Funciones

Funciones

A medida que los programas crecen en extensión y complejidad la resolución se torna más complicada y su depuración y modificaciones futuras resultan casi imposibles. Para resolver este tipo de problemas lo que se hace es dividir el programa en módulos más pequeños que cumplan una tarea simple y bien acotada llamados funciones.

Funciones

Todos los programas en C están formados por una función especial y única llamada **main()** que es la que se ejecuta cuando yo compilo y ejecuto el programa, en está función hasta ahora nosotros desarrollamos nuestro código, pero los programas de acá en adelante van a ir creciendo en tamaño y desarrollar todo en el main seria practicamente imposible.

Por ende con las funciones nosotros vamos a dividir cada segmento de código, cada funcionalidad que desarrollemos en especifico será dividido en ellas para que aparte de que tengamos un código más limpio evitariamos repetir código innecesario y además tendremos todo mucho más organizado.

Funciones

Con las funciones logramos.
1. Que el programa sea más simple de comprender ya que cada función se dedica a realizar una tarea en particular.

2. La depuración queda acotada a cada función.

3. Las modificaciones al programa se reducen a modificar determinados funciones.

4. Cuando cada función está bien probada, se la puede usar las veces que se quiera y así reutilizar código.

5. Se obtiene una independencia del código (cada función es independiente de otra)

Funciones

scanf, printf, fflush, gets, etc son una de las cuantas funciones pertenecientes al sistema, el código que se ejecuta cuando se llaman a dichas funciones está dentro de las bibliotecas que vienen con el compilador, nosotros vamos a aprender a como crear nuestras propias funciones de cero y posteriormente en la próxima clase cómo crear nuestras propias bibliotecas con sus respectivas funciones

Declaración de una Función

Una función se la declara de la siguiente forma.

tipo devuelto NombreFuncion (tipoVariable1, tipoVariable2, ……., tipoVariable N)

-Donde el tipo devuelto es el valor que la función va a devolver cuando se ejecute, por ejemplo yo puedo declarar en el main una variable suma por ejemplo, y a esa variable darle como valor la llamada a dicha función cuando está acabe esa variable valdra lo que la función me retorne (vamos a dar ejemplos para que quede más claro)

Declaración de una Función

-El nombre de la función es el nombre que yo le voy a dar a la función por lo general como buena práctica los nombres de las funciones deben ser verbos (Generar, Imprimir, Mostrar, Listar, Sumar, Restar, etc), para diferenciarlos de las variables su declaración es por UpperCamelCase o sea por ejemplo (ListarUsuarios, CrearPersona, GenerarComprobantes).

-Luego están las variables que se le pasa a la función, estás variables son valores que recibe la función y son locales a ella, yo por ejemplo puedo pasarles valores al llamar a dicha función pero cuando los recibe la función son locales a ella misma y viven y mueren en dicha función (Más adelante se explica mejor)

Declaración de una Función

Acá un pequeño ejemplo de la declaración de una función.

```
int Restar(int numero1,int numero2)
{
    int resta;

    resta = numero1 - numero2;

    return resta;
}
```

Funciones

Cuando declaro una función yo debo indicarle el prototipo este es nada más ni nada menos que la declaración de la función pero sin las llaves y con un ; al final.

Los prototipos se ponen antes del main y después de las bibliotecas, y las declaraciones de las funciones luego de que termine el main, por ahora, luego las funciones las vamos a dividir en bibliotecas para una mejor organización.

Funciones

Existen 3 instancias donde se hace mención a las funciones:

1-Prototipos: En este lugar se declaran todas aquellas funciones propias que se van a utilizar durante el programa. La forma de declararlas es como se explicó al comienzo, teniendo en cuenta que cada declaración de función termina con punto y coma (;) .

Los prototipos son necesarios para que el compilador verifique que sean coherentes los tipos de datos declarados en el prototipo contra los que realmente se usan en la llamada a la función.

2-Llamada a la función: Cuando se llama a la función es para usarla. Desde cualquier parte de main o desde otra función se puede hacer la llamada. En el momento en que se produce la llamada a la función, se produce un salto en el flujo del programa. En ese momento se pasa a ejecutar el código de la función que va a terminar de ejecutarse al encontrar la sentencia return o llegar a la llave que cierre la función. Cuando la función finaliza, se sigue ejecutando el código de la función que produjo la llamada.

3-Desarrollo de las funciones: Esta es la parte en la cuál se escribe el código de la función.

Ejemplo

```c
#include <stdio.h>

int Resta(int numero1, int numero2);//Declaración de Prototipos

int main()//Función Main
{
    int resta;

    resta = Restar(5 , 2);//Llamada a la función

    printf("La resta es %d",resta);

    return 0;
}

int Restar(int numero1,int numero2)//Desarrollo de la función
{
    int resta;

    resta = numero1 - numero2;

    return resta;
}
```

Funciones

Yo puedo tener funciones que no retornen nada (void), puedo tener funciones que no reciban nada, puedo tener funciones que reciban y retornen algo y también incluso puedo tener funciones que no reciban ni retornen nada (estás no sé presentan casi nunca pero son posibles).

La idea es que una función siempre retorne algo, para hacer que una función no retorne nada se usa el tipo de dato void (vacio).

- int Resta (int numero1, int numero2);
- int Resta ();
- void Resta (int numero1, int numero2);
- void Resta ();

Se puede obtener la misma funcionalidad de las cuatro formas.

Funciones

-**Parametros Formales**: Son las variables que se le pasan a la función (estos están presentes en el desarrollo de la Función)

-**Parametros Actuales**:Son las variables o datos que recibe la función en la llamada a ella.

Ejemplo

```c
int main()//Función Main
{
    int resta;

    resta = Restar(5 , 2);//5 y 2 son los parametros actuales
                          //pueden ser variables o datos númericos
                          //por lo general se les pasan variables

    printf("La resta es %d",resta);

    return 0;
}

int Restar(int numero1,int numero2)//numero1 y numero2 son los parametros actuales
{
    int resta;

    resta = numero1 - numero2;

    return resta;
}
```

SCOPE

El scope o ámbito es la región en donde las variables existen, en otras palabras es el área en donde nacen y mueren nuestras variables.

Como dijimos antes las funciones son independientes unas de las otras, por ende tienen scopes diferentes, las variables que yo declare en el main y declare en una función que yo creo no se relacionan, incluso si tuvieran el mismo nombre.

Yo puedo tener la variable suma en el main y la variable suma en una función Sumar y no pasaria nada, ya que cada scope es independiente del otro, serian dos variables diferentes, con valores diferentes y distintas direcciones de memoria. El comienzo y fin del scope lo definen las llaves, cuando se abre una llave comienza un scope y cuando cierra termina.

Las variables que están dentro de su scope se conocen como **Variables Locales**.

SCOPE

-Se les dice **locales** ya que son locales a su función, nacen y mueren ahí, no se relacionan con variables de otras funciones aunque tengan el mismo nombre.

-Las variables que están fuera de su **scope**, por ejemplo arriba del **main** donde no la separa ninguna llave se las denomina **variables globales**, y su funcionalidad abarca todo el programa. No se recomienda el uso de variables globales y por reglas de estilo no las vamos a usar.

SCOPE

1-La variable local tiene validez solo dentro de la función en la que fue definida, fuera de ella no existe.

2-Si existe una variable global y otra local con el mismo nombre tiene validez la variable local dentro de esa función

3-Si en la función “a” tenemos la variable “nombre” y en la función “b” existe la variable “nombre” no existe problema de ningún tipo ya que ambas son variables locales de funciones distintas (aunque tengan el mismo nombre)

4-Es conveniente usar variables con distinto nombre para evitar confusiones.

Pasaje por valor y por referencia

-El **pasaje por valor** es cuando en la llamada de la función se pasa como parametro actual el valor de una variable o dato.

-Mientras que el **pasaje por referencia** es cuando en la llamada de la función se pasa como parametro actual la dirección de memoria de una variable (puntero) en este caso la variable original si se modificara (si es que se la modifica) aunque este en diferente scope.

Con el pasaje por referencia yo puedo modificar una variable que cree en el main en el desarrollo de otra función, ya que recibo la dirección de memoria de la variable y no el valor de la variable.