Segundo Parcial. Programación 1

Instituto de Computación
Noviembre 2024

Ejercicio 1 (25 puntos)

Considerando las siguientes estructuras del laboratorio:

```
const
    MAX_LARGO_TEXTO = ...; (*valor mayor que 0*)
    MAX_USUARIOS = ...; (*valor mayor que 0*)

type
    Texto = record
        tex : array[1..MAX_LARGO_TEXTO] of char;
        tope : 0..MAX_LARGO_TEXTO
    end;

    TUsuario = record
        usuario: Texto;
        serviciosUsuario: TServicios
    end;

    TGestorContrasenia = record
        usuarios: array [1..MAX_USUARIOS] of TUsuario;
        tope:0..MAX_USUARIOS
    end;
```

donde `TServicios` no se especifica dado que no se utilizará. Y el subprograma auxiliar para comparar textos:

```
function igualTexto(tx1,tx2 : Texto) : boolean;
{ Devuelve True si los dos textos ingresados son iguales, false en caso contrario }
```

el cual puede ser invocado directamente sin necesidad de implementarlo. Se definen las siguientes estructuras:

```
const
    MAX_NOMBRES = ...; (*valor mayor que 0*)
type
    TEstado = (usuarioInexistente, usuarioUnico, usuarioRepetido);

    TEstadoUsuario = record
        us : Texto;
        estadoUsuario : TEstado
    end;

    TNombres = array [1..MAX_NOMBRES] of Texto;

    TAuditoriaUsuarios = array [1..MAX_NOMBRES] of TEstadoUsuario;
```

Asuma que no es necesario realizar la autenticación de los usuarios en ningún caso.

Suponga que se ha detectado una falla en un sistema de gestión de contraseñas y se necesita verificar la presencia de ciertos usuarios en el gestor, determinando si no existen, si están registrados una sola vez o si aparecen múltiples veces.

Parte A)

Escriba el procedimiento:

```
procedure revisionUsuario(nombre:Texto; gc:TGestorContrasenia; var estUsr : TEstadoUsuario);
```

que dado un nombre de usuario `nombre` y un gestor de contraseñas `gc`, devuelve en `estUsr` el nombre `nombre` del usuario y si el usuario no existe, existe una única vez o si aparece 2 o más veces en el gestor `gc`.

Parte B)

Se quiere proceder a auditar el estado de determinados usuarios. Escriba el procedimiento:

```
procedure auditarUsuarios(gc:TGestorContrasenia; nombres : TNombres;
                          var auditoria : TAuditoriaUsuarios);
```

tal que para cada usuario cuyo nombre aparezca en `nombres`, devuelva en `auditoria` el nombre del usuario y si el usuario no existe, existe una única vez o múltiples veces en el gestor `gc`. No hay nombres repetidos en `nombres`. Para realizar este procedimiento debe utilizar el procedimiento `revisionUsuario` de la parte A.

Solución:

```
procedure revisionUsuario(nombre:Texto; gc:TGestorContrasenia; var estUsr : TEstadoUsuario);
    var i, cont : integer;
begin
    i := 1;
    cont := 0;
    estUsr.us := nombre;
    while (i <= gc.tope) and (cont < 2) do
    begin
        if igualTexto(nombre,gc.usuarios[i].usuario) then
            cont := cont + 1;
        i := i + 1
    end;
    case cont of
        0: estUsr.estadoUsuario := usuarioInexistente;
        1: estUsr.estadoUsuario := usuarioUnico;
        2: estUsr.estadoUsuario := usuarioRepetido
    end
end;

procedure auditarUsuarios(gc:TGestorContrasenia; nombres : TNombres;
var auditoria : TAuditoriaUsuarios);
var i : integer;
begin
    for i := 1 to MAX_NOMBRES do
        revisionUsuario(nombres[i], gc, auditoria[i]);
end;
```

Ejercicio 2 (10 puntos)

Dadas las siguientes definiciones:

```
type
    Lista = ^TipoCelda;
    TipoCelda = record
        dato: integer;
        sig: Lista
    end;
```

Escriba el procedimiento:

```
procedure revertirLst(lst: Lista; var invertida : Lista);
```

que dada la lista `lst` invierte el orden y retorna la nueva lista invertida en `invertida`. Si la lista es vacía queda vacía. Las listas no deben compartir memoria y la lista original no debe modificarse. La solución debe realizar una **única** recorrida de la lista.

Ejemplo:

Si `lst` es $\boxed{1} \to \boxed{4} \to \boxed{7} \to \boxed{3}$, entonces `invertida` debe ser $\boxed{3} \to \boxed{7} \to \boxed{4} \to \boxed{1}$.

Solución:

```
procedure revertirLst(lst: Lista; var invertida : Lista);
var
    aux, nuevo: Lista;
begin
    invertida := nil;
    aux := lst;

    while aux <> nil do
    begin
        new(nuevo);
        nuevo^.dato := aux^.dato;
        nuevo^.sig := invertida;

        invertida := nuevo;

        aux := aux^.sig;
    end;
end;
```

Ejercicio 3 (16 puntos)

Dadas las siguientes definiciones:

```
const
    N = ...;    { valor mayor que cero }

type
    arregloTxt  = array[1..N] of char;
    arregloHist = array['a'..'z'] of integer;
```

Escriba el procedimiento:

```
procedure calcularHistograma(arrC : arregloTxt; var hist : arregloHist);
```

tal que, dado un arreglo `arrC` de caracteres, calcule la cantidad de ocurrencias de cada letra minúscula y almacene las cantidades en el arreglo `hist`. Si una letra del alfabeto no está presente en `arrC`, su valor en `hist` debe ser 0.

Ejemplo:

arrC =	'P'	'a'	'r'	'c'	'i'	'a'	'l'	' '	'2'	'4'
	1	2	3	4	5	6	7	8	9	10

hist =	2	0	1	...	1	...	1	0	0	0	0	0	1	...	0
	'a'	'b'	'c'	...	'i'	...	'l'	'm'	'n'	'o'	'p'	'q'	'r'	...	'z'

Solución:

```
procedure calcularHistograma(arrC : arregloTxt; var hist : arregloHist);
var c : 'a'..'z';
    i : integer;
begin
    for c := 'a' to 'z' do
        hist[c] := 0;

    for i := 1 to N do
        if (arrC[i] in ['a'..'z']) then
            hist[arrC[i]] := hist[arrC[i]] + 1;
end;
```

Ejercicio 4 (9 puntos)

Considere el siguiente programa:

```
program alcance;
var a, b, n : integer;

function f(a,n,b : integer) : boolean;
begin
    f := (a <= n) and (n <= b)
end;

procedure p(a,n,z : integer; var b : integer);
begin
    ...
end;

begin
    readln(a,b,n);
    p(a,n,b,n);
    if n = 1 then
        write('Si')
    else if n = 0 then
        write('No')
end.
```

en el que no se incluye la implementación del procedimiento `p`.

Dados los valores leídos por entrada estándar de `a`, `b` y `n`, el programa debe desplegar *Si* cuando se cumple alguna de las siguientes condiciones:

- $n \in [a, b]$, es decir, $(a \leq n \leq b)$
- $n^2 \in [a, b]$, es decir, $(a \leq n^2 \leq b)$

y debe desplegar *No* en caso contrario.

Implemente el procedimiento `p` sin utilizar operadores relacionales (=, <=, =>, >, <, <>).

Solución:

```
procedure p (a,n,z : integer; var b : integer);
begin
    if f(a,n,z) or f(a,sqr(n),z) then
        b := 1
    else
        b := 0
end;
```