

15. Tipos de Dato I

Por ahora ya sabemos lo suficiente sobre el muestreo de textos por pantalla, aunque por supuesto a falta de practicar un poco. Vamos a cambiar un poco de temática y vamos a ver otros aspectos avanzados del lenguaje que nos permitirán hacer cosas más complicadas.

Hasta ahora sólo hemos visto conceptos sobre variables cuyos valores eran numéricos. Si haces memoria recientemente hablamos de aplicar el prefijo "int" a la hora de definir una variable, con ello indicábamos que en ella sólo queríamos guardar valores enteros (integer), y por supuesto no siempre será suficiente con ello, de eso trata este nuevo tema.

15.1 Tipos de dato predefinidos

Toda variable en Bennu lleva asociado el tipo de dato que se va a almacenar en ella. Si no lo especificamos, la variable sólo servirá para almacenar enteros, pero por supuesto hay otras cosas que podemos guardar en una variable, como por ejemplo un texto o un número real.

Bennu y la mayoría de los lenguajes de programación vienen con una serie de tipos de dato predefinidos que podemos utilizar con diferentes fines. En el caso particular de Bennu, para especificar el tipo de dato de una determinada variable basta con añadirle un prefijo determinado a la hora de declararla.

Los prefijos que podemos añadir para declarar variables con los tipos de datos predefinidos en Benu son los siguientes:

Prefijo	Utilidad del tipo de dato	Ejemplo de declaración
int	<p>Almacena un número entero, es decir, números positivos y negativos incluido el cero, sin decimales. Es el tipo de dato por defecto de toda variable definida en Benu, así que si omitimos este prefijo la variable que declaremos será de tipo int.</p> <p>¿Para qué sirve entonces añadir este prefijo? Pues para aclararnos más fácilmente a la hora de leer nuestro código, es algo que debemos tener en cuenta a largo plazo, la legibilidad de nuestro código.</p>	<pre>int puntos = 0; int vida = 100;</pre>
float	<p>Almacena un número real, es decir, números positivos y negativos, con o sin decimales. Un float puede almacenar cualquier tipo de dato int, pero no al revés. Si queremos tener una variable de tipo float sí que es obligatorio añadir el prefijo "float".</p> <p>Lo más probable es que para los primeros videojuegos que programemos no tendremos necesidad de utilizar este tipo de dato.</p>	<pre>float porcentaje = 0,0; float precio = 19,95;</pre>
char	<p>Almacena un carácter de teclado, ya sea alfanumérico, un símbolo o incluso un espacio, la tecla de salto de línea o la de borrado. Los caracteres almacenables pertenecen al denominado código ASCII y son 256 en total, aunque cambian según la distribución de teclas. Para asignar un valor a las variables de este tipo es necesario escribirlo entre comillas simples (').</p> <p>No debes darle mayor importancia a este tipo de dato ya que su utilidad actual es muy limitada.</p>	<pre>char letra = 'a'; char otra_letra = 'w';</pre>
string	<p>Almacena un texto de cualquier longitud. Es como una serie de char concatenados y este tipo de dato sí que nos resultará útil para nuestro videojuego.</p> <p>Para asignar un texto a las variables de este tipo es necesario escribirlo entre comillas dobles (").</p>	<pre>string nombre = "Secret of Evermore"; string texto = "GAME OVER";</pre>

Debes tener en cuenta que seguimos hablando de variables, idénticas a todas aquellas con las que hemos trabajado hasta ahora: Pueden ser GLOBAL y PRIVATE, podemos asignarles un valor inicial o no, debemos añadir siempre al final de su declaración un punto y coma (;), etc.

Prueba a añadir unas cuantas variables adicionales a tu videojuego dentro de sus secciones GLOBAL y PRIVATE y experimenta con la función `write_var()` para comprobar cómo almacenan valores que ya no son simples enteros.

Es muy importante que compruebes que el compilador no muestra ningún error, eso significa que el lenguaje Bennu acepta estas nuevas expresiones que estamos incluyendo.

15.2 Arrays o vectores

Aunque su nombre no resulta muy ilustrativo y en la primera toma de contacto se tiene a pensar que un array o vector es algo muy complicado, se trata de un aspecto muy sencillo del lenguaje que facilita algunas tareas de programación.

Hasta ahora declarábamos las variables de 1 en 1, esto es, cada vez que queríamos tener una variable donde almacenar un valor necesariamente teníamos que escribir la línea de código correspondiente. Imagina que queremos que los enemigos de nuestro videojuego puedan tener 10 valores de puntos de vida distintos, no simples valores aleatorios, sino valores fijos como por ejemplo: 10, 25, 50, 100, 250, 750, etc.

La serie de números anterior podría generarse a partir de números aleatorios dados por la función `rand()`, aunque si lo piensas verás que llevaría algunas complicaciones adicionales.

Para almacenar esa serie o lista de números podemos utilizar un array o vector (En adelante lo llamaremos siempre vector). Un vector no es más que una lista de variables, tantas como queramos, cada una de ellas con su valor asociado.

En el caso de la lista anterior necesitamos una lista de 10 números enteros, cada uno con un valor distinto. Observa que el tipo del dato a almacenar es un simple entero (Prefijo `int`), la única diferencia es que queremos una variable que no almacene un único entero sino 10 en nuestro caso.

Podemos declarar una lista de 10 enteros en la sección GLOBAL de nuestro videojuego así:

```
GLOBAL  
    int puntos_de_vida [ 10 ];
```

Observa que al final de la declaración hemos añadido, entre corchetes, el número de variables que queremos declarar.

La sintaxis formal de la declaración de un vector, sea cual sea su tipo de dato, es la siguiente:

```
<tipo_del_dato> nombre_del_vector [ <longitud_del_vector> ] ;
```

Donde el tipo de dato puede ser cualquiera de los vistos anteriormente como int, float o string, y la longitud del vector puede ser cualquier número positivo, teniendo en cuenta que cuanto mayor sea la longitud mayor será la memoria que consuma nuestro videojuego.

Por ejemplo, si tenemos en cuenta que un número entero ocupa 4 bytes en memoria, un vector de 1024 enteros ocuparía 4096 bytes, que son 4 kilobytes. Fíjate que es un valor despreciable teniendo en cuenta que las memorias actuales se miden en gigabytes.

Para almacenar en nuestro vector los valores de vida de los enemigos mencionados anteriormente podemos hacerlo en el mismo momento de su declaración de la siguiente forma:

```
GLOBAL  
    int puntos_de_vida [ 10 ] = ( 10, 25, 50, 100, 250, 750, 1000, 2000, 5000, 10000 );
```

Fíjate que los valores que almacenamos en la lista van separados por comas y encerrados entre paréntesis. ¿Qué ocurriría si almacenamos más números de los que hemos declarado en la lista? Simplemente evita hacerlo, ya que podría provocar errores de ejecución en nuestro videojuego, más adelante veremos por qué.

Sólo nos queda un detalle: ¿Cómo podemos hacer que cada enemigo tenga una vida distinta de entre los valores almacenados en nuestro vector?

Para ello necesitamos acceder a los datos del mismo y asignarlos a su vida. Lo haremos con una simple instrucción de asignación que podremos incluir dentro de su bloque BEGIN – END como cualquier otra instrucción. Por ejemplo:

Instrucción	Utilidad	Aclaraciones
<code>vida = puntos_de_vida [0];</code>	Asignaría a la vida de nuestro enemigo el primero valor del vector de puntos de vida, en nuestro caso el valor que almacena es 10.	La posición 0 es la primera posición de todo vector declarado. Para un vector de longitud 10 podemos acceder a sus posiciones desde la 0 hasta la 9.
<code>vida = puntos_de_vida [6];</code>	Asignaría a la vida de nuestro enemigo el séptimo valor del vector de puntos de vida, en nuestro caso el valor que almacena es 2000.	No olvides que para un vector de longitud N sus posiciones van desde la 0 hasta la N-1, es algo que siempre deberás tener en cuenta.
<code>vida = puntos_de_vida [rand (0 , 9)];</code>	Asignaría a la vida de nuestro enemigo un valor aleatorio, entre la posición 0 y la posición 9 de nuestro vector de puntos de vida.	Sí, podemos utilizar la función <code>rand()</code> para este tipo de propósitos, al fin y al cabo es una función muy sencilla que simplemente devuelve un valor aleatorio dentro del intervalo dado.

Y con lo aprendido sobre muestreo de textos también podríamos hacer que la vida de cada enemigo aparezca sobre el mismo, ¿Te atreves a intentarlo?.