Языки программирования

1. Обзор языков программирования

О курсе

- Распределение часов:
  - Лекции – 36 час.
  - Практические – 36 час.
- Форма контроля: экзамен
- Основная литература:
  - Бен-Ари М. Языки программирования. Практический сравнительный анализ. –М.: Мир, 2000.
  - Себеста Р. Основные концепции языков программирования. –М.: Вильямс, 2001.

Для чего изучать языки программирования?

- Разработка более эффективных алгоритмов и программ
- Расширение набора полезных программных конструкций
- Более обоснованный выбор подходящего языка
- Повышение способности к изучению новых языков
- Повышение способности к разработке новых языков

Язык программирования

- *Язык программирования* – система обозначений для *абстрактного* (не зависящего от архитектуры конкретного компьютера) описания вычислений.
- Абстрактное описание вычислений можно перевести в детализированную форму для последующего выполнения на компьютере.
- Перевод осуществляется специализированной программой (*компилятор*, *интерпретатор*).

Основные элементы языков программирования

- Алфавит, синтаксис, семантика
- Типы данных, значения, литералы, переменные, константы
- Операторы
- Подпрограммы
- Библиотеки
- Стандартизация языков и мобильность программ

Способы задания синтаксиса

- *Расширенные формулы Бэкуса-Наура (РБНФ)*
  - Нетерминальные символы – обозначают синтаксические конструкции языка, заключаются в угловые скобки <...>.
  - Терминальные символы образуют алфавит языка.
  - Метасимволы
    - ::= есть по определению
    - | или
  - Дополнительные метасимволы (расширение БНФ)
    - […] повторение символа 0 или 1 раз
    - {…} повторение символа произвольное число раз (в т.ч. нуль)
    - (…) группировка символов
- *Синтаксические диаграммы* – графическое изображение РБНФ.

РБНФ

Множество целых чисел:

- БНФ

  <целое число>::= <целое без знака> |
  + <целое без знака> | - <целое без знака>

  <целое без знака>::= <цифра> |<целое без знака >
  <цифра>

  <цифра>::= 0|1|2|3|4|5|6|7|8|9

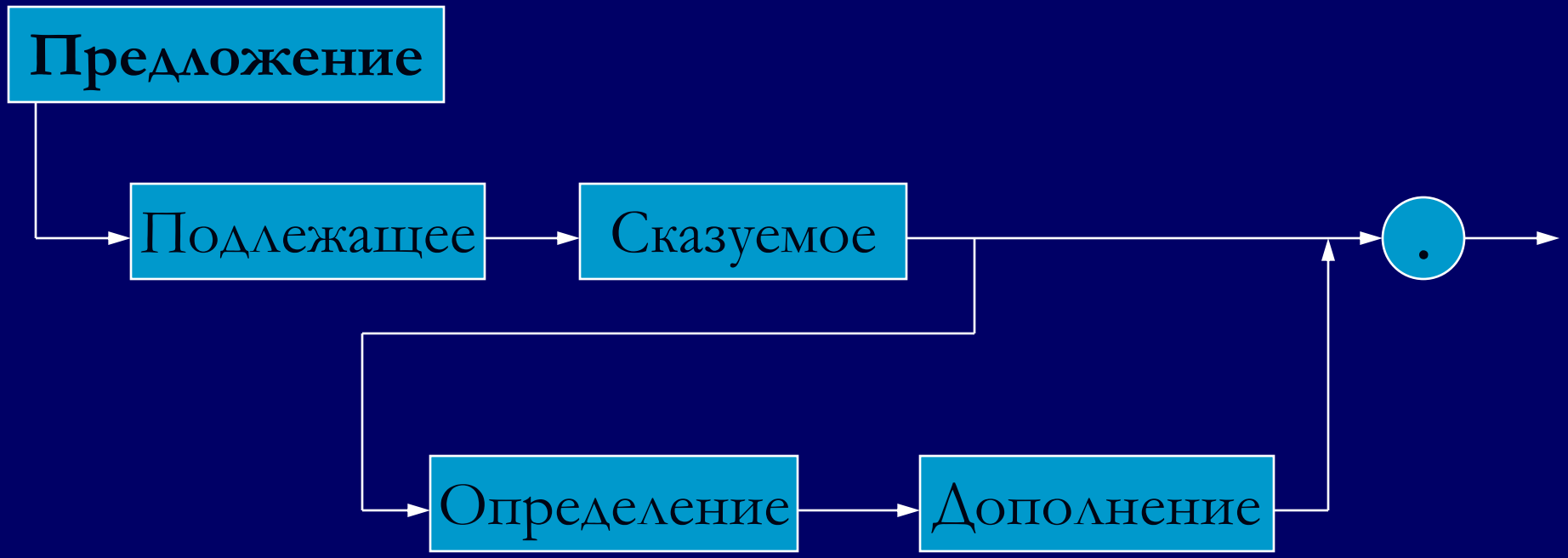
- РБНФ

  <целое число>::= [+ | -] <целое без знака>

  <целое без знака>::= <цифра> { <цифра> }

  <цифра>::= 0|1|2|3|4|5|6|7|8|9

Синтаксические диаграммы



Синтаксис: основные ловушки

- Ограничения на длину идентификаторов
  current_winner_number
  current_width
- Регистр символов
  Number, NUMBER, number
- Комментарии
  - Однострочные
    -- в Ada и PL/SQL, // в C++, C в Fortran
  - Многострочные
    /* … */ в C, { … } и (*…*) в Pascal
  - Пропущенные операторы
    { a:=b+c;
    { Комментарий }

Синтаксис: основные ловушки

- Похожие символы

  Присвоить: **:=** в Pascal, **=** в C и Fortran

  Равно: **=** в Pascal, **==** в C, **.eq.** в Fortran

- Значащие символы, или когда взрываются ракеты?

```
DO 10 I = 1,100    C Это оператор цикла
DO 10 I = 1.100    C Это оператор присваивания
                   C DO10I = 1.100
                   C т.к. пробел игнорируется!
```

Семантика

- Пример: семантика оператора
  if C then S1else S2
  - *Формальное описание*
    (C⇒Семантика S1) & (⌉C⇒Семантика S2)
  - *Неформальное описание (естественный язык)*
    Проверяется условие после if; если результат Истина, то выполняется оператор, следующий после then, иначе выполняется оператор, следующий за else.
- Формальное описание семантики всех конструкций языка может быть очень сложным и, как правило, не применятся.

Семантика языка и правильность программ

- Формальное описание семантики дает возможность доказать *правильность программы* в следующем смысле:
  - оператор – аксиома, описывающая преобразование состояния, для которого верно некоторое входное утверждение, в состояние, для которого верно некоторое выходное утверждение
  - семантика программы "вычисляется" путем построения входных и выходных утверждений на основе утверждений для отдельных операторов
  - результат вычислений – доказательство того, что если входные данные удовлетворяют утверждению на входе, то выходные данные удовлетворяют утверждению на выходе

Типы данных

- *Тип данных* – множество *значений* и множество *операций* над этими значениями.
- Определение множеств значений и операций зависит от языка программирования и его конкретной реализации.
- Примеры:
  - типы Integer в Pascal и int в C – это конечное множество целочисленных значений (в количестве ≅ 65 тыс. или 4 млрд. – в зависимости от компьютера) и конечного набора операций (+, -, *, >, …)
  - тип "массив" – индексированная совокупность элементов некоторого (ранее определенного) типа с операцией индексации; в Ada над массивами определена также операция присваивания.

Переменная, константа

- *Значение* – интуитивно понятный термин.
- *Представление* – строка битов для указанного значения (например: представление целого числа 201 есть 11001001).
- *Переменная* – имя совокупности ячеек, содержащих представление значения указанного типа, которое может изменяться во время выполнения программы.
- *Константа* – имя совокупности ячеек, содержащих представление значения указанного типа, которое *не* может изменяться во время выполнения программы.
- *Литерал* – последовательность символов, которая в программе непосредственно задает значение (например: 201, 2.01, '0', "Строка", TRUE).

Оператор присваивания

- *Операторы* – конструкции языка для управления вычислениями.
- *Оператор присваивания* изменяет значение указанной переменной на указанное новое значение.
  Pascal : A := A+1;
  C : m[a*i] = r[k+2] * a;
- Алгоритм работы оператора присваивания
  1. Вычислить значение выражения в правой части
  2. Вычислить значение выражения в левой части, получить адрес ячейки
  3. Скопировать значение, вычисленное на шаге 1., в ячейки памяти, начиная с адреса, полученного на шаге 2.

Контроль соответствия типов

- *Контроль соответствия типов* – проверка того, что при выполнении присваивания тип выражения совместим с типом адресуемой переменной.
- Подходы к контролю соответствия типов:
  - *Отсутствие контроля*: программист отвечает за последствия выполнения присваивания.
  - *Неявное преобразование* значения выражения к типу адресуемой переменной.
  - *Строгий контроль*: отказ от выполнения присваивания при несовместимости типов.

Управляющие операторы

Используются для изменения порядка выполнения:

- *Условные операторы* позволяют выбрать одну из нескольких альтернативных последовательностей управления.

```
if A>5 then
        Z:=1
else
        Z:=A+B;
```

```
case K of
   0..3 : P:=1;
   7, 9 : P:=233;
   else     P:=0;
end;
```

- *Операторы цикла* позволяют повторить выполнение последовательности операторов.

```
for (i=0; i<N; i++)
        A[i]=0;
```

```
while i<N do begin
                F := i*F;
                i:=i+1;
end;
```

- *Безусловный переход*: goto, break, continue.

Подпрограммы

- *Подпрограмма* – сегмент программы, состоящий из объявлений данных и операторов, которые можно многократно *вызывать* (call) из различных частей программы.
- *Параметры подпрограммы* – последовательность значений, передаваемая в подпрограмму при ее вызове для задания различных вариантов выполнения, передачи исходных данных и получения результатов.

Подпрограммы

- Подпрограмма – важный элемент программной структуры.
  - Каждая подзадача программы должна быть оформлена в виде подпрограммы.
  - Подпрограммы позволяют разделить программу на небольшие, хорошо понятные и читаемые части.
- Виды подпрограмм:
  - Pascal : процедуры (procedure), функции (function)
  - C : функции
  - FORTRAN : суб-рутины (SUBROUTINE)
  - ОО-языки : методы

Библиотеки подпрограмм

- *Библиотека* – вспомогательная неисполняемая программная единица, содержащая определения подпрограмм и данных, которые могут быть вызваны из основной программы.
- Библиотеки необходимы для коллективной разработки больших программных систем (размером от 1 тыс. строк).
- Примеры библиотек:
  - Pascal : нет языковых средств
  - Turbo Pascal : unit
  - Modula : module
  - C : только средства раздельной трансляции
  - Ada : package

Стандартизация языков

- *Стандарт языка* – официальный документ одной из международных организаций по стандартам (ISO – Int. Standards Org., ANSI – Amer. Nat. Standards Inst. и др.), в котором зафиксированы алфавит, синтаксис и семантика языка.
- *Мобильная программа* выполняется одинаково на различных компьютерах и/или операционных системах.
- Мобильность программ возможна, если для языка существует стандарт и различные трансляторы языка поддерживают данный стандарт.

Критерии оценки языков программирования

- Читабельность (легкость чтения и понимания программ) и ортогональность (легкость и удобство языка для создания программ)
- Надежность – выполнение программой предназначенных ей действий при любых условиях
- Среда программирования
- Стоимость

Читабельность

- Простота
  - Количество элементарных конструкций
  - Множественность свойств
  - Перегрузка операторов
- Ортогональность – наличие относительно небольшого количества элементарных конструкций, с помощью которых выражаются управляющие операторы и структуры данных
- Управляющие операторы и оператор goto
- Адекватные средства определения типов и структур данных
- Синтаксис
  - Правила образования идентификаторов
  - Написание и семантика ключевых слов

Легкость создания, надежность

- Легкость
  - Простота и ортогональность
  - Поддержка абстракции
  - Выразительность
- Надежность
  - Проверка совместимости типов при компиляции
  - Средства обработки исключительных ситуаций
  - Совмещение имен (несколько имен для одной и той же ячейки памяти)
  - Легкость чтения и использования

Стоимость

- Затраты на обучение программистов
  - Простота и ортогональность языка, опыт человека
- Затраты на создание программ
  - Легкость и удобство языка для конкретного приложения, наличие среды разработки
- Затраты на разработку компилятора
- Стоимость и эффективность выполнения программ

Критерии классификации языков

- *Поколение* языка
- *Уровень абстракции* языка
  Абстракция – выделение существенных и отбрасывание несущественных в данном контексте характеристик объекта.
- *Парадигма* языка
  Парадигма – базовая модель постановки задач и их решения.
- *Назначение* языка
- …

Классификация языков (по уровню абстракции)

Языки программирования

Языки высокого уровня

- Процедурные языки (Fortran, Cobol, PL/1, Algol, Pascal, C, Ada, …)
- Языки обработки данных (LISP, APL, Snobol, Icon, SETL, …)
- OO-языки (Smalltalk, Simula, Eiffel, C++, Ada95, …)
- …

Языки низкого уровня

- Машинные языки
- Языки ассемблеров

Классификация языков (по уровню абстракции)

- *Языки высокого уровня*
  - Наличие понятия *типа данных*.
  - Независимость от архитектуры конкретного компьютера (*мобильность программ*).
  - Развитые управляющие структуры и средства описания структур данных.
  - Близость к естественному языку.
- *Языки низкого уровня*
  - Отсутствие понятия *типа данных*.
  - Зависимость от архитектуры конкретного компьютера (отсутствует *мобильность программ*).
  - Примитивные управляющие структуры и средства описания структур данных.
  - Близость к машинному языку.

Пример: вычисление факториала

Машинный язык	Язык ассемблера	Язык высокого уровня Паскаль

10111010
00001100
00000001
00101110
10001001
00010110
10010001
00000010
10001011
00011110
00101100
00000000
10001110
11011010
10100011 …
	mov AX,1
	mov BX,N
	@2:
	cmp BX,0
	je @1
	imul BX
	dec BX
	jmp @2
	@1:
		function Factorial (N: **Integer**) : **Integer**;
		var
		i, F: **Integer**;
		begin
		F:=1;
		for i:=1 **to** N **do**
		F:=F*i;
		Fact:=F;
		end;

Классификация языков (по парадигме)

Языки программирования *

Императивные языки

- Процедурные языки

Fortran, Cobol, PL/1, Algol, Pascal, C, Ada, …

- ОО-языки

Smalltalk, Simula, Eiffel, C++, Ada95, …

Декларативные языки

- Функциональные языки

Lisp, ML, Haskell, …

- Логические языки

Prolog, …

Параллельные языки

occam, Ada95, Linda, Parlog, Lisp2D

* Язык может сочетать в себе более одной парадигмы

Классификация языков (по парадигме)

Императивный язык рассматривает программу как описание последовательности действий по получению искомого результата.

- *Процедурно-ориентированный* язык рассматривает программу как совокупность подпрограмм (процедур), обменивающихся данными в процессе их (подпрограмм) вызова из основной программы.
- *Объектно-ориентированный* язык рассматривает программу как совокупность объектов (имитирующих объекты реального мира), обменивающихся сообщениями в процессе вызова их (объектов) методов в основной программе.

Классификация языков (по парадигме)

Декларативный язык рассматривает программу как совокупность описания входных данных и описания искомого результата.

- *Функциональный* язык рассматривает программу как совокупность определений функций, которые обмениваются между собой данными без использования промежуточных переменных и присваиваний.
- *Логический* язык рассматривает программу как описание задачи в терминах фактов и логических формул, а решение задачи выполняет система с помощью механизмов логического вывода.

Классификация языков (по парадигме)

Параллельный язык предполагает использование в программе явных конструкций для параллельного исполнения фрагментов последовательности действий по получению искомого результата.

Пример: вычисление факториала

Императивный процедурный язык Паскаль	Декларативный логический язык ПРОЛОГ

function Factorial (N: **Integer**): **Integer**;
var
  i, F: **Integer**;
begin
  F:=1;
  for i:=1 **to** N **do**
  F:=F*i;
  Fact:=F;
end;
	Factorial (0, 1)**.**
	Factorial (1, 1)**.**
	Factorial (F, N) **:-**
	  N1 **is** N-1,
	  N > 1,
	  Factorial (F1, N1),
	  F **is** N*F1**.**

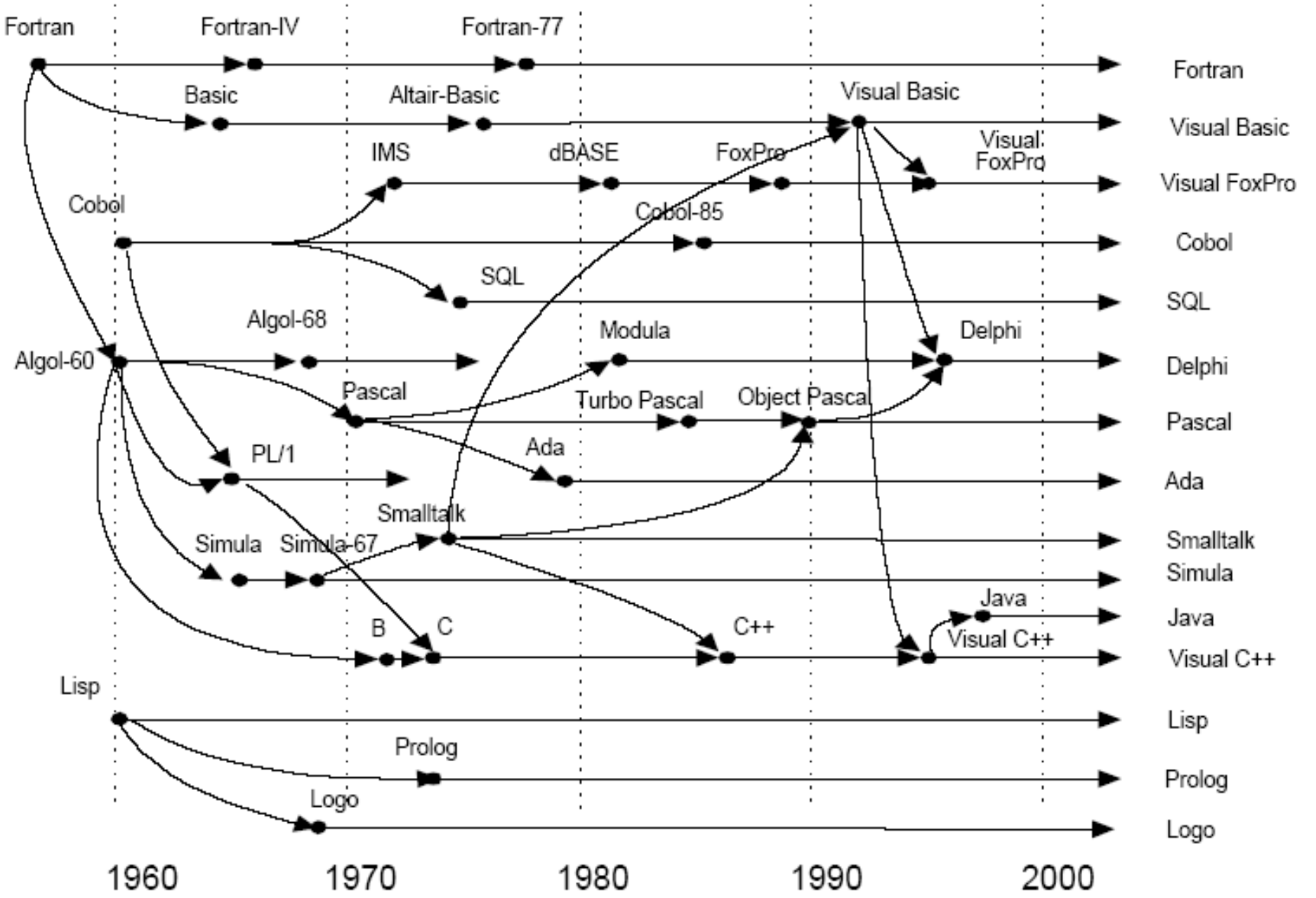
Эволюция языков программирования

Эволюция: язык Plankalkul

Конрад Цузе
(1910-1995)

- Простейший тип данных – бит, на основе которого создавались типы для представления целых чисел и чисел с плавающей точкой.
- Язык содержал:
  - массивы и записи (допускалась рекурсия при определении записей);
  - for-подобный оператор цикла;
  - условный оператор без части else.
- Язык не имел оператора goto.
- Пример оператора A[5]:=A[4]+1

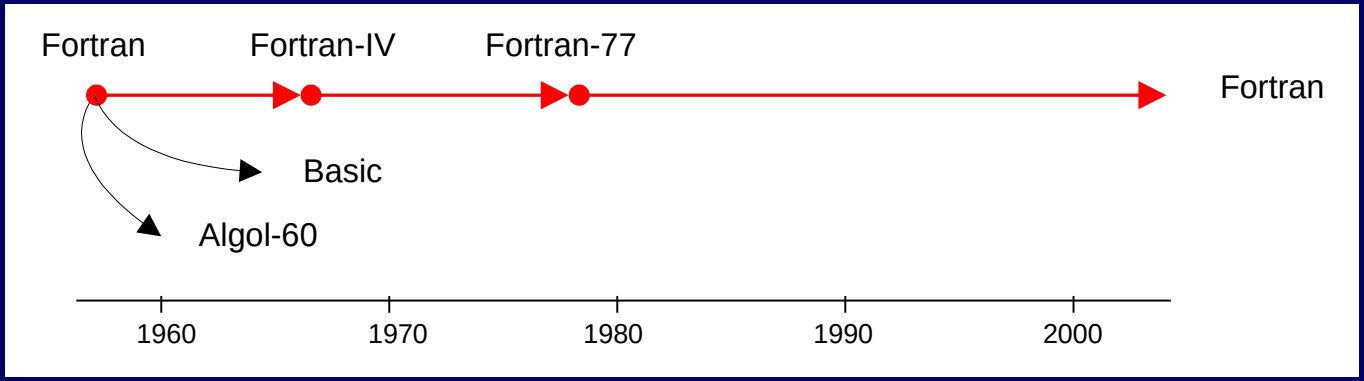
```
    |  A  +  1  =>  A        операторная строка
V   |  4           1        строка индексов
S   |  1.n         1.n      строка типов (n битов)
```

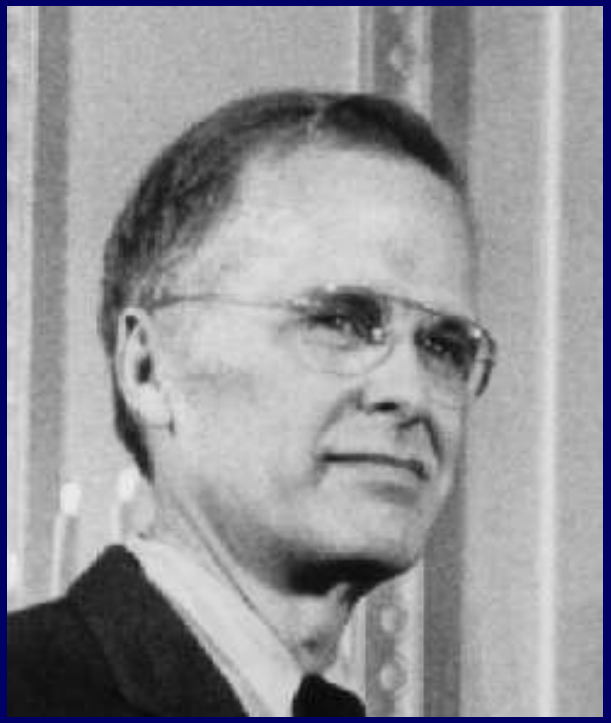


Генеалогическое дерево языков программирования высокого уровня

Эволюция: универсальные языки

Бессмертный Fortran





Fortran = FORmula TRANslator

Первый высокоуровневый язык программирования Fortran был разработан в фирме IBM под руководством Джона Бэкуса (Backus, John; p. 1924).

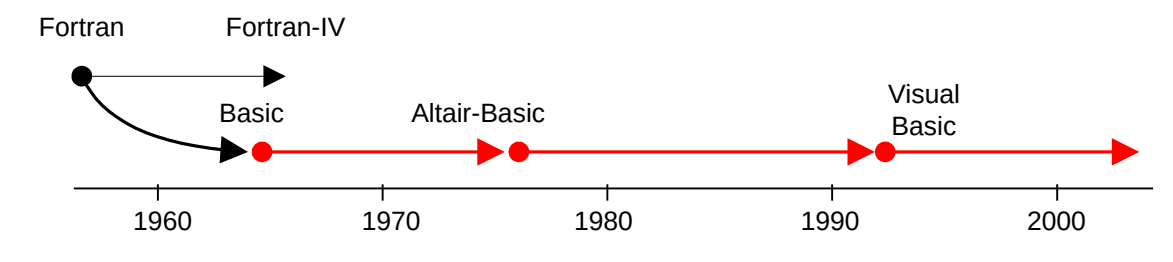
Работа над языком началась в 1954 г., первая реализация для IBM 704 выполнена в 1957 г.

Бессмертный Fortran

```
C                MAIN PROGRAM
          101    FORMAT(208)
          102    FORMAT(//'N=',15, 5X, 'R=', 15
                       1//6X, 'M', 5X, PROB)
          103    FORMAT(18, F14.10)
          201    READ(1,101) N, IR
                 WRITE(3,102) N, IR
                 IF(N) 202, 202, 203
          202    STOP
          203    IF(IR) 202, 202, 204
          204    M=O
                 P=COMBF(N,M)*COMBF(IR-1,N-M-1)
                 1/COMBF(N+IR-1,IR)     ...
```

Фрагмент программы на языке Fortran

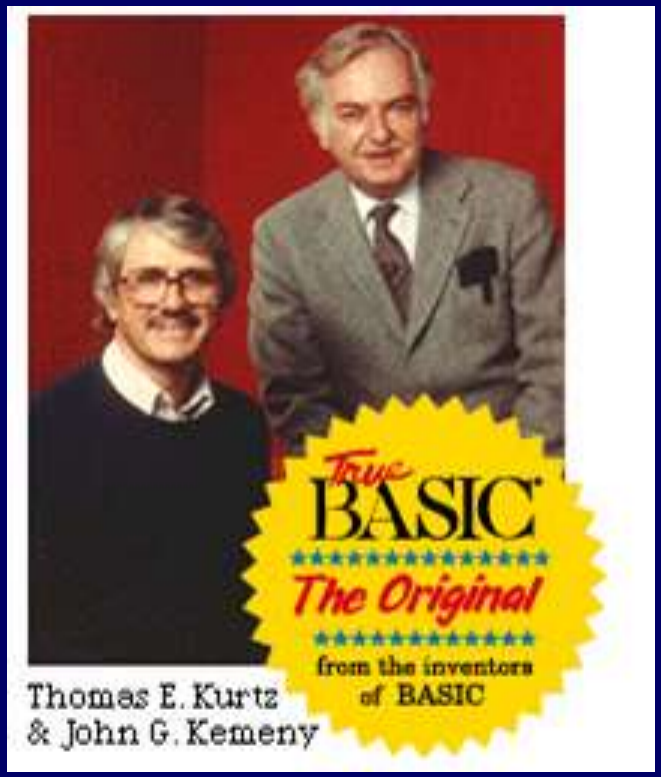
Basic – язык для начинающих



BASIC = Beginners All-purpuse Symbolic Instruction Code

Язык Basic был разработан в 1964 г. в Дармутском колледже в г. Хановере (Darmouth College, Hanover), штат Нью-Хемпшир

Basic – язык для начинающих

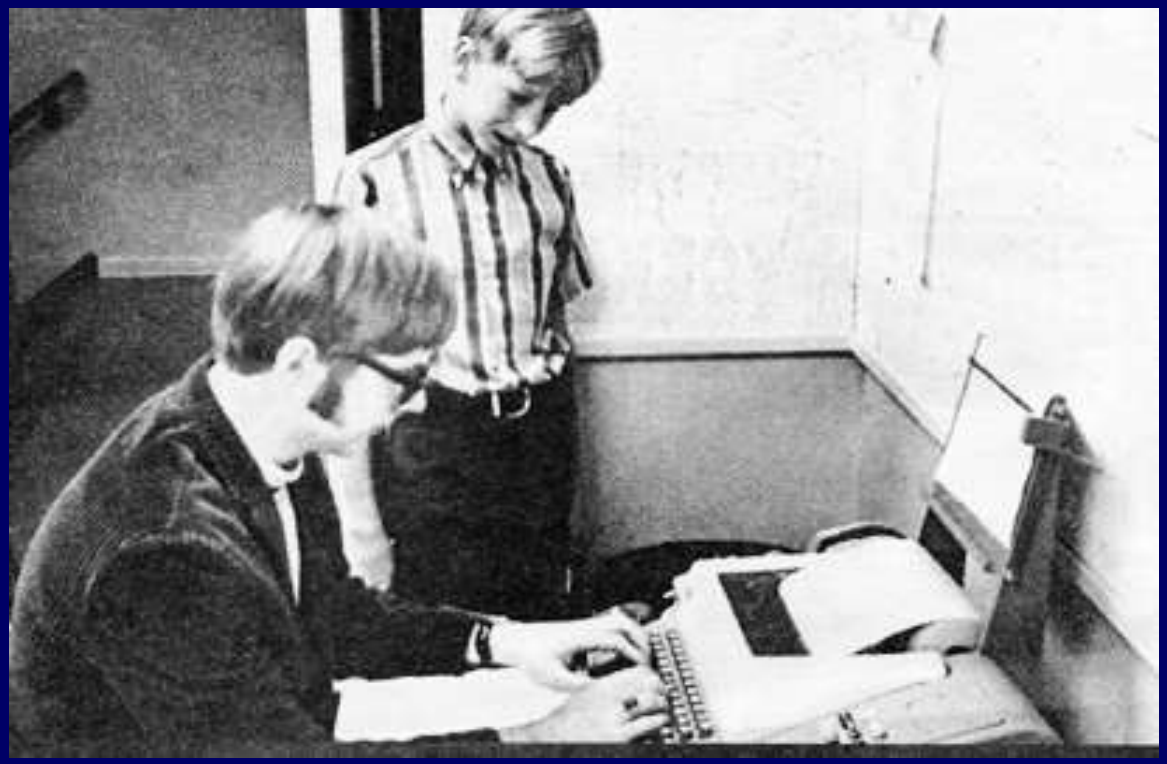


Авторы языка Basic.
Стоит Джон Кемени
(Kemeny, John G.; 1926-1993),
сидит Томас Курц
(Kurtz, Thomas E.; p. 1928)

```
10 dim A(5)
20 for i=1 to 5
30 input A(i)
40 next i
50 if i=5 then goto 140
60 if A(i)<=A(i+1) then goto 90
70 i=i+1
80 goto 130
90 z=A(i)
100 A(i)=A(i+1)
110 A(i+1)=z
120 i=1
130 goto 50
140 for i=1 to 5
150 print A(i)
160 next i
```

Простейшая программа на языке Basic

Basic – язык для начинающих



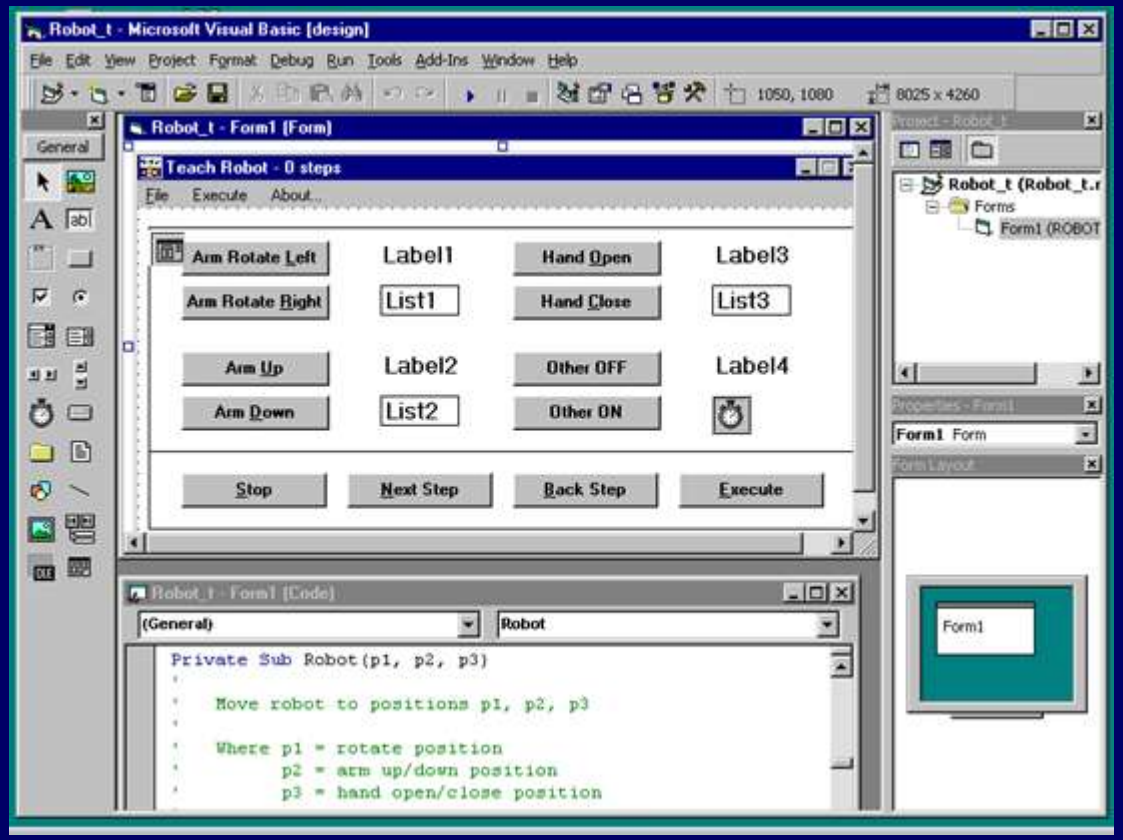
Будущие создатели Microsoft Пол Аллен (Allen, Paul; p. 1954) и Билл Гейтс (Gates, William; p. 1955) познакомились с Бэйсиком, работая в компьютерном классе школы в Сиэтле (снимок 1968 г.)

Basic – язык для начинающих

Начав с Бэйсика, компания Microsoft превратилась в крупнейшую софтверную империю, а Билл Гейтс –стал самым богатым человеком на планете

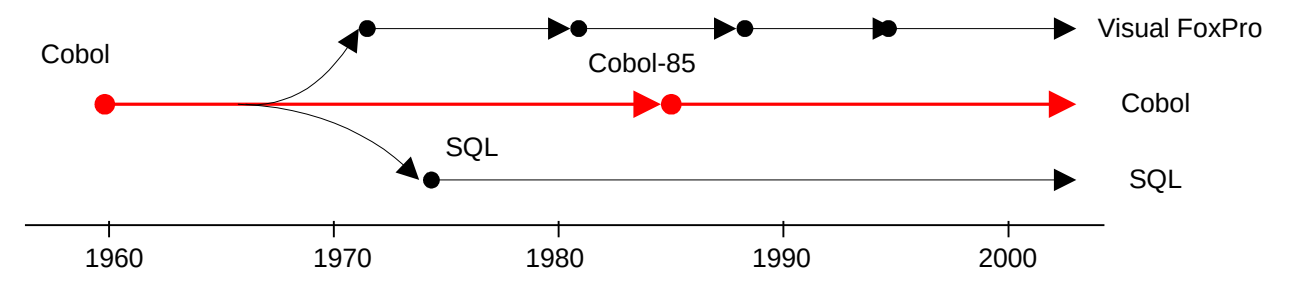
Штаб - квартира корпорации Microsoft в Редмонде (пригород Сиэтла)

Basic – язык для начинающих



На протяжении нескольких десятилетий Visual Basic оставался фирменный языком компании Microsoft. В начале 1990-х годов он стал объектным и приобрел средства визуального проектирования

Cobol – язык для бухгалтеров



COBOL = COmmon Business-Oriented Language

На фото: разработчики языка Cobol у шуточного обелиска, присланного в их адрес в качестве намека на безнадежно медленную работу, способную похоронить саму идею. Справа внизу – Грейс Хоппер

Cobol – язык для бухгалтеров

Основные свойства языка Cobol:

- независимость программ от оборудования;
- независимость программ от данных;
- сложные структуры данных;
- синтаксис, приближенный к естественному английскому языку.

Cobol – язык для бухгалтеров

```
1010 IDENTIFICATION DIVISION.
1020 PROGRAM-ID “EXAMPLE”.
1030 ENVIROMENT DIVISION.
1040 INPUT-OUTPUT SECTION.
1050 FILE-CONTROL.
1060   SELECT CD ASSIGN TO “SYS010” UNIT-RECORD 2540R.
1070   SELECT TT ASSIGN TO “SYS009” UTILITY 2400.
1080 DATA DIVISION.
1090 FILE SECTION.
1100 FD    CD          DATA RECORD IS C
1110                   LABEL RECORDS ARE OMITTED.
1120 01    C.
1130         02   C1  PICTURE 9(4).
1140         02   C2  PICTURE 9.
1150         02   C3  PICTURE X(70).
...
```

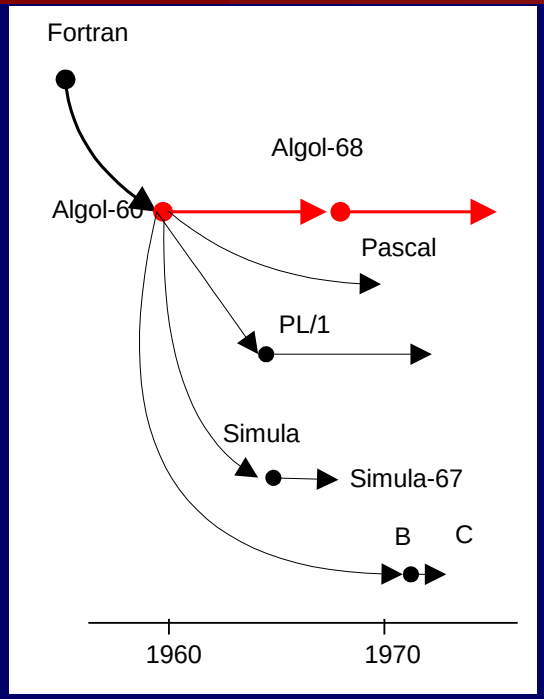
Программа на Коболе (начало)

Cobol – язык для бухгалтеров

```
1290 PROCEDURE DIVISION.
1300 P1.    OPEN INPUT CD, OUTPUT TT.
1310 P2.    READ CD, AT END GO TO P3.
1320                MOVE C1 TO D1.
1330                MONE C2 TO D2.
1340                MOVE C3 TO D3.
1350                ADD C1, C2, GIVING D4.
1360                WRITE T FROM D.
1370                GO TO P2.
1380 P3.    CLOSE SD, TT.
1390                STOP RUN.
```

Программа на Коболе (окончание)

Algol и его влияние



ALGOL = ALGOritmic Language

В 1958 году в Цюрихе (Швейцария) состоялась международная конференция, предложившая проект нового универсального международного языка программирования Algol-58. В 1960 году на парижской конференции была принята окончательная версия под названием Algol-60.

На снимке: участники парижской конференции голосуют за Алгол-60.

Algol и его влияние

Основные свойства языка Algol-60:

- машинная независимость;
- формальный синтаксис;
- описание переменных и блочная структура;
- рекурсия

Нормальная форма Бэкуса-Наура (БНФ)

<цифра>::= 1|2|3|4|5|6|7|8|9|0

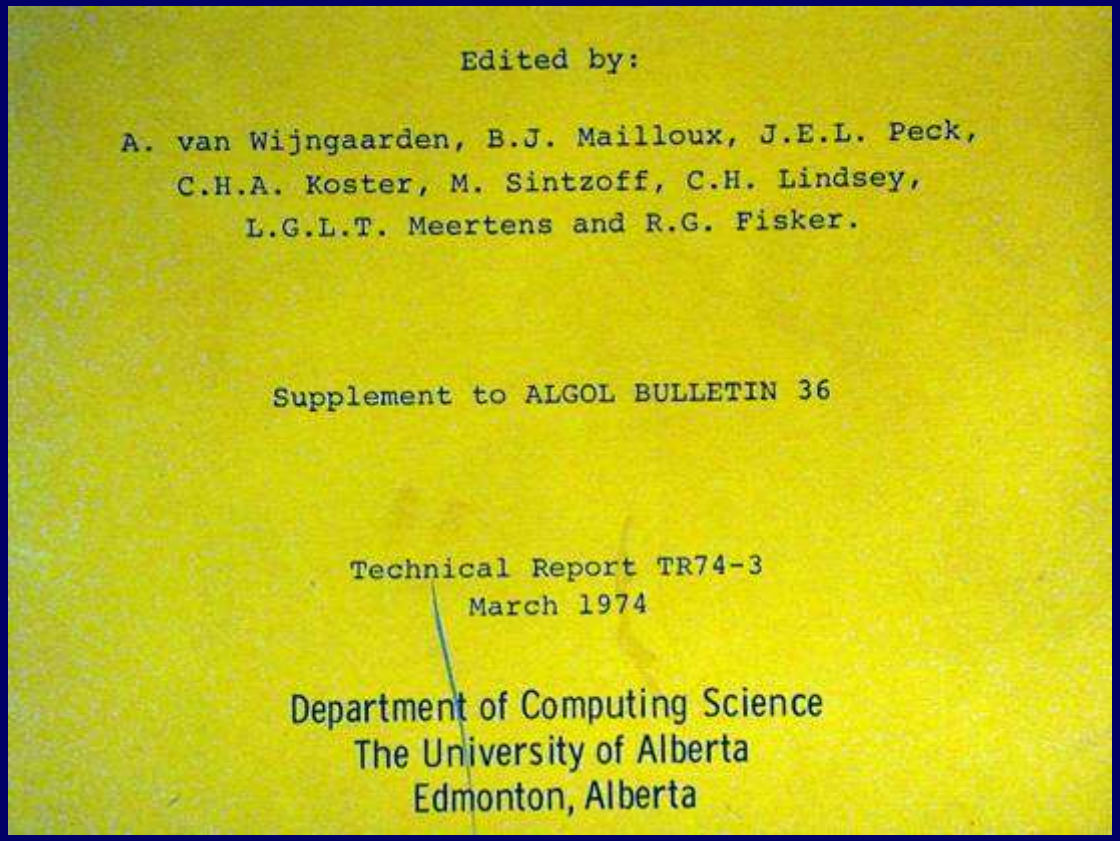
<целое без знака>::= <цифра>| <цифра> <целое без знака>

Algol и его влияние

```
begin
        integer i, n;
        real s;
        real array x[1:n];
        s:=0;
        for i:=1 step 1 to n do
          s:=s+x[i];
        s:=s/n
end
```

Простейшая программа на Алголе-60, вычисляющая среднее арифметическое n чисел.
Синтаксис Алгола-60 сформировал стандарт для всех последующих языков программирования

Algol и его влияние



Edited by:

A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck,
C.H.A. Koster, M. Sintzoff, C.H. Lindsey,
L.G.L.T. Meertens and R.G. Fisker.

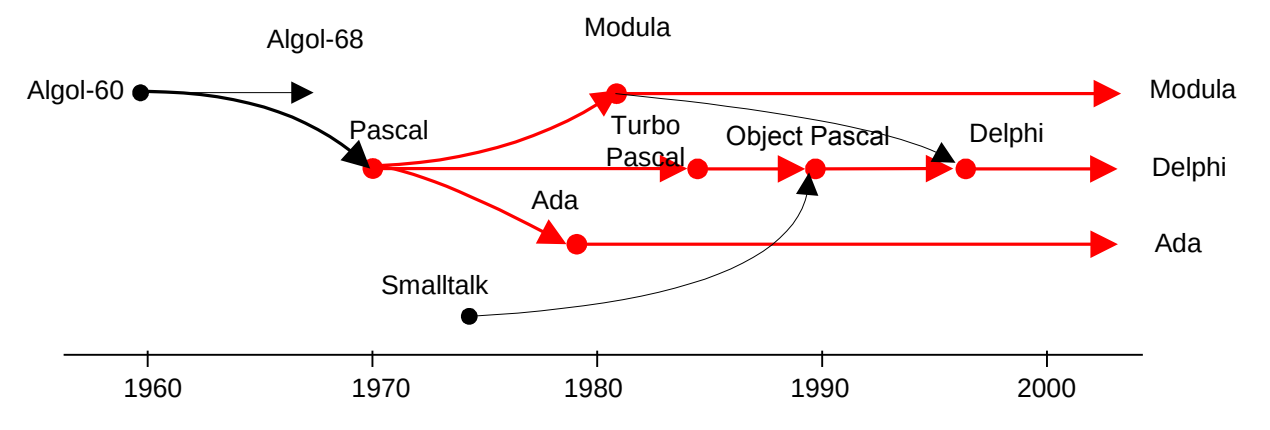
Supplement to ALGOL BULLETIN 36

Technical Report TR74-3
March 1974

Department of Computing Science
The University of Alberta
Edmonton, Alberta

В результате многолетней переработки Алгола-60 комитетом IFIP появился язык Алгол-68 (пересмотренное сообщение под ред. А. ван Вейнгаардена (A. van Wijngaarden) и др. опубликовано в 1975 г.)

Pascal и его потомки



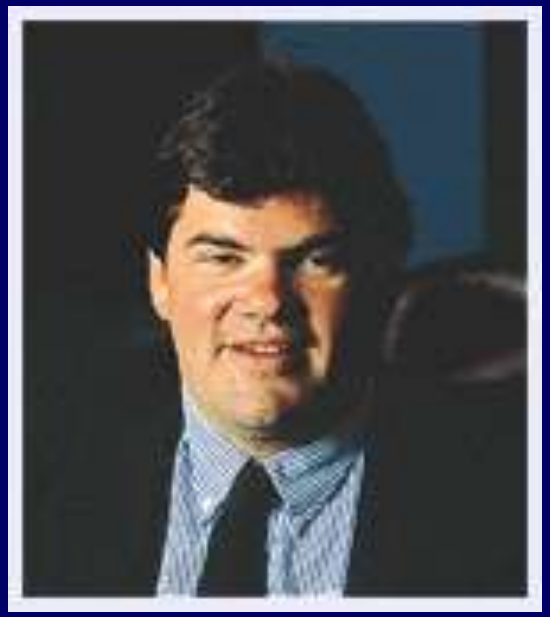
Член комитета по Алголу-68 Никлаус Вирт (Wirth, Niklaus; p. 1934) был против принятия переусложненного стандарта.
В знак доказательства своей правоты он разработал в 1971 г. простой и ясный алголоподобный язык, предназначенный прежде всего для обучения студентов в Федеральном техническом университете в Швейцарии. В честь изобретателя первой вычислительной машины Вирт назвал язык Паскалем.

Pascal и его потомки

```
var
        i, n: integer;
        s: float;
        x: array[1..n] of real;
begin
        s:=0;
        for i:=1 to n do
          s:=s+x[i];
        s:=s/n
end.
```

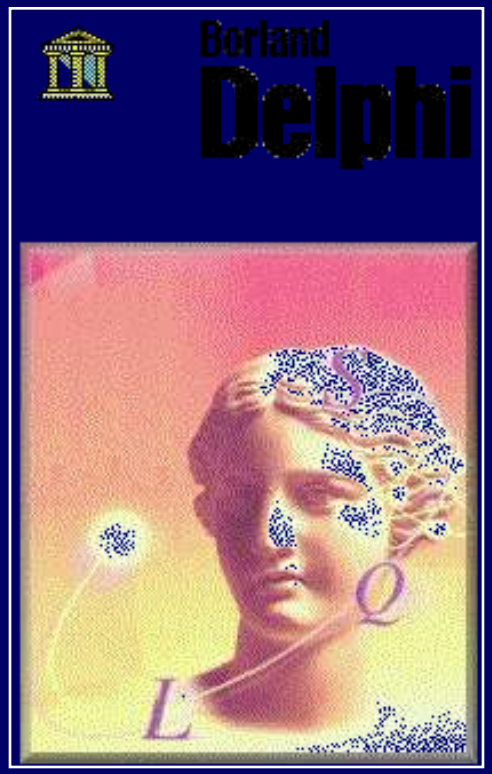
Программа на Паскале, вычисляющая среднее арифметическое n чисел

Pascal и его потомки



Новую жизнь языку Pascal дал Филипп Кан (Kahn, Philippe; p. 1938) – создатель компилятора Turbo Pascal для IBM PC и основатель компании Borland (1984 г.)

Pascal и его потомки



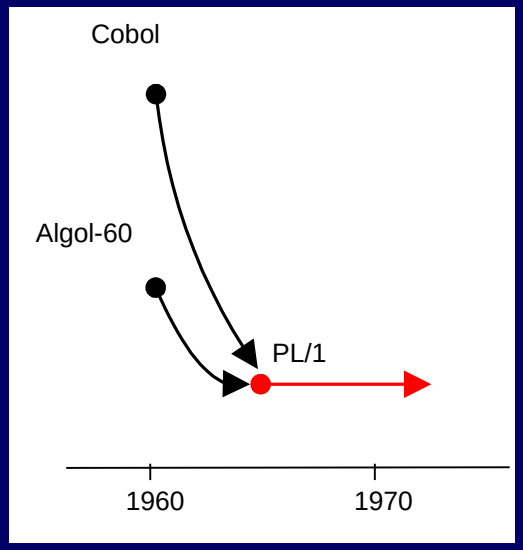
Среда разработки Delphi фирмы Borland объединила передовые достижения технологии программирования: объектное расширение языка Pascal, визуально- событийное проектирование, модульное структурирование и раздельная компиляция.

В отличие от учебного Паскаля, язык программирования Modula-2, предложенные Никлаусом Виртом, изначально предназначался для профессионального применения

Pascal и его потомки

В 1975 году Министерство обороны США приняло решение разработать стандартный язык для программирования сложных и ответственных военных приложений. Был объявлен широкий международный конкурс, в котором приняли участие 15 групп разработчиков. В результате нескольких туров в мае 1979 года выявился победитель — французская фирма С.I.I., руководитель проекта Жан Ихбиа (Ichbiah, Jean).
Снимок сделан на II конференции по истории языков программирования, 1993 г.

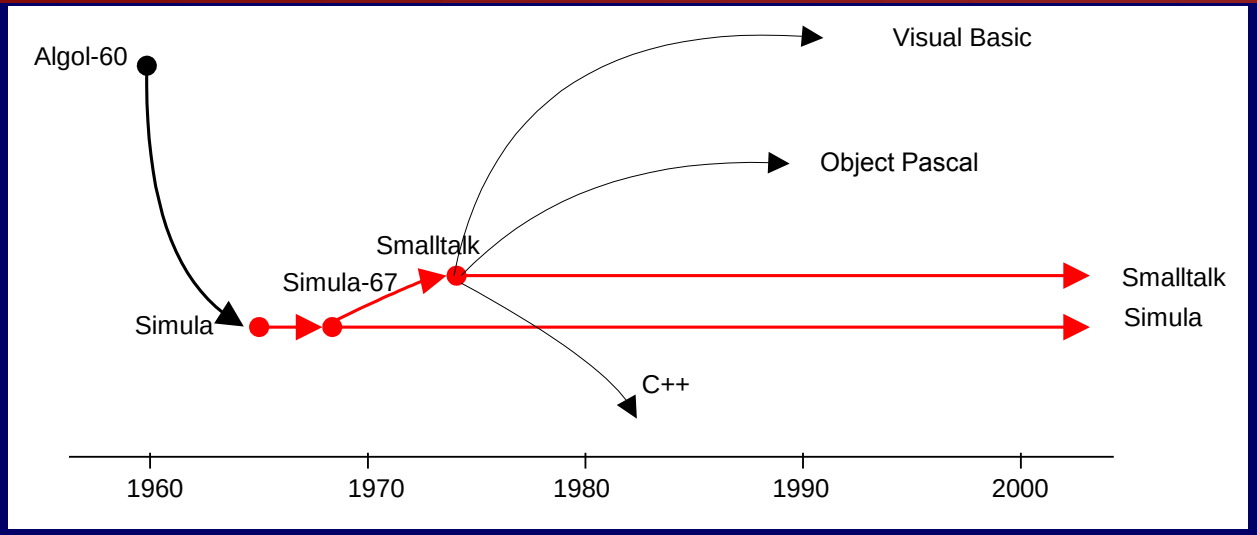
Суперязык PL/1



PL/1 = Programming Language One
Язык PL/1 был частью амбициозного проекта IBM S/360, он создавался в спешке и представлял собой механическую смесь идей из многих языков. Критики сравнивали его с елкой со множеством украшений.

```
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;
             PUT LIST (X1, Y1, X2, Y2);
             GO TO P1;
ENDING:;
END EXAMPLE;
```

Simula и Smalltalk – революция



Simula = SIMULAlation

За разрабртку языка Simula Кристен Нигорд (Nygaard, Kristen; 1926-2002), на снимке слева, и Оле-Йохан Дал (Dahl, Ole-Johan; 1931-2002) были удостоены высшей награды компьютерного сообщества – медали Тьюринга

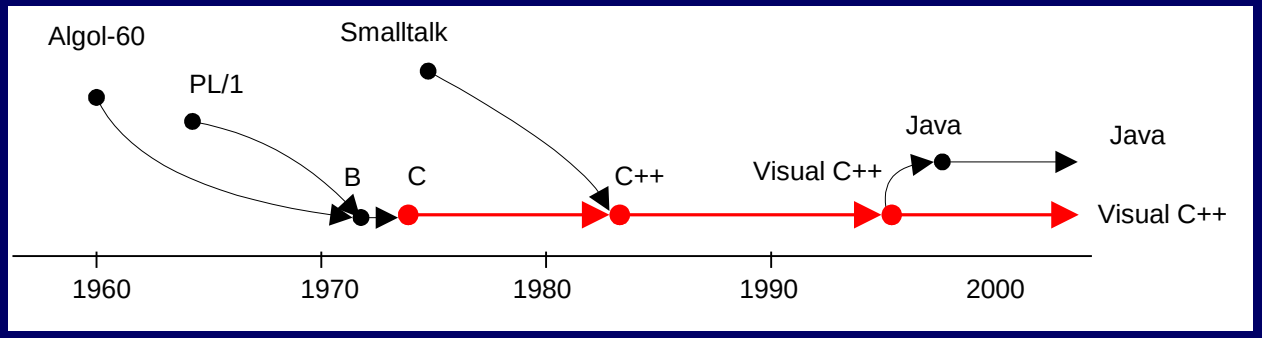
Simula и Smalltalk – революция

```
|a|
a := Array new: 5.
1 to: 5 do: [:i | a at: i put:
  (Prompter prompt: 'Введите элемент массива')
asNumber].
a := a asSortedCollection.
a do: [:i | Transcript putAll: i printString].
```

Простейшая программа на Smalltalk, вычисляющая среднее арифметическое пяти чисел

Алан Кей

C – язык для профессионалов



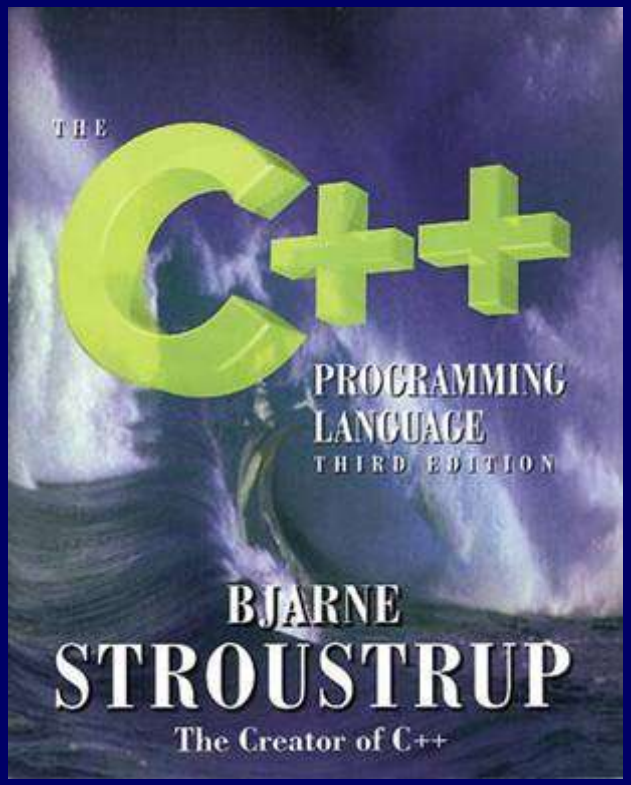
Язык Си (C) был создан Деннисом Ричи (Ritchie, Dennis M.; p. 1941) в 1973 году в Bell Labs в ходе разработки операционной системы UNIX. Он развивал язык Би (B), который основывался на созданном в Кембриджском университете языке BCPL (от Basic Combined Programming Language), который в свою очередь был потомком Алгола-60

C – язык для профессионалов

```
float A[5];
  for(int
i=0;i<5;i++)scanf("%f",&A[i]);
  i=0;
  while(i<4){
    if(A[i]<=A[i+1])i++;
    else{
      z=A[i];
      A[i]=A[i+1];
      A[i+1]:=z;
      i=0;
    }
  };
  for(i=0;i<5;i++)printf("%f\n",A[i]);
```

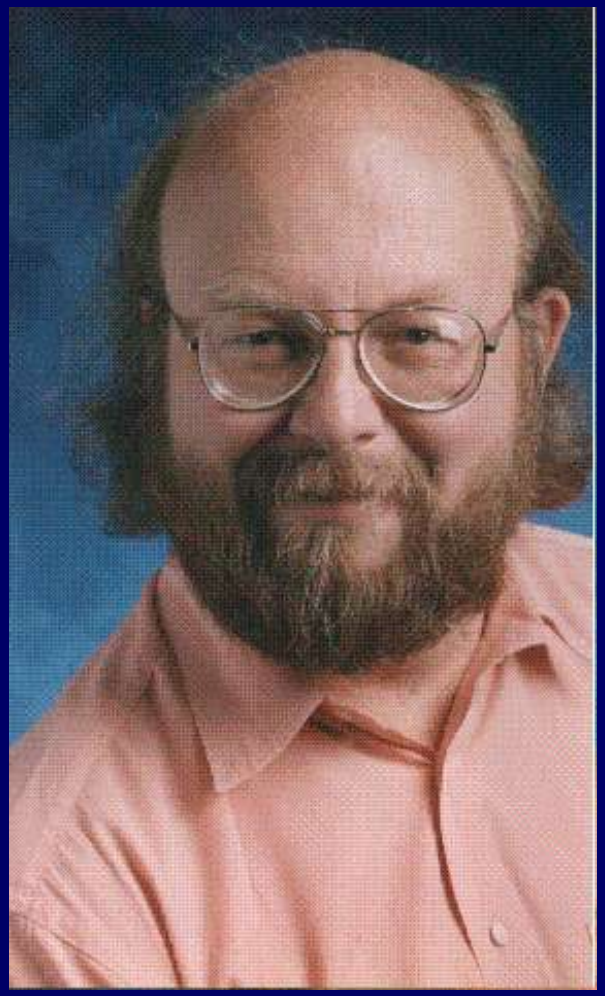
Текст на языке С отличается лаконичностью

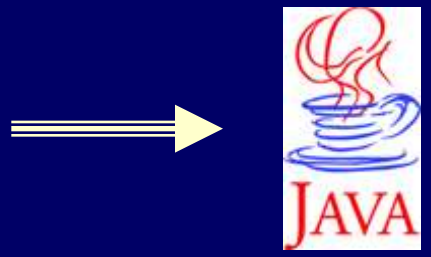
C – язык для профессионалов



Бьярн Страуструп (Stroustrup, Bjarne; p. 1950) ввел в язык C объекты и превратил его в C++

Java – дитя интернета

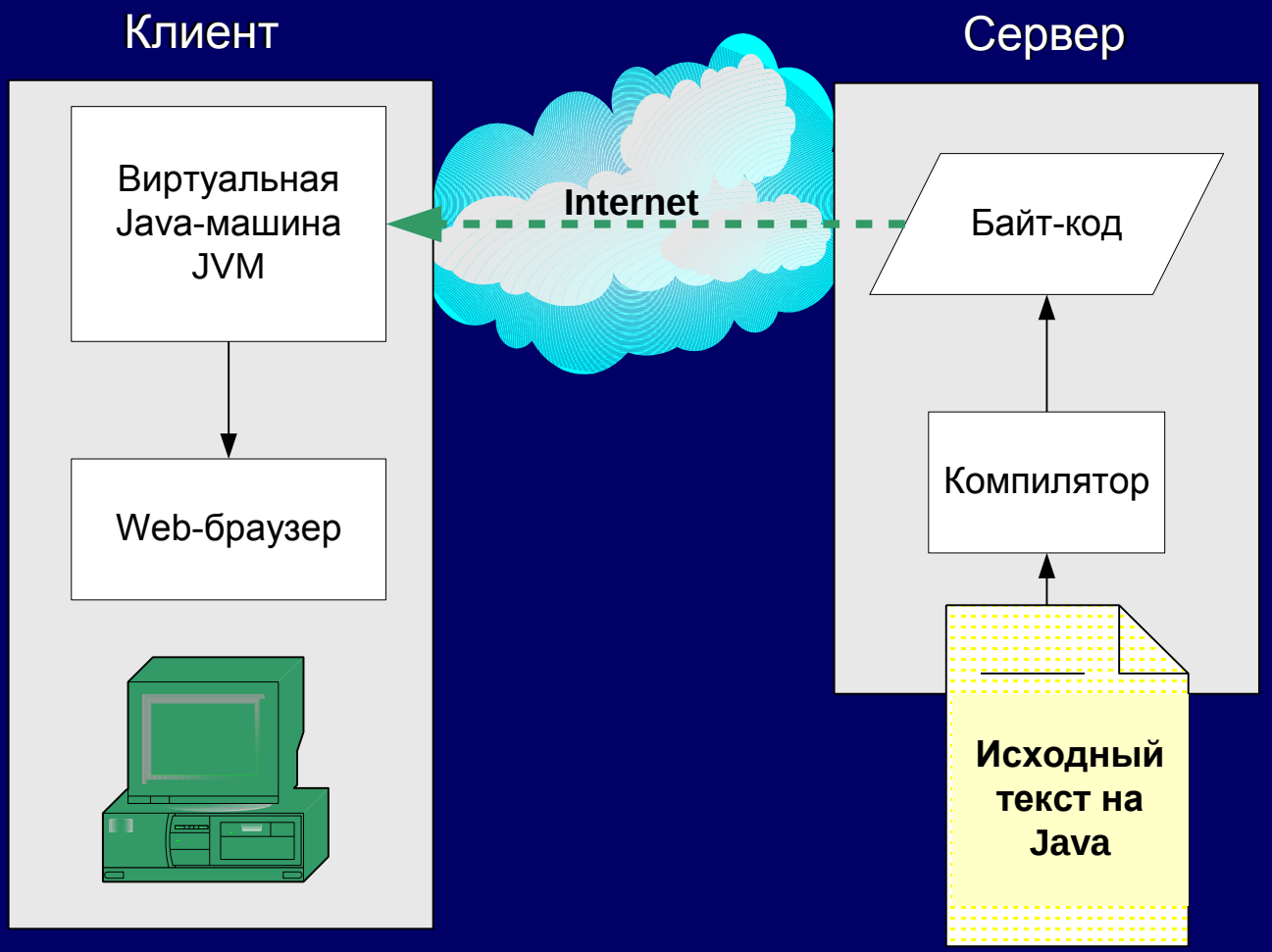




В 1995 г. фирма Sun Microsystems представила язык Java для программирования в интернете.
Он возник в ходе реализации проекта Oak («Дуб»), целью которого было создание системы программирования бытовых микропроцессорных устройств.

Джеймс Гослинг (Gosling, James) – автор Java.

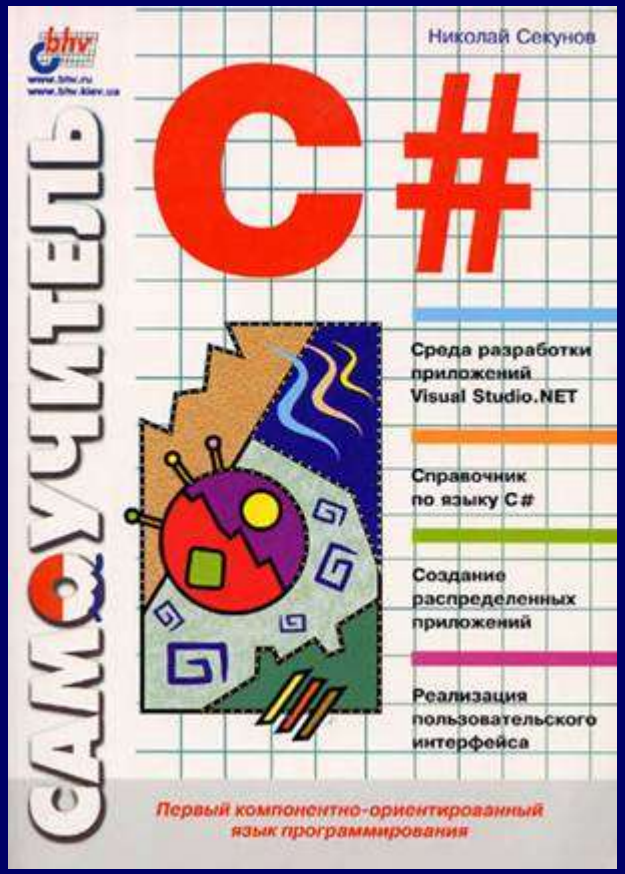
Java – дитя интернета



Java – дитя интернета

```
class test
{
  int i, n;
  float s;
  float x[n];
  public static void main( String
args[] )
  {
    n = 10;
    s = 0;
    for( i=1; i<=n; i++)
    {
      s = s + x[i-1];
      s = s / n;
    }
  }
```

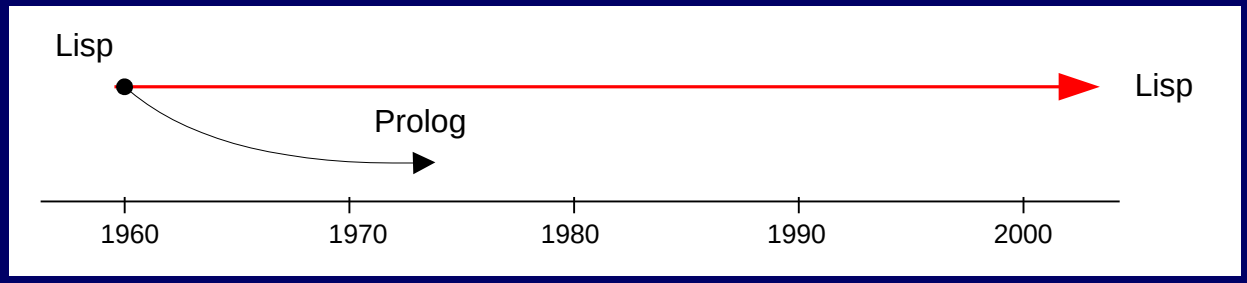
Язык Java основан на C++



В качестве альтернативы Java корпорация Microsoft предложила язык C# (Си-шарп)

Эволюция: функциональные языки

Долгожитель Lisp



Дж. Маккарти и А.П. Ершов
Снимок 1975 г.

Lisp = LISt Processing

Язык Lisp создан в 1960 году Джоном Маккарти (McCarthy, John; p. 1927) в Массачусетском технологи-ческом институте на теоретическом фундаменте лямбда-исчисления, пред-ложенного еще в 1930 году известным американским логиком Алонзо Черчем.

Долгожитель Lisp

```
(setq L `(8 5 13 11 10))
(defun sum (L)
  (cond ((null L) '0)
    (t (add (car L) (sum (cdr L))))
  )
)
(div (sum L) '5)
```

Примитивы:

cond — условная функция, проверяющая с помощью функции null пустоту списка;

add — суммирование аргументов;

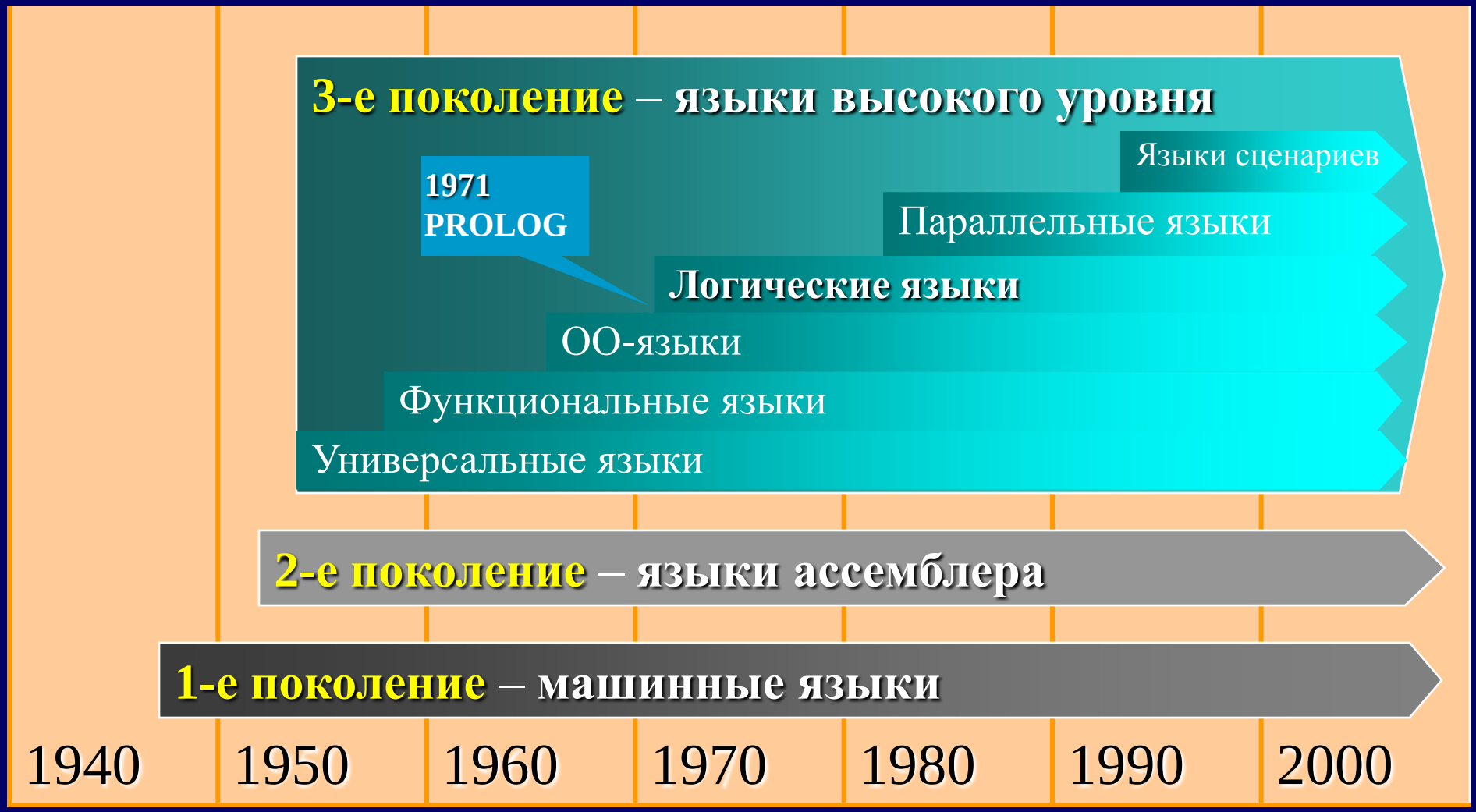
car — извлечение первого элемента из списка;

cdr — извлечение остатка списка (без первого элемента).

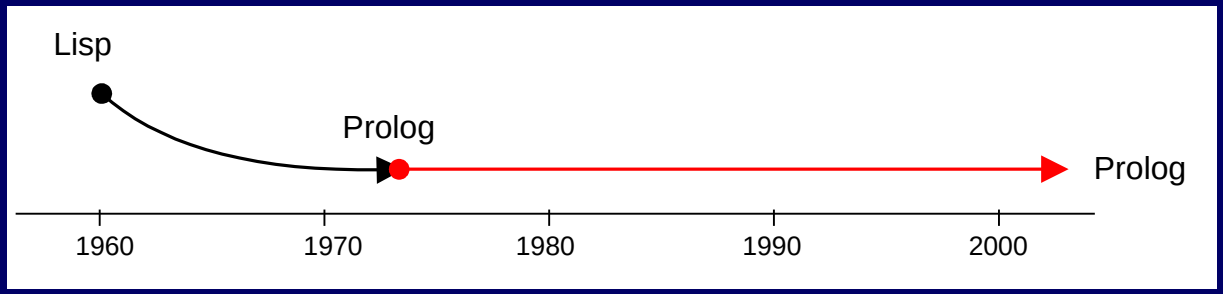
Программа на Lisp имеет специфический вид из-за обилия скобок. За это студенты прозвали его «Lots of Infuriating & Silly Parenthesis» - «Множество раздражающих и глупых скобок»

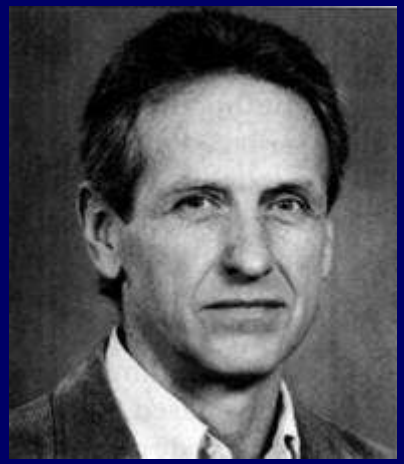
Эволюция: ОО-языки

Эволюция: логические языки



Prolog – несостоявшаяся мечта





Prolog = PROgramming for LOGic
Теоретические основы языка были разработаны Робертом Ковальским (Kowalski, Robert) в Эдинбургском университете (Шотландия) в конце 1960-х годов

Первая практическая реализация языка осуществлена Аленом Кольмари (Colmerauer, Alain) в Марсельском университете (Франция) в 1972 г.

Prolog – несостоявшаяся мечта

Факты:

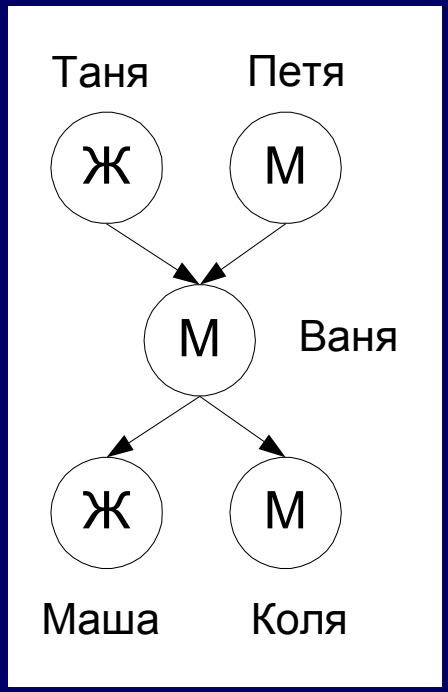
муж (петя), муж (ваня),
муж (коля), жен (таня), жен (маша),
мать (ваня, таня), отец (ваня, петя),
отец (маша, ваня), отец (коля, ваня).

Правила вывода:

родитель (X, Y) :— отец (X, Y)
родитель (X, Y) :— мать (X, Y)
дед (X, Y) :— родитель (X, Z), отец (Z, Y)
брат (X, Y) :— муж (Y), родитель (X, Z),
родитель (Y, Z), X<>Y

Примеры диалога:

GOAL> дед (коля, X) *Кто дед Коли?*
X = Петя
GOAL> брат (маша, X) *Кто брат Маши?*
X = Коля



Описание предметной области семейных отношений на языке Prolog

Prolog – несостоявшаяся мечта

Концептуальные отличия ЭВМ V поколения:

- новая технология производства микросхем, знаменующая переход от кремния к арсениду галлия, и дающая возможность на порядок повысить быстродействие основных логических элементов;
- новая архитектура (не фон-неймановская);
- новые способы ввода-вывода информации — распознавание и синтез речи и образов;
- отказ от традиционных алгоритмических языков программирования (Фортран, Алгол и т. п.) в пользу декларативных;
- ориентация на задачи искусственного интеллекта с автоматическим поиском решения на основе логического вывода.

Эволюция: параллельные языки

Эволюция: языки сценариев