# Логические основы программирования.

Программирование основано на логических конструкциях, которые позволяют принимать решения и управлять выполнением программы в зависимости от условий. Основные элементы:

1. Булева алгебра (Bool)

Используется для работы с истинностными значениями: `True` (истина) и `False` (ложь). Эти значения являются результатом логических выражений.

*Часто переменные со значениями bool называют флажками или кнопками. То есть в значении либо 0, либо 1*

2. Логические операции *(их можно проводить только с булевыми переменными)*

* and — логическое И: результат истинен, только если оба операнда истинны.

*True and False → False*

* or — логическое ИЛИ: результат истинен, если хотя бы один операнд истинен.

*True or False → True*

* not — логическое НЕ: меняет значение на противоположное.

*not True → False*

3. Операции сравнения

**Результатом** всегда является булево значение:

* == — равно
* ! = — не равно
* <, >, <=, >= — сравнение по величине

4. Условные операторы (if, elif, else)

Позволяют выполнять разные блоки кода в зависимости от истинности логического выражения.

age = 18

if age >= 18:

print ("Доступ разрешён")

else:

print("Доступ запрещён")

5. Логические выражения в циклах

Управляют продолжением циклов (while, for), например:

while x > 0:

print(x)

x -= 1

# Общие принципы построения алгоритмов.

Общие принципы построения алгоритмов

Алгоритм — это точная последовательность действий, ведущая к решению задачи. Основные принципы его построения:

1. Дискретность  
   Алгоритм состоит из отдельных шагов (команд), выполняемых одно за другим.
2. Определённость  
   Каждый шаг должен быть чётко и однозначно сформулирован.
3. Массовость  
   Алгоритм должен работать при различных допустимых входных данных.
4. Результативность  
   После выполнения алгоритма должен быть получен конкретный результат.
5. Конечность  
   Алгоритм должен завершаться за конечное число шагов.
6. Формальность  
   Выполнение алгоритма не требует творческого мышления — он должен быть механически исполняем.
7. Эффективность  
   Алгоритм должен использовать минимально возможные ресурсы (время, память).

Алгоритмы описываются с помощью:

* Словесной записи
* Блок-схем
* Псевдокода
* Языков программирования

В основе большинства алгоритмов лежат три структуры:

* Следование — команды выполняются последовательно.
* Ветвление — выбор пути в зависимости от условия (**if**, **else**).
* Цикл — повторение действий (**for**, **while**).

# Основные алгоритмические конструкции.

3. Основные алгоритмические конструкции

Алгоритмические конструкции — это базовые логические структуры, из которых строится любой алгоритм. Их три:

1. Следование  
   Последовательное выполнение шагов в порядке их записи.
2. Ветвление (развилка)  
   Выбор одного из направлений выполнения алгоритма в зависимости от условия.  
   Реализуется через **if**, **elif**, **else**.
3. Цикл (повторение)  
   Многократное выполнение действий при соблюдении определённого условия.  
   Виды циклов:
   * **for** — с параметром
   * **while** — с предусловием

Эти три конструкции составляют основу структурного программирования и позволяют описать любую логику решения задачи.

# Языки и системы программирования.

# Методы программирования.

# Принципы составления логически правильных и эффективных программ.

# Понятие системы программирования.

# Основные элементы языка процедурного языка программирования.

# Структура программы.

# Операторы и операции языка.

# Управляющие структуры, структуры данных.

# Массивы как структурированный тип данных.

# Объявление массива.

# Одномерные массивы.

# Двумерные массивы.

# Структурированные типы данных: строки и множества.

# Объявление строковых типов данных.

# Операции со строками.

# Объявление множеств.

# Операции над множествами.

# Понятие подпрограммы.

# Процедуры и функции, их сущность, назначение, различие.

# Организация процедур, стандартные процедуры.

# Процедуры, определенные пользователем.

# Функции: способы организации описание.

# Вызов функций, рекурсия.

# Программирование рекурсивных алгоритмов.

# Стандартные функции.

# Файлы, классы памяти.

# Типы файлов.

# Организация доступа к файлам.

# Файлы последовательного доступа.

# Файлы произвольного доступа.

# Порядок работы с файлами произвольного доступа.

# Создание структуры записи.

# Открытие и закрытие файла произвольного доступа.

# Запись и считывание из фала произвольного доступа.

# Программирование модулей.

# Библиотеки подпрограмм.

# Схемы вызова библиотек.

# Синтаксическое и динамическое связывание.

# Использование библиотек подпрограмм.

# Объектно-ориентированная модель программирования.

# Основные принципы ООП.

# Событийно-управляемая модель программирования.

# Компонентно-ориентированный подход.

# Понятие классов и объектов, их свойств и методов.

# Требования к аппаратным и программным средствам интегрированной среды разработчика (ИСР).

# Интерфейс среды разработчика.

# Настройка параметров проекта.

# Классы объектно-ориентированного языка программирования.

# Объявление класса, свойства и методы экземпляра класса. Наследование.

# Перегрузка методов.

# Основные компоненты ИСР. Дополнительные элементы управления.

# Свойства компонентов. Виды свойств. Назначение свойств.

# События компонентов, их сущность и назначение.

# Создание процедур на основе событий. Вызов событий.

# Разработка функционального интерфейса приложения.

# Создание интерфейса приложения.

# Разработка функциональной схемы работы приложения.

# Создание процедур обработки событий.

# Компиляция и запуск приложения.