POSTULACIÓN DE MATERIAL ACADÉMICO

FACULTAD DE INGENIERÍA

PROGRAMA INGENIERÍA SISTEMAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

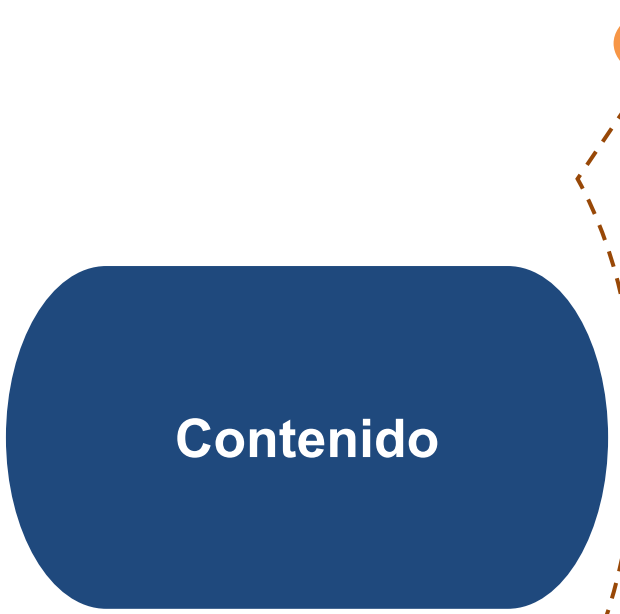
CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Asignatura fundamentos de programación

Estructuras Algorítmicas en C++ que Transformarán tu Código

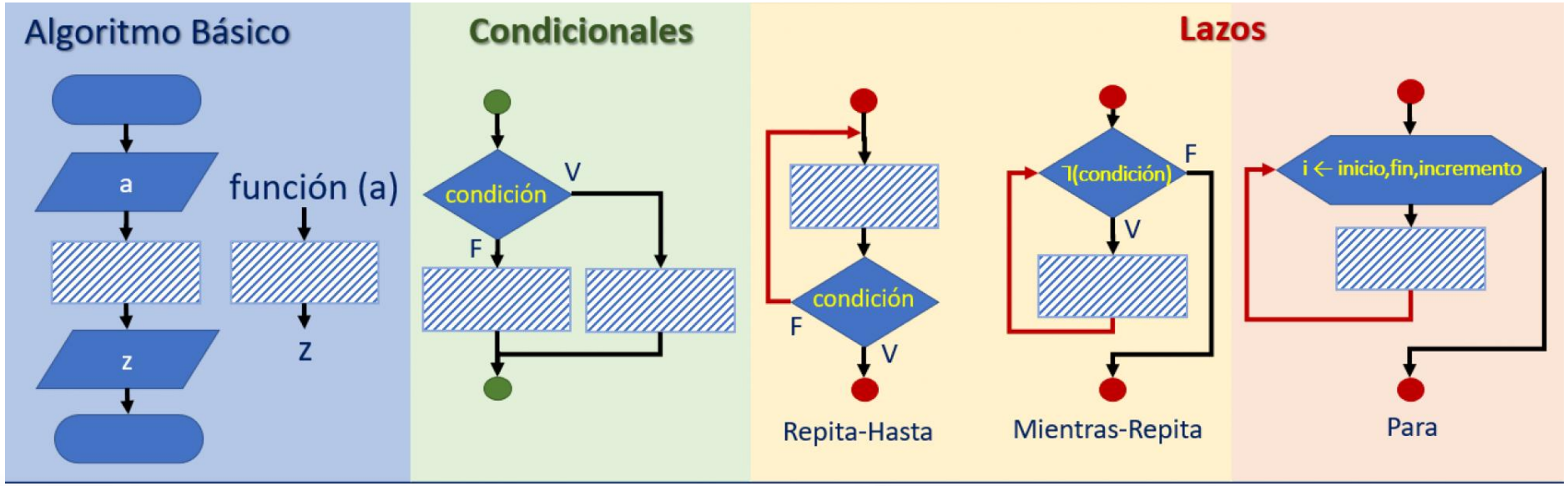
Docente: Edisney Garcia Perdomo

Contenido



1. Estructura Algorítmica
2. Guía para programar en C++ sobre cómo utilizar el depurador OnlineGDB
3. Estructuras Secuenciales en C++
4. Estructuras Condicionales en C++
5. Estructuras Decisión Doble en C++
6. Estructuras Condicional Anidado en C++
7. Estructuras Condicional Múltiple en C++
8. Estructuras de control Repetitivas en C++

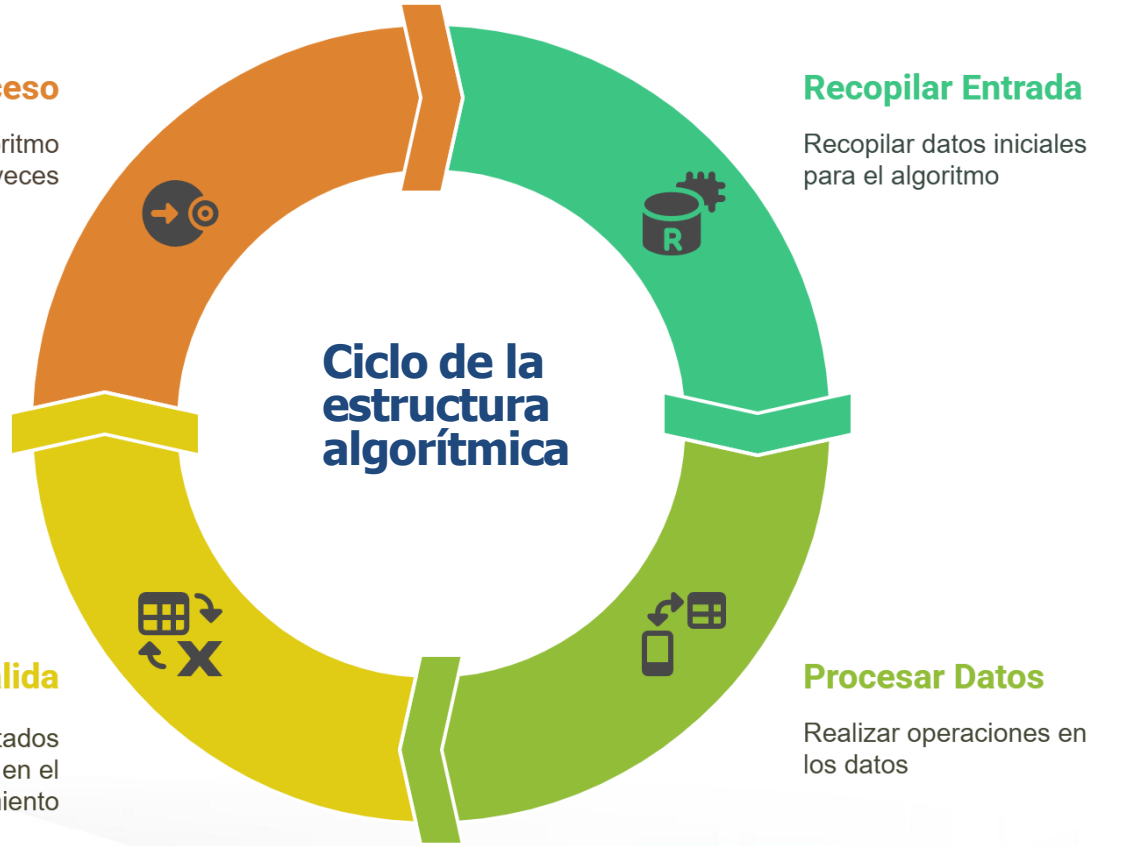
Estructura Algorítmica



Fuente: Elaboración propia

Estructura algorítmica

Las estructuras algorítmicas son los bloques fundamentales que nos permiten construir soluciones lógicas a problemas mediante algoritmos. Comprenderlas es esencial para programar de manera eficiente y ordenada.



Fuente: Elaboración propia

Estructura algorítmica

Estructuras Secuenciales

Instrucciones ejecutadas en orden

Acciones Algorítmicas

Pasos involucrados en la resolución de problemas

Elementos Algorítmicos

Componentes que definen un algoritmo

Estructuras Repetitivas

Instrucciones ejecutadas múltiples veces

Fuente: Elaboración propia

Las estructuras de operación de programas son un grupo de formas de trabajo, que permiten, mediante la manipulación de variables, realizar ciertos procesos específicos que nos lleven a la solución de problemas. Estas estructuras se clasifican de acuerdo con su complejidad.

Clasificación

SECUENCIALES

- Asignación
- Entrada
- Salida

CONDICIONALES

- Simples
- Dobles
- Múltiples

CICLICAS

- Para
- Mientras Que
- Repetir Hasta

Fuente: Elaboración propia

Guía para programar en C++ sobre cómo utilizar el depurador OnlineGDB

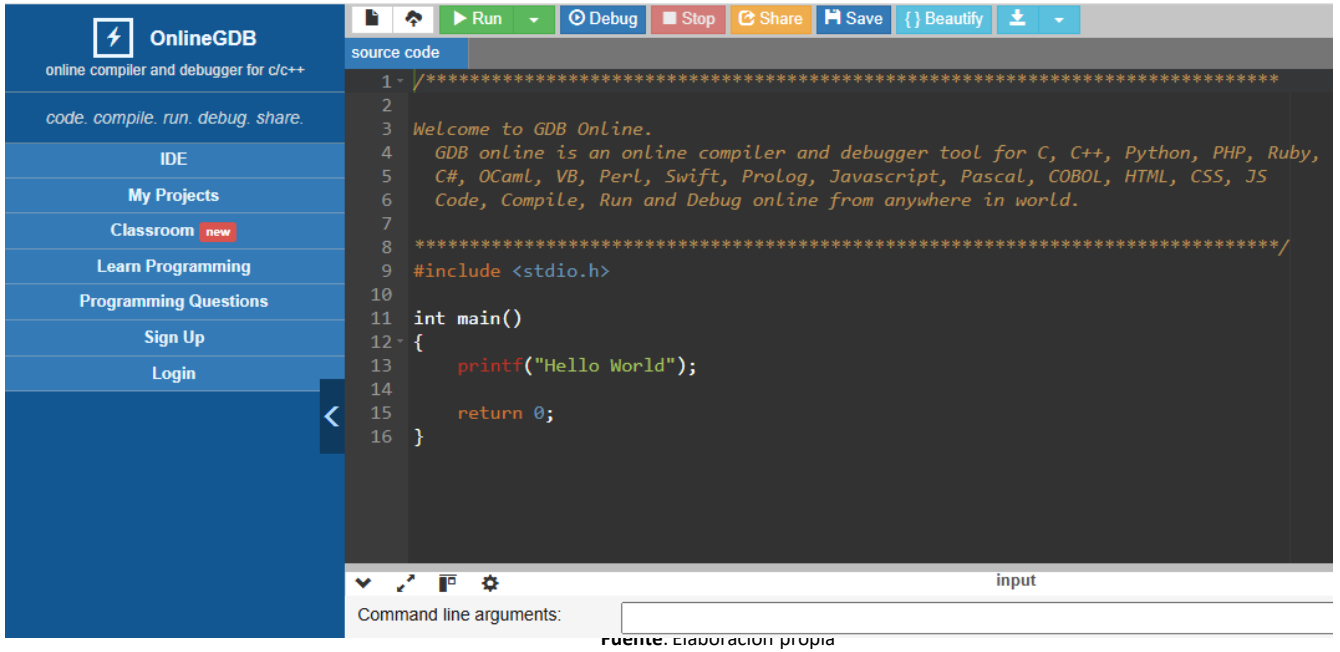
CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Guía sobre cómo utilizar el depurador OnlineGDB

Si eres nuevo en el mundo de la programación, es posible que no hayas oído hablar del depurador. A continuación, explicación brevemente su uso.

Si eres nuevo en el mundo de la programación, es posible que no hayas oído hablar del depurador. A continuación, explicación brevemente su uso.



Fuente: Elaboración propia

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Guía sobre cómo utilizar el depurador OnlineGDB

¿Qué es un depurador?

Imagina que eres un detective tratando de resolver un misterio en un programa de computadora. El depurador es tu lupa y tu kit de herramientas. En esencia, un depurador es una utilidad que te permite ejecutar tu programa en un entorno controlado, como si estuvieras observándolo bajo un microscopio. Puedes pausar la ejecución en cualquier momento, examinar el estado de las variables y entender qué está pasando en cada línea de código.

GDB: Tu aliado en C/C++

GDB es un depurador muy popular, especialmente útil para programas escritos en C y C++. Piensa en él como un asistente que te ayuda a encontrar errores y entender el flujo de tu programa. Con GDB, puedes ejecutar tu código paso a paso, inspeccionar variables y rastrear el origen de los problemas.

Fuente: Elaboración propia

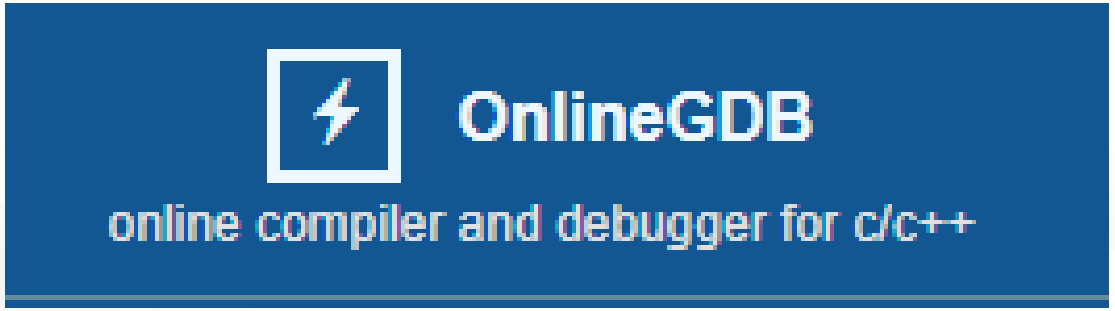
OnlineGDB: Depuración en la nube:

OnlineGDB lleva la depuración al siguiente nivel, permitiéndote usar GDB directamente desde tu navegador. Es como tener un laboratorio de programación en la nube, donde puedes depurar tus programas sin necesidad de instalar nada en tu computadora. Esto es especialmente útil para estudiantes y principiantes, ya que facilita el acceso a herramientas de depuración avanzadas.

Fuente: Elaboración propia

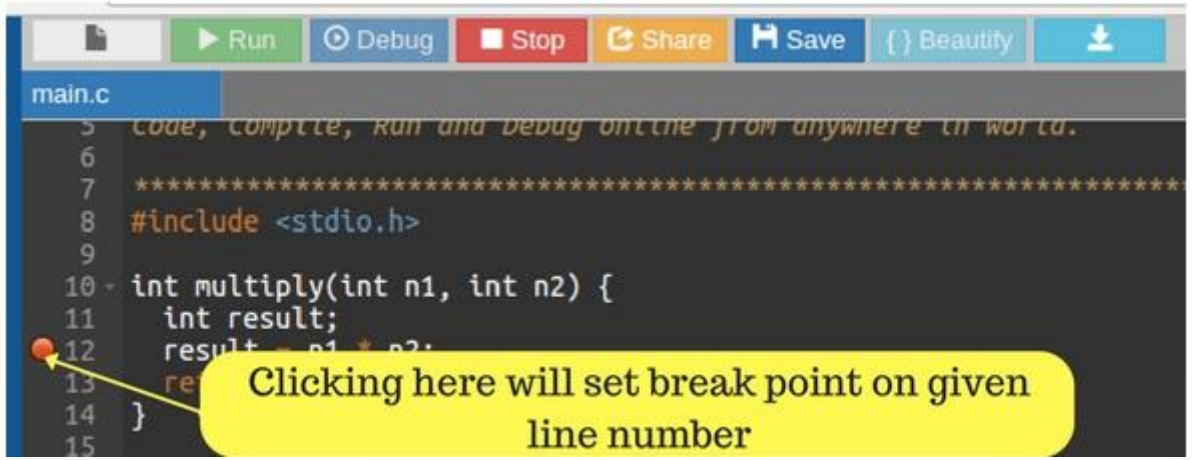
Controlando la ejecución del programa

Una de las funciones más importantes de un depurador es la capacidad de controlar la ejecución del programa. Esto se logra mediante el uso de puntos de interrupción. Un punto de interrupción es como una señal que le dices al depurador para que pause el programa en una línea específica.



Fuente: Elaboración propia

Para configurar un punto de interrupción en OnlineGDB, simplemente haz clic en el área vacía a la izquierda del número de línea en el editor de código. Al hacer clic, verás aparecer un círculo rojo, indicando que el punto de interrupción está activo en esa línea. ¡Es como poner una marca en el libro para recordar dónde quieres detenerte a leer con más detalle!

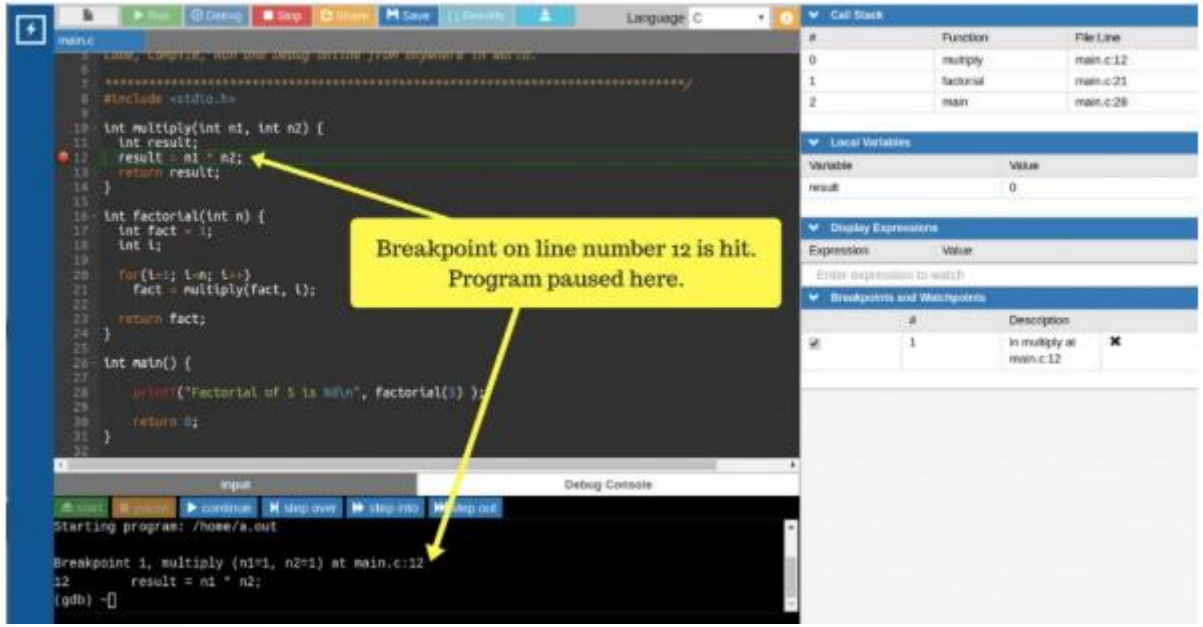


Fuente: Elaboración propia

Una vez que se establece el punto de interrupción, cuando se inicia el programa en modo de depuración, se pausará la ejecución cuando el programa llegue a la línea donde está establecido el punto de interrupción.

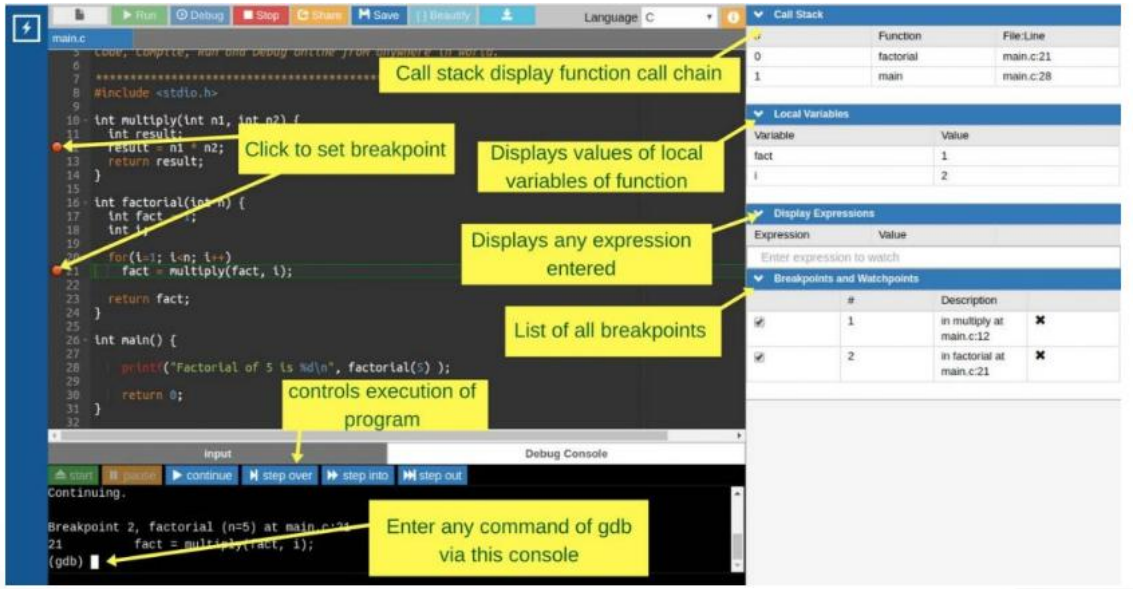
Fuente: Elaboración propia

Una vez que se establece el punto de interrupción, cuando se inicia el programa en modo de depuración, se pausará la ejecución cuando el programa llegue a la línea donde está establecido el punto de interrupción.



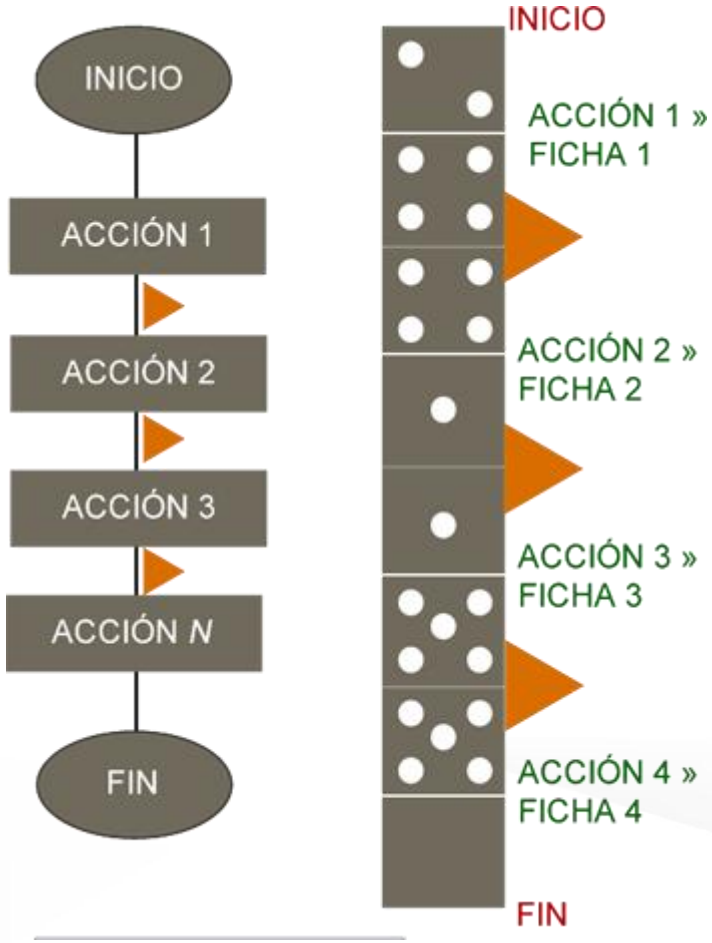
Fuente: Elaboración propia

Los puntos de interrupción, los comandos de ejecución por pasos y la visualización de variables son funciones básicas del depurador.



Fuente: Elaboración propia

Estructuras secuenciales en C++



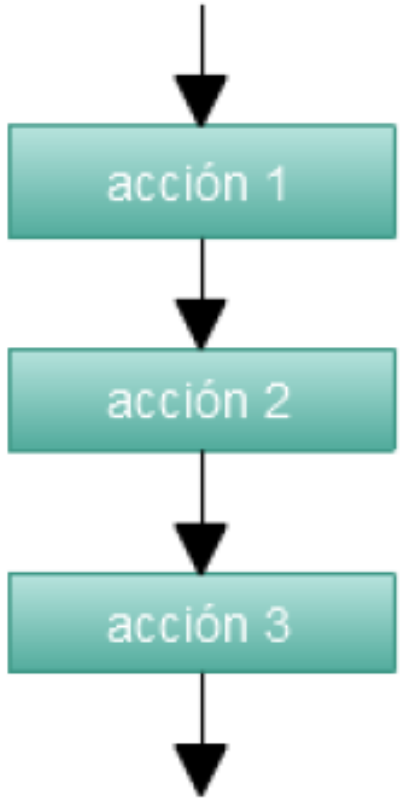
Fuente: Elaboración propia

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

Una estructura secuencial se representa de la siguiente forma:

Pseudocódigo

```
inicio
   accion 1
   accion 2
   accion 3
        .
        .
        .
   accion n
fin
```



Fuente: Elaboración propia

La estructura secuencial es la más simple: las instrucciones se ejecutan una tras otra, en el orden en que aparecen.

Ejemplo cotidiano:
Imagina que sigues una receta de cocina: primero mezclas los ingredientes, luego los horneas y finalmente sirves el plato. Cada paso ocurre uno después del otro.

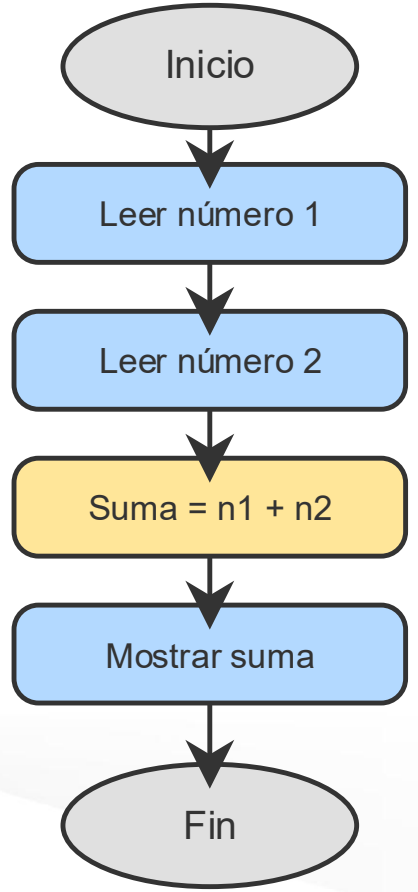
Ejemplo en C++ (con código documentado)A continuación, se presenta un ejemplo sencillo en C++ que ilustra una estructura secuencial. El programa solicita al usuario dos números, los suma y muestra el resultado.

Diagrama de flujo

El diagrama de flujo para este algoritmo secuencial es el siguiente:

1. Inicio
2. Leer número 1
3. Leer número 2
4. Sumar número 1 y número 2
5. Mostrar resultado
6. Fin

Representación grafica



Fuente: Elaboración propia

Prueba de escritorio

La prueba de escritorio consiste en simular la ejecución del programa paso a paso con valores de ejemplo. Supongamos que el usuario ingresa 5 y 7:

Paso	número1	número2	suma
Ingreso de datos	5	7	
Suma	5	7	12
Mostrar resultado	5	7	12

Salida esperada:

La suma es: 12

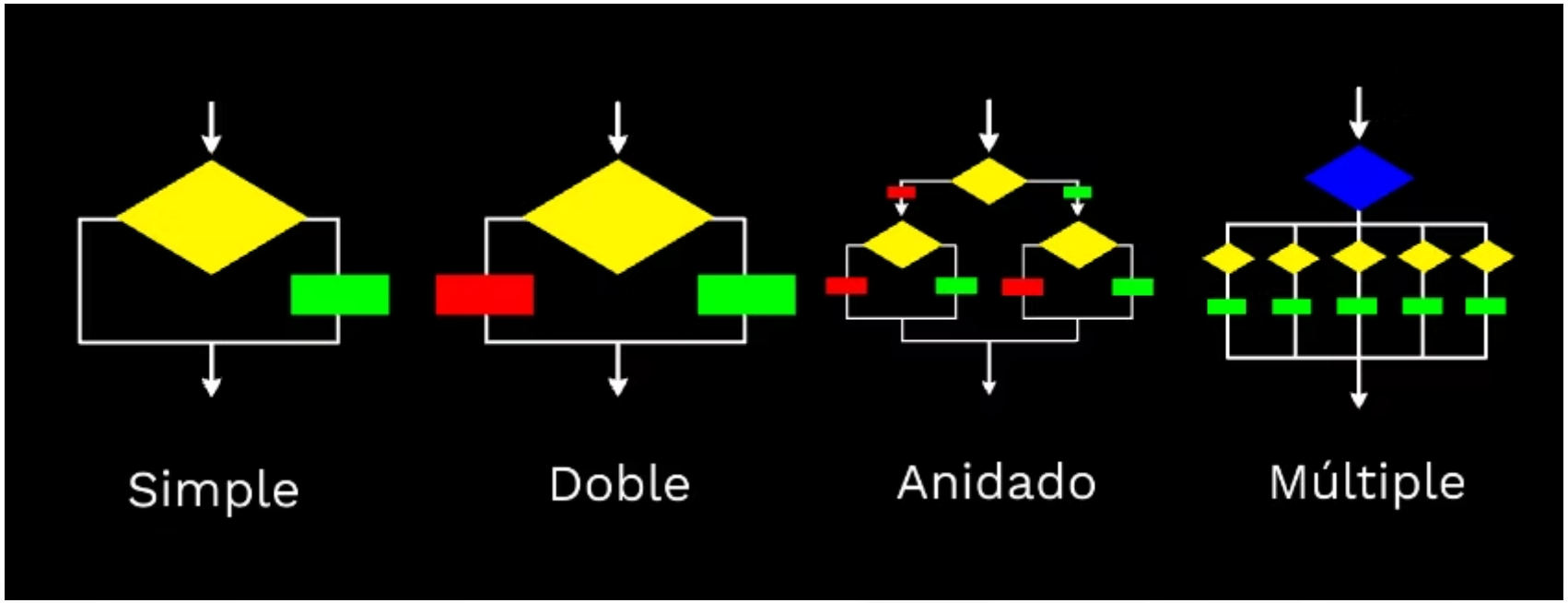
Este enfoque permite comprender cómo se ejecutan las instrucciones de manera secuencial, visualizar el flujo del algoritmo y verificar su funcionamiento antes de implementarlo en el computador. Además, la documentación en el código y la prueba de escritorio ayudan a reforzar la lógica y a detectar posibles errores de manera anticipada.

Código documentado

```cpp
#include <iostream>
using namespace std;

int main() {
    // Declaración de variables
    int numero1, numero2, suma;

    // Solicitar al usuario el primer número
    cout << "Ingrese el primer número: ";
    cin >> numero1;

    // Solicitar al usuario el segundo número
    cout << "Ingrese el segundo número: ";
    cin >> numero2;

    // Realizar la suma de los dos números
    suma = numero1 + numero2;

    // Mostrar el resultado en pantalla
    cout << "La suma es: " << suma << endl;

    return 0;
}
```

Fuente: Elaboración propia

Estructuras Condicionales en C++



Fuente: Elaboración propia

Un condicional en programación es la manera con el cual logra a partir de una disposición un camino por el cual seguir.

En nuestra vida diaria utilizamos mucho este tipo de estructura a la hora de tomar decisiones como:

Fuente: Elaboracion propia

Estructuras de Condicionales en Programación

¿Que color de zapatos me llevo hoy?

¿Estudio mi tarea o la dejo para mas tarde?

¿Para ir a estudiar llevo cuadernos o no?

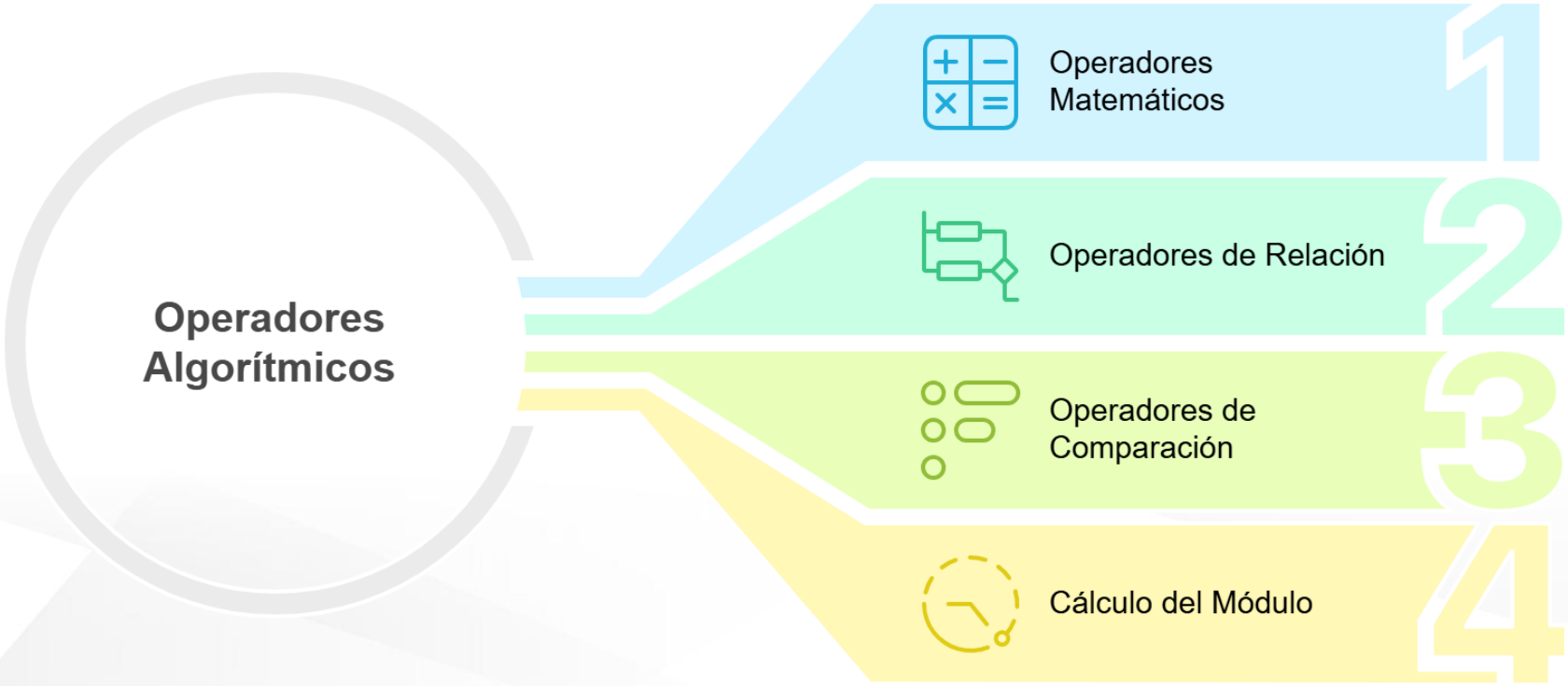
Según la decisión que tomes a este tipo de preguntas tu ves que camino elegir. Algo muy común se presenta a la hora de programar.

Una condicional deben disponerse únicamente variables, valores constantes y operadores relacionados.

Cabe destacar que existen varios tipos de Estructura para los condicionales. Condicionales simples, dobles y múltiples

Explorando los operadores algorítmicos esenciales

Para trabajar eficazmente con los diferentes tipos de estructuras algorítmicas, es fundamental comprender y aplicar correctamente los operadores matemáticos, de relación y comparación.



Fuente: Elaboración propia

Operadores Relacionales de un condicional

> (mayor)

< (menor)

>= (mayor igual)

<= (menor iguai)

== (igual)

!= (distintos)

Operadores Matemáticas

+ (mas)

- (menos)

* (producto)

/ (división)

% (resto de una división)

Tipos de estructura

Condicionales simples, dobles y múltiples

Simple: SI ... ENTONCES ...
Doble: SI ... ENTONCES... SI NO ...
Múltiple: EN CASO DE ... ENTONCES ...

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

1. Operadores Lógicos

Operador	Significado
||	o
&&	y

2. El operador MOD se utiliza para saber si un número es par o impar

Si el residuo es cero es número es par

Si el residuo es uno es número es impar

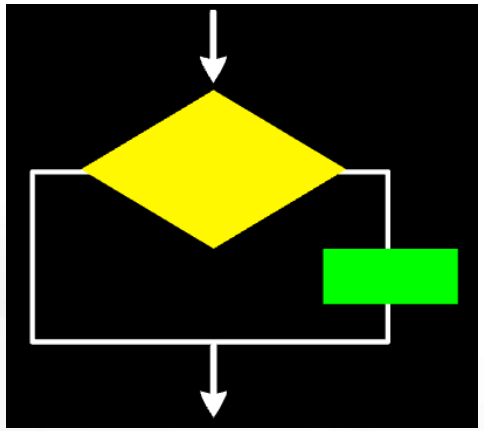
3. Ejemplo:

(numero % 2 == 0)

Estructuras de decisión simple

El condicional simple evalúa una condición y ejecuta un bloque de código si esta condición es verdadera. Es como decir: "si esto es cierto, haz lo siguiente". Este tipo de condicional es útil cuando solo necesitas tomar una acción en función de una única condición.

Estructura



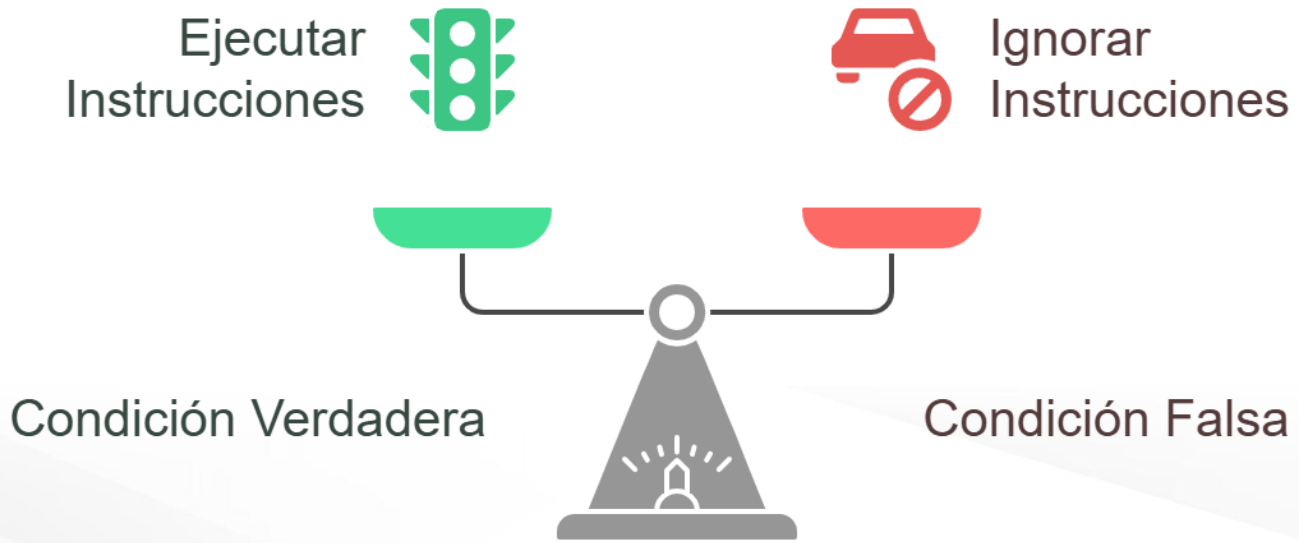
Fuente: Elaboración propia

Ejemplo

```
Escribir "Ingresa tu edad: "
Leer edad
Si edad >= 18 Entonces
Escribir "Eres mayor de
edad"
Fin Si
```

Estructuras de decisión Simple

En este tipo de estructura se evalúa **una condición o expresión lógica**, si es verdadera se ejecutan un conjunto de instrucciones, si la condición **es falsa se ignoran estas instrucciones.**



Fuente: Elaboración propia

Estructuras de decisión simple

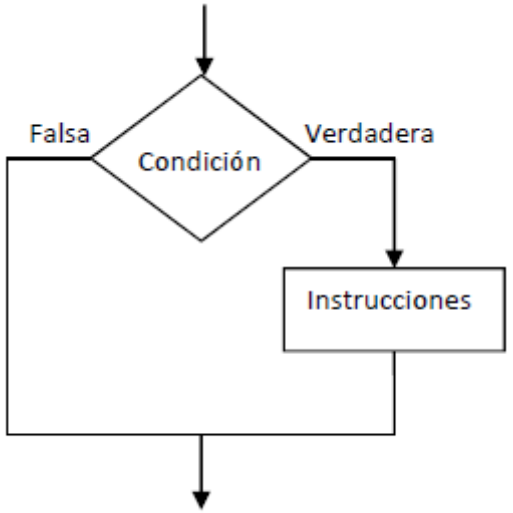
Pseudocodigo

Si condición **entonces**

Instrucciones a ejecutar si la condición es verdadera

Fin de si

Diagrama de flujo



Código en C++

```
//condición
if(edad>=18){
    cout <<"es mayor de edad";
}
```

Código – palabra reservada - sintaxis

If (condición) { entonces

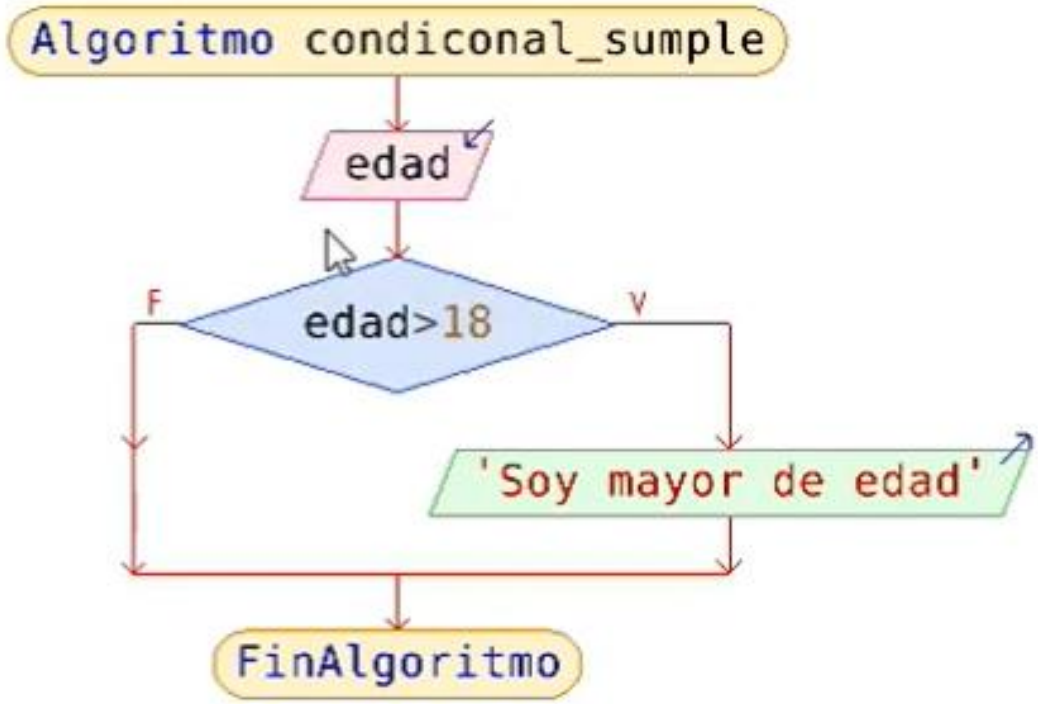
acción (imprimir, realizar una operación

}

Ejemplo 1:

// Ingresar la edad de una persona, y si la edad es mayor o igual a 18 años, imprimir un mensaje que diga es mayor de edad.

Diagrama de flujo

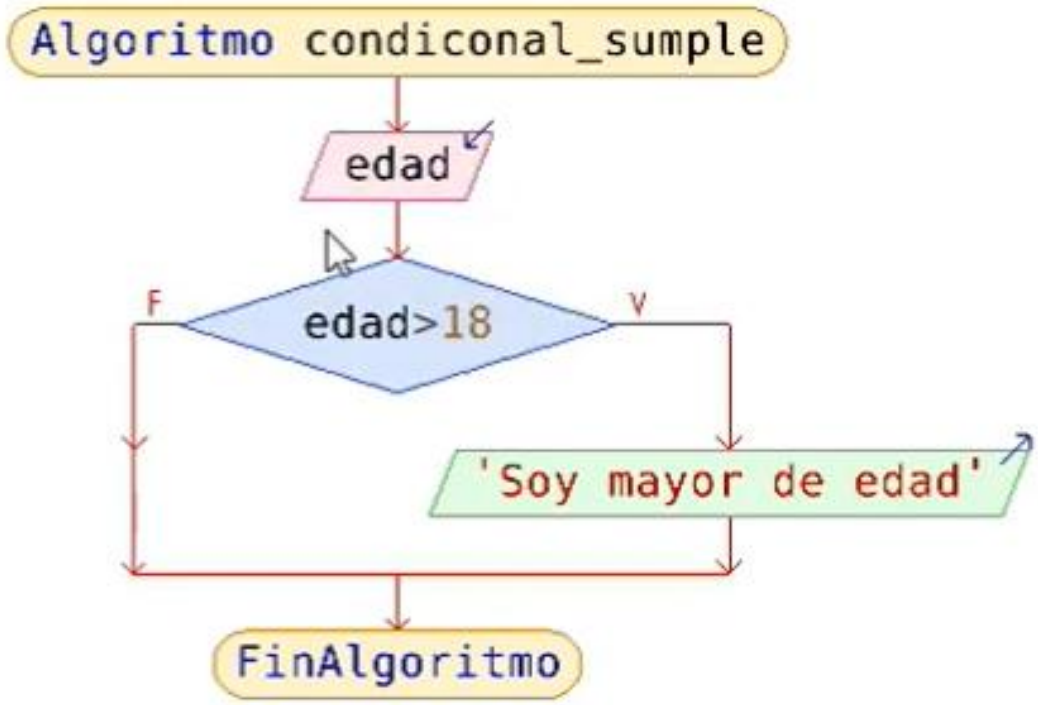


Fuente: Elaboración propia

Código en GDB

```
#include <iostream>
using namespace std;
int main()
{
//defino tipo de dato y variable
  int edad;
  //ingreso los datos
  cout<<"digite la edad"<<endl;
  //escribe la edad
   cin>>edad;
 //condición
 if(edad>=18){
     cout <<"es mayor de edad";

  }
     return 0;
}
```

Diagrama de flujo

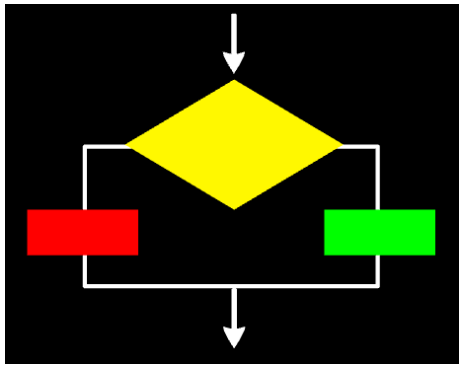


Fuente: Elaboración propia

Estructuras Decisión Doble

El condicional doble evalúa una condición y ejecuta un bloque de código si esta es verdadera, y otro bloque si la condición es falsa. Es como decir: "si esto es cierto, haz esto; si no, haz aquello". Este tipo de condicional es útil cuando necesitas tomar una de dos acciones posibles.

Estructura



Fuente: Elaboración propia

Ejemplo

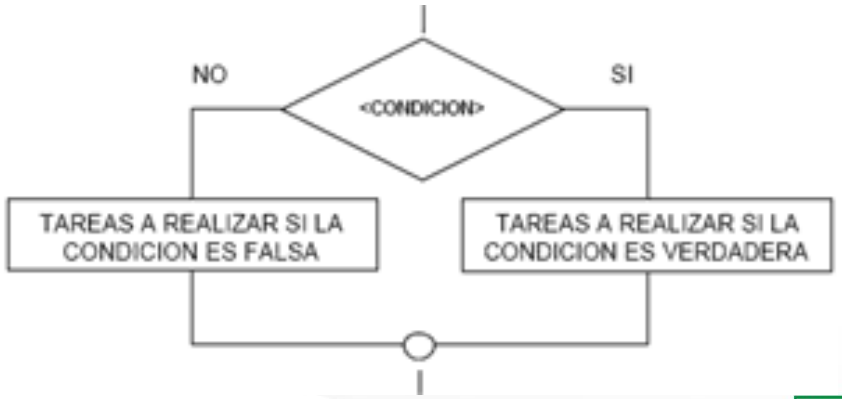
```
Escribir "Ingresa tu edad: "
Leer edad
Si edad >= 18 Entonces
Escribir "Eres mayor de edad"
Sino
Escribir "Eres menor de edad"
FinSi
```

Los condicionales dobles cumple casi las mismas características que los simples, solo que esta estructura si tiene actividades a realizar por los dos caminos.

Pseudocódigo

```
Algoritmo Condiconal_Doble
leer edad
Si edad>=18 entonces
 Escribir "Mayor de edad"
Sino
 Escribir "Menor de edad"

FinSi

FinAlgoritmo
```

Diagrama de flujo



Fuente: Elaboración propia

Ejemplo 1:

// Realice un programa que pida al usuario ingresar la edad y diga si es mayor de edad o menor de edad.

Explicación Detallada del Código:

1. Librerías y Configuración Inicial

```
#include <iostream>
using namespace std;
```

- #include <iostream>: Incluye la librería necesaria para operaciones de entrada (cin) y salida (cout)
- using namespace std;: Permite usar cout y cin directamente sin escribir std::cout o std::cin

2. Declaración de Variables

```
int edad;
```

Declaramos una variable de tipo entero llamada edad para almacenar la edad del usuario

3. **Entrada de Datos**

```
cout << "Por favor, ingrese su edad: ";
cin >> edad;
```

- cout: Muestra un mensaje en pantalla solicitando la edad
- cin: Lee el valor ingresado por el usuario y lo guarda en la variable edad

4. **Validación de Datos**

```
if (edad < 0) {
    cout << "Error: La edad no puede ser un número negativo." << endl;
    return 1;
}
```

Verificamos que la edad sea un valor válido (no negativo)

Si es inválida, mostramos un error y terminamos el programa

5. **Lógica Principal (Estructura Condicional)**

```
if (edad >= 18) {
    // Mayor de edad
} else {
    // Menor de edad
}
```

Usamos una estructura if-else para evaluar si la edad es mayor o igual a 18. Según el resultado, mostramos el mensaje correspondiente

Prueba de Escritorio:

Entrada (edad)	edad < 0	edad >= 18	Salida
25	false	true	"Es MAYOR DE EDAD"
16	false	false	"Es MENOR DE EDAD"
-5	true	-	"Error: edad negativa"
18	false	true	"Es MAYOR DE EDAD"

Fuente: Elaboración propia

Ejemplo de Ejecución:

=== VERIFICADOR DE MAYORÍA DE EDAD ===
Por favor, ingrese su edad:
20 Usted tiene 20 años.
✓ Es MAYOR DE EDAD

Código completo

```
// Incluimos la librería iostream para operaciones de entrada y salida
 #include  <iostream>
// Usamos el espacio de nombres estándar para evitar escribir std:: antes de cout y cin
using namespace std;
int main() {
        // Declaramos una variable entera para almacenar la edad del usuario
        int edad;

        // Mostramos un mensaje solicitando al usuario que ingrese su edad
         cout << "=== VERIFICADOR DE MAYORÍA DE EDAD ===" << endl;
        cout << "Por favor, ingrese su edad: ";

        // Leemos la edad ingresada por el usuario y la almacenamos en la variable 'edad'
        cin >> edad;
        // Validamos que la edad ingresada sea un valor lógico (mayor a 0)
         if (edad < 0) {
                cout << "Error: La edad no puede ser un número negativo." << endl;
                return 1; // Terminamos el programa con código de error
        }
```

Código completo

```
// Estructura condicional para determinar si es mayor o menor de edad // En la mayoría de los países, la mayoría de edad es a los 18 años
if (edad >= 18) {
// Si la edad es 18 o mayor, es mayor de edad
        cout << "Usted tiene " << edad << " años." << endl;
        cout << "✓         Es MAYOR DE EDAD" << endl;
        cout << "Puede votar, conducir y tomar decisiones legales." << endl;
} else {

        // Si la edad es menor a 18, es menor de edad
        cout << "Usted tiene " << edad << " años." << endl;
        cout << "✗ Es MENOR DE EDAD" << endl;
        cout << "Aún no puede votar ni tomar ciertas decisiones legales." << endl;
}
// Mensaje de despedida
        cout << "¡Gracias por usar el programa!" << endl;
// Retornamos 0 para indicar que el programa terminó correctamente
return 0;
}
```

Ejemplo 2:

// Escriba un programa que solicite al usuario ingresar un número entero y determine si es par o impar. Muestre el resultado por pantalla.

Análisis:

- El usuario ingresa un número.
- El programa verifica si el número es divisible por 2 usando el operador módulo (%).
- Si el residuo es 0, el número es par; si no, es impar.
- Se muestra el resultado en pantalla.

Ejemplo 2:

// Escriba un programa que solicite al usuario ingresar un número entero y determine si es par o impar. Muestre el resultado por pantalla.

El Código GDB debe de estar documentado según lo explicado en la sesión anterior:

1. Inclusión de librerías y espacio de nombres
2. Función principal
3. Declaración de variables
4. Solicitud y lectura de datos
5. Estructura condicional para determinar paridad
6. Fin del programa

Ejemplo 2:

// Ingrese a GDB y ejecute el siguiente código .

```
#include <iostream>
using namespace std;
 int main() {
 int numero;
cout << "Ingrese un número: ";
cin >> numero;
if (numero % 2 == 0) {
    cout << "El número es par" << endl;
} else
      { cout << "El número es impar" << endl;
}
    return 0;
}
```

Ejemplo 3:

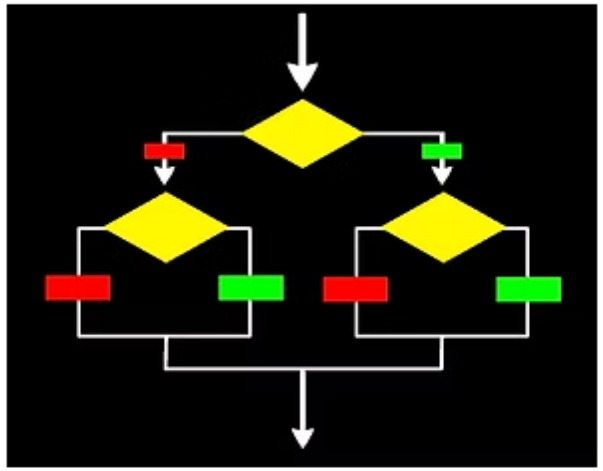
//Mayor de tres números, cree un programa que solicite al usuario ingresar tres números enteros y determine cuál es el mayor de ellos. Muestre el resultado.

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b, c;
    cout << "Ingrese tres números: ";
    cin >> a >> b >> c;

    if (a >= b && a >= c) {
        cout << "El mayor es: " << a << endl;
    } else if (b >= a && b >= c) {
        cout << "El mayor es: " << b << endl;
    } else {
        cout << "El mayor es: " << c << endl;
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main() {
 int A,B,C;
  cout<<"ingrese tres numeros" <<endl;
 cin>> A >> B >> C;
    if (A >= B && A >= C ){ // Pregunta si A es mayor o igual a B y A es mayor o igual que C entonces
            cout<<" El numero mayor es: " << A << endl;
            } else if (B >= A && B >= C) {
          cout << "El mayor es: " << B << endl;
         }else {
          cout << "El mayor es: " << C << endl;
        }
      return 0;
}
```

Estructuras Condicional Anidado

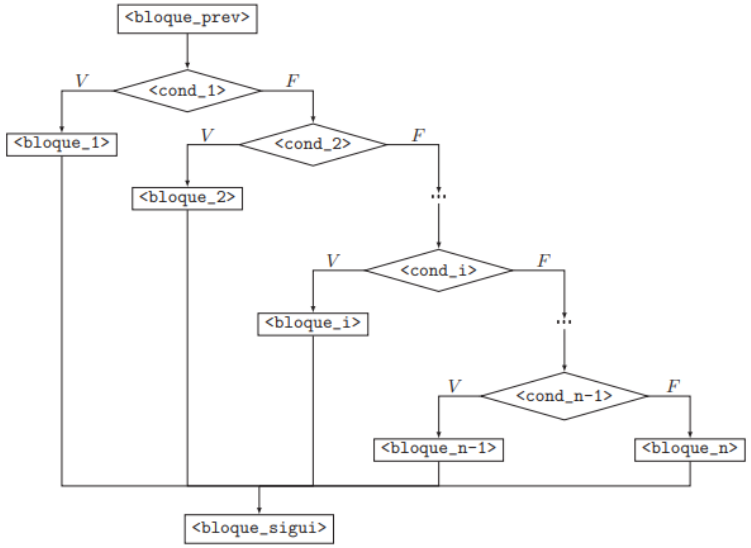
Estructura	Ejemplo

	Escribir "Ingresa un número: "
	Leer num
	Si num > 0 Entonces
	  Escribir "El número es positivo"
	Sino Si num < 0 Entonces
	  Escribir "El número es negativo"
	Sino
	  Escribir "El número es cero"
	FinSi

Fuente: Elaboración propia

Estructuras Condicional Anidado

El condicional anidado evalúa múltiples condiciones en secuencia y ejecuta el bloque de código asociado a la primera condición verdadera. Es como tener varias opciones y elegir la primera que se cumpla. Este tipo de condicional es útil cuando necesitas tomar decisiones basadas en múltiples condiciones



Fuente: Elaboración propia

Estructuras Condicional Anidado

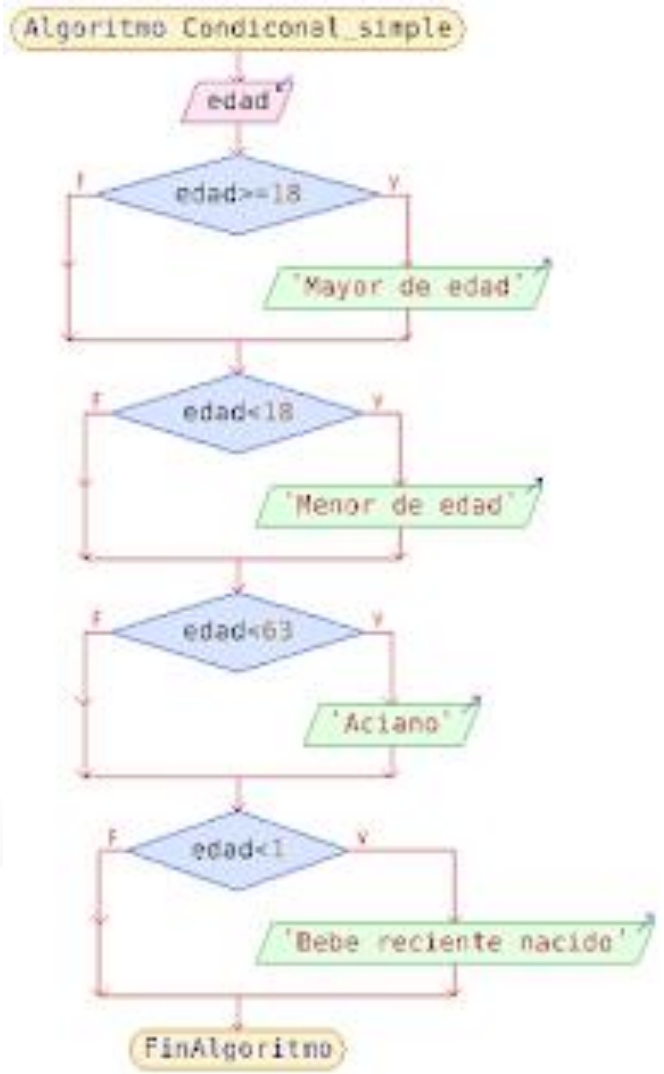
Si en una condición se requiere hacer más de una pregunta se puede utilizar un IF anidado. La siguiente imagen representa el diagrama de flujo de un IF anidado.

La estructura en el lenguaje de programación

```
if (condición ) {
  //instrucciones para el caso verdadero - IF
} else {
  if (condición ) {
  //instrucciones para el caso verdadero - IF
  } else {
  //instrucciones para el caso falso - ELSE
  }
}
```

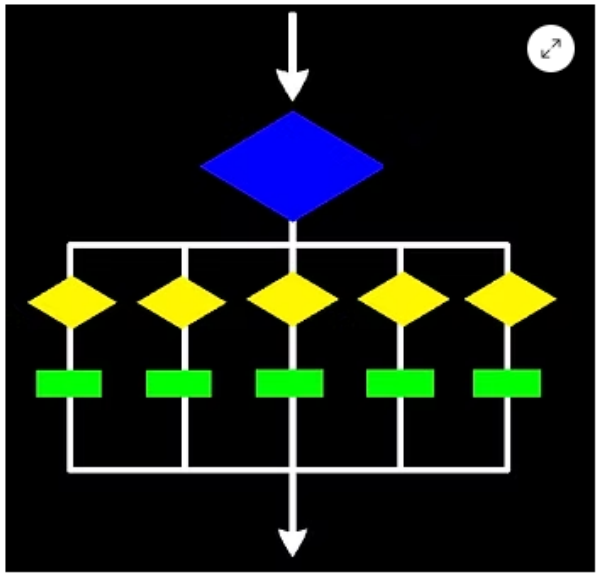
Estructuras Condicional Anidado

```
Algoritmo Condiconal_multiple

leer edad
Si edad>=18 entonces
        Escribir "Mayor de edad"
FinSi
        si edad<18 Entonces
                Escribir "Menor de edad"
        FinSi
        si edad < 63 Entonces
        Escribir "Aciano"
FinSi
si edad < 1 Entonces
Escribir "Bebe reciente nacido"
FinSi


FinAlgoritmo
```



Fuente: Elaboración propia

Estructuras Condicional Múltiple

Estructura	Ejemplo
 **Fuente**: Elaboración propia	Escribir "Ingresa un día de la semana (1-7): " Leer dia Segun dia Hacer 1: Escribir "Lunes" 2: Escribir "Martes" 3: Escribir "Miércoles" 4: Escribir "Jueves" 5: Escribir "Viernes" 6: Escribir "Sábado" 7: Escribir "Domingo" De Otro Modo: Escribir "Número de día inválido" FinSegun

Estructuras Condicional Múltiple

El condicional múltiple evalúa una expresión y ejecuta el bloque de código asociado a uno de los posibles valores de esa expresión. Es útil cuando se tienen múltiples opciones y se desea ejecutar un bloque de código según el valor de una variable.

```
int opcion=2;
    switch (opcion){
      case 1:{
        System.out.println("Usted eligió la opcion 1.");
        break;
      }
      case 2:{
        System.out.println("Usted eligió la opcion 2.");
        break;
      }
      case 3:{
        System.out.println("Usted eligió la opcion 3.");
        break;
      }
      default: {
        System.out.println("Opcion incorrecta");
      }
  }//cierra SWITCH
```

La estructura en el lenguaje de programación

Estructuras Condicional Múltiple

El condicional switch case es una estructura que evalúa más de un caso y se caracteriza por:

- Selección de una opción entre varias.
- Switch recibe un “caso” y lo evalúa hasta encontrar el caso que corresponda.
- Se puede usar la opción “default” para cuando no se encuentra el caso dado.

Este condicional es útil a la hora de definir por ejemplo un menú de usuario en aplicaciones que se ejecutan por consola.

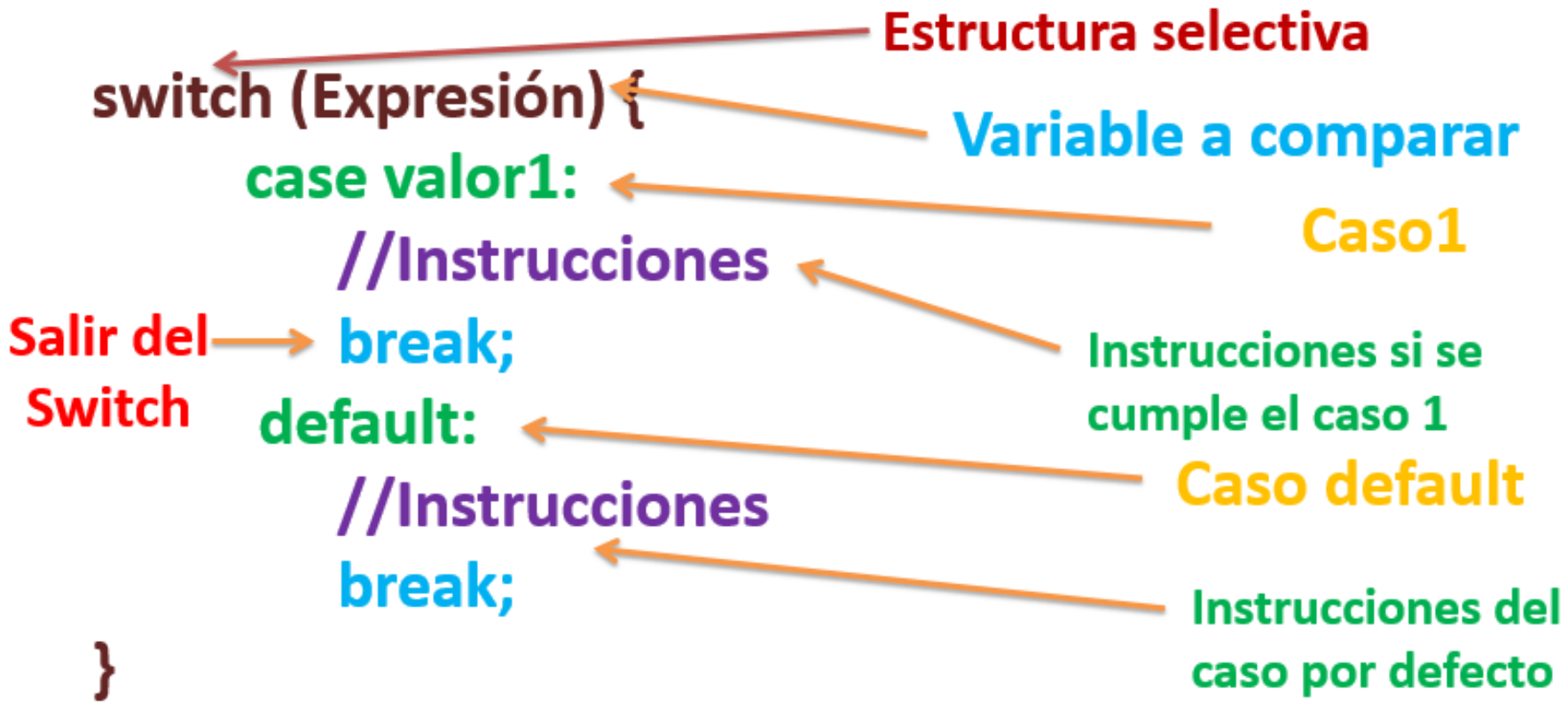
En las condicionales múltiples también existe una estructura en caso de o switch, un ejemplo a continuación:

Usando la estructura switch, que es más adecuada para este tipo de decisiones múltiples basadas en un valor específico:

Diseñe un programa segundo el numero ingresado del 1 a 5, me imprima que día es según:

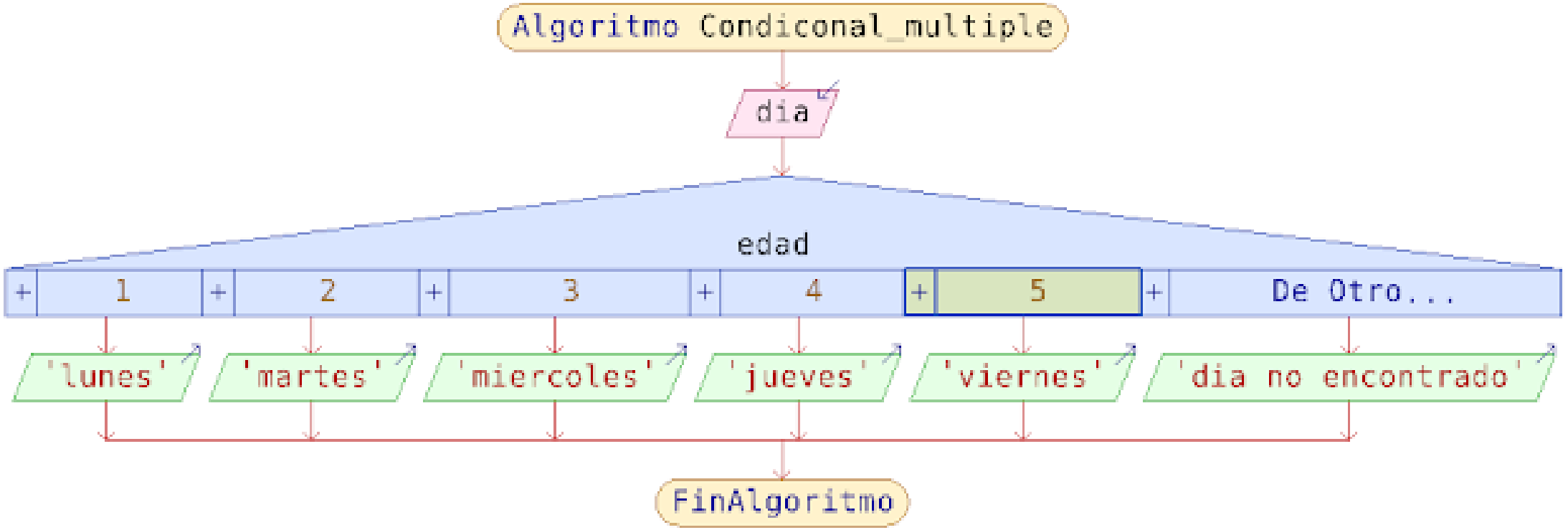
Si se ingresa 1 = lunes, 2 = martes, 3 = miércoles, 4= jueves, 5 = a viernes

Estructuras de decisión Múltiples



Estructuras de decisión Múltiples

Diagrama de flujo



Fuente: Elaboración propia

Código realizarlo en GDB

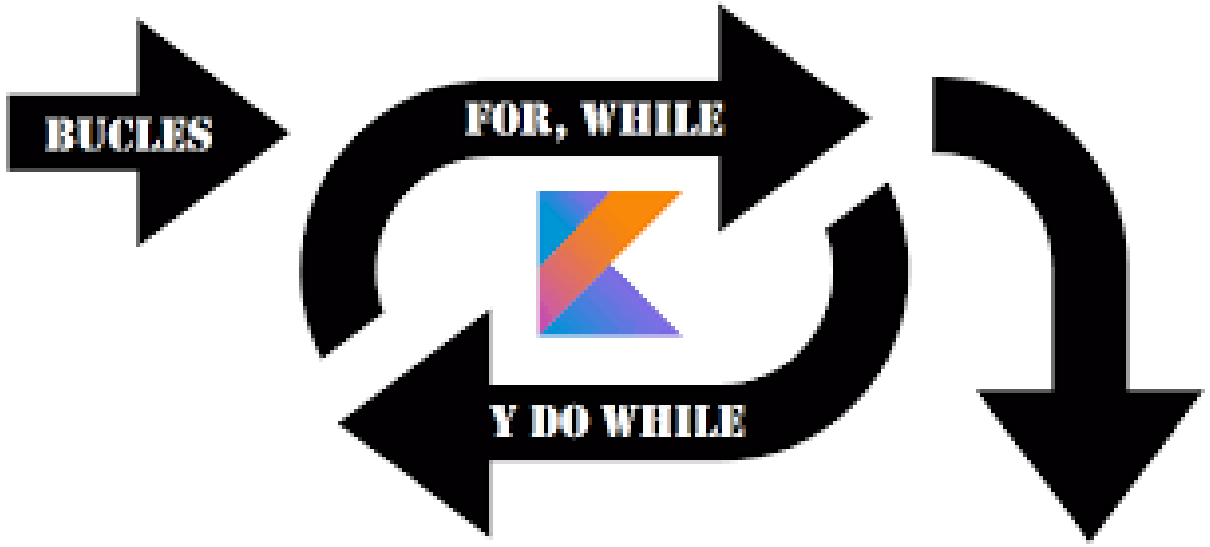
```
using namespace std;
int main() {
    int numero;
    cout << "Ingrese un número del 1 al 5: ";
    cin >> numero;
// Estructura switch para decisión múltiple
switch (numero) {
        case 1:
            cout << "Lunes" << endl;
            break;
        case 2:
            cout << "Martes" << endl;
             break;
        case 3:
            cout << "Miércoles" << endl; break; case 4: cout << "Jueves" << endl;
            break;
       case 5:
           cout << "Viernes" << endl;
           break;
       default:
           cout << "Número inválido. Por favor ingrese un número del 1 al 5." << endl;
  }
       return 0;
}
```

```
switch (variable)
{
case valor1:
        //instrucciones
        break;
case valor2:
        //instrucciones
        break;
case valor3:
        //instrucciones
        break;
default:
        //instrucciones
        break;
}
```

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Estructuras de Control Repetitivas en C++

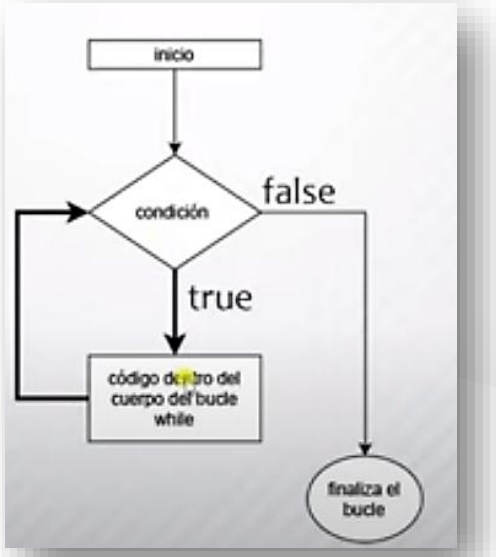


Fuente: Elaboración propia

Estructuras Iterativas

Las estructuras de control iterativas se utilizan para resolver problemas donde sea necesario repetir un determinado número de veces un conjunto de instrucciones.

También se les conoce como estructuras repetitivas. De manera general existen 3 tipos:



Fuente: Elaboración propia

Estructuras Iterativas

1. Bucle while (Repetición Mientras)
2. Bucle do-while (Repetición Hacer Mientras)
3. Bucle for (Repetición Para)

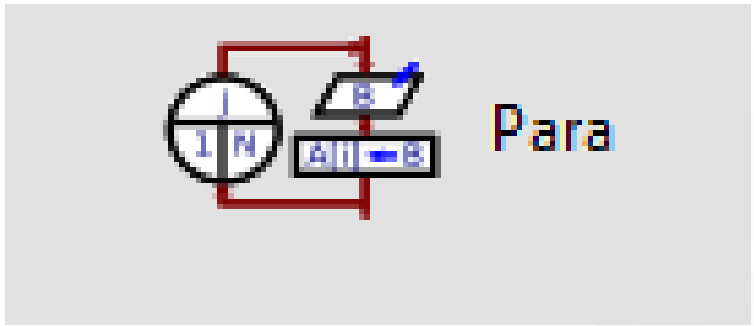
Las estructuras cíclicas se utilizan para ejecutar fragmentos de código un número limitado de veces.

También se les conoce como estructuras **repetitivas**. De manera general existen 3 tipos:

1. Hacer Mientras (While)

2. Hacer Hasta (Do While)

3. Desde Hasta (For)



Fuente: Elaboración propia

¿Para Que Se Utilizan?

También se les conoce como estructuras repetitivas. De manera general existen 3 tipos:

1. Hacer Mientras (While)

En este ciclo primero hago la condición y después hago la acción. Se ejecuta mientras una condición sea verdadera. La condición se evalúa antes de cada iteración.

Fuente: Elaboración propia

¿Para Que Se Utilizan?

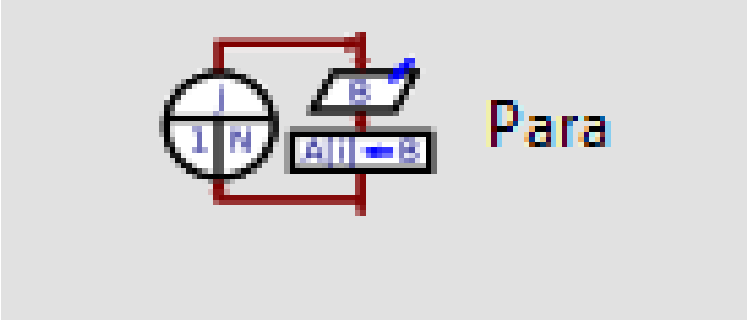
2 Hacer Hasta (Do While)

Haga esto mientras cumpla esta condición.
Primero se hace la acción y luego la condición. Similar al while, pero garantiza al menos una ejecución porque la condición se evalúa al final.

Fuente: Elaboración propia

3 Desde Hasta (For)

Desde Hasta **(For)** En el Para primero hago la condición y luego la acción. Utilizado cuando conocemos de antemano el número de iteraciones

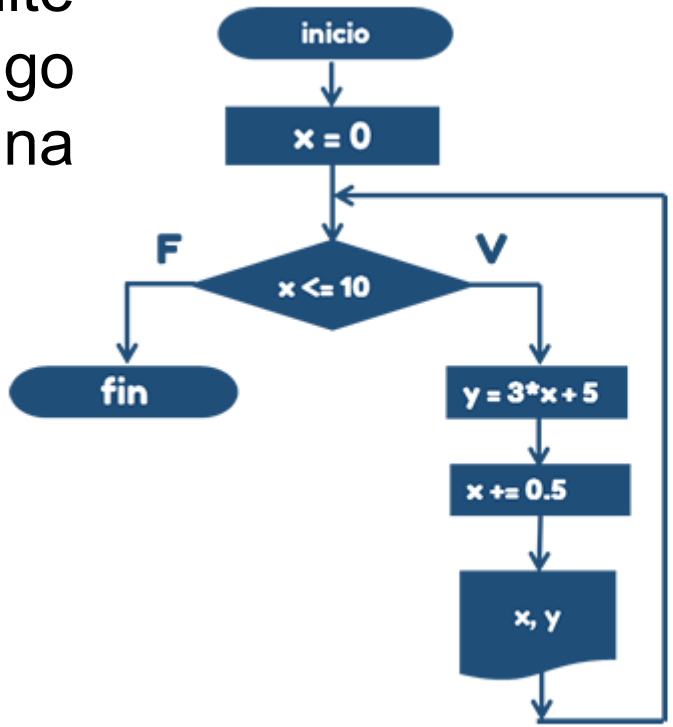


Fuente: Elaboración propia

While (Repetición Mientras) en C++

El bucle while es una estructura de control de iteración que permite ejecutar un bloque de código repetidamente mientras una condición dada sea verdadera.

Mientras que se cumpla la expresión lógica se van a seguir ejecutando las instrucciones, cuando se deje de cumplir se termina el bucle y sale del while.



Fuente: Elaboración propia

```
Inicio
  x = 0
  mientras x <= 10
    Inicio
      y = 3 * x + 5
      x += 0.5
      Imprimir(x,y)
    Fin
Fin
```

< d_codinGames />

Explicación de la imagen:

La condición while (condición) se evalúa antes de cada iteración (repetición). Esto significa que si se cumple (es verdadera) la condición while, el bloque de código dentro del bucle se ejecuta (// código a repetir). En cambio, si dicha condición no se cumple no se activará el bloque de código, y la ejecución continúa con la siguiente instrucción después del bucle.

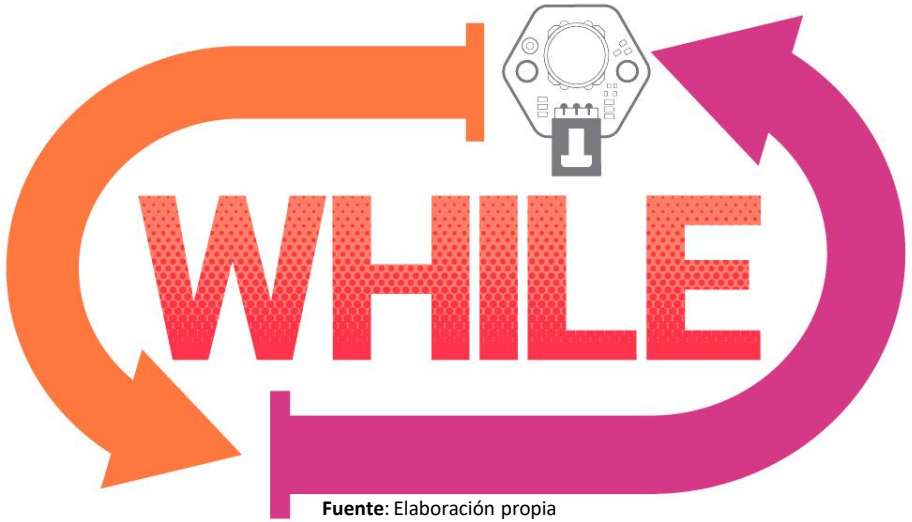
Sintaxis

```
while (condicion) {
    // Código a repetir
}
```

Posible No Ejecución: Es posible que el bloque de código dentro del bucle while no se ejecute en absoluto si la condición inicialmente es falsa.

Sintaxis

```
while (condicion) {
    // Código a repetir
}
```



Fuente: Elaboración propia

Código

```
9       int i;
10      i=1;
11      while (i <= 5) {
12          cout << i << endl;
13          i++;
14      }
15  }
```

Nota: Se debe declarar una variable inicial, dependiendo del ejercicio.

Debe ingresar por lo menos una vez al ciclo.

Nota1:

Una variable **incrementadora** se utiliza para aumentar o disminuir un valor de forma constante en cada iteración, generalmente en pasos fijos, como **+1 o -1**. Su propósito es controlar el número de repeticiones de un ciclo o avanzar en una secuencia. Por ejemplo, **i++** en un ciclo **while o for** es un incrementador típico, que se usa para seguir contando de manera uniforme.

```
#include <iostream>
using namespace std;
int main(){
  int cont = 1,num;
  cin>>num;
  while(cont<=10){
    cout<<num*cont<<endl;
    cont++;//cont=cont+1;
  }

  return 0;
}
```

Nota 2:

Las variables reciben el nombre de contador, se les pueden asignar cualquier carácter (i).

Los contadores almacenan las veces que se repite la acción, incrementando o decrementa dependiendo del ejercicio.

Nota 3:

Podemos decir que depende del ejercicio tenemos variables **acumuladoras o sumadoras**.

Acumuladoras: quiere decir que almacenan algo.

Contadora: lleva o cuenta las veces que se repite una acción.

Nota 3:

Podemos decir que depende del ejercicio tenemos variables **acumuladoras o sumadoras**.

Acumuladoras: quiere decir que almacenan algo.

Contadora: lleva o cuenta las veces que se repite una acción.

Variables contadoras:

Nos guarda la cantidad de monedas depositadas en una alcancía.
Una moneda de 1000
Este nos guarda 1
Cada vez que se ingrese una moneda va contando de una en una para saber cuántas se han ingresado a la alcancía.

```
Contador = contador + constante;

Contador = contador + 1;
```

```
Acumulador = acumulador + variable;

Acumulador = acumulador + valor;
```

Variables Acumuladoras: **N**os permite sumar un conjunto de valores.

Acumulador guarda el valor del billete o moneda.
En si suma un conjunto de valores. Para saber el acumulado de la alcancía de monedas de 1000
Si tengo **50 monedas** y cada una tiene un valor de $1000 nuestro acumulador seria $50.000.

Ejercicio 1

1:Realizar un programa en C++ que cuente los cinco primeros números naturales, **de 1 a 100**.

Realizar el análisis, diagrama de flujo, prueba de escritorio y código en C++

Nota: Se debe declarar una variable inicial, dependiendo del ejercicio.

Debe ingresar por lo menos una vez al ciclo.

1:Realizar un programa en C++ que cuente los cinco primeros números naturales, **de 1 a 100.**

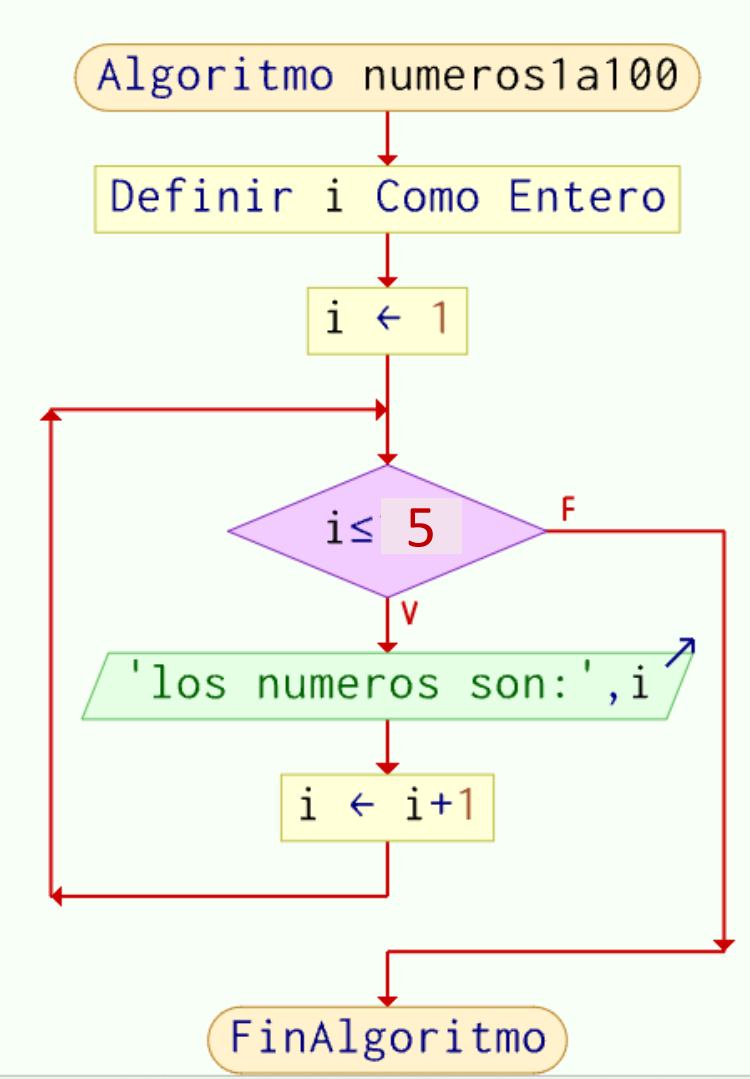
Realizar realiza análisis, diagrama de flujo, prueba de escritorio y código en C++

análisis

Variable	**i**
Incremento	**i=i+1** **i++**

1
.
20
.
30
87
90
.
100

Diagrama de flujo



Fuente: Elaboración propia

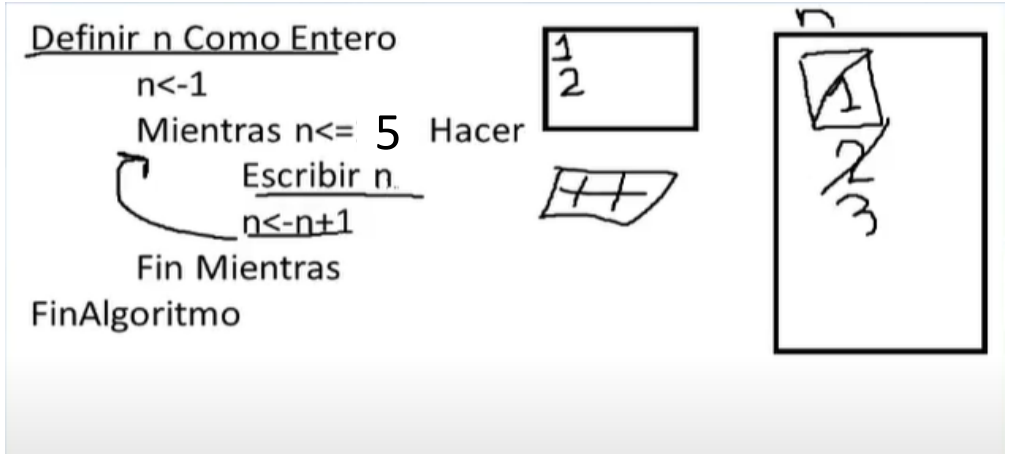
//Realizar un programa en C++ que cuente los cinco primeros números naturales, de 1 a 5.
// ejemplo 1 2 3 4 5….100

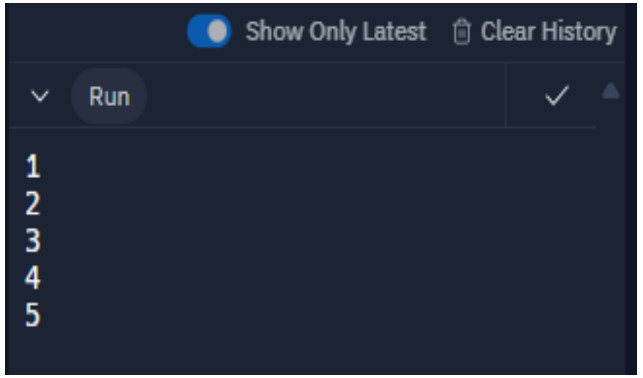
```
#include <iostream>
using namespace std;
int main() {
 //declaracion de variables
    int i;    i=1;
//sentencia while
 while(i<=100){
        cout << i << " ";    i++;
   }
  return 0;
}
```

Código

```
12  #include <iostream>
13  using namespace std;
14  int main() {
15
16  //declaracion de variables
17     int i;
18     i=1;
19  //sentencia while
20  while(i<=5){
21     cout << i << " ";
22       i++;
23      }
24       return 0;
25  }
```

Prueba de escritorio





Ejercicio 2

Realizar un programa en C++ que cuente los cinco primeros números naturales, de 1 a 5.

Realizar el análisis, diagrama de flujo, prueba de escritorio y código en C++

Nota: Se debe declarar una variable inicial, dependiendo del ejercicio.

Debe ingresar por lo menos una vez al ciclo.

1:Realizar un programa en C++ que cuente los cinco primeros números naturales, **de 1 a 100.**

Realizar realiza análisis, diagrama de flujo, prueba de escritorio y código en C++

análisis

Variable	**i**
Incremento	**i=i+1** **i++**

1
2
3
4
5

Análisis de código

```
#include <iostream>
using namespace std;
int main() {
//declaracion de variables
  int i;
  i=1;
//sentencia while
while(i<=5){
  cout<<i<<endl;
    i++;
}
}
```

En la sentencia While primero se piensa y luego se actúa

Mi entras se cumplan la instrucción lógica se van a seguir ejecutando las instrucciones

Ejercicio 3

Realizar un programa en C++ que imprima la suma de los números positivos.

Realizar el análisis, diagrama de flujo, prueba de escritorio y código en C++

Nota: Se debe declarar una variable inicial, dependiendo del ejercicio.

Debe ingresar por lo menos una vez al ciclo.

Código

```
1   //Realizar un programa en C++ que imprima la suma d ellos
    números positivos.
2
3   #include <iostream>
4   using namespace std;
5   int main() {
6     int numero;
7     int suma=0;
8     cout << " ingrese un numero: ";
9     cin >> numero;
10    while (numero >= 0){
11      suma = suma + numero;
12      cout << "ingrese mas numeros: ";
13      cin >> numero;
14
15    }
16    cout << " la suma es: " << suma;
17    }
```

```
 ingrese un numero: 4
ingrese mas numeros: 6
ingrese mas numeros: -9
 la suma es: 10
```

Código ejecutado

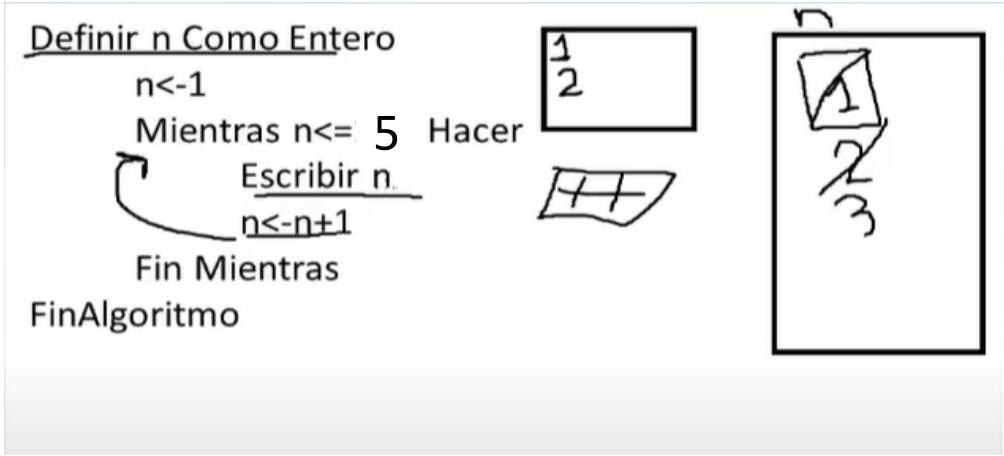
```
5   #include <iostream>
6   using namespace std;
7   int main() {
8     //define variable
9       int i;
10      i=1;
11      while (i <= 5) {
12          cout << i << endl;
13          i++;
14      }
15  }
16
```

Show Only Latest Clear History

Run

```
1
2
3
4
5
```

Prueba de escritorio

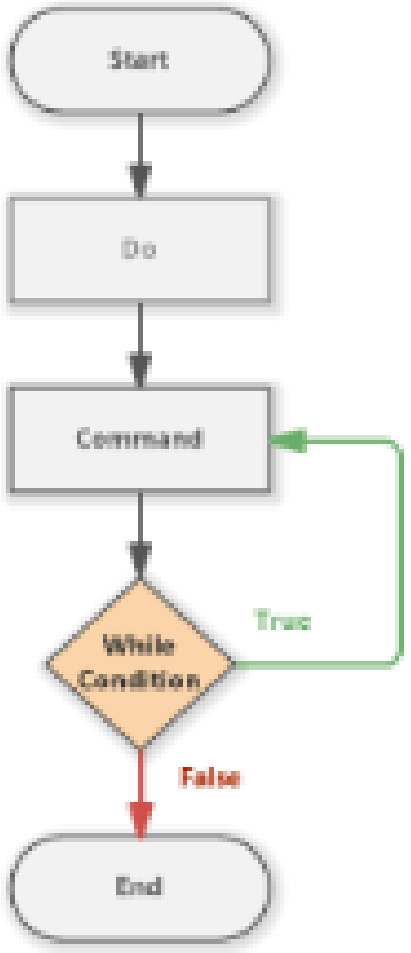


DO WHILE (Hacer Hasta) en C++

El bloque de instrucciones se realizará mientras la condición se cumpla. Es una estructura post-condición, pues, la expresión lógica se comprobará después de haber realizar por primera vez el bloque de instrucciones. En pocas palabras, siempre se realizará el bloque de instrucciones por lo menos una vez. verdadera.

DO WHILE (Hacer Hasta)

Diagrama de flujo



Fuente: Elaboración propia

Código

```
#include <iostream>
using namespace std;
int main(){
   int cont = 1,num;
   cin>>num;

   do{
      cout<<num*cont<<endl;
      cont++;
   }while(cont<=10);

   return 0;
}
```

DO WHILE (Hacer Hasta)

Do while (replit

```
/*la sentencia Do while

    do{
            OJO abre y cierra las llaves
    }while (expresión lógica);
```

}while (i<=10); **Recuerde abrir y cerrar paréntesis y al final lleva punto y coma**

DO WHILE (Hacer Hasta)

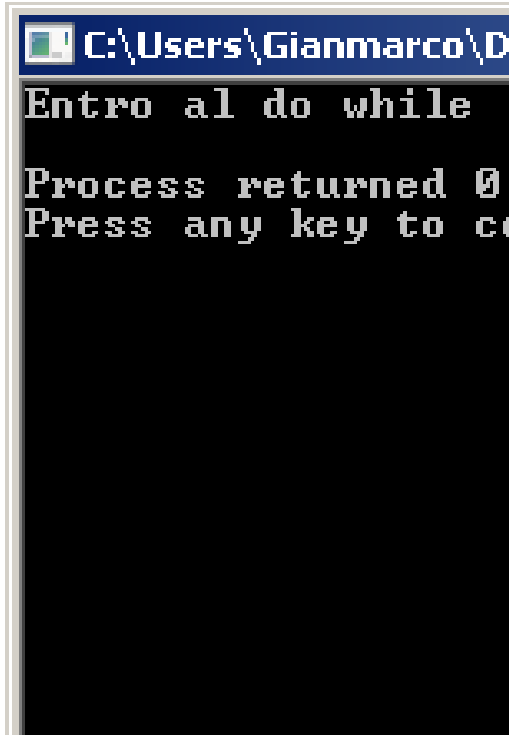
```
/*la sentencia Do while

    do{

}while (expresión lógica);
```

OJO: Nota que la estructura do-while termina con un ; punto y coma al final de la expresión lógica.

DO WHILE (Hacer Hasta)

Se pone en condición que ambos bloques de instrucciones se repetirán mientras la variable x (que tiene de valor 10) sea diferente de 10. En ningún caso esta condición cumple, pero solo entra al bloque do-while pues la condición es comprobada después.

```
#include <iostream>
using namespace std;
int main(){
   int var = 10;
   while(var != 10){
      cout<<"Entro al while"<<endl;
   }
   do{
      cout<<"Entro al do while"<<endl;
   }while(var!=10);

   return 0;
}
```



Ejercicio 1

Realizar un programa que pida un número hasta que sea positivo.

Realizar el análisis, diagrama de flujo, prueba de escritorio y código en C++

Código ejecutado

```
// Pide un número hasta que sea positivo
int numero;
do {
    cout << "Ingrese un número positivo: ";
    cin >> numero;
} while (numero <= 0);
```

Ejercicio 2

Realizar un programa que cuente los números de 1 a 10.

Realizar el análisis, diagrama de flujo, prueba de escritorio y código en C++

Código para digitar en GDB

main.cpp > main Format

```cpp
//Hacer un programa que cuente los números de 1 a 10

#include <iostream>
using namespace std;
int main() {
  //definir variables
  int i;
  i=1;
  //ciclo Do while
  do {
cout << i<< endl;
    i=i+1; // aumenta el iterador de uno en uno

  }while (i<=10);
}
```

Console

Run

```
1
2
3
4
5
6
7
8
9
10
```

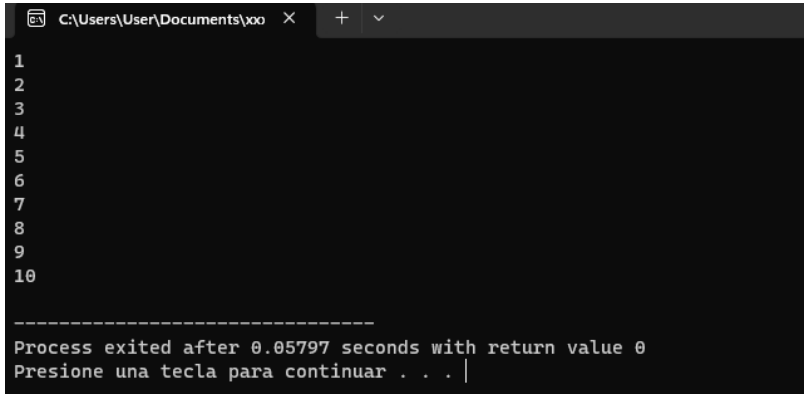
CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Código

```
3  #include<iostream>
4  using namespace std;
5  int main()
6  {
7      int i;
8      i=1;
9      do{
10         cout<<i<<endl;
11         i=i+1;
12     }while(i<=10);
13     return 0;
14 }
```

Resultado al ejecutar



Ejercicio3

Realizar un programa que cuente los números del 10 a 1.

Realizar el análisis, diagrama de flujo, prueba de escritorio y código en C++

main.cpp

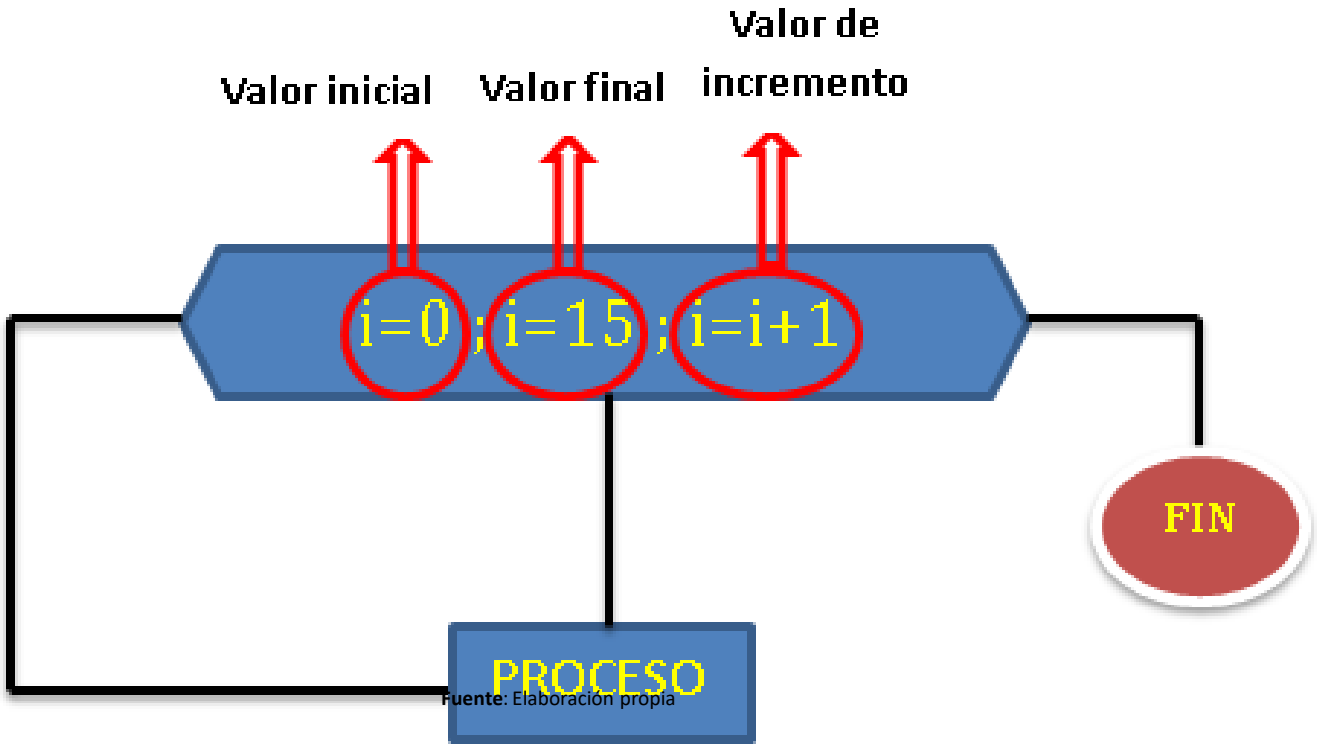
```
4   int main( ) {
5     //definir variables
6     int i;
7     i=10;
8     //ciclo Do while
9     do {
10  cout << i<< endl;
11        i=i-1; // decrementa el iterador de uno en uno
12
13    }while (i>=1);//condición finaliza con punto y coma
14  }
```

Console

Run

```
10
9
8
7
6
5
4
3
2
1
```

Ciclo FOR (Para) en C++



Fuente: Elaboración propia

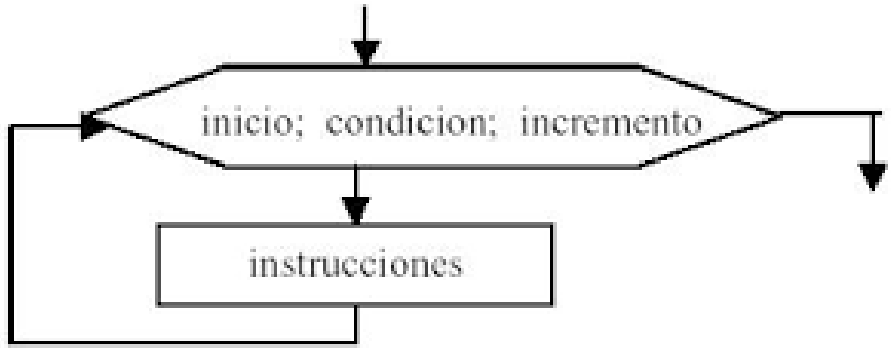
Ciclo FOR (Para) en C++

El ciclo for es una estructura de control en programación ya sea bucle o ciclo que nos permite ejecutar una o varias líneas de código de forma iterativa, en ella se puede indicar el número mínimo o máximo de repeticiones, en esta estructura conocemos el valor inicial y final a diferencia del while o do while

Sintaxis

```
while (condicion) {
    // Código a repetir
}
```

Estructura en Flujograma o diagrama de flujo



Fuente: Elaboración propia

Estructura

```
for (valor inicial; condición de seguimiento; valor actual + incremento){
   instrucciones;
};
```

Estructura en pseudocódigo y código C/C++

Pseudocódigo en español

```
Repita para (expr1; expr 2; expr3)
   S1
   ….
   Sn
fin_rp
```

Código en C

```
for (expr1; expr 2; expr3)
{
   S1
   ….
   Sn
}
```

Conceptos básicos

Instrucciones, acciones que se realizarán en cada ciclo del bucle y que tendrán relación con la condición de finalización.

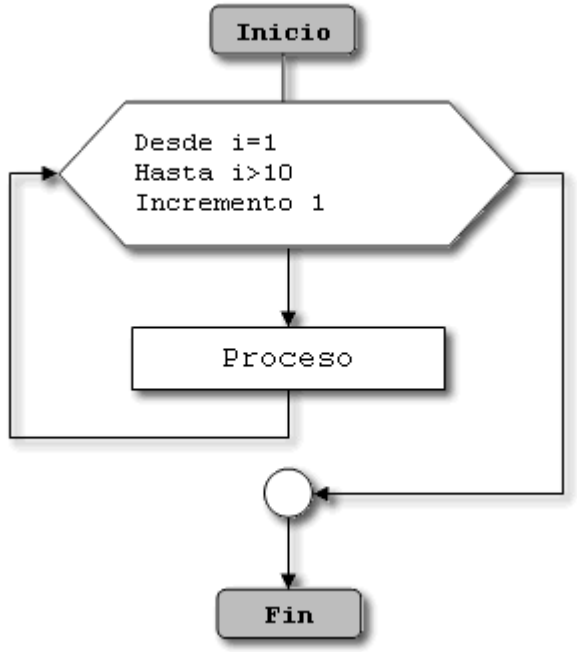
Para saber cuándo es necesario utilizar el bucle FOR tienes que hacerte las siguientes preguntas:

¿Cuándo pararé el ciclo?
¿Necesito comprobar las condiciones antes o después de las instrucciones?
¿Puedo definir el valor inicial/valor actual en la propia instrucción del bucle?

Ciclo FOR (Para) en C++

Generalmente se utiliza cuando se conoce el número de veces que se repetirá el bucle.
La estructura (Repetir o para) necesita:

- Un contador inicial al menos uno para entrar el ciclo.
- Una condición que nos evaluará la existencia del valor final del contador de la estructura
- El incremento o decremento de la expresión cada vez que se repita.

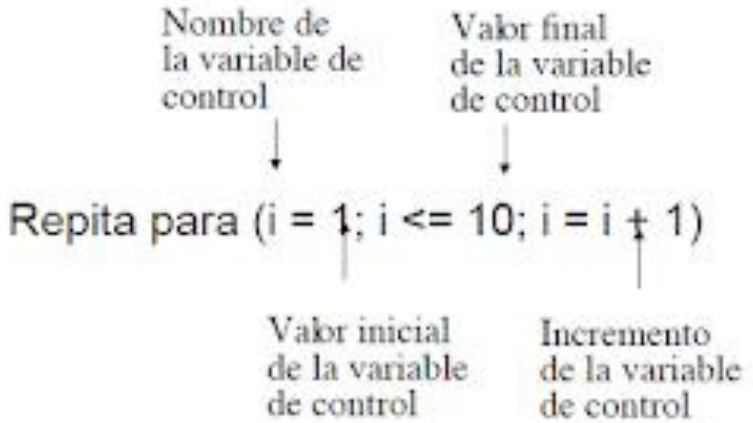


Fuente: Elaboración propia

Ciclo FOR (Para) en C++

El ciclo for es muy potente y completo a diferencia de sus colegas, incluye las tres expresiones en una sola estructura, el inicio, la condición de salida y el incremento o decremento, este bucle tiene algunas variaciones y depende de cada quién como lo use.

Como observación cada una de las tres expresiones ese opcional, pues el for tiene variantes.

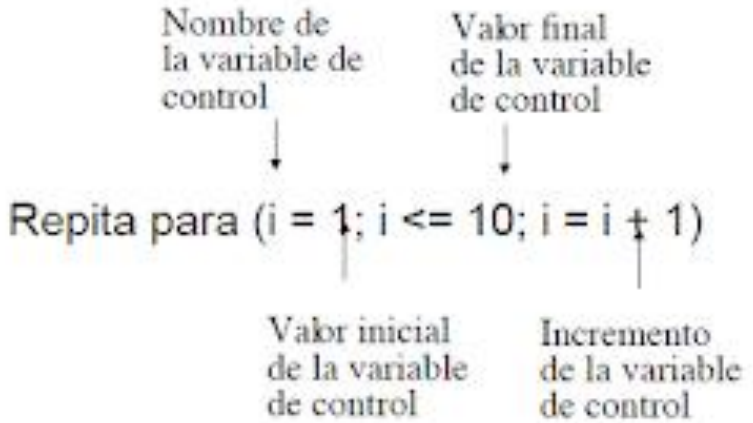


Estructura del for

Ciclo FOR (Para) en C++

El ciclo for es muy potente y completo a diferencia de sus colegas, incluye las tres expresiones en una sola estructura, el inicio, la condición de salida y el incremento o decremento, este bucle tiene algunas variaciones y depende de cada quién como lo use.

Como observación cada una de las tres expresiones ese opcional, pues el for tiene variantes.



Estructura del for

Algunas variantes del for

1.-Lazo infinito

```
for( ; ; )
  {  }
```

1.1 -Variación simultánea de dos contadores

```
for (int i = 0, j = 100; i < j; i++, j--)
{
    printf(" i :  %i  j:  %i" ,i , j );
}
```

Algunas variantes del for

1.3-Variar el contador de (1 a 10) (0 al 9) "Incremento"

a.- `for (int  counter = 1; counter <= 10; counter++)`

b.- `for(int counter = 0; counter < 10; counter++)`

1.4 -Variar el contador de 100 a 1 "Decremento"

```
for (int i = 100; i >= 1; i--)
```

Algunas variantes del for

1.5-Variar el contador (I) de 7 a 77 (de 7 en 7)

```
for (int I = 7; I <= 77; I +=7)
```

1.6 -Variar el contador (I) de 20 a 2

```
for (int I = 20; I >= 2; I -=2)
```

1.7-Variar el contador (I) de 20 a 2

```
for (int j = 99; j >= 0; j -=11)
```

Ejercicio 1

Realizar un programa que muestre un mensaje “Bienvenidos al ciclo para”. El mensaje se debe mostrar hasta cinco veces:

```
Bienvenidos al ciclo para
Bienvenidos al ciclo para
Bienvenidos al ciclo para
Bienvenidos al ciclo para
Bienvenidos al ciclo para
```

Ejercicio 1

Realizar un programa que muestre un mensaje “Bienvenidos al ciclo para”. El mensaje se debe mostrar hasta cinco veces:

Código

```
#include <iostream>
using namespace std;
int main() {
  //Declaro la variable dentro del bucle
  //inicializar el iterador dentro del ciclo

 int i;
for (i=1;i<=5;i++)
cout << " Bienvenidos al ciclo para: " <<endl;

}
```

Ejercicio 2

Realizar un programa que imprima los números del 10 al 1

Ejercicio 2

Realizar un programa que imprima los números del 10 al 1

Código

```cpp
//Realizar un programa que imprima los números del 10 al 1

#include <iostream>
using namespace std;
int main() {
  for (int i=10; i>=1; i--){
    cout << i << endl;
  }
}
```

Run

```
10
9
8
7
6
5
4
3
2
1
```

Ejercicio 3

Realizar un programa en C++, que muestre los números pares hasta 100

Definir e Inicializar variable?

2
4
6
8
.
.
.
100

Código en C++

main.cpp

```cpp
//Realizar un programa en C++, que muestre los números pares hasta 100
#include <iostream>
using namespace std;
int main() {
  for (int i=2; i<=100; i+=2){
    cout<<i<<endl;
  }

}
```

Console

Run

```
2
4
6
8
10
12
14
16
18
20
22
24
```

Ejercicio 2

Realizar un programa que muestre la tabla de multiplicar de un numero entero.

Ingrese un numero

4

```
Console   Shell
Run
ingrese un numero entero: 4
4x1=4

4x2=8

4x3=12

4x4=16

4x5=20

4x6=24

4x7=28

4x8=32

4x9=36

4x10=40
```

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

```cpp
#include <iostream> //Incluye la biblioteca estándar para operaciones de entrada/sal
using namespace std;
//Función principal donde comienza la ejecución del programa.
int main()
{

    //Declara una variable entera para almacenar el número ingresado.
    int n;
    //Muestra un mensaje solicitando al usuario que ingrese un número
    cout << "ingrese un numero entero:" << endl;
    //Lee el número ingresado por el usuario y lo guarda en la variable.
    cin>>n;

    //mostrar el mensaje en pantalla
    cout << "Tabla de multiplicar del numero:"<<n<<endl;

     for (int i=1; i<=10; i=i+1){  //Inicializa i en 1;Se ejecuta mientras i sea men
     //o igual a 10; Incrementa i en 1 después de cada iteración
```

Código en C++

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

```
31
32          //imprime la tabla de multiplicar de acuerdo al valor de n
33          //Imprime cada línea de la tabla con el formato "n x i = resultado".
34          cout<<n<<"x"<<i<<"="<< n*i<<endl<<endl;
35      }
36      //Finaliza el programa indicando que terminó correctamente.
37      return 0;
38  }
```

Código en C++

Ciclo FOR (Para) en C++

Recuerda practicar y experimentar con condicionales y ciclos en tus proyectos para aprovechar al máximo su potencial en el desarrollo de software.

¡Gracias!

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA

Vigilada Mineducación

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA

Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"