De C a C++
MODIFICACIONES MENORES

- **Cambio de extensión.** *.c → *.cpp

- **Comentarios.**
```
/* comentario */

// comentario
```

- **Declaración simplificada de variables tipo enum, struct y union.**
  Una vez definido un tipo enum, struct o union, no es necesario anteponer la palabra clave para declarar variables de ese tipo.

- **Flexibilidad en la declaración de variables.**
  En C++, las variables pueden ser declaradas en cualquier lugar de un bloque.

- **Variables tipo boolean.**
  C++ dispone de tipo bool. Las variables de este tipo permiten almacenar uno de los dos valores lógicos: verdadero o falso (formalmente habrá que poner 1, 0, true ó false).

- **Visibilidad de las variables.**
  Las reglas de duración y visibilidad de variables en C++ son similares a c, con la novedad de lo referente a clases (OOP).

- **Especificador const para variables.**
  En C++, una variable const puede ser utilizada para definir el tamaño de un array.
  En C++, el valor de una variable const no se puede modificar a través de un puntero (en C sí).

- **Especificador const para punteros.**
  C++ admite const para punteros. Un puntero const siempre apuntará a la misma dirección de memoria (pero lo que haya en esa dirección sí puede variar).

- **Conversiones explícitas de tipo (casting).** C → `(tipo)variable`
  (variable puede ser una expresión) C++ añade `tipo(variable)`

- **Especificador inline para funciones.**
  Hace que el código de una función sustituya a cada una de sus llamadas en el código del programa. Es una directiva al compilador, no lo fuerza.
  Dos formas de especificarlo:
  - Anteponiendo 'inline' a la declaración de la función.
  - Introduciendo el código de la función en la declaración (será ya definición tb), poniéndolo entre llaves a continuación de la misma.

- **Sobrecarga de funciones (overload).**
  Consiste en declarar y definir funciones distintas pero con un mismo nombre. Habrán de diferir forzosamente en el nº de parámetros y/o en el tipo de los mismos.

- **Valores por defecto para los parámetros de una función.**
  En C++ se pueden definir valores por defecto para todos o algunos de los argumentos de una función. Si en una llamada a la misma se omite algún argumento, se recurre a su valor por defecto. Reglas:
  - Los argumentos con valores por defecto han de situarse al final de la lista de argumentos.
  - Si en una llamada se omite un argumento, han de omitirse todos los de su derecha.

- **Variables de tipo referencia.**
  En C++ existe un tipo llamado referencia.
  Las variables de tipo referencia se definen con el operador '&', precedido del tipo de variable al que va a realizar la referencia, y deben ser inicializadas en el momento de su declaración:

```
tipo& var = valor;
```

  (var es ahora un alias de valor: son dos nombres distintos para la misma dirección de memoria)

- **Operadores new y delete para gestión dinámica de memoria.**
  Sustituyen a las funciones malloc() y free(). Permiten reservar y liberar memoria de forma dinámica. Una variable creada con new perdura hasta ser explícitamente borrada con delete (puede traspasar la frontera de su bloque), o hasta que finalice el programa, claro.

```
Puntero = new tipo;          Puntero = new tipo[];
delete Puntero;              delete[] Puntero;
```

- **Entrada y salida por consola.**
  Salida por monitor: `cout<<expresión;`
  Entrada por teclado: `cin>>variable;`

  En los flujos no sólo se pueden insertar datos, sino también **manipuladores** (un tipo especial de objetos). Algunos manipuladores más utilizados son:

endl	(avance de línea, vacía el buffer de salida, lo vuelca)
ends	(carácter nulo)
dec	(salida de números en base 10) (opción por defecto)
oct	(" " " " " 8)
hex	(" " " " " 16)
width(n)	(anchura n)

  El efecto de dec, oct y hex perdura hasta que es revocado por otro manipulador.

- **Otras diferencias con C.**

  - Posibilidad de usar clases para controlar los flujos de E/S de disco de una manera más eficiente.

  - Uso de plantillas (permite definir funciones y clases sin tener que especificar el tipo de todos o algunos de sus argumentos o miembros).

  - Manejo de excepciones (permite controlar de forma adecuada los errores de ejecución, etc.)

José Carlos Cruz Parra
josecarlos@programadorphpfreelance.com
www.programadorphpfreelance.com