Введение в языки программирования

Язык формирует особый способ нашего мышления и предопределяет, о чем мы можем думать.

Б. Л. Ворф

Компьютерные науки © М.Л. Цымблер

Содержание

- Понятие языка программирования
- Эволюция языков программирования
- Классификация языков программирования
- Основные элементы языков программирования

Компьютерные науки © М.Л. Цымблер 2

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 (*ассемблер*, *компилятор*, *интерпретатор*).

Компьютерные науки © М.Л. Цымблер 3

Язык программирования

- *ЯП = алфавит + синтаксис + семантика*.
- *Алфавит* – символы для записи конструкций языка.
- *Синтаксис* – правила записи конструкций языка (какие конструкции принадлежат языку).
- *Семантика* – смысл конструкций языка (как конструкции должны отрабатываться компьютером).

Компьютерные науки © М.Л. Цымблер 4

Эволюция языков программирования

Компьютерные науки © М.Л. Цымблер 5

Эволюция языков программирования

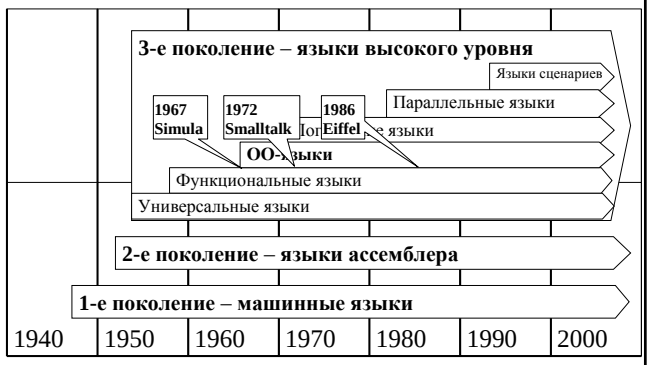
- *Язык 1-го поколения (машинный язык)* – система машинных команд конкретного компьютера.
- *Язык 2-го поколения (язык ассемблера)* – система мнемоник для обозначения машинных команд конкретного семейства компьютеров.
- *Язык 3-го поколения (язык высокого уровня)* – система обозначений для абстрактного описания вычислений.
- *Язык 4-го поколения (?)* – система визуального проектирования пользовательских приложений, выполняющая автоматическую генерацию соответствующих программ на языке программирования 3-го поколения.

Компьютерные науки © М.Л. Цымблер 6

Эволюция: универсальные языки

Эволюция: функциональные языки

Эволюция: ОО-языки



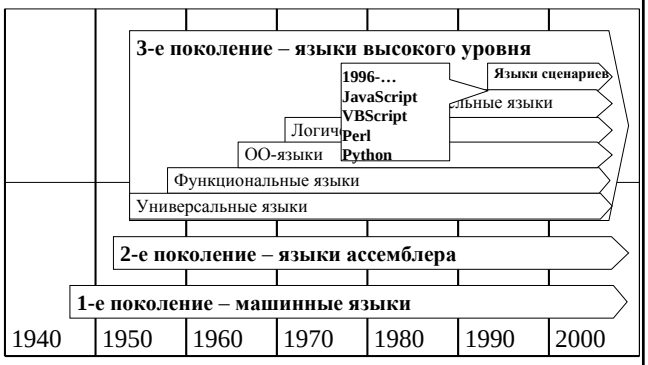
Эволюция: логические языки

Компьютерные науки © М.Л. Цымблер 10

Эволюция: параллельные языки

Компьютерные науки © М.Л. Цымблер 11

Эволюция: языки сценариев



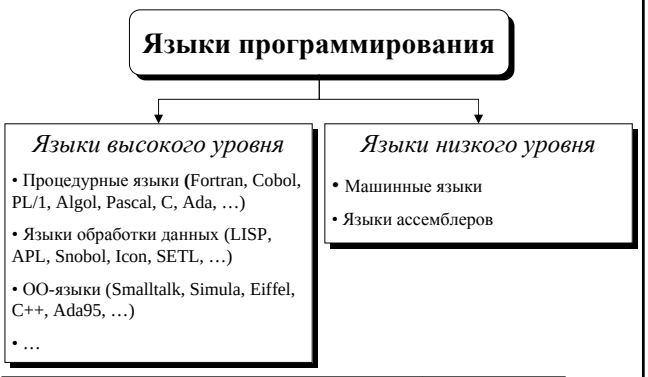
Компьютерные науки © М.Л. Цымблер 12

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
- …

Компьютерные науки | © М.Л. Цымблер | 13

Классификация языков (по уровню абстракции)



Компьютерные науки | © М.Л. Цымблер | 14

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.

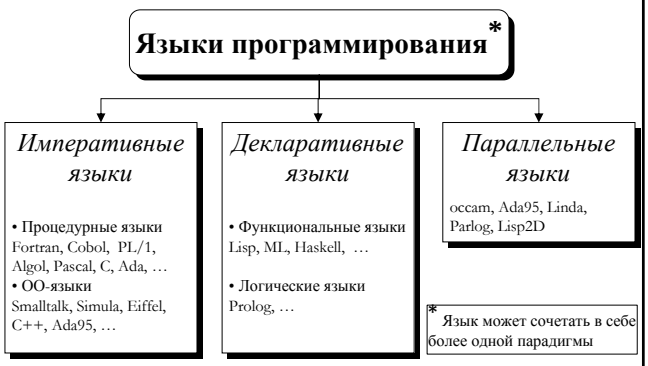
Компьютерные науки | © М.Л. Цымблер | 15

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)



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.

Компьютерные науки | © М.Л. Цымблер | 19

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**.**

Компьютерные науки | © М.Л. Цымблер | 20

Классификация языков (по парадигме)

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

Компьютерные науки | © М.Л. Цымблер | 21

Основные элементы языков программирования

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

Алфавит языка программирования

- Алфавит языка состоит из букв, цифр и лексем.
- *Лексема* – наименьшая единица языка, имеющая самостоятельный смысл.
- Основные классы лексем:
  - *спецсимволы*
    PAS: +, -, ^, >, <, <>, := и др.
    C: +, -, *, >, <, !=, = и др.
  - *ключевые слова* языка
    PAS: begin, case, const, for, repeat, while, type и др.
    C: main, switch, const, for, do, while, typedef и др.

Способы задания синтаксиса

- Расширенные формулы Бэкуса-Наура (РБНФ)
- Синтаксические диаграммы
- Перечисление всех верных конструкций (как правило, не используется)

Расширенные формулы Бэкуса-Наура

Джон Бэкус
род. 1924

Петер Наур
род. 1928

- Разработаны Дж. Бэкусом в 1960 г. и использованы П. Науром для описания синтаксиса языка программирования Алгол-60.

Построение РБНФ

- *Нетерминальные символы* – заключаются в угловые скобки **<...>** и используются для обозначения синтаксических конструкций языка.
- *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

РБНФ: примеры

- <пароль> ::= <место> <действие> <предмет> ?
  <место> ::= здесь | там
  <действие> ::= продается | ремонтируется
  <предмет> ::= [славянский] шкаф |
  [никелированная] тумбочка
- <ответ> ::= [<обычная фраза>] (<провал> | <норма>)
  <обычная фраза> ::= (Хорошая | Плохая) погода, не правда ли?
  <провал> ::= {<предмет>?} Да, он в отличном состоянии.
  <норма> ::= <предмет> уже продан.
- <сказка> ::= <белый бычок> | <белый бычок> <сказка>
  <белый бычок> ::= 
- <S> ::= a<A>
  <A> ::= b | c<A>
  ac, abc, abbc, abbbc
- <S> ::= a{b}c
  ac, abc, abbc, abbbc

Синтаксические диаграммы

- *Синтаксические диаграммы* – графическое изображение РБНФ.
- Нетерминальные символы

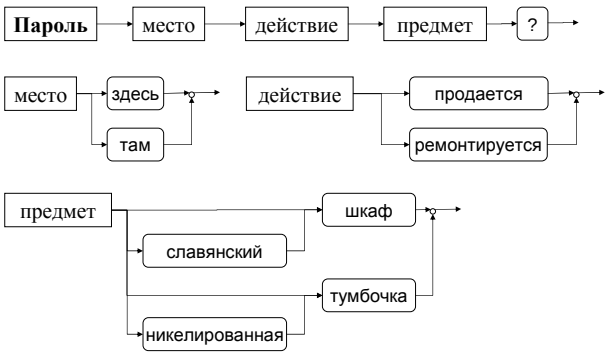
  Нетерминал
- Терминальные символы

  Терминал
- Метасимволы

  →

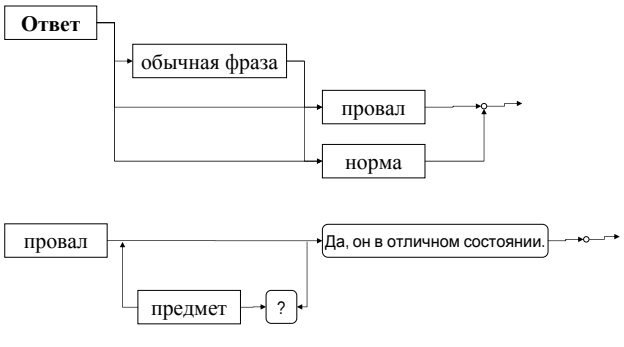
Компьютерные науки © М.Л. Цымблер 28

Синтаксические диаграммы: пример



Компьютерные науки © М.Л. Цымблер 29

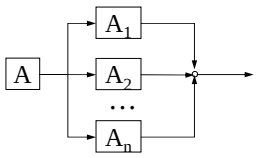
Синтаксические диаграммы: пример



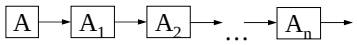
Компьютерные науки © М.Л. Цымблер 30

Эквивалентность РБНФ и диаграмм

<A>::=$A_1|A_2|\ldots|A_n$

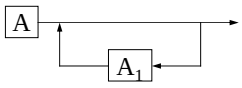


<A>::=$A_1A_2\ldots A_n$

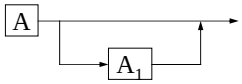


Эквивалентность РБНФ и диаграмм

<A>::={A_1}

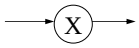


<A>::=[A_1]



<A>

X



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*
    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 состояние, 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 других (ранее определенных) типов 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 ячеек, содержащих представление значения указанного типа, которое может изменяться во время выполнения программы.
- *Константа* – имя 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

- *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;
```


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

Компьютерные науки © М.Л. Цымблер 43

Пример: программа с процедурами

```
{ myprog2.pas, 10-мар-05
  Иванов И.И.
  Пример программы с процедурами }
Program MyProg2;
procedure InpData (var A, B, C: Integer);
{ Осуществляет ввод исходных данных с клавиатуры }
begin
  ...
end;
procedure Calculation (A, B, C: Integer; var Result: Real);
{ Выполняет некоторые вычисления с исходными данными }
begin
  ...
end;
procedure OutData (Result: Real);
{ Осуществляет вывод результатов вычислений на экран }
begin
  ...
end;
var
  A, B, C: Integer; R: Real;
begin
  InpData (A,B,C);
  Calculation (A,B,C, R);
  OutData (Result);
end.
```

Компьютерные науки © М.Л. Цымблер 44

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

Компьютерные науки © М.Л. Цымблер 45

Пример: использование библиотеки

```
{ myprog2.pas, 10-мар-05
  Иванов И.И.
  Пример программы,
  использующей  библиотеку }
Program MyProg2;
uses UCalc;
procedure InpData (var A, B, C: Integer);
{ Осуществляет ввод исходных данных с
клавиатуры }
begin
  ...
end;
procedure OutData (Result: Real);
{ Осуществляет вывод результатов
вычислений на экран }
begin
  ...
end;
var
   A, B, C: Integer; R: Real;
begin
  InpData (A,B,C);
  Calculation (A,B,C, R);
  OutData (Result);
end.
```

```
{ ucalc.pas, 10-мар-05
  Петров П.П.
  Пример библиотеки. }
unit Ucalc;

interface
procedure Calculation
              (A, B, C: Integer; var Result: Real);
{ Выполняет некоторые вычисления с
исходными данными }

implementation
procedure Calculation
              (A, B, C: Integer; var Result: Real);
{ Выполняет некоторые вычисления с
исходными данными }
begin
  ...
end;

  ...
end.
```


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.