Procedimientos y funciones

- **Creación de funciones**

Competencias

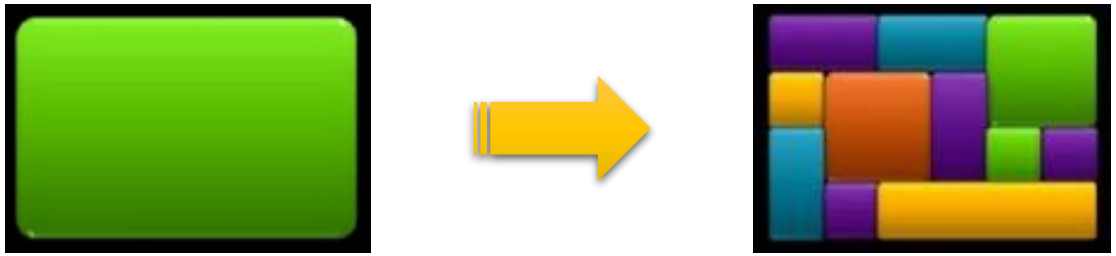
- Define procedimientos dentro del desarrollo de programas
- Construye funciones utilizando parámetros
- Aplica programación orientada a objetos en la solución a problemas

Introducción

Programar puede ser considerado un arte y para lograrlo es necesario buscar las herramientas adecuadas y paradigmas que nos lleven a obtener una solución viable de lo que necesitamos resolver.

Una de los paradigmas de la programación es la modularización que consiste en partir un problema grande en pequeñas partes que son funcionalmente independientes, que encapsulen y operen los datos que se les entreguen.

Por tanto, al solventar las partes pequeñas, tendremos lista una solución al problema grande logrando un solución viable.



Funciones

Utilizar funciones nos permite dividir un programa grande en pequeños segmentos que realizan cada uno, tareas concretas. Por cierto en muchas ocasiones, dentro de un programa se realicen las mismas tareas varias veces, lo que se facilita mediante la utilización de funciones. Sin embargo, es probable que ciertas funciones no sean reutilizables, pero al usarlas se mejora la comprensión del programa.

Funciones

La filosofía en la que se base el diseño de C es el empleo de funciones. Por esta razón, un programa en C contiene al menos una función, la función main . Esta función es particular dado que la ejecución del programa se inicia con las instrucciones contenidas en su interior. Una vez iniciada la ejecución del programa, desde la función main se puede llamar a otras funciones y, posiblemente, desde estas funciones a otras. Otra particularidad de la función main es que se llama directamente desde el sistema operativo y no desde ninguna otra función. De esta manera, un programa en C sólo puede contener una función main.

Con el propósito de permitir un manejo eficiente de los datos, las funciones en C no se pueden anidar. En otras palabras, una función no se puede declarar dentro de otra función, por lo que todas las funciones son globales o externas, lo que hace que puedan llamarse desde cualquier parte de un programa, en otros lenguajes si es posible esta restricción.

Funciones

Se puede acceder (llamar) a una determinada función desde cualquier parte de un programa. Cuando se llama a una función, se ejecutan las instrucciones que constituyen dicha función. Una vez que se ejecutan las instrucciones de la función, se devuelve el control del programa a la siguiente instrucción (si existe) inmediatamente después de la que provocó la llamada a la función.

Cuando se accede a una función desde un determinado punto del programa, se le puede pasar información mediante unos identificadores especiales conocidos como argumentos (también denominados parámetros). Una vez que la función procesa esta información, devuelve un valor mediante la instrucción return.

Estructura de una función

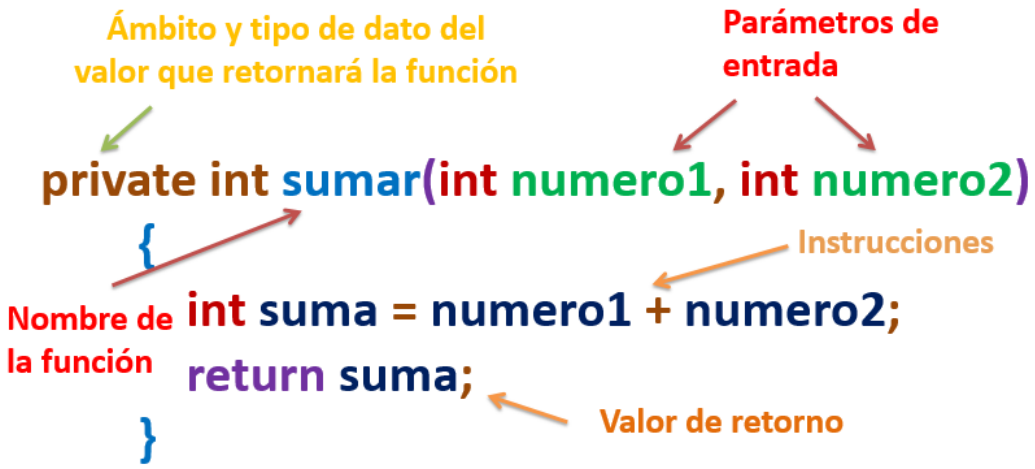
La estructura general de una función en C# es la siguiente:

```
ámbito tipo_de_retorno nombre_de_la_función (lista_de_parámetros)
{
    instrucciones;
    return expresión ;
}
```

Donde:

- ***tipo_de_retorno***: es el tipo del valor devuelto por la función, o, en caso de que la función no devuelva valor alguno, la palabra reservada void.
- ***nombre_de_la_función***: es el identificador o nombre que asignaremos a la función.
- ***lista_de_parámetros***: es la lista de declaración de los parámetros que son enviados a la función. Éstos se separan por comas. (teme en cuenta que pueden existir funciones que no utilicen parámetros).
- ***instrucciones***: compuesto por un conjunto de sentencias que llevan a cabo la tarea específica para la cual ha sido creada la función.
- ***return expresión***: al escribir return, se devuelve el valor de la función, en este caso representado por expresión.

Estructura de una función



Parámetros de una función

Dentro de un programa en c#, existe al menos una función y es la función main, después de esta pueden crearse otras funciones, pero fuera del ámbito de main, como se ve en la siguiente seudocódigo:

```
1   Proceso main
2       Escribir  "Ingrese un dato";
3       Leer dato;
4       imprimirDato(dato);
5   FinProceso
6
7   SubProceso imprimirDato ( dato )
8       Escribir dato;
9   FinSubProceso
```

Puede observarse que la función main únicamente solicita un dato, posteriormente llama a la función imprimirDato en la línea 4, pasando un parámetro a la función imprimirDato, función completa el trabajo.

Este seria el caso de dos funciones main e imprimirDato, donde imprimir dato no retorna nada, pero requiere un parámetro obligatorio para funcionar.

Observe que que la variable ***dato*** es una variable local para el ámbito de la función main, no puede ser accedida o llamada desde la función imprimir dato.

Parámetros de una función

Para el caso del seudocódigo siguiente, la función main no realiza mas que llamar a la función imprimirDato, es esta función la que realiza la tarea de solicitar el dato e imprimirlo, la función no posee parámetros obligatorios y no retorna datos de respuesta

```
1  Proceso main
2      imprimirDato();
3  FinProceso
4
5  SubProceso imprimirDato ( )
6      Escribir  "Ingrese un dato";
7      Leer dato;
8      Escribir dato;
9  FinSubProceso
```

Recuerde que C no permite funciones dentro de funciones, es programación estructurada y secuencial, es decir se ejecuta la función imprimirDato y se regresa al punto de donde fue llamada.

Observe que que la variable ***dato*** es una variable local para el ámbito de la función imprimirDato, no puede ser accedida o llamada desde la función main.

Parámetros de una función y retorno de resultado

Considere el siguientes caso:
Se solicita la sumatoria de dos números, que serán solicitados en la función main, esta pasara los dos parámetros a una función suma, y esta retorna el resultado del calculo para ser impresa en pantalla.

```
1   Proceso main
2       Escribir  "Ingrese un numero 1";
3       Leer numero1;
4       Escribir  "Ingrese un numero 2";
5       Leer numero2;
6       suma <- imprimirDato(numero1, numero2);
7       Escribir "La suma es: " , suma;
8   FinProceso
9
10  SubProceso resultado <- imprimirDato (numero1, numero2 )
11      resultado <- numero1 + numero2;
12  FinSubProceso
```

El seudocódigo muestra que la función main solicita dos números y los pasa a la función suma, esta realiza el calculo y retorna el resultado para ser impreso.

Recuerde que puede haber n funciones en un programa, pero no funciones anidadas entre si.

Parámetros de una función y retorno de resultado

Considere el siguientes caso:
Se solicita la sumatoria de dos números, que serán solicitados en la función main, esta pasara los dos parámetros a una función suma, y esta retorna el resultado del calculo para ser impresa en pantalla.

```
1   Proceso main
2       Escribir  "Ingrese un numero 1";
3       Leer numero1;
4       Escribir  "Ingrese un numero 2";
5       Leer numero2;
6       resultado <- suma(numero1, numero2);
7       Escribir "La suma es: " , resultado;
8   FinProceso
9
10  SubProceso resultado <- suma (numero1, numero2 )
11      resultado <- numero1 + numero2;
12  FinSubProceso
13
```

El seudocódigo muestra que la función main solicita dos números y los pasa a la función suma, esta realiza el calculo y retorna el resultado para ser impreso.

Observe las variables ***resultado*** en la línea 6 y 11, aunque se llaman igual no son las mismas por encontrarse en ámbitos diferentes.

Recuerde que puede haber n funciones en un programa, pero no funciones anidadas entre si.

Entonces los ámbitos de las variables.

Debe tener bien presente que en C es una programación estructurada y secuencial, para que una función sea reconocida debe estar creada antes del la función main (o quien la llame), o en su defecto declarar la función antes de ser llamada si esta no ha sido recorrida por el compilador (después de la función main).

```c
#include <stdio.h>

 // Sumatoria de dos numeros usando funciones

 int sumatoria (int numero1, int numero2){
    return numero1 + numero2;
}

int main (){
    // Evaluacion de dos numero enteros diferentes
    printf("Sumatoria de dos numeros \n");
    int numero1, numero2;

    printf("Ingrese el primer numero \n");
    scanf("%d",&numero1);

    printf("Ingrese el segundo numero \n");
    scanf("%d",&numero2);

    printf("\n La sumatoria es: %d " , sumatoria(numero1, numero2) );

    return 0;
}
```

```c
#include <stdio.h>

 // Sumatoria de dos numeros usando funciones

int main (){
    // Evaluacion de dos numero enteros diferentes
    printf("Sumatoria de dos numeros \n");
    int numero1, numero2;

    printf("Ingrese el primer numero \n");
    scanf("%d",&numero1);

    printf("Ingrese el segundo numero \n");
    scanf("%d",&numero2);

    int sumatoria(int numero1, int numero2); // declaracion
    printf("\n La sumatoria es: %d " , sumatoria(numero1, numero2) );

    return 0;
}

 int sumatoria (int numero1, int numero2){
    return numero1 + numero2;
}
```

A la izquierda la función antes de main, no se declara la función dentro de main.
A la derecha, la función suma después de main, requiere declarar esta dentro de main (línea 16)

Entonces los ámbitos de las variables.

```
27  // variable global
28  int numero1;
29  int numero2;
30
31  main (){
32      // Evaluacion de dos numero enteros diferentes
33
34      printf("Ingrese el primer numero \n");
35      scanf("%d",&numero1);
36
37      printf("Ingrese el segundo numero \n");
38      scanf("%d",&numero2);
39
40      int numero3; // variable local a main
41      int sumatoria(int numero1, int numero2); // declaracion
42      printf("\n La sumatoria es: %d " , sumatoria(numero1, numero2) );
43
44      return 0;
45  }
46
47   int sumatoria (int numero1, int numero2){
48      int numero4; // variable local a sumatoria
49      return numero1 + numero2;
50  }
```

Observe que que para este caso las variables numero1 y numero2 hay sido declaradas de forma global, y para eso se han declarado fuera de la función main, lo que implica que estarán disponibles desde cualquiera parte del programa, e incluso desde otras funciones.

En la línea 40 se declara la variable numero3 accesible solo dentro de la función main.

En la línea 48 se declara la variable numero4 solo disponible para la función sumatoria.

Usando dos funciones y variables globales.

Definición de variables y funciones de manera global.

```
64   int numero=100;
65   void funcion1();
66   void funcion2();
67
68   int main (){
69
70       funcion1();
71       funcion2();
72
73       return 0;
74   }
75
76    void funcion1 (){
77
78       printf("\n Numero impreso desde la funcion 1: %d " , numero );
79   }
80
81    void funcion2 (){
82
83       printf("\n Numero impreso desde la funcion 2: %d " , numero );
84   }
```

Se observa en las línea 64 la definición de la variable global numero.

Se definen de manera global las funciones funcion1 y funcion2.

Luego dentro de las funciones solo se toma el valor de numero y se imprime dado que se ha definido de manera global.

Main solo llama a las dos funciones.

Usando dos funciones y variables globales.

Definición de variables y funciones de manera global.

```
64   int numero=100;
65   void funcion1();
66   void funcion2();
67
68   int main (){
69
70       funcion1();
71       funcion2();
72
73       return 0;
74   }
75
76    void funcion1 (){
77
78       printf("\n Numero impreso desde la funcion 1: %d " , numero );
79   }
80
81    void funcion2 (){
82
83       printf("\n Numero impreso desde la funcion 2: %d " , numero );
84   }
```

Se observa en las línea 64 la definición de la variable global numero.

Se definen de manera global las funciones funcion1 y funcion2.

Luego dentro de las funciones solo se toma el valor de numero y se imprime dado que se ha definido de manera global.

Main solo llama a las dos funciones.

Funciones y otras estructuras de control

Considere el calculo del cubo los primero 5 números (1 al 5)

```
92   int cubo(int numero);
93
94   int main (){
95
96       int i;
97       for (i=1; i<=5; i++){
98           printf("El cubo del numero %d es %d\n", i, cubo(i));
99       }
100      return 0;
101  }
102
103   int cubo (int numero){
104
105      int potencia;
106      potencia = numero * numero * numero;
107      return potencia;
108  }
```

Se define la función de manera global la cual calcula la potencia de los números que se sean pasado por su parámetro.

La función main recorre los primeros 5 números por medio de un for e imprime las potencia de cada uno.

Salida en consola

```
El cubo del numero 1 es 1
El cubo del numero 2 es 8
El cubo del numero 3 es 27
El cubo del numero 4 es 64
El cubo del numero 5 es 125

--------------------------------
```

Ejercicios 2-U2

Completar los siguientes ejercicios:

1. Crear un programa que solicite al usuario que tipo de calculo desea realizar, entre las siguientes:
    1. Calcular el seno de un numero
    2. Calcular la tangente de un numero.
    3. Área de un triangulo.
    4. Área de un cubo.
    5. Área de un circulo.

2. El programa debe solicitar los datos necesarios e imprimir el resultado según lo que haya elegido el usuario.
3. El programa debe preguntar si se desea realizar otro calculo o terminar el programa.
4. Cada calculo del ítems 1 debe ser una función.

Pueda que necesite esta librería: math.h

“El ordenador nació para resolver problemas que antes no existían”
– Bill Gates

Gracias