TEMA 25

PROGRAMACIÓN ESTRUCTURADA. ESTRUCTURAS BÁSICAS. FUNCIONES Y PROCEDIMIENTOS

ÍNDICE

1. ELEMENTOS BÁSICOS DE LA NOTACIÓN ALGORÍTMICA.

Como resultado de su trabajo, el programador proporciona un texto que describe por anticipado el conjunto de las operaciones que un ordenador debe ejecutar para resolver el problema del que tiene encomendado hallar una solución informática.
Buena parte de la dificultad del trabajo reside en que esta descripción debe tener en cuenta todas las posibilidades dque se puedan producir, con el fin de evitar que se produzcan fallos en el conjunto del tramtamineto por no haber previsto algún caso particular.

En este apartado se describen algunos términos usuales en el contexto informático.

1.1. Acción, Información, Estado.

Una acción es un acontecimient que se produce durante una ejecución de un programa. Tiene lugar durante un período de tiempo finito y produce un resultado bien definido y previsto.

Para que sea posible reconocer el resultado de la acción es preciso que el sistema observado durante el lapso de tiempo esté provisto de indicadores que tomen valores diferentes.

El hecho de que el resultado de una acción sea previsible significa que si se conoce el estdo del sistema en t0 entonces se puede decir conprecisión cuál será el estado del sitema en t1 antes includo de que la acción en cuestión se produzca.

1.2. Proceso.

Describir un proceso es interpretar un acontecimiento como un conjunto de acciones elementales cuyo efecto acumulativo es el mismo que el producido por el acontecimiento completo.
Si una acción de este proceso no puede empezar antes de que la acción en curso esté completamente terminada, entonces el proceso se denomina **secuencial**.En el caso contrario, se denomina en **paralelo.**

En lo sucesivo se tratará exclusivamente de procesos secuenciales. de modo que en un instante dado tan sólo podrá ejecutarse una sóla acción.

1.3. Esquema.

Es una descripción o una representación mental reducida a los rangos esenciales (de un objeto, de un proceso).

1.4. Algoritmo.

Definimos un algoritmo como la descripción de un esquema de comportameinto expresado mediante un repertorio de acciones y de informaciones elementales identificadas, bien comprendidas y realizadas a priori. Este repertorio se denomina el **léxico del algoritmo.**

Un **algoritmo** a nivel informático es un procedimeitno efectivo que para todos los valores de datos dados obtiene la solución, si existe, o determina que dicha solución no existe. Esto quiere decir que el resultado de aplicar el algoritmo está definido en todos los casos

La existencia de un algoritmo para resolver un problema implica que dicho problema ha de ser decidible. Si un problema no es decidible no podrá haber ningún algoritmo que lo resuelva.

Un algoritmo debe ser preciso, indicando el orden de realización de cada paso, bien definido, si se sigue el algoritmo dos veces consecutivas con la misma entrada se debe obterner el mismo resultado y finito, es decir , debe terminar en algún momento.

1.5. Programa.

Definimos programa como un algortimo destinado a gobernar una máquina real. El objetivo del programador es la escritura de programas.

La realización de un programa pasa por la escritura de uno o más algoritmos que después se traducen en un lenguaje de alto nivel donde la máquina lo entenderá perfectamente.

2. DEFINICIÓN DE PROGRAMACIÓN ESTRUCTURADA

Se denomina estructura a la manera en que diferentes partes se combinan para construir un todo.
Un programa tendrá la estructura correspondiente a la forma en que las distintas acciones individuales o cálculos parciales se combinan para construir el programa en su conjunto.

Reduciendo el problema de la estructuración de un programa a su expresión más sencilla, el planteamiento al que se llega es como combinar unas pocas operaciones de la manera más clara posible para construir otra más compleja.

3. ESTRUCTURAS BÁSICAS.

3.1. Especificación formal.

Para estudiar los fundamentos de la programación estructurada vamos a utilizar principalmente el pseudocódigo. Esta es una técnica para expresar en lenguaje natural la lógica de un programa, es decir, su flujo de control. Los pseudocódigos utilizan palabras comohacer, ***si-entonces-sino, repeti-hasta, mientras….***
Unida a esta técnica s ha formalizadoun conjunto de reglas que permiten comprobar la validez de un algoritmo. La ejecución de una instrucción supone habitualmente unos cambios en su entorno. Para especificar un proceso (ejecución de programa o algoritmo), describiremos la relaciónentre los estado inicial y final del mismo. Esta relación, denominada especificación, se formaliza con la ayuda de predicados, que siempre tiene un valor de TRUE o FALSE.

En general,la especificación de un algoritmo constará de los siguientes elementos:

- declaración de variables.
- precondición
- nombre del algoritmo
- postcondición

donde la precondición y postcondición son predicados sobre las variables del proceso. Estos predicados reflejan , repectivamente la situación inicial y final pretendida de las variables de nuestro algoritmo.

3.2. Estructuras elementales

3.2.1 Asignación

La asignación consiste en asociar un valor a una variable. Su sintaxis es la siguiente: x:=E, que se lee como “x toma el valor de lo que valga E”. E debe ser una expresión válida del mismo tipo que x. Hay que tener en cuenta que primero se evalua la expresión y luego se copia en la variable el valor obtenido.

3.2.2 Composición secuencial

Consiste en la ejecución incondicional de una secuencia de sentencias.
Una operación compleja puede consistir en ejecutar una tras otra una secuencia de operaciones más sencillas.Así para indicar que éstas deben ejecutarse sucesivamente se dará la lista de dichas acciones en el orden en que deben ser ejecutadas, estas acciones se separaran por ; que juega dos papeles :

- indicar el fin de una acción
- indicar que el proceso no ha terminado.

3.2.3 Composición iterativa

Un operación compleja puede consistir en repetir una o más operaciones un número de veces, o hasta o mientras se cumpla una determinada condición. Para describir esta instrucción adoptamos en nuestra notación algorítmica a estructura:
Mientras B hacer S fminetras

Donde B es una expresión lógica y S una instrucción. Al principio de la ejecución, la condición B es comprobada y en caso de ser cierta, se procesa la instrucción S, conocida como cuerpo del bucle; el proceso se repite hasta que la condición B no se cumpla.

Toda iteración plantea dos tipos de problemas:

a) Determinar el efecto de la instrucción iterativa: para ello hemos de buscar una relación que se mantenga inalterada antes y después de cada iteración y que describa todos los estados por los que atraviesa el cómputo realizado por el bucle, observados justo antes de evaluar la condición B de terminación. A esta relación le llamaremos invariante.
b) Probar que la instrucción iterativa termina. Para ello es útil asociar a cada iteración una función que dependa de las variables y que actúe de limitadora o cota superior del número de iteraciones pendientes haciendo que el aserto B deje de cumplirse al dejar de ser estrictamente positiva.

EXISEN TAMBIEN OTRAS INSTRUCCIONES DE ITERCIÓN COMO SON : repetir S hasta B Y LA INSTRUCCIÓN para, QUE ES UN CASO DEL mientras.

3.3. Teorema de estructura.

Los fundamentos de la denominada programación estructurada se establecieron a principios de los años sesenta y se consolidaron con los trabajos de Dijkstra, Bohm ..El resumen de sus postulados es el siguiente: se dice que un programa es estructurado si se expresa únicamente mediante combinaciones de las estructuras básicas:

Secuencial
Alternativa
Iterativa

El teorema de estructura afirma que dado un programa P no estructurado con una única entrada y una única salida y sin interrupciones en el flujo de control, es posible construir un programa P' estructurado que obtnega los mismos resultados.

3.4. Estructuras que permiten reutilizar el código

Uno de los componentes que los lenguajes estructurados es que incorporan un tipo de secuencias algorítmicas individualizadas que pueden recibir o no valores de entrada y que también pueden devolver o no valores de salida.
Se trataría de sustituir todo un conjunto de instrucciones que puede incluir cualquier combinación de las denominadas estructuras básicas (secuencial, condicional, iterativa) por un identificador que puede incorporar la declaración de valores. Posteriormente en cualquier lugar del programa se podrá invocar al conjunto de instrucciones utilizando el identificador.

Tras la ejecución y según el tipo de estructura utilizada se pueden obtener resutados devueltos.
La diferencia entre procedimientos y funciones tiene su origen en las especificaciones de algunos lenguajes estructurados como PASCAL, aunque esta es muy poca.

3.4.1 Procedimientos

Un procedimiento sirve para definir partes de un programa mediante la asociación de un identificador. Posteriormente dichas partes se pueden activar utilizando sentencias de llamada.

Un procedimiento es pues un algoritmo diseñado de tal modo que es susceptible de ser llamado por otros algoritmos que a su vez pueden ser procedimientos. Al algoritmo que llama se le denomina algoritmo principal o en su caso procedimiento principal.

3.4.2 Funciones

Matemáticamente, una función es una operación que toma uno o más valores llamados **argumentos** y produce un valor denominado **resultado** o **valor** de la función para los argumentos dados. Todos los lenguajes de programación tienen funciones incorporadas o intrínsecas y funciones definidas por el usuario. Así, por ejemplo, la siguiente es una función: $f(x) = 1 - x$, donde *f* es el nombre de la función y *x* es el argumento. Ningún valor específico se asocia con *x*; es un *parámetro formal* utilizado en la definición de la función. Para evaluar *f* debemos darle un valor real o actual a *x*, con este valor se puede calcular el resultado. Una función puede tener uno o varios argumentos. Sin embargo, un único valor se asocia con la función para cualquier conjunto dado de argumentos.

Cada lenguaje de programación tiene sus propias funciones incorporadas, que se utilizan escribiendo sus nombres con los argumentos adecuados. Las funciones se evocan utilizando su nombre en una expresión, con los argumentos actuales o reales encerrados entre paréntesis.

Las funciones incorporadas en el sistema se denominan *funciones internas* o intrínsecas y las funciones definidas por el usuario *funciones externas*. Cuando las funciones estándares o internas no permiten realizar el tipo de cálculo deseado es necesario recurrir a las funciones externas que pueden ser definidas por el usuario mediante una *declaración de función*.

Declaración de funciones

La declaración de una función requiere una serie de pasos que la definen. Una función tiene una constitución similar a un algoritmo; por consiguiente, constará de una *cabecera* con la definición de la función seguida por el *cuerpo* de la función, que serán una serie de acciones o instrucciones cuya ejecución hará que se asigne un valor al nombre de la función. Esto designa el valor particular del resultado que ha de devolverse al programa llamador.

Función nombre_función (par1, par2, par3, ...)
<acciones>

par1, par2, par3, ...	son los parámetros formales o argumentos
nombre_función	nombre asociado con la función (será un nombre de identificador válido)
<acciones>	instrucciones que constituyen la definición de la función; deben contener una sola acción de designación que asigne un valor al nombre de la función.

Ejemplo:

La función f(x) = 1 – x

Se definirá por:

```
función F(x)
Inicio
        F    1 – x
Fin
```

Para que las acciones descritas en un subprograma función sean ejecutadas, se necesita que éste sea invocado desde un programa principal o desde otros subprogramas a fin de proporcionarle los argumentos de entrada necesarios para realizar esas acciones.

Los argumentos de la declaración de la función se denominan *parámetros formales*, ficticios o mudos; son nombres de variable, de otras funciones o procedimientos, y que sólo se utilizan dentro del cuerpo de la función. Los argumentos utilizados en la llamada a la función se denominan *parámetros actuales*, que, a su vez, pueden ser constantes, variables, expresiones, valores de funciones o nombres de funciones o procedimientos.

En la declaración de una función debe aparecer el **tipo** de dicha función. Este tipo podrá ser cualquiera de los soportados por el lenguaje de programación utilizado. Si el lenguaje lo permite, el tipo podrá ser definido por el usuario.

Definir el tipo de una función es definir el tipo del valor que retorna la misma.

Invocación de funciones

Una función se llama *mediante referencia*, de la forma siguiente:

Nombre_función (lista de parámetros actuales)

Nombre_función	es la función que se llama
Lista de parámetros actuales	son constantes, variables, expresiones, valores de funciones, nombres de funciones o procedimientos

Cada vez que se llama a una función desde el algoritmo principal se establece automáticamente una correspondencia entre los parámetros formales y los parámetros actuales. Debe haber exactamente el mismo número de parámetros actuales que de parámetros formales en la declaración de la función, y se presupone una correspondencia uno a uno de izquierda a derecha entre los parámetros formales y los actuales.

Una llamada a la función implica los siguientes pasos:

1. A cada parámetro formal se le asigna el valor real de su correspondiente parámetro actual (esta correspondencia se denomina llamada por valor).
2. Se ejecuta el cuerpo de acciones de la función.
3. Se devuelve el valor de la función y se retorna al punto de llamada.

La forma de devolver el valor de la función va a depender del lenguaje utilizado. Existen dos formas que engloban a la mayoría de los lenguajes de programación:

- Utilizando el nombre de la función como si fuese una variable. Por ejemplo:

```
Function Área (R:real):real;
Const Pi=3.141519;
    Begin
    Área:=Pi*R*R;
    End
```

(sintaxis correspondiente a Pascal)

- Utilizando la instrucción *return valor*, donde valor será una constante o una variable del tipo de la función. Por ejemplo:

```
float Área (float R)
{
const float Pi=3.141519;
return Pi*R*R;
}
```

(sintaxis correspondiente a C)

4. METODOLOGIA DE DESARROLLO DE PROGRAMAS.

Las recomendaciones de la programación estructurada se traducen en una metodología a seguir para desarrollar programas. Dicha metodología recibe el nombre general de **refinamiento progresivo.**

La técnica de refinamiento es del tipo descendiente (topdown).
Consiste en describir inicialmente el programa como una acción global y luego irlo decomponiendolo en partes de manera que en cada paso de descomposición se apliquen sólo las estructuras básicas

recomendadas. La descomposición o refinamiento debe ir aplicándose a los datos al mismo tiempo que a las operaciones.

En programas voluminosos el proceso de refinamiento debe ir acompañado de una descomposición en módulos.

5. BIBLIOGRAFIA

Pérez Lobato, J.M.

Metodología de la programación

Alhambra-Longman, 1994

Joyanes Aguilar, L.

Fundamentos de programación

Mc Graw-Hill, 1992

Alfonso Ureña López

Fundamentos de Informática

Ra-ma, 1997