****

***Laboratorio #5***

***Subprogramas***

**Objetivos:**

* Conocer el funcionamiento de las funciones y procedimientos.
* Reconocer las semejanzas y diferencias de las funciones y procedimientos.
* Implementar diversos subprogramas a través de las funciones y procedimientos.

**Pre-laboratorio**

**Lea cada uno de los elementos descritos en esta sección antes de la práctica**

Las técnicas como la abstracción de datos, la abstracción procedimental o la estrategia del tipo divide y vencerás utilizadas en el diseño descendente permiten dividir un problema en subproblemas más simples. Una vez dividido cada uno de estos subproblemas pueden ser tratados de forma independiente y cuando el subproblema es lo suficientemente sencillo puede ser implementado. El algoritmo completo estará entonces formado por los diferentes algoritmos en los que se ha dividido el problema original. Gracias a la descomposición modular del problema, el programador puede concentrarse en cada módulo de manera independiente. De esta forma tanto la codificación como la depuración o búsqueda de errores son más simples. La siguiente práctica trata acerca de la implementación de subprogramas en Pascal.

**Procedimientos y funciones**

Pascal permite la implementación de dos tipos diferentes de subprogramas, los procedimientos (PROCEDURE) y las funciones (FUNCTION). El lenguaje pascal posee procedimientos predefinidos por ejemplo: leer datos de la entrada estándar (read/readln) o aquellos que permiten mostrar información por la salida estándar (write/writeln). Desde este punto de vista cualquier subprograma en Pascal puede clasificarse en función de quién lo ha programado; por lo tanto puede hablarse de subprogramas:

**Predefinidos por el lenguaje:** como los procedimientos READLN Y WRITELN, o las funciones SQR, ROUND, MOD, etc. Este tipo de subprogramas son proporcionados por Pascal y puede ser directamente utilizados por el programador, sin necesidad de conocer cómo se ha implementado el algoritmo concreto que resuelve el problema. En este caso y en función de la versión de Pascal utilizada, las funciones y procedimientos predefinidos pueden variar y deben ser estudiados por el programador.

**Definidos por el usuario:** No obstante, la utilización de subprogramas predefinidos por el lenguaje, aunque útil, no es realmente interesante sino es posible codificar nuevos algoritmos que permitan resolver nuevos problemas. Los subprogramas definidos por los programadores son en última instancia los realmente encargados de resolver un problema concreto.

**Cuando se declara un subprograma en Pascal, procedimiento o función, se define un identificador que permite posteriormente invocarlo desde otro punto del programa, mediante la utilización de una sentencia. Es importante saber que cualquier procedimiento o función que desee ser invocado por otro subprograma o por el programa principal, debe estar declarado antes de que pueda ser referenciado.**

**Procedimientos**

La estructura de un procedimiento es similar a la de un programa. Todo procedimiento debe estar declarado en la sección de declaración de subprogramas del programa principal. Cualquier procedimiento está formado de 3 elementos esenciales:

1. **Una cabecera de programa:** aquí se define el identificador o nombre del procedimiento que será posteriormente utilizado para invocarlo desde otro subprograma o desde el programa principal. Y un conjunto de parámetros que determinarán la información que se le proporciona al procedimiento (parámetros de entrada), y la información que devuelve una vez que ha sido ejecutado (parámetros de salida).

2. **Una sección de declaraciones:** Al igual que en el caso de un programa, es posible definir constantes, tipos de datos, variables, o incluso otros subprogramas dentro de un procedimiento. La característica de estas declaraciones es que todos los tipos de datos, variables o nuevos subprogramas sólo serán visibles (es decir utilizables) dentro del procedimiento donde han sido declarados.

3. **Un bloque de sentencias:** este bloque de sentencias o cuerpo del procedimiento, es la parte del procedimiento que implementa el algoritmo deseado. Se trata del equivalente al programa principal, con la única diferencia de que este bloque de sentencias está delimitado por un **END; (en lugar de un END. Utilizado en el caso de un programa)**

Existen dos posibles formas de declarar un procedimiento: utilizando una lista de parámetros o sin parámetros. En el caso de utilizar parámetros, se trata de un conjunto de variables que son proporcionadas al subprograma como información de entrada o salida. En el caso de no proporcionarse esa información, el programa no recibirá información, ni devolverá ningún tipo de resultado. (Por ejemplo puede pensarse en un procedimiento que muestre un mensaje por pantalla y luego termine.) La sintaxis para un procedimiento es la siguiente:

|  |
| --- |
| PROCEDURE **identificador;**  Declaraciones locales  BEGIN  Cuerpo del procedimiento  END;  PROCEDURE **identificador (lista de parámetros formales);**  Declaraciones locales  BEGIN  Cuerpo del procedimiento  END; |

**Funciones**

Las funciones son el segundo tipo de subprogramas definibles en Pascal. Al igual que los procedimientos cualquier función definida por el usuario debe ser declarada en la sección de declaraciones de subprogramas. Cualquier función está formada por 3 elementos fundamentales:

**1.Una cabecera de la función:** la cabecera estará formada por la palabra reservada FUNCTION, seguida de un nombre o identificador válido, y de un conjunto de parámetros de entrada. Además de especificar el TIPO del resultado que devuelve. Ver la siguiente figura

|  |
| --- |
| FUNCTION **nombre (parametro1:tipo1,…parámetro N: Tipo N): TIPO;** |

**2.Sección de declaraciones:** En esta sección pueden declararse constantes, tipos, variables y nuevos procedimientos o funciones, que serán locales a la función donde han sido declarados.

**3. Cuerpo de la función:** el bloque de sentencias que forman la función estará delimitado por las palabras reservadas BEGIN Y END;. Además una función debe incluir una sentencia de asignación donde aparezca en la parte izquierda el nombre de la función, seguida del símbolo de asignación, y en la parte derecha aparecerá una expresión del TIPO devuelto por la función.

Al igual que en el caso de los procedimientos existen dos posibles formas de declarar una función: utilizando una lista de parámetros o sin parámetros. La sintaxis para la declaración de una función es la siguiente:

|  |
| --- |
| FUNCTION **nombre: TIPO;**  Declaraciones locales  BEGIN  Cuerpo de la función  **Nombre:=valor devuelto por la función**  END;  FUNCTION **nombre (lista de parámetros formales):TIPO;**  Declaraciones locales  BEGIN  Cuerpo de la función  **Nombre:=valor devuelto por la función**  END; |

**NOTA**: Aunque la misión de ambos tipos de subprogramas es similar (implementar algoritmos que permitan resolver un determinado problema), existen diversas similitudes y diferencias que deben ser estudiadas.

**Paso de parámetros según sea el caso**

**Argumento o parámetro:** es una variable utilizada para recibir valores de entrada en una rutina o subrutina. Dichos valores, que serán enviados desde la rutina invocante, son llamados argumentos. La subrutina usa los valores asignados a sus parámetros para alterar su comportamiento en tiempo de ejecución.

Existen varias formas de pasar un argumento a una función (subrutina) o procedimiento, por valor, por referencia, etc. Dentro de los que están definidos para este curso se tienen:

**Paso por valor:** consiste en copiar el contenido de la variable que queremos pasar en otra dentro del ámbito local de la subrutina, se copia el contenido de la memoria del argumento que se quiere pasar a otra dirección de memoria, correspondiente al argumento dentro del ámbito de dicha subrutina. Se tendrán dos valores duplicados e independientes, con lo que la modificación de uno no afecta al otro.

**Paso por referencia:** consiste en proporcionar a la subrutina a la que se le quiere pasar el argumento la dirección de memoria del dato. En este caso se tiene un único valor referenciado (o apuntado) desde dos puntos diferentes, el programa principal y la subrutina a la que se le pasa el argumento, por lo que cualquier acción sobre el parámetro se realiza sobre la misma posición de memoria.

**¿Cuándo se debe utilizar parámetros por valor y cuándo por referencia?**

Si la información que se pasa al procedimiento no tiene que ser devuelta fuera del procedimiento, el parámetro formal que representa la información puede ser un parámetro valor (parámetro de entrada).

Si se tiene que devolver información al programa llamador, el parámetro formal que representa esa información debe ser un parámetro por referencia (parámetro salida).

Si la información que se pasa al procedimiento puede ser modificada y se devuelve un nuevo valor, el parámetro formal que representa esa información debe ser un parámetro por referencia (parámetro entrada \ salida).

**Similitudes entre funciones y procedimientos**

1. Ambos se declaran en la misma sección del programa. En ambos casos debe especificarse una cabecera, seguida de una lista de parámetros en el caso de que sean necesarios, un conjunto de declaraciones y un cuerpo del subprograma que resuelve el problema planteado.

2. Los parámetros, constantes, y variables definidos en ambos tipos de subprogramas son locales a los mismos. Es decir, son solo visibles (accesibles) dentro del subprograma donde han sido declarados.

3. Cuando se llama a una función o procedimiento, el número de parámetros actuales debe ser el mismo que el de los parámetros formales que han sido declarados en la cabecera.

**Diferencias entre funciones y procedimientos**

|  |  |
| --- | --- |
| **Funciones** | **Procedimientos** |
| La cabecera de una función incluye la declaración del tipo devuelto por la misma. | Al nombre o identificador del procedimiento no se le puede asignar ningún valor y por consiguiente no hay ningún tipo asociado al nombre de un procedimiento. |
| Para devolver el valor de una función debe utilizarse una sentencia de asignación. | En los procedimientos los valores se devuelven a través de parámetros por referencia (o variable VAR) |
| Una función se invoca utilizando su nombre en una expresión. | Un procedimiento se invoca mediante una llamada al mismo. |
| Una función devuelve un único valor a la parte del programa que lo ha referenciado. | Los procedimientos pueden devolver 0, 1 ,2 o N valores (para 0 valores puede estar realizando alguna tarea de entrada/salida.) |

**Laboratorio**

1. Dada una secuencia de números por teclado hasta que se introduzca el cero “0”, escribir una función, en Pascal, que determine si cada número leído es primo o nó.

2.- Elaborar un programa en Pascal que, usando subprogramas, realice el cálculo de las RAICES REALES en una ecuación cuadrática. Para ello debe pedir por teclado los valores de a,b, y c que deben ser pasados como parámetros desde el programa principal a un subprograma , realizar en el subprograma los cálculos de las raíces y devolver los resultado obtenidos hacia el programa principal, en donde se deben visualizar. No debe visualizar los resultados en los subprogramas.

Para calcular las raíces de la ecuación cuadrática se usa la fórmula: **ax2 + bx + c = 0** , teniendo en cuenta que :

1. Si a es igual a 0 y b es igual a 0, imprimiremos un mensaje diciendo que la

ecuación es degenerada.

1. Si a es igual a 0 y b no es igual a 0, existe una raíz única con valor –c/b.
2. En los demás casos, utilizaremos la fórmula: xi = (-b ± √(b2 – 4ac))/(2a)

* La expresión d = b2 – 4ac se denomina discriminante.
* Si d es mayor o igual que 0 entonces hay dos raíces reales
* Si d es menor que 0 entonces hay dos raíces complejas de la forma: x+yi, x-yi.

Siendo x el valor –b/2a e y el valor absoluto de √(b2-4ac)/(2a).

**Post Laboratorio**

**Valor de cada ejercicio ( 5 ptos c/u)**

1.- Se considera a serie definida por:

a1 = 0 a2 = 1 … an = 3\*an-1 + 2\*an-2 (para n≥ 3 )

Elaborar una función en PASCAL que determine el valor del primer término de esta serie superior o igual a un tope dado. Probar con el tope = 1000.

1. Elabore un programa en Pascal, UTILIZANDO SUBPROGRAMAS (por lo menos una 2 funciones y 2 procedimientos), para determinar si un número dado por pantalla es automórfico. Un número automórfico es aquel que al elevarlo sistemáticamente a una potencia mayor que cero, las últimas cifras del resultado corresponden al mismo número. Hacer el programa, para probar que un número es automórfico con un máximo de una potencia N dada. Ejemplo: 76 es automorfico para una potencia N= 4 porque: 761= 76; 762 = 5776; 763 = 438976; 764 = 33362176. Fíjese que al elevar el 76 a las 4 potencias sucesivas su resultado teminan en 76.
2. Se dice que dos números X y Y son cuadráticamente amigos, si al elevar al cuadrado cada uno de ellos, la suma de los dígitos del cuadrado de X es igual a Y y viceversa. Ejemplo X= 16 y Y = 13 son amigos cuadráticos pues 16 al cuadrado es 256 y 2+5+6 = 13 y por su parte 13 al cuadrado es 169 y 1+6+9 = 16.

Realice un programa en Pascal que , usando subprogramas, muestre los números dados, sus cuadrados y que determine si dos números dados, son o NO cuadráticamente amigos .Nota :

Los números deben leerse en el programa principal hacia el subprograma y este debe devolver los resultados, para que el programa principal los muestre.

4. Escribir un programa en Pascal, que muestre en pantalla cada una de las siguientes figuras geométricas y le permita al usuario elegir una de ellas para calcular su área. Implemente una Función para el cálculo del área de cada figura, además usar funciones gráficas para presentar las figuras.

****

**Instrucciones para el envío de la tarea :**

Crear una carpeta comprimida que incluya:

* Un archivo .doc con los algoritmos de los problemas propuesto en el Postlaboratorio y con las capturas de pantalla de las corridas de los programas respectivamente.
* Los archivos .pas de los problemas resuelto (un archivo .pas por cada programa)

Además debe enviar la HOJA DE CONTROL DE PRACTICAS con la autoevaluación de la práctica de la semana.

Adjuntar archivos a la tarea correspondiente en Módulo 7