Tarea #5

Inyerman Alexander Xap Chin 20210097

Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala.

A. Resumen

En esta practica hicimos un simulacro de lo que será nuestro parcial realizamos las distintas series como preparación para el parcial que se aproxima.

B. Objetivos

Tener una idea de lo que se avecina en el parcial.

Poder practicar antes del parcial.

C. Marco Teórico

Obtención de números aleatorios en C

A veces queremos que nuestro programa obtenga números de forma aleatoria, por ejemplo, para simular una tirada de dados o el reparto de cartas en un juego. En C de linux tenemos varias funciones que nos permiten obtener estos valores aleatorios.

En esta explicación vamos a ver un uso básico de estas funciones. Algunas de las cosas que contamos aquí no son útiles para aplicaciones más serias, en las que se requiere que la secuencia de números aleatorios sea muy aleatoria, impredecible, que no se repita hasta pasado muchos números, etc, etc. Sin embargo, las explicaciones aquí presentadas servirán para la mayoría de nuestros programas.

LA FUNCIÓN rand()

En C, para obtener números aleatorios, tenemos la función rand(). Esta función, cada vez que la llamamos, nos devuelve un número entero aleatorio entre 0 y el RAND_MAX (un número enorme, como de 2 mil millones).

El primer problema que se nos presenta es que no solemos querer un número aleatorio en ese rango, sería un dado muy curioso el que tenga tantas caras. Podemos querer, por ejemplo, un número aleatorio entre 0 y 10. O de forma más general, entre 0 y N. La cuenta que debemos echar para eso es esta La operación módulo (%) nos da el resto de dividir rand() entre 11. Este resto puede ir de 0 a 10. De la misma forma, el módulo de rand() entre N+1 va de 0 a N.

¿Y si queremos un rango que no empiece en 0?. Por ejemplo, queremos un rango entre 20 y 30 (de forma más general, entre M y N con N mayor que M). Pues es fácil, obtenemos un número entre 0 y 10 y le sumamos 20 (un número entre 0 y N-M y le sumamos M)

LA FUNCIÓN srand()

Se nos presenta un nuevo problema. Si ejecutamos varias veces nuestro programa, la secuencia de números aleatorios se repite. Imaginemos que tenemos un programa que escribe 3 número aleatorios entre 0 y 10. Lo ejecutamos una vez y sale, por ejemplo 4, 7 y 9. Lo ejecutamos por segunda vez y vuelve a salir 4, 7 y 9. La tercera vez sale lo mismo y cuando ya nos hemos hartado de ejecutar, vuelve a salir lo mismo.

El problema es que rand() "calcula" los números aleatorios. Parte de un número inicial (llamado semilla), echa unas cuentas y saca un número aleatorio. Para el segundo número, echa unas cuentas con el resultado anterior y saca un segundo número y así sucesivamente.

Si volvemos a ejecutar el programa desde el principio, el número inicial (la semilla) que usa rand() es el mismo, con lo que la secuencia de números aleatorios es la misma, ya que las cuentas son las mismas.

Para evitar este problema tenemos la función srand(), a la que se le pasa de parámetro un número que se utilizará como número inicial para las cuentas. A esta función sólo debemos llamarla una vez en nuestro programa.

¿Qué número le ponemos a este srand()?. No podemos ponerle un número fijo, porque entonces no hemos hecho nada. No podemos ponerle un número obtenido con rand(), porque la primera vez siempre nos dará el mismo y el resultado será igual que si le ponemos un número fijo. Debemos buscar la forma de obtener un número que sea distinto en la ejecución de cada programa.

Hay dos números que se utilizan habitualmente para ello:

La fecha/hora del sistema. Este valor cambia si ejecutamos el programa en distinto instante de tiempo. Tendriamos que arrancar el programa dos veces en el mismo segundo para obtener la misma secuencia de números aleatorios. En C de linux esta fecha/hora se obtiene con la función time()

```
srand (time(NULL));
```

El número de proceso del programa. El primer programa que se arranca cuando se enciende el ordenador con el sistema operativo linux, tiene el número de proceso 1, el segundo el 2, el tercero el 3 y así sucesivamente. Cuando arrancamos nuestro programa, se le asignará el número que le toque, por ejemplo, el 215. Cuando lo volvamos a arrancar, se le asignará el que le toque (puede ser 216 si no hemos ejecutado nada entre medias o 345, si nos hemos entretenido con otras cosas). Después de ejecutar nuestro

programa varios miles de veces, el número de proceso puede que se repita, pero ya no nos acordaremos de la secuencia que se sacó la primera vez. El número de proceso se obtiene con getpid()

srand (getpid());

A esta función sólo hay que llamarla una vez al principio de nuestro programa. Cada vez que la llamemos, estaremos reiniciando los calculos de números aleatoriosd desde el principio, con lo que se repetirá todo.

D. Marco Practico

Debimos investigar y reforzar algunos códigos en la librería stdio.h y stdlib.h para poder realizar los programas.

E. Código

Código del juego de 8

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main () {
    int dado1, dado2, resultado;
    srand(time(NULL));
    dado1=rand() % 6 + 1;
    dado2=rand() % 6 + 1;
    resultado = dado1 + dado2;
    /*(0 a 6)*/
    printf("Juego del 8\n");
    printf("Reglas lanzas 2 dados y si sumas 8 ganas\n");
    printf("Si sacas 7 pierdes\n");
    printf("Si sacas cualquier otro numero puedes seguir jugando\n");
    /*Reglas del juego */
    printf("Tus dados dieron:\n");
    printf("Dado 1: %i\n", dado1);
    printf("Dado 2: %i\n", dado2);
    printf("Resultado: %i\n", resultado);
    if(resultado==7) {
       printf("Lo lamento perdiste\n");
    }
    if(resultado==8) {
        printf("Felicidades Ganaste\n");
        }
        else {
        printf("Sigue participando\n");
        }
    return 0;
}
```

Código De Cálculo de Impuestos

```c
#include<stdio.h>
int main() {
    printf("Este programa se encarga de calcular el iva de un producto\n");
    printf("Por favor Ingrese el precio de su articulo \n");
    double precio;
    double iva;
    double preciolibre;
    scanf("%lf",&precio);
    /*calculo*/
    iva = (precio*0.12);
    preciolibre= precio-iva;
    printf("Precio total :%.2lf\n", precio);
    printf("IVA :%.2lf\n", iva);
    printf("Precio sin IVA :%.2lf\n", preciolibre);

    return 0;
}
```

F. Resultados

Código de juego del ocho Corriendo

```
c:\Codigos769>dados
Juego del 8
Reglas lanzas 2 dados y si sumas 8 ganas
Si sacas 7 pierdes
Si sacas cualquier otro numero puedes seguir jugando
Tus dados dieron:
Dado 1: 3
Dado 2: 4
Resultado: 7
Lo lamento perdiste
Sigue participando

c:\Codigos769>
```

Código De Cálculo de Impuestos Corriendo

```
c:\Codigos769>iva
Este programa se encarga de calcular el iva de un producto
Por favor Ingrese el precio de su articulo
100.00
Precio total :100.00
IVA :12.00
Precio sin IVA :88.00

c:\Codigos769>
```

G. Conclusiones

Pudimos hacernos una idea de lo que podemos esperar para el parcial y nos dimos cuenta que necesitamos practicar mas.

Pudimos practicar y reforzar nuestros conocimientos de las distintas librerías.

H. Referencias

https://es.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/mean-median-basics/a/mean-median-and-mode-review#:~:text=Para%20encontrar%20la%20mediana%3A,al%20medio%20de%20la%20lista.

https://old.chuidiang.org/clinux/funciones/rand.php#:~:text=En%20C%2

C%20para%20obtener%20números,como%20de%202%20mil%20millones).

https://www.youtube.com/watch?v=ZBiRjjUdM40&t=681s