**Universidad de Costa Rica**
**Escuela de Ingeniería Eléctrica**
**Programación Bajo Plataformas Abiertas**
**MSc. Andrés Mora Zúñiga**
**I Ciclo 2020**
**IE-0117**

# Práctica # 5: C: Bucles, numeros aleatorios y Makefile

## 1. Adivina el entero(Juego)

Para esta parte se realizó un archivo llamado Adivina.c.El cual corresponde a:

```
#include <stdio.h>
#include<stdlib.h>
#include <time.h>
int numeroaleatorio(int limitemenor, int limitemayor){
    srand(time(NULL));
    int numaleatorio =rand()%(limitemayor+1-limitemenor)+limitemenor;
    return numaleatorio;
}
void ciclo(int numaleatorio, int limitemenor,int limitemayor){
    int num1;
    printf("Inserte un numero:");
    scanf("%d",&num1);
    while (num1!=numaleatorio)
        {

            if(num1>=limitemenor && num1<=limitemayor && num1<numaleatorio){
                printf("El numero es mayor\n");
            }
            if(num1>=limitemenor && num1<=limitemayor && num1>numaleatorio){
                printf("El numero es menor\n");
            }
            if (num1<limitemenor || num1>limitemayor)
            {
            printf("error\n");
            break;
            }
            printf("Inserte otro numero:");
            scanf("%d",&num1);
        }
        if (num1==numaleatorio)
        {
            printf("correcto\n");
        }


}

int main (int argc, char const *argv[])

{
```

```
41      int limitemenor= atoi(argv[1]);
42      int limitemayor= atoi(argv[2]);
43
44      int numaleatorio =numeroaleatorio(limitemenor,limitemayor);
45      ciclo(numaleatorio,limitemenor,limitemayor);
46
47
48      return 0;
49  }
```

Primeramente se puede ver que se importaron las bibliotecas stdio.h, la stdlib.h para el caso de obtener los parámetros en la linea de comandos y time.h para generar un número aleatorio. Empezando por el main en la linea 38 se tiene, primero se extraen los parámetros de la linea de comandos con ayuda del atoi como se ve en la linea 41, donde estos corresponderían a los limites mayor y menor que forman un rango entre los posibles valores que el número aleatorio puede tener. Posteriormente se llama a la función numeroaleatorio con los parámetros limite menor y limite mayor como se ve en la linea 44. Por último se llama a la función ciclo con los parámetros numeroaletorio, limitemenor y limitemayor.

La función numeroaleatorio consiste básicamente en generar un número aleatorio. Se utiliza en la linea 5 el stand(time(NULL)) de modo que siempre que se ejecute el programa aparezca un nuevo numero aleatorio. En la linea 6 se utiliza el rand() % (limitemayor+1-limitemenor)+limitemenor, cabe explicar que el + limitemenor se le suma tanto al rand(que primeramente tiene valor 0) y al limitemayor,razón por la cual se le resta entre parentesis al limitemayor el limitemenor, además que es necesario sumarle uno porque osino el limitemayor no da el valor deseado. De esta manera se obtiene un numero aleatorio entre el rango del limitemenor y limitemayor.

La función ciclo primeramente se introduce mediante scanf el numero num1, que es el que el usuario ingresa y para ganar debe de cumplirse que ese num1 tenga igual valor que el numero aleatorio. Como se puede ver en la linea 12. Posteriormente en la linea 13 se entra a un while que permanecera simpre que el num1 sea diferente al numaleatorio. En la linea 14 se ve la pista de que el numaleatorio es mayor que el numero que ntrodujo el usuario y en la linea 19 que más bien es menor. El if de la linea 22 imprime un error en caso que el numero del usuario no se encuentre entre los rangos posibles y hace un break y el juego termina. En la linea 27 se sustituye el num1 para que el usuario vuelva a intentarlo. Saliendo del while el if que se muestra en la linea 30 posee la condición que cuando el num1 sea igual a numaleatorio imprima correcto y así el usuario sabe que ganó.En la figura 1 se muestra un ejemplo cuando el usuario adivina. En la figura 1 se muestra un ejemplo cuando el usuario inserta un número que esta fuera del rango.

```
alberto@debian:~/laboratorio5/juego$ gcc -o salida.x Adivina.c
alberto@debian:~/laboratorio5/juego$ ./salida.x 20 30
Inserte un numero:20
El numero es mayor
Inserte otro numero:25
El numero es mayor
Inserte otro numero:28
correcto
```

**Figura 1:** Adivinanza correcta

```
alberto@debian:~/laboratorio5/juego$ gcc -o salida.x Adivina.c
alberto@debian:~/laboratorio5/juego$ ./salida.x 20 30
Inserte un numero:12
error
```

**Figura 2:** Adivinanza error

## 1.1. Primer punto extra

Para esta parte se realizó otro código con el nombre de puntosextra1.c. Como se muestra en:

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  #include <time.h>
4  int numeroaleatorio(int limitemenor, int limitemayor){
5      srand(time(NULL));
6      int numaleatorio =rand() %(limitemayor+1-limitemenor)+limitemenor;
7      return numaleatorio;
8  }
9  void ciclo(int numaleatorio, int limitemenor,int limitemayor){
10     int num1;
11     float N= (1.0*(limitemayor-limitemenor)/3);
12     int n=0;
13     while (num1!=numaleatorio && n<=N)
14         {
15             printf("Inserte un numero:");
16             scanf("%d",&num1);
17
18             if(num1>=limitemenor && num1<=limitemayor && num1<numaleatorio){
19                 printf("El numero es mayor\n");
20             }
21             if(num1>=limitemenor && num1<=limitemayor && num1>numaleatorio){
22                 printf("El numero es menor\n");
23             }
24             if (num1<limitemenor || num1>limitemayor)
25             {
26             printf("error\n");
27             break;
28             }
29             n++;
30
```

```
31          }
32          if (n>N &&num1!=numaleatorio)
33          {
34              printf("Se acabaron los intentos\n");
35          }
36
37          if (num1==numaleatorio)
38          {
39              printf("correcto\n");
40          }
41
42
43  }
44
45  int main (int argc, char const *argv[])
46
47  {
48      int limitemenor= atoi(argv[1]);
49      int limitemayor= atoi(argv[2]);
50
51      int numaleatorio =numeroaleatorio(limitemenor,limitemayor);
52      int num1;
53      int num2;
54
55      ciclo(numaleatorio,limitemenor,limitemayor);
56
57      return 0;
58  }
```

El código es muy parecido al anterior, las diferencias están en la función ciclo. En la linea 11 se crea un float llamado N que van a ser la cantidad de oportunidades que se tendrán para jugar, así mismo se crea también n que tendra un valor inicial de cero. Seguidamente dentro del while de la linea 23 la diferencia es que mediante AND se agrega la condición de que si n>N pare el ciclo, ya que esto significará que se acabaron las oportunidades. Por ende en el if de la linea 32 dice respectivamente esto en caso de que se acaben las oportunidades y no se llegue a adiivinar el número donde se imprime que se acabaron las oportunidades.En la figura 1.1 se muestra un ejemplo cuando el usuario adivina y cuando el usuario se queda sin numero de intentos.

```
alberto@debian:~/laboratorio5/juego$ ./salida.x 0 5
Inserte un numero:1
correcto
alberto@debian:~/laboratorio5/juego$ ./salida.x 0 5
Inserte un numero:1
El numero es mayor
Inserte un numero:1
El numero es mayor
Se acabaron los intentos
```

**Figura 3:** Adivinanza con numero de intentos

## 1.2. Segundo punto extra

Para esta parte se realizó otro código con el nombre de puntosextra2.c. Como se muestra en:

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  #include <time.h>
4  int numeroaleatorio(int limitemenor, int limitemayor){
5      srand(time(NULL));
6      int numaleatorio =rand()%(limitemayor+1-limitemenor)+limitemenor;
7      return numaleatorio;
8  }
9  void ciclo(int numaleatorio, int limitemenor,int limitemayor){
10     float N= (1.0*(limitemayor-limitemenor)/3);
11     int n=0;
12     float num1;
13     float rangohirviendo= numaleatorio-(0.95*numaleatorio);
14     float rangocaliente= numaleatorio-(0.75*numaleatorio);
15     float rangofrio= numaleatorio-(0.50*numaleatorio);
16     int caliente=0;
17     int frio=0;
18     int hirviendo=0;
19     float menorhirviendo = 1.0*(numaleatorio-rangohirviendo);
20     float mayorhirviendo = 1.0*(numaleatorio+rangohirviendo);
21     float menorcaliente = 1.0*(numaleatorio-rangocaliente);
22     float mayorcaliente = 1.0*(numaleatorio+rangocaliente);
23     float menorfrio = 1.0*(numaleatorio-rangofrio);
24     float mayorfrio = 1.0*(numaleatorio+rangofrio);
25     while (num1!=numaleatorio && n<=N)
26         {
27             printf("Inserte un numero:");
28             scanf("%f",&num1);
29             if(num1!=numaleatorio && num1>=limitemenor && num1<=limitemayor &&
                   menorhirviendo<=num1 && num1<=mayorhirviendo){
30                 printf("Hirviendo\n");
31                 hirviendo++;
32             }
33             if(num1!=numaleatorio && num1>=limitemenor && num1<=limitemayor &&
                   menorcaliente<=num1 && num1<=mayorcaliente && hirviendo==0){
34                 printf("Caliente\n");
35                 caliente++;
36             }
37             if(num1!=numaleatorio && num1>=limitemenor && num1<=limitemayor &&
                   menorfrio<=num1 && num1<=mayorfrio && caliente==0 &&
                  hirviendo==0){
38                 printf("Frio\n");
39                 frio++;
40             }
41             if(num1!=numaleatorio && num1>=limitemenor && num1<=limitemayor &&
                   frio==0 && caliente==0 && hirviendo==0){
42                 printf("Helado\n");
43
44             }
45
46             hirviendo=0;
```

```
47              caliente=0;
48              frio=0;
49
50              if (num1<limitemenor||num1>limitemayor)
51              {
52              printf("error\n");
53              break;
54              }
55              n++;
56
57          }
58          if (n>N &&num1!=numaleatorio)
59          {
60              printf("Se acabaron los intentos\n");
61          }
62
63          if (num1==numaleatorio)
64          {
65              printf("correcto\n");
66          }
67
68
69  }
70
71  int main (int argc, char const *argv[])
72
73  {
74      int limitemenor= atoi(argv[1]);
75      int limitemayor= atoi(argv[2]);
76
77      int numaleatorio =numeroaleatorio(limitemenor,limitemayor);
78      printf("Numero aleatorio %d\n",numaleatorio);
79      ciclo(numaleatorio,limitemenor,limitemayor);
80
81      return 0;
82  }
```

Para este código se tiene la diferencia con los pasados en cuanto a que se eliminó la pista de mayor y menor. Para definir la nueva pista se define mediante el porcentaje, de modo que si el numero se encuentra en el rango del 95 % al usuario le va a aparecer un hirviendo, para el caso de caliente es del 70 %, frio 50 % y es inferior al 50 % ya saldría helado. Para ello respecto a hirviendo,caliente y frío se calculan los mismos párametros variando únicamente el porcentaje.Para sacar el valor de rangohirviendo como se ve en la linea 23, se multiplica el numaleatorio por 0,95 que seria el del porcentaje respectivo y se le resta el numaleatorio. Posteriormente para obtener los parametro menor hirviendo y maypr hirviendo lo que se hace es restarles o sumarle el rangohirviendo al numaleatorio. Cabe mencionar que también se realiza una variable llamada hirviendo. En los if dentro del while en vez de la pista de mayor y menor se sustituyen con la condicion de que num1 esté en los rangos respectivos. Como se sabe que si es hirviendo también pertenecería a caliente y frío, se utilizan las variables hirviendo, frío y caliente de modo que si entran en el if se aumenta su valor en 1, y por ejemplo en el if con intervalos para caliente se tiene que solo podrá imprimirse caliente si hirviendo es cero, y así se evita que se imprima también caliente en la condición de hirviendo.

Cabe mencionar que en cada ciclo estas variables hirviendo, caliente y frio se reinician a su valor 0.En la figura 1.2 se muestra un ejemplo cuando el usuario adivina y cuando el usuario se queda sin numero de intentos.

```
alberto@debian:~/laboratorio5/juego$ ./salida.x 20 30
Inserte un numero:20
Frio
Inserte un numero:25
Caliente
Inserte un numero:27
Caliente
Inserte un numero:29
correcto
```

**Figura 4:** Juego con modificación de pista

# 2. Comprender el funcionamiento de un Makefile

1. ¿Qué suelen contener las variables CC, CFLAGS, CXXFLAGS y LDFLAGS en un makefile?

- CC: programas para compilar en C
- CFLAGS: banderas extra para C
- CXXFLAGS: banderas extra para el preprocesador
- LDFLAGS: banderas extra para linker o enlazador para la ejecución

2. ¿De qué se compone una regla en un Makefile?
Una regla se compone de un target, un prerrequisito y comandos.

3. Defina qué es un target y cómo se relaciona con sus prerequisitos.
El target es un rótulo que generalmente se refiere a un nombre de archivo. Este archivo es aquel que queremos actualizar con esta regla. También hay targets que indican la acción que la regla realiza.La regla hace un target o lo actualiza en caso de que sea más viejo que el prerequisito, hay que hacer todos los prerequisitos y correr todos los comandos, de esta manera se relacionan.

4. ¿Para qué se utiliza la bandera -I, -c y -o del compilador gcc?

- -I agrega directorio de los encabezados de los archivos
- -c compila archivos fuente sin vincularlos
- -o escribe la salida de compilación en un archivo de salidas

5. ¿Cómo se definen y se utilizan las variables en un Makefile?¿Qué utilidad tienen?
Una variable se define utilizando la sintaxis nombre=valor. Las variables facilitan la escritura y evitan repetir listas de palabras iguales en líneas distintas.Ademas que facilitan el

mantenimiento de un makefile.

6. ¿Qué utilidad tiene un @ en un Makefile?
El @ es el nombre del archivo del target de la regla. Si el objetivo es un miembro de archivo, entonces @ es el nombre del archivo. En una regla de patrón que tiene múltiples targets,@ es el nombre del objetivo que causó la ejecución de la receta de la regla.

7. ¿Para qué se utiliza .PHONY en un Makefile?
Un phony target es uno que no es realmente el nombre de un archivo; más bien es solo un nombre para que una receta se ejecute cuando realiza una solicitud explícita. Hay dos razones para usar un phony target: para evitar un conflicto con un archivo del mismo nombre y para mejorar el rendimiento.

# 3. Crear un Makefile para el proyecto

Para esta sección se genera un archivo llamado Makefile y dentro del contenido esta:

```
1 build:
2         gcc -o juego.x Adivina.c
3 run:
4         ./juego.x 20 0
5 all: build run
6
7 clean:
8         rm -f *.o *.x
```