

1.-Especificación y Verificación Formal. (3 puntos).

- a) Explique los conceptos de invariante y función de cota.
 b) Indique si los siguientes algoritmos satisfacen sus especificaciones respectivas.
 Demuestre los resultados.

b.1) var s:booleano; a,b:entero fvar
si (a+b) mod 2 = 0 → s:=verdad
 (a+b) mod 2 ≠ 0 → s:=falso
fsi
 $P \equiv \{ a=A \wedge B=b \wedge a,b \neq 0 \}$
 $Q \equiv \{ s=verdad \vee s=falso \}$

b.2) var a,b,c:entero fvar
 c:=1;
mientras c≤b hacer
 r:=a-c;
 c:=c+1
fmientras
 $P \equiv \{ a=A \wedge B=b \wedge r \geq 0 \wedge b,a > 0 \}$
 $Q \equiv \{ r=a-b \wedge r \geq 0 \}$
 Invariante ≡ { r ≥ 0 ∧ c > 0 }
 Función de cota ≡ c

AYUDA.-

Asignación:
$$P \Rightarrow Q$$

$$E$$

Alternativa
$$P \Rightarrow B_1 \vee B_2$$

$$P \Rightarrow \underline{\text{no}} (B_1 \wedge B_2)$$

$$\{ P \wedge B_i \}$$

$$S_i \{ Q \}$$

Iterativa:
$$P \Rightarrow I$$

$$\{ I \wedge B \}$$

$$S \{ I \}$$

$$I \wedge \underline{\text{no}} B \Rightarrow Q$$

$$I \wedge B \Rightarrow t > 0$$

$$\{ I \wedge B \wedge t = T \}$$

$$S \{ t < T \}$$

2.- Tablas y diseño descendente.- (3,5 puntos)

Dada una estructura de tabla de 100 caracteres completamente llena empleada para almacenar palabras separadas unas de otras por al menos un carácter blanco, pudiendo existir blancos iniciales en la tabla:

- a) Dar la declaración de tipo de dicha estructura.
 b) Diseñar una acción que tome como parámetro de entrada una variable de ese tipo y ofrezca como parámetro de salida una variable del mismo tipo en la que se han seleccionado las palabras que empezaban por "A".
 c) Justificar qué tipo de esquemas se ha utilizado para la acción.

Ejemplo

Entrada:

_	_	L	O	S	_	A	N	I	M	A	L	E	S	_	_	C	O	M	O
_	L	A	S	_	_	A	R	D	I	L	L	A	S		

Salida:

A	N	I	M	A	L	E	S	_	A	R	D	I	L	L	A	S	.	.	.
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

3.- Diseño Modular.- (3,5 puntos).-

Para llevar el control de un almacén que tiene gran variedad de piezas y teniendo disponible el módulo modupieza , complete el módulo ALMACEN que viene a continuación:

Módulo modupieza;

exporta

tipo tpieza;

(* para contener información sobre una pieza: nombre, precio, proveedor... *)

tipo tproveedor;

(* información relativa sobre un proveedor de piezas*)

función stock (p : tpieza) : entero;

(* devuelve el número de unidades que hay de la pieza **p** *)

función stockminimo (p : tpieza) : entero;

(* devuelve el número mínimo de unidades que debe de haber de la pieza **p** *)

función nombrepieza (p : tpieza) : tabla[1..20] de carácter;

(* devuelve el nombre de la pieza **p** *)

acción proveedores (p : tpieza; var prov: tabla[1..3] de tproveedor; var n: entero)

(* pone en **prov**, en sus primeras **n** posiciones, los proveedores de la pieza **p** *)

función nombreproveedor (t : tproveedor) : tabla[1..30] de carácter;

(* devuelve el nombre del proveedor **t** *)

función comparar (a,b : tabla[1..20] de carácter) : booleano;

(* devuelve cierto si **a** y **b** son idénticas, en caso contrario falso *)

fexporta

fmódulo

Módulo ALMACEN;

...

exporta

tipo tpiezas;

(* información relativa a un grupo de piezas ... *)

acción pedidos (a : tpiezas; n : entero);

(* sacará por pantalla para las **n** piezas que hay en **a** cuyo stock sea inferior al stockmínimo el nombre de la pieza y el nombre de los proveedores de esa pieza *) (* esquema de recorrido*)

función piezarepetida (a : tpiezas; n : entero) : booleano;

(* devolverá cierto si al menos hay una pieza repetida con el mismo nombre dentro de las **n** piezas que hay en **a**, en caso contrario devolverá falso *) (* esquema de búsqueda *)

fexporta

implementación

tipo tpiezas = tabla[1..500] de tpieza;

(* si hay **m** piezas estarán en la tabla desde la posición **1** hasta la **m** *)

...

fimplementación

fmódulo