APRENDERAPROGRAMAR.COM

GENERAR NÚMEROS O SECUENCIAS ALEATORIOS EN C. SRAND Y RAND. TIME NULL. RAND_MAX. (CU00525F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

Fecha revisión: 2031

Resumen: Entrega nº25 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

GENERAR NÚMEROS ALEATORIOS EN C: SRAND Y RAND

La generación de números aleatorios adquiere gran relevancia para un programador, pudiendo usarse para distintas tareas de las que vamos a citar algunas, aunque hay tantas posibilidades que resultan prácticamente infinitas.

- Construcción preliminar de programas, en los que a falta de datos definitivos introducimos datos aleatorios.
- Simular procesos aleatorios (número resultante de tirar un dado, elección de un color por parte de una persona, número premiado en un sorteo de lotería, cantidad de personas que entran a un supermercado en una hora...)
- Verificación de programas, haciendo pruebas con datos aleatorios.
- Otras aplicaciones.

Conviene recordar que "aleatorio" no puede confundirse con "cualquier cosa", "descontrol", "incierto", "impredecible", etc. Usaremos aleatorio más en el sentido de no predeterminado que en el de no predecible, ya que en general vamos a definir qué tipo de resultado queremos obtener y en qué rango de valores debe estar. Vamos a imaginar que el lenguaje C genera números aleatorios como si fuera un robot lanzador de dardos muy preciso ("robot" *rand*). De este modo, cuando se le dice que comience a tirar dardos en distintas posiciones, repite siempre los lugares. Por ejemplo, si la diana está marcada con números, cada vez que le decimos que tire genera la misma secuencia: 7, 5, 6, 3, etc. ¿Cómo conseguir convertir este proceso predefinido en aleatorio? Pues simplemente poniendo a girar la diana ("mayordomo" *srand*) esta vez a una velocidad que depende del segundo del día en que nos encontremos. Así pues, el proceso lo dividimos en decirle al mayordomo que ponga a girar la diana y en decirle al robot que dispare. Bueno, no es una explicación técnica, pero ¿para qué complicarnos? Veamos la sintaxis a emplear:

1) Asignación de valor aleatorio a una variable.

srand (time(NULL)); [Instrucción que inicializa el generador de números aleatorios]

int variable = rand(); [Obtención de un número aleatorio entero entre cero y un valor muy grande]

2) Mostrar un valor aleatorio.

```
srand (time(NULL));
printf ("%d\n",rand());
```

El valor devuelto por *rand()* es de tipo entero, aunque si la variable a la que se asigna el contenido de *rand()* es de otro tipo, tendrá lugar la conversión del valor originario para adaptarlo al tipo de variable de que se trate. Un aspecto importante a tener en cuenta es que el valor devuelto se encuentra en el rango 0 <= *valor* < *constante numérica RAND_MAX*. Es decir, el número devuelto puede ser cero o un valor cualquiera hasta el límite que define una constante de C (que podemos conocer escribiendo printf("%d", RAND_MAX);. Ejecuta este código y comprueba los resultados:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// Ejemplo aprenderaprogramar.com
int main() {
srand(time(NULL)); //El mayordomo pone a girar la diana
int test = rand(); //Primer disparo del robot
printf ("El numero aleatorio de test vale %d\n", test);
printf ("Otros numeros aleatorios son: %d, %d, %d\n",rand(),rand(),rand()) ;
printf("La constante RAND_MAX vale %d\n", RAND_MAX);
return 0;
}
```

Un posible resultado con este programa es: *El numero aleatorio de test vale 21051. Otros numeros aleatorios son: 12546, 22788, 7054. La constante RAND_MAX vale 32767.*

Si repites la ejecución del programa obtendrás otros valores distintos, excepto el de la constante RAND_MAX que como su propio nombre indica es fijo (para un sistema y compilador dados).

La sentencia *time(NULL)* devuelve el número de segundos que han pasado desde el 1 de enero de 1970 según el reloj del sistema y nosotros la veremos como un número a partir del que se genera la velocidad de giro de la diana. Siempre será distinto porque nunca ejecutaremos un programa en el mismo segundo en que se ejecutó anteriormente. La sentencia *srand(valor);* la veremos como la instrucción que pone a girar la diana a una velocidad calculada a partir del número indicado en el parámetro "valor".

Las funciones *srand* y *rand* están disponibles gracias a la directiva del preprocesador stdlib.h. Para poder hacer uso de la función *time* hemos de incluir la línea #include <time.h>, que es la que nos facilita el uso de las funciones relacionadas con el tiempo en C.

En muchos casos querremos obtener un número con un valor máximo acotado: por ejemplo un número aleatorio entre 0 y 3. Para ello nos podemos valer de la operación matemática que nos devuelve el resto de una división entera, que en C se expresa con el símbolo % y que hemos denominado operación de módulo o de obtención del resto. Por ejemplo 6%3 es el resto de dividir 6 entre 3. Como la división es exactamente 2, el resto es 0. En cambio 7%3 genera un resto 1. 8%3 genera un resto 2 y 9%3 genera un resto 0 nuevamente.

Si nos fijamos los valores posibles al hacer la operación Numero%3 son 0 (la división es exacta), 1 (el resto es 1) ó 2 (el resto es 2). En ningún caso el resto podría ser 3 ni un valor superior a 3. Basándonos

en esto podemos obtener números aleatorios entre 0 (incluido) y un número NUM (incluido) haciendo la operación módulo al número aleatorio que nos devuelve *rand()* así: rand%(NUM+1)

Por ejemplo: para obtener un número aleatorio comprendido entre 0 y 10 escribiríamos *rand()%11*. El resultado de esta operación será 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ó 10.

Ejecuta este código varias veces y comprueba cómo varían los resultados:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// Ejemplo aprenderaprogramar.com
int main() {
srand(time(NULL)); //El mayordomo pone a girar la diana
int test = rand() % 5; //Primer disparo del robot
printf("Numero aleatorio entre 0 y 4 es %d\n" ,test);
printf("Otros aleatorios entre 0 y 4 son %d, %d, %d\n" ,rand()%5,rand()%5,rand()%5);
printf("Otros mas son %d, %d, %d\n" ,rand()%5,rand()%5,rand()%5);
return 0;
}
```

Finalmente también será habitual que además de tener un extremo superior, también queramos tener un valor mínimo inferior. Por ejemplo para simular la tirada de un dado necesitaremos que el número mínimo sea un 1 y el número máximo un 6. Esto lo podemos simular escribiendo *rand()%6+1*. En este caso *rand()%6* nos devuelve 0, 1, 2, 3, 4 ó 5. Al sumarle un 1 estaremos obteniendo siempre alguno de los valores deseados: 1, 2, 3, 4, 5 ó 6.

Si quisiéramos obtener números entre 20 y 30 escribiríamos *rand()%11+20*. La componente fija 20 nos da el mínimo y el módulo nos da números comprendidos entre 0 y 10 de modo que el valor mínimo es 20 y el máximo 30. La fórmula general sería rand()%(INTERVALO+1)+MINIMO

Una instrucción *srand* puede utilizarse para múltiples *rand.* De todas formas, si es un programa extenso y tienes dudas, puedes poner un *srand* asociado a cada uno de los *rand* o grupos de *rand* que tengas en distintas partes del programa. De esta manera reduces el "determinismo" o posibilidad de repetición de ciertos valores o intervalos que podrían resultar poco adecuados para representar aleatoriedad.

La generación de números aleatorios puede presentar problemas cuya base es matemática. Estos problemas pueden ser del tipo "que la aleatoriedad no sea uniforme en el rango deseado". Por ejemplo, si generamos números entre 1 y 100 querremos que cualquier número tenga igual probabilidad de salir siempre que esté dentro del rango. Sin embargo, puede ocurrir que ciertos números tengan mayor probabilidad de salir debido a las cuestiones matemáticas que subyacen a este respecto. La serie que obtienes puede ser "aparentemente aleatoria" y sin embargo no estar uniformemente distribuida. En algunos casos esto puede considerarse aceptable, pero en otros no. Nosotros no vamos a profundizar en el estudio matemático de la generación de aleatorios y nos resulta suficiente el tratamiento que hemos visto, pero si tuvieras que realizar programas donde es importante el aspecto matemático, deberás estudiar más a fondo el funcionamiento de los números aleatorios en C.

EJERCICIO

Crea un programa donde se muestre el mensaje: “El número de pacientes en la cola es x. Transcurridos 10 minutos el número de pacientes en la cola es x”. Donde debes sustituir x por un aleatorio comprendido entre 1 y 225 ambos inclusive. El resultado del programa debe ser, por ejemplo, <<El número de pacientes en la cola es 132. Transcurridos 10 minutos el número de pacientes en la cola es 204>>.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00526F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210