|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Karina García Morales |
| *Asignatura:* | Fundaments de programación |
| *Grupo:* | 22 |
| *No de Práctica(s):* | 9 |
| *Integrante(s):* | Mora de los Santos José Manuel |
|  |  |
|  |  |
| *Semestre:* | 2018-2 |
| *Fecha de entrega:* | 24 de abril del 2018 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Objetivo:**

Elaborar programas en C para la resolución de problemas básicos que incluyan las estructuras de repetición y la directiva define.

**Conceptos**

*Mientras*

El ciclo mientras permite ejecutar un bloque de instrucciones mientras que una expresión lógica dada se cumpla, es decir, mientras su evaluación dé como resultado verdadero. La expresión lógica se denomina condición y siempre se evalúa antes de ejecutar el bloque de instrucciones. Si la condición no se cumple, el bloque no se ejecuta. Si la condición se cumple, el bloque se ejecuta, después de lo cual la instrucción vuelve a empezar, es decir, la condición se vuelve a evaluar.

En el caso en que la condición se evalué como falsa, el bloque de instrucciones no será ejecutado, lo cual quiere decir que el número de repeticiones o iteraciones de este bloque será cero. Si la condición siempre evalúa a verdadero, la instrucción se ejecutará indefinidamente, es decir, un número infinito de veces.

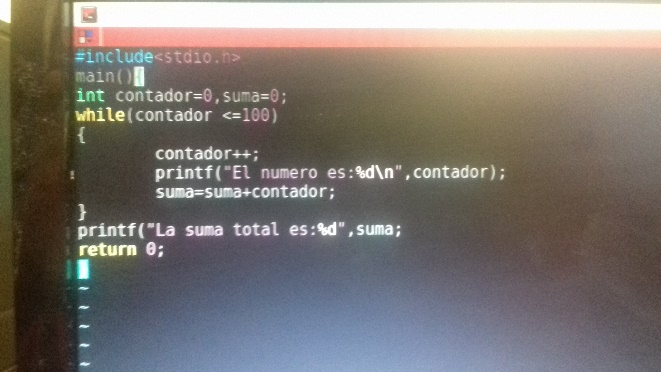
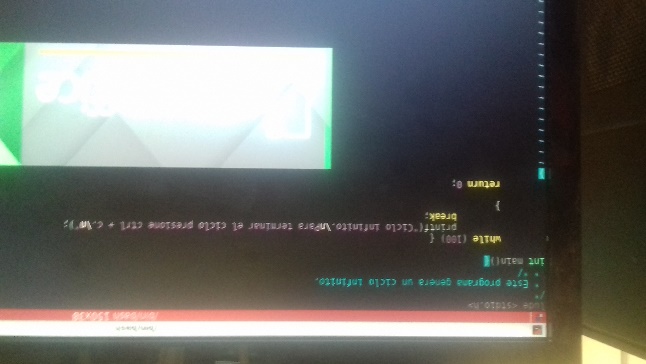
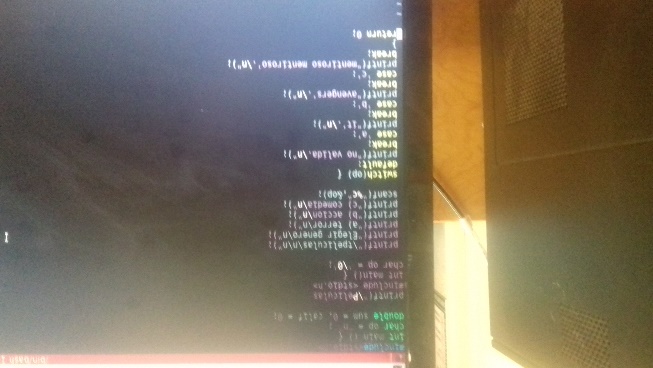
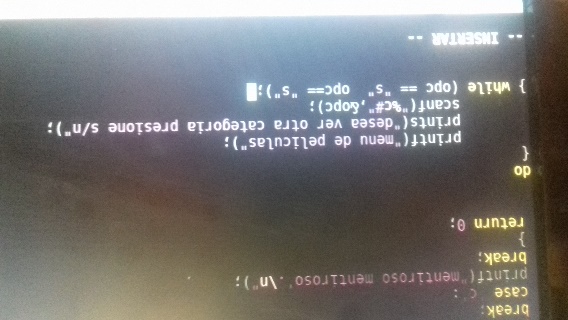
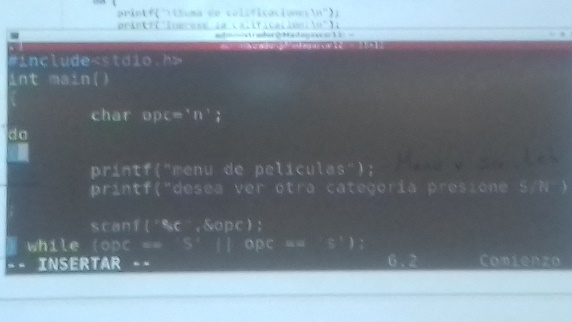
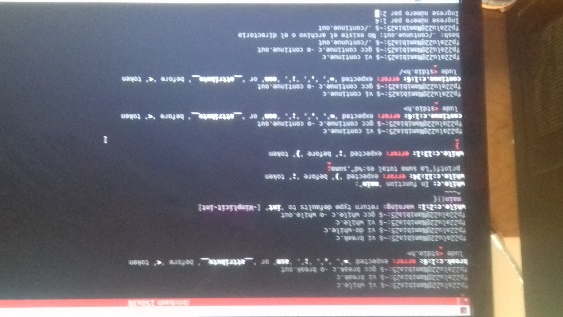
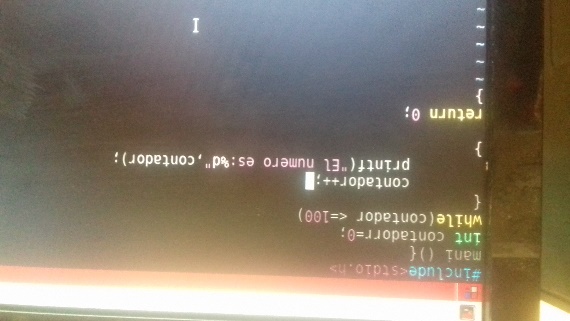
*Hacer mientras*

El ciclo hacer-mientras es similar al ciclo mientras, la diferencia radica en el momento de evaluación de la condición. En el ciclo hacer-mientras la condición se evalúa después de ejecutar el bloque de instrucciones, por lo tanto, el bloque se ejecuta por lo menos una vez. Este bloque se ejecuta nuevamente si la condición evalúa a verdadero, y no se ejecuta más si se evalúa como falso.

*Ciclo para*

El ciclo para ejecuta un bloque de instrucciones un número determinado de veces. Este número de veces está determinado por una variable contadora que toma valores desde un límite inferior hasta un límite superior. En cada ciclo después de ejecutar el bloque de instrucciones, la variable contadora es incrementada en 1 automáticamente y en el momento en que la variable sobrepasa el límite superior el ciclo termina. De la definición de ciclo para se puede inferir que el bloque de instrucciones no se ejecuta si el límite inferior es mayor al límite superior.

**Desarrollo de la practica**



**Ejercicios propuestos**

**Ejercicio 1**

#include <stdio.h>

int main(){

char op;

do {

printf("\tMenu de peliculas\n\n");

printf("Elegir la opción deseada\n");

printf("a) Comedia\n");

printf("b) Terror\n");

printf("c) Accion\n");

scanf("%c",&op);

switch(op) {

default:

printf("Opción no válida.\n");

break;

case 'a':

printf("Ms. Deeds'.\n");

break;

case 'b':

printf("Evil dead'.\n");

break;

case 'c':

printf("Terminator'.\n");

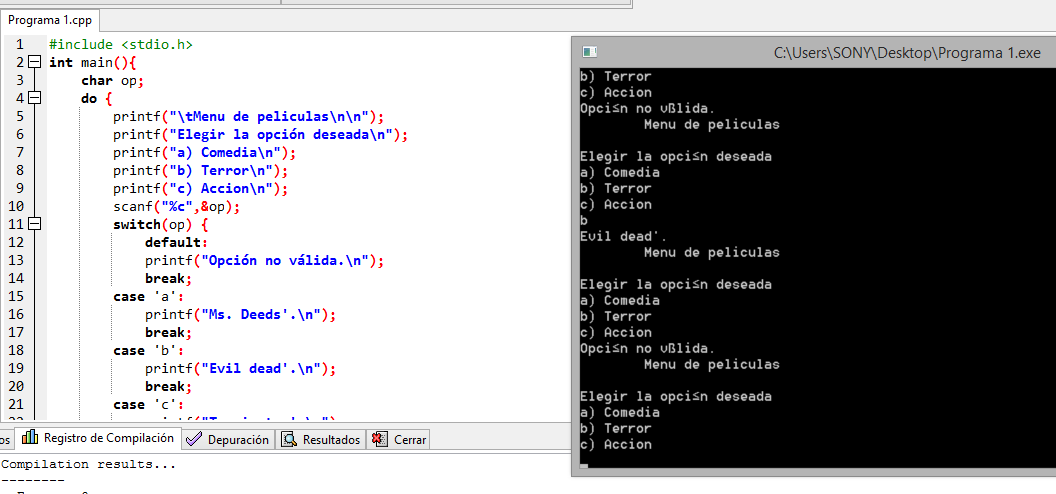
break;

}

} while (op != 3);

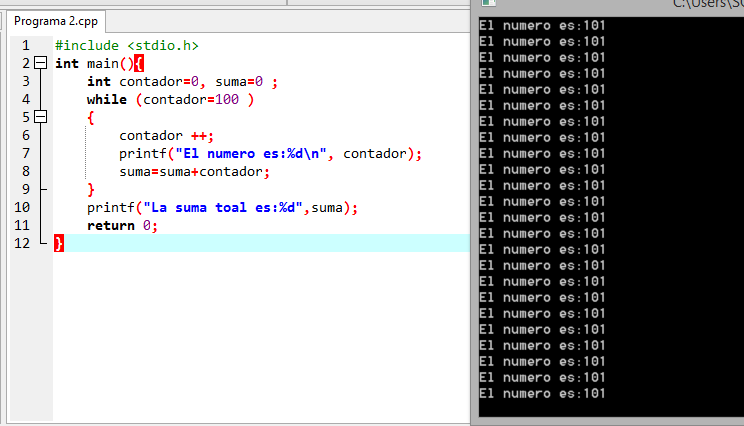
return 0;

}



**Ejercicio 2**

El programa solo produce el ultimo valor pues en lugar de indicar un rango establecemos un valor especifico y se atora con este valor.



#include <stdio.h>

int main(){

int contador=0, suma=0 ;

while (contador=100 )

{

contador ++;

printf("El numero es:%d\n", contador);

suma=suma+contador;

}

printf("La suma toal es:%d",suma);

return 0;

}

**Ejercicio 4**

#include <stdio.h>

int main(){

int op;

do {

printf("\tSignos zodiacales\n\n");

printf("Introducir numero\n");

printf("1) Aries\n");

printf("2) Tauro\n");

printf("3) Géminis\n");

printf("4) Cáncer\n");

printf("5) Leo\n");

printf("6) Virgo\n");

printf("7) Libra\n");

printf("8) Escorpión\n");

printf("9) Sagitario\n");

printf("10) Capricornio\n");

printf("11) Acuario\n");

printf("12) Piscis\n");

scanf("%c",&op);

switch(op) {

default:

printf("Error.\n");

break;

case '1':

printf("Fuego'.\n");

break;

case '2':

printf("Tierra'.\n");

break;

case '3':

printf("Aire'.\n");

break;

case '4':

printf("Agua'.\n");

break;

case '5':

printf("Fuego'.\n");

break;

case '6':

printf("Tierra '.\n");

break;

case '7':

printf("Aire'.\n");

break;

case '8':

printf("Agua'.\n");

break;

case '9':

printf("Fuego'.\n");

break;

case '10':

printf("Tierra'.\n");

break;

case '11':

printf("Aire'.\n");

break;

case '12':

printf("Agua'.\n");

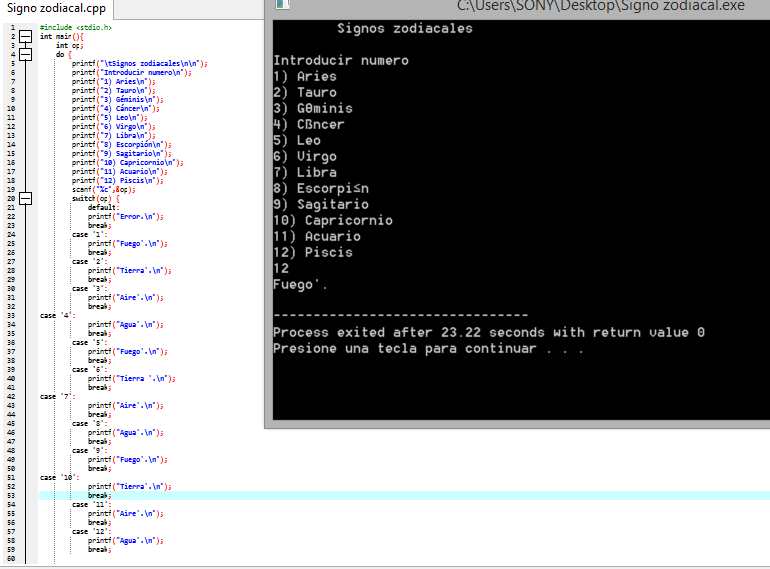
break;

}

} while (0<op>=12);

return 0;

}

****

**Ejercicio 4**

#include <stdio.h>

int main(){

char op;

int a, b, fact = 1;

printf("Introducir numero\n");

scanf("%d", &a);

for (b = a; b > 1; b--){

fact = fact \* b;

}

printf("El factorial de %d = %d\n", a, fact);

printf("\Desea repetir el proceso?\n");

printf("a) Si\n");

printf("b) No\n");

scanf("%c", &op);

switch(op) {

case 'a':

printf("si.\n");

break;

case 'b':

printf("No\n");

break;

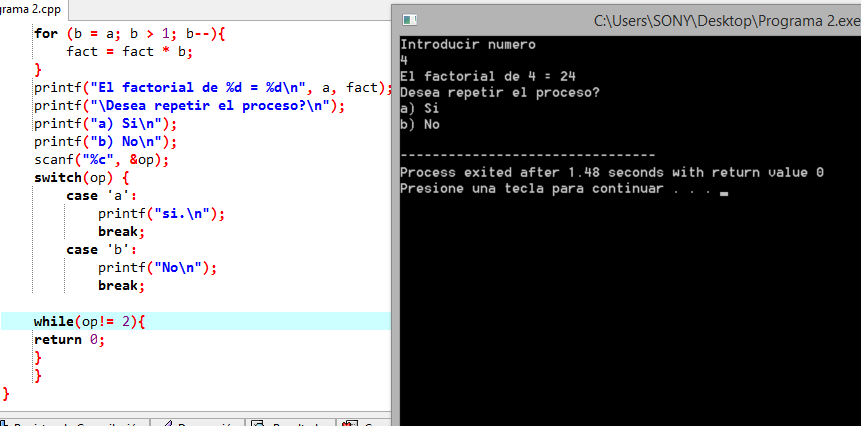
while(op!= 2){

return 0;

}

}

}



**Ejercicio 5**

Ninguna pues no se está comprobando ninguna expresión lógica, ni se están utilizando ninguna variable.

**Conclusiones**

Las estructuras repetitivas son muy utiles para sintetisar trabajo, pues siempre y cuando se establescan las condiciones adecuadas, el programa realizara su función por su cuenta hasta que se cumpla su objetivo, pero esto mismo su suele convertir en un obstáculo, pues cualquier error el o equivocación podría resultar en el programa no funcionando o funcionando incorrectamente, por ello es importante analizar el problema primero y en base a ese analizar comenzar con la elaboración del programa. Una vez logrado se notará la gran variedad de programas que podemos hacer para que lleven a cabo funciones por nosotros facilitando nuestro trabajo.

**Bibliografía**

* <https://www.fing.edu.uy/tecnoinf/mvd/cursos/prinprog/material/teo/prinprog-teorico04.pdf>
* http://dis.unal.edu.co/~programacion/book/modulo2b.pdf