Языки программирования

Программа – это логически упорядоченная последовательность команд, необходимых для управления компьютером (выполнения им конкретных операций).

Программирование – процесс создания программы.

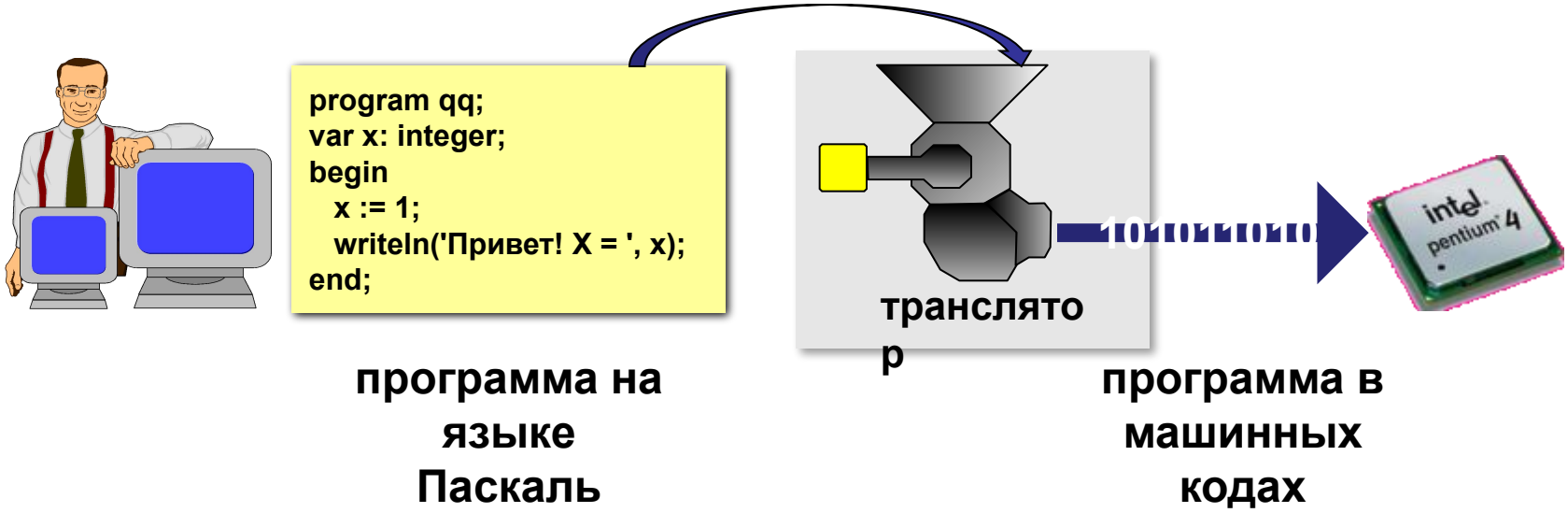
Главным исполнителем в компьютере является процессор

Процессор может исполнять только программы, написанные на языке машинных кодов (язык двоичных кодов: последовательность нулей и единиц).

Программы, написанные на других языках программирования, необходимо перевести на язык машинных кодов при помощи специальных программ – трансляторов.

Трансляторы

Транслятор – это программа, которая переводит текст других программ в машинные коды.



программа на языке Паскаль

программа в машинных кодах

Типы трансляторов

- **интерпретатор – переводит в коды 1 строчку программы и сразу ее выполняет;**
  -  удобнее отлаживать программу
  -  программы работают медленно
  - для выполнения программы нужен транслятор
- **компилятор – переводит в коды сразу всю программу и создает независимый исполняемый файл (*.exe, *.com);**
  -  сложнее отлаживать программу
  -  программы работают быстро
  - для выполнения программы не нужен транслятор

Система программирования

включает в себя транслятор,
а также вспомогательные программы
(текстовый редактор, средства отладки
программ и др.)

Два основных режима работы системы программирования:

– режим ввода текста программы

– режим исполнения программы

Что такое язык программирования?

Языки программирования – искусственные языки. От естественных они отличаются ограниченным числом «слов», значение которых понятно транслятору, и очень строгими правилами записи команд (*операторов*).

Синтаксис языка программирования - совокупность правил построения из символов алфавита алгоритма программы.

Семантика – система правил толкования смысла каждой команды и конструкций языка.

Нарушение формы записи приводит к тому, что транслятор не может понять назначение оператора и выдает сообщение о синтаксической ошибке, а правильно написанные команды, но не отвечающие алгоритму, приводят к семантическим ошибкам.

Тестирование - процесс поиска ошибок в программе.

Отладка - процесс устранения ошибок.

Уровни языков программирования

- **Низкий уровень -** язык программирования ориентирован на конкретный тип процессора и учитывает его особенности.

«Низкий уровень» не значит «плохой». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.

- **Высокий уровень**

Языки программирования высокого уровня значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка.

Языки программирования

Всего более 600, широко используется примерно 20.

Машинно-ориентированные языки:

- **машинные коды: 09 FE AC 3F**
- **ассемблеры: символическая запись машинных команд:**
  mov AX, BX
- **макросассемблеры: одна команда языка заменяет несколько машинных команд**

Языки высокого уровня (алгоритмические):

- **для обучения: Бейсик (1965), Паскаль (1970), Лого, Рапира**
- **профессиональные: Си (1972), Паскаль (Delphi), Кобол, Фортран (1957), Алгол, Visual Basic**
- **для задач искусственного интеллекта: ЛИСП, Пролог**
- **для параллельных вычислений: Ада**
- **для программирования в Интернете: HTML, VRML, XML, JavaScript, Java, PHP, Perl, ASP, …**

Структурное программирование - **представление программы в виде иерархической структуры блоков команд.**

Границы блока строго определены: BEGIN...END (язык Паскаль)
фигурные скобки {...} (язык С)

Основой структурного программирования являются:

1) **алгоритм любой логической задачи можно составить только из структур**

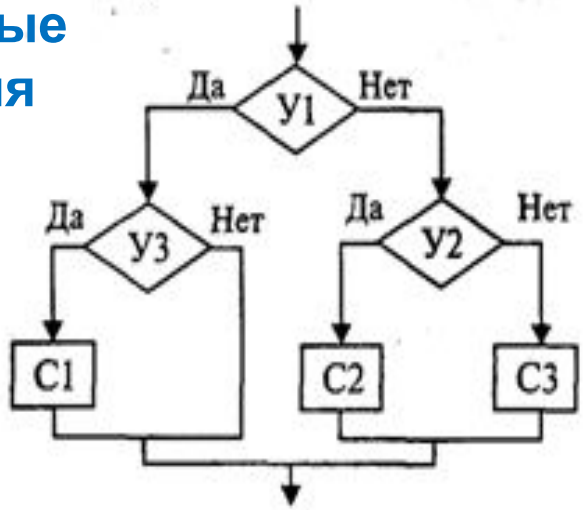
- **следование,**
- **ветвление,**
- **цикл.**

→ **базовые алгоритмические структуры**

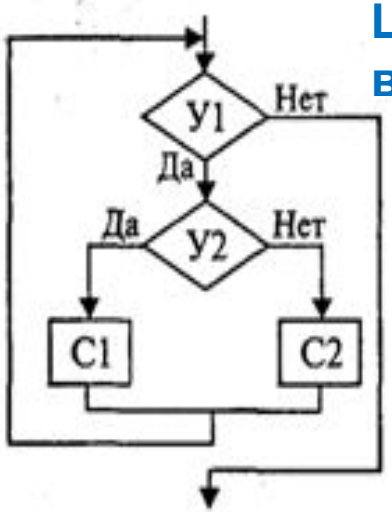
Сложный алгоритм состоит из соединенных между собой базовых структур. Соединяться эти структуры могут двумя способами:

- **последовательным,**
- **вложенным.**

Вложенные ветвления



Цикл с вложенным ветвлением



2) Принцип модульной разработки программ.

Программа разбивается на отдельные смысловые части - **модули** (должен иметь один вход и один выход).

Модуль – это функционально законченная часть программы.

Модуль на языке программирования – это функция или процедура: модуль вычисления определителя матрицы, модуль нахождения суммы элементов ряда и т.д.

Каждый модуль программируется отдельно, а затем модули объединяются в единую программу.

Методики (стратегии) разработки программы

- **программирование «сверху вниз»**

Задача разбивается на ряд подзадач. Затем каждая из полученных подзадач также анализируется для возможного разбиения на подзадачи. Процесс заканчивается, когда подзадачу невозможно или нецелесообразно далее разбивать на подзадачи.

В данном случае программа конструируется иерархически - сверху вниз: от главной программы к подпрограммам самого нижнего уровня.

- **программирование "снизу вверх"**

Программирование начинается с разработки подпрограмм (процедур, функций), в то время когда проработка общей схемы не закончилась.

Такая методика является менее предпочтительной по сравнению с программированием «сверху вниз», т.к. часто приводит к нежелательным результатам, переделкам и увеличению времени разработки.

Понятие заглушки модуля

При структурном программировании программа в основном реализуется (собирается и тестируется) сверху вниз. Сначала из 20–30 модулей пишется ядро. Чтобы начать тестировать, недостающие модули нижних уровней заменяются заглушками. По окончании тестирования ядра, заглушки заменяются новыми готовыми модулями, но если программа еще не закончена, то понадобятся все новые заглушки недостающих модулей. Теперь можно приступать к тестированию собранной части и т. д.

Самая простая заглушка — это подпрограмма или функция без действий. Более сложная заглушка может выводить сообщение о том, что отработал такой-то модуль.

Еще более сложные заглушки могут выводить тестовую информацию и т.д.

Объектно-ориентированное программирование (ООП)

Первый язык ООП - Simula 67 (конец 60-х годов, Норвегия).
Современные языки ООП - C++, Java , Object Pascal Smalltalk и др.

В ООП два ключевых понятия:
Класс – это абстрактный тип данных. С помощью класса описывается некоторая сущность (ее характеристики и возможные действия). Например, класс может описывать студента, автомобиль и т.д.
Объект (экземпляр) – это конкретный представитель класса.

Пример.
Программа должна работать со странами.
Страна – это абстрактное понятие.
Её характеристики: название, население, площадь, флаг и др.
Для описания такой страны будет использоваться класс с соответствующими полями данных.
Россия, Китай, Польша - объекты (конкретные представители типа страна).

Основные принципы ООП:

Инкапсуляции – это механизм, который объединяет данные и методы, манипулирующие этими данными, и защищает и то и другое от внешнего вмешательства или неправильного использования.

Наследования – это процесс, посредством которого, один объект может приобретать свойства другого. Точнее, объект может наследовать свойства другого объекта и добавлять к ним черты, характерные только для него.

При этом на базе одного класса создаётся новый класс.

Класс, на базе которого создается новый класс, называется базовым, а базирующийся новый класс – наследником.

Например. Базовый класс - Животное. В нем описаны общие характеристики для всех животных.

Классы наследники - Собака, Обезьяна со своими специфическими свойствами.

Все свойства и методы базового класса при наследовании переходят в класс наследник.

Полиморфизм – это принцип, который позволяет одно и тоже имя действий использовать для решения нескольких технически разных задач.

Например, есть два класса - Круг и Квадрат. У обоих классов есть метод Square, который считает и возвращает площадь. Но площадь круга и квадрата вычисляется по-разному, соответственно, реализация одного и того же метода различная.

Интегрированные среды (оболочки) разработки (Integrated Development Environment, IDE) позволяют избежать большого объема однообразных действий и тем самым существенно повысить эффективность процесса разработки и отладки программы.

Интегрированные среды разработки программ (ИСР) —

— система программных средств, используемая программистами для разработки программного обеспечения (ПО).

Среда разработки включает:

- **текстовый редактор;**
- **библиотеки справочных программ (функций,процедур)**
- **компилятор и / или интерпретатор;**
- **редакторы связей (связывают объектные модули и функции, находя их в библиотеках);**
- **отладчик и др.**

Наиболее популярные ИСР:

- **Visual Basic,**
- **Delphi,**
- **C++ Builder,**
- **Visual C++,**
- **Turbo Vision.**