///////////////////////// 1. PROGRAMA 11 ////////////////////////////////////

#include <stdio.h>

/\*

11. Que pida dos números y muestre todos los números que van desde el primero al segundo.

Se debe controlar que los valores son correctos.

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

// Esta función regresa 0 si los valores son válidos y regresa 1 si no lo son

*int* verificar\_numeros(*int* x, *int* y) {

*int* valido;

if (x == y) {

printf("Los valores no pueden ser iguales\n\n");

valido = 1;

}

else if (x > y) {

printf("El segundo valor debe ser mas grande que el primero\n\n");

valido = 1;

}

else {

printf("Valores validos!\n\n");

valido = 0;

}

return valido;

}

*void* imprimir\_valores (*int* x, *int* y) {

printf("Valores del %d al %d\n\n", x, y);

for (*int* i = x; i <= y; i++)

printf("%d\n", i);

}

*int* main (*void*) {

*int* x, y;

*int* validos = 1;

while (validos != 0) { //mientras los datos no sean válidos, los seguirá pidiendo

x = obtener\_dato();

y = obtener\_dato();

validos = verificar\_numeros(x, y);

}

imprimir\_valores(x, y);

return 0;

}

///////////////////////// 2. PROGRAMA 12 ////////////////////////////////////

#include <stdio.h>

/\*

12. Que pida dos números y muestre todos los números pares que van desde el primero al segundo.

Se debe controlar que los valores son correctos.

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

// Esta función regresa 0 si los valores son válidos y regresa 1 si no lo son

*int* verificar\_numeros(*int* x, *int* y) {

*int* valido;

if (x == y) {

printf("Los valores no pueden ser iguales\n\n");

valido = 1;

}

else if (x > y) {

printf("El segundo valor debe ser mas grande que el primero\n\n");

valido = 1;

}

else {

printf("Valores validos!\n\n");

valido = 0;

}

return valido;

}

*void* imprimir\_valores (*int* x, *int* y) {

printf("Valores pares del %d al %d:\n\n", x, y);

for (*int* i = x; i <= y; i++)

if (i % 2 == 0)

printf("%d\n", i);

}

*int* main (*void*) {

*int* x, y;

*int* validos = 1;

while (validos != 0) { //mientras los datos no sean válidos, los seguirá pidiendo

x = obtener\_dato();

y = obtener\_dato();

validos = verificar\_numeros(x, y);

}

imprimir\_valores(x, y);

return 0;

}

///////////////////////// 3. PROGRAMA 13 ////////////////////////////////////

#include <stdio.h>

/\*

13. Que pida dos números y muestre todos los números impares que van desde el primero al segundo.

Se debe controlar que los valores son correctos.

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

// Esta función regresa 0 si los valores son válidos y regresa 1 si no lo son

*int* verificar\_numeros(*int* x, *int* y) {

*int* valido;

if (x == y) {

printf("Los valores no pueden ser iguales\n\n");

valido = 1;

}

else if (x > y) {

printf("El segundo valor debe ser mas grande que el primero\n\n");

valido = 1;

}

else {

printf("Valores validos!\n\n");

valido = 0;

}

return valido;

}

*void* imprimir\_valores (*int* x, *int* y) {

printf("Valores impares del %d al %d:\n\n", x, y);

for (*int* i = x; i <= y; i++)

if (i % 2 == 1)

printf("%d\n", i);

}

*int* main (*void*) {

*int* x, y;

*int* validos = 1;

while (validos != 0) { //mientras los datos no sean válidos, los seguirá pidiendo

x = obtener\_dato();

y = obtener\_dato();

validos = verificar\_numeros(x, y);

}

imprimir\_valores(x, y);

return 0;

}

///////////////////////// 4. PROGRAMA 19 ////////////////////////////////////

#include <stdio.h>

/\*

19. Que muestre la tabla de multiplicar de un número cualquiera.

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

*void* imprimir\_tabla (*int* x) {

printf("Tabla de multiplicar del %d:\n\n", x);

for (*int* i = 0; i <= 10; i++)

printf("%d \t\* %d \t= %d\n", x, i, x \* i);

}

*int* main (*void*) {

*int* x = obtener\_dato();

imprimir\_tabla(x);

return 0;

}

///////////////////////// 5. PROGRAMA 23 ////////////////////////////////////

#include <stdio.h>

/\*

23. Que calcule la media de X números, se dejarán de solicitar números hasta que se introduzca el cero.

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

*float* media (*int* acumulado, *int* n) {

*float* media = (*float*) acumulado / n;

return media;

}

*void* imprimir (*int* acumulado, *int* n, *float* media) {

printf("Su total acumulado fue de %d obtenido de %d numeros:\n\n", acumulado, n);

printf("Su media es: %f\n", media);

}

*int* main (*void*) {

*int* acumulado = -1;

*int* total\_de\_numeros = -1;

*int* nuevo\_numero = 1;

*float* promedio = 0;

/\*Inicia el acumulado en -1 para que entre en el ciclo y ahí se sumará el valor de nuevo\_numero, en

la primera iteración el acumulado y el total de números se igualarán a 0 (-1 + 1)

y a partir de ahí se pedirán nuevos números hasta que se dé un 0\*/

while (nuevo\_numero != 0) {

acumulado += nuevo\_numero;

total\_de\_numeros++;

nuevo\_numero = obtener\_dato();

}

promedio = media(acumulado, total\_de\_numeros);

imprimir(acumulado, total\_de\_numeros, promedio);

return 0;

}

///////////////////////// 6. PROGRAMA 24 ////////////////////////////////////

#include <stdio.h>

#include <math.h>

/\*

24. Que calcule la suma de los cuadrados de los 100 primeros números.

\*/

//Toma un número y regresa ese número elevado al cuadrado

*double* cuadrado (*double* x) {

return pow(x, 2.0);

}

*int* main (*void*) {

*double* acumulado = 0;

for (*double* i = 0; i <= 100; i++)

acumulado += cuadrado(i);

printf("La suma de los cuadrados de los primeros 100 números es: %lf\n", acumulado);

return 0;

}

///////////////////////// 7. PROGRAMA 31 ////////////////////////////////////

#include <stdio.h>

#include <math.h>

/\*

31.- Considere siguiente proceso repetitivo para un número entero dado:

si el número es 1, el proceso termina.

De lo contrario, si es par se divide entre 2,

y si es impar se multiplica por 3 y se le suma 1.

Si empezamos con 6, por ejemplo, obtendremos la sucesión de números 6, 3, 10, 5, 16, 8, 4, 2, 1.

La conjetura de Collatz dice que, a partir de cualquier número inicial,

la sucesión obtenida siempre termina en 1.

Escriba un programa que permita verificar la conjetura de Collatz para cualquier entero dado,

y que imprima la secuencia correspondiente.

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

/\*

La función entra con un número x, lo imprime y después condiciona:

Si el número es uno, la función terminará imprimiendo éxito,

si no lo es decidirá si el número es par o impar

y llamará a la función misma con el valor indicado por Collatz (la mitad si es par o 3 veces más 1 si es impar)

\*/

*void* collatz (*int* x) {

printf("%d\t", x);

if (x == 1)

printf("\n\n Fin de la ejecucion! Conjetura probada con exito!\n");

else {

if (x % 2 == 0)

collatz (x / 2);

else

collatz (x \* 3 + 1);

}

}

*int* main (*void*) {

*int* x;

x = obtener\_dato();

collatz(x);

return 0;

}

///////////////////////// 8. PROGRAMA 32 ////////////////////////////////////

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*

32. El juego de dados conocido como “craps” (tiro perdedor) es muy popular,

realice un programa que simule dicho juego, a continuación se muestran las reglas para los jugadores.

• Un jugador tira dos dados. Cada dato tiene seis caras. Las caras contienen 1, 2, 3, 4, 5 y 6 puntos.

• Una vez que los dados se hayan detenido, se calcula la suma de los puntos en las dos caras superiores.

• Si a la primera tirada, la suma es 7, o bien 11, el jugador gana.

• Si a la primera tirada la suma es 2, 3 o 12 (conocido como “craps”), el jugador pierde (es decir la casa “gana”).

• Si a la primera tirada la suma es 4, 5, 6, 8, 9 ó 10, entonces dicha suma se convierte en el “punto” o en la “tirada”.

• Para ganar, el jugador deberá continuar tirando los dados hasta que haga su “tirada”.

• El jugador perderá si antes de hacer su tirada sale una tirada de 7.

\*/

*int* dado () {

*int* random = (*int*) (rand () % 6) + 1;

return random;

}

/\*

Desglose de Estado:

Victoria = 1

Derrota = -1

Juego continúa = 0

Puedes ganar si:

- En el turno 1 consigues sacar 7 u 11

- En cualquier otro turno sacas tu tirada

Puedes perder si:

- En el turno 1 sacas 2, 3 o 12

- En cualquier otro turno sacas 7

El juego continúa si:

- En el turno 1 la suma no es 7, 11, 2, 3 ò 12

- En cualquier otro turno no sacas ni tu tirada ni 7

\*/

*int* verificar\_estado (*int* turno, *int* dado1, *int* dado2, *int* tirada) {

*int* suma = dado1 + dado2;

*int* estado = 0;

if (turno == 1) { // Si es el primer turno

if (suma == 7 || suma == 11) //Victoria inmediata

estado = 1;

else if (suma == 2 || suma == 3 || suma == 12) //CRAPS = Derrota inmediata

estado = -1;

else //Juego continúa

estado = 0;

}

else { //Turnos siguientes

if (suma == 7)

estado = -1;

else if (suma == tirada)

estado = 1;

else

estado = 0;

}

return estado;

}

/\*

Esta función se encarga únicamente de imprimir en pantalla el estado del juego:

El turno

El valor de tus dados

La tirada que necesitas

La tirada que conseguiste

Un letrero que te avisa si ganaste, perdiste o el juego sigue

\*/

*void* imprimir\_estado (*int* estado, *int* turno, *int* dado1, *int* dado2, *int* tirada) {

printf("----------------------------------\n");

printf("\tTurno %d\n\n", turno);

printf(" Dado 1: %d\n", dado1);

printf(" Dado 2: %d\n", dado2);

printf(" Suma actual: %d\n\n", dado1 + dado2);

printf(" > Tu tirada: %d\n\n", tirada);

switch (estado) {

case -1:

printf(" Has perdido. Mas suerte a la proxima!");

break;

case 0:

printf(" Sigue jugando!", tirada);

break;

case 1:

printf(" Felicidades, has ganado!", tirada);

break;

}

printf("\n----------------------------------\n\n");

}

*int* main (*void*) {

srand(time (NULL)); //instrucción para generar números aleatorios con base en el tiempo actual

*int* turno;

*int* dado1, dado2;

*int* tirada;

*int* fin\_de\_juego;

turno = 1;

dado1 = dado();

dado2 = dado();

tirada = dado1 + dado2;

fin\_de\_juego = verificar\_estado(turno, dado1, dado2, tirada);

while (fin\_de\_juego == 0) {

imprimir\_estado(fin\_de\_juego, turno, dado1, dado2, tirada);

turno++;

dado1 = dado();

dado2 = dado();

fin\_de\_juego = verificar\_estado(turno, dado1, dado2, tirada);

}

imprimir\_estado(fin\_de\_juego, turno, dado1, dado2, tirada);

return 0;

}

///////////////////////// 9. PROGRAMA 38 ////////////////////////////////////

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*

38.- 10. Escribe un programa que reciba un número entero N del usuario e imprima el contorno de un

cuadrado de N x N de asteriscos. Ej.

>> 3

\*\*\*

\* \*

\*\*\*

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

*void* dibujar\_cuadrado(*int* x) {

for (*int* i = 0; i < x; i++) {

if (i == 0 || i == x - 1)

for (*int* j = 0; j < x; j++)

printf("\*");

else

for (*int* j = 0; j < x; j++)

if (j == 0 || j == x - 1)

printf("\*");

else

printf(" ");

printf("\n");

}

}

*int* main (*void*) {

*int* x = obtener\_dato();

dibujar\_cuadrado(x);

return 0;

}

///////////////////////// 10. PROGRAMA 48 ////////////////////////////////////

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

/\*

48. Escribe un programa que reciba un número entero N del usuario e imprima el contorno de

pirámide de N pisos de asteriscos. Ej.

>> 3

\*

\* \*

\*\*\*\*\*

\*/

//Pide un número al usuario y lo regresa

*int* obtener\_dato() {

*int* x;

printf ("Ingrese un numero: ");

scanf ("%d", &x);

return x;

}

//Verifica que el número sea positivo, regresa 0 si lo es, 1 si es no-positivo

*int* verificar\_valor (*int* x) {

*int* valido = 1;

if (x <= 0)

printf ("El numero debe de ser mayor que 0\n");

else

valido = 0;

return valido;

}

*void* imprimir\_rombo (*int* x) {

for (*int* i = 0; i < x; i++) {

for (*int* j = x - i - 1; j >= 0; j--)

printf(" ");

for (*int* j = 0; j < i \* 2 + 1; j++)

printf("\*");

printf("\n");

}

}

*int* main (*void*) {

*int* x, valido;

do {

x = obtener\_dato();

valido = verificar\_valor(x);

} while (valido != 0);

imprimir\_rombo(x);

return 0;

}