UTN Universidad Tecnológica Nacional
Facultad Regional La Plata

Ingeniería en Sistemas de Información

Algoritmos y Estructuras de Datos

Proyecto

Pascal // Fascículo 5

Contenido:

Tipos de datos definidos por el usuario
1. Arreglos
2. Registros
3. Arreglo de Registros

Jefe de Cátedra:
- Lic. Carlos A. López

Jefe de Trabajos Prácticos:
- Ing. Gustavo Cerveri

Tutores de Proyecto:
- Marcelo Cardós - marcelo@ayed.com.ar
- Eduardo M. Puricelli - edus@ayed.com.ar
- Sergio R. Vaquero - sergio@ayed.com.ar

Bibliografía:
Programación en Turbo/Borland Pascal 7
3ra edición - Osborne / McGraw-Hill
Luis Joyanes Aguilar

http://www.frlp.utn.edu.ar/materias/algoritmos/

http://www.ayed.com.ar/

Tipos de datos definidos por el usuario

Al definirnos como “programadores” estamos declarándonos creadores de software y, como tal, tenemos necesidades de herramientas que nos sirvan para este trabajo que realizamos. En los fascículos anteriores se han visto tipos de datos simples. Estos tipos de datos son útiles para muchas situaciones, pero a veces esto nos resulta poco y nos interesa tener tipos de datos que agrupen tipos de datos simples.

He aquí algunos tipos de datos definidos por el usuario, siendo quizás los más comunes, los arreglos y los registros, los cuales se pasarán a explicar a continuación.

1. Arreglos

Comúnmente conocidos como “arrays” en la programación con Pascal, estos tipos de datos son grupos de variables de un mismo tipo de datos, las cuales son accedidas por un índice. La restricción más importante es, justamente, que puede almacenarse un único tipo de datos dentro de él. Otra de las restricciones es que deben declararse de un tamaño prefijado en el código, de manera que no es posible ampliar su tamaño en tiempo de ejecución. Pero más allá de estos detalles, es una buena estructura para almacenar importantes cantidades de datos y tener acceso a ellos de una manera ordenada y sencilla. Además, se pueden declarar arreglos de varias dimensiones, de manera que tenemos más flexibilidad al momento de planear nuestra solución utilizándolos.

Es importante tener en cuenta que debemos ser cuidados al acceder a las posiciones de un arreglo, ya que acceder a posiciones que no existen nos generará un error. En general, las posiciones se manejan con valores enteros, aunque Pascal también nos permite hacerlo con letras y, aunque no es común, quizás según sea el caso, nos podría convenir recurrir a esta propiedad. Otro detalle, es que pueden declararse las dimensiones de un vector, ya sea directamente por números, o por constantes, a modo de poder rápidamente quizás modificar un código para que la cantidad de valores guardados en sus estructuras sea distinta. Esto es un detalle que da flexibilidad a nuestro programa.

Seguidamente pasaremos a declarar un arreglo de una dimensión (vector) y un arreglo de dos dimensiones (comúnmente llamado “matriz de n x m”, siendo n: filas y m: columnas) y accederemos para leer y escribir distintas posiciones de los mismos, mediante un ejemplo. Se verá en el ejemplo la posibilidad de definir con constantes sus dimensiones.

```
PROGRAM Arreglos;
Uses Crt;

CONST
  Max = 5;

TYPE
  Vector = array[1..Max] of integer;
  Matriz = array[1..Max,1..2] of char;

VAR
  V1,V2: Vector;
  M: Matriz;

PROCEDURE CargaV (var V: Vector);
VAR
  i: integer;
BEGIN
  for i:= 1 to Max do
  begin
    write('Ingrese valor ', i, ': ');
    readln(V[i]);
  end;
END;


PROCEDURE CargaM (var M: Matriz);
VAR
  i,j: integer;
BEGIN
  for i:= 1 to Max do
  begin
    writeln('Cargando fila ', i);
    for j:= 1 to 2 do
    begin
      write('Ingrese valor ', j);
      readln(M[i,j]);
    end;
  end;
END;


BEGIN { principal }
  crlscr;

  writeln('Se cargará el vector 1');
  CargaV(V1);

  writeln('Se cargará el vector 2');
  CargaV(V2);

  writeln('Se cargará la matriz);
  CargaM(M);


END.
```

```
PROGRAMA Arreglos


CONSTANTES
  Max = 5

TIPOS ESTRUCTURADOS
  Vector = arreglo[Max]: entero
  Matriz = arreglo[Max,2]: caracter 1

VARIABLES
  V1,V2: Vector
  M: Matriz

PROCEDIMIENTO CargaV (Ref V: Vector)
VARIABLES
  i: entero
HACER
  Repetir Para i:= 1,Max

    Imprimir: 'Ingrese valor ', i, ': '
    Leer: V[i]
  Fin Repetir Para
FIN HACER
FIN PROCEDIMIENTO

PROCEDIMIENTO CargaM(Ref M: Matriz)
VARIABLES
  i,j: entero;
HACER
  Repetir Para i:= 1,Max

    Imprimir: 'Cargando fila ', i
    Repetir Para j:= 1,2

      Imprimir: 'Ingrese valor ', j
      Leer: M[i,j]
    Fin Repetir Para
  Fin Repetir Para
FIN HACER
FIN PROCEDIMIENTO

HACER /* principal */


  Imprimir: 'Se cargará el vector 1'
  CargaV(V1)

  Imprimir: 'Se cargará el vector 2'
  CargaV(V2)

  Imprimir: 'Se cargará la matriz'
  CargaM(M)


FIN HACER
FIN PROGRAMA
```

Se ha incluido en el ejemplo de Pascal la función de limpiar pantalla y para ello necesitó agregarse al programa la línea `Uses Crt`. Estos conceptos vienen del fascículo anterior, por lo cual si se presenta alguna duda, no estaría mal dar una mirada al fascículo cuatro.

2. Registros

Los conocidos “records” de Pascal son estructuras que pueden almacenar un conjunto de variables de igual o distinto tipo, lo cual es interesante cuando queremos manejar algunos datos como un todo o unidad.

Un detalle interesante es, a pesar de no poder ser utilizado en pseudocódigo, el hecho de copiar variables tipo registro entre sí, es decir que si tengo las variables a y b del mismo tipo registro que seguramente he definido en la parte superior de mi programa, podría utilizar sin problemas la siguiente manera de asignar `a:= b;` para copiar el valor de todos los datos que contienen en su interior.

Se pasará rápidamente a declarar y utilizar básicamente un registro.

```
PROGRAM Registros;

TYPE
  Persona = record
    Nombre: string;
    Edad: integer;
  end;

VAR
  Amigo1,Amigo2: Persona;

BEGIN
  write(‘Nombre del amigo 1: ’);
  readln(Amigo1.Nombre);
  write(‘Edad del amigo 1: ’);
  readln(Amigo1.Edad);

  write(‘Nombre del amigo 2: ’);
  readln(Amigo2.Nombre);
  write(‘Edad del amigo 2: ’);
  readln(Amigo2.Edad);
END.
```

```
PROGRAMA Registros

TIPOS ESTRUCTURADOS
  Persona = registro
    Nombre: caracter
    Edad: entero


VARIABLES
  Amigo1,Amigo2: Persona

HACER
  Imprimir: ‘Nombre del amigo 1’
  Leer: Amigo1.Nombre
  Imprimir: ‘Edad del amigo 1’
  Leer: Amigo1.Edad

  Imprimir: ‘Nombre del amigo 2’
  Leer: Amigo2.Nombre
  Imprimir: ‘Edad del amigo 2’
  Leer: Amigo2.Edad
FIN HACER
FIN PROGRAMA
```

3. Arreglo de Registros

Surge algo muy interesante cuando se combinan los conceptos de arreglo y registro. Esta solución es útil cuando necesitamos guardar varios datos de distinto tipo y a su vez, sucede que estos datos se repiten. Es decir, puede suceder, por ejemplo, que tengamos que guardar datos de personas, a lo que podríamos proceder definiendo un tipo persona y luego un arreglo que contenga datos de tipo persona, el cual fue definido antes. Cabe destacar que esta no es la única solución a este caso, sino que podríamos también tener varios arreglos, agrupando en cada arreglo los datos de cada persona que tengan el mismo tipo. Luego, cuidando de contener en las mismas posiciones de todos los arreglos los datos que describen a cada persona, se solucionaría de manera alternativa este mismo problema. Al final, la utilización de una manera u otra queda a criterio del programador.

Pasemos a ver un ejemplo, en el cual simplemente se cargará un vector de registros persona.

```
PROGRAM ArregloRegistros;

TYPE
  Persona = record
    Nombre: string;
    Edad: integer;
  end;

  Personas = array[1..10] of Persona;

VAR
  Amigos: Personas;
  i: integer;

BEGIN
  for i:= 1 to 10 do begin
    writeln('Datos del amigo ', i);
    write('Nombre: ');
    readln(Amigos[i].Nombre);
    write('Edad: ');
    readln(Amigos[i].Edad);
  end;
END.
```

```
PROGRAMA Registros

TIPOS ESTRUCTURADOS
  Persona = registro
    Nombre: caracter
    Edad: entero


  Personas = arreglo[10]: Persona

VARIABLES
  Amigos: Personas
  i: entero

HACER
  Repetir Para i:= 1,10
    Imprimir: 'Datos del amigo ', i
    Imprimir: 'Nombre: '
    Leer: Amigos[i].Nombre
    Imprimir: 'Edad: '
    Leer: Amigos[i].Edad
  Fin Repetir Para
FIN HACER
FIN PROGRAMA
```