ЛЕКЦИЯ 12

Языки программирования: история, области применения, классификации, парадигмы

Краткая история развития языков программирования

✔ *Машинные языки (1940-е гг.)*

Машинный код - система команд конкретной вычислительной машины (машинный язык), который интерпретируется непосредственно микропроцессором или микропрограммами данной машины. Машинные языки предполагают кодирование команд (инструкций в виде «слов» - числовых кодов) конкретного процессора на аппаратном уровне. Каждая инструкция представляет собой одно элементарное действие для центрального процессора (например, считывание информации из ячейки памяти). Основные носители информации - перфокарты и перфоленты.

✔ *Языки ассемблера (машинно-ориентированные языки, 1950-е гг.)*

Языки ассемблера предполагают мнемоническое кодирование машинных команд и использование символических меток вместо физических адресов ячеек с данными.
Ассемблер - компьютерная программа, предназначенная для преобразования мнемонических инструкций в исполняемый машинный код.

Программирование на ассемблере требует хорошего знания архитектуры машины, на которой программа будет выполняться, является низкоуровневым (программирование в терминах ячеек памяти, индексных и базовых регистров, указателя стека и т.п.).
Язык ассемблера, вследствие жесткой привязки к архитектуре компьютера определенного класса, делает программы непереносимыми.

```
.586 ;Будут использоваться дополнительные команды
assume   CS:code,DS:data
code segment use16
main proc
        mov  AX, data                   ;Настроим DS наш
        mov  DS,AX                      ;сегмент данных
;Сохраним исходный вектор 4Ah
        mov  AX,354Ah
        int  21h
        mov  word ptr old_4a,BX
        mov  word ptr old_4a+2,ES
;Установим наш обработчик прерывания 4Ah
        mov  AX,254Ah
        push DS                         ; Сохраним DS
        push CS                         ;Настроим DS на сегмент
        pop  DS                         ;команд
        mov  DX,offset new_4a;DS:DX->new_4a
        int  21h
        pop  DS                         ;Восстановим DS
;Установим будильник
        mov  AH,02h                     ;Чтение текущего времени
        int  1Ah
        call add_time                   ;Прибавим 1 секунду
```

✔ FORTRAN (FORmula TRANslator, 1954-57 гг.)

Языки высокого уровня (ЯВУ) имитируют машинные языки, используя слова разговорного языка и общепринятые математические символы. **FORTRAN** - первый алгоритмический язык программирования высокого уровня (первая версия разработана в корпорации IBM). Исходный код программы переводится в машинный с помощью специальной программы - компилятора.

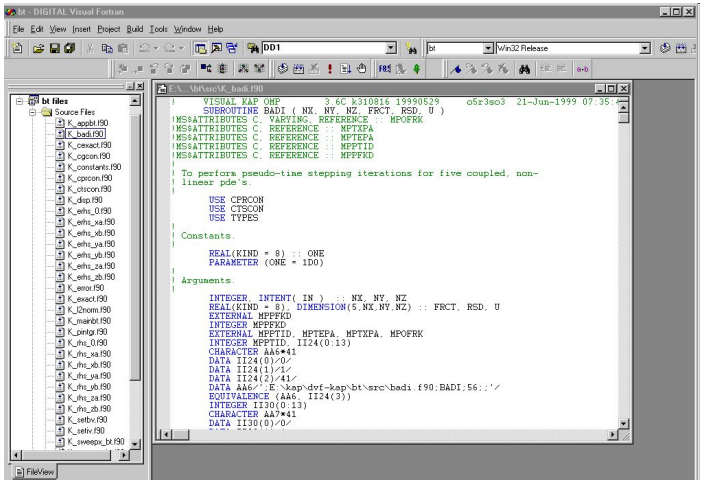
Стандарты языка: 66 (отступ слева для ввода кода программ с перфокарт), 77 (новые типы данных, условные операторы и др.), 90 (свободный формат записи кода, дополнительные описания и управляющие конструкции, элементы ООП), 95, 2003, 2008.

Компиляторы: VS Fortran, XL Fortran (IBM), Digital Visual Fortran (DVF, DEC), Microsoft Visual Studio (FPS), Lahey, Watcom, Intel Fortran Compiler, Sun Studio Fortran Compiler, GNU (g77, g95).
Компиляторы для разных платформ (MS Windows, UNIX). Трансляторы с FORTRAN-а на другие (F2C).

FORTRAN имеет большой набор встроенных математических функций, поддерживает работу с целыми, вещественными и комплексными числами высокой точности, имеет типовые лексические конструкции (условия, циклы), использует операторы безусловного перехода и метки, форматный ввод-вывод, эффективно работает с многомерными массивами.

FORTRAN широко используется для научных и инженерных вычислений, доступно большое количество написанных на нём программ и специализированных библиотек подпрограмм (вычислительные - SCALAPACK, BLAS, IMSL; организация распределенных вычислений - MPI, PVM; построение графических интерфейсов - Quickwin, FORTRAN/Tk).

```
      PROGRAM REDUCE
      WRITE(UNIT=*, FMT=*)'Enter amount, % rate, years'
      READ(UNIT=*, FMT=*) AMOUNT, PCRATE, NYEARS
      RATE = PCRATE / 100.0
      REPAY = RATE * AMOUNT / (1.0 - (1.0+RATE)**(-NYEARS))
      WRITE(UNIT=*, FMT=*)'Annual repayments are ', REPAY
      WRITE(UNIT=*, FMT=*)'End of Year  Balance'
      DO 15,IYEAR = 1,NYEARS
           AMOUNT = AMOUNT + (AMOUNT * RATE) - REPAY
           WRITE(UNIT=*, FMT=*) IYEAR, AMOUNT
15    CONTINUE
      END
```



✔ *COBOL (COmmon Business Oriented Language, 1959 г.)*

COBOL - язык программирования, предназначенный, в первую очередь, для разработки бизнес-приложений. Одна из целей создателей - максимально приблизить конструкции к английскому языку (это приводит к неоправданному увеличению объема кода программы). Отличительной особенностью языка является возможность эффективной работы с большими массивами данных и файлами. До сих пор считается, что на языке COBOL написано больше всего кода.

Пример кода программы:

```
      $ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID.  Multiplier.
AUTHOR.  Michael Coughlan.
* Example program using ACCEPT, DISPLAY and MULTIPLY to
* get two single digit numbers from the user and multiply them together

DATA DIVISION.

WORKING-STORAGE SECTION.
01  Num1                                PIC 9  VALUE ZEROS.
01  Num2                                PIC 9  VALUE ZEROS.
01  Result                              PIC 99 VALUE ZEROS.

PROCEDURE DIVISION.
    DISPLAY "Enter first number  (1 digit) : " WITH NO ADVANCING.
    ACCEPT Num1.
    DISPLAY "Enter second number (1 digit) : " WITH NO ADVANCING.
    ACCEPT Num2.
    MULTIPLY Num1 BY Num2 GIVING Result.
    DISPLAY "Result is = ", Result.
    STOP RUN.
```

Разработчик языка -
Грейс Хоппер
(Grace Hopper, 1906-1992)

✔ *ALGOL (ALGOrithmic Language, 1958-60 гг.)*

ALGOL - название ряда языков программирования, применяемых при составлении программ для решения математических задач на компьютерах. ALGOL относится к языкам высокого уровня и позволяет легко переводить алгебраические формулы в программные команды. Язык удобен для описания алгоритмов и применяет систему обозначений, близкую к принятой в математике.

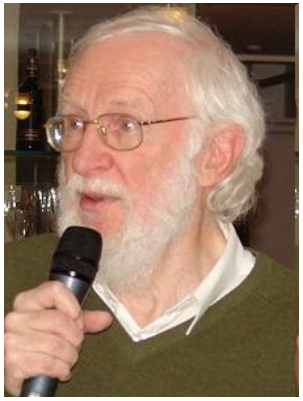
Программа на языке ALGOL - не свободная последовательность команд, а блочная структура, состоящая из четко описанных и отделенных друг от друга частей (первый *структурный или процедурный* язык). В языке выделены структурные управляющие конструкции: ветвления, циклы, последовательные участки, исполняющие условно или многократно вложенные наборы операторов, что дало возможность описывать логику программы без использования безусловных переходов. Появилась возможность использовать рекурсивные процедуры.

Стандарт языка полностью проигнорировал средства ввода-вывода (каждая реализация языка должна решать этот вопрос самостоятельно, исходя из особенностей целевой машины и потребностей пользователей, что делает несовместимыми программы, написанные для разных компиляторов).

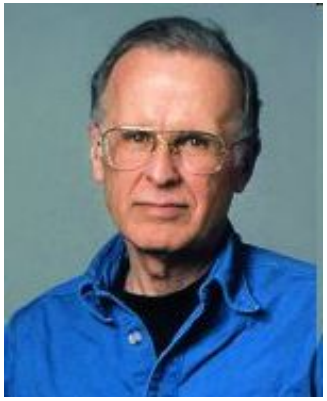
Реализации: ALGOL-58, ALGOL-60, ALGOL-68. Развитие - PL/I (Programming Language I).

```
'begin'
        'comment'
        эта программа демонстрирует передачу параметров по имени ;
        'integer' x, y;
        'procedure' sub(a, b);
        'integer' a, b;
        'begin'
                x := 5;
                print(''First in sub : a='', a, '' b='', b, newline);
                a := 7;
                print(''Second in sub: a='', a, '' b='', b, newline);

        'end';
        x := 7;
        y := 11;
        sub(y, x + y);
        print(''After sub    : x='', x, '' y='', y, newline);
'end'
```

Основные создатели языка



Пи́тер На́ур
(Peter Naur)



Джон Бэкус
(John Backus)

✔ *BASIC (Beginner's All-purpose Symbolic Instruction Code, 1963 г.)*

BASIC (многоцелевой язык символических инструкций для начинающих) - простой язык программирования, задумывался в первую очередь как средство обучения и как первый изучаемый язык.

Синтаксис языка напоминает FORTRAN и многие элементы заимствованы из него. Язык задумывался для обучения, поэтому его конструкции максимально просты. Ключевые слова для операторов взяты из английского языка. Основные типы данных: строки и числа.

BASIC традиционно подвергается критике за то, что его простота и бесструктурность поощряют неверные методики и привычки программирования, которые могут быть уместны только при разработке небольших программ.

Развитие языка: Microsoft Visual Basic - средство разработки программного обеспечения, включающее язык программирования и среду разработки. Ориентирован на оперативную разработку Windows-приложений с графическим интерфейсом пользователя.

```
10 CLS 'Очистка экрана
20 PRINT "Добро пожаловать!" 'Заголовок в первой строке
30 'Цикл, выводящий линию под заголовком, на всю ширину экрана
40 FOR I=1 TO 80
50 PRINT "=";
60 NEXT I
65 'Ввод символьных данных от пользователя (комментарий добавлен после
70 INPUT "Имя:  ",N$
80 INPUT "Фамилия: ",S$
90 INPUT "Отчество: ",T$
95 'Вырезаем копию первых символов из имени и отчества
100 N2$=LEFT$(N$,1,1)
110 T2$=LEFT$(T$,1,1)
120 'Выводим результат
130 PRINT "Ваше имя кратко: ";S$;" ";N2$;". ";T2$"."
140 INPUT "Повторить программу? (Y/N) ",$U
150 IF U$="Y" THEN GOTO 10
160 END
```

Создатели языка

Томас Курц и Джон Кемени
(Thomas E. Kurtz & John G. Kemeny)

✔ *Pascal (1970 г.)*

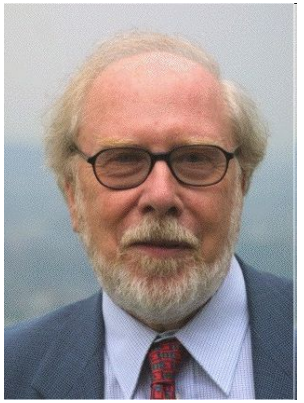
Паскаль (Pascal) - высокоуровневый язык программирования общего назначения, назван в честь выдающегося французского ученого Блеза Паскаля. Один из наиболее известных языков программирования, широко применяется в промышленном программировании, обучении программированию, является базой для большого числа других ЯВУ.

Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Паскаль включает в себя множество «алголоподобных» структур и конструкций (*if, then, else, while, for*…), а также возможности для структурирования информации и абстракций (определение типов, записи, указатели, перечисления, множества).

Распространенные реализации: Turbo Pascal, Borland Pascal, GNU Pascal, TMT Pascal, Free Pascal. Объектное расширение языка: Object Pascal, Delphi. Развитие: языки *Modula-2* и *Oberon*.

```
program MaximuM;
const
     m = 7;      {строки}
     n = 4;      {столбцы}
var
 Matrix: array[1..m, 1..n] of integer;  {двумерный массив}
 i,j,max,smax: integer; {max - максимальный элемент; smax - строка, в которой max}

Begin

Writeln('Матрица:');

max:= -41;
smax:= 1;

Randomize;
for i:=1 to m do begin  {задание и вывод массива, поиск максимального элемента}
 for j:=1 to n do begin
                  Matrix[i, j]:=Random(71)-40;  {задаём массив числами от -40 до 30}
                  Write( (Matrix[i, j]):4 );
                  if (Matrix[i, j] > max) then  {как только нашли элемент строго больше max}
                      begin
                      max:= Matrix[i, j];       {сохраняем его в max}
                      smax:= i;                 {и номер его строки в smax}
                      end;
                  end;
```

Создатель языка



Никлаус Вирт
(Niklaus Wirth)

✔ *C (1972 г.)*

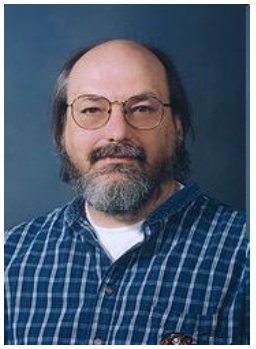
C - стандартизированный процедурный язык программирования общего назначения, исходно созданный для разработки и использования в операционной системе UNIX (в том числе для создания ядра системы, компиляторов других языков и т.п.). Язык создавался «программистами для программистов».

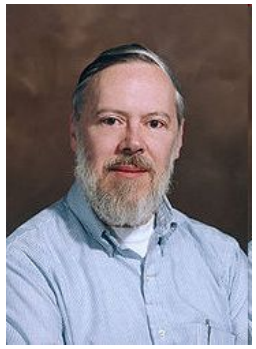
Сегодня *C* является самым популярным языком для создания как системного программного обеспечения, так и прикладных программ. Отличается высокой эффективностью, экономичностью и переносимостью. При этом многие элементы языка при некорректном использовании представляют потенциальную опасность.

Особенности языка *C*:
- простая языковая база (многие существенные возможности вынесены в библиотеки);
- ориентация на процедурное программирование, обеспечивающая удобство применения структурного стиля программирования;
- многие операции языка соответствуют машинным кодам и допускают прямую трансляцию в него;
- поддержка указателей на функции и статические переменные (возможность непосредственного доступа и манипуляции с адресами памяти);
- статическое, автоматическое и динамическое выделение памяти;
- содержит препроцессор, который обрабатывает текстовые файлы перед компиляцией (определение программных констант, макросов для замены вызовов функций, включение файлов с исходным кодом...)
- возможность использования записей (собирательные типы данных - структуры);
- передача параметров в функции по значению, а не по ссылке;
- области действия имен…

Стандарты и спецификации: K&R C (1978 г.), ANSI C (1989 г.), C99.

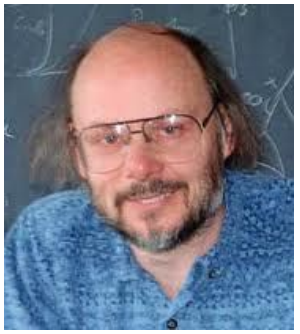
Компиляторы: Borland C++, Intel C/C++ compiler, Microsoft Visual Studio, C++ Builder, GNU Compiler Collection, Open Watcom, Sun Studio.





Кен Томпсон и Деннис Ритчи
(Kenneth Thompson & Dennis M. Ritchie)

✔ C++ (1983 г.)



Бьёрн Страуструп
(Bjarne Stroustrup)

C++ - компилируемый статически типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования, сочетает в себе свойства как высокоуровневых, так и низкоуровневых языков.

C++ добавляет к языку C объектно-ориентированные возможности: вводит классы, которые обеспечивают три важнейших свойства ООП: инкапсуляцию, наследование и полиморфизм.

C++ широко используется для разработки ПО различного назначения:

- системное ПО
- драйверы устройств
- приложения для встраиваемых систем, высокопроизводительных серверов
- прикладные приложения

Стандарты языка: ISO/IEC 14882:1998 «Standard for the C++ Programming Language» (1998 г.), ISO/IEC 14882:2003 (2003 г.).

Стандарт C++ включает описание ядра языка и стандартной библиотеки. Существует большое количество библиотек C++, не входящих в стандарт. Стандартная библиотека C++ состоит из следующих разделов: поддержка языка, средства диагностики, средства общего назначения, работы с текстовыми строками, интернационализации, контейнеры, итераторы, численные алгоритмы, средства ввода-вывода.

```
#include <iostream.h>
template <class T> class Array {
public: Array (T Size=1) : M (new T[Size]),
N(Size), n(0) {}
Array (void) { delete [] M;}
T Count (void) const { return n; }
T operator [] (int i) const { return M[i]; }
void Add (T Data);
int N, n; };
```

✔ *Java (1995 г.)*

Java - объектно-ориентированный язык прикладного программирования (обладает преемственностью по отношению к C++ и разрабатывался как альтернативна ему). Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) - программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Байт-код является полностью независимым от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Исполнение программы полностью контролируется виртуальной машиной: любые операции, которые превышают установленные полномочия программы вызывают немедленное прерывание.

Исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java (в сравнении с C/C++ - большее время решения задач и потребление памяти).

Основные возможности языка:

автоматическое управление памятью;
- расширенные возможности обработки исключительных ситуаций;
- богатый набор средств фильтрации ввода/вывода;
- набор стандартных коллекций, таких как массив, список, стек и т. п.;
- наличие простых средств создания сетевых приложений;
- наличие классов, позволяющих выполнять HTTP-запросы и обрабатывать ответы;
- встроенные в язык средства создания многопоточных приложений;
- унифицированный доступ к базам данных
- поддержка шаблонов;
- параллельное выполнение программ.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

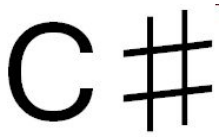
Компилируемые и интерпретируемые языки

✔ Программа на **компилируемом языке** при помощи специальной программы - компилятора - преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и записывается в исполняемый файл, который может быть запущен на выполнение как отдельная программа. Компилятор переводит программу с языка высокого уровня на низкоуровневый язык, понятный данному процессору.

✔ **Интерпретируемый язык** предполагает хранение программы на исходном языке и перевод ее текста на машинный язык (интерпретацию) непосредственно в момент запуска программы. Программа не может быть запущена без интерпретатора языка программирования.

Примеры компилируемых языков: Fortran, Pascal, C/C++
Примеры интерпретируемых языков: Perl, Python, языки командных оболочек UNIX (sh, bash, tcsh)

Промежуточный вариант предполагает компиляцию программы не в машинный язык, а в машинно-независимый код низкого уровня (байт-код) и последующее выполнение байт-кода «виртуальной машиной». Для выполнения байт-кода обычно используется интерпретация, хотя отдельные его части для ускорения работы программы могут быть транслированы в машинный код непосредственно во время выполнения программы по технологии компиляции «на лету». Примеры языков, использующих байт-код: Java, C#.

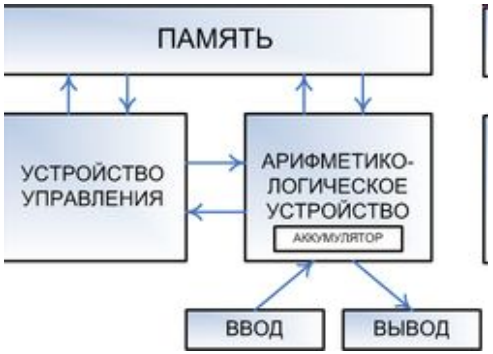


Парадигмы программирования

Парадигма программирования - это устоявшаяся система взглядов, определяющая стиль программирования, некоторый цельный набор идей и рекомендаций, определяющих стиль написания программ.

Парадигмы программирования:

- процедурная (императивная)
- объектно-ориентированная
- функциональная
- логическая



✔ *Процедурное программирование*

Процедурное (императивное) программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 1940-х годах.

Программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи. Основным является оператор присваивания, служащий для изменения содержимого областей памяти. Концепция памяти как хранилища значений, содержимое которого может обновляться операторами программы, является фундаментальной в императивном программировании.

Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.

Примеры процедурных языков: FORTRAN, Pascal, C, Ada.

Процедурное программирование: структурное (C, Pascal), операционное (FORTRAN, BASIC).

Парадигмы программирования

✔*Объектно-ориентированное программирование*

Объектно-ориентированное программирование (ООП) - это парадигма программирования, при использовании которой главными элементами программ являются объекты.

В языках программирования понятие объекта реализовано как совокупность свойств (структур данных, характерных для данного объекта), методов их обработки (подпрограмм изменения их свойств) и событий, на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта.

Инкапсуляция - объединение данных и свойственных им процедур обработки в одном объекте.

Класс (фундаментальное понятие парадигмы ООП) - это шаблон, на основе которого может быть создан конкретный программный объект, он описывает свойства и методы, определяющие поведение объектов этого класса. Каждый конкретный объект, имеющий структуру этого класса, называется экземпляром класса.

Наследование предусматривает создание новых классов на базе существующих и позволяет классу-потомку иметь (наследовать) все свойства класса-родителя.

Полиморфизм - возможность объектов с одинаковой спецификацией иметь различную реализацию (использовать различные методы обработки для разных объектов).

Модульность: объекты заключают в себе полное определение их характеристик, никакие определения методов и свойств не должны располагаться вне его, это делает возможным свободное копирование и внедрение одного объекта в другие.

Примеры объектно-ориентированных языков: C++, Java, Object Pascal, Python.

Парадигмы программирования

✔*Функциональное программирование*

Функциональное программирование - парадигма программирования, в которой процесс вычисления трактуется как вычисление значений функций в их математическом понимании (в отличие от функций как подпрограмм и процессу вычислений как последовательному изменению состояний в процедурном программировании). Функциональное программирование - это способ составления программ, в которых единственным действием является вызов функции.

Роль основной конструкции в функциональных языках играет выражение, к которым относятся скалярные константы, структурированные объекты, функции, тела функций и вызовы функций. Программа представляет собой совокупность описаний функций и выражения, которые необходимо вычислить. Вычисление выражения описывается как комбинация вызовов функций того же или более низкого уровня (часто рекурсивных). Каждая следующая функция в этой комбинации описывается аналогичным образом, до тех пор, пока описание не сведётся к предопределённым функциям, вычисление которых считается заданным. В отличие от императивного стиля, описывающего шаги, ведущие к достижению цели, функциональный стиль описывает математические отношения между данными и целью.

Функциональное программирование основано на вычислении результатов функций в зависимости от исходных данных и результатов выполнения других функций и не предполагает явного хранения состояния программы (и переменных в памяти).

Операторы присваивания отсутствуют (изменения состояния переменных не производится), вследствие чего переменные обозначают не области памяти, а объекты программы, что полностью соответствует понятию переменной в математике.

Примеры функциональных языков: Lisp, Haskell, ML.

```
(defun fibonacci (x)
    (if (= x 0)
        1
        (loop
            with result = 1
            with prev = 1
            for i from 0 below (1- x)
            do
            (
                psetq result
                (+ result prev)
                prev result
            )
        )
    )
)
```

Парадигмы программирования

✔*Логическое программирование*

Логическое программирование - парадигма программирования, базирующаяся на автоматическом доказательстве теорем с использованием механизмов логического вывода информации на основе заданных фактов и правил вывода.

Центральным понятием в логическом программировании является отношение. Программа представляет собой совокупность определений отношений между объектами (в терминах условий или ограничений) и цели (запроса). При этом нужно только специфицировать факты, на которых основывается алгоритм, а не определять последовательность шагов, которые требуется выполнить.

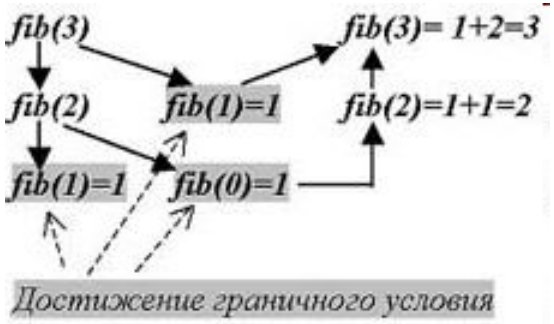
Программа строится из последовательности фактов и правил, затем формулируется утверждение, которое язык пытается доказать с помощью правил. Язык сам ищет решение с помощью методов поиска и сопоставления, которые в нем заложены. Логические программы не отличаются высоким быстродействием, так как процесс их выполнения сводится к построению прямых и обратных цепочек рассуждений разнообразными методами поиска.

Примеры логических языков: Prolog, Planner, Popler, Mercury.

Пример программы на языке Prolog, вычисляющей числа Фибоначчи:

fib(0) = 1, fib(1) = 1
fib(n) = fib(n-1) + fib(n-2)

```
predicates
      fib(integer,integer)
clauses
      fib(0,1).                  % граничное условие
      fib(1,1).                   % граничное условие
      fib(N,F) :- N>1,
                  N1=N-1,
                  fib(N1,F1),
                  N2=N-2,
                  fib(N2,F2),
                  F=F1+F2.
```



Рейтинг языков программирования

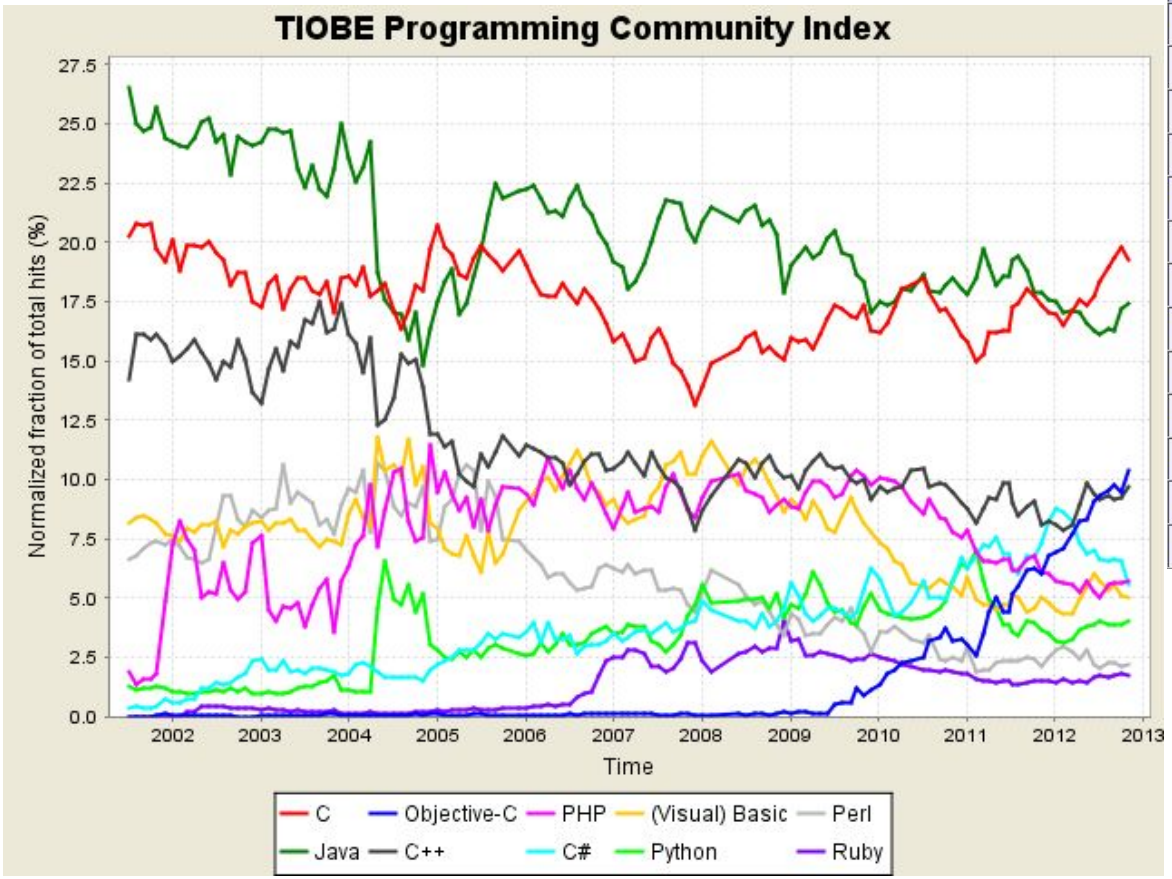
TIOBE Programming Community (*http://www.tiobe.com*)

Position Nov 2012	Position Nov 2011	Delta in Position	Programming Language	Ratings Nov 2012	Delta Nov 2011	Status
1	2	↑	C	19.224%	+1.90%	A
2	1	↓	Java	17.455%	-0.42%	A
3	6	↑↑↑	Objective-C	10.383%	+4.40%	A
4	3	↓	C++	9.698%	+1.61%	A
5	5	=	PHP	5.732%	-0.36%	A
6	4	↓↓	C#	5.591%	-1.73%	A
7	7	=	(Visual) Basic	5.032%	-0.01%	A
8	8	=	Python	4.062%	+0.45%	A
9	10	↑	Perl	2.182%	+0.10%	A
10	11	↑	Ruby	1.739%	+0.24%	A
11	9	↓↓	JavaScript	1.278%	-1.29%	A
12	16	↑↑↑↑	Delphi/Object Pascal	0.995%	+0.12%	A
13	13	=	Lisp	0.951%	-0.23%	A
14	14	=	Pascal	0.881%	-0.11%	A
15	23	↑↑↑↑↑↑↑↑	Visual Basic .NET	0.769%	+0.24%	A-
16	19	↑↑↑	Ada	0.662%	+0.04%	B
17	12	↓↓↓↓↓	PL/SQL	0.632%	-0.81%	B
18	18	=	Lua	0.631%	0.00%	A-
19	15	↓↓↓↓	MATLAB	0.620%	-0.34%	B
20	24	↑↑↑↑	Assembly	0.585%	+0.06%	B

Position	Programming Language	Ratings
21	SAS	0.577%
22	Bash	0.562%
23	Transact-SQL	0.473%
24	COBOL	0.471%
25	ABAP	0.442%
26	Fortran	0.437%
27	Scheme	0.418%
28	R	0.417%
29	Scratch	0.406%
30	Logo	0.389%
31	Prolog	0.360%
32	Erlang	0.339%
33	RPG (OS/400)	0.330%
34	Scala	0.324%
35	Haskell	0.323%
36	D	0.283%
37	Smalltalk	0.254%
38	APL	0.236%
39	Forth	0.228%
40	ML	0.226%
41	Awk	0.211%
42	NXT-G	0.208%
43	ActionScript	0.181%
44	Common Lisp	0.175%
45	Alice	0.161%
46	CFML	0.158%
47	LabVIEW	0.155%
48	Tcl	0.146%
49	Eiffel	0.143%
50	Groovy	0.142%

Рейтинг языков программирования

TIOBE Programming Community (*http://www.tiobe.com*)



Programming Language	Position Nov 2012	Position Nov 2007	Position Nov 1997	Position Nov 1987
C	1	2	1	1
Java	2	1	4	-
Objective-C	3	47	-	-
C++	4	4	2	6
PHP	5	5	-	-
C#	6	8	-	-
(Visual) Basic	7	3	5	5
Python	8	7	29	-
Perl	9	6	7	-
Ruby	10	9	-	-
Lisp	13	15	17	3
Ada	16	20	18	2
COBOL	24	17	3	12

Category	Ratings Nov 2012	Delta Nov 2011
Object-Oriented Languages	58.2%	+2.8%
Procedural Languages	37.4%	-0.6%
Functional Languages	3.1%	-1.2%
Logical Languages	1.3%	-0.9%

Category	Ratings Nov 2012	Delta Nov 2011
Statically Typed Languages	71.4%	+0.4%
Dynamically Typed Languages	28.6%	-0.4%

Области применения языков программирования

- ✔ научные вычисления (FORTRAN, C/C++, Pascal);
- ✔ системное программирование (C/C++, Java);
- ✔ искусственный интеллект (Lisp, Prolog);
- ✔ издательская деятельность (Postscript, TeX);
- ✔ удаленная обработка информации, веб-программирование (Perl, PHP, Java, Ruby);
- ✔ описание документов (HTML, XML).