Guía rápida de programación en C / C++

Decisión: if

Ejecuta una acción sólo si se cumple una condición:

```
if (condición)
    Acción;
else if (otra cond.)
    otra acción;
...
else if (otra cond. Más)
    Otra acción más;
...
else
    acción eliminatoria
```

1ª cond. Si (cond)

2ª cond. Si no si (cond)

Más condiciones si es necesario

Última cond. En todos los demás casos...

Decisión: operador ternario

Establece el valor de una variable en función de una condición

C = (cond)?a:b;

si (cond)
  c=a
si no
  c=b

if (cond)
  c=a;
else
  c=b;

Decisión: switch

Ejecuta una tarea dependiendo de los posibles valores de una variable:

```
switch (variable)
{
    case valor1 : accion1;
                  break;
    case valor2 : accion2;
                  break;
    (podemos incluir muchos casos)
    default:
       accion por defecto;
}
```

Datos: Arrays

Crea una colección de variables de un mismo tipo

Declaración:
`Tipo nombre[tam];`

Acceso:
`nombre[pos];`

Datos: Tipos de variable

Tipo	Tam	Rango	Uso
char unsigned **char**	8	-128 .. 127 0 .. 255	Caracteres o números muy pequeños.
short / *short* **int** *unsigned short*	16	-32,768 .. 32,767 0 .. 65,535	Números pequeños, contadores, etc
int unsigned **int**	32	-2,147,483,648 .. 2,147,483,647 0 .. 4,294,967,295	Números en general, fechas, contadores
long long **int** *unsigned long long*	64	-2^{63} .. $2^{63}-1$ 0 .. 2^{64}	Números grandes o muy grandes
float	32	1.10×10^{38} .. 3.40×10^{38}	Núms. con decimales
double	64	2.23×10^{308} .. 1.79×10^{308}	Núms. con decimales de precisión
long **double**	80	3.37×10^{4932} .. 1.18×10^{4932}	Núms. con decimales (Alta precisión)

Bucles: for

Partiendo de un valor inicial, repetiremos una acción o varias mientras no lleguemos a un valor final:

Valor inicial | Incremento o cambio | Valor final

```
for (i=0; i < 100; i++)
{
  acciones;
}
```

Si queremos, en lugar de usar valor final, podemos especificar una condición de continuación

Útil cuando sabemos cuántas veces vamos a repetir o iterar

Bucles: while

Repetiremos acciones mientras se cumpla una condición o condiciones.

```
while (condicion)
{
  acciones;
}
```

Siempre comprobamos la condición antes de iterar.

Bucles: do...while

Repetiremos acciones mientras se cumpla una condición o condiciones.

```
do
{
   accciones;
} while (condicion);
```

Comprobamos la condición después de cada iteración.

Datos: Registros

Son estructuras que agrupan varias variables dentro, del tipo que sean, creando un nuevo tipo de dato.

Declaración:

```
struct nombre_tipo
{
   Tipo1 variable1;
   Tipo2 variable2;
   ...
};
```

Acceso:

```
nombre_tipo grupo;
grupo.variable1
```

Estructura: Funciones

Nos permiten utilizar subprogramas que ya hemos hecho, y así reutilizar código. Nos facilitan tareas complejas dividiéndolas en tareas más sencillas.

Devolvemos un valor. Cuidado con el tipo

```
tipo nombre (T1 arg1, T2 arg2, … )
{
  tipo res;
  Acciones;
  return res;
}
```

Existe el tipo especial *void*, que no devuelve nada. No necesitamos **return**.

Los argumentos son datos necesarios para completar la tarea que tiene que realizarse. T1, T2, … son los tipos de los argumentos Arg1, arg, son nombres que asignamos a los argumentos, y sólo son válidos dentro de la función.

Una vez tengamos la función hecha, tenemos que pensar en ella como una caja negra, a la que le pasamos unos datos, realiza una tarea y nos devuelve otros datos.

Referencia básica de C / C++
http://totaki.com/poesiabinaria
http://totaki.com/gaspy

gaspy@totaki.com