Funciones: Pasos por Referencia Recursividad

Fundamentos de Programación
Fundamentos de Programación I

Parámetros por referencia

- Con la instrucción ***return*** sólo se puede devolver un valor calculado.
- A veces las funciones calculan varios datos y deberían poder devolverse

Algunos de los parámetros de la lista de parámetros se pueden utilizar como parámetros de salida

Parámetros por referencia = parámetros de salida

```
int f1 ( int a, int &b )
```

p. valor

p.referencia

- Los cambios realizados sobre una variable pasada por referencia en una función afectan directamente a la variable

```
int f1(int &b)
{
     b =b+1;
     return 2*b;
}
```

```
int main()
{
   int x,y;

   x = 3;
   cout << x;

   y = f1(x);

   out << x;

   return 0;
}
```

3

4

- Ejercicio 1: Escribe una función que transforme un punto en coordenadas polares a cartesianas

Entradas: Un punto como coordenadas polares **(modulo y ángulo)**
Salidas: Coordenadas x e y (x,y)
Análisis: ¿cuáles son las expresiones que te permiten pasar de Coordenadas polares a cartesianas??

```
#include <iostream.h>
#include <math.h>

//Prototipo de la función
void PolaresACartesianas(float modulo, float angulo, float &x, float &y);

int main()
{
  float m,ang;
  float posx, posy;

  cout << "Introducir el modulo y el angulo del punto en coordenadas polares:\n";

  cin >> m >> ang;

  //Calculo la posicion x, y

  PolaresACartesianas(m, ang, posx, posy);

  cout << "Las coordenas cartesianas correspondientes a: " <<m << ',' << ang << endl;
  cout << "("<< posx << "," << posy<<")"<< endl;

  return;

}


void PolaresACartesianas(float modulo, float angulo, float &x, float &y)
{
   x = modulo*cos(angulo);
   y = modulo*sin(angulo);

   return;

}
```

- Ejercicio 2:Escribe una función que, dados dos enteros positivos **x** e **y**, calcule el cociente de la división entera y el resto utilizando únicamente restas.

Entradas: Dos números enteros (a y b)
Salidas: El cociente y el resto que han sido calculados
Análisis: ¿Cúal es el algoritmo que permite calcular la división entera mediante restas??

```
/*****************************************************************************
* Funcion division entera
* Descripcion: Calcula el cociente y el resto de la division entera
* Parámetros:
* Nombre: E/S
*   a      E
*   b      E
*   resto  S
*
* Valor devuelto:
* char Es de tipo  carácter (valor devuelto)
*****************************************************************************/
int DivisionEntera(int a, int b, int &resto)
{
 int cociente;

 cociente=0;

 while( a > b)
 {
   a = a –b;
   cociente ++;
  }
  resto = a;

  return cociente;
}
```

```
#include <iostream>
//Prototipo de la funcion

int DivisionEntera(int a, int b, int &resto);

int main()
{
   int dividendo, divisor;
   int cociente, resto;
   //Leo datos
   cout << “Introducir dividendo y divisor:” << endl;
   cin >> a;
   cin >> b;

   cociente = DivisionEntera(dividendo, divisor,resto);
  //Escribo datos
   cout << “El cociente es:” << cociente << y el resto:” << resto <<endl;

   return 0;
}
```

Llamada a la función

Recursividad

- Una función recursiva es aquella que en su definición hay una llamada así misma.
- En una función recursiva siempre debe haber:
  - Un caso base (o condición de parada): caso en que la función realiza su tarea sin necesidad de usar llamadas recursivas
  - Un caso general, donde se llama a la misma función (recursión). Estas llamadas recursivas deberán resolver versiones más pequeñas de la tarea que realiza la función que se está definiendo.

siempre debemos buscar el caso base

Se debe evitar que la recursón sea infinita

- Toda llamada a la función recursiva debe conducir tarde o temprano al caso base, de lo contrario, la función nunca terminará (número infinito de llamadas)
- Los parámetros que se pasan a la función deben modificarse de una llamada a otra, y de forma que se alcance el caso base.
- La función recursiva podrá incluir otras operaciones para construir el resultado.

- Ejercicio 3: Realizar una función recursiva que calcule el sumatorio y el productorio de los n primeros números naturales.

```
/************************************************
Funcion que calcula el sumatorio de forma recursiva
************************************************/
int Sumatorio(int n)
{
  int res;

  if (n == 1)
     res = 1;

  else
     res = n + sumatorio( n -1);

  return res;

}
/************************************************
Funcion que calcula el productorio de forma recursiva
************************************************/
int Productorio(int n)
{
   int res;

if ( n == 1)
     res = 1;
else
     res = n * Productorio ( n-1);

return res;
}
```

Case base

Llamadas recursivas de talla menor

Case base

Llamadas recursivas de talla menor

```
#include <iostream>

using namespace std;


//Prototipo de la funcion

int Sumatorio(int n);
int Productorio(int n);
int main()
{
  int num;
  int suma, producto;

  cout <<"Este programa calcula el sumatorio  y productorio de  n numeros naturales\n";
  cout << "Utilizando una función recursiva. Introduce n:\n";

  cin >> num;

  suma = Sumatorio(num);

  producto = Productorio(num);


 cout << "El productorio es:" << producto;
 cout << "El sumatorio es:" << suma;

 return 0;
}
```

Llamada a la función

- Ejercicio 4: Escribir una función recursiva que calcule la división entera entre dos números:

```
int DivisionEntera(int a, int b, int &resto)
{
  int cociente;

  if ( a < b) //Condicion de parada
  {
    resto = a;
    cociente = 0;
  }
  else
  {
    a= a - b;  //Preparacion de los parametros

    cociente = DivisionEntera(a, b, resto);

    cociente = cociente + 1; //Construccion de la solución

  }
  return cociente;
}
```

- Ejercicio 11: Realizar una función recursiva que calcule la suma de dos números enteros utilizando solamente operaciones de incremento y decremento.

```
/*****************************************************************************
* Funcion que suma dos numero enteros
* Descripcion: Calcula el cociente y el resto de la division entera
* Parámetros:
* Nombre: E/S
*   a      E
*   b      E
*
* Valor devuelto:
* int (valor devuelto)
*****************************************************************************/
int Suma(int a, int b)
{
  if ( a == 0)
     res = b;
  else
     res = suma(a-1,b+1)

  return res;
}
```

- Ejercicio 5: Desarrollar una sencilla función rescursiva que dada una frase la muestre en orden inverso, letra letra por pantalla, ejemplo:

Ingeniería informática >>>> acitámrofni aíreinegnI

```
void ImprimirCadena(void)
{

  char ch;

  cin.get(ch);

  if(ch!= '\n')
  {

    ImprimirCadena();
    cout << ch;

  }

  return;
}
```

- Ejercicio 6: Recursividad indirecta. Escribe una función que devuelva true si un número entero pasado como parámetro es par y false en caso contrario.

```
#include <iostream.h>

// Prototipos
bool Par(int n);
bool Impar(int n);



int main()
{
    int a;

    cout << "Introduce un número entero : ";

    cin >> a ;
    cin.ignore();

    if( Par(a) )
      cout << " ... es par " << endl;

    else
      cout << " ... es impar " << endl;

    system("pause");

    return 0;
}
```

```
bool Par(int n)
{
  bool res;
  if (n == 0)

    res = true;

  else

    //Llamo a la otra función recursiva
   res =Impar(n-1);
   return res;
}
```

```
bool Impar(int n)
{
  bool res;
  if (n == 0)

    res = false;
  else
 //Llamo a la otra función recursiva
     res = Par(n-1);
   return res;
}
```

- Ejercicio2: Se desea calcular la trayectoria de un determinado móvil sometido a un movimiento uniforme (v = e/ t). El móvil estará situado en una posición inicial (x_0, y_0) y deberá recorrer 100 metros. Calcular las posiciones intermedias para los siguientes móviles:
  - **Una hormiga (0.5 km/hora)**
  - **una persona (3.5 km/hora)**
  - **Un caballo (30 km/hora)**

Nota: Para la resolución del ejercicio diseñar una función que dada una posición inicial, una velocidad y un intervalo de tiempo calcule la posición alcanzada por el móvil.