Analista Universitario en Sistemas

Bucle for y generacíón de números aleatorios

La instrucción **for** explota la capacidad del los equipos para realizar procesos repetitivos.
Se utiliza generalmente cuando se conoce previamente la cantidad de veces a repetir una serie de instrucciones.

Sintaxis:

```
int i;       //denominada variable índice, puede ser cualquier nombre de variable
for (i=0; i<10; i++) {    //en cada iteración se ejecuta este bloque de instrucciones
   instruccion_1;
   instruccion_2;
   . . .
   instruccion_N;
}
```

Esta instrucción de control iterativa o bucle consta de varias partes o términos separados por el símbolo punto y coma (;):

▷ 1er término: inicialización
`i=0` inicialización de la variable índice
▷ 2do término: condición
`i<10` condición que debe cumplir la variable índice para que se ejecute el bloque de instrucciones
▷ 3er término: cambio o modificación
`i++` cambio en la variable índice (en este caso i = 1+1)

Orden de ejecución:
- La inicialización de variables se ejecuta una única vez y antes que cualquier otra instrucción o término
- Luego se evalua la condición, de evaluar a verdadero se ejecuta el bloque de instrucciones
- Al finalizar el bloque de instrucciones se ejecuta el 3er término, donde se alteran variables
- Se vuelve a evaluar la condición, si es verdadera se repite este procedimiento, si es falsa se finaliza el bucle o ciclo for.

Mientras no evalue a falso la condición se continua con iteraciones. Es un riesgo la generación de bucles infinitos, por ello prestar particular atención a la condición y como cambia la variable índice luego de cada ejecución del bloque de instrucciones.

Ejemplos:
● Muestra los primeros diez enteros positivos

```
int i;
for (i=1; i<=10; i++) {
   printf("%d ", i);
}
```

● Muestra los números 0, 2, 4, 6, 8, 10

```
int i;
for (i=0; i<=10; i+=2) {
   printf("%d ", i);
}
```

● Muestra los números 10 al 1 en forma decremental

```
int i;
for (i=10; i>0; i--) {
   printf("%d ", i);
}
```

● Suma los primeros diez enteros positivos, al final se muestra dicha sumatoria.

```
int i, suma=0;  //importante, cuando una variable actua como acumulador, debe inicializarse
for (i=1; i<=10; i++) {
      suma = suma + i;
}
printf("La suma es %d ", suma);
```

● Un for sin sus términos, mostrando los números del 1 al 10, notar el uso del if

```
int i=0;
for ( ; ; ) {
  if ( i<=10 ) {
      printf("%d ", i);
      i++;
  } else {
     break;     //instrucción que se emplea para interrumpir un bucle o ciclo, vea continue
  }
}
```

● Un for con dos variables índice, una para mostrar los números del 1 al 10 y la otra para realizar lo inverso, es decir mostrar del 10 al 1.

```
int i, j;
for (i=1,j=10; i<=10; i++,j--) {
      printf("%d - %d ", i, j);
}
```

Pregunta: cómo lo haría con una sola variable?

Analista Universitario en Sistemas

Bucle for y generacíón de números aleatorios

Números aleatorios

Supongamos que queremos generar una lista de 10 números entre 0 y 1000 aleatorios, es decir números escogidos al azar en dicho rango.

Podemos generar este listado gracias a las funciones de **<stdlib.h>**

```
int rand(void);
void srand(unsigned int seed);
```

La función **rand** retorna un número pseudo-randómico entre [0, RAND_MAX).
La función **srand** establece una "semilla" para la generación de dichos números.

Para los curiosos

```
En stdlib.h:  #define RAND_MAX 2147483647
```

Veamos un ejemplo utilizando solo **rand**.

```
int i;
for (i=1; i<=10; i++) {
    printf("%d ", rand() % 1001);
}
```

Observar el uso del operador módulo (%), se emplea para asegurar que cualquiera sea el número retornado por **rand()** no supere el 1000.

Genere un programa con este código y observe la salida de sucesivas ejecuciones.
¿nota algo particular?
Seguramente observa que siempre se generan los mismos 10 enteros, solo cambia esto en caso de volver a compilar el código nuevamente, entonces se generarán otros 10 enteros aleatorios, pero en cada ejecución se repetirán.

Aquí es donde entra en juego **srand**, se invoca una única vez y permite que se establezca un valor cambiante como argumento para que se genere en cada ejecución de un programa una lista diferentes de valores.

Ejemplo:
Se pasará como argumento a **srand** la salida de la función **time** de **<time.h>** que retorna un valor númerico diferente cada segundo (como está siendo utilizada).

```
int i;
srand(time(NULL));
for (i=1; i<=10; i++) {
    printf("%d ", rand() % 1001);
}
```

Cree y complie un programa que incluya este código, no ovlide los archivos de cabecera (header files) y ejecute el programa varias veces.
Si realiza ejecuciones dejando pasar más de un segundo entre ellas notará que la secuencia de números varía en cada ejecución.
Si ejecuta el programa para que tengan lugar dos ejecuciones durante el mismo segundo, verá que la secuencia se repite.

¿Cómo resolver esto?
Existe otra función que puede ser utilizada como argumento para **srand** a fin de que se genere siempre una semilla diferente en cada ejecuón más allá de que tan velozmente se pueden realizar ejecuciones.
Deberá incluir estos headers files

```
#include <sys/types.h>
#include <unistd.h>
```

La función es

```
pid_t getpid(void);
```

que retorna el número de proceso (único) que corresponde a cada programa en ejecución. Así, cada vez que ejecute el código **getpid** retornará un valor diferente, por ende se generará un semilla diferente y, por consiguiente, una secuencia de números diferentes.

Cree y complie un programa que incluya el siguiente código:

```
int i;
printf("Mostrando PID del proceso %d.", getpid());

srand(getpid());
for (i=1; i<=10; i++) {
    printf("%d ", rand() % 1001);
}
```

TAREA:
- realizar programas para ejecutar los ejemplos
- leer en bibliografía bucle for y bucles anidados (nested loops)
- leer en bibliografía bucle while y do...while
- leer en bibliografía la instrucción continue