PRÁCTICA DE METODOLOGÍA

Librería de cadenas de caracteres

Índice

Apartado Páginas

Descripción del problema

Tenemos que construir una librería o Unit (según el lenguaje Pascal) que trabaje con variables de tipo *string*. Nuestra librería está compuesta de varios procedimientos y funciones, que son las siguientes:

- **Función mi_Concat(string1,string2): string.**
- **Función mi_Copy(string, índice, ncar): string.**
- **Procedimiento mi_Delete(string,índice,ncar)**
- **Procedimiento mi_Insert(string,string,index)**
- **Función mi_Length(string):integer**
- **Función mi_Pos(string,string): integer**
- **Procedimiento mi_Str(numero,string)**
- **Procedimiento mi_Val(string,numero,coderr)**
- **Procedimiento Tkstr(string,ncar)**
- **Procedimiento Tkint(numero,ndigitos)**
- **Función StrUpcase(string)**

Comentario sobre la solución

Para solucionar el problema planteado, he creado una unidad especial llamada Mi_Unit que trabaja con variables de tipo string.

Dicha unidad está compuesta por una serie de funciones y procedimientos, también creados por mí que nos permiten realizar una serie de operaciones sobre los strings dentro de cualquier programa de Pascal, sin necesidad de definir dichas funciones cada vez que las necesitemos en un futuro programa de Pascal.

Pseudocódigo

- **Función mi_concat: string;**

Variables

i: integer;

aux: string;

Parámetros

string1, string2: string;

Para i:= 1 hasta (ord(string1{0}) + ord (string2{0})) hacer

si i<= ord(string1{0}) entonces

aux{i}:= string1{i}

sino

aux{i}:= string2{i−ord(string1{0})}

finsi;

finpara;

aux{0}:= chr(ord(string1{0})+ord(string2{0}));

mi_concat:= aux;

- **Función mi_copy: string;**

Variables

i: integer;

aux: string,

Parámetros

string1: string**;**

indice: integer;

ncar: integer;

Para i:= 1 hasta ncar hacer

aux{1}:= string1{indice+i−1}

finpara

Si indice+ncar <= ord (string1{0}) entonces

aux{0}:= chr(ncar)

sino

aux{0}:= chr(ord(string1{0})−indice+1)

finsi

mi_copy:= aux

- **Procedimiento mi_delete;**

Variables

i: integer;

Parámetros

string1: string;

indice: integer;

ncar: integer;

Para i:= ncar+indice hasta length(string1) hacer

string1{i−ncar}:= string1{i}

finpara

string1{0}:= chr(ord(string1{0})−ncar)

- **Procedimiento mi_insert;**

Variables

i,j: integer;

aux: string;

Parámetros

string1: string;

string2: string;

index: integer;

j:= 0

Para i:= index + ord(string1{0}) to ord(string2{0}) hacer

j:= j+1

aux{j}:= string2{i}

finpara

Para i:= 1 to ord(string1{0}) hacer

string2{index+i−1}:= string1{i}

finpara

Para i:= 1 to ord(aux{0}) hacer

string2{index+ord(string1{0})+i−1}:= aux{i}

finpara

string2{0}:= chr(index+ord(string1{0})−1)

- **Función mi_length: integer;**

Parámetros

string1: string;

mi_length:= ord(string1{0})

- **Función mi_pos: integer;**

Variables

encontrado: boolean;

pcad,pscad: integer;

Parámetros

string1,string2: string;

Pcad:= 1

Pscad:= 1

Encontrado:= false;

Mientras (not encontrado) y (pcad<= ord(string2{0})) hacer

Si string2{pcad}<> string1{pscad} entonces

pcad:= pcad+1

pscad:= 1

sino

si pscad = ord(string1{0}) entonces

encontrado:= true

sino

pcad:= pcad+1;

pscad:= pscad+1

finsi

finsi

finmientras

Mi_pos:= (pcad−pscad)+1

- **Procedimiento mi_str;**

Variables

aux, a, pos, i, cont: integer;

parado: boolean;

Parámetros

numero: integer,

string1: string;

I:= 0

Cont:= 10000

Parado:= false

Mientras (i<=5) and (not parado) hacer

a:= numero div cont

Si a<> 0 entonces

parado:= true

pos:= 5−i

finsi

cont:= cont div 10

i:= i+1

finmientras

Cont:= 1

Si pos>1 entonces

Para i:= 1 hasta pos−1 hacer

cont:= cont*10

finpara

finsi

Aux:= numero

Para i:= 1 hasta pos hacer

string1{i}:= chr((aux div cont) + 48)

aux:= aux − ((ord(string1{i})−48)*cont)

cont:= cont div 10

finpara

string1{0}:= chr(pos)

Procedimiento mi_val;

Variables

i: integer;

error: boolean;

Parámetros

string1: string;

numero: integer;

coderr: integer;

Cd:= 0

Numero:= 0

I:= 1

Error:= false

Mientras (i<= ord(string1{0})) and (not error) hacer

Si (ord(string1{i})>= 48) y (ord(string1{i})<=57) entonces

numero:= numero*10+(ord(string1{i})−48)

sino

si (string1{1}<>`−') entonces

numero:= 0

coderr:= i

error:= true

finsi

finsi

i:= i+1

finmientras

Si (string1{1}=`−') entonces

numero:= −numero

- **Procedimiento Tkstr;**

Variables

i: integer;

c: char;

Parámetros

string1: string;

cifras: integer;

Repetir

c:= readkey

hasta (ord(c) = 45) o ((ord(c)>= 65) y (ord(c)<=90)) o ((ord(c)>= 97) y (ord(c)<=122)) o ((ord(c)>= 128) y (ord(c)<=165)) o (ord(c)=13) o (ord(c)=8)

I:= 1

String1:= `'

Mientras (c<>chr(13)) hacer

Si ord(c)= 8 entonces

Si i>1 entonces

i:= i−1

escribir(chr(8))

escribir(chr(32))

escribir(8)

finsi

sino

Si i<= cifras entonces

string1{i}:= c

escribir(c)

i:= i+1

sino

sound(6000)

delay(100)

nosound

finsi

finsi

Repetir

c:= readkey

hasta (ord(c) = 45) o ((ord(c)>= 65) y (ord(c)<=90)) o ((ord(c)>= 97) y (ord(c)<=122)) o ((ord(c)>= 128) y (ord(c)<=165)) o (ord(c)=13) o (ord(c)=8)

finmienras

string1{0}:= chr{i−1}

- **Procedimiento Tkint;**

Variables

ca: string;

e,i: integer;

c: char;

Parámetros

numero: integer;

ndigitos: integer;

Repetir

c:= readkey

hasta (ord(c) = 45) o ((ord(c)>=48) y (ord(c)<= 57)) o (ord(c)= 13) o (ord(c)= 8)

I:= 1

Ca:= `'

Mientras (c<> chr(13)) hacer

Si c=`−' entonces

Si i= 1 entonces

ca{i}:= c

escribir(c)

i:= i+1

finsi

```
sino
Si ord(c) =8 entonces
Si i>1 entonces
i:= i−1
escribir (chr(8))
escribir (chr(32))
escribir (chr(8))
finsi
sino
Si i<= cifras entonces
ca{i}:= c
escribir(c)
i:= i+1
sino
sound(5000)
delay(100)
nosound
finsi
finsi
finsi
Repetir
c:= readkey
hasta (ord(c) = 45) o ((ord(c)>=48) y (ord(c)<= 57)) o (ord(c)= 13) o (ord(c)= 8)
finmientras
Ca{0}:= chr(i−1)
Mi_val(ca,numer,e)
```

- **Función StrUpcase: string;**

Variables

i: integer;

aux: string;

Parámetros

string1: string;

Para i:= 1 hasta ord(string1{0}) hacer

aux{i}:= upcase(string1{i})

finpara

Ord(aux{0}):= ord(string1{0});

String1:= aux

Código comentado

Unit mi_unit;

Interface

Uses crt;

- **Function mi_concat(string1,string2: string): string;**

{Nombre: mi_concat

Descripción: nos devolverá en una string la concatenación de todas las

cadenas

Parámetros de entrada:

− string1,string2: las dos cadenas a concatenar

Variables locales:

− i : el contador de la posición de la frase final

− aux: en esta frase guardaremos la concatenación de las anteriores}

- **Function mi_Copy(string1: string;indice:integer;ncar: integer): string;**

{Nombre: mi_Copy

Descripción: muestra de la cadena, a partir de inicio, tantos caracteres como

indique l

Parámetros de entrada:

− string1: es la frase principal de la que vamos a mostrar los caracteres.

− índice : es la posición a partir de la cual vamos a mostrarlos

− ncar : es el número de caracteres a mostrar

Variables locales:

− i : es el contador de los caracteres que vamos copiando

− aux: es la frase en la que nos apoyamos para copiar las letras}

- **Procedure mi_delete(string1: string;indice: integer;ncar: integer);**

{Nombre: MI_DELETE.

Descripción: Borra l caracteres en una cadena a partir de la posición ini,

introducida por teclado.

Parámetros de entrada:

indice: Indica la posición a partir de la cual borrar.

ncar: Indica el número de caracteres a borrar.

Parámetros de salida:

string1: Es la cadena principal sobre la cual vamos a borrar.

Variables locales:

i: Indice.}

- **Procedure mi_insert(string1,string2: string;index: integer);**

{Nombre: mi_insert

Descripción: insertamos la subcadena 'nuevo' en 'cadena' en la posición

'inicio'

Parámetros de entrada:

− string1 : subcadena que insertamos

− Index: posición de la cadena principal en la cual insertamos

Parámetros de salida:

− string2: frase en la cual insertamos.

Variables locales:

− j : posición de la frase auxiliar

− i : posición de la cadena principal

− aux: la utilizamos para guardar la parte siguiente a donde insertamos}

- **Function mi_length(string1: string): integer;**

{Nombre: mi_length

Descripción: nos da el tamaño de la frase

Parámetros de entrada:

− string1: es la frase de la que hayamos su tamaño.}

- **Function mi_pos(string1,string2: string): integer;**

{Nombre: mi_pos

Descripción: busca la posición en la que se encuentra la subcadena dentro

de la cadena principal

Parámetros de entrada:

− string1: es la cadena a buscar dentro de la principal

− string2: es la frase principal

Variables locales:

− pcad : es el contador de la posición de la cadena principal que esta

mos comparando con la primera letra de la subcadena.

− pscad : es el contador de la posición de la subcadena que estamos

comparando con la principal.

− encontrado: nos confirma que hemos encontrado la posición de la subcadena}

- **Procedure mi_str(numero: integer;string1: string);**

{Nombre: MI_STR.

Descripción: Convierte una variable de tipo integer en una de tipo string.

Parámetros de entrada:

numero: Número de tipo integer que vamos a transformar.

Parámetros de salida:

string1: Variable de tipo string en la que almacenaremos el número.

Variables locales:

aux: Variable de tipo integer que utilizamos para guardar el número

a transformar y poder operar con él.

a: Variable que usamos para apoyarnos al hallar el número de dígitos

de subnum.

cont: Contador que usamos para ir haciendo sucesivas divisiones por

10, 100, etc.. y hallar las unidades, decenas, etc...

i: Indice.

parado: Boolean que nos dice en el primer while cuando hemos

hallado el número de cifras de subnum y que hay que parar.}

- **Procedure mi_val(string1: string; numero: integer; coderr: integer);**

{Nombre: mi_val

Descripción: convierte una cadena de caracteres numéricos en un entero

Parámetros de entrada:

− string1: es la cadena que debemos convertir en número

Parámetros de salida:

− numero: es el número que hemos hallado

− cd : nos indica la posición del error

Variables locales:

− i : contador de la posición del string de caracteres

− error: nos indica si en el string hay algún carácter no numérico}

- **Procedure tkstr (var string1: string; cifras:integer);**

{Nombre: tkstr

Descripción: devolverá una cadena en la variable pasada como primer parámetro,

que como máximo deberá tener *cifras* caracteres.

Parámetros de entrada:

− cifras: nos indica la longitud máxima del integer que vamos a insertar

Parámetros de salida:

− string1: es la frase que vamos creando.

Variables locales:

− i : es la longitud del string que vamos creando

− c : es el carácter que vamos insertando}

- **Procedure tkint(cifras:integer;var numer:integer);**

{Nombre: tkint

Descripción:

Parámetros de entrada:

− cifras: nos indica la longitud máxima del integer que vamos a insertar

Parámetros de salida:

− numer: es el integer creado

Variables locales:

− ca: es el string en el que vamos creando

− i : es la longitud del string que vamos creando

− c : es el carácter que vamos insertando

− e: lo utilizamos para que no halla problemas con el procedure mi_val}

- **Function strupcase(string1: string): string;**

{Nombre: strupcase

Descripción: devuelve la cadena pasada como parámetro en mayúsculas.

Parámetros de entrada:

− string1: es la frase que vamos a convertir en mayúsculas

Variables locales:

− i: posición de la palabra

− aux: frase auxiliar en la que nos apoyamos}

Implementation

- **Function mi_concat;**

Var

i: integer;

aux: string;

begin

For i:= 1 to (ord(string1[0]) + ord(string2[0])) do

if i<= ord(string1[0]) then

aux[i]:= string1[i]

else

aux[i]:= string2[i−ord(string1[0])];

aux[0]:= chr(ord(string1[0])+ord(string2[0]));

mi_concat:= aux

end;

- **Function mi_copy;**

Var

i: integer;

aux: string;

begin

```
For i:= 1 to ncar do
aux[i]:= string1[indice+i−1];
if indice+ncar<=ord(string1[0]) then
aux[0]:= chr(ncar)
else
aux[0]:= chr(ord(string1[0])−indice+1);
mi_copy:= aux;
end;
```

- **Procedure mi_delete;**

```
var
i:integer;
{− Consiste simplemente en desplazar l posiciones hacia la izquierda los
caracteres que queden a la derecha de l+ini.}
begin
for i:=ncar+indice to length(string1) do
string1[i−ncar]:=string1[i];
string1[0]:=chr(ord(string1[0])−ncar);
end;
```

- **Procedure mi_insert;**

```
var
i,j: integer;
aux: string;
begin
j:= 0;
For i:= index+ ord(string1[0]) to ord(string2[0]) do
begin
```

```
j:= j+1;
aux[j]:= string2[i]
end;
For i:= 1 to ord(string1[0]) do
string2[index+i−1]:= string1[i];
For i:= 1 to ord(aux[0]) do
string2[index+ord(string1[0])+i−1]:= aux[i];
string2[0]:= chr(index+ord(string1[0])−1);
end;
```

- **Function mi_length;**

```
begin
mi_length:= ord(string1[0]);
end;
```

- **Function mi_pos;**

```
Var
encontrado:boolean;
pcad,pscad:integer;
Begin
pcad:=1;
pscad:=1;
encontrado:=false;
while (not encontrado) and (pcad<=ord(string2[0])) do
begin
if string2[pcad]<>string1[pscad] then
begin
pcad:=pcad+1;
```

```
pscad:=1
end
else
if pscad=ord(string1[0]) then
encontrado:=true
else
begin
pcad:=pcad+1;
pscad:=pscad+1
end
end;
mi_pos:=(pcad−pscad)+1
End;
```

- **Procedure mi_str;**

```
var
aux,a,pos,i,cont:integer;
parado:boolean;
{− Primero hallamos el número de digitos que tiene subnum utilizando un
while y haciendo sucesivas divisiones entre 10.000, 1.000, etc.. hasta
que sea distinto de 0. Entonces 5−i nos indicará el número de cifras.
− Luego inicializamos el contador dependiendo del número de cifras de subnum.
− Despues con un for desde i=1 hasta el número de cifras de subnum, le
asignamos a subc[i] el caracter que corresponde al auxiliar dividido
entre el contador, y le restamos al auxiliar el contador multiplicado
por el resultado de la primera división, quedando así un número con una
cifra menos por la izquierda.
```

− Luego se divide el contador entre 10 y se procede sucesivamente hasta terminar por las unidades.

− Al final se le asigna a la cadena subc el número de posiciones que le corresponda.}

```
begin
i:=0;
cont:=10000;
parado:=false;
while (i<=5) and (not parado) do
begin
a:=numero div cont;
if a<>0 then
begin
parado:=true;
pos:=5−i;
end;
cont:=cont div 10;
i:=i+1;
end;
cont:=1;
if pos>1 then
for i:=1 to pos−1 do
cont:=cont*10;
aux:=numero;
for i:=1 to pos do
begin
```

```
string1[i]:=chr((aux div cont)+48);
aux:=aux−((ord(string1[i])−48)*cont);
cont:=cont div 10;
end;
string1[0]:=chr(pos);
end;
```

- **Procedure mi_val;**

```
var
i: integer;
error: boolean;
begin
cd:= 0;
numero:= 0;
i:= 1;
error:= false;
While (i<= ord(string1[0])) and (not error) do
begin
if (ord(string1[i])>=48) and (ord(string1[i])<=57) then
numero:= numero*10+(ord(string1[i])−48)
else
begin
if (string1[1] <> '−') then
begin
numero:= 0;
coderr:= i;
error:= true
```

```
end

end;

i:= i+1

end;

if (string1[1]= '−') then

numero:= −numero

end;
```

- **Procedure tkstr;**

```
var

i:integer;

c:char;

begin

repeat

c:=readkey;

Until (ord (c) = 45) or ((ord(c)>=65) and (ord(c)<=90)) or ((ord(C)>=97) and (ord(c)<= 122))

or ((ord(c)>=128) and (ord(c)<= 165)) or (ord(c)=13) or (ord(c)=8);

i:=1;

string1:='';

while (c<>chr(13)) do

begin

if ord(c)=8 then

begin

if i>1 then

begin

i:=i−1;

write(chr(8));
```

```
write(chr(32));
write(chr(8))
end
end
else
begin
If i <=cifras then
begin
string1[i]:=c;
write(c);
i:=i+1
end
else
begin
sound (6000);
delay (100);
nosound;
end;
end;
repeat
c:=readkey;
Until (ord (c) = 45) or ((ord(c)>=65) and (ord(c)<=90)) or ((ord(C)>=97) and (ord(c)<= 122))
or ((ord(c)>=128) and (ord(c)<= 165)) or (ord(c)=13) or (ord(c)=8);
end;
string1[0]:=chr(i−1);
end;
```

- **Procedure tkint;**

```
var
ca:string;
e,i:integer;
c:char;
begin
repeat
c:=readkey;
Until (ord (c) = 45) or ((ord(c)>=48) and (ord(c)<=57)) or (ord(c)=13) or (ord(c)=8);
i:=1;
ca:='';
while (c<>chr(13)) do
begin
if c='−' then
begin
if i=1 then
begin
ca[i]:=c;
write(c);
i:=i+1
end
end
else
if ord(c)=8 then
begin
if i>1 then
```

```
begin
i:=i−1;
write(chr(8));
write(chr(32));
write(chr(8))
end
end
else
begin
If i <=cifras then
begin
ca[i]:=c;
write(c);
i:=i+1
end
else
begin
sound (6000);
delay (100);
nosound;
end;
End;
Repeat
c:=readkey;
Until (ord (c) = 45) or ((ord(c)>=48) and (ord(c)<=57)) or (ord(c)=13) or (ord(c)=8);
end;
```

```
ca[0]:=chr(i−1);

mi_val(ca,numer,e);

end;
```

- **Function StrUpcase;**

```
Var

i: integer;

aux: string;

Begin

For i:= 1 to ord(string1{0}) do

aux[i}:= upcase(string1{i});

ord(aux{0}):= ord(string1{0});

string1:= aux

end;

End.
```

Explicación de la implantación de la librería

Tras escribir el programa en Pascal, debemos guardar el archivo con el mismo nombre físico que el que le hemos dado al definir la unidad. El lugar en el que guardaremos el archivo será el mismo directorio en el cual se encuentran las unidades (en nuestro caso, *c:\compi\pascal\units*).

Por último y para conseguir que Pascal convierta el archivo en una librería debemos ir al menú *Compiler* y hacer click con el mouse donde pone *Destination Memory* pasando a convertirlo en *Destination Disk*. Una vez hecho esto debemos compilar el archivo, pulsando F9. De esta manera, Pascal, al leer en la cabecera del programa la palabra *Unit* en lugar de *Program*, convierte nuestro archivo en una unidad más que pueden utilizar los usuarios del programa.

Si queremos utilizar nuestra librería para un futuro programa de Pascal, en dicho programa, a la hora de definir las librerías a utilizar deberemos incluir *Uses mi_unit*. A partir de ese momento, trabajaremos con las funciones y procedimientos de dicha librería como si fuesen propios de Pascal.