2. Языки программирования.

Что такое язык программирования? Это язык, на котором можно разговаривать с компьютером. Диалог осуществляется с помощью специальных команд, понятных для компьютера. Существует два принципиальных типа языков программирования — *низкого* и *высокого* уровня. Низкий уровень программирования включает машинный язык и язык ассемблера. Команды, инструкции на этих языках по смыслу понятны компьютеру непосредственно. Языки программирования высокого уровня более близки к естественному, писать программы на них более удобно человеку, но, прежде чем они станут понятны для компьютера, они должны быть предварительно расшифрованы, «переведены» на низкий, машинный язык.

Каждый естественный язык обладает синтаксисом и семантикой. Каждый язык программирования обладает тремя основными характеристиками:

Синтаксис определяет, какие структуры фраз и предложений являются легальными, дозволенными. Таким образом, синтаксис определяет форму, но ничего не может сказать о смысле, содержании структурных частей. Это грамматика языка.

Второй компонент – это семантика, содержание. Это фундаментальная составляющая языка. Если провести сравнение с естественными языками, там также можно построить синтаксически правильное предложение, но совершенно лишенное смысла. Без семантики язык программирования – это просто набор бессмысленных фраз.

И, наконец, прагматизм. Как и в естественных языках, в языках программирования существуют идиомы (алгоритмические) «обороты», подходы к построению предложения, которые помогают эффективно выразить мысль. Эти идиомы приобретаются с практикой и опытом.

К сожалению, из-за того, что синтаксис языка – это первое, с чем сталкивается начинающий программист, то часто этот аспект воспринимается как важнейший. Но такой подход всегда замедляет понимание основ и принципов программирования. Некоторые считают хорошей практикой изучать несколько языков одновременно и писать одни и те же программы на них. Это помогает выявлять преи-

мущества и недостатки того или иного языка и их относительную гибкость и пригодность для решения различного типа проблемных задач. В данной книге мы начнем изучать язык программирования высокого уровня Турбо-Паскаль, диалект языка Паскаль, который был сконструирован специально для целей обучения и обладает, по мнению автора, самым красивым, удобным и наглядным синтаксисом. Однако мы старались построить курс таким образом, чтобы не «тяготиться» лимитами синтаксиса.

В этой главе приводится короткий анализ языков программирования различного уровня.

2.1. Машинный язык.

Как уже было отмечено, все компьютеры понимают внутренний машинный язык, инструкции на котором они могут выполнить. Этот язык закодирован в двоичном представлении и писать программы на нем очень неудобно и утомительно, так как и данные и инструкции выражаются двоичными числами. Большинство инструкций состоят из частей кода оператора и адреса. Код оператора определяет, какая именно операция должна быть реализована, тогда как код адреса инструкции указывает, какое местоположение используется в качестве операнда инструкции. Например, последовательность байтов программы может содержать:

Код операции	Адрес	Значение
00010101	10100001	Загрузить с(129) в аккумулятор
00010111	10100010	Добавить с(130) к аккумулятору
00010111	10100011	Сохранить с (акк) в локации 131

где с( ) означает «содержимое от ( )», а аккумулятор — это специальный регистр центрального процессора. Эта последовательность кода добавляет содержимое локации 130 к содержимому аккумулятора, которое предварительно было заполнено содержимым локации 129, а затем сохраняет результат в локации 131. Только ЦП способен различить, какие последовательности битов представляют данные, а какие — инструкции.

Используя машинный язык, программист должен очень внимательно определять, какие из локаций он использует для сохранения данных, а какие из них формируют выполняемые инструкции. Очень сложно исправлять ошибки, когда происходит замещение данных инструкциями или когда программа пытается запустить на выполнение часть своих данных. Однако способность интерпретировать один и тот же образец битов и как данные, и как инструкции, является мощным свойством, предоставляющим программам возможность генерировать другие программы и запускать их на выполнение.

2.2. Язык ассемблера.

Учет использования системных ресурсов при программировании на машинном языке очень утомителен. Если программист должен модифицировать программу и добавить новый элемент данных, то это может повлечь за собой изменение адресов других данных, и программист должен очень осторожно проверить всю программу, выявляя адреса битов, подлежащих модификации, и изменить их.

Человек, в противоположность компьютерам, не очень хорошо выполняет простую рутинную работу, требующую повторений. *Языки ассемблера* (assembly language) являются более дружественной формой машинного языка. Команды машинного языка взаимооднозначно замещаются мнемоническими командами. Программа ассемблера обеспечивает преобразование мнемонических кодов на коды машинного языка. Программист может использовать также символьные адреса для данных – ассемблер сам присвоит машинные адреса и проверит, не перекрываются ли различные элементы данных в памяти, что очень часто возникает в программах на машинном языке. Например, фрагмент программы, приведенный выше, на языке ассемблера может быть переписан в виде:

Код операции	Адрес
LOAD	A
ADD	B
STORE	C

Очевидно, что такой подход оставляет меньше возможностей для ошибок. Так как компьютер непосредственно не воспринимает язык ассемблера, то код программы должен быть переведен на машинный язык специальной (компонирующей) программой, называемой ассемблером (assembler). Она замещает коды мнемонических операций, такие, как LOAD, ADD и др. соответствующими двоичными кодами и распределяет адреса памяти для всех символьных переменных, которые используются программой. Ассемблер присоединяет к символам A, B и C соответствующие адреса и проверяет, что они все различны.

Таким образом, в попытке облегчить человеку процесс программирования, появился новый уровень обработки информации на компьютере. Языки ассемблера до сих пор используются в некоторых программах, где временной фактор является важным, так как они дают программисту возможность непосредственно контролировать, что происходит внутри компьютера. Языки ассемблера все еще подразумевают знание программистом внутренней структуры компьютера. Например, для различных типов данных требуются различные инструкции ADD. В языках ассемблера все еще присутствует машинная специфичность и, следовательно, чтобы программу, написанную для одного типа компьютера, можно было бы использовать для другого типа, ее нужно переписать заново.

2.3. Языки программирования высокого уровня.

На протяжении развития компьютеров постоянно предпринимались попытки облегчить процесс программирования, чтобы для составления программы конкретного назначения требовалось как можно меньше знания внутренней конструкции компьютера. Если программы можно было бы представить на языке, близком к человеческому, то возникало бы меньше ошибок. Языки программирования высокого уровня позволяют представить решение проблемы в терминах, более близких к тем, которые используются человеком в процессе решения задач. Эти языки проектировались с тем, чтобы сделать программирование более легким, избежать ошибок и избавить программиста от детального знания внутренней структуры конкретного компьютера. Эти языки гораздо ближе к человеческому. Одним из первых языков был Фортран (Fortran),

который был разработан в 1957 году. На этом языке вышеприведенная программа запишется в виде:

C=A+B,

что явно более читабельно, легко модифицируемо и избежать ошибок гораздо легче.

Как и в случае языков ассемблера, компьютер непосредственно не понимает языков высокого уровня и, следовательно, они должны быть обработаны специальной программой, которая называется *компилятором.* Она обеспечивает перевод текста программы с языка высокого уровня на внутренний, машинный, язык перед тем, как она будет запущена на выполнение.

Другое преимущество языков высокого уровня появляется при их некоторой стандартизации. Тогда каждый производитель компьютеров может разработать компилятор для *компилирования* стандартных программ на внутренний машинный язык собственного компьютера. Это обстоятельство дает возможность, написав программу на стандартном языке, запускать ее на компьютерах с различным внутренним языком с помощью соответствующих компиляторов. Это довольно важное преимущество было достигнуто для некоторых языков высокого уровня.

Как ассемблер, так и языки высокого уровня экономят время программиста и пользователя за счет увеличения времени вычислений, требуемого для трансляции программы на внутренний машинный язык. Если сравнить эти два параметра, то секунды времени вычислений равноценны неделям сбереженного человеческого времени.

После Фортрана появилось (и исчезло) много языков программирования высокого уровня. Важнейшими и наиболее популярными из них являются:

Процедуральные и объектно-ориентированные языки программирования

Название	Характеристики языка
FORTRAN	(FORmula TRANSlator), 1957. Для научных и инженерных прикладных задач.
ALGOL	(ALGOrithmic Language), 1958. Для наглядного представления алгоритмов.
COBOL	(COmmon Business-Oriented Language), 1960. Общего назначения, для прикладных задач из сферы бизнеса.
BASIC	(the Beginners' All-purpose Symbolic Instruction Code). Один из первых языков для систем с разделением времени.
PASCAL	(В честь Блейза Паскаля), 1971. Общего и обучающего назначения.
ADA	(В честь дочери Байрона), 1980. Для прикладных задач управления.
C	1972. Общего назначения, наиболее популярный в настоящее время.
C++	1980. Объектно-ориентированный язык программирования.
HTML	(HyperText Mark-up Language) Для создания файлов со связями (Гипер-тексты), используется в базах Интернета.
Java	Для программирования в Интернете.

Функциональные и логические языки программирования

Название	Характеристики языка
LISP	(LISt Processing), 1958. Язык функционального программирования для обработки символьной информации.
Scheme, Haskel	Языки функционального программирования.
PROLOG	(PROgramming in LOGic), 1970. Язык логического программирования. Искусственный интеллект.
FRIL	(Fuzzy Relational Inference Language — Язык нечетких логических выводов). Язык нечеткого логического программирования. Искусственный интеллект.

Все эти языки доступны на многих компьютерах различного типа.

Важно отметить принципиальное различие между Прологом (и другими языками логического программирования (ЛП) для искусственного интеллекта) и языками высокого уровня (такими, как Фортран, Паскаль и др.). Если в последнем случае улучшение в процессе программирования носит скорее количественный характер (экономия времени пользователя), то языки логического программирования являются качественно новой ступенью. Чтобы написать программу, например, на Турбо-Паскале, сначала задачу (после формализации) нужно решить вручную, затем уже алгоритм перевести на Турбо-Паскаль. В языках логического программирования предпринимается попытка имитировать на компьютере процесс решения проблемы человеком. В общем случае, при ментальной постановке задачи, имеется набор исходных аксиом, набор правил, позволяющих делать из них определенные выводы, и цель (или несколько целей), которые можно вывести (доказать) из аксиом с помощью правил. Этот механизм можно реализовать на компьютере на основе логического подхода к программированию, т.е.

использовав аппарат логических вычислений. Здесь цель воспринимается как теорема, которая доказывается на основе предварительно сформулированных аксиом и правил. Отметим, что логический подход более эффективен для задач, которые трудно или невозможно формализовать и описать с помощью численных параметров.