Типы алгоритмов. Циклы. Понятие рекурсии

Откройте для себя фундаментальные принципы, лежащие в основе всех компьютерных программ и систем, — мир алгоритмов.

Линейный алгоритм: Последовательность действий

Линейные алгоритмы — это набор команд, выполняющихся строго одна за другой, без каких-либо условий или повторений.

Строгая последовательность

Команды выполняются строго по порядку, одна за другой.

Однократное выполнение

Каждое действие выполняется ровно один раз от начала до конца.

Простота и предсказуемость

Обеспечивает простой и предсказуемый поток выполнения без ответвлений или циклов.

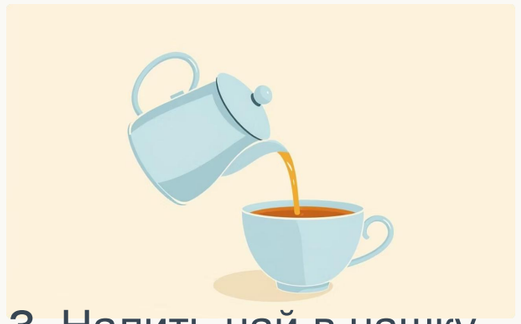
Пример: Алгоритм Заваривания Чая

1. Вскипятить воду

Начните с кипячения необходимого количества воды.

2. Заварить чайные листья

Поместите чайные листья в заварочный Чайник и залейте кипятком.



3. Налить чай в чашку

Аккуратно налейте заваренный чай в вашу любимую чашку.

4. Добавить сахар по вкусу

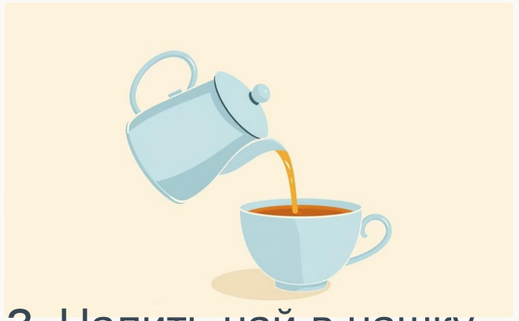
При желании добавьте сахар или другие подсластители.

Линейный алгоритм: Последовательность действий

Пример: Алгоритм Заваривания Чая

1. Вскипятить воду

2. Заварить чайные листья



3. Налить чай в чашку

4. Добавить сахар по вкусу

```
1  a = input("Вскипятить чайник")
2  b = input("Заварить чайные листья")
3  c = input("Налить чай в чашку")
4  d = input("Добавить сахар по вкусу")
5  print(a,b,c,d)
6
7
```

```
Вскипятить чайник
Заварить чайные листья
Налить чай в чашку
Добавить сахар по вкусу
```

Разветвляющийся алгоритм: Выбор по условию

Разветвляющиеся алгоритмы позволяют программе принимать решения и выбирать один из нескольких путей выполнения в зависимости от заданного условия.

Проверка Условия

Встречается точка принятия решения, где система проверяет заданное условие.

Два Пути

Если условие истинно — выполняется один набор действий; если ложно — другой.

Слияние

После выполнения соответствующих действий, поток выполнения обычно снова объединяется.

Пример из Жизни: Собираясь на улицу

Если идёт дождь

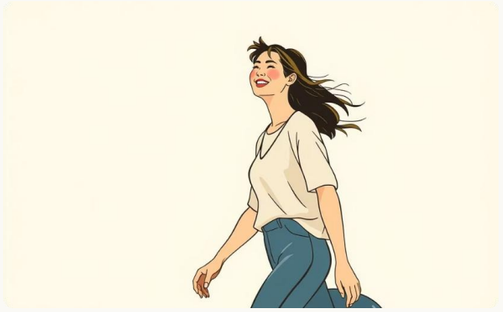
Проверяем, идет ли дождь.

Тогда взять зонт

Если условие истинно, берем зонт.

Иначе

Если условие ложно (дождя нет).



Идти без зонта

Идем на улицу без зонта.

Разветвляющийся алгоритм: Выбор по условию

Пример из Жизни: Собираясь на улицу

Если идёт дождь

Тогда взять зонт

Иначе

Идти без зонта

```
a = input("Идёт дождь? ")
if a == "Да":
    print("Взять зонт")
else:
    print("Не брать зонт")
print("Идти гулять")
```

```
Идёт дождь? Да
Взять зонт
Идти гулять
```

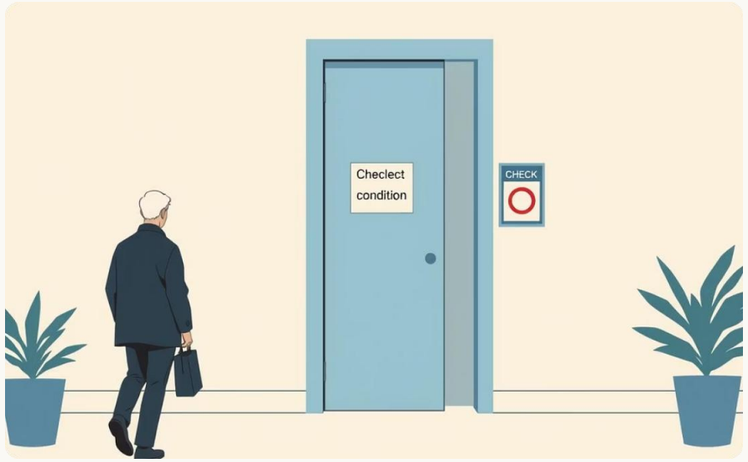
```
Идёт дождь? нет
Не брать зонт
Идти гулять
```

Циклический алгоритм: Повторение действий

Циклические алгоритмы предназначены для многократного выполнения одного и того же блока команд, пока не будет достигнуто определённое условие.

Цикл с предусловием (While)

Условие проверяется **до** начала каждой итерации. Блок команд выполнится, только если условие истинно.



Цикл с постусловием (Do...While)

Блок команд выполняется **хотя бы один раз**, а условие проверяется после выполнения тела цикла.

Цикл с параметром (For)

Используется, когда известно **точное количество** повторений, часто для итерации по диапазону значений.

Блок-схемы циклических алгоритмов всегда включают "петлю", указывающую на повторное выполнение шагов.

Цикл с предусловием (WHILE)

Этот тип цикла выполняет набор инструкций до тех пор, пока заданное условие остается истинным. Условие проверяется перед каждой потенциальной итерацией.

Принцип Работы

Сначала Проверка

Цикл начинается с проверки заданного условия. Это происходит перед каждой возможной итерацией.

Возможность Нулевых Итераций

Если условие изначально ложно, тело цикла не будет выполнено ни разу, то есть количество итераций равно нулю.

Повторение

Тело цикла выполняется только в том случае, если условие истинно. После выполнения тело, условие проверяется снова, и цикл продолжается, пока условие остается истинным.

Визуализация Потока WHILE-Цикла

Пример

Пока батарея заряжена (условие истинно), **продолжать** работу устройства. Как только батарея разряжается, устройство перестает работать.

Цикл с предусловием (WHILE)

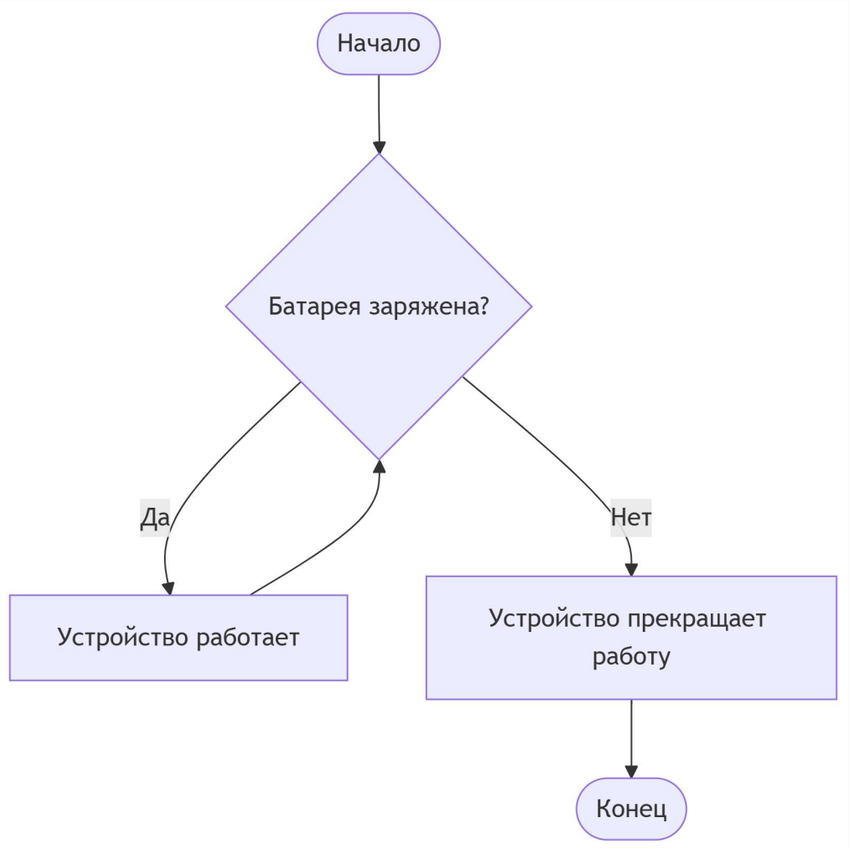
Визуализация Потока WHILE-Цикла

Пример

Пока батарея заряжена (условие истинно), **продолжать** работу устройства. Как только батарея разряжается, устройство перестает работать.

```python
import random
def simulate_device():  1 usage
    """Упрощенная симуляция работы устройства"""
    battery_level = 100
    while battery_level > 0:  # Пока батарея заряжена
            print(f"Устройство работает... Уровень заряда: {battery_level}%")
            # Разряжаем батарею
            battery_level -= random.randint( a: 5,  b: 15)
            if battery_level < 0:
                battery_level = 0

            # Условие стало ложным - выходим из цикла
    print("Батарея разрядилась! Устройство прекращает работу.")
    print("Устройство выключается.")
# Запуск симуляции
simulate_device()
```



Цикл с постусловием (DO...WHILE)

Цикл с постусловием гарантирует, что тело цикла будет выполнено хотя бы один раз, поскольку условие проверяется только после первой итерации.

Принцип Работы

- Минимум Одно Выполнение: Тело цикла выполняется хотя бы один раз.
- Проверка После: После выполнения тела цикла происходит проверка условия.
- Повторение: Если условие истинно, цикл повторяется; если ложно — завершается.

Пример

Делать: Спросить пароль.
Пока: Пароль неверный (повторять запрос).

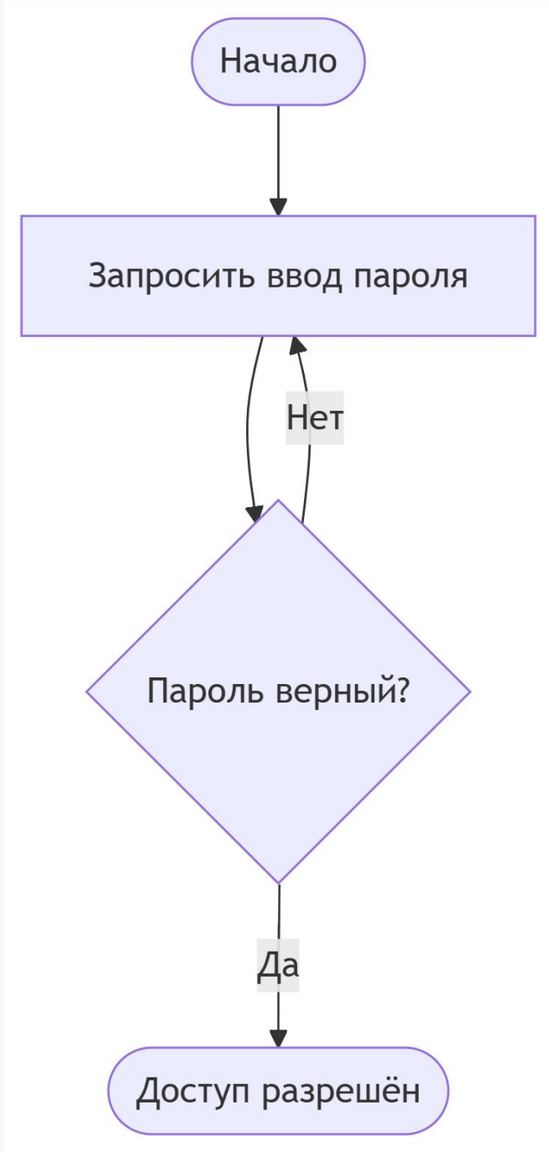
Цикл с постусловием (DO...WHILE)

Пример

Делать: Спросить пароль.
Пока: Пароль неверный (повторять запрос).

```
1   password_correct = "123456"
2   while True:
3     password_user = input("Введите пароль: ")
4     if password_user == password_correct:
5         print("Пароль верный! Доступ разрешен.")
6         break  # Выходим из цикла
7     else:
8         print("Пароль неверный! Попробуйте еще раз.")
9         print("---")
10
11  print("Вы успешно вошли в систему!")
12
```

```
Введите пароль: 3
Пароль неверный! Попробуйте еще раз.
---
Введите пароль: 1234567
Пароль неверный! Попробуйте еще раз.
---
Введите пароль: 123456
Пароль верный! Доступ разрешен.
Вы успешно вошли в систему!
```



Цикл с параметром (FOR)

Цикл с параметром, также известный как "for" цикл, идеален, когда заранее известно точное количество повторений или когда нужно итерировать по определенному диапазону значений.

Компоненты Цикла "For"

- Инициализация: Устанавливает начальное значение для переменной-счетчика (например, `i = 1`).
- Условие: Определяет, когда цикл должен остановиться (например, `i <= 10`).
- Инкремент/Декремент: Изменяет значение переменной-счетчика после каждой итерации (например, `i++`).

Пример

Для каждого числа **от** 1 **до** 10: **вывести** число.

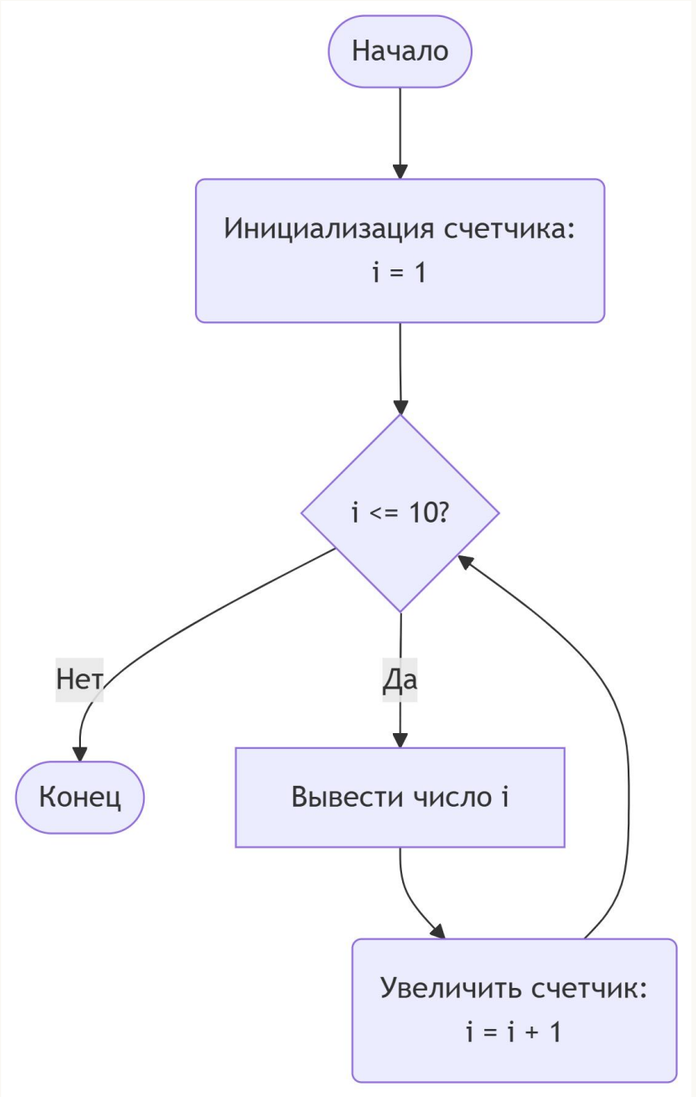
Цикл с параметром (FOR)

Пример

Для каждого числа **от** 1 **до** 10: **вывести** число.

```
1  for number in range(1, 11):
2      print(number)
```

```
1
2
3
4
5
6
7
8
9
10
```



Использование циклов для обработки массивов

Циклы являются незаменимым инструментом для эффективной работы с массивами, позволяя последовательно обращаться к каждому элементу или выполнять над ними операции.

Массив — это структура данных, которая позволяет хранить множество элементов одного типа в одной переменной. Массивы позволяют компактно хранить информацию и манипулировать большими объемами данных, обеспечивая быстрый доступ к ним.

Типичные Задачи

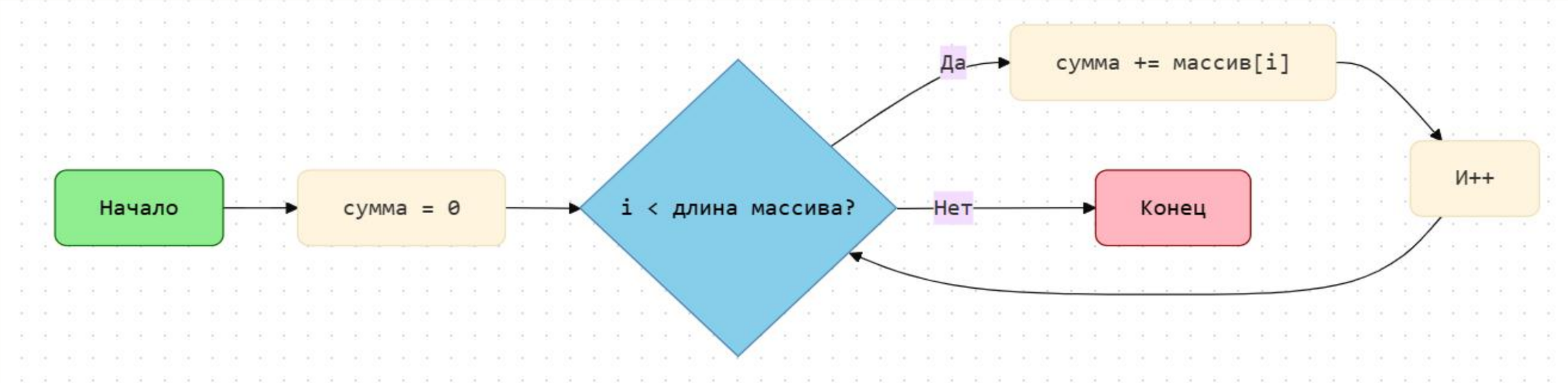
- Перебор Элементов: Доступ к каждому элементу массива по порядку.
- Поиск: Нахождение конкретного значения в массиве.
- Агрегация (объединение): Вычисление суммы, среднего, минимума или максимума всех элементов.
- Модификация (видоизменение): Изменение значений элементов массива.

Пример: Сумма Элементов
Используем цикл **for**, чтобы проитерировать по всем числам в массиве и добавить каждое к общей сумме.

Использование циклов для обработки массивов

Пример: Сумма Элементов

Используем цикл **for**, чтобы проитерировать по всем числам в массиве и добавить каждое к общей сумме.




```python
numbers = [1, 2, 3, 4, 5]
total_sum = 0
for num in numbers:
    total_sum += num
print(f"Сумма чисел {numbers} равна: {total_sum}")
```

```
Сумма чисел [1, 2, 3, 4, 5] равна: 15
```

Понятие рекурсии: Алгоритм, вызывающий сам себя

Рекурсия — это метод решения задач, при котором функция (или алгоритм) вызывает саму себя для решения подзадач, пока не достигнет базового, легко решаемого случая.

Ключеые Элементы

- Базовый Случай: Условие, при котором рекурсия завершается и возвращает значение без дальнейших вызовов. Это критически важно для предотвращения бесконечного цикла.
- Рекурсивный Вызов: Функция вызывает саму себя, но с измененным (уменьшенным или упрощенным) входным параметром.

Пример: Факториал

Факториал числа **n** (**n!**) определяется как произведение всех положительных целых чисел от 1 до **n**.

$$n! = n \times (n - 1)!$$

Базовый случай: `1! = 1` и `0! = 1`.

Понятие рекурсии: Алгоритм, вызывающий сам себя

Пример: Факториал

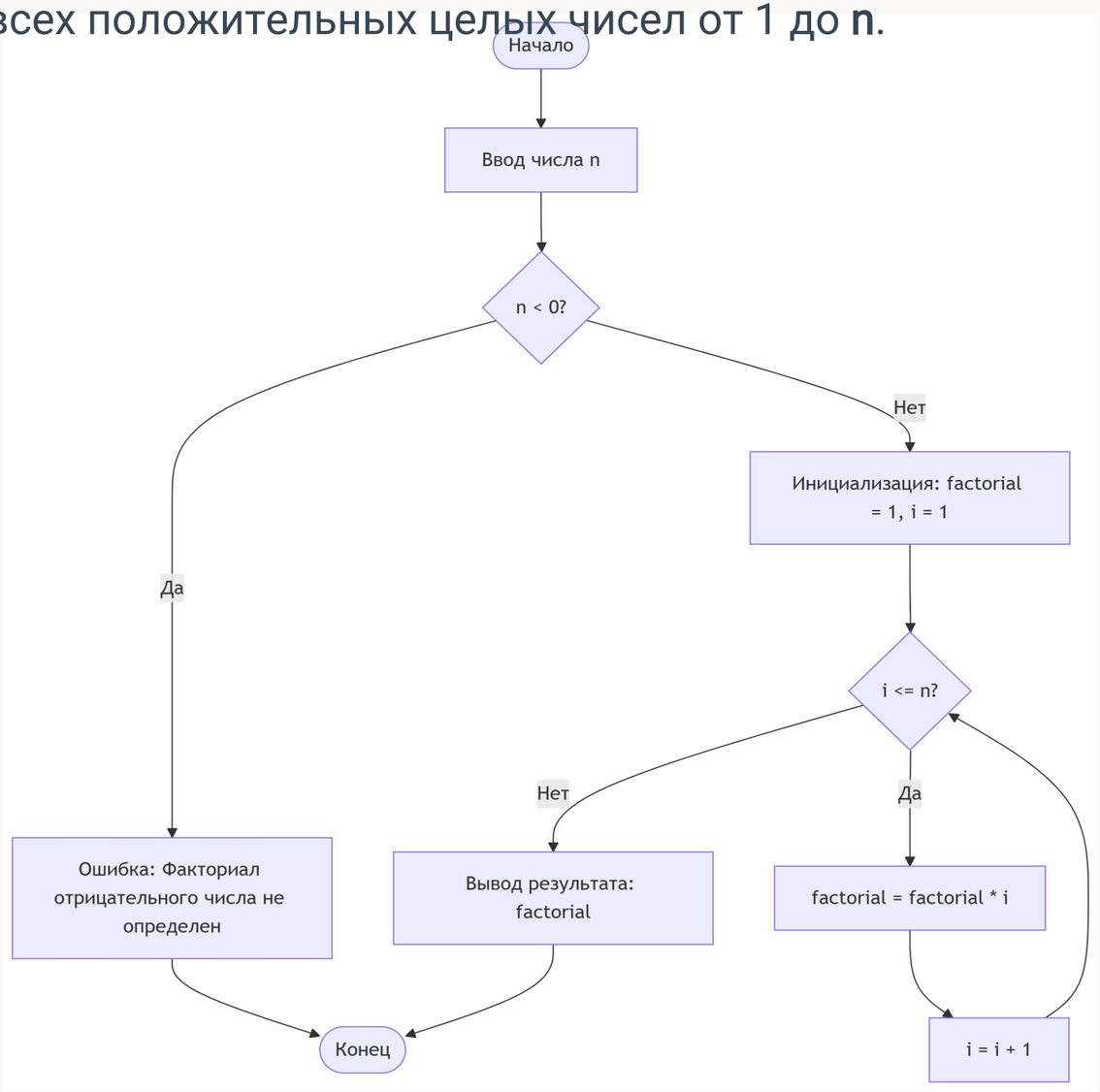
Факториал числа **n** (**n!**) определяется как произведение всех положительных целых чисел от 1 до **n**.

$$n! = n \times (n - 1)!$$

Базовый случай: `1! = 1` и `0! = 1`.

```python
n = int(input("Введите неотрицательное целое число n: "))
# Проверка на отрицательное число
if n < 0:
    print("Ошибка: Факториал отрицательного числа не определен")
else:
    # Инициализация переменной для хранения результата
    factorial = 1
    # Вычисление факториала с помощью цикла for
    for i in range(1, n + 1):
        factorial *= i  # Умножаем текущее значение на i
    # Вывод результата
    print(f"Факториал числа {n}! = {factorial}")
```

```
Введите неотрицательное целое число n: 23
Факториал числа 23! = 25852016738884976640000
```



Итоги и Интерактив по Теме

Мы рассмотрели основные типы алгоритмов, которые являются строительными блоками любой программы. Понимание этих концепций критически важно для разработки эффективных и логичных решений.

1 Линейные, Разветвляющиеся и Циклические

Фундаментальные структуры для последовательного, выборочного и повторяющегося выполнения действий.

2 Три Вида Циклов

While (предусловие), **Do...While** (постусловие) и **For** (с параметром) — каждый для своих задач.

3 Рекурсия

Элегантный способ решения задач путём их декомпозиции до простейшего базового случая.

Вопросы для Обсуждения:

- Как бы вы преобразовали повседневную задачу в один из рассмотренных типов алгоритмов?
- Приведите примеры, когда использование рекурсии будет оптимальным?