Алгоритмизация и программирование. Языки программирования высокого уровня

Алгоритмизация
Чтение блок-схемы алгоритмов
Программирование
Эволюция языков программирования

Алгоритмизация

Постановка задачи связана с конкретизацией основных параметров ее реализации, определением источников и структурой входной и выходной информации, востребуемой пользователем.

Алгоритмизация – это процесс создания алгоритма решения задачи.

Алгоритм – точное предписание, определяющее вычислительный процесс, ведущий от варьируемых исходных данных к искомому результату.

В алгоритме обязательно должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения комплекса задач.

Само слово алгоритм происходит от имени арабского математика аль Хорезми. Его полное имя было Абу Абдуллах (или Абу Джафар) Мухаммад ибн Муса аль Хорезми. Он использовал индийскую позиционную систему счисления с нулем и в своих трудах сформулировал правила четырех арифметических действий над многозначными числами. Эти действия и стали впоследствии называть алгоритмами.

Составляя алгоритм для какого-нибудь автоматического устройства, мы должны принимать во внимание следующее:

- Первое - это кто *исполнитель*, тот, кто будет работать по нашему алгоритму. Какие действия он может выполнять, какие команды и в каком виде мы можем ему отдавать. Исполнителем, в принципе, может быть кто или что угодно: человек, компьютер, станок с ЧПУ, автоматический луноход и т.д.
- Второе - это как *скомбинировать* команды, чтобы исполнитель сделал то, что от него требуется. Это самый трудный этап в составлении алгоритма, использовать набор команд, понятных исполнителю, какие действия и в каком порядке выполнять.
- Третье - это *форма записи* алгоритма. Человек понимает смысл сказанной фразы по множеству неуловимых оттенков и интонаций. Одну и ту же мысль мы можем выразить различными словами. Компьютер же понимает алгоритм буквально (Хозяин, тебе дрова не нужны?), для него каждая конкретная команда всегда имеет один и тот же смысл. Поэтому алгоритмы для компьютеров записываются по строгим, заранее определенным правилам, чтобы их нельзя было истолковать по-разному.

Основные свойства алгоритма

- *Дискретность* – возможность разбиения алгоритма на отдельные элементарные действия, которые можно реализовать на ЭВМ и результат их выполнения определен и понятен.
- *Детерминированность (определенность)* - каждая команда алгоритма (предписание, выдаваемое на каждом шагу) должна быть понятна исполнителю, не оставлять места для ее неоднозначного толкования и неопределенного исполнения.
- *Результативность* алгоритма означает, что для любого допустимого набора исходных данных он должен через определенное число шагов завершить работу.
- *Массовость* алгоритма предполагает возможность изменения исходных данных в определенных пределах. Свойство определяет пригодность алгоритма для решения множества задач данного класса.

Чтение блок-схемы алгоритма

Алгоритм должен давать строгую и четкую последовательность действий, поэтому для него существенным является способ его задания. От исполнителя требуется лишь четкое выполнение каждого действия. Мы не должны объяснять ему, для каких целей предназначается алгоритм.

Возможности любого исполнителя всегда ограничены. Поэтому, прежде чем составлять алгоритм решения задачи, нужно узнать, какие действия исполнитель может выполнить. Эти действия называются *допустимыми действиями исполнителя*. Для решения одних и тех же задач исполнители с более «бедным» набором допустимых действий требуют более сложных подробных алгоритмов.

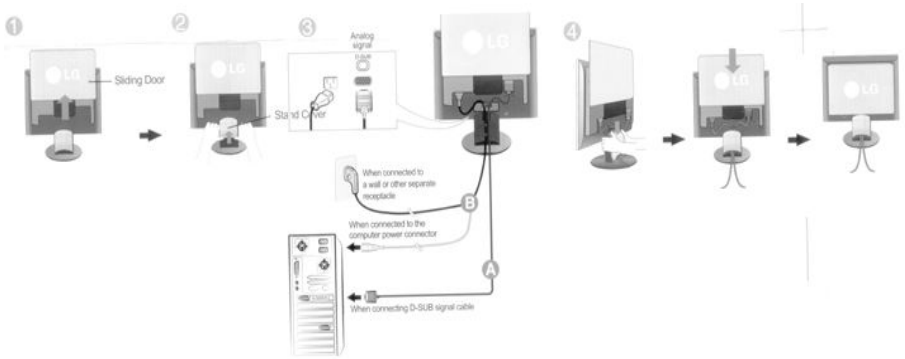
Алгоритм можно записывать разными способами ☞

Способы записи алгоритма

- С помощью рисунка, (например, процесс подключения монитора),
- На естественном языке - построчно, каждая команда - с новой строки (последовательность проявления фотопленки, последовательность склеивания поверхностей на тюбике с клеем и т.д.
- Использование псевдокода – некоторую систему обозначений и правил. Псевдокод занимает промежуточное место между естественным и формальным языками.

Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций (например школьный АЯ).

- Графическое представление – блок-схема

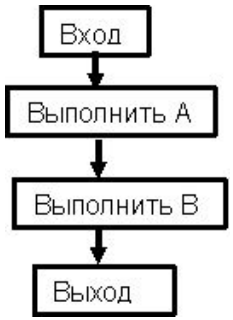


Коль кругом все будет мирно,
Так сидеть он будет смирно;
Но лишь чуть со стороны
Ожидать тебе войны
Иль набега силы бранной,
Иль другой беды незваной,
Вмиг тогда мой петушок
Приподымет гребешок,
Закричит и встрепенется
И в то место обернется.

А.С. Пушкин

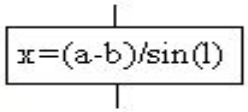
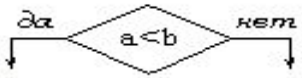
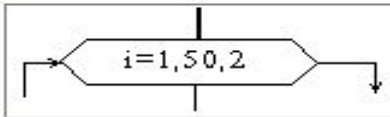
Вот как я хвостом махну,
В те котлы мордой макну,
На тебя два раза прысну,
Громким посвистом присвисну,
Ты, смотри же, не зевай:
В молоко сперва ныряй,
Тут в котел с водой вареной,
А оттудова в студеный.

П.П. Ершов



Основные функциональные элементы блок-схем алгоритмов в соответствии с ГОСТ 19002-89 ЕСПД

Графические обозначения блоков стандартизованы

Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод		Ввод-вывод в общем виде
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ

Алгоритм любой сложности может быть представлен комбинацией трех базовых структур:

- следование;
- ветвление (в полной и сокращенной форме);
- цикл (с предусловием или постусловием).

Основные типы алгоритмов:

- Линейные,
- Разветвляющиеся,
- Циклические.

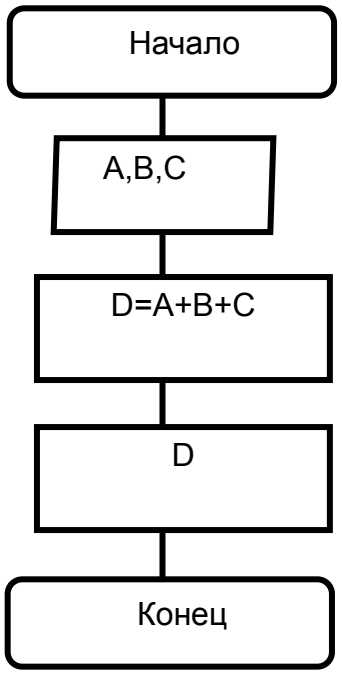
Первый:
Если А, то:
Действие 1
Д2
…
Дn
Иначе:
П1
П2
…
Пn
Конец ветвления

Второй:
Если В, то:
Д1
Д2
…
Дn
Конец ветвления

Линейные

Линейные алгоритмы, в которых все действия совершаются одно за другим, независимо от исходных данных и результатов промежуточных вычислений.

Характерная форма для линейного алгоритма - последовательное выполнение команд.



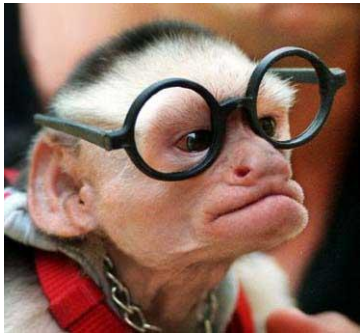
Разветвляющиеся

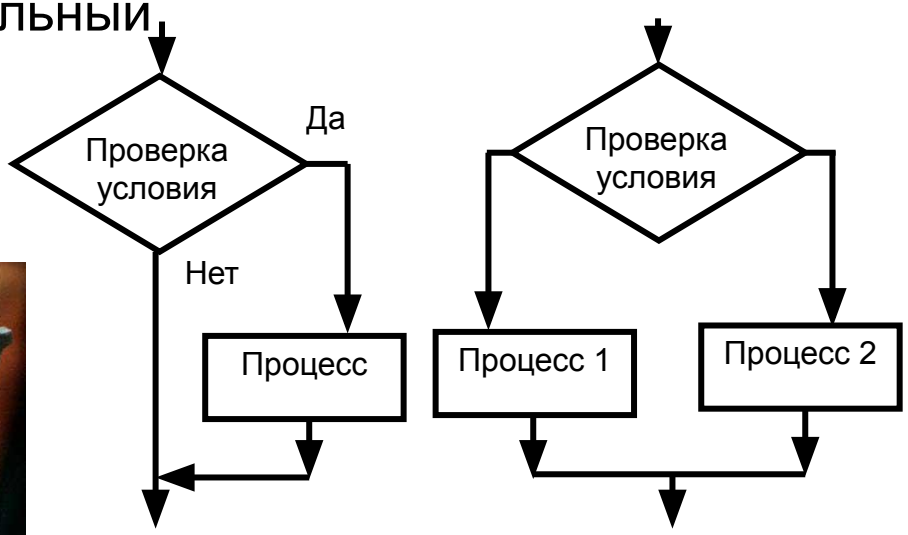
Разветвляющимся называют алгоритм, в котором в зависимости от исходных данных и результатов промежуточных вычислений осуществляется выбор по одному из возможных вариантов.

Варианты (направления вычислений), по которым может реализоваться вычислительный процесс, называют *ветвями*.

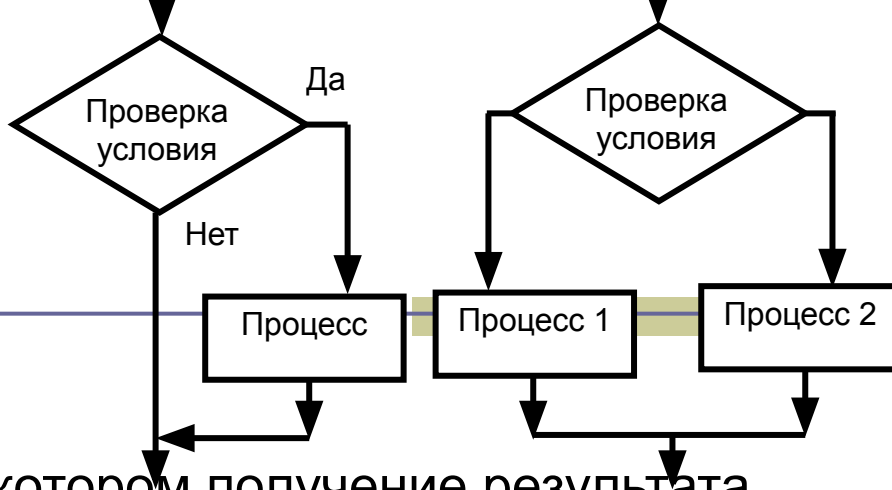
Выбор ветви зависит от результатов проверки некоторого условия. Если условие выполняется, то выбирается одна ветвь, если не выполняется, то другая ветвь.

В конце ветви должен быть специальный указатель, показывающий последнее действие. Можно, например, употребить слова "Конец ветвления".





ческие



Циклическим называют алгоритм, в котором получение результата обеспечивается многократным выполнением одних и тех же операций.

Цикл - многократно повторяющийся участок вычислительного процесса.

В цикле всегда имеется четыре действия:

- подготовка – задание начального значения параметру цикла,
- основные действия (тело цикла) – реализация необходимых вычислений,
- подготовка к следующему циклу (модификация) – изменение параметра цикла,
- проверка условия – проверка условия окончания цикла.

Способ организации цикла зависит от условия задачи. Иногда указывается количество повторений цикла. Это так называемые *циклы со счетчиками (или арифметические алгоритмы)*.

Типы циклических алгоритмов

- Цикл с *предусловием*. Перед выполнением цикла проверяется условие выполнения цикла. Если условие истинно, то цикл выполняется. При ложности условия цикл заканчивается.
- Цикл с *постусловием*. Условие продолжения цикла проверяется уже после того, как выполнено тело цикла.
- Основное различие: во втором случае цикл выполняется, по крайней мере, один раз, а в первом - может получиться, что цикл вообще не выполняется.
- Цикл с заданным числом повторений, когда указывается количество повторений цикла. Это так называемые циклы со *счетчиками* (или арифметические циклы).
- *Итерационный* цикл используется, когда задана точность вычисления результата. В таком цикле на каждом шаге (итерации) происходит постепенное уточнение результата.

В большинстве задач вычислительный процесс, реализующий алгоритм, является *комбинированным,* т.е. он содержит разветвления, является циклическим, или итерационным.

Программирование

Алгоритм, записанный на языке программирования, называется программой.

В алгоритме обязательно должны быть предусмотрены все ситуации, которые могут возникнуть в процессе решения комплекса задач.

Программирование – теоретическая и практическая деятельность, связанная с созданием программы.

Программа – результат интеллектуального труда, для которого характерно творчество, индивидуальность разработчиков.

Вместе с тем программирование предполагает и рутинные работы, которые могут и должны иметь строгий регламент выполнения и соответствовать стандартам.

Программирование базируется на комплексе научных дисциплин, направленных на исследование, разработку и применение методов и средств разработки программ (специального инструментария создания программы).

Эволюция языков программирования

20-е годы XIX века. Предварительная запись порядка действий машины на перфокарте для последующей автоматической реализации вычислений – программе (предложена Ч.Бэббиджем). Ада Лавлейс теоретически разработала некоторые приемы управления последовательностью вычислений, которые используются до сих пор.

40-е годы XX века. Создание программ на основе кодирования *машинных* команд (Грейс Мюрей Хоппер).

50-60-е годы. Роль программирования в машинных кодах уменьшается, появляются *процедурные* языки программирования высокого уровня (FORTRAN, ALGOL). Для преобразования команд в машинные коды используются *транслятор*ы.

Середина 60-х годов. Создание специализированного языка программирования, состоящего из простых слов английского языка (BASIC), попытки создать универсальный язык (PL/1, АЛГОЛ-68).

Начало 70-х. Развитие идеи АЛГОЛА о структуризации разработки алгоритмов, создание Н. Виртом языка Паскаль. Создание языка АДА, предназначенного для создания и длительного сопровождения больших программных систем, допускающего возможность параллельной обработки, управления процессами в реальном времени и др.

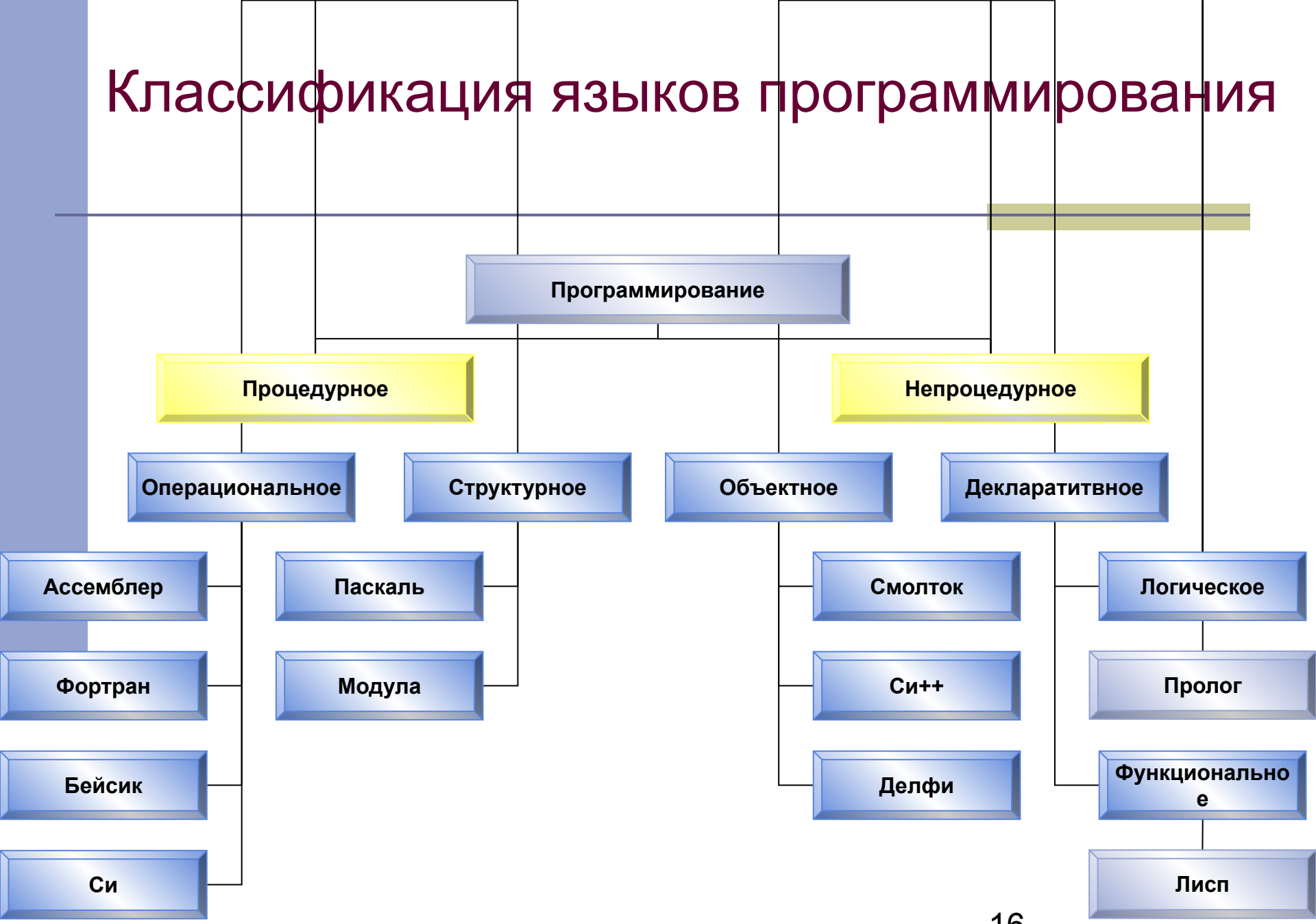
Продолжение

1972 г. (Первая версия языка Си). Появление языка сочетающего черты языка высокого уровня с *машинно-ориентированным* языком, который допускает программиста ко всем машинным ресурсам

В течение многих лет программное обеспечение строилось на основе *операциональных* и *процедурных* языков (Ассемблеры, Фортран, Бейсик, Паскаль, Ада, Си). По мере эволюции языков программирования широкое распространение получили и другие принципиально новые подходы к созданию программ *непроцедурное* программирование: *объектно-ориентированное* программирование, (языки Си++, Delphi, Visual Basic) и *декларативное* программирование. Декларативные языки делятся на *логические* (Пролог) и *функциональные* (Лисп).

В настоящее время разработаны языки работающие в управляемом окружении, обеспечивающие высокую надежность и защищенность создаваемых программ (Java, C#, VB.net).

Классификация языков программирования



Понятие о языках программирования высокого уровня

Языки программирования - это формальные языки специально созданные для общения человека с компьютером. Каждый язык программирования, равно как и "естественный язык" (русский, английский и т.д.) имеет:

Алфавит - фиксированный для данного языка набор основных символов, допускаемых для составления текста программы на этом языке.

Синтаксис - система правил, определяющих допустимые конструкции языка программирования.

Семантика - система правил однозначного толкования отдельных язы-ковых конструкций, позволяющих воспроизвести процесс обработки данных.

При описании языка и его применении используют понятия языка. **Понятие** - некоторая синтаксическая конструкция и определяемые ею свойства программных объектов или процесса обработки данных.

Взаимодействие синтаксических и семантических правил определяет те или иные понятия языка, например, *операторы, идентификаторы, переменные, функции и процедуры, модули* и т.д. В отличие от естественных языков правила грамматики и семантики для языков программирования, как и для всех формальных языков, должны быть явно, однозначно и четко сформулированы.

Языки программирования, имитирующие естественные языки, обладающие укрупненными командами, ориентированными на решение прикладных содержательных задач, называются **языками высокого уровня.**

ЯЗЫК ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

(*high-level programming language*).

Язык программирования, в который введены элементы, допускающие описание задачи в наглядном, легко воспринимаемом виде, упрощающие и автоматизирующие процесс программирования. Управляющие конструкции и структуры данных.

Я. п. в. у. отражают естественные для человека понятия, а не архитектуру вычислительной системы. Поэтому программа, составленная на Я. п. в. у., сначала *транслируется* самой ЭВМ на машинный язык (низкого уровня), а затем выполняется.

В алфавит Я. п. в. у. могут входить буквы, цифры, математические символы и даже так называемые ключевые слова, например, if (если), then (тогда), else (иначе) и т. п.

Из исходных символов по правилам синтаксиса строятся предложения, обычно называемые операторами, например, if x*1 следует воспользоваться формулой y = x - 1.*

Достоинства Я.п.в.у

- Алфавит языка значительно шире машинного, что делает его гораздо более выразительным и существенно повышает наглядность и понятность текста;
- набор операций, допустимых для использования, не зависит от набора машинных операций, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса;
- конструкции команд (операторов) отражают содержательные виды обработки данных и задаются в удобном для человека виде;
- используется аппарат переменных и действий с ними;
- поддерживается широкий набор типов данных.

Таким образом, языки программирования высокого уровня являются **машинно-независимыми** и требуют использования соответствующих программ-переводчиков (трансляторов) для представления программы на языке машины, на которой она будет исполняться.

Примеры языков высокого уровня

Fortran. Первый компилируемый язык созданный Джимом Бэкусом в 50-е годы. Для этого языка было создано огромное количество библиотек ,начиная от статических комплексов и кончая пакетами управления спутниками, поэтому Фортран продолжает активно использоваться во многих организациях, а сейчас ведутся работы над очередным стандартом Фортрана F2k,который появился в 2000 году. Имеется стандартная версия Фортрана HPF(High Perfomance Fortran) для параллельных супер компьютеров со множеством процессоров.

Cobol. Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х г.Он отличается большой "многословностью"-его операторы выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных , хранящимися на различных внешних носителях. На этом языке создано много различных приложений, которые активно эксплуатируются и сегодня. Достаточно сказать, что наибольшую зарплату в США получают программисты на Коболе.

Algol. Компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968 году была создана версия Алгол68,по своим возможностям опережающая и сегодня многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

Pascal. Язык Паскаль, созданный в конце 70-х годов основоположником множества идей современного программирования Николаусом Виртом, во многом напоминает Алгол, нр в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Продолжение

Basic. Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Он создавался в конце 60-х годов в качестве учебного пособия и очень прост в изучении.

C. Данный язык был создан в лаборатории Bell и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного вида процессора.

C++. это объектно-ориентированное расширения языка Си, созданное Бьярном Страуструпом в 1980 году. Множество новых мощных возможностей, позволивших резко увеличить производительность программистов, наложилось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки.

Java. Этот язык был создан компанией Sun в начале 60-х годов на основе Си++. Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка - компиляция не в машинный код, а в платформо-независимый байт-код. Этот байт-код может выполнятся с помощью интерпритатора-виртуальной машины Java-машины JVM (Java Virtyal Machine), версии которой созданы сегодня для любых платформ. Благодаря наличию Java-машин программы на Java можно переносить на не только на уровне исходных текстов, но и на уровне обычного байт-кода, поэтому по популярности язык Ява сегодня занимает второе место в мире после Бейсика.