Introducción, características y sintaxis en C

Antonio Espín Herranz

Introducción al desarrollo de aplicaciones en C

- Características de C.
- Programación Estructurada.
- Nacimiento y características del lenguaje C.
- Estructuras básicas de programación en C.
- Elementos básicos de un programa en C.
- Conceptos fundamentales en C:
  - Tipos de datos, variables, tipos y declaración de variables, constantes, alcance de las variables.
- Utilización del compilador de C.
- Ficheros en un proyecto en C.

Características de C

- C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores.

- No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación.

- Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje.

Programación estructurada

- La programación estructurada es una forma de escribir programas de forma clara, para ello utiliza únicamente tres estructuras:
  - **Secuencial**: Las instrucciones se ejecutan de forma secuencial una detrás de otra.
  - **Selectiva**: Permite la realización de una instrucción u otra según un criterio, solo una de estas instrucciones se ejecutará.
  - **Iterativa**: Una secuencia de instrucciones, hace que se repitan mientras se cumpla una condición, en un principio el número de iteraciones no tiene porque estar determinado.
- No permitiéndose el uso de la instrucción o instrucciones de transferencia incondicional (GOTO).

Nacimiento del Lenguaje C

- C es un lenguaje de programación diseñado por Dennis Ritchie, de los Laboratorios Bell, y se instalo en un PDP-11 en 1972.

- Se diseño para ser el lenguaje de los Sistemas Operativos UNIX. A su vez, UNIX es un Sistema Operativo desarrollado por Ken Thompson, quien utilizo el lenguaje ensamblador y un lenguaje llamado B para producir las versiones originales de UNIX, en 1970.

- C se invento para superar las limitaciones de B. C es un lenguaje maduro de propósitos generales que se desarrollo a partir de estas raíces.

Mas características del Lenguaje C

- C trabaja con tipos de datos que son directamente tratables por el hardware.
- Estos tipos son los caracteres, números y direcciones. Pueden ser manipulados por las operaciones aritméticas.
- No proporciona mecanismos para tratar tipos de datos que no sean los básicos, debiendo ser el programador el que los desarrolle.
- El código generado es muy eficiente.
- No proporciona otros mecanismos de almacenamiento de datos que no sea el estático y no proporciona mecanismos de entrada ni salida.
- Ello permite que el lenguaje sea reducido y los compiladores de fácil implementación en distintos sistemas.
- Estas carencias se compensan mediante la inclusión de funciones de librería para realizar todas estas tareas, que normalmente dependen del sistema operativo.
- Es un código compilado y se genera código ejecutable.

Estructuras básicas de programación en C

- Un programa escrito en C es:
- Conjunto de variables y funciones externas.
- En la definición de una variable se especifica el tipo y el nombre.
- En la definición de una función distinguimos entre:
  - cabecera: nombre de la función, parámetros con su tipo (si es que los tiene) y el tipo que devuelve la función.
  - cuerpo: declaración de variables internas a la función y un conjunto de sentencias.

Elementos básicos de un programa en C

- Identificadores: da nombres a las variables y las funciones. Letras, dígitos y _.
- Se hace distinción entre mayúsculas y minúsculas.
- Palabras claves, definidas por el lenguaje.
- Operadores que se utilizan para especificar las operaciones a realizar con los datos.
- Elementos sintácticos:
  - **;** terminador de sentencia.
  - { } principio / fin de un bloque o función.
  - ( ) para las funciones y agrupar operaciones.
  - **,** separador de parámetros y declaración de variables.
  - /* */ Comentarios. // Comentario de C++

Tipos de datos

- **void** No ocupa nada.
- **char** 1 byte -127 … 128
- **int** 2 bytes -32768 … 32767
- **float** 4 bytes 3.4 E-107 … 3.4E107
- **double** 8 bytes 1.7E-308 … 1.7E308
- Estos tamaños pueden variar de una máquina a otra, o de un sistema operativo a otro. Utilizar: **sizeof(tipo)**

Modificadores de Tipo

- **long** long int 4 bytes.
  long double 10 bytes.
- **short** Tamaño y rango iguales que int
  depende del número que
  almacenemos se reduce o no a 1 byte.
- **signed** Por defecto, se asume con signo.
- **unsigned**
  - unsigned char 1 byte rango: 0 …255
  - unsigned int 2 bytes rango: 0…65.535

Valores constantes

- char → ‘a’.
- Cadena de caracteres → “a”, “34”.
- Entero decimal → 1.
- Entero octal → 010, comienzan por 0.
- Entero hexadecimal → 0xAAFF.
- Real → 1.0.
- Exponencial → 1.24e7

Conversiones de tipo en C

- **Dentro de una operación** los operandos de menor tipo (tamaño en bytes) se convierten automáticamente al de mayor tipo.

- Ejemplo:

```
int x;
char c;
float f;
(x+1)+c+(f * 5)     // 1) c se pasa a int.
                    // 2) x + 1 se pasa a int.
                    // 3) f * 5 se pasa a float.
                    // 4) int con float se pasa a float.
```

Conversiones de tipo en C

- **Dentro de una asignación**, prevalece el tipo de la parte izquierda de la asignación.

- Ejemplo:

```
int x;
char c;
float f;
f = x + c;  // x + c se convierte a int.
            // f = x + c se convierte a float.
```

Conversiones de tipo en C

- **Conversión explícita CASTING.**
- Formato: (tipo destino)variable.
- Ejemplo:

```
int x = 2;

x / 2; // → 1    (float)x / 2; // → 1.0
```

Conversiones de tipo en C

- **En la llamada a una función**: con cualquier tipo de dato, estos tipos quedan modificados con los tipos predefinidos para los parámetros formales.
- Ejemplo:

  int suma(int a, int b);

  // llamada: suma(1.2, 3.4); → recibe: 1 y 3

  // llamada: suma(2, 3); → recibe 2, 3

Secuencias de escape

- Son constantes de tipo carácter:
  - ‘\n’→ Salto de línea con alimentación.
  - ‘\r’ → Salto de línea.
  - ‘\número’ → ‘\96’ → ‘a’
  - ‘\t’ → Tabulador horizontal.
  - ‘\v’ → Tabulador vertical.
  - ‘\”’ → Mete una comilla doble.
  - ‘\’’ → Mete una comilla simple.
  - ‘\\’ → Mete la barra.
  - ‘\f’ → Alimentación de la hoja en la impresora.
  - ‘\0’ → Nulo. Fin de una cadena.

Variables

- Definición:
  - Nombre de tipo identificador.
- Uso de letras, números y _.
- Que no empiecen por número y no superen los 256 char.

Modo de almacenamiento de variables

- **auto**: Todas las variables que se crean en el momento y se destruyen cuando se dejan de utilizar.
  - Comportamiento habitual, no es necesario especificarlo.

- **register**: Le indica al compilador que intente alojar el máximo tiempo posible la variable en los registros del ordenador.

- **static**: Se crean en el ámbito al que pertenecen y solo se inicializan la primera vez. No se destruyen conservan el valor anterior.
  - ***Este comportamiento incluso se mantiene si nos encontramos dentro de una función.***

- **extern**: Variables externas. Cuando usamos una variable que está en otro módulo del programa.
  - ***Se utiliza para acceder a una variable global declarada en otro módulo.***

- **volatile**: El compilador debe asumir que la variable está relacionada con un dispositivo y que puede cambiar de valor en cualquier momento.
  - ***Esto se utiliza más en el ámbito de programación de micros***

Constantes

- No cambian durante la ejecución del programa.
- Uso de la palabra reservada **const**.
- const int numero = 10;
- Las constantes también se pueden especificar mediante **macros**. Representan valores de sustitución.

Operadores

- Aritméticos binarios: + - * / %
- Aritméticos unarios: ++ -- (prefijos / postfijos).
- Notación corta: += -= *= /= %= &= |= ^=
- De relación: == > >= < <= !=
- De asignación: =
- Lógicos: && || !
- Sobre bits: & | ^ ~ >> <<
- sizeof(tipo / variable) → Devuelve el número de bytes de un tipo o una variable.
- Operador ternario: **? :** b = (a == 6) ? 3 : c;
- Tener en cuenta la prioridad de los operadores.
- Uso de paréntesis.

Arrays

- Es una estructura de almacenamiento en memoria.
- Conjunto de elementos todos del mismo tipo.
- Definición: tipo de dato nombre [dimensión];
  - int array[10];
- Inicialización:
  - int a[10] = {1, 2, 3}; A partir del 3º se ponen a 0.
  - int a[] = {1, 2, 3, 4}; De 4 elementos.
- Acceso: nombre[posición], empiezan en 0.
- El nombre del array indica la primera posición del array.

Arrays de caracteres

- Cuando definimos un array de caracteres el último elemento se marca con un ‘\0’.

- Definición:
  - char c[6] = {‘a’,’b’,’c’,’d’,’e’,’\0’};
  - char c[7] = {‘a’,’\0’}; Solo tengo 1 elemento.
  - char c[5] = {“Hola”}; El **‘\0’** se pone de forma automática.

Alcance

- Variables locales y globales.
- **Local**: sólo existe en el ámbito donde se define. Dentro de una función, definida dentro o como un parámetro formal.
- **Global**: accesible en todos los ámbitos. Se definen fuera de todas las funciones.

Utilización del compilador de C

- **El compilador de C, se divide en tres partes:**
  - **El preprocesador**
  - **El compilador**
  - **El enlazador**

El Preprocesador de C

- Transforma el programa fuente, convirtiéndolo en otro archivo fuente “predigerido”. Las transformaciones incluyen:
- Eliminar los comentarios.
- Incluir en el fuente el contenido de los ficheros declarados con **#include <fichero>** (a estos ficheros se les suele llamar **cabeceras**)
- Sustituir en el fuente las macros declaradas con **#define** (ej. #define CIEN 100)

El Compilador de C

- Convierte el fuente entregado por el preprocesador en un archivo en lenguaje máquina: **fichero objeto.**

- Algunos compiladores pasan por una fase intermedia en lenguaje ensamblador.

El Enlazador (linker)

- Un fichero objeto es código máquina, pero no se puede ejecutar, porque le falta código que se encuentra en otros archivos binarios.
- El **enlazador** genera el ejecutable binario, a partir del contenido de los ficheros objetos y de las **bibliotecas**.
- Las bibliotecas contienen el código de funciones precompiladas, a las que el archivo fuente llama (por ejemplo **printf**).

Ficheros en un proyecto en C

- Los tipos de fichero que pueden formar parte de un proyecto:
  - Ficheros de cabecera de C. Tienen extensión H.
    - Se utilizan para declaraciones. En un proyecto se pueden tener ficheros de cabecera propios con declaraciones de prototipos, tipos, etc.
  - Ficheros fuentes en lenguaje C. Tienen extensión C.
    - Se utilizan para la implementación de funciones.