Examen - Programación 1

Instituto de Computación
Diciembre 2022

Leer con atención:

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje Pascal tal como fue dado en el curso.
- En todos los problemas se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso.
- Escribir las respuestas de un solo lado de la hoja.
- Entregue solamente la carátula y las hojas de solución escritas a lápiz.
- En cada hoja entregada se debe incluir nombre, cédula, nro. de examen y qué número de hoja es del total entregadas.

Ejercicio 1 (50 puntos)

Se consideran las siguientes definiciones:

```
const
  MaxProd = . ..  {valor entero positivo}
  MaxPedidos =  ...  {valor entero positivo}

type
  TProducto = 1 .. MaxProd;
  TProducto0 = 0 .. MaxProd;
  TPedido = record
               producto : TProducto;
               cantidad : integer
            end;
  RangoPedidos = 1 .. MaxPedidos;
  TPedidos = record
             info  : array [RangoPedidos] of TPedido;
             tope  : 0 .. MaxPedidos
           end;
  TTotalesPorProducto = array [TProducto] of integer;
```

Cada pedido tiene un código de producto y la cantidad de unidades del producto que se solicita. Para cada producto puede haber varios pedidos y algunos productos pueden no tener ningún pedido. Los pedidos son almacenados en cualquier orden.

a) (15 puntos) Escribir la función:

```
function TienePedidos(prod:TProducto;
                      pedidos:TPedidos) : boolean;
```

que retorna true si y sólo si existe al menos un pedido para el producto `prod` en `pedidos`, que es la secuencia de todos los pedidos de productos realizados en un cierto periodo.

Solución:

```
function TienePedidos(prod: TProducto;
                        pedidos: TPedidos) : boolean
                            ;
var i : integer;
begin
    i := 1;
    with pedidos do
    begin
       while (i <= tope) and
             (info[i].producto <> prod) do
           i := i + 1;
    TienePedidos := i <= tope
    end
end;
```

b) (20 puntos) Escribir la función:

```
function ProdSinPedidos(pedidos:TPedidos):TProducto0;
```

Que retorna el código del producto con código más bajo que no tiene ningún pedido. Si todos los productos tienen pedidos retorna 0. No se debe usar ninguna estructura auxiliar.

Solución:

```
function ProdSinPedidos(pedidos : TPedidos) :
    TProducto0;
var i : integer;
begin
    i := 1;
    while (i <= MaxProd) and
          TienePedidos(i, pedidos) do
        i := i + 1;
    if i <= MaxProd then
        ProdSinPedidos := i
    else
        ProdSinPedidos := 0
end;
```

c) (15 puntos) Escribir el procedimiento:

```
procedure CalcularTotales(pedidos : TPedidos;
                var totales : TTotalesPorProducto);
```

El procedimiento recibe en el parámetro `pedidos` todos los pedidos de productos realizados en un cierto periodo. El procedimiento debe construir y retornar el parámetro `totales`. Este parámetro es un arreglo que contiene en la celda de índice `k`, la cantidad total de unidades del producto `k` que fueron solicitadas (considerando todos los pedidos).

Ejemplo. Si suponemos que `MaxProd` $= 6$, a partir de los siguientes `pedidos` se obtiene `totales` como se muestra:

```
pedidos =
   - info = (producto: 1, cantidad: 20),
            (producto: 3, cantindad: 5),
            (producto: 1, cantidad: 11),
            (producto: 3, cantidad: 2),
            (producto: 4, cantidad: 9)
   - tope = 5

totales = [31, 0, 7, 9, 0, 0 ]
```

Solución:

```
procedure CalcularTotales(pedidos : TPedidos; var
    totales : TTotalesPorProducto);
var i : integer;
begin
    for i := 1 to MaxProd do
        totales[i] := 0;
    for i := 1 to pedidos.tope do
    with pedidos.info[i] do
        totales[producto] := totales[producto] +
                             cantidad;
end;
```

Ejercicio 2 (35 puntos)

Considere la definición de las siguientes estructuras en Pascal:

```
type
   ListaVeleros = ^NodoVelero;
   NodoVelero = record
                   disponible : boolean;
                   capacidad : integer;
                   sig : ListaVeleros;
                end;
```

El tipo `ListaVeleros` permite representar un conjunto de veleros, por ejemplo aquellos amarrados en un puerto. De cada velero se tiene la información de si se encuentra disponible para navegar y su capacidad máxima de pasajeros.

a) (20 puntos) Escribir la función:

```
function capacidadMaximaPasajeros
              (veleros:ListaVeleros): integer;
```

que retorna la cantidad máxima de personas que pueden viajar desde un puerto, considerando los veleros que se encuentran `disponibles`, la `capacidad` máxima de personas que pueden llevar y que en un viaje participan todos los veleros disponibles

Solución:

```
function capacidadMaximaPasajeros
              (veleros: listaVeleros): integer;
var acum: integer;
begin
   acum := 0;
   while veleros <> nil do
   begin
      if veleros^.disponible  then
          acum := acum + veleros^.capacidad;
      veleros := veleros^.sig
   end;
   capacidadMaximaPasajeros := acum
end;
```

b) (15 puntos) Escribir el procedimiento:

```
procedure nuevoVelero (capMax:integer;
                       var veleros: ListaVeleros);`
```

que agrega un nuevo velero al final de la lista de veleros. El nuevo velero deberá comenzar como disponible y tendrá como capacidad la cantidad máxima de pasajeros recibida en `capMax`.

Solución:

```
procedure nuevoVelero(capMax:integer;
                      var veleros: ListaVeleros);
var nuevo, p : ListaVeleros;
begin
    new(nuevo);
    nuevo^.disponible := true;
    nuevo^.capacidad := capMax;
    nuevo^.sig := nil;
    if veleros <> nil then
    begin
        p := veleros;
        while p^.sig <> nil do
            p := p^.sig;
        p^.sig := nuevo;
    end
    else
        veleros := nuevo
end;
```

Ejercicio 3 (15 puntos)

Determinar la salida del siguiente programa, cuando se le da como entrada el dígito de las unidades (el último dígito antes del guión) de su número de cédula. Por ejemplo, si su cédula es 1234567-8, la entrada será 7.

```
program Alcance;
var  x,y,z: integer;
   procedure suma (a : integer; var b, c : integer);
      function calc (a, b : integer) : integer;
      var i : integer;
      begin
         i := 1;
         a := a + i;
         calc := a;
      end;
   begin
      b := calc(y,b);
      writeln(b);
      c := c + a
   end;
   function evalua (b : integer): integer;
   begin
      if b < 5 then
         evalua := b * 3
      else
         evalua := b * 2
   end;
begin
   readln(x);
   y := evalua (x);
   writeln(y);
   z := 0;
   suma (x,y,z);
   writeln(x,y,z)
end.
```

Solución:

Entrada	Salida
0	0 1 0 1 0
1	3 4 1 4 1
2	6 7 2 7 2
3	9 10 3 10 3
4	12 13 4 13 4
5	10 11 5 11 5
6	12 13 6 13 6
7	14 15 7 15 7
8	16 17 8 17 8
9	18 19 9 19 9