Curso C

Funciones

Programación modular

- Programación modular
  - División de problemas en tareas sencillas.
- Función:
  - “Subalgoritmo”que realiza una tarea concreta en base a unos datos que se le suministran (argumentos o parámetros) pudiendo devolver o no un resultado.

Programación modular

- Ventajas de utilizar funciones :
  - Mayor claridad en los fuentes
  - Evita repeticiones innecesarias de código
  - Clarifica el flujo de control del programa
  - Abstrae operaciones complejas
  - Facilita modificaciones posteriores
  - Facilita depuración de errores
  - Aumenta la productividad del programador. Puede confeccionar su propia biblioteca
  - Mayor portabilidad

Def. Funcion

```
tipo_de_variable nombre_de_la_función( argumentos )
      {
      definición de variables;

      cuerpo de la función;

      return 0;
      }
```

Argumentos:

• Lista de identificadores de variables precedidos por sus tipos y separados por comas.

Si no posee parámetros, colocaremos (void)

return (valor)

• Devuelve el valor de la función (del tipo indicado). Si la función no devuelve ningún valor, la definiremos con tipo void:

void miFuncion (int p)

La llamada a nombreFunción se puede realizar desde cualquier parte del código, incluso desde dentro de la propia definición de la función (funciones recursivas).

Funciones

- Declaración Función:
  - TipoDato NombreFuncion (Argumentos);
- Definición
  - Incluye el contenido de la funcion
- Generalmente, declaraciones de funciones en ficheros de cabecera .h, mientras que definiciones en ficheros fuente .c/.cpp
  - Includes de cabeceras

Paso de Parámetros

- Por Valor
  - La función utiliza los argumentos para obtener determinados valores con los que realizar su trabajo.
  - Podemos utilizar como argumentos:

Identificadores de variables	fejem plo(x,s)
Llamadas a otras funciones	fejemplo(mayor(a,b),s)
Expresiones aritméticas y lógicas	fejemplo(x,(y+x))
Constantes	fejemplo(3,true,y)

Paso de parámetros

- Por Valor:
  - Los parámetros definidos en cada función se comportan como variables dentro de élla misma. Sin embargo, los valores de las variables se copian en otro lugar de la memoria y es con esta copia con la que se trabaja.
  - Cuando finaliza la ejecución de la función, las variables toman su valor previo a la llamada.

Paso de parámetros

- Por Referencia:
  - Se utiliza cuando la función necesita devolver más de un valor al finalizar su ejecución.
  - Los argumentos que enviamos a la función son las direcciones de memoria donde se encuentran las variables, de forma que cualquier modificación dentro de la función se refleja fuera de ella

Paso de parámetros

- Por Referencia:
  - DECLARACION
    - tipo nombreFuncion (tipo *nombreVariable);
  - LLAMADA
    - nombreFuncion (&nombreVariable);
- Operador & ->dirección de la variable
- Operador * ->contenido de la dirección de memoria que apunta la variable

- Explicar dif. Variable Local y GLOBAL

- Calificación funciones: Funciones static/extern por defecto (visibles en todos los ficheros)
  - static void Mifunc(void);

Recursividad

- Funciones recursivas.
  - Aquellas que se llaman a si mismas.
  - Casos base finalizacion recursividad
  - Pila de llamadas (memoria)

Practicas

- Visitar la siguiente pagina Web:
  - Sobre cómo NO realizar una práctica
  - http://www.di.uniovi.es/~cernuda/noprog.html