UAEM | Universidad Autónoma del Estado de México

UNIDAD ACADÉMICA PROFESIONAL TIANGUISTENCO

LICENCIATURA EN PRODUCCIÓN INDUSTRIAL.

UNIDAD DE APRENDIZAJE: PROGRAMACIÓN

Créditos institucionales de la UA: 6
Material visual: Diapositivas

Unidad de competencia II

P R O G R A M A C I Ó N E S T R U C T U R A D A

Elaborado por M. en C. Selene Palacios Astudillo

Período 2015-A

¿Cómo emplear este material?

El presente material tiene como finalidad facilitar la exposición gráfica del tema “Programación Estructurada” que se aborda en la unidad de aprendizaje “Programación” que corresponde al primer semestre de la Licenciatura en Ingeniería en Producción Industrial.

La presentación deberá ir acompañada de una explicación oral del docente, ya que la aportación que pueda hacer mediante ejemplos y situaciones cotidianas brindará la oportunidad de que los estudiantes comprendan la importancia de construir argumentos sólidos, creíbles y bien soportados.

INTRODUCCIÓN A LA PROGRAMACIÓN ESTRUCTURADA

- ÍNDICE -

Tema	Diapositiva
Programación Estructurada	5
Sentencias Compuestas	9
Sentencias Repetitivas	16
Aplicaciones	24
Bibliografía	35

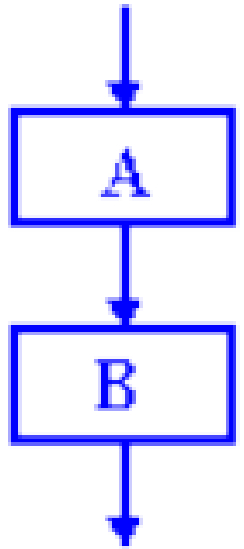
INTRODUCCIÓN A LA PROGRAMACIÓN ESTRUCTURADA

Objetivo de la Unidad Temática.

Al término de la unidad temática, el estudiante tendrá las bases para programar en forma estructurada, problemas simples usando sentencias compuestas y repetitivas.

Programación Estructurada

- C, lenguaje de programación que permite programar de manera estructurada.
- Establece una sintaxis y semántica propia para expresar operaciones.
- Ofrece sentencias de control para describir **secuencias**,

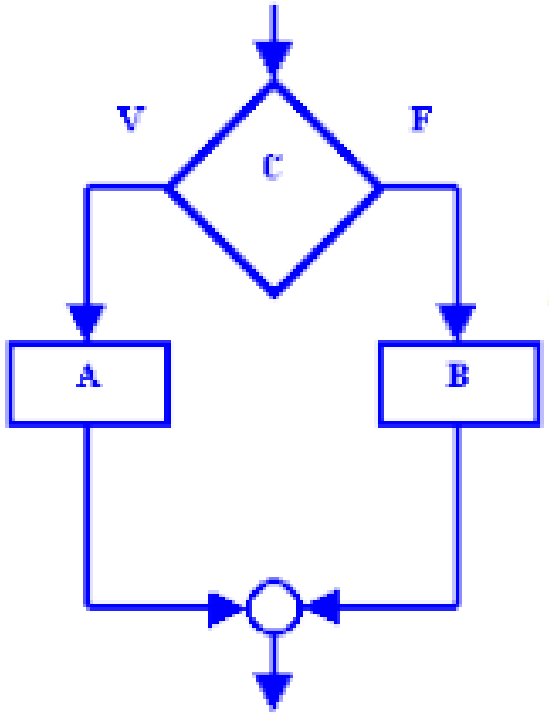


SECUENCIA

Programación Estructurada

- C, lenguaje de programación que permite programar de manera estructurada.
- Establece una sintaxis y semántica propia para expresar operaciones.
- Ofrece sentencias de control para describir

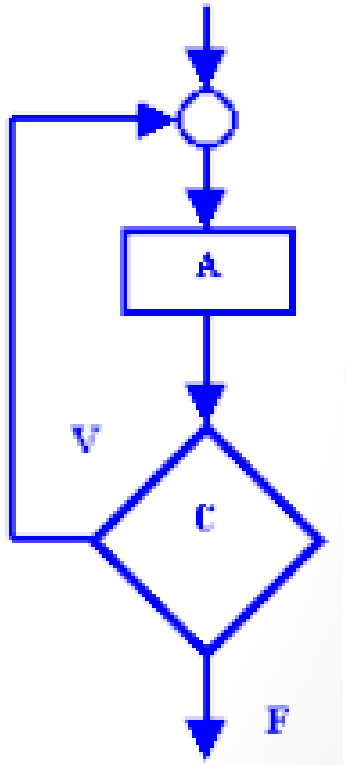
  secuencias, selecciones



SELECCIÓN
if ..else
switch

Programación Estructurada

- C, lenguaje de programación que permite programar de manera estructurada.
- Establece una sintaxis y semántica propia para expresar operaciones.
- Ofrece sentencias de control para describir **secuencias**, **selecciones, iteraciones**.



ITERACIÓN
while
do… while
for

Estructuras Secuenciales

- Son las sentencias (asignaciones e invocaciones a funciones) escritas en el código fuente, cada una de estas será ejecutada de manera secuencial (en orden de aparición) una vez compilado el código fuente

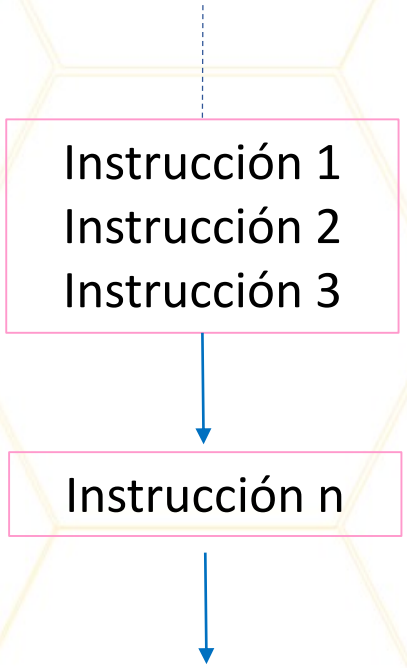


Diagrama de Flujo

```
Instrucción 1
Instrucción 2
Instrucción 3
.
.
.
Instrucción n
```

Pseudocódigo

```c
#include<stdio.h>
int main(void)
{
    int byte=0xFF;
    printf("\nElvalor de byte es: %4X H",byte);
    byte&=0x00;
    printf("\nElvalor de byte es: %4X H",byte);
    byte |=0xFF;
    printf("\nElvalor de byte es: %4X H",byte);
    byte>>=1;
    printf("\nElvalor de byte es: %4X H",byte);
    byte<<=2;
    printf("\nElvalor de byte es: %4X H",byte);
    return 0;
}
```

Código C

Estructuras Selectivas if {…}

Selectiva Simple

- Se utilizan para tomar una decisión en base a la evaluación de una expresión.

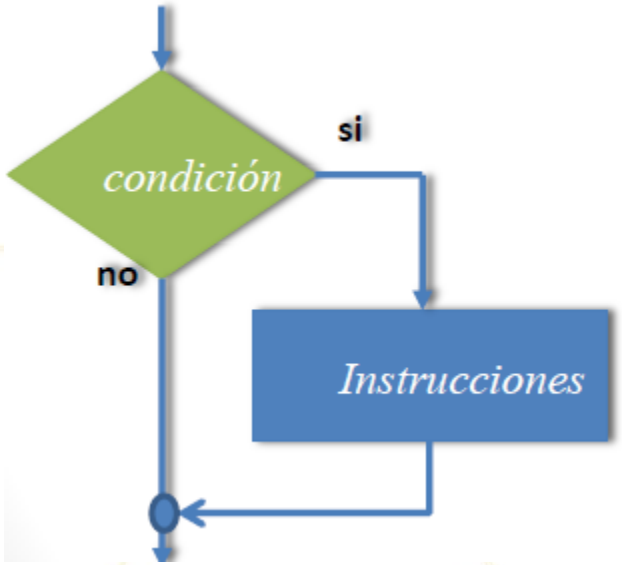


Diagrama de Flujo

```
Si condición Entonces
    Instrucciones
FinSi
```

Pseudocódigo

```
if(condición)
{
    Instrucciones;
}
```

Código C

Estructuras Selectivas if {…}

Selectiva doble

- Se utilizan para tomar una decisión en base a la evaluación de una expresión.

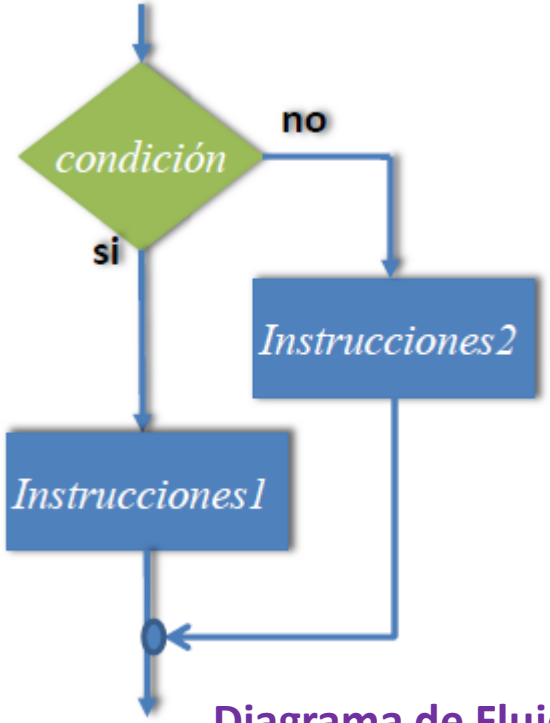


Diagrama de Flujo

```
Si condición Entonces
    Instrucciones 1
Sino
    Instrucciones 2
FinSi
```

Pseudocódigo

```
if(condición)
{
    Instrucciones1;
}
else
{
    Instrucciones2;
}
```

Código C

Estructuras Selectivas if {…}

Selectiva múltiple

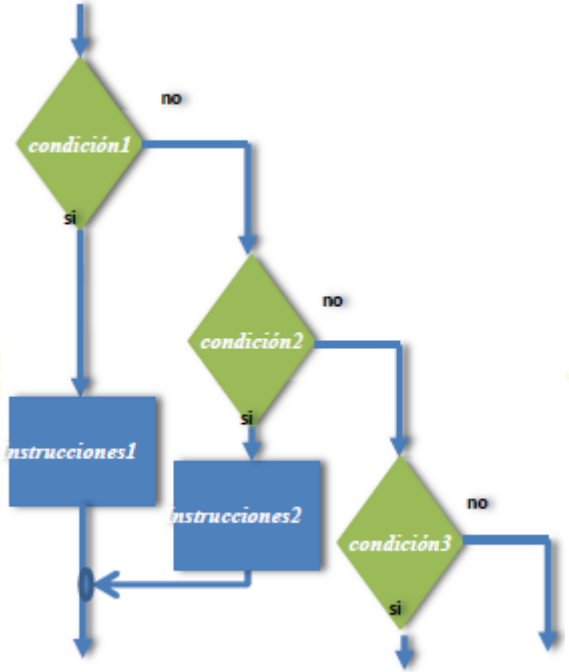


Diagrama de Flujo

```
Si condición1 Entonces
    Instrucciones 1
Sino
    Si condición2 Entonces
        Instrucciones 2
    Sino
        Si condición3 Entonces
                Instrucciones 3
        . . .
        Sino
            Instrucciones n
        FinSi
    FinSi
FinSi
```

Pseudocódigo

```
if(condición1)
{
    Instrucciones1;
}
else if (condición2)
{
    Instrucciones2;
}
else if (condición3)
{
    Instrucciones3;
}
…
else
{
    Instruccionesn;
}
```

Código C

Estructuras Selectivas if {…}

- La estructura selectiva if, se utiliza para expresar decisiones.
- La **sintaxis** es:

```
if(expresión1)
{
    sentencias1;
}
else if(expresión2)
{
    sentencias2;
}
else if(expresión3)
{
    sentencias3;
}
else
{
    sentencias4;
}
```

Estructuras Selectivas if {…}

- La estructura selectiva if, se utiliza para expresar decisiones.
- La **sintaxis** es:

```
if(expresión1)
{
    sentencias1;
}
else if(expresión2)
{
    sentencias2;
}
else if(expresión3)
{
    sentencias3;
}
else
{
    sentencias4;
}
```

Las sentencias **else if (sino si)** y **else (sino)** pueden omitirse

Si al evaluarse expresión1 da como resultado verdadero, entonces se lleva a cabo el código inmediato entre llaves.

Si no si (else if) la expresión2 se evalúa y da como resultado verdadero entonces se lleva a cabo el código inmediato entre llaves.

Si no si (else if) la expresión3 se evalúa y si esta da como resultado verdadero, entonces se lleva a cabo el código inmediato entre llaves.

…

Si no (else) si ninguna condición ha resultado verdadero entonces se lleva a cabo el código inmediato entre llaves.

Estructuras Selectiva switch

- Facilita implementar algunas decisiones múltiples cuando se presentan, todas con base en la evaluación del valor de una variable (expresión).

```
switch(expresión)
{
        case exp-constante1: <acción 1>;
                             break;
        case exp-constante2: <acción 1>;
                             break;
              ...
        case exp-constanteN: <acción N>;
                             break;
        default:             <acción M>;
}
```

Estructuras Selectiva switch

- Facilita implementar algunas decisiones múltiples cuando se presentan, todas con base en la evaluación del valor de una variable (expresión).

```
switch(expresión)
{
        case exp-constante1: <acción 1>;
                             break;
        case exp-constante2: <acción 1>;
                             break;
             ...
        case exp-constanteN: <acción N>;
                             break;
        default:             <acción M>;
}
```

La estructura selectiva **switch**, se utiliza bajo la teoría de la programación estructurada; incluye una sentencia de salto (break), que debe ser utilizada analíticamente.

Sentencias Repetitivas

Iterativa mientras

- La sentencia while, se utiliza para ejecutar más de una vez el mismo conjunto de instrucciones, con base a una condición.

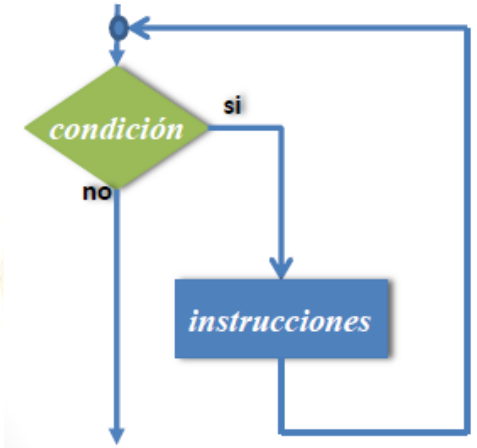


Diagrama de Flujo

```
mientras condición1  hacer
    Instrucciones
Fin mientras
```

Pseudocódigo

```
while(condición)
{
    Instrucciones;
}
```

Código C

Sentencias Repetitivas

Iterativa repetir

- La sentencia do…while, se asegura que las instrucciones dentro del ciclo, se ejecuten al menos una vez.

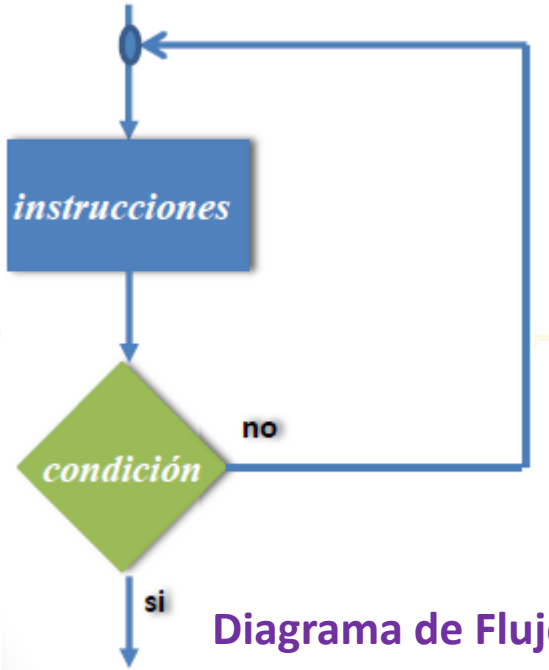


Diagrama de Flujo

```
repetir
    Instrucciones
hasta que  condición
```

Pseudocódigo

```
do
{
    Instrucciones;
}while(!(condición));
```

Código C

Estructura while *vs* do… while

```
while(condición)
{
    Instrucciones;
}
```

- ✓ La estructura de control **while**, evalúa la condición antes de iniciar cada iteración.
- ✓ Si al evaluar la condición, genera como resultado verdadero, se realizan las sentencias que están dentro de las llaves.

Estructura while *vs* do… while

```
do
{
    Instrucciones;
}while(!(condición));
```

- ✓ La estructura de control **do…while**, se utiliza, cuando se quiere asegurar que las sentencias que se encuentran dentro del ciclo se ejecuten al menos una vez; puesto que la evaluación de la condición (expresión lógica), se realiza al finalizar el ciclo.

Sentencias Repetitivas

Iterativa para

- La sentencia **for**, permite definir un bucle que es controlado por una variable de control o inducción, conocido como contador

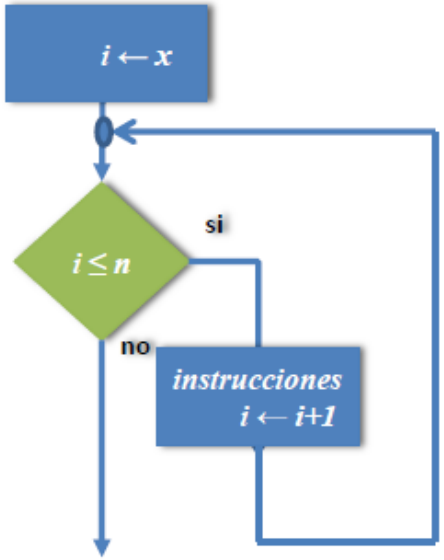


Diagrama de Flujo

```
Para i<- x hasta n hacer
    Instrucciones
fin para
```

Pseudocódigo

```
for(i=x;i<=n;i++)
{
    instrucciones;
}
```

Código C

Sentencias Repetitiva for

Estructura *for*

- El encabezado de un bucle **for**, se compone de tres partes, separadas por '**;**'.

```
for(i=x;i<=n;i++)
```

Se inicializan las variables de control y sólo se ejecuta una vez. Antes de la primera iteración.

Sentencias Repetitiva for

Estructura *for*

- El encabezado de un bucle **for**, se compone de tres partes, separadas por '**;**'.

```
for(i=x;i<=n;i++)
```

Expresión que indica la condición lógica, que debe cumplirse para que la próxima iteración se ejecute; esta condición se evalúa antes de cada iteración y, cuando deja de satisfacerse, el bucle for termina.

Sentencias Repetitiva for

Estructura *for*

- El encabezado de un bucle **for**, se compone de tres partes, separadas por '**;**'.

```
for(i=x;i<=n;i++)
```

Representa la actualización (incremento o decremento) de las variables de control y se ejecuta después de cada iteración.

Aplicaciones

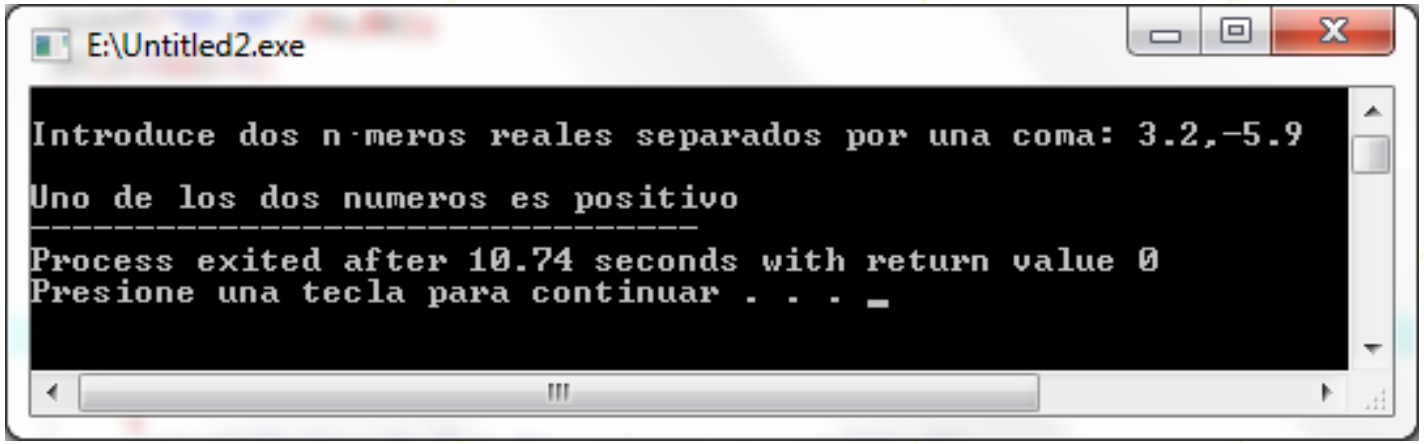
Ejemplo if… else

```c
#include<stdio.h>
int main(void)
{
    float a, b;
    printf("\nIntroduce dos números reales separados por una coma: ");
    scanf("%f,%f",&a,&b);
    if(a<0&&b<0)
    {
        printf("\nAmbos numeros son negativos");
    }
    else if(a>0&&b>0)
    {
        printf("\nLos dos numeros son positivo");
    }
    else if(a>0||b>0)
    {
        printf("\nUno de los dos numeros es positivo");
    }
    else if(a==0 && b==0)
    {
        printf("\nAmbos numeros son igual a cero");
    }
    else
    {
        printf("\n...");
    }
    return 0;
}
```

Aplicaciones

Ejemplo if… else

```
#include<stdio.h>
int main(void)
{
    float a, b;
    printf("\nIntroduce dos números reales separados por una coma: ");
    scanf("%f,%f",&a,&b);
    if(a<0&&b<0)
    {
        printf("\nAmbos numeros son negativos");
    }
    else if(a>0&&b>0)
    {
        printf("\nLos dos numeros son positivo");
    }
    else if(a>0||b>0)
    {
        printf("\nUno de los dos numeros es positivo");
    }
    else if(a==0 && b==0)
    {
        printf("\nAmbos numeros son igual a cero");
    }
    else
    {
        printf("\n...");
    }
    return 0;
}
```



Aplicaciones

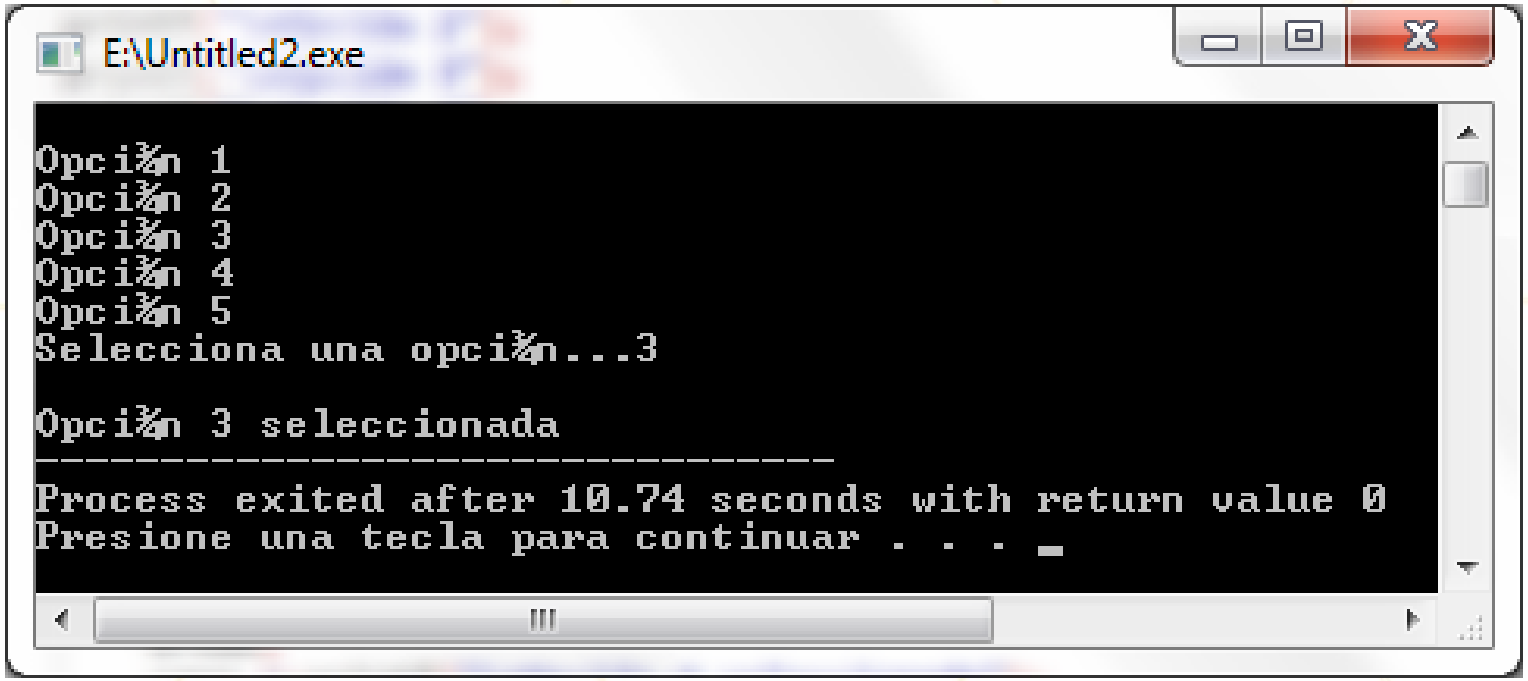
Ejemplo switch

```c
#include<stdio.h>
int main(void)
{
    int opcion;
    printf("\nOpción 1");
    printf("\nOpción 2");
    printf("\nOpción 3");
    printf("\nOpción 4");
    printf("\nOpción 5");
    printf("\nSelecciona una opción...");
    scanf("%d",&opcion);
    switch(opcion)
    {
        case 1:
            printf("\nOpción 1 seleccionada");
            break;
        case 2:
            printf("\nOpción 2 seleccionada");
            break;
        case 3:
            printf("\nOpción 3 seleccionada");
            break;
        case 4:
            printf("\nOpción 4 seleccionada");
            break;
        case 5:
            printf("\nOpción 5 seleccionada");
            break;
        default:
            printf("\nOpción no valida");
    }
    return 0;
}
```

Aplicaciones

Ejemplo switch

```
#include<stdio.h>
int main(void)
{
    int opcion;
    printf("\nOpción 1");
    printf("\nOpción 2");
    printf("\nOpción 3");
    printf("\nOpción 4");
    printf("\nOpción 5");
    printf("\nSelecciona una opción...");
    scanf("%d",&opcion);
    switch(opcion)
    {
        case 1:
            printf("\nOpción 1 seleccionada");
            break;
        case 2:
            printf("\nOpción 2 seleccionada");
            break;
        case 3:
            printf("\nOpción 3 seleccionada");
            break;
        case 4:
            printf("\nOpción 4 seleccionada");
            break;
        case 5:
            printf("\nOpción 5 seleccionada");
            break;
        default:
            printf("\nOpción no valida");
    }
    return 0;
}
```



Aplicaciones

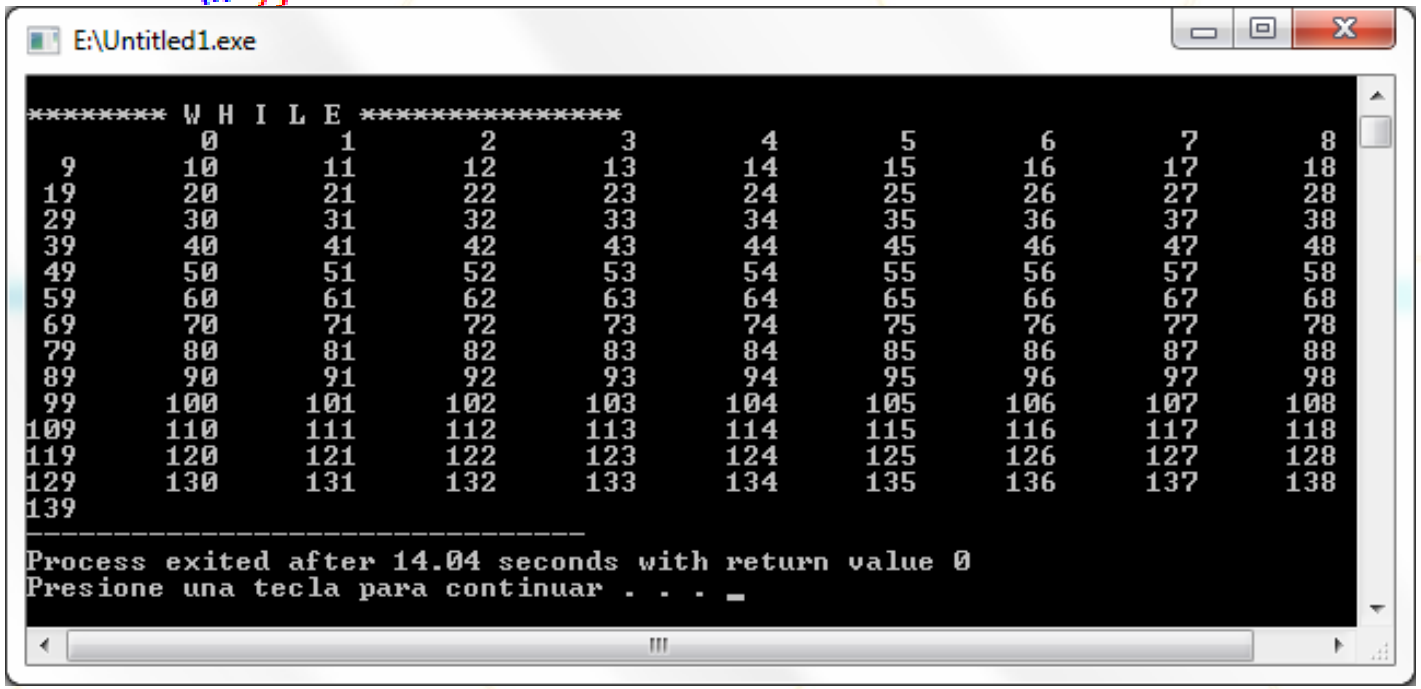
Ejemplo while

```
#include<stdio.h>
int main(void)
{
    int i=0,n=10000;
    printf("\n******** W H I L E ***************\n");
    while (i<n)
    {
        printf("\t%3d", i);
        i++;
        n=n-i;
    }
    return 0;
}
```

Aplicaciones

Ejemplo while

```
#include<stdio.h>
int main(void)
{
    int i=0,n=10000;
    printf("\n******** W H I L E ***************\n");
    while (i<n)
    {
        printf("\t%3d", i);
        i++;
        n=n-i;
    }
    return 0;
}
```



Aplicaciones

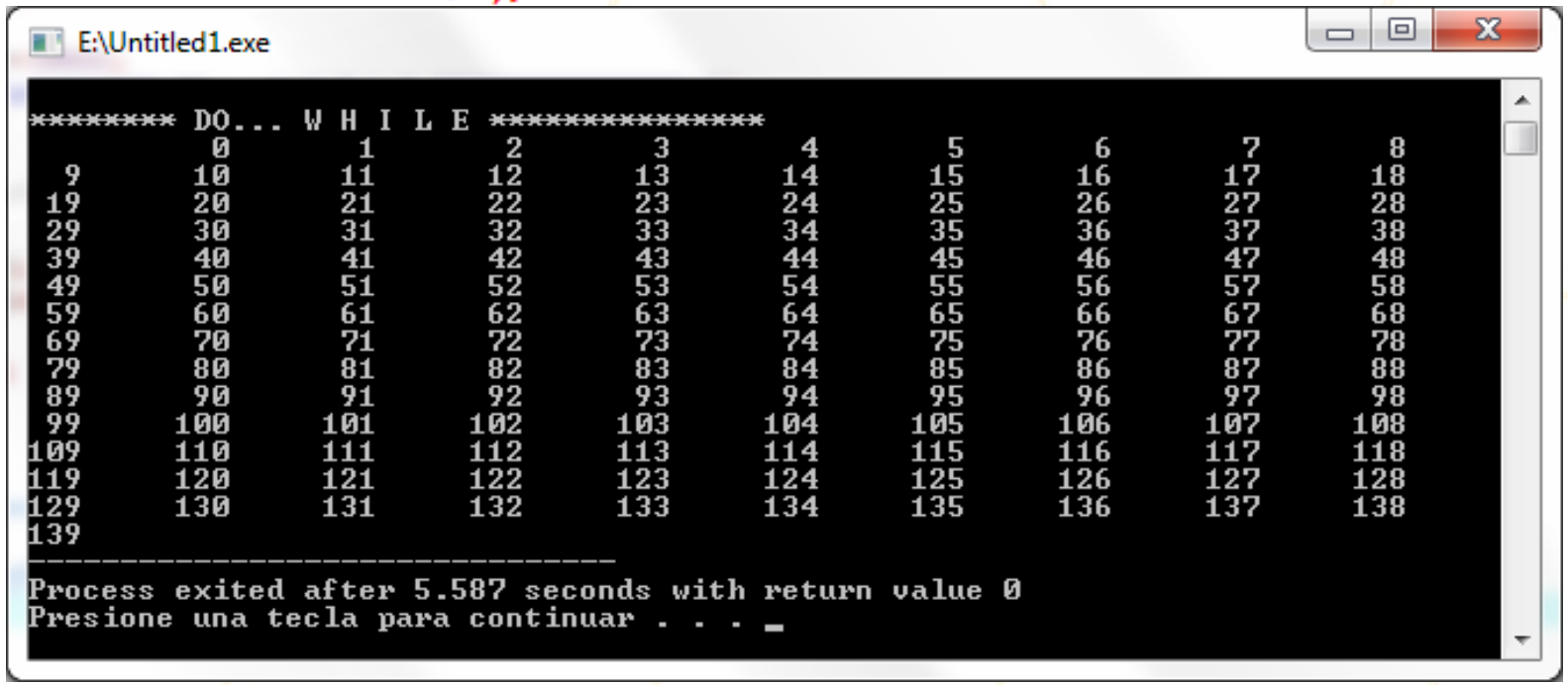
Ejemplo do... while

```
#include<stdio.h>
int main(void)
{
    int i=0,n=10000;
    printf("\n******** DO... W H I L E ***************\n");
    do
    {
        printf("\t%3d", i);
        i++;
        n=n-i;
    }while (i<n);

    return 0;
}
```

Aplicaciones

Ejemplo do... while

```
#include<stdio.h>
int main(void)
{
    int i=0,n=10000;
    printf("\n******** DO... W H I L E ***************\n");
    do
    {
        printf("\t%3d", i);
        i++;
        n=n-i;
    }while (i<n);

    return 0;
}
```



Aplicaciones

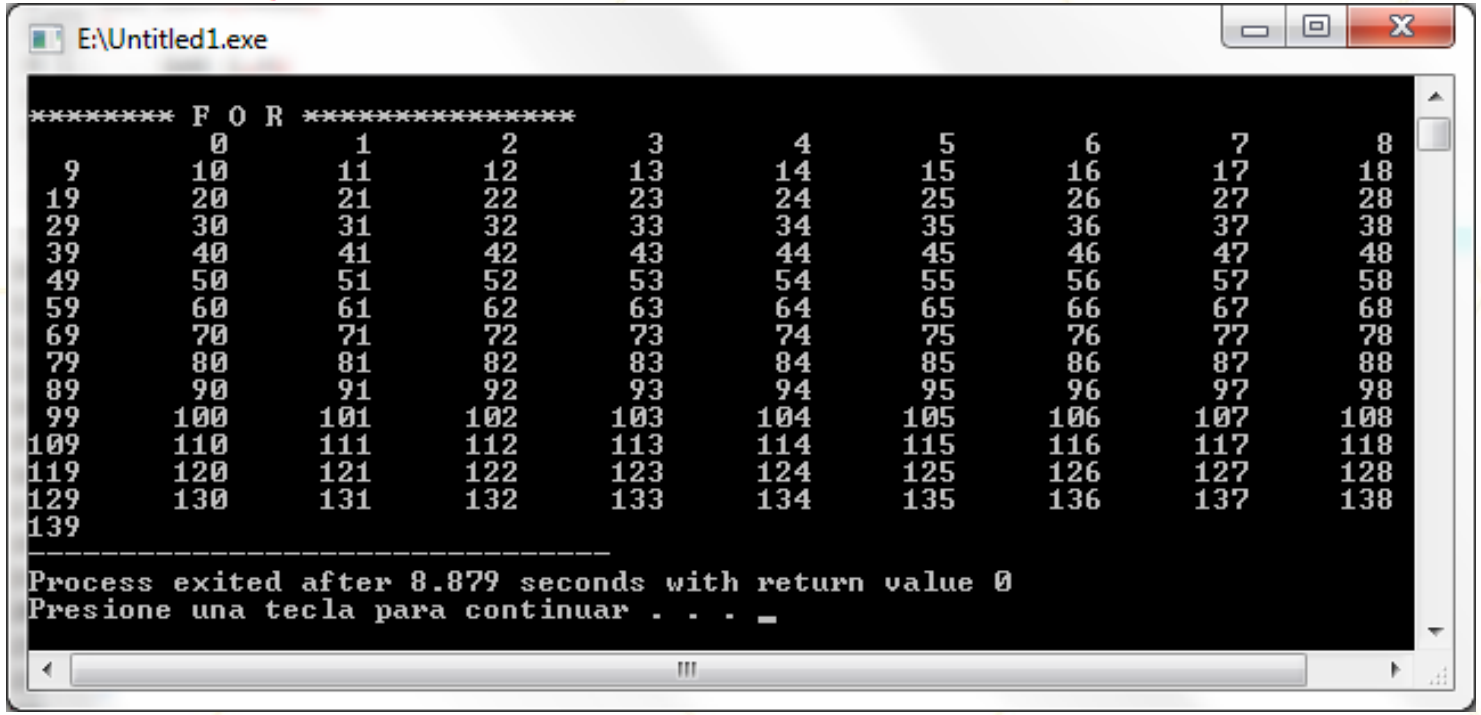
Ejemplo for(…)

```
#include<stdio.h>
int main(void)
{
    int i,n;
        printf("\n********* F O R ****************\n");
    for(i=0,n=10000;i<n;i++,n=n-i)
    {
        printf("\t%3d",i);
    }

}
```

Aplicaciones

Ejemplo for(…)

```
#include<stdio.h>
int main(void)
{
    int i,n;
        printf("\n******** F O R ***************\n");
    for(i=0,n=10000;i<n;i++,n=n-i)
    {
        printf("\t%3d",i);
    }

}
```



Aplicaciones

Ejercicio integral

Escribir un programa que mediante un menú, permita:

1. Imprimir intervalo.
2. Salir.

- Si el usuario selecciona opción 1, el programa tendrá que solicitar dos números enteros, correspondientes a un intervalo.
  - Si el valor de inicio es mayor al valor final, imprimir todos los números pares que estén desde el valor de inicio al valor final.
  - En caso contrario, si el valor de inicio es menor al valor final, desplegar los números consecutivos desde el valor de inicio al valor final.
  - Si los números son iguales imprimir un mensaje indicándolo.
- Si el usuario selecciona la opción 2, el programa termina la ejecución.
- Si el usuario selecciona un número diferente de 1 o 2 el programa desplegará un mensaje de error.

Bibliografía

- Cairo Osvaldo y Guardati Silvia. Metodología de la Programación. Algoritmos, diagramas de flujo y programas. Alfa Omega, 2005. México.
- Ceballos Sierra Francisco Javier. Enciclopedia del lenguaje C. Alfa Omega, 2007. México.
- Gottfried, Byron. Programación en C. McGraw Hill. 2005
- Joyanes Aguilar, Luis. Programación en C++. Algoritmos, estructuras de datos y objetos (3ª edición). McGraw-Hill, 20063. España.
- Joyanes Aguilar, Luis. Fundamentos de programación. Libro de problemas (2ª edición). McGraw-Hill, 2003. España.

❑ Índice