Principales diferencias de C respecto a Java

C no es orientado a objetos

- No hay clases ni objetos. En C lo equivalente a los métodos serían las funciones, pero las funciones no están ligadas a objetos, cualquier función puede ser llamadas desde cualquier otra.

Tipos de datos

- No existe el tipo `boolean`. En su lugar suele usarse el tipo `int` (0 representa falso y cualquier otro valor representa cierto). Las expresiones booleanas en C funcionarán correctamente con operandos que sigan este convenio.

Punteros

- Puedes (y con frecuencia tendrás que hacerlo) acceder a elementos en la memoria a través de su dirección, mediante punteros (que no son más que un tipo de variable que contiene una dirección). De esta forma C te permite cambiar el contenido de la memoria casi arbitrariamente. Esta es una magia poderosa pero también peligrosa. En pocas palabras: cuando programas en C, estás más cerca de la máquina. Esto le da a C más flexibilidad, pero menos protección.

```
int* p; //declaración de un puntero a elementos de tipo entero
```

Arrays

- Los arrays son más simples en C: no hay verificación de límites. Hay que llevar cuidado en no acceder a índices fuera de rango, lo que puede provocar acceder erróneamente a otras variables adyacentes o que se fuerce la finalización del programa. El tamaño debe establecerse obligatoriamente en el momento de su declaración, bien poniéndolo explícitamente o deduciéndolo de su inicialización:

```
int numeros[3];
char letras[] = {'a', 'b', 'c', 'd', 'z'};
```

- Si no se inicializan explícitamente, el valor de los elementos de un array será habitualmente 0 (aunque no deberías confiar en que siempre sea así).

- Para especificar un tipo array para un parámetro de función, en ambos lenguajes puedes usar un array sin tamaño:

```
float producto_escalar(float v1[], float v2[], n)
{   ...     }
```

- Como en Java, los arrays en C son tipos de referencia. Es decir, el valor de una variable de array es una referencia al lugar donde se almacenan los valores del elemento. De hecho, el valor de una variable de array en C es la dirección en memoria del primer elemento del array, con el resto de los elementos ocupando las siguientes ubicaciones de memoria contiguas. Los arrays y otros tipos de referencia a menudo también se denominan punteros en C, ya que "apuntan" a los datos en la memoria. Las funciones reciben argumentos de tipo array siempre en forma de referencia, de hecho, son equivalentes estas dos declaraciones:

```
float producto_escalar(float v1[], float v2[], n)…

float producto_escalar(float* v1, float* v2, n)…
```

- Al igual que en Java, la longitud de un array no se puede cambiar después de que se ha creado, pero a diferencia de Java, no hay forma de obtener el tamaño del array o matriz. O se realiza un seguimiento de cuántos elementos hay en él o se marca la última entrada de alguna manera (así se hace con los arrays de caracteres, como se verá a continuación). Mediante una llamada `sizeof()` se obtendrá siempre el tamaño (en bytes) del espacio total reservado inicialmente en la declaración del array, sea cual sea su contenido.

- Finalmente, no hay soporte integrado para imprimir el contenido de un array. Necesitarás escribir una función que tome el array (y su longitud u otra forma de saber qué elemento es el último) e itere sobre sus elementos, imprimiéndolos.

Arrays de caracteres

- No existen Strings, lo más similar son los arrays de caracteres (también llamados tiras o cadenas de caracteres):

```
char nombre[]= "Pedro"; //declaración de una cadena de caracteres
```

- Estos arrays se manejan como los arrays de cualquier otro tipo, aunque para facilitar su manipulación, existe una colección de funciones estandarizadas cuyos nombres comienzan por `str…()`. Por ejemplo una llamada a la función `strlen(tira)` devolverá la longitud del contenido de la cadena `tira`. Para tener acceso a estas funciones es necesario añadir en la cabecera del programa en C:

```
#include <string.h>
```

- El final del contenido de las cadenas se marca con un carácter nulo (‘\0’) tras el último carácter. (Casi) todas las funciones de manipulación de cadenas siguen esta convención. Es frecuente en C procesar cadenas recorriéndolas hasta que se encuentre este carácter nulo como condición de terminación:

```
int i; char letra;
for(i=0; (letra=nombre[i])!='\0'; i++)
  printf("%c",letra);
```

- Como cualquier otro array, las cadenas de caracteres se pueden alterar (no son inmutables en el sentido de los Strings de Java), aunque, igual que en Java su tamaño máximo sí que es fijo, el que se estableció en el momento de su declaración, y por ello hay que llevar cuidado en no escribir cadenas de mayor longitud que el tamaño declarado.

- No se pueden concatenar cadenas de caracteres con el operador "+". Existe la función `strcat` para este fin:

```
#include <stdio.h>
#include <string.h>

int main(){
   char s1[11] = "Hola ";
   char s2[] = "mundo";
   strcat( s1, s2 );
   printf( "s1=%s\n", s1 );
}
```

  (El contenido de la cadena s2 se añade al final de s1. Hay que tener cuidado de que el tamaño declarado para s1 sea suficiente.)

- Para comparar dos cadenas se hace uso de la función `strcmp()`.

Manejo de memoria dinámica

- En C no existe la sofisticada administración de memoria de Java: debes asignar explícitamente (`malloc`) y liberar memoria (`free`) para usar estructuras dinámicas de datos como listas, árboles, etc. Todo lo que se asigna debe ser liberado explícitamente (mediante `free`) para que ese espacio pueda ser reutilizado, C no lo hará por ti, aunque sean datos que ya no se referencien.

Referencias

- George Ferguson, C for Java Programmers, (2020) https://www.cs.rochester.edu/u/ferguson/csc/c/c-for-java-programmers.pdf