Funciones y Bibliotecas

Diseño de una Función

Para diseñar una función yo debo tener en cuenta tres cosas:

1- **Determinación de la tarea a realizar**: Se necesita tener bien en claro que va a hacer nuestra función antes de diseñarla.

2-**Determinación de los parámetros formales**: Cuando ya tenemos en claro que es lo que hay que hacer se debe determinar cómo obtener los datos para realizar la tarea que se debe hacer. Ejemplo, si la función lee algo del teclado no es necesario pasarle datos con lo cual el parámetro formal será **VOID**. En cambio si la función se dedica a realizar alguna búsqueda podemos pensar 2 formas: El dato a buscar se lo paso a la función o lo lee la función. De acuerdo al origen de los datos los parámetros formales cambian en cantidad y tipo. La respuesta a cuál de las opciones a elegir la da el sentido común y el análisis de cual de las 2 formas es mas genérica. Entonces en este momento se decide de acuerdo al origen de los datos cuales van a ser los parámetros formales de la función.

3-**Determinación del valor a retornar**: Finalmente resta definir cual será el tipo de dato que retorna la función. Al igual que en el punto anterior el valor a retornar va íntimamente relacionado con la acción que haga la función.

Recursividad

Se denomina Recursividad al proceso de definir algo en terminos de si mismo. En otras palabras es la forma en la cual se especifica un proceso basado en su propia definición, la construcción a partir de un mismo tipo.

En C una función recursiva hace referencia a una función **que se llama a si misma** consiguiendo su objetivo dependiendo sólo de si misma.

Ejemplo

```c
void ImprimirNumerosHastaElCero(int numero);

int main()
{
    ImprimirNumerosHastaElCero(5);

    return 0;
}

//Función recursiva a la que le paso un número
//y me imprime desde ese número hasta el cero.
void ImprimirNumerosHastaElCero(int numero)
{
    if(numero != -1)
    {
        printf("El numero es %d\n",numero);
        numero--;
        ImprimirNumerosHastaElCero(numero);
    }
}
```

```
El numero es 5
El numero es 4
El numero es 3
El numero es 2
El numero es 1
El numero es 0
```

Este es uno de los ejemplos que podremos tener como función recursiva, lo que hace está función es imprimirme todos los números desde el número que le pase al parametro actual hasta el cero.

Está función preguntara si mientras el parametro formal (numero) sea distinto de -1, muestra, decrementa y repite el proceso **llamando a la misma función que se está ejecutando** y ahí está la **Recursividad.**

Uso de Función Recursiva

El uso de funciones recursivas no es recomendado y vamos a tratar de evitarlas a toda costa. La única razón para usarlas es si no me queda otra opción (cosa que es muy poco probable, ya que siempre hay una forma alternativa de hacer algo), la razón de que no las usamos es que cómo se llama a si misma muchas veces no es muy óptima que digamos, está función hace 7 llamados, incluido cuando numero vale -1, mientras que si lo hago con un for o while, sólo haria 6 iteraciones.

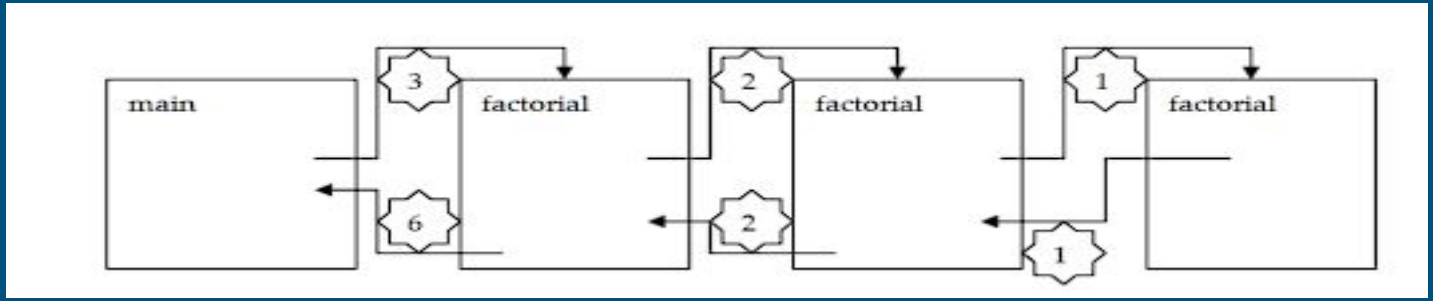
Otro Ejemplo

```c
int CalcularFactorial(int numero);

int main()
{
    printf("%d ",CalcularFactorial(5));

    return 0;
}

int CalcularFactorial(int numero)
{

    if (numero < 2)
    {
        return 1;
    }

    return numero * CalcularFactorial(numero - 1);
}
```

Un ejemplo más común de recursividad es en el uso de la función factorial.

Está función va a hacer su recursividad si el número que yo ingreso es menor a 2, cada vez que ingrese un número menor a 2 está función va a retornar el número multiplicado por la misma función pasandole el número -1, hasta que número valga 1 y la recursividad termine.

Otro Ejemplo



Imaginemos que le paso un 3 a la función que le mostre de ejemplo anteriormente, yo voy a llamarlo por primera vez (numero = 3), hace la recursividad, repito el proceso (numero = 2), repito el proceso (numero = 1), esas tres llamadas, deben devolver algo en el main, por ende como se muestra en la imagen el resultado en el main viaja desde el último retorno hasta el resultado final (va y vuelve)

Todas las funciones recursivas deben tener una condición que permita el comienzo del retorno, si esto no ocurre se puede producir un desbordamiento de la pila

El programa hizo 6 acciones al microprocesador, siendo muy poco eficiente.

Forma más óptima

```
int CalcularFactorial(int numero)
{
    int i;
    int retorno;

    retorno = 1;

    for(i=1; i<numero + 1; i++)
    {
        retorno *= i;
    }

    return retorno;
}
```

En vez de realizar una función recursiva yo puedo calcular el factorial simplemente con un for, la función se va a llamar una sóla vez y por ende va a ser mucho más eficiente.

Tipos de Recursividad

Existen cuatro tipos de recursividad, si bien no vamos a entrar en profundidad en ninguna ni tampoco de su obligación saberlas las mencionamos.

Recursividad simple: Aquella en cuya definición sólo aparece una llamada recursiva. Se puede cambiar a una función con ciclo iterativo.

Recursividad múltiple: Se da cuando hay más de una llamada a sí misma dentro del cuerpo de la función

Tipos de Recursividad

Recursividad anidada: En algunos de los argumentos de la llamada recursiva hay una nueva llamada a sí misma.

Recursividad cruzada o indirecta: Son algoritmos donde una función provoca una llamada a sí misma de forma indirecta, a través de otras funciones.

Bibliotecas

Todas aquellas funciones que hemos desarrollado y creemos que son óptimas y eficientes para usar en otros programas las podemos incluir en una biblioteca propia. La idea de las funciones, es poder usarlas en muchos programas independizándonos del código que ejecutan. Cada vez que se use una función no se debe volver a escribir el código, solamente se la llama. Para esto la idea es dividir nuestras funciones en archivos diferentes, dependiendo de que segmento de nuestro programa pertenezcan.

Un ejemplo podría ser un archivo Funciones.c

Bibliotecas

```
int Sumar(int numero1,int numero2)
{
    int resultado;

    resultado = numero1 + numero2;

    return resultado;
}
```

En el archivo Funciones.c sólo vamos a poner el desarrollo de las funciones (el código que va a ejecutar).

Bibliotecas

```
/** \brief Función que suma dos enteros y retorna el resultado de su suma
 *
 * \param numero1: Primero número que ingreso
 * \param numero2: Segundo número que ingreso
 * \return retorna el resultado de la suma
 *
 */
int Sumar(int numero1,int numero2);
```

Dentro de mi .h voy a tener los prototipos de mis funciones, acompañado de los comentarios que me definen que hace la función, que parametros recibe (si es que recibe alguno) y cual retorna (si es que retorna uno), comentar las funciones es una buena práctica para que otro programador sepa que hace dicha función.

Bibliotecas

En ninguno de los dos archivos, aparece el main. Es simplemente escribir un programa que no tiene main y en el cual se escriben todas las funciones que el programador ha desarrollado. Este archivo solamente se compila y se verifica que no contenga errores de compilación, se supone que no los debe tener ya que se han copiado las funciones de programas en los cuales estaban funcionando. Para poder incluir estas funciones en un programa solo se necesita agregar en el programa un include como el siguiente:

#include "Funciones.h"

Se usan las “ ” para diferenciarlas de las bibliotecas por defecto del sistema, en este archivo .h también ubicariamos el desarrollo de las estructuras de nuestro programa (Tema que van a ver en próximas clases)