UTN Universidad Tecnológica Nacional
Facultad Regional La Plata

Ingeniería en Sistemas de Información

Algoritmos y Estructuras de Datos

Proyecto

Pascal // Fascículo 4

Contenido:

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/

1. Modularización: Declaración de Procedimientos y Funciones.

1.1. Procedimientos.

Los dos tipos de subprogramas o módulos que permite Pascal, son procedimientos y funciones. Ambas, son similares a las existentes en pseudocódigo.

```
PROGRAM procedimiento;
VAR
  lado,sup,perim: integer;

PROCEDURE x(l: integer; var s,p: integer);
begin
  s:= l*l;
  p:= l*4;
end;

BEGIN
  writeln('Ingrese lado de un cuadrado');
  readln(lado);
  x(lado,sup,perim);
  writeln('Superficie: ',sup);
  writeln('Perímetro ',perim);
END.
```

```
PROGRAMA procedimiento
VARIABLES
  lado,sup,perim: entero

PROCEDIMIENTO x(l:entero; ref s,p:entero)
hacer
  s:= l*l
  p:= l*4
fin hacer
FIN PROCEDIMIENTO

HACER
  imprimir:'Ingrese lado de un cuadrado'
  leer(lado)
  x(lado,sup,perim)
  imprimir:'Superficie: ',sup
  imprimir:'Perímetro ',perim
FIN HACER
FIN PROGRAMA
```

Tenga en cuenta siempre que:

* Los parámetros deber ser del mismo tipo.

* Los parámetros variables (var) deben ser siempre una variable, es decir, que son pasados por referencia.

* La correspondencia entre los parámetros dados en un procedimiento y una llamada al mismo, está dada por su orden de secuencia.

* Los nombres de los parámetros dados en un procedimiento y la llamada serán preferentemente diferentes, especialmente cuando se trata de parámetros pasados por copia.

* Siempre que los parámetros de un procedimiento pueden ser modificados en su contenido, éstos deberán ser parámetros variables, es decir, pasados por referencia.

1.2. Funciones.

Una función es un subprograma que **siempre** devuelve un resultado. Al igual que los procedimientos, las funciones disponen de parámetros, pero **ninguno** de ellos deberá ser pasado por referencia. Veamos un pequeño ejemplo:

```
PROGRAM funcion;
VAR
  lado: integer;

FUNCTION x(l: integer): integer;
begin
  x:= l*l;
end;

BEGIN
  writeln('Ingrese lado de un cuadrado');
  readln(lado);
  writeln('Superficie: ',x(lado));
END.
```

```
PROGRAMA funcion
VARIABLE
  lado: entero

FUNCION x(l: entero)
hacer
  x:= l*l
fin hacer
FIN FUNCION

HACER
  imprimir:'Ingrese lado de un cuadrado'
  leer(lado)
  imprimir:'Superficie: ',x(lado)
FIN HACER
FIN PROGRAMA
```

2. UNITS (unidades) en Turbo Pascal 7.0 de Borland.

2.1. Introducción

Una unidad es un conjunto de procedimientos y funciones incorporados al lenguaje de programación. Para poder acceder a los procedimientos o funciones incluidos en una unidad o librería, sólo debe llamársela al iniciar el programa.

En el caso de Turbo Pascal 7.0 de Borland, para inicializar una librería debe agregarse la siguiente línea luego de la sentencia PROGRAM:

USES nombre_libreria; o **USES nombre_libreria_1, nombre_libreria_2;**

*donde **nombre_libreria** puede ser CRT, DOS, GRAPH, etc..*

2.2. Librería CRT

La librería CRT nos permite utilizar algunas funciones y procedimientos dedicados a mejorar la presentación en modo texto de la pantalla. En el modo texto la pantalla consta de 25 filas por 80 columnas. Cada punto se identifica con una pareja de números, que indica la fila y la columna.

Veamos algunos ejemplos de funciones y procedimientos:

- **clrscr** procedimiento sin parámetros, que simplemente limpia la pantalla.
- **gotoxy(x,y: integer)** ubica al cursor en la posición de coordenadas "x" e "y" de la pantalla, tal que "x" representa el número de columna e "y" el número de fila.
- **textcolor(a: integer)** especifica el color del texto que se imprimirá a partir de esta instrucción. Los colores son entre 0 y 15, repitiéndose los mismos colores a partir del 16, pero en modalidad intermitente.
- **readkey** función que espera al usuario a que pulse una tecla, y difiere del "read" en que no necesita que se pulse "Enter" para seguir a la siguiente instrucción. Esta función también es útil cuando necesitamos leer un carácter desde teclado, como los clásicos "s" o "n" para capturar las decisiones del usuario.
- **delay(tiempo: integer)** realiza una pausa por una cantidad de tiempo en milisegundos, determinada por el parámetro. En realidad, no siempre son verdaderos milisegundos, ya que es muy relativo al procesador de la computadora.

Los dos últimos, son ampliamente utilizados para generar pausas y visualizar resultados, ya que al compilar y correr un programa en Pascal y terminar de ejecutar la última instrucción, éste regresa rápidamente al código sin dejarnos visualizar los resultados.

También se pueden escribir caracteres del código ASCII y así poder hacer líneas y gráficas en modo texto. Probemos lo siguiente:

```
PROGRAM grafico;
USES crt;
BEGIN
  textbackground(1); {fondo azul}
  clrscr;
  textcolor(15); {letras blancas}
  gotoxy(27,12);
  write('Pascal, también es lindo!');
  readkey;
END.
```

2.3. Librería GRAPH

Contiene más de 50 procedimientos y más de 20 funciones, muchas constantes, variables y tipos de datos. Incluye controladores para los adaptadores de gráficos más importantes y fuentes para dibujar una gran cantidad de estilos de caracteres. La pantalla de gráficos está formada por píxels ordenados en líneas horizontales y verticales. Esto es válido para todos los modos gráficos de la computadora; la diferencia esta en el tamaño del píxel.

Para utilizar esta unidad debemos, además de agregar la correspondiente línea "USES GRAPH;", inicializar los controles gráficos (especificando el tipo de adaptador y el modo gráfico) y después llamar a **InitGraph** que es el que preparará el Hardware para los gráficos, y se indicará dónde debe encontrar el archivo **.BGI** que es el que contiene el adaptador de gráficos.

Después de llamar a **InitGraph**, el programa llama a **GraphResult**, una función que devuelve un código de estado. Si el código no es igual a **grOK**, se sabe que se ha producido un error (grOK es una constante definida en la unidad GRAPH y tiene el valor cero).

A continuación viene el código del programa, es aquí donde pondremos las instrucciones para realizar los gráficos que se desean y finalizaremos con **CloseGraph** que devuelve el Hardware al modo en que se encontraba antes de entrar en modo gráfico. Ejemplo:

```
Program Ejemplo;
Uses CRT, GRAPH;
Var
 DrvGraph,ModoGraph: Integer;

Begin
  DrvGraph:= Detect;
  InitGraph(DrvGraph,ModoGraph, 'c:\pascal\bgi');
  SetLineStyle(dottedLn,0,ThickWidth);
  Rectangle(280,150,40,30);
  Delay (5000);
  CloseGraph;
End.
```

Recuerde que lo marcado en negrita es el camino (path) donde se encuentra GRAPH.TPU (que es el archivo que contiene la librería). Para más información sobre esta y otras librerías, así como también las clásicas instrucciones de Pascal, no dude en consultar el Help de Pascal (Shift+F1).