17. Языки программирования их развитие

Язык программирования – формальная знаковая система, предназначенная для описания команд (инструкций) и данных, которые могут быть обработаны средствами ЭВМ.

Языки программирования являются формальными, или искусственными языками, обеспечивающие взаимодействие ЭВМ и пользователя, и предназначены для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т. п. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

Со времени создания первых программируемых машин человечество придумало более восьми тысяч **языков программирования**. Каждый год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты могут владеть несколькими языками программирования

Формальный язык определен формальной грамматикой. Согласно А.Н. Хомскому[1], формальные языки классифицируются в соответствии с типами грамматик, которыми они задаются. Языки программирования определяются контекстно-свободными грамматиками при условии, что символами алфавита являются токены (объекты, создающиеся из лексем[2] в процессе лексического анализа), образованные по правилам регулярной грамматики.

Естественные языки определяются грамматиками общего вида и в этом случае отличаются от языков программирования, хотя языки программирования высокого уровня внешне похожи на естественные языки.

Как и все языки, языки программирования имеют собственный алфавит, **синтаксис и семантику.**

[1] Авраам Ноам Хомский – американский лингвист, автор классификации формальных языков (иерархия Хомского). Предложил четыре типа формальных грамматик: неограниченные, контекстно-зависимые, контекстно-свободные и регулярные (самые простые).

[2] Лексема – минимальная смысловая единица для языка программирования (например: константа, ключевые слова и т.п.). В прикладном программировании токены и лексемы могут не различаться.

Алфавит – конечный набор символов для конкретного языка программирования.

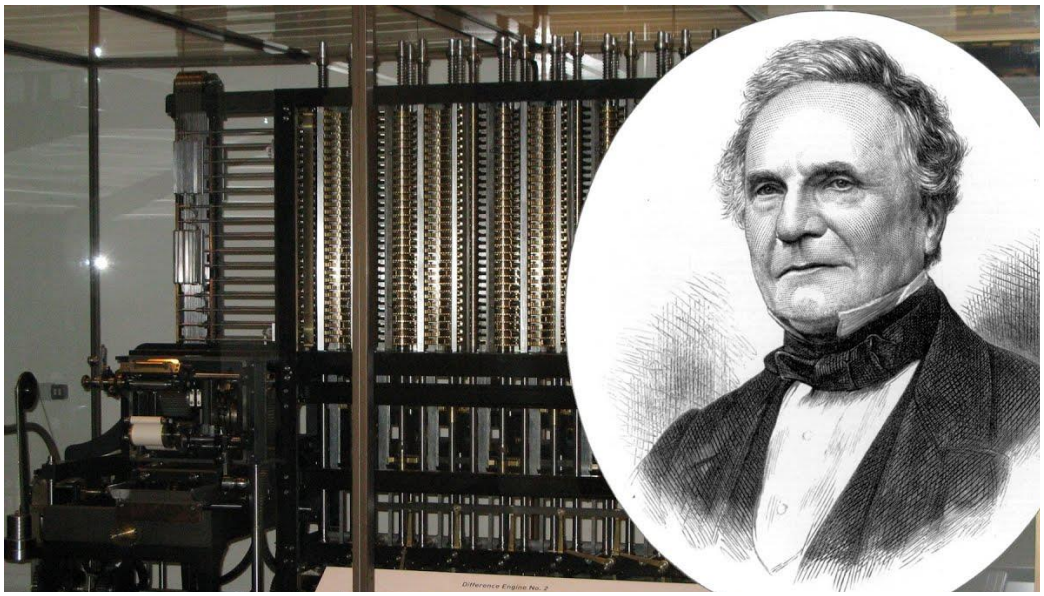
Синтаксис определяет правила образования токенов и правила формирования последовательностей токенов.

Семантика – формальное содержание (смысл) последовательности токенов.

Если в формальных языках построенная фраза грамматически правильна, то это означает, что семантически она тоже правильна.

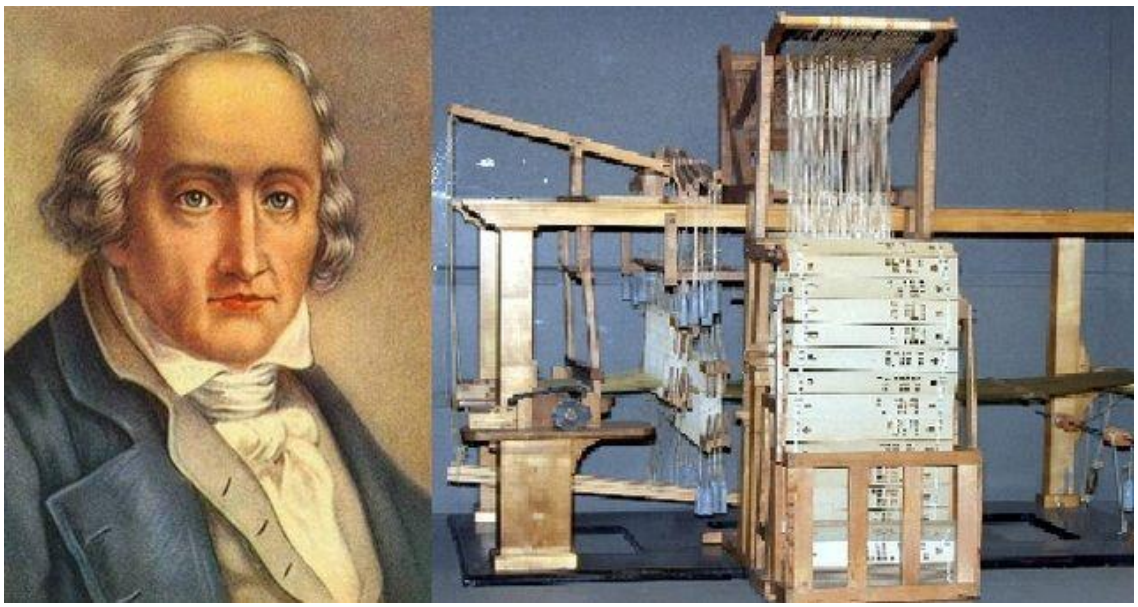
Развитие языков программирования неразрывно связано с развитием средств вычислительной техники и определенными задачами, которые необходимо было решать.

Основоположником программирования считают английского математика **Чарльза Беббиджа** (1791-1871), который изобрел первую в мире аналитическую вычислительную машину (прототип современных ЭВМ), состоящую из валиков и шестерней, приводящиеся во вращение с помощью рычага.

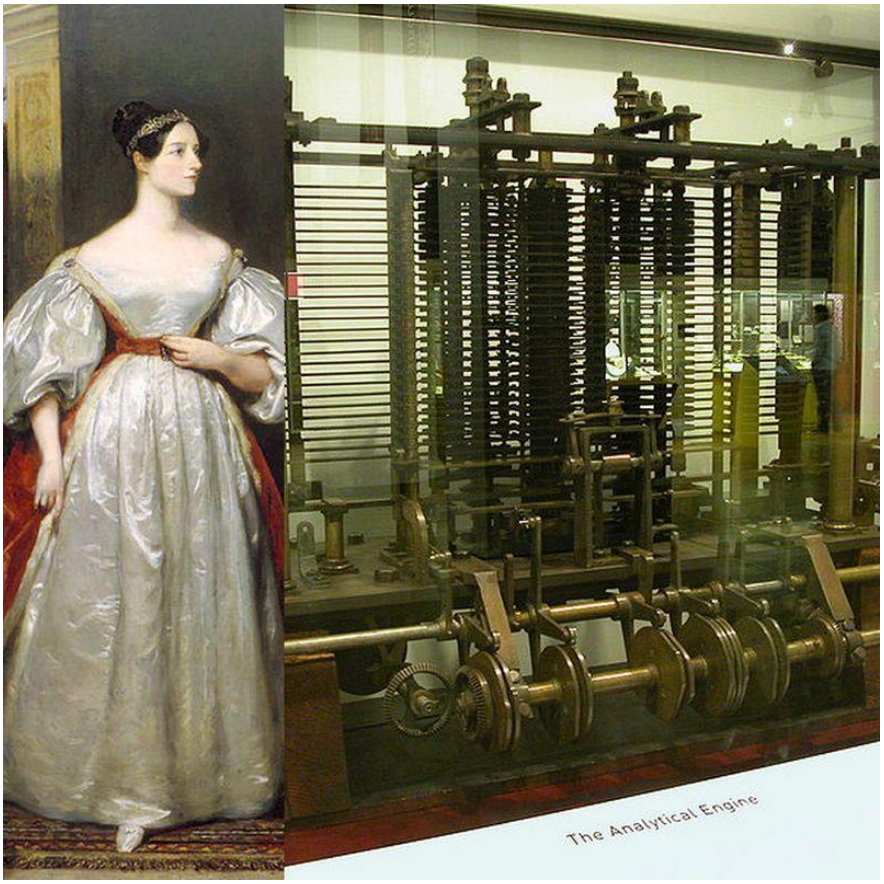


Принципы работы машины были основаны на вычислении таблиц разностным методом. При этом использовалась только операция сложения.

Следующий шаг в области программирования сделал **Жозеф Мари Жаккар** (1752-1854), который стал использовать перфокарты для алгоритмизации действий ткацких станков.



Первым в мире программистом мировое сообщество считает **Аду Лавлейс** (1915-1852)- друга и соратника Ч. Беббиджа, которая разработала описание большой разностной вычислительной машины[3], составила для нее первую программу, а также ввела такие термины, как «рабочая ячейка» и «цикл».



[3] Полностью собранную большую разностную машину удалось сконструировать только в 1906 г. компании Monroe Calculating Machine Co/

Первая программа включала алгоритм вычисления чисел Бернулли, и ее текст был опубликован в комментариях к запискам о вычислительной машине итальянского инженера Луиджи Манабреа. Для того чтобы осуществить программирование таких машин, использовались особые комбинации цифр, которые «понимала» бы только данная машина. Эти цифровые комбинации назвали машинными кодами. Процесс программирования был очень сложным и трудоемким делом, которое могли освоить только отдельные специалисты.

В конце 1940-х гг., одновременно с созданием первых ЭВМ, появились языки программирования первого поколения, которые относятся к машинно-ориентированной парадигмы программирования[4]. Они представляли собой машинные инструкции-коды в двоичной системе счисления. Программисту необходимо было знать уникальные для каждой машины инструкции (набор команд), чтобы правильной реализовать алгоритм решения задачи на данной машине.

В 1949 г. появляется машинно-ориентированный язык Ассемблер (от анг. assembler – сборщик), команды которого строго соответствовали командам машины.

Программирование стало легче за счет введения символьной формы записи команд – мнемокодов. Язык Ассемблера активно применялся в эпоху ЭВМ третьего поколения.

Язык Ассемблера имеет различные вариации. Из них наиболее известны Ассемблер A86, Microsoft Macro Assembler (MASM), Borland Turbo Assembler (TASM), Watcom Assembler (WASM).

Языку Ассемблера соответствовала своя среда разработки (рис.17.1)
В том же 1949 г. сотрудник Пенсильванского университета США Джон Моучли, участвуя в проекте создания первого компьютера ЭНИАК и на начальном этапе проекта создания более совершенного компьютера EDVAC, разработал особую систему кодирования машинных команд Short Code, которые вводились непосредственно в память машины. Short Code – это интерпретатор – переводчик с языка программирования на машинный код. Интерпретатор использовал примитивный язык программирования высокого уровня. Программист, применяя этот язык, писал задачу математическими формулами, затем пользовался специальной таблицей, где каждому символу

[4] Термин «парадигма программирования» был введен в 1978 г. Робертом Флойдом. Парадигма программирования – это подход к программированию, включающий совокупность идей и понятий, определяющих стиль написания компьютерных программ. Машинно-ориентированную парадигму иногда называют допарадигмальной. Некоторые современные языки программирования поддерживают реализацию нескольких парадигм.

формул соответствовали двухлитерные коды. Завершающим этапом работы было выполнение специальной программы компьютера, которая переводила

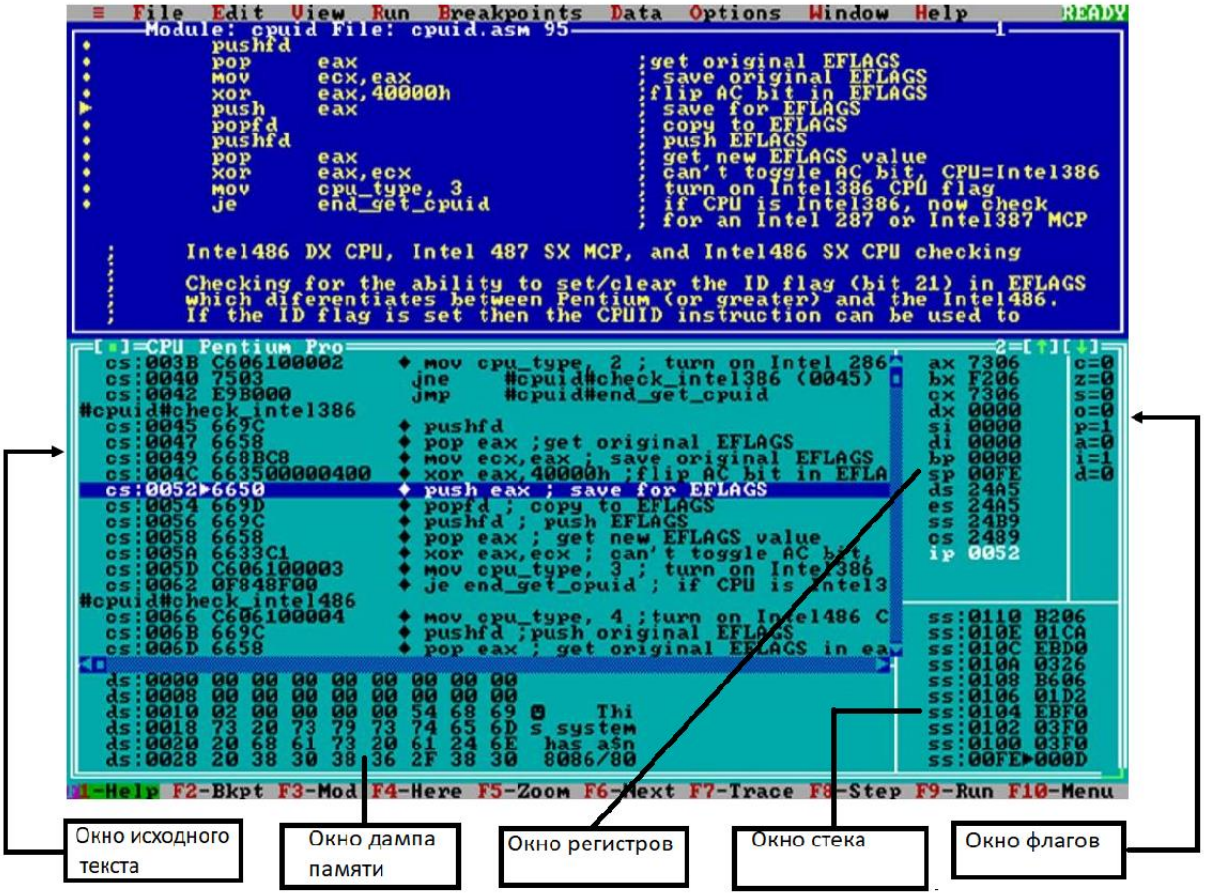


Рис. 17.1 Среда разработки языкаАссемблер

двухлитерные коды в двоичные машинные коды. Таким образом, уже в то время программисты заставили компьютер переводить информацию с языка программирования на машинный код. В настоящее время система Джона Моучли Short Code считается началом развития современных языков программирования, реализующих процедурную парадигму[5].

В 1951 г. Грейс Хоппер (1906-1992), которая была сотрудницей компании Джона Моучли, создала первый компилятор для компьютера МАРК-II, который представлял собой усовершенствованный вычислитель МАРК-I, в основе работы которого использовались электромеханические реле и переключатели (рис. 17.2).

Компилятором Хоппер осуществлялась функция объединения команд. Компилятор выделял память компьютера, преобразовывал команды высокого уровня (в то время – псевдокоды) в комплекс машинных команд.

[5] Процедурная (императивная, директивная, модульная) парадигма – одна из вычислительных парадигм, реализуемых в языках программирования..

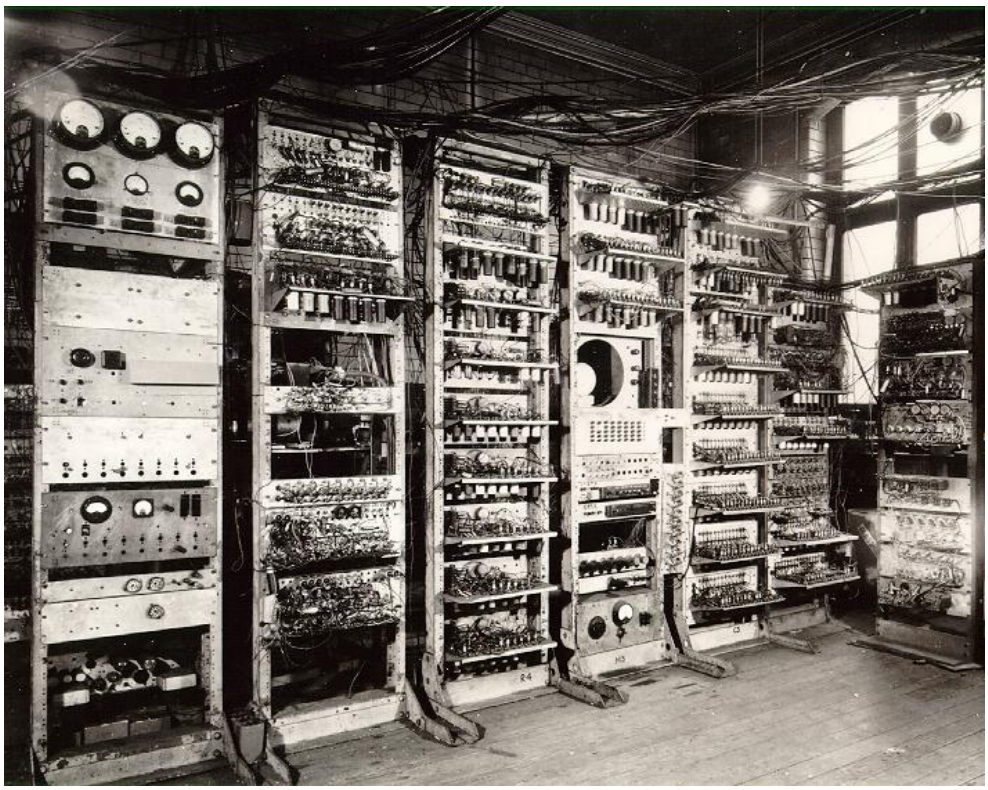


Рис. 17.2. Первый программируемый компьютер МАРК-I.

С середины 1950-х гг. в области программирования начинаются большие изменения. Специалисты все больше переходят к разработке и использованию языков – посредников между программистом и машиной, отдавая предпочтение **императивной парадигме программирования**.

Императивное программирование – это парадигма программирования, которая имеет следующие особенности. Исходный код программы содержит команды (инструкции), которые выполняются последовательно. Данные, необходимые для выполнения команды, могут читаться из памяти, в которую они были занесены при выполнении предыдущей инструкции.

К наиболее известным языкам программирования, реализующим императивную парадигму, относят языки **FORTRAN, ALGOL, PL/1, BASIC, COBOL**.

Язык FORTRAN. Первый язык программирования высокого уровня, получивший практическое применение, имеющий транслятор и получивший дальнейшее развитие. Создан в период с 1954 по 1957 год группой программистов под руководством Джона Бэкуса в корпорации IBM. Название Fortran является сокращением от **FOR**mula **TRAN**slator (формульный транслятор). Фортран широко используется в первую очередь для научных и инженерных вычислений. Одно из преимуществ современного

Фортрана — большое количество написанных на нём программ и библиотек подпрограмм. На рис. 17.3 приведен фрагмент программы на языке Fortran.

```
      PROGRAM JAC_F77
      PARAMETER (L=8, ITMAX=20)
      REAL A(L,L), B(L,L)
      PRINT*,*******TEST_JACOBI********

      DO IT=1,ITMAX
        DO J=2, L-1
          DO I=2, L-1
            A(I,J)=B(I,J)
          ENDDO
        ENDDO
        DO J=2, L-1
            DO I=2,L-1
              B(I,J)=(A(I-1,J)+A(I,J-1)+
     *         A(I+1,J)+A(I,J+1))/4
           ENDDO
          ENDDO
        ENDDO
        END
```

Рис. 17.3 Фрагмент программы на языке Fortran.

Язык ALGOL (Алгóл) (от algorithmic language - «алгоритмический язык») - один из первых языков высокого уровня. Разработан комитетом по языку высокого уровня IFIP в 1958-1960 годах (Алгол 58 Алгол 60). Был популярен в Европе, в том числе в СССР, в качестве языка как практического программирования, так и академического языка (языка публикации алгоритмов в научных работах), но в США и Канаде не смог превзойти распространённый там Fortran. Оказал заметное влияние на все разработанные позднее императивные языки программирования - в частности, на язык **Pascal.** Кардинально переработан в 1964-1968 годах (Алгол 68).

Обычно названием Алгол (без уточнения версии) именуют Алгол 60, в то время как Алгол 68 рассматривается как самостоятельный язык. На рис. 17.4 приведен фрагмент кола на языке Алгол.

```
begin
  procedure p (a, b);
    name a, b; integer a, b;
  begin
    for a:=1 step 1 until 10 do
      b := 0
  end p;
  integer i; integer array s [1:10];
```

```
        p (i, s[i])
    end
```

Рис. 17.4 Фрагмент программы на языке Алгол.

Ярык PL\1. (PL/I, Programming Language I — «язык программирования номер один»,») - разработанный в 1964 году фирмой IBM для научных, инженерных и бизнес-ориентированных вычислений. Содержит широкий набор синтаксических конструкций и встроенных функций, что, не существует не позволило создать компилятор, поддерживающего все предусмотренные в спецификации языка возможности. Поддерживает рекурсию и структурное программирование, широко применялся в обработке данных. Настоящее международное название языка PL/I (из латинских литер, как и PL/M, PL/S) в СССР не прижилось.

Разрабатывался IBM как универсальный язык, превосходящий по возможностям предшественников, но получил лишь ограниченное распространение из-за сложности самого языка и его реализаций. В современном программировании он почти не употребляется.

```
EXAMPLE PROCEDURE OPTIONS (MAIN);
ON ENDFILE (SYSIN) GO TO ENDING;
P1          GET LIST(A,B,C);
            D=B*B-4*A*C;
            E==B/(A+A);
            IF D<0 THEN DO;
            X1,X2=E;
            Y1=SQRT(-D)/(A+A);
            END;
            ELSE DO;
            R=SQRT(D)/(A+A);
            …..
            Y1=0;
            END;
            Y2=-Y1;
            PUTLIST(X1,Y1,X2,Y2);
            GO TO P1;
ENDING;
END EXAMLE;
```

Рис. 17.5 Фрагмент кода на языке PL\1.

Язык BASIC. (**Бе́йсик)** (**BASIC**, сокращение от англ. Beginner’s All-purpose Symbolic Instruction Code — универсальный код символических инструкций для начинающих).

BASIC был разработан в 1964 году профессорами Дартмутского колледжа Томасом Курцем и Джоном Кемени[1] и создавался как инструмент, с помощью которого студенты-непрограммисты могли самостоятельно создавать компьютерные программы для решения своих задач. Получил широкое распространение в виде различных диалектов, прежде всего как язык для домашних компьютеров. К настоящему моменту претерпел существенные изменения, значительно отойдя от характерной для первых версий простоты, граничащей с примитивизмом, и превратившись в достаточно ординарный язык высокого уровня с типичным набором возможностей. Используется как самостоятельный язык для разработки прикладных программ, главным образом, работающих под управлением ОС Windows различных версий. Также широко распространён в качестве встроенного языка прикладных программных систем различного назначения и в качестве языка для программируемых калькуляторов.

После создания базового языка BASIC появилось множество его версий, как, например, Visual Basic, разработанный компанией Microsoft, в основе которого лежал **объектно-ориентированный** подход к программированию. Его основное назначение на первоначальном этапе заключалось в упрощении разработки пользовательских приложений в операционной среде Windows (одна из его версий – язык VBA – Visual Basic Application, предназначенный для офисного программирования, в основном для MS Excel и MS Access)

Начиная с 1950-х годов языки программирования, реализующие императивную парадигму, не могли успешно справляться с существующими на тот момент задачами, связанными, например, с необходимостью проведения быстрых математических вычислений, что привело к развитию другой парадигмы – **декларативной**.

Декларативное программирование – это парадигма программирования, представляющая собой спецификацию задачи, т.е. описание проблемы и ожидаемый результат. Подразделяется на функциональное и логическое программирование. Наиболее известными языками программирования, реализующими декларативное программирование, являются, например, **LISP, LOGO, PROLOG, Scheme, ML, Haskell.**

Язык LISP. (LISP, от англ. LISt Processing — «обработка списков») — семейство языков программирования, основанных на представлении программы системой линейных списков символов, которые притом являются

основной структурой данных языка. Лисп считается вторым после Fortran старейшим высокоуровневым языком программирования.

Язык Лисп был предложен Дж. Маккарти в работе в 1960 году и ориентирован на разработку программ для решения задач не численного характера. Английское название этого языка – LISP является аббревиатурой выражения LISt Processing (обработка списков) и хорошо подчеркивает основную область его применения. Понятие “список” оказалось очень емким. В виде списков удобно представлять алгебраические выражения, графы, элементы конечных групп, множества, правила вывода и многие другие сложные объекты. Списки являются наиболее гибкой формой представления информации в памяти компьютеров. Неудивительно поэтому, что удобный язык, специально предназначенный для обработки списков, быстро завоевал популярность.

После появления Лиспа различными авторами был предложен целый ряд других алгоритмических языков ориентированных на решение задач в области искусственного интеллекта. Однако это не помешало Лиспу остаться наиболее популярным языком для решения таких задач. На протяжении почти сорокалетней истории его существования появился ряд диалектов этого языка: Common LISP, Mac LISP, Inter LISP, Standard LISP и др. Различия между ними не носят принципиального характера и в основном сводятся к несколько отличающемуся набору встроенных функций и некоторой разнице в форме записи программ. Поэтому программист, научившийся работать на одном из них, без труда сможет освоить и любой другой.

Язык LOGO. LOGO - образовательный язык программирования, разработанный в 1967 году Уолли Фейрзейгом, Сеймуром Папертом и Синтией Соломон. LOGO не является аббревиатурой. Название было придумано Фейрзейгом и происходит от греческого logos, означающего слово или мысль.

Язык общего назначения LOGO широко известен своим использованием “черепашьей” графики, в которой команды для перемещения и рисования производят линейную или векторную графику либо на экране, либо с помощью небольшого робота, называемого черепахой. Язык был задуман для обучения концепциям программирования, связанным с LISP.

LOGO обычно является интерпретируемым языком, хотя были разработаны скомпилированные диалекты LOGO (такие как Lhogho и Liogo).

Язык Prolog. Prolog (от “PROgramming in LOGic”) - декларативный язык программирования общего назначения был создан в 1972 с целью сочетать использование логики с представлением знаний. С тех пор у него появился ряд диалектов, расширяющих основу языка различными возможностями.

Prolog — один из старейших и все еще один из наиболее популярных языков логического программирования, хотя он значительно менее популярен, чем основные императивные языки. Он используется в системах обработки естественных языков, исследованиях искусственного интеллекта, экспертных системах, онтологиях и других предметных областях, для которых естественно использование логической парадигмы.

Главной парадигмой, реализованной в языке Prolog, является логическое программирование, основанное на получении логических выводов из имеющихся посылок (“фактов”). Базой логического программирования является аппарат математической логики.

Как и для большинства старых языков, более поздние реализации, например, Visual Prolog, добавляют в язык более поздние парадигмы, например, объектно-ориентированное или управляемое событиями программирование, иногда даже с элементами императивного стиля.

Язык Scheme. Язык Scheme является функциональным языком программирования и был разработан в 1975 г. Гаем Л. Стилом и Джеральдом Джеем Сассманом. Scheme состоит из небольшого стандартного ядра с несколькими инструментами для его расширения. Он имеет много общих характеристик с другими членами семейства языков программирования LISP и считается дальнейшим его развитием.

Семейство языков ML. (Meta Language) - семейство языков программирования, реализующих функциональную парадигму, было представлено в 1973 г. разработчиками из Эдинбургского университета во главе с Робином Милнером. Следует отметить, что ML преподаётся во многих западных университетах (в некоторых даже как первый язык программирования*).*

Язык Haskell. Язык Haskell является строго типизированным функциональным языком, относящийся к семейству ML языков. Сегодня он является одним из самых распространенных языков программирования общего назначения с поддержкой отложенных вычислений. Поскольку язык функциональный, то основная управляющая структура — это функция. Имеются средства взаимодействия с кодом на других языках

программирования. Есть поддержка многозадачного и параллельного программирования, развитый инструментарий (средства автоматического тестирования, отладки и профилирования, в том числе для параллельных программ), существует несколько тысяч библиотек с открытым исходным кодом.

В конце 1960-х, начале 1970-х годов в связи с расширением области применения ЭВМ, увеличением мощности вычислительной техники и появлением высокоуровневых (близких к естественным) языков программирования наметился «кризис программного обеспечения». Это активно обсуждалось на знаменательной международной конференции в 1968 г., на которой Чарльз Хоаэр и Никлаус Вирт высказали идею о необходимости придания языкам программирования свойств, позволяющих обеспечивать создание сложного и надежного программного обеспечения. До этого периода большинство программ были «штучными» продуктами и в программном коде было сложно разобраться, поскольку применение операторов языка, порядок их следования никак не регламентировались. Такой подход был назван позже хаотическим программированием. В том же 1968 г. была опубликована работа Эдсгера Дейкстры «Доводы против оператора Go To», которая положила начало новой парадигмы – **структурной.**

Структурная парадигма основана на следующих принципах:

- Каждый программный блок (модуль, процедура) должен иметь один вход и выход;
- Использование четырех видов конструкций:
  - Последовательность;
  - Ветвление;
  - Цикл;
  - Выбор из нескольких альтернатив.
- Разработка программы осуществляется сверху вниз.

В этот период высокоуровневых языков программирования было совсем немного, однако их число через несколько лет возросло до 3000, но в практической деятельности активно использовались порядка 20-25 языков высокоуровневых языков программирования.

В 1960-1970-х гг. начали появляться **объектно-ориентированные языки программирования**, реализующие соответствующую парадигму.

Объектно-ориентированное программирование (ООП) – парадигма программирования, в основе которой лежит понятие объекта, имеющего определенные свойства, над которым можно осуществлять определенные

действия (применять соответствующие методы), изменяющие его состояние.

Предпосылками создания объектно-ориентированных языков программирования являлось:

- Необходимость разработок сложных проектов;
- Обеспечение взаимосвязи множества встроенных алгоритмов;
- Требование минимизации избыточности данных и обеспечение их целостности;
- Необходимость повышения управляемости процесса разработки программ.

Одним из первых объектно-ориентированных языков программирования стал **Simula67**, разработанный в 1967 г. сотрудниками норвежского вычислительного центра. Simula67 – первый язык, поддерживающий семантику современных языков объектно-ориентированного программирования.

Язык SmallTalk Язык Smalltalk был разработан в 1970-х г одах группой ученых компании Xerox PARC и является объектно-ориентированным языком программирования с динамической типизацией. Представляет собой интегрированную среду разработки и исполнения, объекты которой доступны для модификации через неё саму, и программирование в которой в итоге сводится к модификации её собственного поведения. Язык был представлен как Smalltalk-80.

Smalltalk является одним из многих объектно-ориентированных языков, основанных на языке Simula, который сам оказал большое влияние на развитие таких объектно-ориентированных языков, как: Objective-C, Actor, Java, Erlang, Groovy, Ruby и многих других.

Многие идеи 1980-х и 1990-х по написанию программ появились в сообществе языка Smalltalk. К ним можно отнести рефакторинг, шаблоны проектирования, экстремальное программирование.

Язык Pascal (Паскаль). Язык программирования Pascal (назван в честь выдающегося французского математика и философа Блеза Паскаля (1623 — 1662)), разработан в 1968 — 1971 гг. Никлаусом Виртом, профессором Института информатики Швейцарской высшей политехнической школы. Язык Pascal, созданный первоначально для обучения программированию как систематической дисциплине, скоро стал широко использоваться для разработки программных средств в профессиональном программировании. Его современные модификации,

такие как Object Pascal, широко используются в промышленном программировании (среда Delphi). Также на основе синтаксиса языка Паскаль создан язык программирования Structured Text (ST) или Structured Control Language (SCL) для программируемых логических контроллеров.

На рис. 17.6 приведен пример программы на языке Паскаль

```
Program massiv;
uses  crt;
var
     X : array (1..100) of real;
     i : integer;
     summa: real;
begin
     randomize;
     for i:=1 to 100 do
     begin
       X[i]:=random;
       writeln(X[i]);
       summa:=summa+X[i];
     end;
     writeln;
     writeln(summa);
end.
```

Рис. 17.6 Фрагмент программы на языке Паскаль.

Развитие языка Pascal привело к созданию в 1983 г. языка **Turbo Pascal**, представляющего интегрированную среду для языка программирования Pascal. Для того времени это была потрясающая среда разработки. Она была проста и интуитивно понятна, с хорошо организованным меню. Ранние версии использовали раскладку горячих клавиш WordStar. В более поздних версиях появилась возможность быстро получить определение ключевого слова языка, просто поставив курсор на ключевое слово и нажав клавишу справки. Справочные статьи часто включали примеры кода, использующего данное ключевое слово. Это позволяло неопытным программистам изучать Паскаль даже без помощи книг, используя лишь среду разработки. В версии 5.5 в него были введены передовые возможности объектно-ориентированного программирования. Интегрированная среда так же была реализована с использованием объектно-ориентированного программирования. Последней выпущенной версией была 7.0. Профессиональная поставка Borland Pascal 7.0 включала в себя среду

разработки, компилятор и отладчики для создания программ под реальный режим DOS, а также под Windows 3.x.

Язык Delphi. Delphi[6] — среда программирования на основе языка Object Pascal. Он считается наследником Turbo Pascal, а тот — «чистого» Pascal Никлауса Вирта, созданного в 1970 году.

Изначальный Pascal обладал наиболее быстрым компилятором, но его интегрированная среда разработки (IDE) работала только на базе DOS. С появлением Windows, возникла потребность создания среды программирования для этой платформы. Раньше чтобы создать элементарную программу, разработчикам приходилось писать несколько страниц кода. В компании Borland понимали, что нужно как-то облегчить их жизнь, поэтому решили доработать Pascal, который к тому моменту уже почти не использовался. Но изменения оказались настолько серьёзными, что в результате появился новый язык программирования — Delphi.

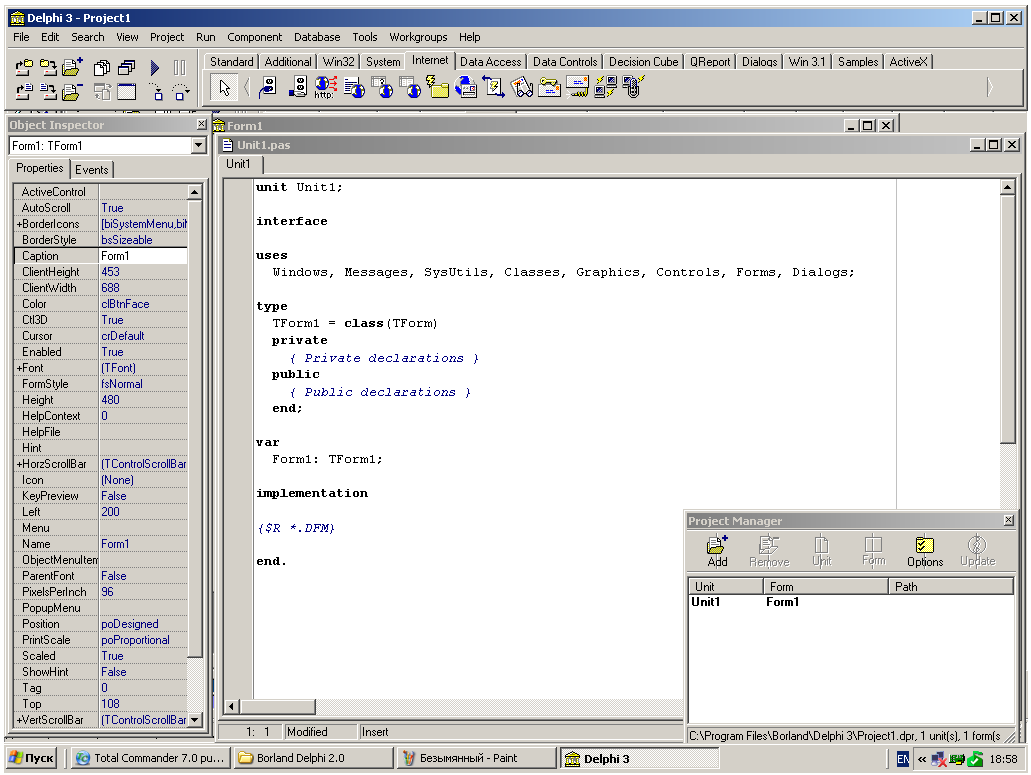


Рис. 17.7 Среда разработки языка Delphi

В России он начал применяться в конце 1993 году и сразу обрёл большую популярность для разработки прикладного программного обеспечения. Программисты оценили Delphi потому, что это объектно-ориентированное программирование за счет использования инструментов

[6] Своё имя система Delphi получила в честь древнегреческого города Дельфы. Название было выбрано неспроста: город Дельфы связан с именем бога мудрости и покровителем искусств Аполлона. Согласно легенде главное святилище Аполлона находилось именно в этом городе. Его жрицы-сивиллы изрекали пророчества желающим узнать свою судьбу.

разработки (case-средств) даёт высокую скорость разработки (рис. XX.H), сокращает количество возможных ошибок и обеспечивает эффективную работу полученного приложения. При разработке, например, приложений под Windows, эффективность Delphi очевидна, т.к. в нём есть много готовых компонентов, эффективный код и быстрая компиляция, а синтаксис менее запутан, чем, например, у C++.

В настоящее время использование языка Delphi несколько потеряло свою актуальность.

Язык ADA. Нельзя обойти стороной и мультипарадигменный язык программирования ADA, созданный в 1979—1980 годах в ходе проекта Министерства обороны США с целью разработать единый язык программирования для встроенных систем (то есть систем управления автоматизированными комплексами, функционирующими в реальном времени). Имелись в виду прежде всего бортовые системы управления военными объектами (кораблями, самолётами, танками, ракетами, снарядами и т. п.). Перед разработчиками не стояло задачи создать универсальный язык, поэтому решения, принятые авторами Ады, нужно воспринимать в контексте особенностей выбранной предметной области. Язык назван в честь Августы Ады Лавлейс.

В исходном варианте, стандартизованном в 1983 году, ADA - это структурный, модульный язык программирования, содержащий высокоуровневые средства программирования параллельных процессов. Синтаксис Ады унаследован от языков типа Algol или Pascal, но расширен, а также сделан более строгим и логичным. ADA — язык со строгой типизацией, в нём исключена работа с объектами, не имеющими типов, а автоматические преобразования типов сведены к абсолютному минимуму: допускается неявное приведение значения общего целого или вещественного числового типа к совместимому числовому типу. В стандарте 1995 года в язык были добавлены базовые средства объектно-ориентированного программирования, в стандарте 2007 эти средства были дополнены, поэтому современный ADA -объектно-ориентированный язык программирования. На рис.17.8 приведен фрагмент кода на языке ADA.

```
with Ada.Text_IO;

procedure Hello is
   use Ada.Text_IO;
begin
  Put Line(“Hello, world!”);
```

```
end Hello;
```

Рис.17.8 Фрагмент кода на языке ADA.

Язык Си (C). Язык Си-компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969—1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би. Первоначально был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств –от суперкомпьютеров до встраиваемых систем. Его преимуществами являлись компактность и объектный подход, однако сложность синтаксиса вызывала легкое «непонимание», граничащее с «неприятием» со стороны программистов, начинавших с Pascal и работающих под ОС Windows. Программирование на Си требует хорошей подготовки, при этом бывает сложно разобраться с текстом чужих программ. На рис. 17.9 приведен фрагмент кода на языке Си.

Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для таких языков программирования, как C++, C#, Java и Objective-C.

```
#include <studio.h>
#include <conio.h>
main()
{
        int a, b, c;
        printf (“Введите два целых числа\n”);
        scanf (“%d%d”, &a, &b);
        c=a+b;
        prinf(“%d+%d=%d”, a, b, c);
        getch();
}
```

Рис. 17.9 Фрагмент кода на языке Си.

Язык Си++. Язык Си++ - компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование,

обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником –зыком C , наибольшее внимание уделено поддержке объектно-ориентированного и обобщенного программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Изначально одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C. Множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Язык Си# Си# (читается как «Си шарп») объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

По сути, платформа .NET является средой, позволяющей управлять выполнением программ корпорации Microsoft и обеспечивать безопасность кода и совместимость различных языков программирования. Главной идеей стало использование промежуточного машинно-независимого языка CIL (Common Intermediate Language (IL)).

Технология работы с платформой заключается в том, что адаптированные под .NET высокоуровневые языки программирования переводятся компилятором сначала на язык CIL, а затем в режиме JIT (Just-In-Time, «по мере надобности») сама среда компилирует машинный код.

На рис. XX.X приведен фрагмент кода на языке C#.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java.

```
using System;
class Hello
{
        public static void Main()
        {
                Console.WriteLine(“Hello, World”);
        }
}
```

Рис. 17.10 Фрагмент кода на языке C#.

Переняв многое от своих предшественников - языков C++, Delphi, Modula, Smalltalk и, в особенности, Java - C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

Язык HTML. Язык HTML (от англ. HyperText Markup Language - «язык гипертекстовой разметки») - стандартизированный язык разметки документов для просмотра веб-страниц в браузере и предназначен для обмена информацией различного типа между людьми, не знакомыми с технологиями верстки документов. Т.е. Язык HTML- это каркас всех современных сайтов, и являясь языком разметки, он не является языком программирования, а если его и называют языком программирования, то это условно. Под языком разметки понимается набор символов созданный с целью передачи информации о строении текста. Это значит, что в отличии от языков программирования, язык разметки только управляет "внешним видом" текста для конечного пользователя.

Сегодня язык HTML является одним из основных языков, с помощью которого формируют интернет-представления информации. Позже в него были добавлены технологии работы с мультимедиа, выполнения скриптов и встраивания других языков программирования.

Язык XML. Язык XML (от англ. eXtensible Markup Language) - расширяемый язык разметки, был разработан в качестве инструмента для универсального хранения и передачи информации программным интерфейсом, объединенным в единую пользовательскую сеть. XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов как программами, так и человеком, с акцентом на использование в Интернете.

Документ, написанный в XML состоит из пролога, в котором хранятся директивы обработки информации и тела (основной части), содержащего непосредственно саму информацию. Интерпретатор XML, пользуясь директивами, обрабатывает информацию, находящуюся в теле (основной части) документа, и передает ее получателю в заданном виде (рис. 17.11).

```
<?xml version=”1.0”encoding=”windows=1251”?>
<TEST>
<NODE>
<ENG>HELLO SarPhTI!</ENG>
<RU>Привет СарФТИ!</R>
</NODE>
</TEST>
```

Рис. 17.11 Фрагмент кода на языке XML/

Язык Java. Язык Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последствии приобретенной компанией Oracle). Отличительной особенностью Java является то, что приложения Java обычно транслируются в специальный байт-код Java, выполняемый виртуальной машиной Java (JVM) – программой, обрабатывающей байтовый код и передающий инструкции интерпретатору. Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

Датой официального выпуска языка Java считается 23 мая 1995 г. Язык Java занимает высокие места в рейтинге языков программирования.

Язык PHP. Язык PHP (Hypertext PreProcessor, «препроцессор гипертекста») - скриптовый язык программирования, созданный для генерации HTML-страниц на веб-сервере и работы с базами данных. Имеет открытый исходный код. Изначально создавался для разработки веб-приложений, но в процессе обновлений стал языком общего назначения. В

настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

На рис.17.12 Приведен фрагмент кода на PHP.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>

    <?php
    echo "Привет студентам СарФТИ!";
     ?>

  </body>
</html>
```

Рис. 17.12 фрагмент кода на PHP.

Язык SQL. Язык SQL (Structured Query Language)- структурированный язык запросов, используемый в системах управления реляционными базами данных, или для их потоковой обработки, и является непроцедурным языком программирования. SQL - это декларативный язык программирования, основанный на множествах, а не императивный язык программирования, такой как C или BASIC. Язык имеет всего пять основных команд, которые позволяют манипулировать хранящимися структурированными данными. Однако расширения стандартного SQL используют процедурные функции языка программирования для построения конструкций управления потоком. Также его поддерживают постреляционные и многомерные системы управления базами данных (СУБД) в качестве инструмента обработки и извлечения данных. Его расширением является, например, T-SQL (Transact SQL), осуществляющий обработку транзактов.

```
USE СарФТИ НИЯУ МИФИ
SELECT Id_number, Surname, Name
FROM Staff
WHERE (Id_number=25 AND Surname='Петров')
OR (Name='Сергей' AND LastName='Александрович');
```

Рис. 17.13 Фрагмент кода на языке SQL

Конечно, имеются и другие языки программирования, которые решают вполне определенные задачи. Например, CSS (каскадные таблицы стилей), JavaScript (встраиваемый язык для программного доступа к объектам приложений), Ajax (для построения интерактивных пользовательских интерфейсов и веб-приложений), jQuery (библиотека функций для обеспечения доступа к приложениям HTML) и т.п.

История развития языков программирования демонстрирует эволюцию алфавита, синтаксиса и семантики, приближая их к естественным языкам.

Многие программисты не относят к языкам программирования языки, программы на которых представляют собой смесь исходных данных и команд языка. Таким языком является HTML, на котором написана читаемая Вами страница сайта. Команды языка HTML (они называются тегами) указывают, что нужно сделать с данными, но не как. Например, тег *TABLE* приказывает отобразить следующие за ним данные в виде таблицы, но не говорит о том, как это сделать, каким образом при сжатии пользователем рабочей области браузера менять размеры таблицы, переносить со строки на строку текст в сжимаемой клетке. Языки программирования в которых указывается, что нужно сделать, но не как, называются *декларативными*.

Самое общее определение программы и языка программирования можно сформулировать так.

Программа - алгоритм, представленный так, что его можно выполнить на ЭВМ. Язык программирования - это совокупность слов и правил, определяющих форму записи алгоритма.

В настоящее время создано несколько тысяч языков программирования. Почему так много? Причина в бесконечном разнообразии алгоритмов. Придумать язык, удобный для записи всех алгоритмов невозможно.

Только для имитационного моделирования разработано более семисот языков программирования, в том числе, считающийся первым объектно-ориентированным языком язык Simula, язык GPSS и, по утверждению его автора, язык C++. Такое разнообразие объясняется тем, что имитируемые процессы сильно отличаются друг от друга.

Люди, разрабатывающие какой-либо язык программирования, вольно или невольно ограничивают возможности языка своими интуитивными представлениями о его назначении. Например, при разработке языка *C* не был предусмотрен строковый тип данных. Позже оказалось, что этот тип необходим в большинстве решаемых на ЭВМ задач. Объективно неизбежная ограниченность любого языка программирования порождает желание создать новый язык, свободный от нежелательного ограничения. Но новый язык также имеет ограничения, неудобства, сложности применения и так до бесконечности.

Кроме того, программирование - самая высокодоходная область человеческой деятельности. Поэтому создаются языки программирования, в которых нет объективной необходимости, но на распространении которых можно хорошо заработать.

Классификация языков программирования

Дать исчерпывающую классификацию языков программирования невозможно. Существует несколько классификаций. Рассмотрим две из них:

- по степени приближенности к естественным языкам;
- по степени детализации алгоритма.

По степени приближенности к естественным языкам языки программирования делят на три уровня:

1. языки высокого уровня;
2. автокоды (языки ассемблера);
3. языки машинных команд (машинных кодов).

В современных ЭВМ есть ещё один уровень - ***языки микропрограммирования.*** На них пишутся (микро)программы машинных команд.

ЭВМ, построенная по принципам фон Неймана, может выполнять только программу, записанную в машинных кодах. Программист, пишущий в машинных кодах, может использовать любые, самые тонкие возможности конкретной модели ЭВМ. Но для этого программист должен знать эти возможности. В отличие от программы, написанной на языке высокого уровня, программа в машинных кодах может выполняться только на ЭВМ с соответствующей системой команд. Программирование в кодах очень трудное, дорогое.

Пример. Пусть нужно сложить два числа, обозначенные через *A* и *B*. Число *A* хранится в ячейке памяти с адресом 000000001011_2. Число *B* имеет адрес 000100001010_2. Код команды сложения - 00000101_2. Запишем команду сложения двумя способами: в виде машинной команды и в условном виде (в виде мнемокода).

1) 00000101 000000001011 000100001010
   код опер. адрес A адрес B

и

2) Сл A B;
   Сл - сложить.

Какой способ легче, более нагляден? Очевидно мнемокод. Языки, в которых машинные коды заменены более понятными мнемокодами, сначала называли автокодами. Фирма IBM стала использовать автокод как промежуточный язык при переходе от представления программы на языке

высокого уровня к представлению в машинных кодах. При сборке программы из модулей, первоначально написанных на (может быть, разных) языках высокого уровня, модули сначала переводились на автокод, затем собирались *(ассемблировались*) в одну программу, которая транслировалась в коды. Часть операционной системы, выполнявшая *ассемблирование* называлась Ассемблером

Автокод, использовавшийся программой-ассемблером, стали называть языком ассемблера, сократив на профессиональном жаргоне до просто ассемблера.

Что делать математику, желающему проверить на ЭВМ новый разработанный им численный метод? Чтобы написать программу в кодах или на автокоде, ему нужно получить вторую профессию программиста. Уже на самом раннем этапе развития ЭВМ появилась идея разработать формальный, но максимально приближенный к естественному язык записи алгоритма, с которого можно было бы автоматически перевести алгоритм (программу) в машинные коды.

Языки программирования, максимально приближенные к естественным, называются языками высокого уровня

По степени детализации алгоритма языки высокого уровня делятся на:

1. императивные языки (Fortran, C, Perl, JavaScript и т. д.)
2. декларативные (ЛИСП, Пролог, SQL, HTML и т. д.)

На императивом языке программа записывается подробно шаг за шагом, а в программе на декларативном языке только указывается, что нужно сделать (какой алгоритм выполнить). Например, чтобы решить квадратное уравнение $ax^2 + bx + c = 0,$ на декларативном языке нужно указать тип уравнения и коэффициенты, а на императивом написать программу для расчёта по формуле: $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

Пример запроса к базе данных на декларативном языке SQL (Structured Query Language - язык структурированных запросов):

```
SELECT fio,adres FROM sotrudniki WHERE fio LIKE 'Ка%';
```

Этот запрос читается так:

Выбрать из таблицы sotrudniki ФИО (fio) и адреса всех сотрудников, ФИО которых начинаются на "Ка".

Как это делается, в SQL не указывается.

Большинство языков высокого уровня удовлетворяют условию полноты по Тьюрингу, то есть на них можно записать любой алгоритм. Разница лишь в трудоёмкости. Но есть специализированные языки, на которых пишутся программы лишь для узких классов задач. Например на SQL пишутся только запросы к базам данных.

Границу между императивными и декларативными языками провести трудно. Например, однажды написав на императивном языке функцию решения квадратного уравнения, можно в дальнейшем просто вызывать (декларировать) её:

```
(x1,x2)=kvadrUr(a,b,c);
```

В этом случае формально алгоритм решения квадратного уравнения скрыт, так же как и в программе на декларативном языке.

Сходство и различия языков высокого уровня

Наиболее сильные различия декларативных языков как внутри класса так и с императивными языками. Между императивными языками есть значительное сходство. Например, рассмотрим фрагменты сложения первых пяти натуральных чисел на различных языках программирования:

Язык Fortran

```
N=0
do i=1,5
  N=N+i
end do
```

Язык Pascal

```
   var N,i:integer;
N:=0;
for i:=1 to 5 do
  N:=N+1;
```

Язык Basic

```
   N=0
For i=1 To 5
  N=N+i
Next i
```

Язык PL/1

```
   DECLARE (N,I);
N=0;
DO I=1 To 5 BY 1;
  N=N+I;
END;
```

Языки C, C++, C#, Java

```
   int N,i;
N=0;
for(i=1;i<=5; i++)
  N=N+i;
```

Язык JavaScript

```
   N=0;
for(i=1; i<=5; i++)
  N+= i;
```

Языки Perl, PHP

```
$N=0; #Имя скалярной переменной в Perl и PHP начинается со знака $
 for($i=1; $i<=5; $i++)
 {  $N+= $i; }
```

Сходство настолько большое, что при переходе с одного языка на другой часто бывает трудно вспомнить эти мелкие, непринципиальные отличия. Но сходство не только в этом операторе. Во всех перечисленных языках есть следующие операторы:

1. безусловного перехода;
2. условного перехода (оператор условия);
3. оператор цикла с предусловием;
4. оператор цикла с постусловием;
5. оператор цикла с заданным числом повторений.

Этот список можно сократить, заменив операторы цикла оператором условного перехода

```
if(арифметическое выражение)метка 1, метка 2, метка 3
```

Если арифметическое выражение в этом операторе отрицательное, то происходит переход к оператору с меткой 1, если оно равно нулю, то будет переход к оператору с меткой 2, наконец при положительном значении - к оператору с меткой 3. Например, предыдущий пример сложеня первых пяти членов натурального ряда будет выглядеть так:

```
      N=0
  i=1
1 N=N+i
  i=i+1
  if(i-6)1, 2, 3
2 . . .
  . . .
```

Сильнее всего языки программирования высокого уровня отличаются способами задания областей видимости переменных. В двух принципиальных случаях возникает необходимость сделать так, чтобы в разных областях программы под одним названием можно было бы использовать разные по смыслу переменные. Первая причина: если в большой программе с несколькими сотнями переменных всем переменным присвоить уникальные имена, то легко запутаться. По аналогии, если решить, что в России все улицы должны иметь уникальные имена, то очевидны неудобства. Жители Петербурга, Воронежа, Вологды, Москвы и ещё нескольких десятков городов не захотят переименовывать свои Садовые улицы. Также в группе программистов, разрабатывающих одну большую программу удобнее предоставить каждому возможность называть переменные так, как ему удобно.

Вторая причина - передача аргументов (параметров) в функцию. Например, нужно вычислить $\sin(x)$ и $\sin(y)$. В этом случае в функции используется своя видимая только в ней переменная. Этой переменной

можно присвоить любое имя (*t, x, y, z, ...*). Перед выполнением функции её аргументам присваиваются значения, указанные в операторе вызова.

Всё разнообразие способов задания областей видимости в рамках данной лекции невозможно даже перечислить.

Расширение возможностей языков программирования. Библиотеки функций

В состав языка программирования можно включить библиотеки функций, например графическую и математическую. Любой программист может добавить в состав своей системы программирования свои библиотеки. С помощью библиотек можно сколь угодно расширить возможности языка. Поэтому нужно рассматривать возможности языка программирования только вместе с присоединёнными к нему библиотеками. В некоторых случаях приходится выбирать не очень удобный базовый язык с библиотекой, имеющей хороший набор функций для решения поставленной задачи.

Выбор языка программирования

Программист приступая к решению задачи должен выбрать наиболее подходящий язык программирования. Первоначально в пятидесятые годы прошлого века большинство задач, которые нужно было решать на ЭВМ, разделили на две группы:

1. экономические,
2. инженерные и научные.

Считалось, что в экономических задачах много исходных данных и результатов, но мало расчётов, а в научных и инженерных наоборот - мало исходных данных и результатов, но много расчётов. Для экономических задач в США был разработан язык КОБОЛ, а для инженерных и научных - ФОРТРАН. В дальнейшем оказалось, что есть экономические задачи с огромным количеством расчётов, например, задачи линейного программирования, а научные задачи, связанные с обработкой результатов наблюдений, требуют обработки огромного количества накопленных данных.

В шестидесятые годы прошлого века шло бурное развитие операционных систем и появился новый класс задач - разработка системного программного обеспечения. Операционная система - это мостик между оборудованием, из которого состоит ЭВМ, и прикладными программами. Например, прикладная программа лишь указывает объём передаваемой информации и два устройства - источник и приёмник информации, а передачу информации выполняет система ввода-вывода операционной системы.

Языки высокого уровня, в отличие от языков ассемблера и машинных кодов, не привязаны ни к какой ЭВМ, а системные программы пишутся для

конкретной архитектуры ЭВМ. Поэтому понадобился язык, по своей наглядности не уступающий языкам высокого уровня, но позволяющий программисту получить доступ ко всем аппаратным возможностям ЭВМ. Таким языком стал язык С (Си), в котором программист может использовать адреса переменных (указатели), вставлять в программу фрагменты на языке ассемблера.

В языке Си использован весь накопленный к моменту его появления опыт написания программ на языках высокого уровня. Оказалось, что удобство языка программирования определяется простотой и краткостью тех конструкций языка, которые используются в программе очень часто. Программы на ранних версиях языка ФОРТРАН загромождались огромным количеством меток. На АЛГОЛе начало и конец блока обозначались словами begin и end. Использование в СИ блочного оператора if и фигурных скобок вместо begin и end очень сильно увеличило наглядность, "читаемость" программы. Введение ряда подобных, на первый взгляд мелких, изменений и хорошая реклама сделали язык СИ очень популярным и для написания прикладных задач.

В восьмидесятые годы прошлого века вычислительная техника проникла практически во все сферы человеческой деятельности. Этому способствовало повсеместное применение персональных ЭВМ. Было написано огромное количество программ и выявился, казалось бы, очевидный факт. Оказалось, что большинство программ, независимо от прикладной области, состоят из чередующихся блоков (частей) ввода данных, обработки (вычислений) и вывода результатов. Типичным примером таких задач являются имитационные модели, в которых сначала вводится большое количество данных, задающих параметры модели, затем производятся расчёты (имитация работы моделируемого объекта), после завершения которых выводятся часто очень объёмные результаты моделирования. При вводе-выводе производится в основном формирование и разборка строк текста. Для этих операций наиболее удобен тип данных *строка*. При вычисленях к типам данных предъявляются более разнообразные требования. Например, для достижения требуемой точности используются числа с плавающей запятой двойной длины (тип double на языке С).

В настоящее время нет языка программирования, одинаково хорошо приспособленного и для обработки строк и для операций с числовой информацией. Характерной особенностью языков, предназначенных для обработки текста (Perl, PHP, JavaScript) является *контекстная зависимость типа*, т.е. тип одной и той же переменной может меняться в зависимости от контекста. Любопытно, что зависимость типа данных от контекста есть и в

языке С, который относится к языкам со строгим, явным заданием типов переменных. Вот пример фрагмента программы на С, в котором переменная *i*, рассматривается как целая, логическая и действительная, а переменная *b* как действительная и ... строковая (!!!), хотя формально в С нет логических и строковых типов переменных.

```
  int i=5;       //i - целая
float a=2.5,b;
if(i)          // i - логическая (true)
  b=i+a;       // i - действительная
else b=a;
  cout <<"b="<< b << endl; // b - строковая
//результат:  b=7.5
```

Практика показала, что язык, хорошо приспособленный к обработке текстовой информации, должен иметь в своём составе:

1. тип данных *строка*;
2. операцию *конкатенация*;
3. операцию *интерполяция*;
4. регулярные выражения.

Все четыре составляющие имеются в Perl и PHP. В JavaScript отсутствует операция интерполяции и одним символом + обозначаются операции конкатенации и сложения. В Perl и PHP конкатенация обозначается точкой. Операция интерполяции заключается в том, что в строке, заключённой в двойные кавычки, имена переменных заменяются их значениями.

Пример формирования строки с помощью **операции интерполяции** на языках Perl и PHP.

```
$F='Павлов'; $Im='Иван'; $Otch='Петрович'; $God=1849;
print "Академик $Im $Otch $F родился в $God году.";
# Напечатается:
# Академик Иван Петрович Павлов родился в 1849 году.
```

Без интерполяции придётся использовать много операций конкатенации (точек) и одинарных кавычек:

```
print 'Академик '.$Im.' '.$Otch.' '.$F.'  родился в '.$God.' году.';
```

Регулярные выражения используются для поиска и замены в строке подстроки (части строки), соответствующей заданному образцу (шаблону). Найдём в строке дату, имеющую формат ДД.ММ.ГГГГ. Фрагмент программы напишем на языке Perl.

```
$str='А.С. Пушкин родился 06.06.1799 в Москве'; #$str - строка для поиска даты

$str=~m/\d\d\.\d\d\.\d{4}/; #Опертор поиска
                    # /\d\d\.\d\d\.\d{4}/ - регулярное выражение (шаблон)
print "Дата - $&. \n";
#   Результат:
#   Дата - 06.06.1799
```

\d означает, что на этом месте должна быть любая цифра. Точка в регулярном выражении - специальный символ, на месте которого в строке может быть любой символ, а **\.** - это точка.

\d{4} - четыре цифры подряд. Весь оператор поиска читается так: *найти в строке $str подстроку, имеющую вид* : **две цифры точка две цифры точка четыре цифры**.

В общем виде задача описания вида искомой в тексте подстроки невероятно трудна, поэтому и регулярные выражения имеют соответствующий, часто пугающий своей непонятностью вид, но применение регулярных выражений для решения задачи поиска по сравнению с другими методами позволяет сократить время разработки и длину программы в несколько раз. Поэтому профессиональный программист, как это ни трудно, должен владеть техникой применения регулярных выражений.

На практике оказалось иногда удобным использовать для программирования разных частей одной задачи разных языков программирования. Например для программирования ввода и вывода данных в имитационной модели целесообразно использовать язык Perl, а для процесса собственно моделирования - язык C, т.е. разбить одну программу на три. Программа ввода обеспечит хороший интерфейс с пользователем и преобразует исходные данные из формата, удобного пользователю, в формат, предназначенный для вычислений на языке C. Программа вывода преобразует результаты моделирования в формат, удобный пользователю. Проще всего данные между этими программами передавать в виде текстовых файлов.

Для отдельных классов задач используются специализированные языки. Например, для разметки страниц сайтов используется язык HTML, для запросов к реляционным базам данных - SQL. Интеллектуальные системы пишутся на языке ПРОЛОГ.