UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Unidad 2. La lógica de programación

Tema 5. Subprogramas. Procedimientos y pase de parámetros

UNIVERSIDAD DE LOS ANDES

Subprogramas

- Contenido:
  - Tipos:
    - Funciones
    - Procedimientos
  - Pase de parámetros:
    - Por valor
    - Por referencia
  - Representación algorítmica
  - Codificación
  - Pase de vectores como parámetros
- Objetivo:

  Desarrollar habilidades en el uso de subprogramas
- Bibliografía:
  - Deitel y Deitel, cap. 5 y sec. 7.1-7.4.
  - http://www.ing.ula.ve/~ibc/pr1
  - http://www.webdelprofesor.ula.ve/ingenieria/ibc
  - Navas y Besembel, tema VI.
  - Joyanes, sec. 4.2 y 5.1-5.5.

UNIVERSIDAD DE LOS ANDES

Funciones en C/C++

- Funciones de biblioteca (librerías) : El lenguaje C/C++ tiene sus propias funciones incorporadas que permiten realizar ciertas operaciones o cálculos de uso común
  - Contiene una amplia colección de funciones para llevar a cabo cálculos matemáticos comunes, manipulaciones con cadenas de caracteres, manipulaciones con caracteres, operaciones de entrada/salida y muchas otras operaciones útiles.
  - Esta biblioteca de funciones comunes construida una vez, puede ser re-utilizada por diferentes programas
- Funciones definidas por el programador para realizar determinadas tareas

TDSO	**Definición** ⇒ nomFunción(Tipo: parametroFormal,...):tipoResultado
	Llamada ⇒ Xj = objeto.operación(listaParmetrosActuales)
C/C++	**Definición** ⇒ tipoResultado nomFuncion(listaParametrosFormales)
	Llamada ⇒ nomVar = nomFuncion(listaParametrosActuales)

UNIVERSIDAD DE LOS ANDES

Tipos de parámetros

- **Parámetros formales** (parámetros de entrada o argumentos): Declaraciones de los parámetros en la definición de la función

  Son variables locales, ya que se declaran en la definición de la función

  Dejan de existir cuando la función retorna o regresa a la función que llamó o invocó al subprograma

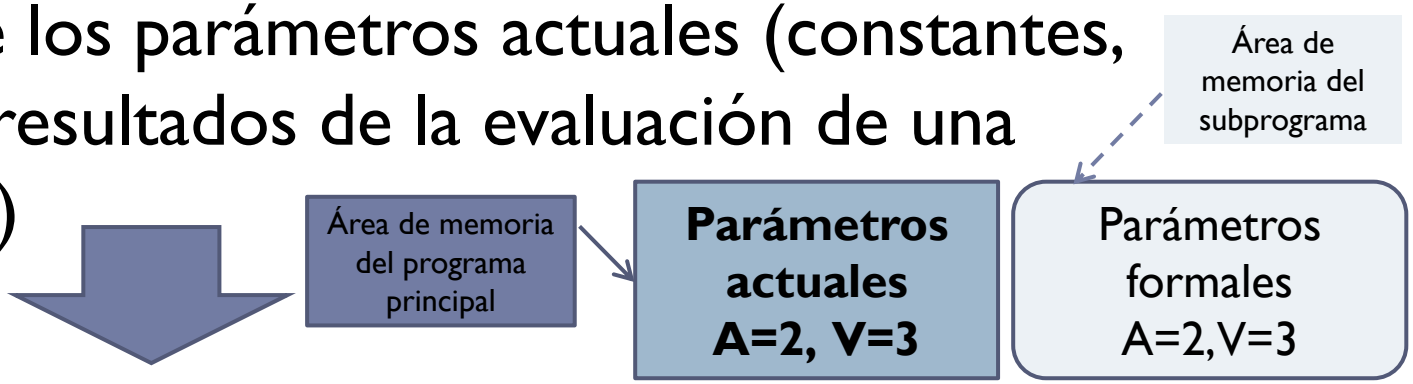
  Toman valor cuando se realiza una llamada o invocación a la función

  Toman valor a través del pase de parámetros

- Pase de parámetros:
  - Por valor
  - Por referencia
- **Parámetros actuales**: Valores que toman los parámetros formales y que son proporcionados a la función que es llamada por la función que la llamó

Pase de parámetros por valor

- Los parámetros formales reciben una COPIA de los valores de los parámetros actuales (constantes, variables, resultados de la evaluación de una expresión)



- Los cambios que realice la ejecución del subprograma en dichos valores no se reflejan en los parámetros actuales
- En C/C++ todas las llamadas, por defecto u omisión, se realizan de esta manera, es decir, por valor

Ejemplo 1

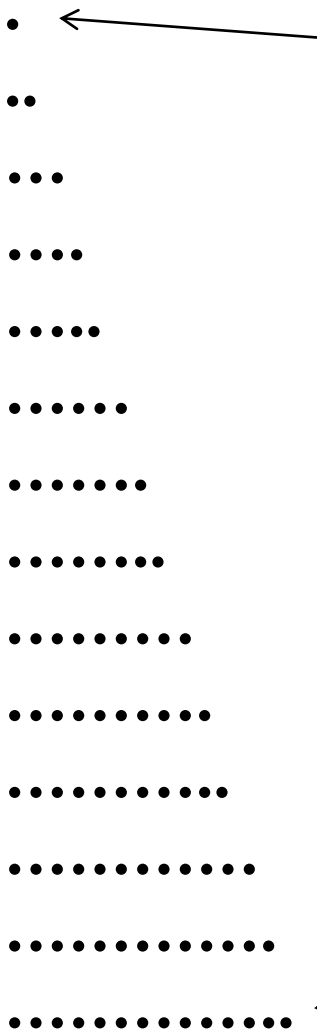
```
#include<stdio.h>
//      Procedimiento para imprimir puntos

void     dibujarPuntos ( int numPuntos )
{       int   i;
        for( i = 1; i <= numPuntos; i++)     pintf("%c", '.');
}
//    Programa principal
int    main()
{
        dibujarPuntos( 14);
        return 0;
}
```

Subprogramas al comienzo

14

Normalmente se coloca al final

Salida del programa del ejemplo 1

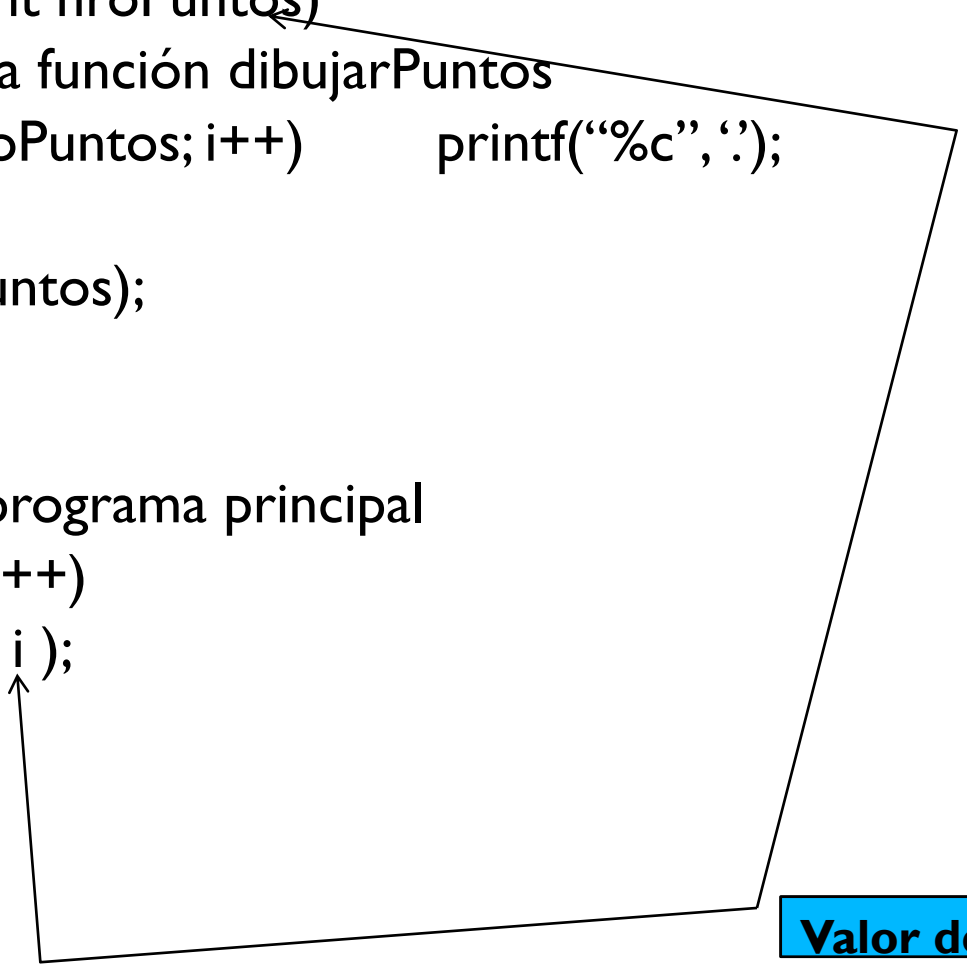


I
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

UNIVERSIDAD DE LOS ANDES

Ejemplo 2

```
void   dibujarPuntos( int nroPuntos)
{     int  i;       //  i  de la función dibujarPuntos
       for (i = 1; i <= nroPuntos; i++)        printf("%c", '.');
       nroPuntos+= 2;
       printf( "%i", nroPuntos);
}
int   main()
{    int  i;      //  i   del programa principal
      for ( i = 1; i <= 5; i++)
      {    dibujarPuntos( i );
           printf ("%i", i);
      }
      return 0;
}
```



UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1

- Enunciado del problema

  Encontrar el máximo valor de tres valores dados max(x1, x2, x3)

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	X1 X2 X3	Primer valor Segundo valor Tercer valor	Real Real Real	[min, max] [min, max] [min, max]
Proceso	Dados los valores de X1, X2, y X3 Utilizar la función max(X1, X2, X3) para calcular el máximo valor mediante la comparación de los mismos			
Salida	Mensaje indicando max	Máximo valor de los tres	Real	[min, max]

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 1. Diseño

maximo(Real x, Real y, Real z): Real {pre: }		{pos: max ≥ x, y, z}

1
2
 	
3
 	
4
	max = x
	Si (y > max) entonces
	  max = y
	fsi
	Si (z > max) entonces
	  max = z
	fsi
	devolver max
		●**x:** Real. Variables locales declaradas como parámetros formales.
		●**max:** Real. Variable auxiliar para guardar el máximo valor entre los valores enviados por los parámetros formales.

1 2	x=4.0, y = 5.0, z = 60.0 => max = 60.0 x = 0.0, y = 0.0, z = 0.0 => max = 0.0	●Caso con 3 valores diferentes ●Caso con 3 valores iguales

mayor {pre: }		{pos: maximo∈ℜ }
1 2 3	Escribir (“Introduzca 3 números”) Leer (a, b, c) Escribir (“Mayor = ”, maximo(a, b, c))	●**a, b, c:** Real. Números dados. ●**maximo()**. Función que calcula el mayor o máximo de tres valores dados.
1 2	a=4.0, b = 5.0, c = 60.0 => Mayor = 60.0 a = 0, b = 0, c = 0 => Mayor = 0	●Caso con 3 valores diferentes ●Caso con 3 valores iguales

Ejercicio resuelto 1. Implementación

```
#include <stdio.h>
float  maximo (float x, float y, float z)
{   float    max;

    max = x;
    if (y > max)                    max = y;
    if (z > max)          max = z;
    return   max;
}
```

```
int   main()
{
   float    a, b, c;
   printf ("Introduzca 3 números")
   scanf("%d %d %d", &a, &b, &c);
   printf("Mayor = %d", maximo(a, b, c));
   return 0;
}
```

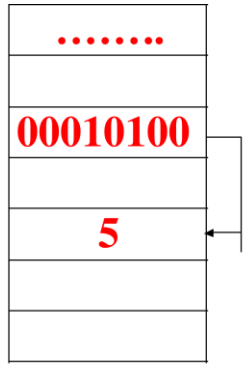
```
int   main()   //  Otro ejemplo
{
   float    a, b, c, P, Q, R, S;
   printf ("Introduzca 3 números")
   scanf("%d %d %d", &a, &b, &c);
   P = maximo(3, 4, 6);
   Q = maximo(a, b, c);
   R = maximo(a + 1, b, 1);
   S = maximo(P, Q, R);
   printf("Mayor = %d", S);
   return 0;
}
```

Pase de parámetros por referencia

- **Pasa la dirección de memoria del parámetro actual**
- Se comparte la variable entre el programa y el subprograma

- El parámetro actual puede ser accedido y modificado por el subprograma
- **Apuntador:** variable especial cuyo contenido es la dirección o localización de memoria (referencia) de otra variable

Ap1 es una variable de tipo **Apuntador**
y
A una variable de tipo Entero



UNIVERSIDAD DE LOS ANDES

Pase de parámetros por valor vs. pase de parámetros por referencia

- El pase de parámetros por referencia permite devolver VARIOS valores desde un subprograma, permitiendo la implementación de los procedimientos (tipo de subprograma) en C/C++ a través de los parámetros actuales
- Si se pasa la dirección de una variable (parámetro actual) a un procedimiento, entonces éste puede cambiar el valor de dicha variable
- Un procedimiento devuelve CERO, UNO O MAS valores en los parámetros formales declarados como referencias a los parámetros actuales
- Si una función C/C++ devuelve un solo valor usando la sentencia *return*, **todos** sus parámetros deben ser pasados por valor
- Si se pasa el valor de una variable (parámetro actual) a una función, ésta puede usar el valor de la variable PERO no lo cambia

Pase de parámetros por referencia

Definición del procedimiento en C/C++

```
void nombreProcedimiento (tipoDato *parámetroFormal, ...)
{
        // Cuerpo de la función
}
```

Llamada al procedimiento

```
nombreProcedimiento(parámetroActual, ...)
```

UNIVERSIDAD DE LOS ANDES

Ejemplo

```
#include <stdio.h>

void   cuadrado( int   num, int   *aptCuadrado )
{
     *aptCuadrado = num * num;
}

int main( )
{
   int   c =12;

   printf("valor de c antes de la llamada %d \n", c);
   cuadrado( 5, &c);
   printf("valor de c después de la llamada %d \n", c);
   printf("El cuadrado de 5 es %d \n", c);
   return 0;
}
```

UNIVERSIDAD DE LOS ANDES

Resumen

```
void        nombreProcedimiento (tipoDato *parámetroFormal, ...)
{
                    // Cuerpo de la función
}
```

nombreProcedimiento(listaParámetrosActuales)

TDSO

Definición ⇒ **nomFunción**(Tipo: parámetroFormal,...)[:*tipoResultado*]
Llamada ⇒ Xj = **nomFunción**(listaParámetrosActuales)

C/C++

Función

Definición ⇒ *tipoResultado* **nomFuncion**(listaParametrosFormales)
Llamada ⇒ nomVar = **nomFuncion**(listaParametrosActuales)

Procedimiento

Definición ⇒ *void* **nomProcedimiento**(listaParametrosFormales)
Llamada ⇒ **nomProcedimiento**(listaParametrosActuales)

UNIVERSIDAD DE LOS ANDES

Importante

- Un programa en C/C++ consta de una o varias funciones/procedimientos
- Un programa siempre debe contener la función *main*
- La ejecución de un programa siempre comienza por la función *main*
- Una función devuelve un sólo valor en el nombre de la función
- Un procedimiento devuelve cero, uno o mas valores en los parámetros formales (parámetros por referencia)

UNIVERSIDAD DE LOS ANDES

Pase de vectores como parámetros en C/C++

- Un vector completo se puede pasar a una función como parámetro
- Para pasar un vector como parámetro actual a una función, se especifica únicamente su nombre, sin corchetes ni subíndices.

```
char  caracteres[80] = "esta cadena es constante";
……
F1(caracteres, longitud);
……
```

- El parámetro formal debe ser definido dentro de la función, se escribirá un par de corchetes vacíos, es decir, el tamaño del vector no se especifica.

```
void F1  (char string[], int lng)
```

Pase de vectores como parámetros en C/C++

- En C/C++ los arreglos en general son pasados como parámetros por referencia. Esto es, el nombre del arreglo es la dirección del primer elemento del arreglo
- En C/C++ un elemento cualquiera de un arreglo puede ser pasado a una función por valor o por referencia, tal y como se hace con una variable simple

```
float     media(int a, float x[])     // Definición de la función
{                                     // Note que se incluyen los corchetes vacíos
    …….
}
void    main ()
{   int n;
    float med, lista[100];
    ……
    med = media(n, lista);            // Esta llamada pasa como parámetros actuales la longitud
                                      // del vector y el vector
    ……..                              // Note que no se incluyen los corchetes
}
```

Paso de vectores como parámetros

```
// Paso de vector y elementos individuales del vector como
parámetros
# include <iostream.h>
#define NROELTOS 5
using spacename std;

void modificarVector(int [], int);   // Prototipo de la función
void modificarValor(int);            // Prototipo de la función
void modificarReferencia(int &);     // Prototipo de la función

void main()
{    int vector[NROELTOS] = {0, 1, 2, 3, 4}, j;
     cout << "Los valores del vector original son:" << endl;
     for (j = 0; j < NROELTOS; j ++)  cout << vector[j] << endl;
     modificarVector(vector, NROELTOS);
     cout << "Los valores del vector modificado son:" << endl;
     for (j=0; j<=NROELTOS-1; j ++)   cout << vector[j] << endl;
     cout << "Efectos de pasar un elemento de un vector como
parámetro por valor" << endl;
     modificarValor(vector[3]);
```

Paso de vectores como parámetros

```
        cout <<"El valor del cuarto elemento del vector es:" << vector[3]
             << endl;
        modificarReferencia(vector[3]);
        cout <<"El valor del cuarto elemento del vector es:" << vector[3]
            << endl;
}
void modificarVector (int b[], int num)
{    int k;
        for (k = 0; k <= num - 1; k ++)    b[k] *= 2;
}
void modificarValor (int e)
{   e *= 2;
     cout << "Valor modificado del elemento ="  << e << endl;
}

void modificarReferencia (int &e)
{   e *= 2;
    cout << "Valor modificado del elemento ="  << e << endl;
}
```

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2

- Enunciado del problema

  Calcular la media de un conjunto de n números y luego calcular la desviación de cada número respecto de la media

- Análisis E-P-S

Entrada	Variable	Descripción	Tipo de dato	Rango válido
	n numeros(i)	Número total de valores Vector de n valores	Entero+ Real[MV]	$0 \leq n \leq MV$ Según el lenguaje
Proceso	Definir MV y Obtener n validando según su rango Obtener el conjunto numeros de tamaño n Calcular la media = $\sum$ numeros(i) / n Calcular la desviacion(i) = numero(i) - media			
Salida	Mensaje indicando media desviacion(i)	 Valor promedio del conjunto $\forall$i que tan alejado está el valor i de la media	 Real Real[MV]	 El disponible en el lenguaje Según el lenguaje

Ejercicio resuelto 2. Diseño

media(Entero t, Real x[]): Real {pre: t∈[1,MV]} {pos: t ∈ [1, MV], (m/t) ∈ [x(1), x(n)]}		
1 2 3	m=0 [m=m+x(i)] i=1,t Devolver (m/t)	•**m:** Real. Acumulador para la suma de todos los valores del vector x •**i:** Entero. Subindice para recorrer el vector x
1 2	t=4, x = [4.0, -2.1, 9.3, 2.3] ⇒ 3.375 t=1, x = [1.] ⇒ 1.0	Caso exitoso Caso borde

Función

Procedimiento

desviacion(Entero nv, Real v[], Real me, Real des[]) {pre: nv∈[1, MV] ∧ me∈[v(1), v(nv)]} {pos: nv ∈ [1, MV] ∧ me, des(i) ∈ [v(1), v(nv)] ∀i∈ [1, nv]}		
1	[des(i)=v(i)-me] i=1, nv	•**i:** Entero. Subindice para recorrer los vectore
1 2	nv=4, v=[4., -2.1, 9.3, 2.3], me=3.375 ⇒ des=[0.625, -5.475, 5.925, -1.075] nv=1, v=[1.0], me=1.0 ⇒ des=[0.0]	Caso exitoso Caso borde

Ejercicio resuelto 2. Diseño

mediaYdesviacion() {pre: } {pos: n ∈ [1, MV] ∧ m, desviaciones(i) ∈ [numeros(1), numeros(n)] ∀i∈ [1, n]}		
1 2 3 4 5 6 7	[n=valor suministrado] (n < 0 ∨ n > MA) [numeros(i)=valor suministrado] i=1,n m=media(n, numeros) desviacion(n, numeros, m, desviaciones) [desplegar numeros(i)] i=1,n desplegar "Media calculada = ",media [desplegar desviaciones(i)] i=1,n	•**n:** Natural. Total de valores dados •**numeros:** Vector[MV]De Real. Valores a los que se le calculará su media y su desviación de la media. •**m:** Real. Contiene la media de los valores dados •**media()** : Función que regresa la media del vector de valores reales •**desviaciones:** Vector[MV]De Real. Contiene las desviaciones de los valores dados respecto de la media. •**desviacion():** Procedimiento que regresa el vector que contiene la desviación de cada valor en el vector de reales respecto de la media.
1 2	n=4, numeros=[4.0, -2.1, 9.3, 2.3] ⇒ Media calculada = 3.375, desviaciones=[0.625, -5.475, 5.925, -1.075] n=1, numeros=[1.0] ⇒ Media calculada = 1.0, desviaciones=[0.0]	Caso exitoso Caso borde

UNIVERSIDAD DE LOS ANDES

Ejercicio resuelto 2. Implementación

```
#include <iostream>
#define  MV        100
using spacename std;                    // MV: Maximo numero de valores
float      media(int t, float x[]);
void       desviacion(int nv, float v[], float me, float des[]);
void       main ()
{   int n, i;
    float numeros[MV], m, desviaciones[MV];
    do
    {       cout << "Introduzca el total de numeros a procesar" << endl;
            cin >> n;                       // Leer el total de numeros a procesar
    }while(n < 0 || n >= MV);
```

Ejercicio resuelto 2. Implementación

```
    for (i = 0; i < n; i++)                    // Leer cada numero
    {       cout << "\nIntroduzca numero[" << i << "]= ";
            cin >> numeros[i];
    }
    m=media(n, numeros);
    desviacion(n, numeros, m, desviaciones);
    for (i = 0; i < n; i++)                    //  Escribir cada numero leido
            cout<< "\n numero[" << i << "]= " << numero[i];
     cout << "Media calculada = " << m << endl;
     for (i = 0; i < n; i++)                   //  Escribir cada desviacion calculada
            cout << "\nLa desviacion de " << numeros[i] << " es " << desviaciones[i];
}
```

Ejercicio resuelto 2. Implementación

```
float   media(int t, float x[])
{  float  m=0;
   int    i;
   for(i=0; i<t; i++)        m+=x[i];        // Sumar cada numero
   return (m/t);                             // Regresa la media
}
void    desviacion(int nv, float v[], float me, float des[]);
{  int    k;
   for (k = 0; k < nv; k++)
          des[k] = v[k] – me;        // Calcular la desviacion de cada numero
}
```

Función que regresa un valor real

Procedimiento que calcula el vector de desviaciones

- ¿Qué es pase de parámetros?
- ¿Cómo se realiza el pase de parámetros por valor y por referencia?
- ¿Qué es un procedimiento?
- ¿Cómo se implantan los procedimientos en C/C++?
- ¿Cuál es la diferencia entre funciones y procedimientos?
- ¿Cómo pasar los vectores como parámetros en C/C++?

Resumen

¿Cuáles son los conceptos relevantes de esta clase?

UNIVERSIDAD DE LOS ANDES

Ejercicios

- **Para cada uno de los enunciados dados a continuación realizar: Análisis en E-P-S, diseño en pseudocódigo y codificación en C o C++ utilizando funciones de librería**

1. Escribir una función que reciba los valores enteros de x y n > 0 como parámetros de entrada y devuelva el valor de x^n como salida.
2. Escribir un programa que tome un conjunto de pares de números enteros a y b, y calcule la potencia a^b de cada par. El fin de entrada de datos viene dado por a = -1 y b = -1
3. Escribir una función que tome un carácter como parámetro de entrada y devuelva su tipo que podrá ser uno de los siguientes: (1) letra mayúscula de la 'A' a la 'Z', (2) letra minúscula de la 'a' a la 'z', (3) dígito del '0' al '9'

UNIVERSIDAD DE LOS ANDES

Ejercicios

4. Escribir una función que tome como parámetros de entrada dos instantes de tiempo expresados en horas, minutos y segundos e indique si el primero es anterior al segundo
5. Escribir una función que reciba como parámetro un número y devuelva como resultado un valor de tipo lógico que indique si el número es o no par
6. Escribir una función que reciba como parámetro dos números y devuelva como resultado un valor de tipo lógico que indique si el primer número es múltiplo del segundo
7. Utilizando la función factorial, escribir una función que calcule el número combinatorio dado por la fórmula siguiente: $n! / (m! (n-m)!)$

UNIVERSIDAD DE LOS ANDES

Ejercicios

8. Escriba una función que reciba como parámetros las dos coordenadas cartesianas (x, y, z) de un punto en el espacio y devuelva como resultado un número del 1 al 8 que indique el cuadrante al cual pertenece al punto (no considere los ejes de coordenadas)
9. Los registros mensuales de visitantes al parque Chorros de Milla durante el año 2011 son los siguientes:

Mes	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
#visitantes	100	90	120	150	210	250	300	310	280	230	160	120

Escribir un programa, estructurado en funciones, para realizar las siguientes tareas:

Almacenar los datos en un vector y los nombres de los meses en otro vector

Calcular y escribir el promedio de visitantes durante el año 2011

Calcular y escribir los nombres de los meses con el mayor y el menor número de visitantes

Ordenar de menor a mayor ambos vectores y desplegar el resultado ordenado en forma creciente y en forma decreciente

Ejercicios

10. Determinar la salida del siguiente programa si el dato de entrada es el entero 10

```
#include <iostream>
using spacename std;

int a (int y)
{   y *= 2;
    return y;
}
int b (int x)
{    x += 5;
    return x;
}
```

```
void main ()
{
    int x, x1;
    cout << "Introduzca un numero" << endl;
    cin >> x;
    x1 = b(x);
    cout << x1 << endl;
    x1 = a(x);
    cout << x1 << endl;
}
```