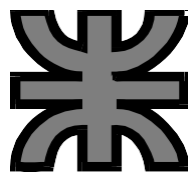


Universidad Tecnológica Nacional
Facultad Regional Resistencia
Técnico Superior en Programación

PROGRAMACIÓN I

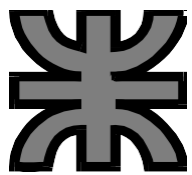
Sentencia
While



Ejemplo: Ingresar tres números y mostrarlos por pantalla

```
#include <stdio.h>
int main()
{
    int num;

    printf ("Ingrese una variable");
    scanf ("%d",&a);
    printf (" Numero ingresado: %d ",a);
    printf ("Ingrese una variable");
    scanf ("%d",&b);
    printf (" Numero ingresado: %d ",b);
    printf ("Ingrese una variable");
    scanf ("%d",&c);
    printf (" Numero ingresado: %d ",c);
    return 0;
}
```

Para el ejercicio planteado, se puede observar que por cada vez que se ingresa una variable se repiten las instrucciones printf y scanf. Así por ejemplo si se quisiera realizar la misma operación para 10 números, siguiendo este método de programación se debería escribir 30 líneas de código lo cual es ineficiente.

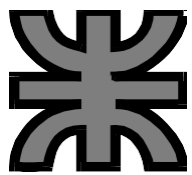
Para solución este problema se utilizan las ESTRUCTURAS CÍCLICAS O REPETITIVAS.

Las tres construcciones para realizar bucles son:

WHILE

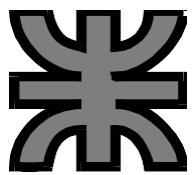
DO – WHILE

FOR



Estructuradas Repetitivas o cíclicas

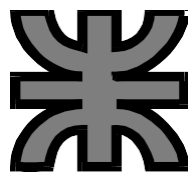
La estructura repetitiva se utiliza cuando se quiere que un conjunto de instrucciones se ejecuten un cierto número finito de veces. Se le llama bucle o ciclo a todo proceso que se repite un cierto número de veces dentro de un programa.



Estructuradas Repetitivas o cíclicas

Existen dos tipos de estructuras repetitivas;

- la primera es aquella en donde se tiene perfectamente establecido el número de veces que un grupo de acciones se van a ejecutar (20, 5, 2 veces). En este caso se utiliza un contador. En este caso el contador se puede incrementar como decrementar.
- y la segunda en la que el número de repeticiones es desconocido y se hará hasta que se cumpla o no cierta condición. Por ejemplo que un número sea mayor que cero.



Acciones Simples

- Contador: Es una variable que se incrementa, cuando se ejecuta, en una unidad o en una cantidad constante.

Ejemplo: **contador := contador + 1**
multiplo := multiplo + 3

- Acumulador: Es una variable que se incrementa en una cantidad variable.

Ejemplo: **suma := suma + numero**

Sentencia WHILE – (Mientras)

While primero evalúa la condición y si se cumple entra en el ciclo hasta que la condición no se cumpla.

Sintaxis

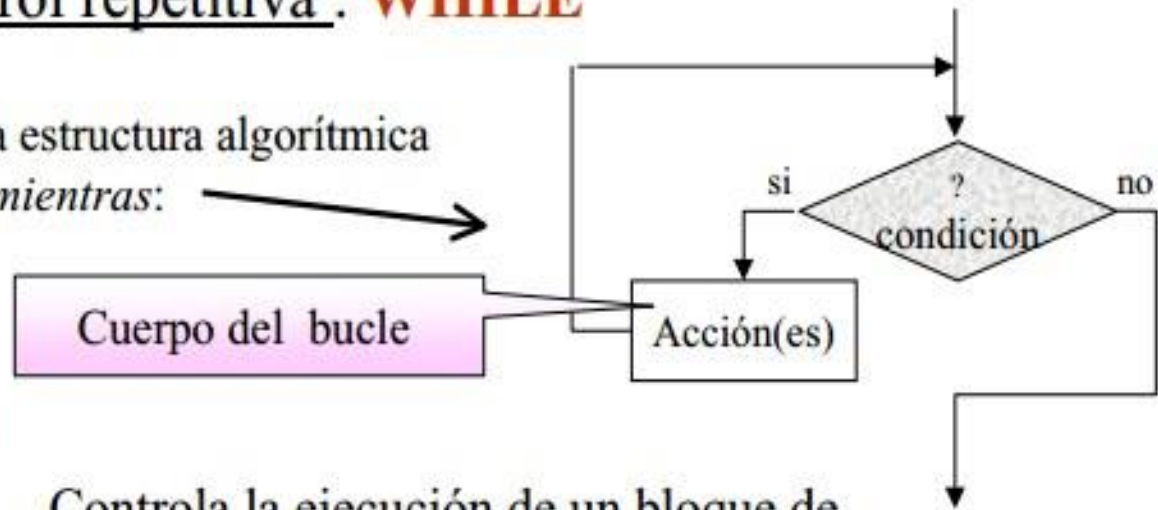
1 `while` (*condición-bucle*)
 Sentencia; \longrightarrow *cuerpo*

2 `while` (*condición-bucle*)
 {
 sentencia-1;
 sentencia-2;
 :
 :
 sentencia-n;
 } *cuerpo*

Estructuras Repetitivas o cíclicas

Instrucción de control repetitiva : WHILE

Se corresponde con la estructura algorítmica
hacer_mientras:



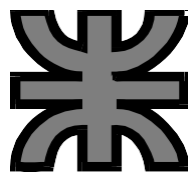
Formato general de la
sentencia **while**

```
while ( condición )  
{  
    instrucción 1;  
    ...  
    instrucción n;  
}
```

Controla la ejecución de un bloque de **instrucciones** de tal forma que éstas **se ejecutan mientras se cumpla la condición**, que será evaluada siempre **antes** de cada repetición.

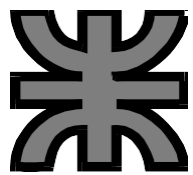
```
while ( condición )  
    instrucción;
```

Cada repetición del cuerpo
del bucle se denomina
iteración



Funcionamiento Sentencia WHILE – (Mientras)

1. Evalúa la condición
2. Si la condición es verdadera ejecuta el bloque de sentencias y vuelve a paso 1.
3. En caso contrario pasa a ejecutar la sentencia que se encuentra después del cuerpo.



Funcionamiento Sentencia WHILE – (Mientras)

Instrucción de control repetitiva : **WHILE**

Inicialización

Se realiza antes de la instrucción **while**

```
...  
contador = 0;  
while (contador < 100)  
{  
    cout << "Hola";  
    contador ++;  
}  
...
```

cuerpo

Actualización

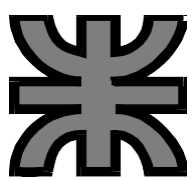
Se realiza dentro del cuerpo del bucle durante cada iteración

La variable que representa la condición del bucle se denomina *variable de control del bucle*. Dicha variable debe ser:

- inicializada
- comprobada
- actualizada

Comprobación

Se comprueba el valor de la variable antes de comenzar la repetición



Ejemplo: Ingresar tres números y mostrarlos por pantalla usando WHILE

```
# include <stdio.h>
# include <conio.h>
```

```
int main()
{
```

```
    int cont,acum,a;
    cont=0; /* inicializa contador*/
    acum=0; /* inicializa acumulador*/
```

```
    while (cont <3)
```

```
    {
```

```
        printf ("Ingrese un valor");
        scanf ("%d",&a);
        printf (" Numero ingresado:%d\n ",a);
        cont=cont+1; /* incrementa contador*/
        acum=acum+a;
```

```
    }
```

```
    return 0;
```

```
    printf(" El valor acumulado es :%d\n",acum);
}
```

En este problema se tiene perfectamente establecido el número de veces que un grupo de acciones se van a ejecutar. En este caso tres veces.

La condición será evaluada antes de cada iteración

El cuerpo del bucle while se ejecuta mientras la condición sea verdadera

El contador y el acumulador se actualiza en cada iteración

¿Qué hace el siguiente código?

```
int main()
{
    int a,b,c;
    printf("\n Introduce un número: ");
    scanf("%d",&a);
    b=1;
    while (b <= 10)
    {
        c=a*b;
        printf (" %d x %d es = %d \n",a,b,c);
        b=b+1;
    }
    return 0;
}
```

Comprobación o Prueba de Escritorio

a	b	c
4	1	4
4	2	8
4	3	12
4	4	16
4	5	20
4	6	24
..
4	10	40

Ejercicio



Calcular el perímetro de un cuadrado solamente cuando el valor ingresado sea correcto. La verificación se realizará solo tres veces. En caso de que se supere las tres posibilidades se mostrará por pantalla un mensaje explicando el error. Usar sentencia While.


```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int p,a,c;
```

```
    p=0;
```

```
    a=0;
```

```
    c=0;
```

```
    while (a <= 0 && c < 3 )
```

```
    {
```

```
        printf ("Ingrese un valor positivo\n");
```

```
        scanf ("%d",&a);
```

```
        c++;
```

```
    }
```

```
    if ( c==3) printf("Ingreso tres veces un valor incorrecto para calcular el  
perímetro.Debe ingresar un valor positivo");
```

```
    else
```

```
    {
```

```
        p= a*4;
```

```
        printf ("Perimetro= %d\n",p);
```

```
    }
```

```
    return 0;
```

```
}
```

The logo for 'Zinjal' is located in the top right corner. It features the word 'Zinjal' in a stylized, blue, handwritten-style font with a red underline. The background of the logo is yellow.

```
# include <stdio.h>
# include <conio.h>
```

```
int main()
{
    int p,a;
    p=0;
    a=0;
    while (a <= 0)
    {
        printf ("Ingrese un valor positivo\n");
        scanf ("%d",&a);
    }
    p= a*4;
    printf ("Perimetro= %d\n",p);
    getch();
    return 0;
}
```

¿Qué dificultad
tiene este
programa?

Ejemplo: Calcular el perímetro de un cuadrado. Verificar que el lado ingresado sea correcto.

Zinjal

Calcular el perímetro de un cuadrado solamente cuando el valor ingresado sea correcto. Ingresar la cantidad de veces que se verificará que el valor ingresado sea correcto.

```
#include <stdio.h>
```

```
int main()  
{
```

```
    int p,a,b,i;  
    p=0;  
    a=0;  
    i=0;
```

```
    printf ("Ingrese la cantidad de veces que verificará el valor del lado\n");  
    scanf ("%d",&b);
```

```
    while (a <= 0 && i<b )  
    {  
        printf ("Ingrese un valor positivo\n");  
        scanf ("%d",&a);  
        i++;  
    }
```

```
    if (i==b) printf ("Supero la cantidad de intentos permitidos");  
    else  
    {  
        p= a*4;  
        printf ("Perimetro= %d\n",p);  
    }
```

```
    return 0;
```

```
}
```


Ejemplo: Calcular el perímetro de un cuadrado. Verificar que el lado ingresado sea correcto.

Zinjal

Modificar el programa anterior para que el usuario pueda realizar otro cálculo de perímetro si lo desea.

Ejemplo: Calcular el perímetro de un cuadrado. Verificar que el lado ingresado sea correcto.

Zinjal

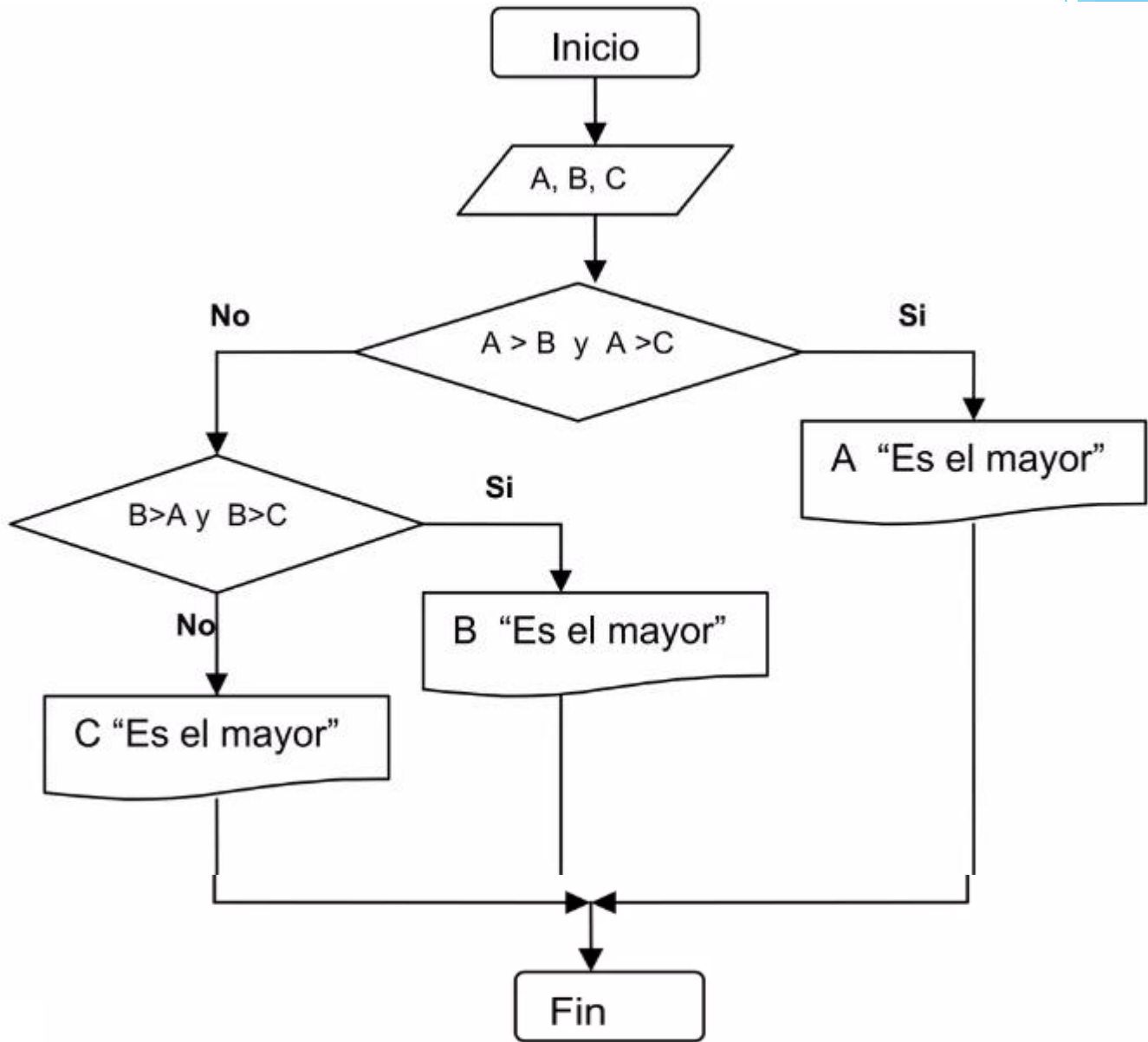
Realizar un programa que permita contar la cantidad de números pares ingresados por teclado.

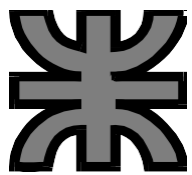
```
# include <stdio.h> # include  
<conio.h>
```

The logo for 'Zinjal' is located in the top right corner. It features the word 'Zinjal' in a stylized, blue, handwritten-style font with a red underline. The background of the logo is yellow.

```
int main()  
{  
    /* Escribir y contar los números pares de 10 números ingresados */  
  
    int num, c, d, nu; c=1;  
    d=0;  
    printf("Ingrese cantidad de numeros a ingresar\n");  
    scanf ("%d",&nu);  
  
    while(c<=nu)  
    {  
        printf("Ingrese un numero\n");  
        scanf ("%d",&num);  
        if (num % 2 ==0)  
        {  
            printf ("Numero:%d\n",num);  
            d=d+1;  
        }  
        c++;  
    }  
    if (d==0) printf ("No se ingresaron números pares\n");  
    else printf("Se ingresaron %d números pares \n",d);  
    getch();  
    return 0;  
}
```

Dado el programa realizado anteriormente, modificarlo para que el usuario pueda repetirlo si lo desea.





Responder las siguientes preguntas

1. La sentencia break ¿Se puede utilizar en una estructura repetitiva?
2. La sentencia continue ¿Se puede utilizar en una estructura repetitiva?
3. ¿Existe relación entre la sentencia break y la sentencia continue?
4. ¿Qué es un ciclo controlado por Bandera o cantinela? Ejemplificar, crear un enunciado y realizar el correspondiente programa.