# Лекция №7. Языки программирования для описания задач в АСУП

**7.1. Классификация языков программирования для АСУП**

**7.2. Характеристика языков программирования**

# 7.1. КЛАССИФИКАЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ В АСУП

Классификация существующих языков программирования в АСУП приведена на рис.7.1

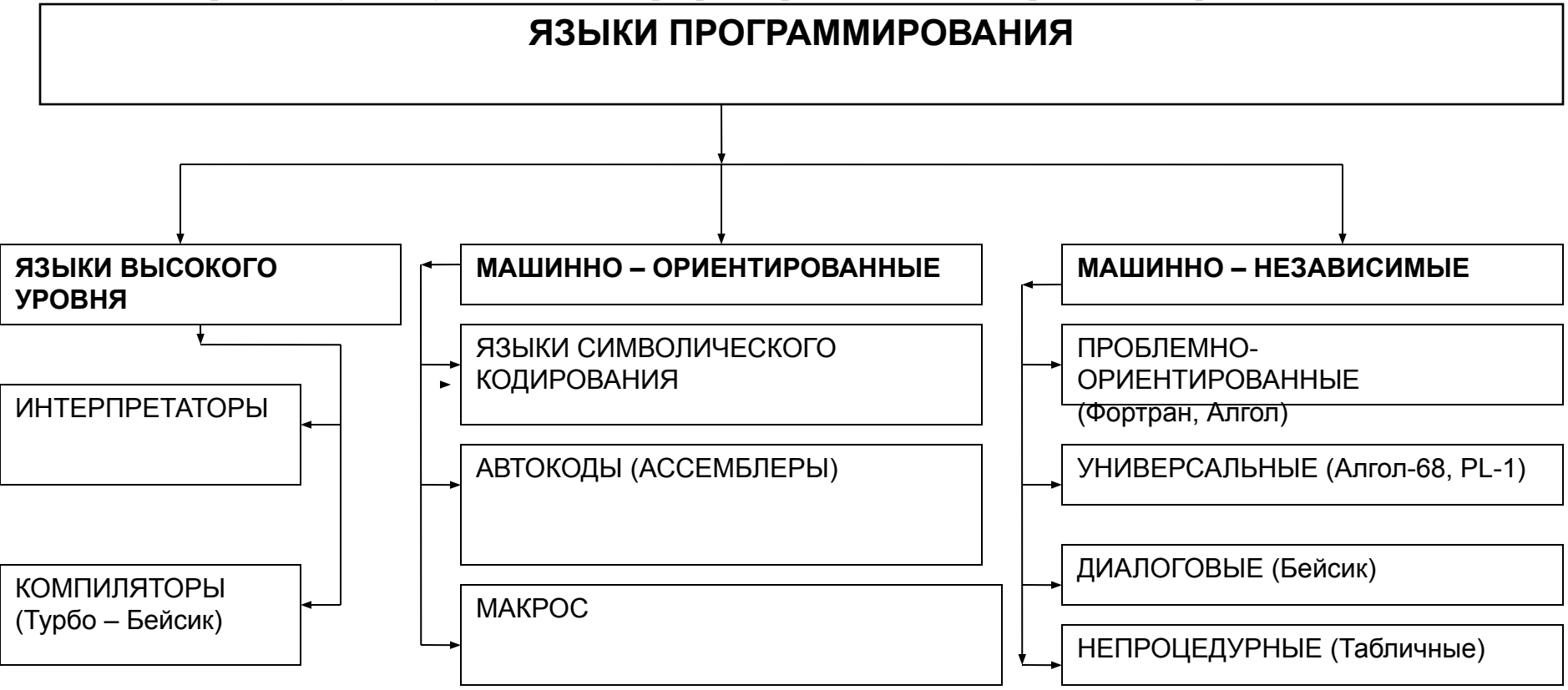


**Рис.7.1 Классификация языков программирования**

# языки высокого уровня

- **языки высокого уровня** (т.е. немашинные языки), которые стали своеобразным связующим мостом между человеком и машинным языком компьютера. Языки высокого уровня работают через **трансляционные программы,** которые вводят "исходный код" (гибрид английских слов и математических выражений, который считывает машина), и в конечном итоге заставляет компьютер выполнять соответствующие команды, которые даются на машинном языке. Существует **два основных вида трансляторов: интерпретаторы**, которые сканируют и проверяют исходный код в один шаг, и **компиляторы,** которые сканируют исходный код для производства текста программы на машинном языке, которая затем выполняется отдельно.
- **7.1.1. Интерпретаторы**
- Одно из преимуществ интерпретаторной реализации состоит в том, что она допускает "непосредственный режим". Непосредственный режим позволяет задавать компьютеру задачу вроде PRINT 3.14159*3/2.1 и возвращает ответ, как только вы нажмете клавишу ENTER (это позволяет использовать компьютер стоимостью 3000 долларов в качестве калькулятора стоимостью 10 долларов). Кроме того, интерпретаторы имеют специальные атрибуты, которые упрощают отладку. Можно, например, прервать обработку интерпретаторной программы, отобразить содержимое определенных переменных, бегло просмотреть программу, а затем продолжить исполнение.

# 7.1.2. Компиляторы

- Компилятор-это транслятор текста на машинный язык, который считывает исходный текст. Он оценивает его в соответствии с синтаксической конструкцией языка и переводит на машинный язык. Компилятор не исполняет программы, он их строит.
- Интерпретаторы невозможно отделить от программ, которые ими прогоняются, компиляторы делают свое дело и уходят со сцены.
- *При работе с компилирующим языком, таким как Турбо-Бейсик, вы придете к необходимости мыслить о ваших программах в признаках двух главных фаз их жизни: периода компилирования и периода прогона.*

## 7.2. Характеристика ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

### 7.2.1. Машинно – ориентированные языки

**Машинно – ориентированные языки** – это языки, наборы операторов и изобразительные средства которых существенно зависят от особенностей ЭВМ (внутреннего языка, структуры памяти и т.д.). Машинно –ориентированные языки позволяют использовать все возможности и особенности машинно – зависимых языков:

- ***высокое качество создаваемых программ (компактность и скорость выполнения);***
- ***возможность использования конкретных аппаратных ресурсов;***
- ***предсказуемость объектного кода и заказов памяти;***
- ***для составления эффективных программ необходимо знать систему команд и особенности функционирования данной ЭВМ;***
- ***трудоемкость процесса составления программ ( особенно на машинных языках и ЯСК), плохо защищенного от появления ошибок;***
- ***низкая скорость программирования;***
- ***невозможность непосредственного использования программ, составленных на этих языках, на ЭВМ других типов.***

# Машинно-ориентированные языки подразде-ляются на классы:

### 7.2.1.1. Машинный язык (МЯ)

**МЯ** предписывают выполнение указываемых операций над определяемыми ими операндами, поэтому **МЯ** является командным. Однако, некоторые семейства ЭВМ (например, ЕС ЭВМ, IBM/370/ и др.) имеют единый **МЯ** для ЭВМ разной мощности. В команде любого из них сообщается информация о местонахождении операндов и типе выполняемой операции.

### 7.2.1.2. Языки Символического Кодирования

**Языки Символического Кодирования (далее ЯСК), так же, как и МЯ, являются командными. Однако коды операций и адреса в машинных командах, представляющие собой последовательность двоичных (во внутреннем коде) или восьмеричных (часто используемых при написании программ) цифр, в ЯСК заменены на символы (идентификаторы), форма написания которых помогает программисту легче запоминать смысловое содержание операции. Это обеспечивает существенное уменьшение числа ошибок при составлении программ.**

# Автокоды, Макрос

- **7.2.1.3. Автокоды**
- **Есть также языки, включающие в себя все возможности ЯСК, посредством расширенного введения *макрокоманд* - они называются Автокоды.**
- **Развитые автокоды получили название Ассемблеры. Сервисные программы и пр., как правило, составлены на языках типа Ассемблер**
- **7.2.1.4. Макрос**
- Язык, являющийся средством для замены последовательности символов описывающих выполнение требуемых действий ЭВМ на более сжатую форму - называется **Макрос** (средство замены).
- В основном, **Макрос** предназначен для того, чтобы сократить запись исходной программы. Компонент программного обеспечения, обеспечивающий функционирование макросов, называется **макропроцессором**.

### 7.2.2. Машинно – независимые языки

**Машинно – независимые языки** – это средство описания алгоритмов решения задач и информации, подлежащей обработке. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ЭВМ и ВС.

Подобные языки получили название высокоуровневых языков программирования. Программы, составляемые на таких языках, представляют собой последовательности операторов, структурированные согласно правилам рассматривания языка(задачи, сегменты, блоки и т.д.). Операторы языка описывают действия, которые должна выполнять  система после трансляции программы на **МЯ**.

Программист получил возможность не расписывать в деталях вычислительный процесс на уровне машинных команд, а сосредоточиться на основных особенностях алгоритма.

#### 7.2.2.1.  Проблемно – ориентированные языки

С расширением областей применения вычислительной техники возникла необходимость формализовать представление постановки и решение новых классов задач. Необходимо было создать такие языки программирования, которые, используя в данной области обозначения и терминологию, позволили бы описывать требуемые алгоритмы решения для поставленных задач, ими стали **проблемно – ориентированные языки**. Эти языки, языки ориентированные на решение определенных проблем, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме.Проблемных языков очень много, например:**Фортран, Алгол** – языки, созданные для решения математических задач;

**Simula, Слэнг** - для моделирования; **Лисп, Снобол** – для работы со списочными структурами.

- **7.2.2.2. Универсальные языки**
- **Универсальные языки** были созданы для широкого круга задач: коммерческих, научных, моделирования и т.д. Первый универсальный язык был разработан фирмой IBM, ставший в последовательности языков **Пл/1**. Второй по мощности универсальный язык называется **Алгол-68**. Он позволяет работать с символами, разрядами, числами с фиксированной и плавающей запятой. **Пл/1** имеет развитую систему операторов для управления форматами, для работы с полями переменной длины, с данными организованными в сложные структуры, и для эффективного использования каналов связи. Язык учитывает включенные во многие машины возможности прерывания и имеет соответствующие операторы. Предусмотрена возможность параллельного выполнение участков программ.
- Программы в **Пл/1** компилируются с помощью автоматических процедур. Язык использует многие свойства **Фортрана, Алгола, Кобола**. Однако он допускает не только динамическое, но и управляемое и статистическое распределения памяти.
- **7.2.2.3. Диалоговые языки**
- Появление новых технических возможностей поставило задачу перед системными программистами – создать программные средства, обеспечивающие оперативное взаимодействие человека с ЭВМ их назвали **диалоговыми языками**.
- Эти работы велись в двух направлениях. Создавались специальные управляющие языки для обеспечения оперативного воздействия на прохождение задач, которые составлялись на любых раннее неразработанных (не диалоговых) языках. Разрабатывались также языки, которые кроме целей управления обеспечивали бы описание алгоритмов решения задач.

Необходимость обеспечения оперативного взаимодействия с пользователем потребовала сохранения в памяти ЭВМ копии исходной программы даже после получения объектной программы в машинных кодах. При внесении изменений в программу с использованием диалогового языка система программирования с помощью специальных таблиц устанавливает взаимосвязь структур исходной и объектной программ. Это позволяет осуществить требуемые редакционные изменения в объектной программе.

Одним из примеров диалоговых языков является **Бэйсик**.

**Бэйсик** использует обозначения подобные обычным математическим выражениям. Многие операторы являются упрощенными вариантами операторов языка **Фортран**. Поэтому этот язык позволяет решать достаточно широкий круг задач.

## 7.2.2.4. Непроцедурные языки

составляют группу языков, описывающих организацию данных, обрабатываемых по фиксированным алгоритмам (табличные языки и генераторы отчетов), и языков связи с операционными системами.

Позволяя четко описывать как задачу, так и необходимые для её решения действия, таблицы решений дают возможность в наглядной форме определить, какие условия должны быть выполнены прежде чем переходить к какому-либо действию. Одна таблица решений, описывающая некоторую ситуацию, содержит все возможные блок-схемы реализаций алгоритмов решения.