## §2. ЯЗЫКИ ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

### 2.1. 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, словарный запас, свои грамматику и синтаксис, а также семантику.

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

**Синтаксис** - система правил, определяющих допустимые конструкции языка программирования из букв алфавита.

**Семантика** - система правил однозначного толкования отдельных языковыхконструкций, позволяющих воспроизвести процесс обработки данных.

При описании языка и его применении используют понятия языка. **Понятие** подразумевает некоторую синтаксическую конструкцию и определяемые ею свойства программных объектов или процесса обработки данных.

Взаимодействие синтаксических и семантических правил определяют 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, называют языками «высокого уровня». В настоящее время насчитывается несколько сотен таких языков, а если считать и их диалекты, то это число возрастет до нескольких тысяч. Языки программирования высокого уровня существенно отличаются от машинно-ориентированных (низкого уровня) языков. Во-первых, машинная программа в конечном счете записывается с помощью лишь двух символов 0 и 1. Во-вторых, каждая ЭВМ имеет ограниченный набор машинных операций, ориентированных на структуру процессора. Как правило, этот набор состоит из сравнительно небольшого числа простейших операций, типа: переслать число в ячейку; считать число из ячейки; увеличить содержимое ячейки на +1 и т.п. Команда на машинном языке содержит очень ограниченный объем информации, поэтому она обычно определяет простейший обмен содержимого ячеек памяти, элементарные арифметические и логические операции. Команда содержит код и адреса ячеек, с содержимым которой выполняется закодированное действие.

Языки программирования высокого уровня имеют следующие достоинства:

• 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 действия сними;

• 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.

### 2.2. МЕТАЯЗЫКИ ОПИСАНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Интерпретация конструкций языка программирования должна быть абсолютно однозначной, ибо фраза на языке программирования превращается в машинный код автоматически, с помощью программы-транслятора, и любой намек на неоднозначность либо делает эту фразу непереводимой, либо приводит к ошибке. В этом отношении языки программирования значительно отличаются от естественных языков, допускающих неоднозначно интерпретируемые фразы, рассчитанные на здравый смысл и жизненный опыт человека - слушателя и исполнителя, способного додумать содержание фразы. «Додумывание» не входит в способности компьютеров, поэтому необходимы приемы описания конструкций языков программирования типа: «Оператором присваивания называется ...», причем продолжение подобной фразы на естественном языке чаще всего оказывается либо слишком громоздким, либо неоднозначным, либо и тем, и другим одновременно.

Для строгого и точного описания синтаксиса языкапрограммирования, как правило, используют специальные **метаязыки** (языки для описания других языков). Наиболее распространенными метаязыками являются **металингвистические формулы Бэкуса - Наура** (язык БНФ) и **синтаксические диаграммы Вирта.**

**Язык БНФ** (называемый также языком нормальных форм) представляет компактную форму в виде некоторых формул, похожих на математические. Для каждого понятия языка существует единственная метаформула (нормальная форма). Она состоит из левой и правой частей. В левой части указывается определяемое понятие, а в правой - задается множество допустимых конструкций языка, которые объединяются в это понятие. В формуле используют специальные метасимволы в виде угловых скобок, в которых заключено определяемое понятие (в левой части формулы) или ранее определенное понятие (в ее правой части), а разделение левой и правой частей указывается метасимволом **«::=»,** смысл которого эквивалентен словам «по определению есть».

Например, метаформулы

<переменная>::=А]В

<выражение>::=<переменная>|<переменная>+<переменная>|<переменная><переменная>-<переменная>

означают, что в том (сугубо модельном) языке, на который эта метаформула распространяется, под термином <переменная> понимается любая из букв А или В, а под термином <выражение> - любая из следующих десяти записей: А; В; А+А; А+В; В+А; В+В; А-А; А-В: В-А; В-В Знак 1 следует читать «или».

Правая часть метаформулы может содержать правило построения допустимых последовательностей. Допускаются рекурсивные определения терминов и понятий, т.е. когда в правой части формулы участвует понятие, определяемое левой частью. Например, пусть необходимо ввести понятие <двоичный код>, под которым понимался любая непустая последовательность цифр 0 и 1. Тогда простое и компактное рекурсивное определение с помощью метаформул выглядит так:

<двоичная цифра>::= 0|1

<двоичный код>::=<двоичная цифра>|<двоичный код> <двоичная цифра>

Рекурсия здесь не мешает конструктивному построению понятия <двоичный код>, так как по принятым правилам при первом обращении к рекурсивно определяемому понятию следует ограничиться нерекурсивной частью формулы, т.е. под двоичным кодом понимать двоичную цифру - 0 или 1. Но при втором обращении к метаформуле, определяющей двоичный код, мы имеем варианты (конечно, неполные) понятия <двоичный код>, и можем применить рекурсию, которая даст нам следующие варианты этого понятия: 0 1 00 01 10 11, т.е. все возможные одно- и двухцифровые двоичные коды. Очевидно, что при следующих применениях рекурсии мы получим любой возможный двоичный код.

Для задания синтаксических конструкций произвольной длины часто используют фигурные скобки как метасимволы. Фигурные скобки означают, что конструкция может повторяться нуль или более раз. В частности, термин <двоичный код> можно определить по другому, а именно:

<двоичный код>::=<двоичная цифра><двоичная цифра>

И еще, для полноты множества синтаксических конструкций, необходимо определить конструкцию <пусто>:

**<пусто>::=.**

В большинстве учебных пособий по программированию, технических описаний языков, метаформулы рассматриваемого языка представлены полностью.

**Синтаксическая диаграмма** является графическим представлением значения ме-тапеременной метаязыка. Диаграмма состоит из основных символов или понятий языка.

Каждая диаграмма имеет входящую и выходящую стрелки, означающие начало и конец синтаксической конструкции и отражающие процесс ее чтения и анализа . Из каждого элемента выходит одна или несколько стрелок, оказывающих на те элементы, которые могут следовать непосредственно за данным элементом.

Для сравнения с метаформулами приведем несколько примеров.

Синтаксическая диаграмма

*<переменная>:: =*



*цсвивалентна метаформуле <переменная>::= А\В.*

Еще примеры:



Читатель может поупражняться в составлении синтаксических диаграмм для известных ему языков программирования.

Металингвистические формулы в некотором виде заложены в трансляторы; с их помощью ведется проверка конструкций, используемых программистом, на формальное соответствие какой-нибудьиз конструкций, синтаксически допустимых в этом языке (синтаксический контроль).

### 23. ГРАММАТИКА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Описанию грамматики языка предшествует описание его алфавита. Алфавит любого языка состоит из фиксированного набора символов, однозначно трактуемых. Алфавит языков программирования, как правило, связан с литерами клавиатуры печатной машинки. Клавиатуры персональных компьютеров близки к ним по наличию литер.

Алфавиты большинства языков программирования близки друг другу и основываются на буквах латинского алфавита, арабских цифрах и общепринятых спецсимволах, таких как знаки препинания, математических операций, сравнений и обозначений. Большинство популярных языков программирования в своем алфавите содержат следующие элементы:

<буква> : : = AaBbCcDdEeFf и т.д.

<цифра> ::=0123456789

<знак арифметической операции >::=\*/+-

<разделитель> ::=.,;:()[]{}':=

<служебное слово> :: = begin end if then else for next и т.д.

<спецсимвол> :: = <знак арифметической операции> | <разделитель> |

<служебное слово>

<основной символ>::=<буква> | <цифра> | <спецсимвол>

<комментарий>::=<любая последовательность символов>

Несмотря на значительные различия между языками программирования, ряд фундаментальных понятий в большинстве из них схожи. Приведем часть этих понятии.

**Оператор** - одно из ведущих понятий всех языков программирования (теоретически, за исключением чисто декларативных; но в действительности и они используют родственное понятие). Каждый оператор представляет собой законченную фразу языка и определяет однозначно трактуемый этап обработки данных В соответствии с теорией алгоритмов выделяют основные (базисные) операторы языка: присвоения, условный и безусловный переход, пустой оператор. К производным, не основным, относят составной оператор, оператор выбора, оператор цикла и оператор присоединения.

Все операторы языка в тексте программы отделяются друг от друга явными или неявными разделителями, например:

Sl;S2;...;Sn

Операторы выполняются в порядке их следования в тексте программы. Лишь с помощью операторов перехода этот естественный порядок может быть нарушен.

Большая часть операторов ведет обработку величин. **Величины** могут быть **постоянными и переменными.** Значения постоянных величин не изменяются в ходе выполнения программы. Величина характеризуется **типом, именем и значением.** Наиболее распространенные типы величин - числовые (целые и вещественные), символьные, логические. Тип величины определяется ее значением.

Другая важная классификация величин - простые и структурированные. Простая величина в каждый момент может иметь не более одного значения. Ей соответствует одна ячейка памяти (поскольку термин «ячейка» несколько устарел, часто говорят «машинное слово») или ее эквивалент во внешней памяти компьютера. Структурированная величина, имея одно имя, может иметь разом несколько значений. Эти значения представляют собой элементы (компоненты) величины. Самый широкоизвестный пример - массив, у которого элементы различаются по индексам (номерам). Вопрос о структурировании величин - входных, выходных и промежуточных - для успеха решения прикладной задачи не менее важен, чем вопрос о правильном написании последовательности операторов.

Важнейшие характеристики структурированной величины таковы: **упорядоченность** (да или нет), **однородность** (да или нет), **способ доступа** к элементам, **фиксированность числа элементов** (да или нет). Так, массив является упорядоченной однородной структурой с прямым доступом к элементам и фиксированным их количеством.

Всем программным объектам в языках даются индивидуальные **имена.** Имя программного объекта называют **идентификатором** (от слова «идентифицировать»). Чаще всего идентификатором является любая конечная последовательность букв к цифр, начинающаяся с буквы:

*<идентификатор>::=<буква> | <идентификатор> | <буква>*

 *<идентификатор><цифра>*

Как правило, в большинстве языков программированияв качестве идентификатора запрещается использовать служебные слова языка.

Многим слово «идентификатор» не нравится, и в настоящее время чаще употребляют слово «имя», поскольку

<имя>::=<идентификатор>.

Программисты выбирают имена по своему усмотрению. Принципы выбора и назначения имен программным объектам естественны. Следует избегать мало выразительных обозначений, не гоняться за краткими именами. Имена должны быть понятны, наглядны, отражать суть обозначаемого объекта. Например,

*Summa, Time, i, j, integral, init и т. п.*

Некоторым идентификаторам заранее предписан определенный смысл и их называют стантартными, например, Sin - это имя известной математической функции.

**Описания** или объявления программных объектов связаны с правилами обработки данных. Данные бывают разные и необходимо для каждого из них определить его свойства. Например, если в качестве данных выступает массив, то необходимо задать его размерность, границы индексов, тип элементов массива. Описательная часть языка программирования является необходимой как для системных программистов - разработчиков трансляторов, которые должны, в частности, проводить синтаксическую и семантическую диагностику программ, - так и для «прикладного» программиста, которому объявления программныхобъектов частооблегчают процесс разработки и отладки программ.

В некоторых языках стандартные описания простых числовых и символьных данных опускают (описания по умолчанию), или в них задаются правила описания по имени объекта. Например, в Фортране переменные, имена которых начинаются с букв I, J, К, L, M, N, могут принимать целые значения (при отсутствии явного описания типа, которое возможно), т.е. определены как числовые данные целого типа. В Бейсике-MSX данные строкового типа присваиваются переменным, имена которых заканчиваются специальным символом $: A$, S1$.

Особый интерес представляют в языках программирования описания нестандартных структур данных, таких как запись, файл, объект, список, дерево и т.п.

Приведем список наиболее употребительных обозначений типов данных, используемых в описаниях:

*Целый - Integer*

*Вещественный - Real*

*Логический - Boolean*

*Символьный - Char*

*Строковый - String*

*Массив - Array*

*Множество -Set*

*Файл - File*

*Запись - Record*

*Объект - Object*

**Переменные** играют важнейшую роль в системах программирования. Понятие «переменная» в языках программирования отличается от общепринятого в математике. Переменная - это программный объект, способный принимать некоторое значение с помощью оператора присваивания. В ходе выполнения программы значения переменной могут неоднократно изменяться. Каждая переменная после ее описания отождествляется с некоторой ячейкой памяти, содержимое которой является ее значением. Синтаксис переменной, точнее, ее идентификатора, как правило, имеет вид:

*<имя переменной>::=*

*——><буква>*———*>*

—*><буква>*———*>*

—*><цифра>*——*>*

—*><спецсимвол>*

Семантический смысл переменной заключается в хранении некоторого значения. соответствующего ее типу (например, переменная целого типа может принимать значение произвольного целого числа), а также в выполнениис ней операций пересылки в нее и извлечения из нее этого значения.

**Функция** - это программный объект, задающий вычислительную процедуру определения значения, зависимого от некоторых аргументов. Вводится в языки программирования для задания программистом необходимых ему функциональных зависимостей. В каждом языке высокого уровня имеется в наличии библиотека стандартных функций: арифметических, логических, символьных, файловых и т.п. Функции -стандартные и задаваемые программистом - используются в программе в выражениях.

Выражения строятся из величин - постоянных и переменных, функций, скобок. знаков операций и т.д. Выражение имеет определенный тип, определяемый типом принимаемых в итоге его вычисления значений. Возможны выражения арифметические, принимающие числовые значения, логические, символьные, строковые и т.д. Выражение 5+7 является, несомненно, арифметическим, выражение А + В может иметь cамый разный смысл - в зависимости от того, что стоит за идентификаторами А и В.

**Процедура** - это программный объект, представляющий некоторый самостоятельный этап обработки данных. По сути, процедуры явились преемниками подпрограмм, которые были введены для облегчения разработки программ еще на самых ранних стадиях формирования алгоритмических языков. Процедура имеет входные и выходные параметры, называемые формальными. При использовании процедуры формальные параметры заменяются на фактические.

**Модуль** (Unit) -это специальная программная единица, предназначенная для создания библиотек и разделения больших программ на логически связанные блоки.

По сути, модуль - это набор констант, типов данных, переменных, процедур и функций. В состав модуля входят разделы: заголовок, интерфейс, реализация, инициализация.

*Заголовок* необходим для ссылок на модуль.

*Интерфейс* содержит объявления, включая процедуры и функции.

Раздел *«реализация»* содержит тела процедур и функций, перечисленных в интерфейсной части.

Раздел *«инициализация»*содержит операторы, необходимые для инициализации модуля.

Каждый модуль компилируется отдельно, и каждый элемент модуля можно использовать в программе без дополнительного объявления.