T5LP45ATP2

Nombre: *Sánchez Bonaldo, Camila*

DO WHILE

Es una sentencia que implementa un bucle. Cuando se ejecute esta sentencia se ejecutarán primero las sentencias interiores (las que están entre el do y el while) y, tras hacerlo, se comprobará la condición especificada en el while y si ésta continúa siendo verdadera, se volverán a ejecutar las sentencias interiores. El proceso se repetirá hasta que la condición del while resulte falsa, continuando entonces la ejecución del programa tras esta sentencia.

```
do{
    sentencia ;
    ...
}while ( condición );
```

EJEMPLO

```
x=10
do{
    x=x+10
    printf("%d",x)

}while (x<100);
```

Se imprimirá el valor de x (el cual va en aumento de 10 en 10) mientras este sea menor que 100.

PROGRAMA 4- SALARIO DE UN TRABAJADOR CON SU NOMBRE

El programa presenta dos variables de tipo entera 'ht' y 'ch', tres variables de tipo flotante 't1', 't2' y 't3' y una de tipo carácter 'nombre', (esta cadena tiene una longitud máxima de 60 caracteres). Se imprime en pantalla "escriba su nombre: ", donde el usuario digita su nombre, el cual se guarda en la variable 'nombre'. Luego se muestra en pantalla "horas trabajadas: ", para que le usuario ingrese un número correspondiente que se guardará en la variable ´ht´. Después se imprime en pantalla "costo por hora: ", aquí el usuario digita el número correspondiente y este se almacena en 'ch'. Luego de estos procesos se establece que la variable t1 es igual al valor de 'ht' * el valor 'ch'.

Si 'ht' es menor o igual a 40 y t1 es menor o igual a 1200.00, se imprime en pantalla "nombre del trabajador:" y en la siguiente línea el nombre anteriormente digitado. Luego se muestra "horas trabajadas: ´ht´". En la siguiente línea se imprime "costo por hora: ´ch´". Luego se imprime "sueldo base: 182.00", en la próxima línea "sueldo con deducciones: 't1'", el valor de esta variable se mostrará con 2 decimales. Por último, se imprime en pantalla "sueldo total: ´t1´", al igual que en la línea anterior, esta variable también presenta 2 decimales.

Si no cumple con la condición anterior, pero si cumple con que ´ht´ sea mayor o igual a 40 y ´t1´ mayor o igual a 1200.00, se establece que la variable ´t2´ es igual al producto entre ´t1´ y 0.02 (esto quiere decir que t2 equivale al 2% de t1). Y también establece que la variable 't3' equivale a la diferencia entre ´t1´ y ´t2´. se imprime en pantalla "nombre del trabajador:" y en la siguiente línea el nombre anteriormente digitado. Luego se muestra "horas trabajadas: ´ht´". En la siguiente línea se imprime "costo por hora: ´ch´". Luego se imprime "sueldo base: ´t1´", en la próxima línea "sueldo con deducciones: 't2'". Por último, se imprime en pantalla "sueldo total: ´t3´". Tanto t1, t2 y t3, son mostrados en pantalla con dos decimales.

En el caso de que no cumpla con ninguna de las condiciones anteriores, pero cumple con que 'ht' es mayor o igual a 40 y 't1' es mayor a 1200.00, se establece como en la condición anterior, que ´t2´ es igual al producto entre ´t1´ y 0.02 y que 't3' equivale a la diferencia entre ´t1´ y ´t2´. En pantalla se imprime lo mismo que en la sentencia anterior.

Al finalizar este proceso la función **getch()** evita que se cierre automáticamente el programa, ya que este espera que se ingrese un carácter mediante el teclado antes de finalizarlo.

PROGRAMA 5- REPITE EL NOMBRE "N" NUMERO DE VECES

El programa presenta seis variables, cinco de tipo entero ('a', 'i', 'o', 'p' y 'r') y una de tipo carácter ('n'), esta cadena tiene una longitud máxima de 40 caracteres. Se imprime en pantalla "teclea TU NOMBRE ", la cadena introducida se guarda en la variable n. Luego se muestra "teclea cuantas veces quieres que se repita", el usuario digita un numero entero que se almacena en la variable r. Después de esto se despliega el mensaje "teclea con que op. Deseas que trabaje el programa" seguido de tres opciones "1= for 2=while 3= do while". El entero digitado se guarda en la variable o.

En caso de que el valor de la variable 'o' sea 1, se establece que 'a' es igual a 0. Luego se utiliza la sentencia for. En esta se establece el valor de 'i' (1), una condición ('i' menor o igual a 'r') y un incremento (i aumenta su valor de 1 en 1). Mientras se cumpla la condición se imprime en pantalla el nombre anteriormente ingresado y al valor de 'a' se le irá sumando el valor de 'i'.

En caso de que la opción elegida haya sido 2, se establece que 'a' es igual a 0 y que 'i' es igual a 1. Se utiliza la sentencia while y se establece que mientras 'i' sea menor o igual a 'r', se imprimirá el nombre anteriormente ingresado. Al valor de 'a' se le irá sumando el valor de 'i' y el valor de 'i' aumentará de 1 en 1.

En caso de que 'o' equivalga a 3, se establece que 'a' es igual a 0 y que 'i' es igual a 1. Se usa la sentencia do while. Dentro del bucle do, se imprime en pantalla el nombre anteriormente ingresado y al valor de 'a' se le va sumando el valor de 'i' y el valor de 'i' aumentará de 1 en 1. Mientras 'i' sea menor o igual a 'r'.

Si se ingresa un número que no corresponda con 1,2 o 3, se imprime en pantalla "opción incorrecta".

Después de que se ejecutara alguno de estos procesos, se muestran en pantalla dos opciones "para continuar presiona 1" o "para salir presiona 2". El numero ingresado se guarda en 'p' y luego se "limpia" la pantalla con **system("cls")**.

Todas estas sentencias se encuentran dentro de un bucle do while, entonces el programa las volverá a ejecutar mientras la condición ('p' es igual a 1) sea verdadera. Caso contrario, se cierra el programa.

PROGRAMA 6- TABLA DE MULTIPLICAR

Declara cinco variables de tipo entero, 'i'-'a'-'n'-'op'-'r'. Se imprime en pantalla "escriba un numero", luego de que el usuario digite un número y este se almacene en 'n', se muestra "escoja una estructura repetitiva", seguido de 3 opciones "1.-desde (for) 2.-mientras (while) 3.-hacer mientras (do while)". Una vez que se introduce el entero correspondiente, este se guarda en 'op'.

Si la opción elegida fue 1, se utiliza la sentencia for y se establece que el valor de 'i' es 1, la condición es 'i' menor o igual a 10 y el incremento de 'i' es de 1 en 1. Mientras la condición se cumpla se imprimirá en pantalla "'n'*'i'= 'valor de n*i'", de acuerdo a los valores correspondientes de las variables.

En caso de haber elegido 2, se establece que 'i' vale 1 y luego se utiliza la sentencia while. La condición dada es 'i' menor o igual que 10. Mientras esta resulte verdadera, se imprimirá en pantalla "'n'*'i'= 'valor de n*i'", de acuerdo a los valores correspondientes de las variables e 'i' incrementará su valor de 1 en 1.

En caso de que el valor de ‘op’ sea 3, se establece que el valor de ‘i’ es 1 y se utiliza la sentencia do while. se imprimirá en pantalla “’n’*’i’= ‘valor de n*i’”, de acuerdo a los valores correspondientes de las variables e ‘i’ incrementará su valor de 1 en 1. Mientras se cumpla la condición establecida en el while (‘i’ menor o igual a 10).

Si se digita una opción que no corresponda a 1,2 o 3, se imprime en pantalla “opción incorrecta”.

Después de que se ejecutara alguno de estos procesos, se muestran en pantalla dos opciones “1.-continuar 2.-salir”, el numero ingresado se almacena en la variable r y luego se “limpia” la pantalla con system (“cls”).

Todas estas sentencias se encuentran dentro de un bucle do while, entonces el programa las volverá a ejecutar mientras la condición (‘op’ es igual a 1) sea verdadera. Caso contrario, se cierra el programa.

MI PROGRAMA- EJERCITACION MATEMATICA

Utiliza funciones que como anteriormente no las explique las detallo a continuación.

RAND()

Esta función, cada vez que la llamamos, nos devuelve un número. SI no se establece un mínimo y un máximo, la función toma como mínimo a 0 y máximo RAND_MAX.

x=rand();	Este establece que el valor de la variable x es igual a un número aleatorio que se encuentra entre 0 y el máximo
x=10+rand()%20;	Este establece que el valor de la variable x es igual a un número aleatorio que se encuentra entre 10 y 20
x=rand()%50;	Este establece que el valor de la variable x es igual a un número aleatorio que se encuentra entre 0 y 50.

Para generar este número aleatorio la función parte de un número inicial llamado semilla. Si volvemos a ejecutar el programa desde el principio, la semilla que usa rand() es el mismo, con lo que la secuencia de números aleatorios es la misma

SRAND()

Esta función viene a solucionar el problema anterior y solo es necesario llamarla una vez en todo el programa. A esta función se le pasa un número que se utilizará como número inicial para rand() . No se debería poner un número fijo, porque entonces no habría cambio alguno. No se puede poner 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. Se debe buscar la forma de obtener un número que sea distinto en la ejecución de cada programa.

Generalmente se utiliza la fecha/hora del programa. Ya que este valor cambia si ejecutamos el programa en distinto instante de tiempo. Tendríamos que arrancar el programa dos veces en el mismo segundo para obtener la misma secuencia de números aleatorios.

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