Resumen Pseudocódigo

Antonio Vélez Estévez

19 de octubre de 2016

Versión 1.0

Resumen

Este documento es un pequeño manual-resumen del pseudocódigo usado en la asignatura de Introducción a la Programación en la Universidad de Cádiz.

Índice general

1 Vista general

La estructura general de un algoritmo en pseudocódigo es la siguiente:

```
Algoritmo nombreAlgoritmo
  const
    //sección de definición de constantes.
  tipos
    //sección de definición de tipos.
  var
    //sección de definición de variables globales.


    //definición de funciones y procedimientos.
principal
  var
    // sección de definición de variables locales a Principal.
  inicio
    // inicialización de variables...
    // instrucciones del algoritmo principal...
fin_principal

fin_algoritmo
```

2 Tipos de datos

En pseudocódigo disponemos de los tipos de datos simples y estructurados para diseñar nuestros programas.

2.1 Simples

2.1.1 Numéricos

Enteros es un subconjunto finito de los enteros ($\mathbb{Z}$), el rango disponible es:

$$[-32768, 32768]$$

Para declarar una variable de tipo entero en pseudocódigo usaremos la palabra reservada `entero`.

Reales es un subconjunto finito de los reales ($\mathbb{R}$), el rango disponible es:

$$\left[1,17549 \times 10^{-38}, 3,40282 \times 10^{38}\right]$$

Para declarar una variable de tipo real en pseudocódigo usaremos la palabra reservada `real`.

2.1.2 Lógicos

Este tipo de dato solo puede tener dos valores **verdadero** o **falso**, sobre él se pueden aplicar los operadores del algebra booleana [1].

Para declarar una variable de este tipo en pseudocódigo usaremos la palabra reservada `logico`.

[1] Conjunción, disyunción, negación y disyunción excluyente (habitualmente conocida como **XOR** de "eXclusive OR").

2.1.3 Carácter

Puede representar los carácteres alfanuméricos especificados en el código ASCII[2].

Para declarar una variable de tipo carácter en pseudocódigo usaremos la palabra reservada `caracter`.

2.2 Estructurados

Los tipos de datos estructurados se definirán en la sección de definición de tipos, que está encabezada por la palabra reservada `tipos`.

2.2.1 Vectores

Un vector es un conjunto finito de elementos *del mismo tipo*, que están almacenados consecutivamente y que pueden ser identificados de forma independiente.

Para definir un vector en la sección de tipos usaremos la siguiente sintaxis:

```
vector [tamaño] de <tipo_dato> : <identificador_tipo_vector>
```

Luego usaremos ese identificador para declarar el tipo de una variable.

Nota[3]: los vectores en pseudocódigo empiezan en la posición 1.

2.2.2 Matrices

Una matriz multidimensional es una estructura homogénea, en la cual para hacer referencia a un elemento necesitamos dos o más índices dependiendo de su dimensión.

Para definir una matriz en la sección de tipos usaremos la siguiente sintaxis:

```
matriz [tam1, ..., tamN] de <tipo_dato> : <identificador_tipo_matriz>
```

Para declarar una variable de dicho tipo usaremos el identificador que establezcamos.

2.2.3 Cadenas de caracteres

Una cadena de caracteres es una secuencia de caracteres consecutivos. Para declarar una variable como cadena debemos usar la palabra clave `cadena`.

- Se puede asignar a una variable de tipo `cadena` una constante de cadena.
- Se pueden usar las operaciones de lectura y escritura con cadenas de caracteres.
- Podemos calcular la longitud de una cadena con `longitud`(#1).
- Se pueden comparar cadenas con los operadores relacionales conocidos[4].
- Podemos concatenar dos cadenas con `concatena`(#1, #2)

2.2.4 Registros

Es una estructura de datos formada por un conjunto de elementos que contienen información relativa a algo. Los elementos que constituyen un `registro` se denominan *campos* y cada campo puede ser de un tipo diferente.

[2] American Standart Code for Information Interchange. El código ASCII utiliza 7 bits para representar los carácteres.

[3] En Vary debido a temas de diseño los vectores empiezan en la posición 0 al igual que en el lenguaje C.

[4] menor-que($<$), mayor-que($>$), igual($=$), menor o igual ($\leq$), mayor o igual ($\geq$) y distinto ($\neq$).

Para definir un **registro** en la sección de tipos usaremos la siguiente sintaxis:

```
registro : nombreRegistro
  tipo1 : idCampo1
  tipo2 : idCampo2
     .
  tipoN : idCampoN
fin_registro
```

Consideraciones:

- Se puede realizar la asignación entre dos registros completos siempre que sean del mismo tipo.

2.2.5 Ficheros

Para definir un tipo de **fichero** en la sección de tipos usaremos la siguiente sintaxis:

```
archivo de tipo_dato : tipoFichero
```

Para abrir un fichero usaremos la funcion **abrir**() que asocia una variable de tipo fichero con un archivo. El formato de dicha función es el siguiente:

```
abrir(varFichero, modo, nombreFichero)
```

Donde:

- **varFichero** es la variable, declarada previamente a la que se le asociará el fichero físico.
- **modo** indica el modo de acceso al fichero(escritura o lectura).
- **nombreFichero** es el nombre del fichero que se encuentra almacenado en memoria masiva.

Para el manejo de ficheros disponemos de cuatro operaciones:

- `escribir(varFichero, elemento).`
- `leer(varFichero, elemento).`
- `feof(varFichero).`
- `cerrar(varFichero).`

2.2.6 Enumerados

El uso de tipos enumerados permite mejorar la legibilidad de los algoritmos y reducir posibles errores. Para definir un tipo enumerado usaremos la siguiente sintaxis en la s ección de tipos:

```
identificador_tipo_enumerado = {elemento1, ..., elementoN}
```

Luego para declarar una variable como este tipo enumerado usaremos dicho identificador.

2.2.7 Subrango

Contiene un rango de valores de otro tipo existente, sea predefinido o definido por el usuario. Para definir un subrango usaremos la siguiente sintaxis en la sección de tipos:

```
identificador_tipo_subrango = limInf .. limSup
```

Para declarar una variable como este tipo subrango usaremos dicho identificador.

3 Variables, constantes y expresiones

Una **variable** es un objeto que contiene un valor que puede variar durante la ejecución del programa, mientras que una **constante** también es un objeto que contiene un valor, pero que, a diferencia de una variable, su valor no cambia.

Para usar una variable primero tenemos que declararla, esto es, asociar un tipo con la misma. En pseudocódigo esto se hace en la sección `var` de la siguiente forma:

```
identificador_del_tipo : identificador_de_la_variable
```

Para usar una constante primero tenemos que definirla, esto es darle un valor. En pseudocódigo esto se hace en la sección `const` de la siguiente forma:

```
identificador_de_la_constante = valor_de_la_constante
```

3.1 Asignación

Se utiliza para darle valor a una variable y se representa por ←, el formato es:

```
identificador_de_variable ← expresión
```

Esta operación es destructiva, el valor que tenía antes la variable desaparece.

3.2 Expresiones aritméticas

Las expresiones aritméticas que se pueden realizar vienen resumidas en la siguiente tabla:

+	Suma	|	OR a nivel de bits
-	Resta	&	AND a nivel de bits
*	Producto	xor	XOR a nivel de bits
/	División entera		
div	División real		

3.3 Expresiones lógicas

Las expresiones lógicas que se pueden realizar vienen resumidas den la siguiente tabla:

no ó !	Negación lógica
y ó &&	Conjunción
o ó ||	Disyunción
=	Igualdad
!= ó <>	Desigualdad
≥ ó >=	Mayor o igual
≤ ó <=	Menor o igual
<	Menor que
>	Mayor que

4 Estructuras de control

4.1 Selectivas

Permiten decidir entre varios bloques de código en base a una condición.

4.1.1 Simple

La estructura general de esta estructura selectiva es:

```
si (condición) entonces
  // Bloque que se ejecuta únicamente si la condición
  // es verdadera.
fin_si
```

4.1.2 Doble

La estructura general de esta estructura selectiva es:

```
si (condición) entonces
  // Bloque que se ejecuta únicamente si la condición
  // es verdadera.
si_no
  // Bloque que se ejecuta únicamente si la condición
  // es falsa.
fin_si
```

4.1.3 Múltiple

La estructura general de esta estructura selectiva es:

```
segun_sea (expresión) hacer
  1:
    // Bloque de código si la expresión resultante es 1.
  2:
    // Bloque de código si la expresión resultante es 2.
  .
  .
  .
  n:
    // Bloque de código si la expresión resultante es n.
  en_otro_caso:
    // Bloque de código en un caso contrario a los anteriores.
fin_segun
```

4.2 Repetitivas

Permiten repetir una o varias instrucciones varias veces en función de la evaluación de una determinada condición. A las estructuras repetitivas se les denomina *bucles*, y se llama *iteración* a cada repetición de la ejecución de la secuencia de instrucciones que forman el llamado *cuerpo del bucle*.

4.2.1 Mientras

La plantilla de esta estructura repetitiva es:

```
mientras (condición) hacer
  // cuerpo del bucle
fin_mientras
```

4.2.2 Repetir

La plantilla de esta estructura repetitiva es:

```
repetir
  // cuerpo del bucle
hasta_que (condición)
```

Esta estructura tiene una peculiaridad, **se ejecuta mientras la condición se evalúa como falsa**, cuando es verdadera termina su ejecución.

4.2.3 Desde

La plantilla de esta estructura repetitiva es:

```
desde i ← Vi hasta Vf hacer
  // cuerpo del bucle
fin_desde
```

Por defecto el bucle `desde` aumenta la variable de control en una unidad en cada iteración, podemos cambiar esto poniendo:

```
desde i ← Vi hasta Vf hacer i ←i +n
  // cuerpo del bucle
fin_desde
```

Siendo `n` el valor en el que se quiere incrementar o decrementar `i` en cada iteración.

5 Subalgoritmos

5.1 Funciones

Para declarar funciones en pseudocódigo usaremos la siguiente plantilla:

```
<especificación_de_la_función>
<tipo_del_resultado> funcion <nombre_función> (<lista_de_parámetros_formales>)
  var
  // Sección de definición de variables locales a la función,
  // esta sección es opcional.
  inicio
  // cuerpo de la función
  devolver (<expresión_resultado>)
fin_funcion
```

Donde:

- `lista_de_parámetros_formales` es una lista de la siguiente forma:

  `({E|S|E/S} tipo1: param1,..., {E|S|E/S} tipoN: paramN)`

  - El significado de `E`, `S` y `E/S` se detallará en 5.3Paso por valor y por referencia.

5.2 Procedimientos

Para declarar procedimientos en pseudocódigo usaremos la siguiente plantilla:

```
<especificación_del_procedimiento>
procedimiento <nombre_procedimiento> (<lista_de_parámetros_formales>)
  var
  // Sección de definición de variables locales al procedimiento,
  // esta sección es opcional.
  inicio
  // cuerpo del procedimiento.
fin_procedimiento
```

La lista de parámetros formales sigue la misma forma que en la sección anterior. Se puede observar como no hay `devolver`, ya que los procedimientos no devuelven nada.

5.3 Paso por valor y por referencia

En el paso por valor se produce una copia del valor de los parámetros actuales en los parámetros formales. Este tipo de parámetros siempre será de entrada y llevarán en su declaración una `E`.

En el paso por referencia tenemos dos casos, en ambos el parámetro formal recibe una referencia del parámetro actual:

- **Salida** en este tipo solo podemos usar el parámetro formal en un valor izquierdo, es decir, en la parte izquierda de las asignaciones. Se declara poniendo `S`.
- **Entrada/Salida** en este tipo podemos usar el parámetro formal tanto en un valor izquierdo como en uno derecho, es decir podemos usar su valor para calcular cualquier cosa y también podemos guardar valores en el mismo. Se declara poniendo `E/S`.

En el paso por valor los parámetros actuales ***no se modifican*** mientras que en el paso por referencia ***pueden modificarse***.

6 Ámbito de las variables

En pseudocódigo tenemos dos tipos de ámbito:

- **Global**. Las variables son accesibles y visibles en el resto del código.
- **Local**. Las variables locales son accesibles y visibles solamente en el bloque en el que se definen.

Para declarar una variable **global** en pseudocódigo debemos escribirla en la sección `var` del `Algoritmo`.

Para declarar una variable **local** a un bloque en pseudocódigo debemos escribirla en la sección `var` de dicho bloque.

7 Funciones y procedimientos como parámetros

Para pasar funciones o procedimientos como parámetros primero debemos declarar un tipo que sea dicha función o procedimiento siguiendo la siguiente sintaxis:

Para funciones:

```
[tipo_resultado] funcion (<lista_parametros_formales>) : id
```

Para procedimientos:

```
procedimiento (<lista_parametros_formales>) : id
```

Luego se usa dicho tipo como un tipo normal y corriente. Lo único especial que tienen las variables declaradas de este tipo es que pueden ser llamadas como una función.

A Palabras reservadas

Algoritmo	fin_algoritmo	inicio	var	tipo
principal	fin_principal	mientras	hacer	fin_mientras
repetir	hasta_que	desde	hasta	fin_desde
si	si_no	fin_si	segun_sea	en_otro_caso
entonces	vector	de	matriz	entero
real	logico	caracter	cadena	funcion
fin_funcion	devolver	procedimiento	fin_procedimiento	registro
fin_registro	escribir	leer	archivo	concatena
abrir	cerrar	feof	const	tipos
y	o	no		

B Ejemplos

B.1 Selectiva simple

```
// Algoritmo: Ejemplo de estructura selectiva simple.
// Autor: Antonio Vélez Estévez
// Fecha: 23/04/2016

Algoritmo EstructurasSelectiva01

//Sección de definición de variables constantes
const

//Sección de definición de tipos
tipo

//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal
var
        // Definimos la variable condición de tipo lógico, es decir tomará
        // dos valores únicos, verdadero o falso.
        logico : condicion
inicio

        // Asignamos a la variable condicion el valor verdadero.
        condicion ← verdadero

        // Si el valor de la variable condición es verdadero, entonces
        // se ejecutará el código que hay entre el entonces y el fin_si.
        si (condicion = verdadero) entonces
                escribir("La condicion era verdadera!")
        fin_si

        // Asignamos a la variable condicion el valor falso.
        condicion ← falso

        // Si el valor de la variable condición es verdadero(nunca pasará
        // ya que acabamos de establecer la variable condicion a falso)
        // , entonces se ejecutará el código que hay entre el entonces y el fin_si.
        si (condicion = verdadero) entonces
                escribir("Esto nunca se escribira!")
        fin_si
fin_principal

fin_algoritmo
```

B.2 Selectiva doble

```
/*
 * Algoritmo: Ejemplo de estructura selectiva doble.
 * Autor: Antonio Vélez Estévez
 * Fecha: 23/04/2016
 */

Algoritmo EstructuraSelectiva02

//Sección de definición de variables constantes
const
```

```
//Sección de definición de tipos
tipo

//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal
var
        // Definimos la variable condición de tipo lógico, es decir tomará
        // dos valores únicos, verdadero o falso.
        logico : condicion
inicio

        // Asignamos a la variable condicion el valor verdadero.
        condicion ← verdadero

        // Si el valor de la variable condición es verdadero, entonces
        // se ejecutará el código que hay entre el entonces y el fin_si.
        // Si el valor de la variable es falso, entonces se ejecutará,
        // el código entre el si_no y el fin_si.
        si (condicion = verdadero) entonces
                escribir("La condicion era verdadera!")
        si_no
                escribir("Esto nunca se escribira!")
        fin_si

fin_principal
fin_algoritmo
```

B.3 Selectiva múltiple

```
/*
 * Algoritmo: Ejemplo de estructura selectiva múltiple.
 * Autor: Antonio Vélez Estévez
 * Fecha: 23/04/2016
 */

Algoritmo EstructurasSelectiva03

//Sección de definición de variables constantes
const

//Sección de definición de tipos
tipo

//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal
var
        // Definimos la variable condición de tipo entero, es decir tomará
        // valores en dicho conjunto.
        entero : numero
inicio

        // Le muestra al usuario el siguiente mensaje por la pantalla
```

```
        escribir("Ingrese un numero de un solo digito: ")
        // Lee un valor y lo guarda en la variable numero.
        leer(numero)


        // Segun sea el numero, en el caso de que sea uno, escribira
        // Su numero es el uno.
        segun_sea(numero) hacer
                caso 1: escribir("Su numero es el uno.")
                caso 2: escribir("Su numero es el dos.")
                caso 3: escribir("Su numero es el tres.")
                caso 4: escribir("Su numero es el cuatro.")
                caso 5: escribir("Su numero es el cinco.")
                caso 6: escribir("Su numero es el seis.")
                caso 7: escribir("Su numero es el siete.")
                caso 8: escribir("Su numero es el ocho.")
                caso 9: escribir("Su numero es el nueve.")
                caso 0: escribir("Su numero es el cero.")
                en_otro_caso: escribir("No se cual es su numero, probablemente haya intr
        fin_segun

fin_principal
fin_algoritmo
```

B.4 Bucle mientras

```
/*
 * Algoritmo: Ejemplo de la estructura repetitiva mientras.
 * Autor: Antonio Vélez Estévez
 * Fecha: 23/04/2016
 */

Algoritmo EstructurasRepetitiva01

//Sección de definición de variables constantes
const

//Sección de definición de tipos
tipo

//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal
var
        // Definimos la variable condición de tipo entero, es decir tomará
        // valores en dicho conjunto.
        entero : numero
inicio
        // Le muestra al usuario el siguiente mensaje por la pantalla
        escribir("Ingrese un numero de un solo digito: ")
        // Lee un valor y lo guarda en la variable numero.
        leer(numero)

        // Le muestra al usuario el siguiente mensaje por la pantalla
        escribir("Ahora le mostrare todos los numeros que hay desde el numero que ha int

        // Ahora tenemos el número que ha introducido el usuario,
        // imaginemos que es el 992, como el 992 es menor que el
        // 1000, se ejecutará el cuerpo del mientras, se escribirá
```

```
        // el numero 992, y se incrementará en uno la variable número.

        // La última iteración será cuando el valor de numero sea
        // mil, mil es menor o igual que mil, así que entraríamos
        // en el cuerpo del mientras, incrementaríamos en uno, y
        // tendríamos 1001, al intentar entrar otra vez en el bucle
        // tenemos que comparar si 1001 es menor o igual que 1000,
        // cosa que es falsa, así que se sale del bucle y termina.
        mientras (numero ≤ 1000) hacer
                escribir("Voy por el numero ", numero)
                numero ← numero + 1
        fin_mientras


fin_principal
fin_algoritmo
```

B.5 Bucle repetir

```
/*
 * Algoritmo: Ejemplo de la estructura repetitiva repetir. Algoritmo
 * para hacer un menu.
 * Autor: Antonio Vélez Estévez
 * Fecha: 23/04/2016
 */

Algoritmo EstructurasRepetitiva02

//Sección de definición de variables constantes
const

//Sección de definición de tipos
tipo

//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal
var
        // Definimos la variable opcion de tipo entero, es decir tomará
        // valores en dicho conjunto.
        entero : opcion
inicio
        // Esta estructura es muy útil para hacer menús
        // porque al menos se repiten una vez, ya que no
        // comprueban la condición al principio.

        repetir
                escribir("Opcion 1: Volar.")
                escribir("Opcion 2: Caminar.")
                escribir("Opcion 3: Nadar.")
                escribir("Opcion 4: Salir.")
                escribir("\n Elija una: ")
                leer(opcion)
                // Seguirá repitiendo el cuerpo, hasta que opción sea cuatro.
        hasta_que (opcion = 4)

        escribir("Usted ha decidido salir")

fin_principal
```

```
fin_algoritmo
```

B.6 Bucle for

```
/*
 * Algoritmo: Ejemplo de la estructura repetitiva desde.
 * Autor: Antonio Vélez Estévez
 * Fecha: 23/04/2016
 */

Algoritmo EstructurasRepetitiva03

//Sección de definición de variables constantes
const

//Sección de definición de tipos
tipo

//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal
var
        // Límite inferior del intervalo.
        entero: limInf
        // Límite superior del intervalo.
        entero: limSup
        // Variable de control del bucle.
        entero: z
        // Acumulador.
        entero: acumulador
inicio
        // Inicializamos el acumulador a cero.
        acumulador ← 0

        escribir("Introduzca el límite inferior del intervalo:")
        leer(limInf)
        escribir("\nIntroduzca el límite superior del intervalo")
        leer(limSup)

        // Algoritmo para sumar todos los números
        // en los límites del intervalo [i,j] que
        // introduzca el usuario.

        desde z ← limInf hasta limSup hacer
                acumulador ← acumulador + z
        fin_desde

        escribir("El resultado de la suma es: ", acumulador)
fin_principal
fin_algoritmo
```

B.7 Funciones

B.7.1 Paso por valor

B.7.2 Paso por referencia

```
// Algoritmo: Ejemplo de función con paso por referencia.
// Realizar una función que dados los parámetros, vector por entrada y un real p
```

```
// or referencia, devuelva la media de los elementos del vector y guarde en el s
// egundo parámetro la suma de los elementos de dicho vector.

// Autor: Antonio Vélez Estévez
// Fecha: 01/07/2016

Algoritmo FuncionesPasoReferencia01

//Sección de definición de variables constantes
const
        MAX = 30
//Sección de definición de tipos
tipo
        vector [MAX] de entero: Vector
//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos
real funcion operacionesVector(E Vector: vect, E/S real: suma)
var
        entero: i
inicio
        suma ← 0

        desde i ← 0 hasta MAX - 1 hacer
                suma ← suma + vect[i]
        fin_desde

        devolver suma div MAX
fin_funcion

//Comienzo del algoritmo (Obligatorio)
principal

var
        entero: i
        Vector: vect
       real: media
       real: suma
inicio
        desde i ← 0 hasta MAX - 1 hacer
                vect[i] ← i
        fin_desde

        media ← operacionesVector(vect, suma)

        escribir("La media es ", media)
        escribir("La suma es", suma)
fin_principal
fin_algoritmo
```

B.8 Procedimientos

B.8.1 Paso por valor

```
// Algoritmo: Ejemplo de procedimiento con paso por valor.
// Autor: Antonio Vélez Estévez
// Fecha: 30/06/2016

Algoritmo ProcedimientosPasoReferencia01

//Sección de definición de variables constantes
```

```
const
//Sección de definición de tipos
tipo
//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos
procedimiento dimeNumero(E entero: x)

inicio

        segun_sea(x) hacer
                caso 1: escribir("Su numero es el uno.")
                caso 2: escribir("Su numero es el dos.")
                caso 3: escribir("Su numero es el tres.")
                caso 4: escribir("Su numero es el cuatro.")
                caso 5: escribir("Su numero es el cinco.")
                caso 6: escribir("Su numero es el seis.")
                caso 7: escribir("Su numero es el siete.")
                caso 8: escribir("Su numero es el ocho.")
                caso 9: escribir("Su numero es el nueve.")
                caso 0: escribir("Su numero es el cero.")
                en_otro_caso: escribir("No se cual es su numero, probablemente
        haya introducido algo fuera del rango establecido.")
        fin_segun

fin_procedimiento
//Comienzo del algoritmo (Obligatorio)
principal

var
         entero: numero
inicio
     escribir("Introduzca un número entre 0 y 9 y le diré con
     letras qué numero es:")
         leer(numero)
         dimeNumero(numero)
fin_principal
fin_algoritmo
```

B.8.2 Paso por referencia

```
// Algoritmo: Ejemplo de procedimiento con paso por referencia
// Autor: Antonio Vélez Estévez
// Fecha: 30/06/2016


Algoritmo ProcedimientosPasoReferencia01

//Sección de definición de variables constantes
const
//Sección de definición de tipos
tipo
//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos
procedimiento intercambiar(E/S real: x, E/S real: z)
var
       real: aux
inicio
        aux ← x
        x ← z
        z ← aux
```

```
fin_procedimiento
//Comienzo del algoritmo (Obligatorio)
principal

var
        real: numero1
        real: numero2
inicio
         numero1 ← 5
         numero2 ← 8
         escribir("La variable numero1 tiene el valor: ", numero1)
         escribir("La variable numero2 tiene el valor: ", numero2)

         escribir("Intercambiando...")

         intercambiar(numero1, numero2)

         escribir("La variable numero1 tiene el valor: ", numero1)
         escribir("La variable numero2 tiene el valor: ", numero2)
fin_principal
fin_algoritmo
```

B.9 Registros

```
// Algoritmo: Ejemplo de registro.
// Autor: Antonio Vélez Estévez
// Fecha: 01/07/2016

Algoritmo Registros01

//Sección de definición de variables constantes
const
//Sección de definición de tipos
tipo
        registro: Persona
                cadena: nombre
                cadena: apellido
                entero: edad
                entero: NIF
        fin_registro
//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal

var
        Persona: per
inicio
        escribir("Introduzca el nombre de la persona(máx. 30 caracteres):")
        leer(per.nombre)

        escribir("Introduzca el apellido de la persona(máx. 30 caracteres):")
        leer(per.apellido)

        escribir("Introduzca la edad de la persona:")
        leer(per.edad)

        escribir("Introduzca el NIF de la persona:")
        leer(per.NIF)
```

```
        escribir("Su persona es:")
        escribir("Nombre: ", per.nombre)
        escribir("Apellido: ", per.apellido)
        escribir("Edad: ", per.edad)
        escribir("NIF: ", per.NIF)


fin_principal
fin_algoritmo
```

B.10 Registros anidados

```
/*
 * Algoritmo: Ejemplo de registros anidados.
 * Autor: Antonio Vélez Estévez
 * Fecha: 01/07/2016
 */

Algoritmo RegistrosAnidados01

//Sección de definición de variables constantes
const
//Sección de definición de tipos
tipo
        registro: Coche
                cadena: marca
                entero: numBastidor
                cadena: matricula
        fin_registro

        registro: Persona
                cadena: nombre
                cadena: apellido
                entero: edad
                entero: NIF
                Coche: coche
        fin_registro
//Sección de declaración de variables globales
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal

var
        Persona: per
inicio
        escribir("Introduzca el nombre de la persona(máx. 30 caracteres):")
        leer(per.nombre)

        escribir("Introduzca el apellido de la persona(máx. 30 caracteres):")
        leer(per.apellido)

        escribir("Introduzca la edad de la persona:")
        leer(per.edad)

        escribir("Introduzca el NIF de la persona:")
        leer(per.NIF)

        escribir("Introduzca la marca del coche de la persona:")
        leer(per.coche.marca)
```

```
        escribir("Introduzca el número de bastidor del coche de la persona:")
        leer(per.coche.numBastidor)

        escribir("Introduzca la matrícula del coche:")
        leer(per.coche.matricula)

        escribir("Su persona es:")
        escribir("Nombre: ", per.nombre)
        escribir("Apellido: ", per.apellido)
        escribir("Edad: ", per.edad)
        escribir("NIF: ", per.NIF)
        escribir("Además esta persona tiene un coche cuyos datos son:")
        escribir("Marca: ", per.coche.marca)
        escribir("Número de bastidor: ", per.coche.numBastidor)
        escribir("Matrícula: ", per.coche.matricula)

fin_principal
fin_algoritmo
```

B.11 Matrices

```
/*
 * Algoritmo: Ejemplo de matrices. Comprueba si una matriz es la identidad.
 * Autor: Antonio Vélez Estévez
 * Fecha: 01/07/2016
 */

Algoritmo Matrices01

//Sección de definición de variables constantes
const
        MAX = 3
//Sección de definición de tipos
tipo
        matriz [MAX][MAX] de entero: Matriz
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal

var
        Matriz : mat
        entero : i
        entero : j
        logico : esIdentidad
inicio
    escribir("A continuacion introducira los elementos de la matriz para comprobar
    si es identidad")
        desde i ← 0 hasta MAX - 1 hacer
                desde j ← 0 hasta MAX - 1 hacer
                        escribir("Introduzca el elemento ", i, ", ", j)
                        leer(mat[i][j])
                fin_desde
        fin_desde

        esIdentidad ← verdadero

        i ← 0
        mientras(i < MAX y esIdentidad) hacer
                j ← 0
                mientras(j < MAX y esIdentidad) hacer
```

```
                    // Si estamos en la diagonal el elemento debe ser uno si no,
            // no será identidad.
                        si(i = j) entonces
                                si(mat[i][j] ≠ 1) entonces
                                        esIdentidad ← falso
                                fin_si
                        // Si no estamos en la diagonal el elemento debe ser cero si
                        // no, no será identidad.
                        si_no
                                si(mat[i][j] ≠ 0) entonces
                                        esIdentidad ← falso
                                fin_si
                        fin_si
                        j ← j + 1
                fin_mientras
                i ← i + 1
        fin_mientras

        si(esIdentidad = verdadero) entonces
                escribir("La matriz introducida era la identidad.")
        si_no
                escribir("La matriz introducida no era la identidad.")
        fin_si

fin_principal
fin_algoritmo
```

B.12 Vectores

```
/*
 * Algoritmo: Ejemplo de vectores.
 * Autor: Antonio Vélez Estévez
 * Fecha: 01/07/2016
 */

Algoritmo Vectores01

//Sección de definición de variables constantes
const
        MAX = 10
//Sección de definición de tipos
tipo
        vector [MAX] de entero: Vector
var

//Sección de definición de subalgoritmos: funciones y procedimientos

//Comienzo del algoritmo (Obligatorio)
principal

var
        Vector : numeros
        entero : i
        real   : suma
        real   : media
inicio
    escribir("A continuación ingresará una serie de numeros para calcular la
    media y la suma de los mismos")
        desde i ← 0 hasta MAX - 1 hacer
                escribir("Ingrese el elemento ", i)
                leer(numeros[i])
                suma ← suma + numeros[i]
        fin_desde
```

```
        media ← suma div MAX

        escribir("La media del vector introducido es: ", media)
        escribir("La suma del vector introducido es:", suma)

fin_principal
fin_algoritmo
```