Lenguajes de Programación
Capítulo 1.
Introducción

Carlos Ureña Almagro

Curso 2011-12

Contents

1 El concepto de Lenguaje de Programación

El concepto de Lenguaje de Programación

Un lenguaje de programación es un convenio entre personas que puede definirse así:

> Conjunto de reglas o normas que permiten asociar a cada programa correcto un cálculo que será llevado a cabo por un ordenador (sin ambigüedades).

- por tanto, un lenguaje de programación es un convenio o acuerdo acerca de como se debe de interpretar el significado de los programas de dicho lenguaje
- muchas veces se confunden los lenguajes con los compiladores, interpretes o con los entornos de desarrollo de software

Estándarización de los lenguajes

El convenio suele estar reflejado en un documento (un libro) que se hace público y mediante el cual se determinan las reglas de interpretación correcta de los programas

- algunos lenguajes están definidos por un documento estandarizado en un organismo oficial como ISO (p.ej. C++ es el estándard ISO/IEC 14882, de 1998)
- en otros casos la descripción del lenguaje no está oficialmente estandarizada, el lenguaje se define por el documento de referencia que lo describe (pej. Java, descrito en este libro http://java.sun.com/docs/books/jls/)

¿ que es un programa ?

En este contexto, un programa es una:

> Secuencia finita de dígitos y caracteres, directamente legible por las personas y por el ordenador

- estas secuencias se suelen almacenar en uno o varios archivos de texto ASCII o UNICODE.
- también es posible almacenar programas en formatos alternativos de archivos, como XML u otros, aunque en stos casos los archivos no son directamente legibles por las personas, sí son legibles mediante herramientas de traducción.

¿ que es un programa correcto ?

Cada lenguaje tiene asociado un conjunto (que debe estar bien definido en el lenguaje) de programas correctos.

- Las reglas de cada lenguaje deben definir ese conjunto sin ambigüedades.

- Para esto se suelen usar métodos mas o menos formales. La validez de un programa se define a varios niveles con técnicas distintas:
  - léxico : expresiones regulares
  - sintaxis : gramáticas libres de contexto
  - semántica : descripciones o reglas no formales

¿ que es un cálculo ?

Es necesario definir adecuadamente que se entiende por calculo, podemos verlo desde dos perspectivas:

- en sentido amplio:

  Cualquier proceso automático de recogida, manipulación, y difusión de información

- en sentido formal:

  Cualquier proceso automático cuyos aspectos relevantes puedan ser modelados matemáticamente con una máquina de turing

¿ que es el significado de un programa ?

Al cálculo asociado por un lenguaje a un programa correcto se le considera el significado del programa.

- a las metodologías para especificar el significado de los programas se les llama semántica de los lenguajes de programación.
- las reglas de cada lenguaje deben permitir establecer el significado sin ambigüedad alguna, de forma clara y concisa

¿ que es el significado de un programa ?

Ejemplo de significado: consideremos este programa

```
int fact(int n)
{
   if (n>0)  return n*fact(n-1) ;
   else      return 1 ;
}
int main()
{
   int n ;
   while( cin >> n )
      cout << fact(n) ;
}
```

¿ que es el significado de un programa ?

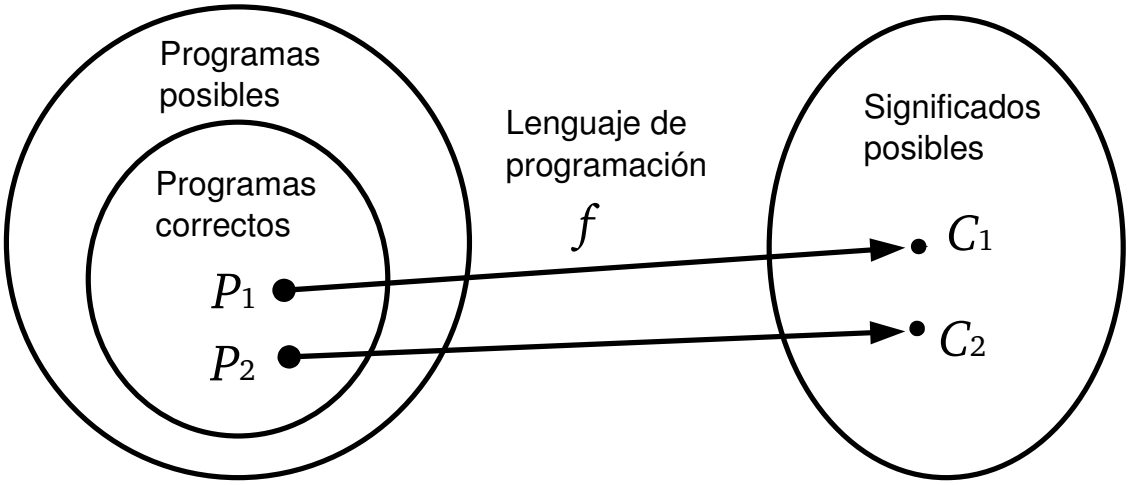
El significado del anterior programa es el cálculo necesario para evaluar una determinada función f.

- La función f es la que asocia, a una secuencia de números enteros, otra secuencia que contiene el factorial de cada uno de ellos.

$$\begin{array}{rcll} f & \in & N^* & \rightarrow & N^* \\ & & (a_1, \ldots, a_n) & \rightarrow & f(a_1, \ldots, a_n) = (a_1!, \ldots, a_n!) \end{array}$$

- Existen varias aproximaciones a la semántica de los lenguajes de programación.
- Lo que hemos visto se denomina semantica denotacional (un programa denota una función)

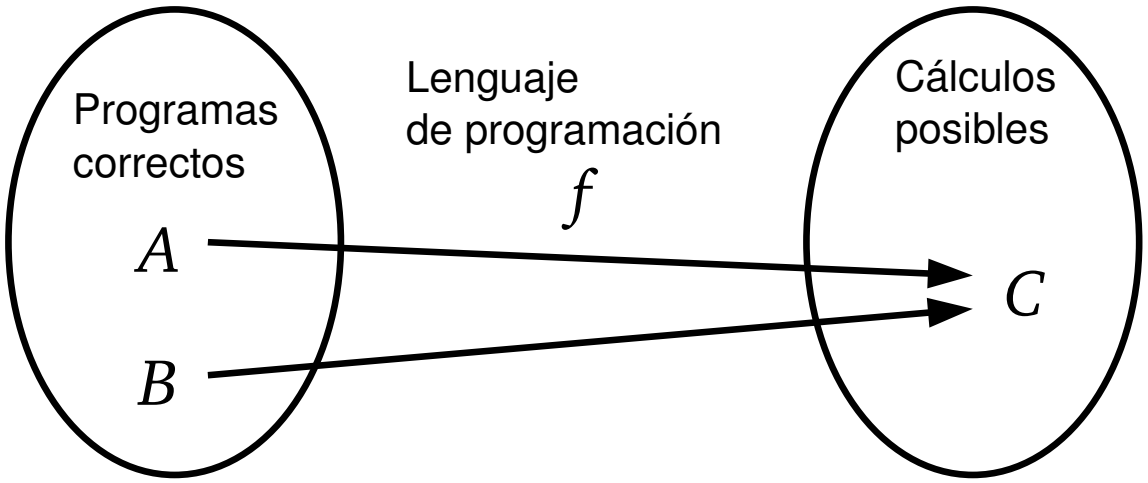
El lenguaje como una función que asigna significado



$$\begin{array}{rcl} C_1 & = & f(P_1) \\ C_2 & = & f(P_2) \end{array}$$

Programas equivalenes

Distintos programas pueden tener el mismo significado (serían equivalentes)



$$C = f(A) = f(B)$$

Ambigüedad en los programas

Los lenguajes deben ser no ambiguos:

- En cuanto a la corrección: debe ser posible calcular si un programa es correcto o no
- En cuanto al significado: el significado de un programa debe ser único.

Un ejemplo de ambiguedad es la siguiente expresion de C/C++

```
int a = 3 ;
cout << ((++a) + (a*=2)) ;  // 4+8 o 7+6
```

Según el estándard, puede denotar dos calculos distintos: se puede imprimir 12 o 13 como salida.

¿ para que sirven los lenguajes de programación ?

Esencialmente:

> para comunicar a un ordenador un procesamiento que va a llevarse a cabo en dicho dispositivo (de forma cómoda para la persona que escribe el programa).

Pero también:

> para comunicar entre personas información no ambigua sobre algoritmos matemáticos o procesos reales

¿ como deben ser los lenguajes de programación ?

Como toda actividad de diseño en ingeniería, un lenguaje de programación debe ser diseñado siguiendo unos principios básicos.

2 Criterios de diseño

Criterios de diseño de LL.PP.

A continuación se incluyen una serie de características deseables de los lenguajes de programación:

- Casi siempre, lograr una de ellas nos acerca a otras características relacionadas, pero inevitablemente también nos aleja de otras características contrapuestas a la primera.
- Cada diseño de un lenguaje supone un compromiso o equilibrio entre las características deseables.

Facilidad de lectura y escritura

Facilidad de escritura (writability)

> Facilidad para expresar un cálculo de forma clara, correcta, concisa, y rápida. (es una característica muy genérica, que se concreta en otras)

Legibilidad

> El diseño del lenguaje debe permitir que la lectura de los programas lleve fácilmente a una compresión correcta del cálculo que significan.

Generalidad y Ortogonalidad

Generalidad

> Las características o construcciones del lenguaje deben ser aplicables uniformemente y con la mínimas restricciones en todos los contextos posibles.

Ortogonalidad (independencia)

> Las diferentes características deben ser lo más independientes posible entre ellas, en el sentido de que el uso de una no debe modificar, limitar o impedir el uso simultáneo o combinado de otra.

Uniformidad y Simplicidad

Uniformidad

> Características parecidas deberían tener apariencias parecidas. Características distintas no deberían tener la misma apariencia

Simplicidad

> El numero de características o posibilidades distintas o independientes debería ser el mínimo posible. No debería haber características distintas pero muy parecidas.

Expresividad y fiabilidad

Expresividad

> El lenguaje es expresivo cuando permite expresar con facilidad procesos o estructuras complejos.

Fiabilidad

> El lenguaje debe hacer fácil la construcción de programas fiables, y debe impedir o dificultad la construcción de programas no fiables.

Definición correcta y portabilidad

Defincón precisa y clara

> Como ya se ha comentado, la correctitud y el significado de todo programa deben estar definidos sin ambigüedades, y además, estas definiciones deben fácilmente legibles y comprensibles.

Independencia de la máquina y el S.O.

> En general, el lenguaje debe permitir la construcción con la máxima facilidad posible de programas usables en arquitecturas hardware y SS.OO. distintas con un mínimo de cambios.

Eficiencia

Eficiencia en la traducción

> El diseño del lenguaje debe permitir fácilmente la construcción de traductores e intérpretes que sean eficientes en uso de tiempo y memoria

Eficiencia de ejecución

> El diseño del lenguaje no debe dar lugar a que la interpretación o ejecución de los programas traducidos conlleve un alto gasto en tiempo o memoria.

3 Paradigmas de Programación

Paradigmas de Programación

Constituyen tres categorías de los lenguajes de programación:

- Imperativos
- Funcionales (o aplicativos)
- Lógicos (o declarativos)

¿ que es un paradigma de programación ?

- Cada uno de estos paradigmas se caracteriza por un modelo formal distinto de lo que constituye un cálculo.
- El conjunto de cálculos realizables es igual en los tres casos (los que se pueden hacer con máquinas de turing)

3.1 Lenguajes Imperativos

Lenguajes Imperativos

Un cálculo es un conjunto de instrucciones que establecen explícitamente como se debe manipular la información digital presente en memoria, y/o como se debe recoger o enviar información desde/hacia los dispositivos

Ejemplos:

- Ejemplos de lenguajes imperativos: Fortran, Algol, Pascal, C, Ada, C++, Java, C#
- La mayoría de los lenguajes usados para desarrollo de software comercial son imperativos

Ejemplo en C (1/2)

```
int mcd( int x, int y )
{
  if ( x == 0 && y == 0 )
  {  printf("error!"); exit(1);
  }
  if ( x < 0 ) x = -x ;
  if ( y < 0 ) y = -y ;
  while( y != 0 )
  {  int r = x % y ;
     x = y ; y = r ;
```

```
  }
  return x ;
}
```

Ejemplo en C (2/2)

```
int main()
{
   printf("%d\n",mcd(10,102));
}
```

3.2 Lenguajes Funcionales

Lenguajes Funcionales (o aplicativos)

Un cálculo es el proceso de aplicar una función recursiva a un valor de su dominio para obtener el correspondiente valor del rango (el resultado).

El término función recursiva debe entenderse aquí según se introduce en la teoría de la computabilidad, es decir, como una función calculable con una máquina de turing (no como un subprograma que se invoca a si mismo)

Los programas en los Lenguajes Funcionales

Un programa en estos lenguajes consiste en una especificación de la función recursiva que queremos calcular, junto con los argumentos sobre los que se aplica.

Normalmente, dicha función estará especificada en términos de otras, que también se incluyen en el programa

Ejemplos:

- Ejemplos de lenguajes funcionales son: Lisp, Scheme, ML, Miranda, Haskell
- Menos difundidos que los imperativos para el desarrollo de software comercial

Ejemplo de programa en Haskell

En este lenguaje, el máximo común divisor se calcula con la función `mcd`

```
mcd 0 0 = error "mcd no está definido para 0,0"
mcd x y = mcd1 (abs x) (abs y)
where
   mcd1 x 0 = x
   mcd1 x y = mcd1 y (resto x y)
```

```
mcd 10 102;
```

3.3 Lenguajes Declarativos

Lenguajes Declarativos (logicos

Un cálculo es el proceso de encontrar que elementos de un dominio cumplen determinada relación definida sobre dicho dominio, o bien determinar si un determinado elemento cumple o no dicha relación.

- Un programa en estos lenguajes consiste en una especificación de la relación que queremos calcular
- Normalmente, dicha relación estará especificada en términos de otras, que también se incluyen en el programa

El lenguaje Prolog

El lenguaje declarativo por excelencia es Prolog

- Ejecutar un programa consiste en buscar recursivamente en una base de datos de relaciones
- Prolog está especialmente indicado para aplicaciones muy específicas como:
  - sistemas expertos
  - demostración de teoremas
  - consulta de bases de datos relaccionales,
  - procesamiento del lenguaje natural
- Para estos casos, los programas en prolog son más cortos y claros que los equivalentes en otros paradigmas

Ejemplo de programa en Prolog

En este caso, `mcd` es una relacción entre tres valores enteros (se cumple si el tercero es el m.c.d. de los dos primeros)

```
mcd1(x,y,r) :- y is 0, r is x
mcd1(x,y,r) :- nx is y, resto(x,y,ny),
               mcd1(nx,ny,r)
mcd(x,y,r)  :- abs(x,ax), abs(y,ay),
               mcd1(ax,ay,r)
?- mcd(10,102,z).
```

Los paradigmas de prog. en la asignatura.

- Esta asignatura se centra en el paradigma de programación imperativa, con algunas menciones a características de lenguajes funcionales
- No trataremos aspectos de programación declarativa

Puntos en común entre paradigmas.

- Algunos conceptos de los lenguajes de programación son comunes a la programación imperativa y la funcional
- Algunos de los conceptos surgidos en la programación imperativa se han incorporado a los lenguajes funcionales (y viceversa)

4 Antecedentes

Antecedentes

- Los primeros lenguajes de programación surgen debido a las dificultades de la programación directa en código máquina o con ensambladores básicos.
- Revisaremos tres de los primeras construcciones que pueden llamarse lenguajes de programación

4.1 Plankalkül

El lenguaje Plankalkül

Puede considerarse el primer lenguaje específicamente diseñado para la programación de ordenadores.

- Diseñado por Konrad Zuse en Alemania, entre 1943 y 1945. Inicialmente pensado para sus ordenadores Z1, Z2, que fueron de los primeros de la historia.
- No se pudo implementar un compilador o interprete para los Z por los problemas de la postguerra alemana.
- No se publicó hasta 1972, y el primer (y único) intérprete se implementó en 2000.

Conceptos que introduce

Introduce varios de los conceptos básicos de la programación de alto nivel:

- Variables: sin declaración explícita, con nombres de una letra (R,V o Z) seguida de un dígito.

- Expresiones con operadores infijos
- Sentencia de asignación de una expresión a una variable
- Ejecución condicional de sentencias (if sin else)
- Bucles indefinidos (mediante sentencias equivalentes a continue o break, no con una expresión lógica)
- Subprogramas que producen un resultado

Conceptos que introduce: tipos de datos

- Tipos de datos definidos por el usuarios (con declaración explícita de la representación como secuencias de bits), también llamados estructuras
- Tipos de datos (estructuras) predefinidos: enteros, enteros en BCD, reales, números complejos, números racionales (con y sin signo)
- Pares, arrays y listas (con elementos de tipo arbitrario), implementados usando árboles binarios.

4.2 Shortcode

El lenguaje Shortcode

Es el lenguaje usado por el primer programa intérprete conocido..

- Desarrollado en EE.UU por John Mauchly en 1949, fue usado en el UNIVAC I.
- El ordenador UNIVC I tenía una longitud de palabra de 72 bits. Las instrucciones de asignación con expresiones complejas se codifican en dichas palabras de 72 bits.
- Se buscaba la facilidad de programación a cambio de perder eficiencia en tiempo (ya que se interpretaba)

4.3 A-0

El lenguaje A-0

Diseñado e implementado por Grace Hopper entre 1951 y 1953 para la compañía UNIVAC, se diseño para el primer compilador (que generaba código máquina a partir de especificaciones de más alto nivel en A-0)

- La entrada del compilador es un programa con llamadas a subrutinas:
  - Las subrutinas se identifican por nombres
  - Podían incluir lecturas o escrituras sobre parámetros formales.

- La salida es código máquina con las subrutinas expandidas en el lugar de las llamadas, y con las referencias a los parámetros particularizadas a los actuales
- Esto se hacía antes manualmente, con muy alta probabilidad de errores.

5 Los primeros lenguajes de programación

Lenguajes pioneros

- Existen multitud de lenguajes, y la mayoría se han diseñado incorporando nuevos conceptos
- En los primeros lenguajes casi todos los conceptos eran nuevos
- Seleccionamos tres por la relevancia a largo plazo de los conceptos que introdujeron:
  - **Fortran**: introduce la **programación de alto nivel**
  - **Algol**: Introduce la **programación estructurada**
  - **Simula**: Introduce la **programación orientada a objetos**

5.1 Fortran

El lenguaje Fortran

Es el primer lenguaje de alto nivel con amplia difusión y amplia disponibilidad de compiladores. La motivación de su diseño fue sustituir la programación en ensamblador.

- Diseñado e implementado por primera vez entre 1955 y 1957 en IBM, para el IBM 704.
- Ha tenido una amplísima difusión y uso
- Durante muchos años se ha considerado el lenguaje por excelencia para aplicaciones científicas, ha tenido muchas ampliaciones.

Conceptos introducidos en Fortran

- Variables cuyo nombre seria de hasta 6 caracteres (2 inicialmente). El nombre determina implícitamente el tipo, que solo podía ser entero o de coma flotante.
- Sentencia de asignación con expresiones infijas a la derecha (incluyendo sub-expresiones entre paréntesis.
- Vectores o arrays uni o bi-dimensionales.
- Sentencias IF-GOTO con expresiones aritméticas, incluyendo operadores aritméticos relacionales

- Sentencia DO (bucles definidos).
- Subrutinas definidas por el usuario (sin compilación separada)
- Entrada y de salida con formato.

Ejemplo de programa en Fortran

Calcula el producto y la suma de los elem. de un array, e imprime 0 (producto $\leq$ suma), o 1 (producto $>$ suma) :

```
    DIMENSION A(10)
    ISUM = 0
    IPROD = 1
    DO 100 I=1,10
    ISUM = ISUM + A(I)
100 IPROD = IPROD * A(I)
    IF (IPROD-ISUM) 200,200,300
200 PRINT 0
250 GOTO 500
300 PRINT 1
500
```

5.2 Algol

El lenguaje Algol

Es el lenguaje que introduce lo que hoy conocemos como programación estructurada. La motivación inicial era mejorar algunas limitaciones de Fortran.

- Definido por un comité internacional en varias reuniones entre 1958 y 1962.
- Es el primer lenguajes descrito formalmente por BNFs
- No fue apoyado por IBM, y se difundió poco, aunque se uso mucho para comunicar algoritmos
- Tuvo un impacto enorme en posteriores lenguajes: Pascal, Modula, Ada, C, Simula

Conceptos introducidos en Algol

- Declaración explícita de variables y parámetros, incluyendo su tipo.
- Arrays multidimensionales, con posibilidad de fijar el rango de índices.
- Agrupación de sentencias en sentencias compuestas delimitadas por las palabras clave begin-end (estructura de bloques).

- Sentencia if-else, bucles definidos e indefinidos con posibilidad de anidamiento arbitrario.
- Variables de ámbito restringido a los bloques.
- Paso de parámetros por valor y por nombre
- Procedimientos recursivos.

Ejemplo de (sub)programa en Algol (1/2)

Subprograma de nombre `ejemplo` que acepta dos parámetros enteros:

```
procedure ejemplo (a, b);

   value   a, b ;
   integer a, b ;

   begin
     integer k ;
     real    e ;
     ....
```

Ejemplo de (sub)programa en Algol (2/2)

```
    ....
    for k := a/2 step 2 until b do
    begin
      e := if es_primo(k) then 0 else 1 ;
      if e == 0 then
         putlist(k,' es primo')
      else
         putlist(k,' no es primo')
    end
end
```

5.3 Simula

El lenguaje Simula

Es el lenguaje que introduce los conceptos esenciales de la programación orientada a objetos.

- Basado en Algol para todos los aspectos de la programación estructurada.
- Su diseño fue motivado por la necesidad de crear software de simulación.

- Fue diseñado e implementado en el Centro de Computación Noruego de Oslo entre 1962 y 64, por Kristen Nygaard y Ole-Johan Dahl.
- No se ha usado mucho para software comercial, sí para comunicar algoritmos.
- Influencia decisiva en lenguajes OO: Smalltalk, Eiffel, C++, Java.

Conceptos introducidos en Simula

- Definición de clases como variables de instancia más metodos para acceder a ellas
- Los objetos como variables instancias de una clase
- Herencia o extensión de clases.
- Métodos virtuales (polimorfismo de herencia).
- Corutinas (es un concepto de la programación concurrente)

Ejemplo de programa en Simula (1)

Clase de nombre rectángulo con dos atributos reales (`ancho` y `alto`) y un método que devuelve un valor lógico (`escuadrado`).

```
begin
   figura class Rectangulo (ancho, alto );

       real    ancho, alto;

       boolean procedure esCuadrado ;
          esCuadrado:= ancho=alto ;

   end of Rectangulo ;
   ....
```

Ejemplo de programa en Simula (2)

```
   ....

   ref(rectangulo) r ;
   r:- new rectangulo(2.0,2.0) ;

   if ( r.esCuadrado ) then
   begin
     OutText("es un cuadrado") ;
     OutImage ;
```

```
   end

end of program
```

fin del capítulo.