


IBD - 2008

**Registros de
Longitud Variable.**

- 
- Creación de un archivo de nombres de personas o apertura de uno existente para ver su contenido en pantalla o actualizarlo (altas, bajas o modificaciones de registros).
 - Para la creación, los nombres se ingresan desde teclado, y la carga finaliza con un nombre nulo, que no se incorpora al archivo.
 - Los registros se almacenan con longitud variable y en bloques, y el espacio libre se controla con un archivo de words cada uno de cuyos registros indica los bytes libres en el bloque con mismo número de orden.

Program Personas;

Const

TamBloque = 64;

CapacBloque = TamBloque-SizeOf(Word);

PorcCarga = 0.9; {Porcentaje de carga de los bloques (se deja espacio disponible por si se actualizan registros aumentando su longitud, para no tener que cambiarlos de bloque)}

PersonaFin = 'fin'; {para representar el fin del archivo}

Type

tNombre = String[31]; {para manejar personas en memoria}

tBloque = **Record**

 cantRegs: Word; {cantidad actual de registros en el
bloque}

 contenido: Array[1..CapacBloque] of Byte

end;

tPersonas = **Record**

 arch: File of tBloque; {archivo de personas}

 espLibre: File of Word; {archivo de control de bytes
libres por bloque del archivo de personas}

 bloque: tBloque; {ultimo bloque leído de personas}

 iBloque: Word; {índice de posición dentro del bloque}

 espLibreBloque: Word

end;



Var

```
opc: Byte; {opción}  
nomArch: String; {nombre de archivo  
que se solicita al usuario}  
pp: tPersonas;  
persona: tNombre; {variable para  
leer una persona}
```



Begin {Principal}

Repeat

WriteLn('PERSONAS');

WriteLn('0. Terminar el Programa');

WriteLn('1. Crear un archivo');

WriteLn('2. Abrir un archivo existente');

Write('Ingrese el nro. de opción: '); ReadLn(opc);

If (opc=1) or (opc=2) **then begin**

Write('Nombre del archivo (sin extension): ');

ReadLn(nomArch);

Assign(pp.arch, nomArch + '.dd');

Assign(pp.espLibre, nomArch + '.el');

end;



Case opc of

1: begin

Crear(pp);

Write('Nombre (para terminar ingrese un nombre nulo): ');

ReadLn(persona);

While (persona <> '') **do begin**

Cargar(pp, persona);

Write('Nombre: ');

ReadLn(persona);

end;

If (pp.bloque.cantRegs > 0) **then**


With pp **do begin** {se escribe el último bloque}

Write(arch, bloque); **Write**(espLibre, espLibreBloque)

end;


Close(pp.arch); **Close**(pp.espLibre);

end;



```
2: Procesar(pp); {opción numero 2}  
end; {Case}  
ClrScr;  
Until opc = 0;  
end. {Principal }
```

¿Preguntas?



Procedure Procesar(**var** pp: tPersonas);

Var

o: Byte; {opción}

cl: Byte; {contador de líneas}

p, pb: tNombre; {persona y persona a buscar}

resto: Word; {cantidad de bytes a mover

cuando se borra una persona para

compactar el bloque}

tamReg: Byte;



Begin

Repeat

```
WriteLn('PERSONAS');
```

```
WriteLn;
```

```
WriteLn('3. Listar el archivo');
```

```
WriteLn('4. Agregar una persona');
```

```
WriteLn('5. Dar de baja a una persona');
```

```
WriteLn('6. Modificar el nombre de una  
persona');
```

```
WriteLn('7. Terminar con este archivo');
```

```
WriteLn;
```

```
Write('Ingrese el nro. de opción: '); ReadLn(o);
```

Case o of

3: **begin**

Primero(pp, p);

cl:=0;

While (p <> PersonaFin) **do begin**

WriteLn(p);

Inc(cl);

If (cl=18) **then begin**

cl:=0;

Write('Oprima Entrar para continuar...');

ReadLn;

end;

Siguiente(pp, p)

end;

Write('Oprima Entrar para continuar...');

ReadLn;

end;



4: begin

WriteLn;

Write('Nombre: ');

ReadLn(p);

Agregar(pp, p);

end;

5: begin

```
Write('Nombre de la persona a dar de baja: ');
ReadLn(pb); Primero(pp, p);
While (p <> PersonaFin) and (p <> pb) do Siguiente(pp, p);
If (p = pb) then
    With pp do begin {se encuentra a la persona}
        resto:= CapacBloque – espLibreBloque – iBloque + 1;
        tamReg:= Length(p) + 1;
        If (resto> 0) then Move(bloque.contenido[iBloque],
                                bloque.contenido[iBloque-tamReg], resto);
        Dec(bloque.cantRegs); Inc(espLibreBloque, tamReg);
        Seek(arch, FilePos(arch)-1); Write(arch, bloque);
        Seek(espLibre, FilePos(espLibre) - 1);
        Write(espLibre, espLibreBloque);
        Close(arch); Close(espLibre)
    end {With}
else begin {no se encuentra a la persona}
    WriteLn('No existe tal persona. '); Write('Entrar para continuar... ');
    ReadLn;
end;
end;
```

6: begin

```
Write('Nombre a buscar: ');  
ReadLn(pb); Primero(pp, p);  
While (p<>PersonaFin) and (p<>pb) do Siguiente(pp, p);  
If (p=pb) then  
  with pp do begin  
    resto:=CapacBloque – espLibreBloque – iBloque + 1;  
    tamReg:= Length(p) + 1;  
    If (resto>0) then Move(bloque.contenido[iBloque],  
                           bloque.contenido[iBloque-tamReg], resto);  
    Inc(espLibreBloque, tamReg);  
    iBloque:=CapacBloque-espLibreBloque+1;  
    Write('Nuevo nombre: ');  
    ReadLn(p); tamReg:= Length(p) + 1;
```

If (tamReg>espLibreBloque) **then begin** {en este caso no se considera el factor de carga}

Seek(arch, FilePos(arch) - 1); **Write**(arch, bloque);

Seek(espLibre, **FilePos**(espLibre)-1);

Write(espLibre, espLibreBloque);

Close(arch); **Close**(espLibre); Agregar(pp, p)

end

else begin

Move(p, bloque.contenido[iBloque], tamreg);

Dec(espLibreBloque, tamReg);

Seek(arch, **FilePos**(arch) - 1); **Write**(arch, bloque);


Seek(espLibre, **FilePos**(espLibre) - 1);

Write(espLibre, espLibreBloque);

Close(arch); **Close**(espLibre)

end;


end {With}



```
else begin {del primer if}  
    WriteLn;  
    WriteLn('No existe tal persona.');
```

Write('Oprima Entrar para continuar...');

```
    ReadLn  
  
end;  
End {opción 6}  
end; {case}  
ClrScr;  
Until (o = 7)  
end; {Proceso}
```

```
Procedure Crear(var pp: tPersonas);  
{Inicializa la estructura y crea los archivos. Los  
archivos deben estar asignados}
```

```
Begin
```

```
With pp do begin
```

```
    Rewrite(arch); Rewrite(espLibre);
```

```
    bloque.cantRegs:= 0;
```

```
    iBloque:= 1;
```

```
    espLibreBloque:= CapacBloque;
```

```
end;
```

```
end;
```



Procedure Cargar(**var** pp: tPersonas; p: tNombre);

{uso secuencial}

{Agrega una persona en el bloque actual o comienza un nuevo bloque al final del archivo (no se busca espacio libre). El bloque actual es el ultimo del archivo}


Var

tamReg: Byte;

disponibles: Word; {bytes disponibles en un bloque, sin exceder el porc. de carga}

Begin

```
tamReg:= Length(p) + 1; {longitud del nombre mas un byte por el
prefijo de longitud}
disponibles:=pp.espLibreBloque-Round((1-PorcCarga)*CapacBloque);
If (tamReg>disponibles) then {el registro sobrecarga al bloque actual}
    With pp do begin
        Write(arch, bloque); Write(espLibre, espLibreBloque);
        bloque.cantRegs:=0; espLibreBloque:=CapacBloque;
        iBloque:=1;
    end;
With pp do begin {se agrega el registro al bloque actual}
    Move(p, bloque.contenido[iBloque], tamReg);
    Inc(iBloque, tamReg);
    Inc(bloque.cantRegs);
    Dec(espLibreBloque, tamReg);
end;
end; {Proceso}
```



Procedure Agregar(**var** pp: tPersonas; p: tNombre); {uso individual}
{Agrega una persona en el primer bloque que tenga espacio o, si no hay ninguno, en un bloque nuevo al final del archivo}

Var

tamReg: Byte;
libres, disponibles: Word;

Begin

With pp **do begin**

Reset(arch); **Reset**(espLibre);


tamReg:= Length(p) + 1;

Repeat


Read(espLibre, libres);

disponibles:=libres-Round((1- PorcCarga)*CapacBloque);

Until EOF(espLibre) or (disponibles>=tamReg);



```
If (disponibles < tamReg) then begin {hay que agregar un nuevo  
    bloque al final del archivo}  
    bloque.cantRegs:=0;  
    iBloque:=1;  
end  
else begin  
    Seek(arch, FilePos(espLibre)-1); Read(arch, bloque);  
    iBloque:= CapacBloque - libres + 1;  
    Seek(arch, FilePos(espLibre)-1); Seek(espLibre, FilePos(espLibre)-1)  
end;  
Move(p, bloque.contenido[iBloque], tamReg);  
Inc(bloque.cantRegs);  
Dec(libres,tamReg);  
Write(arch, bloque); Write(espLibre, libres);  
Close(arch); Close(espLibre)  
end; {With}  
end; {Proceso}
```



```
Procedure Primero(var pp: tPersonas; var p: tNombre); {inicio  
de secuencia} {Devuelve el primer registro de persona. El  
archivo debe tener al menos una persona}
```

```
Var
```

```
    tamReg: Byte;
```

```
Begin
```

```
With pp do begin
```

```
    Reset(arch); Read(arch, bloque);
```


```
    tamReg:=bloque.contenido[1]+1;
```

```
    Move(bloque.contenido[1], p, tamReg);
```

```
    iBloque:=tamReg+1; Reset(espLibre); Read(espLibre,  
    espLibreBloque);
```

```
    WriteLn('Bloque 0: ', bloque.cantRegs, ' personas - ',  
    espLibreBloque, ' bytes libres')
```

```
end; end;
```



Procedure Siguiente(**var** pp: tPersonas; **var** p: tNombre); {uso
secuencial} {Devuelve la siguiente persona a partir de la ultima
devuelta, o PersonaFin si la ultima devuelta fue la ultima del archivo}

Var

tamReg: Byte;

Begin

With pp **do begin**

If (iBloque>CapacBloque-espLibreBloque) and (not **eof**(arch)) **then**


begin {no hay

más registros en el bloque}

Read(arch, bloque); iBloque:=1; **Read**(espLibre, espLibreBloque);

WriteLn('Bloque ', **FilePos**(arch) - 1, ': ', bloque.cantRegs, '
personas - ', espLibreBloque, ' bytes libres');

end;



If (iBloque<CapacBloque-espLibreBloque) **then**
begin

tamReg:=bloque.contenido[iBloque]+1;

Move(bloque.contenido[iBloque], p, tamReg);

Inc(iBloque, tamReg);

end

else begin

p:=PersonaFin;

Close(arch); **Close**(espLibre) **end;**

end; {With}

end; {Proceso}