INTRODUCIÓN A LA PROGRAMACIÓN
PRÁTICO – 6
ARRAYS, SUBRUTINAS y FUNCIONES

1 – Ejercicio con arrays

a) Escribir un programa que realice la suma de dos vectores de dimensión 5 cada uno y de componentes reales.

b) (*) Escribir un programa que efectué el producto escalar entre dos vectores dados de dimensión dada. Comprobar con la función intrínseca dot_product.

c) (*) Escribir un programa que intercambie los valores de dos índices dados de un array.

d) Escribir un programa que invierta el orden de los valores de un array, por ej.:

(5,8,3,9) pasa a ser (9,3,8,5)

Se debe utilizar un solo array el original. Sugerencia: utilizar la parte c)

e) (*) Escribir un programa que verifique en un array dado de dimensión N, si existen elementos repetidos. El programa debe devolver en pantalla todos los elementos y la cantidad de veces que aparece cada uno.

f) Escribir un programa para que dado un array de enteros, lo modifique de acuerdo a un umbral ("threshold"), esto es si algún valor del array no supera un cierto umbral entero k, entonces ese valor debe cambiarse a cero.

2) Este ejercicio presenta algunos algoritmos más complicados con arrays.

a) (*) Escribir un programa que encuentre el máximo valor de un array de reales.

b) (*) Escribir un programa que ordene un array dado. Para esto considere un algoritmo que haga lo siguiente:

- Halle el máximo de un array dado en el rango de 1 a N
- Luego intercambie el elemento hallado con el último del rango (N)
- Vuelva a ejecutar los pasos 1 y 2 para el array que queda definido desde el primer elemento al elemento N-1, en forma repetitiva hasta llegar a N=1.

Sugerencia: encapsule las operaciónes anteriores en funciones y subrutinas:
function *findMaxElement(array,firstIndex,lastIndex,indexMaxElement):*

firstIndex y lastIndex son el primer y último índice del rango donde efectúo la búsqueda y indexMaxElement el índice del elemento máximo encontrado.

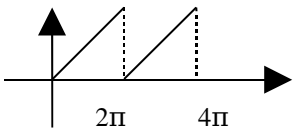
subroutine *swapElements(array, index1, index2):*

Intercambia los valores de ambos índices pasados del array.

Estas funciones pueden ser internas al programa (con "contains" pues no llaman a otras)

c) Las funciones periódicas pueden ser desarrolladas en series trigonométricas (series de Fourier). Por ejemplo la función:

y = x para $0 < x < 2\pi$ de período 2*pi



Se pude escribir como la suma infinita:

$$2(\frac{\sin(x)}{1}-\frac{\sin(2x)}{2}+\frac{\sin(3x)}{3}-\frac{\sin(4x)}{4}$$

Escribir un programa que aproxime la función con la suma hasta un término N dado (sin usar arrays).

Luego escribir el mismo programa usando arrays. Sugerencia: crear un array con los enteros 1,2,..,hasta N con el correspondiente signo según la serie usando un do implícito, luego tener en cuenta que al multiplicar un escalar (como "x") por un array, se multiplica dicho número por cada componente. Igualmente al aplicar una función intrínseca (como sin) a un array, se aplica la función a cada componente. Usar una función intrínseca aplicada al array para que devuelva la suma de sus elementos.

d) Considere una función de dos variables reales x,y: $\exp(-(x^2+y^2))$ definida en el dominio [x,y] = [0..1,0..1]. Para evaluarla en 10 intervalos en la variable x e y (o sea desde x=0 hasta x=1 con incrementos de 0.1 y l mismo para la variable y), construimos una grilla cuadrada de 10*10 y evaluamos la función en cada uno de sus puntos. Escribir un programa que haga esto.

e) (*) Dadas dos matrices de n*k y k*m respectivamente, escribir un programa que devuelva el producto matricial en una tercera matriz de n*m. Sugerencia: encapsular el producto escalar de una fila por una columna de ambas matrices, en una función interna *scalarProduct (fila, columna*).

f) Escribir un programa que tranponga una matriz, esto es: dada una matriz A n,m ,el programa debe devolver otra B m,n de forma que A(i,j) = B(j,i).

3) Ejercicios con subrutinas y funciones.

a) Escribir un programa que calcule el binomio de newton del práctico 5 pero con funciones. Una función que calcule el factorial de un nro., otra que desarrolle el binomio mismo. El programa principal debe verse algo así como:

```
program binomioDeNewton
    ….
binomio = desarrollarBinomio(a,b,n)
    ….
```

Las funciones factorial y desarollarBinomio deben ser externas y previamente compiladas antes de compilar el programa principal.

b) (*) Escribir un programa que dado N vectores espaciales (o sea con tres componentes) calcule: la suma vectorial de todos ellos y el producto vectorial de todos ellos (el primero con el segundo, su resultado con el tercero, su resultado con el cuarto,etc.).
Sugerencia: Definir un tipo de dato s*paceVector* que contenga tres tipos de datos intrínsecos x,y,z todos ellos reales. Luego definir un array de dimensión N donde cada elemento es de tipo *spaceVector.* Luego definir las funciones *sumVectors(vector1,vector2)* donde vector1, vector2 y sumVectors son de tipo *spaceVector.* También definir la función c*rossProduct*(*vector1,vector2*) de forma análoga.

c) Escribir un programa que escriba en pantalla el triángulo de Pascal:

```
          1
        1   1
      1   2   1
    1   3   3   1
  .....................
```

Cada línea contiene los coeficientes binomiales de un desarrollo del binomio de Newton; la cuarta línea tiene los coefs. Para n=3 C(3,0), C(31), C(3,2), C(3,3).
Para escribir el programa se deben usar las siguientes subrutinas y funciones:

- Una función factorial
- Una función combinaciones
- Una subrutina imprimir línea: tener en cuenta que argumentos se le debe pasar; observar que los caracteres que imprime están separados por un espacio y debe comenzar en una posición determinada.

- Una subrutina que imprima el triángulo de Pascal hasta una profundidad h dada.

Sugerencia: Siempre esrcribir las funciones y subrutinas y sus argumentos de la forma más general posible, por ejemplo la subrutina para imprimir una línea debe aceptar un argumento *posiciónInicial* (donde arranca a imprimir); luego le vamos pasando el valor adecuado al argumento a medida que se imprime cada línea de acuerdo a la lógica del programa.