блок-схема алгоритма: Вычислить 25!



# Элементы ЯПВУ

С

## Оператор цикла **while**

Примеры

**Вычислить значение 25!**

```
#include <stdio.h>
int main ()
{
    int n, k; float p;
    /* n – переменная цикла, k – число факториала, p - значение факториала */
    printf ("Введите число k = ");
    scanf ("%d",&k); /* Ввод числа */
    p = 1; n = 1; /* Начальные значения */
    while (n <= k)
    {
        /* Вычисление факториала в цикле */
        p = p * n;
        /* Приращение переменной цикла */
        n++;
    }
    printf ("Значение факториала p = %G\n ",p);
    return 0;
}
```

## Цикл с постусловием

Эта конструкция используется для выполнения цикла, условие завершения которого проверяется в конце цикла. Таким образом, действия тела цикла выполняются до проверки условия продолжения итераций цикла.

В псевдокоде для описания цикла с предусловием используется следующая конструкция:

**ПОВТОРЯТЬ**

Тело цикла

**ПОКА** условие **P** истинно

условие **P** – логическое условие.

Конструкция **ПОВТОРЯТЬ-ПОКА** обеспечивает выполнение алгоритма запрограммированного в теле цикл *до* проверки условия, таким образом, действия тела цикла будут обязательно выполнены хотя бы один раз.

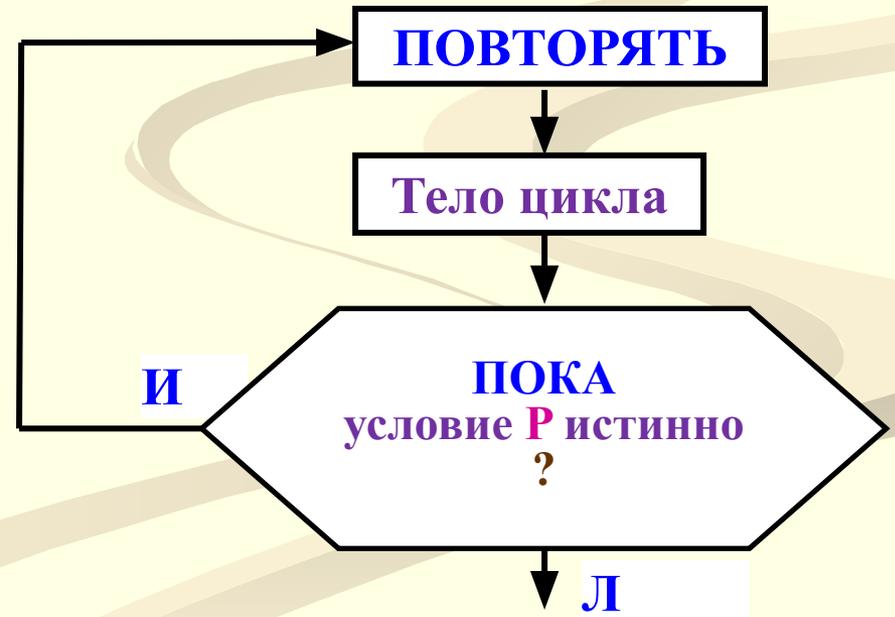
Принудительное завершении всего цикла или текущей итерации – операторы безусловного перехода: **break, continue, return, goto.**

## Цикл с постусловием

Работа цикла **ПОВТОРЯТЬ-ПОКА**:

1. Один раз выполняются действия (операции) заданные в теле цикла,
2. Проверяется логическое условие **Р**,
3. Если условие **Р** истинно [ложно], выполняется переход к началу цикла и снова выполняется тело цикла,
4. Снова проверяется условие **Р**, если оно по-прежнему истинно [ложно], то снова повторяется тело цикла,
5. Если условие **Р** становится ложно [истинно], то управление передаётся к действию, следующему за проверкой условия **ПОКА**, и тело цикла больше не выполняется.

В виде блок-схемы эта конструкция выглядит так:



в яп С – do-while

# Элементы ЯПВУ

C / C++

## Операторы

### Цикл **do-while**

**do** <оператор;>

**while** (<условие>); *где*

- **do** и **while** – ключевые слова выполнять и пока,
- **оператор** – одиночный или составной оператор,
- **условие** – выражение.

# Элементы ЯПВУ

## Пример цикла do-while

**Блок-схема:** Определить, является ли введенное с клавиатуры целое число простым

Является ли введенное с клавиатуры целое число простым

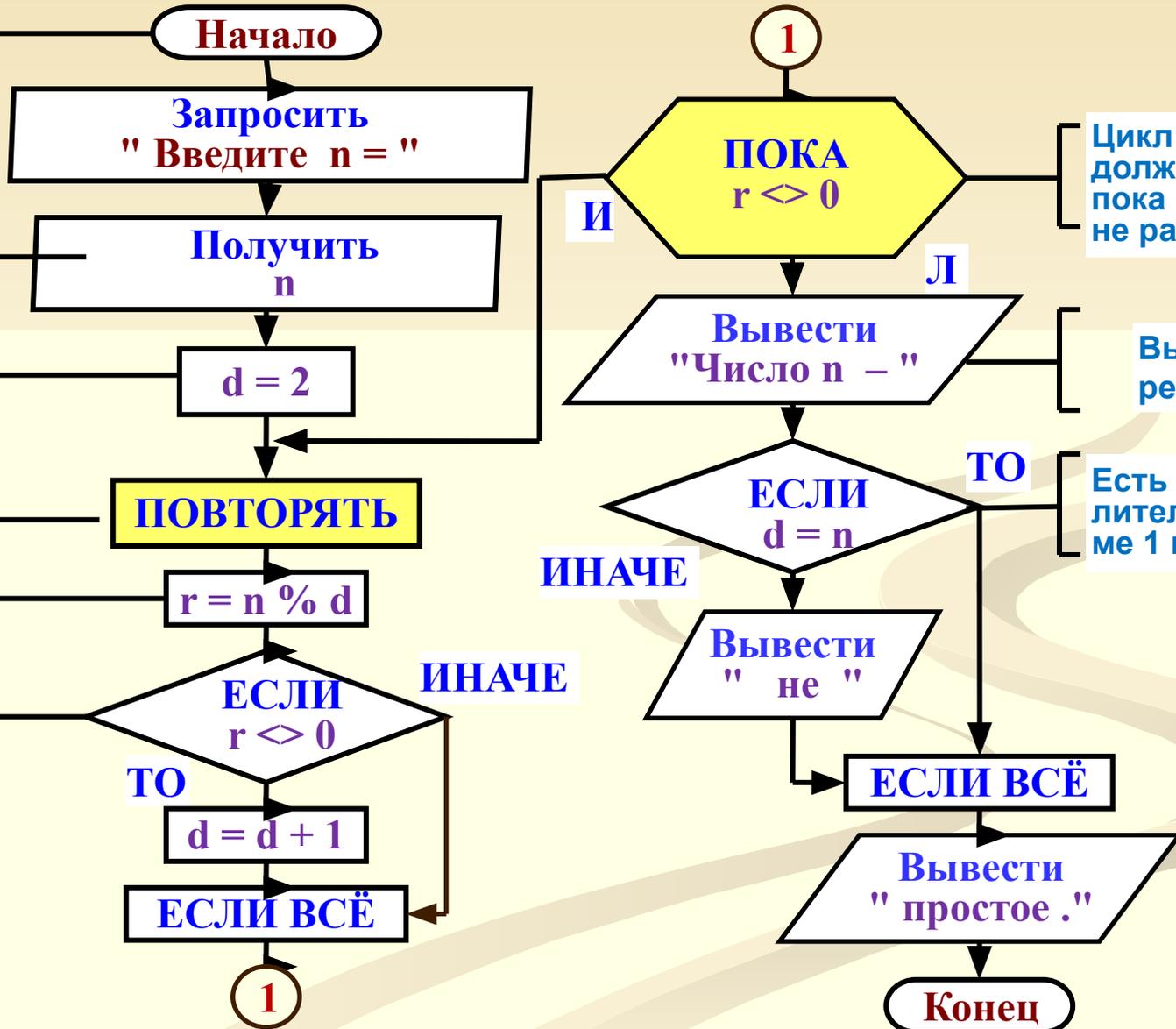
Ввод числа n

Начинаем с делителя равного 2

Цикл поиска делителей числа n

Остаток от деления на d

Если разделилось не нацело, то переход к следующему кандидату в делители



Цикл продолжается пока остаток не равен 0

Вывод результата

Есть ли делители кроме 1 и d=n?

C / C++

## Операторы Цикл do-while

Пример

Определить, является ли введенное с клавиатуры целое число простым

```
#include <stdio.h>
#include <math.h>
int main ()
{
    int n, d, r; /* n – проверяемое число, d – текущий делитель,
                r – текущий остаток от деления */
    printf ("Введите целое число n = ");
    scanf ("%d",&n);
    d = 2; /* Сначала делим на 2 */
    do
    {
        r = n % d; /*Остаток от деления на d*/
        if (r != 0) d = d + 1; /* не нацело */
    }
    while ( r != 0);
    if (d == n)
        { printf("- %d",n);
          printf("простое число\n"); }
    else { printf("%d",n);
          printf("не простое число\n"); }
    return 0;
}
```

C

## Операторы

### Задание на дом:

решить, используя операторы цикла с пред- и пост-условием 2-а индивидуальные задания.

Нарисовать блок-схемы алгоритмов решения и написать программы на C.