C++

Fundamentos

Algoritmo

Secuencia de actividades previamente estudiadas, coordinadas y organizadas cronológicamente para resolver una actividad en particular.

Programa

Podemos describir que en términos generales un programa es como la descripción procesual de una actividad.

Esta descripción no puede variar ya que si cambia sería otra nueva.

Cómo es entonces que se puede atender la demanda de distintos usuarios.

Para que un programa le responda íntimamente a un usuario, emplea lo que comúnmente llamamos Variables.

Programa Genérico a Programa Particular

Cómo es entonces que se puede atender la demanda de distintos usuarios.

Para que un programa le responda íntimamente a un usuario, emplea lo que comúnmente llamamos Variables.

Receta - Torta

- Batí la manteca con el azúcar hasta formar una crema. Incorporá la esencia de vainilla y la ralladura de limón. Integrá bien todos los ingredientes.
- Agregá los huevos de a uno mientras seguís batiendo hasta tener una textura homogénea.
- Incorporá la harina previamente tamizada con el polvo para hornear y la leche poco a poco. Podés usar batidora o batidor de alambre. Para que no se formen grumos, unificá con movimientos envolventes la mezcla luego de cada taza.
- Una vez lograda una masa homogénea, batí nuevamente de forma envolvente para que ingrese aire. Una vez integrados todos los ingredientes batí en forma envolvente por 5 minutos.
- Enmantecá y enhariná un molde para torta.
- Volcá la preparación en el molde y cociná en horno moderado por 40 minutos aproximadamente.
- Comprobá si esta cocida la preparación pinchando con un cuchillo seco en el centro del bizcochuelo. Si sale limpio sacalo del horno.

Elección de IDE

Un entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI)

IDES

online : Replit, Edube.org, onlinegdb.https://cpp.sh/

TOP 5 REPLIT

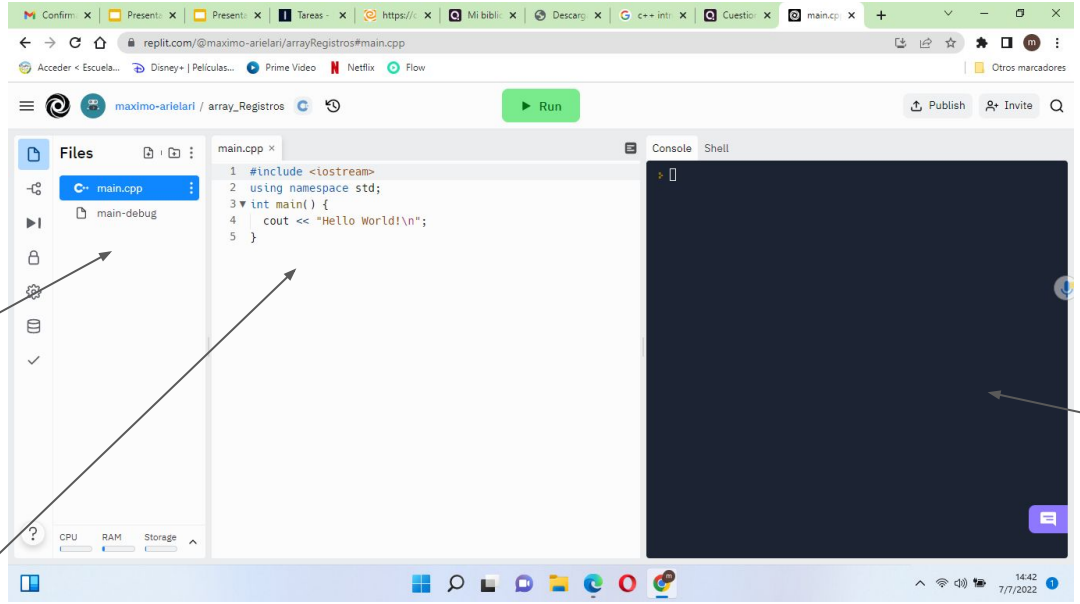
01	**Multiplataforma**	• Permite a los usuarios desvincularse de la realidad tecnológica de su Host.
02	**Online**	• El usuario se abstrae de todos los problemas que pueden ocurrir cuando se trabaja en un nodo local.
03	**No consume recursos**	• La simulación es corrida en una máquina virtual en el servidor de Re
04	**Compartir**	• Permite de un modo muy amigable compartir las actividades.
05	**Free**	• Con limitaciones nos permite generar una cuenta gratuita para emplear en la plataforma.

Replit

Permite a los usuarios escribir código y crear aplicaciones y sitios web mediante un navegador. El sitio también tiene varias funciones de colaboración, incluida la capacidad de edición multiusuario en tiempo real con una fuente de chat en vivo. Admite lenguajes de programación y marcado como Java , Python y HTML , lo que permite a los usuarios crear aplicaciones y sitios web, pero puede ejecutar cualquier lenguaje existente usando Nix .

https://replit.com

Replit



Manipulación de Archivos

Código Fuente

Consola Visualización de Resultados

Archivo Fuente Básico

```
#include <iostream>

using namespace std;

int main() {

  cout << "Hello World!\n";

}
```

#include <iostream>

iostream : Llamado a Librería que permite y soporta el flujo de datos desde el teclado y el monitor.

#include : Palabra reservada por el sistema para soportar la vinculación de otros archivos . Es una directiva de precompilación.

using namespace std;

Realmente cuando programamos en C++ estamos haciendo uso de la clase std. Su objeto instanciado es el que corre en nuestra aplicación y por ende la referencia a sus métodos se hace por medio de :

std::cout

std::cin

De esta forma, al indicar el namespace std, que es el namespace dónde está incluida toda la librería estándar de C++, ya no tenemos que indicar std::cout, sino que directamente indicamos cout.

int main()

Función Principal: Es el punto de partida de todo programa en C++.

int : Indica el tipo de función (Entera). Esto tiene que ver con su retorno de datos.

main: Nombre de la función principal. (palabra reservada).

(): Los paréntesis se emplean para pasar información a la función.

cout << "Hello World!\n";

cout : función para presentar información en la consola.

<< : Este símbolo se emplea para indicar el flujo de la información. Va en dirección de la consola.

“ Cadena” : El texto a presentar se define entre comillas.

\n: Este comando en consola representa línea nueva.

; :Toda sentencia de código debe tener este símbolo para finalizar la instrucción.

Manipuladores de presentación de Información

Un manipulador que está diseñado para cambiar la transmisión.

```
int byte = 255;

cout << hex << byte;

cout << byte << dec << byte;
```

Manipuladores de presentación de Información

Técnicamente, un manipulador es una función que cambia una de las propiedades del flujo de salida, llamada basefield. La propiedad se utiliza para determinar qué número debe usarse como base (como diría un matemático: una base) durante la conversión de cualquier int valor en texto legible por humanos.

Controlador	Representación
hex	**Hexadecimal**
dec	**decimal**
oct	**octal**

Más información.

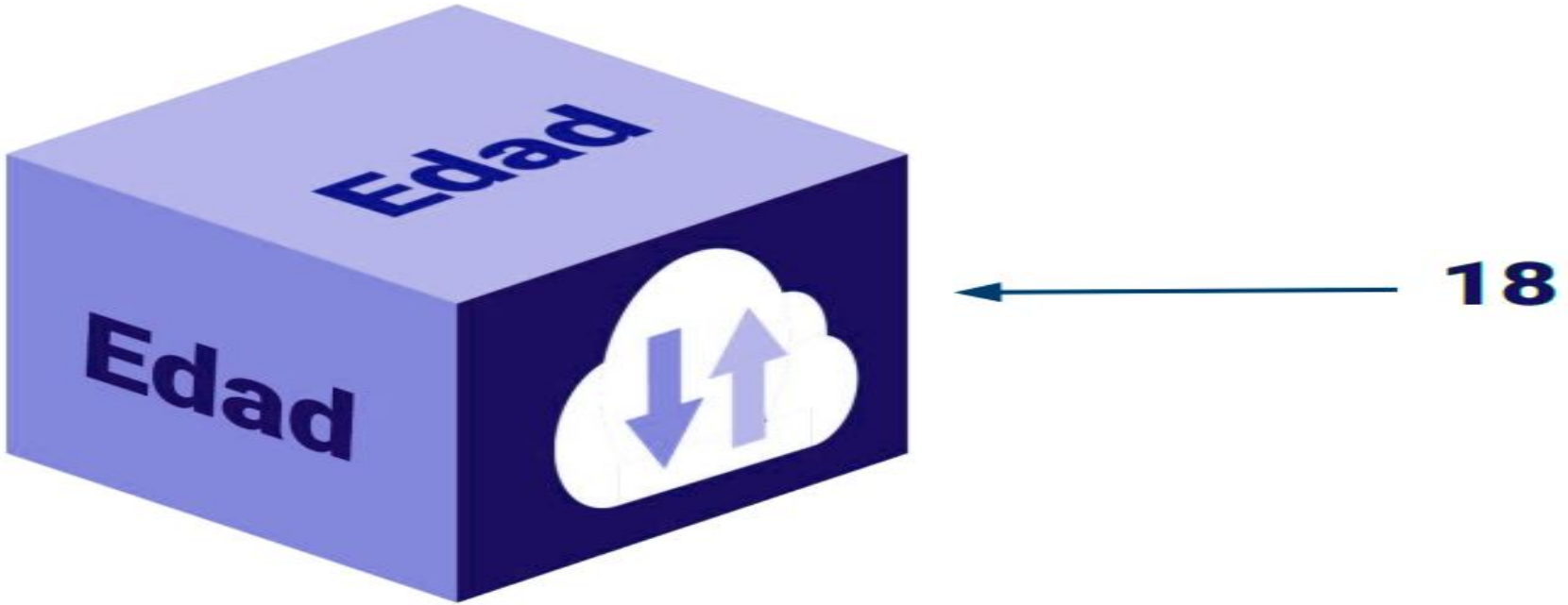
Variables

Variables

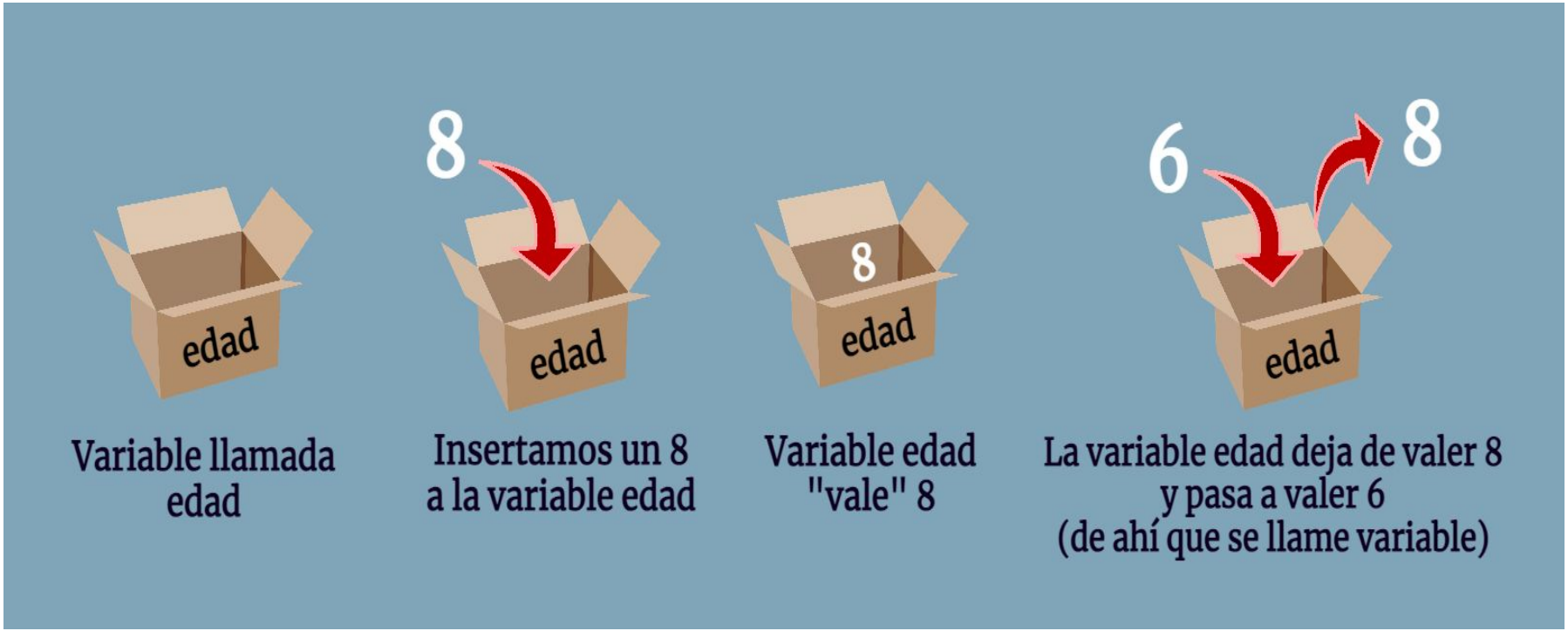
La función de los algoritmos es definir un relato de un proceso, pero la adaptación particular que se le da es en virtud a la información que se le suministra al sistema. A partir de esta es que se elabora una respuesta en particular para cada escenario propuesto.

Una variable es un contenedor de información que se ha solicitado para mantener y soportar información que es relevante en un momento dado, el ciclo de vida de la misma dependerá de su definición, también está influenciada por el tipo de campo de información que manipule. Para acceder a la misma tenemos que consignarle un nombre.

Variable



Variable


edad
Variable llamada edad
8
edad
Insertamos un 8 a la variable edad
8
edad
Variable edad "vale" 8
6
8
edad
La variable edad deja de valer 8 y pasa a valer 6 (de ahí que se llame variable)

Declaración de variables

Cuando queremos usar una variable necesitamos primero declararla, y para hacer esto tomamos un formato como sigue:

tipo-de-variable nombre-de-variable[=valor]

Lo que está entre corchetes es opcional, puedes o no dar el valor a la variable al mismo tiempo que la declaras. Podemos además declarar varias variables del mismo tipo separándolas con comas.

Declaración de variables

Si quisiéramos declarar variables como las básicas, sería así:

```
int T=1401, variable=1, a;

float pi=3.14, decimal, otra=0.23;

char letra='C', mas;

bool dicho=true, bandera=false;
```

Declaración de variables

Una variable puede tener cualquier nombre conformado por letras y/o números, que no incluya espacios, símbolos especiales, sea una palabra reservada del compilador (como el tipo de variable), ya lo tenga otra variable o función, o sea enteramente compuesta por números. Sí hay distinción de mayúsculas y minúsculas por lo que int Variable es diferente de int variable.

Limitaciones de las variables básicas

`int` Almacena típicamente cualquier número entero x donde $-2147483648 \leq x \leq 2147483647$

`float` Almacena típicamente cualquier número x donde $1.18e^{-38} \leq x \leq 3.40e^{38}$ con una precisión científica de 7 dígitos después del punto decimal.

`char` Almacena sólo 1 carácter de los 256 posibles del código ASCII

`bool` Almacena sólo un valor `true` o `false`

Constantes en C++

Las constantes son elementos que no varían en el tiempo. La forma de consignarla es emplear la palabra reservada const

Se debe indicar también el tipo de dato al que pertenece.

```
const int dato=20;
```

OPERADORES ARITMÉTICOS

Se llaman operadores aritméticos a aquellos que permiten realizar cálculos con valores numéricos para obtener un resultado. Los operadores aritméticos más habituales son la suma, resta, multiplicación y división. En C++ y en otros lenguajes disponemos de un operador adicional al que denominamos operador módulo (%), que nos permite obtener el resto de una división entre enteros.

Operador	Significado
Operador =	Asignación
Operador *	Multiplicación
Operador /	División
Operador %	Resto de división entera (mod)
Operador +	Suma
Operador -	Resta

OPERADORES ARITMÉTICOS

Existen otros operadores admitidos que constituyen formas de expresar abreviadamente una operación. Por ejemplo += se puede usar para indicar que la variable a la izquierda toma el valor resultante de sumarse a sí misma con la variable o expresión a la derecha. Si A=4 y se ejecuta A +=3; entonces A pasa a tomar el valor 7, equivalente a realizar la operación A = A + 3;

OPERADORES ARITMÉTICOS

Operador	Acción	Ejemplo	Resultado
=	Asignación Básica	X = 6	X vale 6
*=	Asigna Producto	X *= 5	X vale 30
/=	Asigna División	X /= 2	X vale 3
+=	Asigna Suma	X += 4	X vale 10
-=	Asigna Resta	X -= 1	X vale 5
%=	Asigna Modulo	X %= 5	X vale 1
<<=	Asigna Desplazamiento Izquierda	X <<= 1	X vale 12
>>=	Asigna Desplazamiento Derecha	X >>= 1	X vale 3
&=	Asigna AND entre Bits	X &= 1	X vale 0
^=	Asigna XOR entre Bits	X ^= 1	X vale 7
|=	Asigna OR entre Bits	X |= 1	X vale 7

OPERADORES RELACIONALES

Los operadores relacionales, también denominados operadores binarios lógicos y de comparación,

Se utilizan para comprobar la veracidad o falsedad de determinadas propuestas de relación (en realidad se trata de respuestas a preguntas). Las expresiones que los contienen se denominan

expresiones relacionales. Aceptan diversos tipos de argumentos, y el resultado, que es la respuesta a

la pregunta, es siempre del tipo cierto/falso, es decir, producen un resultado booleano.

OPERADORES RELACIONALES

Operador	Relación	Ejemplo	Resultado
<	Menor	X = 5; Y = 3; if(x < y) x+1;	X vale 5 Y vale 3
>	Mayor	X = 5; Y = 3; if(x > y) x+1;	X vale 6 Y vale 3
<=	Menor o igual	X = 2; Y = 3; if(x <= y) x+1;	X vale 3 Y vale 3
>=	Mayor o igual	X = 5; Y = 3; if(x >= y) x+1;	X vale 6 Y vale 3
==	Igual	X = 5; Y = 5; if(x == y) x+1;	X vale 6 Y vale 5
¡=	Diferente	X = 5; Y = 3; if(x ¡= y) y+1;	X vale 5 Y vale 4

OPERADORES LÓGICOS

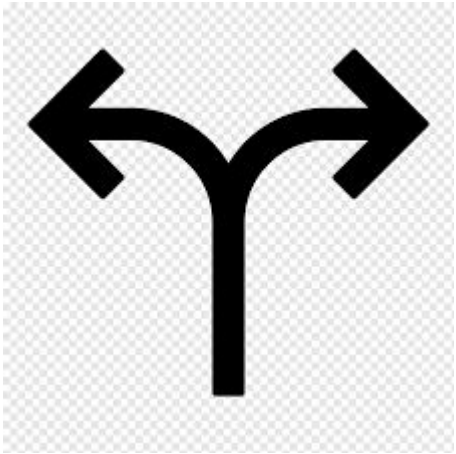
Los operadores lógicos producen un resultado booleano, y sus operandos son también valores lógicos o asimilables a ellos (los valores numéricos son asimilados a cierto o falso según su valor sea cero o distinto de cero). Por el contrario, las operaciones entre bits producen valores arbitrarios.

Operador	Acción	Ejemplo	Resultado
&&	AND Lógico	A && B	Si ambos son verdaderos se obtiene verdadero(true)
||	OR Lógico	A || B	Verdadero si alguno es verdader
!	Negación Lógica	!A	Negación de a

Condicionales

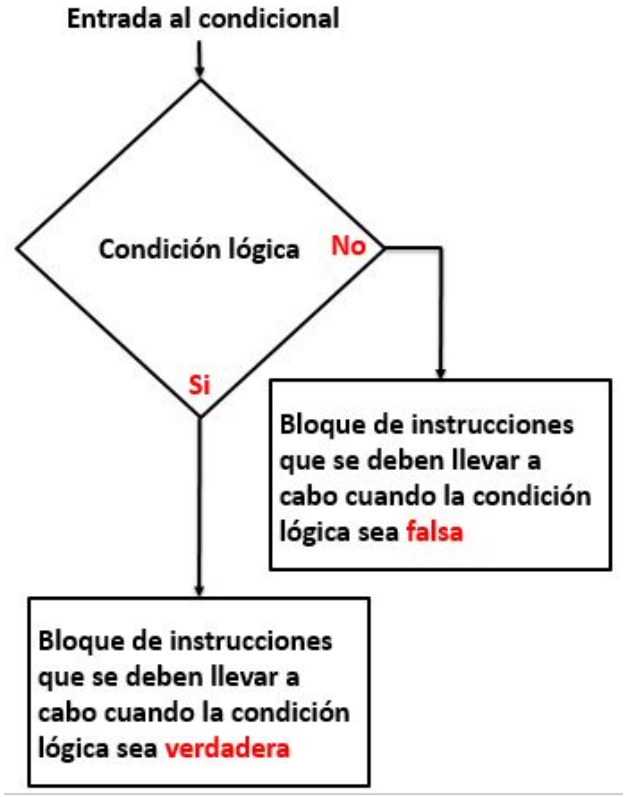
Condicionales en C++

Este tipo de procesos condicionales en C++, se puede llevar a cabo con las instrucciones if y else



Condicional if – else

Este tipo de sentencia condicional, toma como valor de comparación una sentencia lógica, por ejemplo a < b, en caso de la condición lógica llegar a ser verdadera, el flujo del programa tomará un rumbo y en caso de llegar a ser falsa, el flujo del programa tomará un rumbo diferente. Viendo desde la simbología de un diagrama de flujo, tendríamos lo siguiente.



Condicional if – else

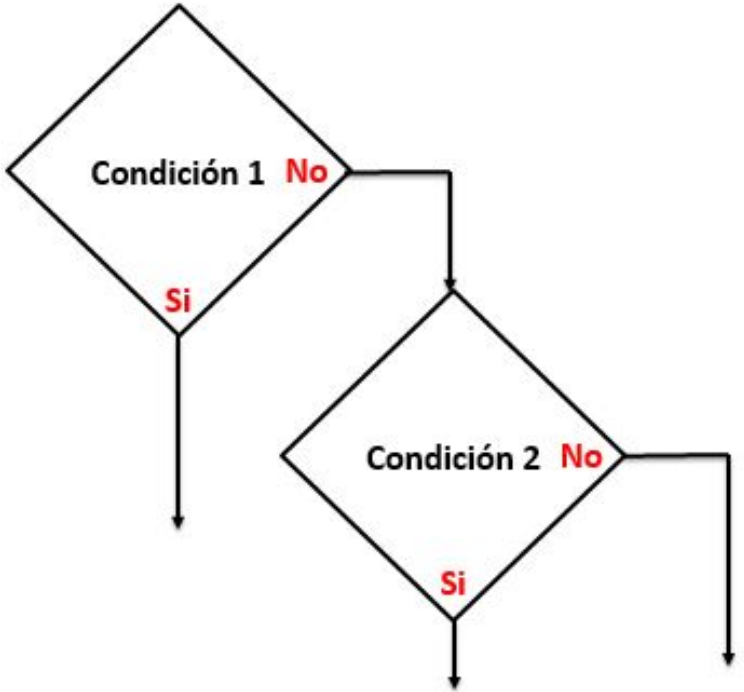
Entonces, para el caso de las sentencias if – else las cuales trabajan en conjunto. A la instrucción if se le agrega la condición lógica que se quiere evaluar, además entre corchetes se agregan el conjunto de instrucciones que debe ejecutar el programa en caso que la condición lógica sea verdadera, por su parte a la sentencia else se le agregan las instrucciones que el programa debe cumplir en caso que la condición anterior sea falsa.

Condicional if – else

```
if(a<b)
{
    cout<<"El numero menor es el guardado en la variable a\n";
}
else
{
    cout<<"El numero menor es el guardado en la variable b\n";
}
```

Condicionales Anidados

Muchas veces, requerimos evaluar varias condiciones distintas que van encadenadas y dependen del resultado de la comparación anterior, esto visualmente se podría ver como si tuviéramos una condición dentro de otra condición. A estos casos son a los que se les llama condicionales anidados.



Condicionales Anidados

```
if(a==b)//Se verifican si los dos valores son iguales
{
    cout<<"Los dos numeros son iguales\n";
}
else //Si no son iguales
{
if(a<b)//Se verifica si el valor guardado en a es menor al de b
{
    cout<<"El numero menor es el guardado en la variable a\n";
}
else //Si a no es menor
{
    cout<<"El numero menor es el guardado en la variable b\n";
}
}
```

Evaluar varias condiciones en un solo if

Para llevar a cabo estos tipos de comparaciones, es necesario hacer uso de los operadores lógicos AND (&&) y OR (||) ya que dependiendo de las necesidades del programa se requiere del uso de uno o del otro.

Si necesitamos que la condición if sea verdadera, solamente cuando todas las comparaciones sean verdaderas, entonces tendremos que utilizar un operador AND (&&).

Evaluar varias condiciones en un solo if

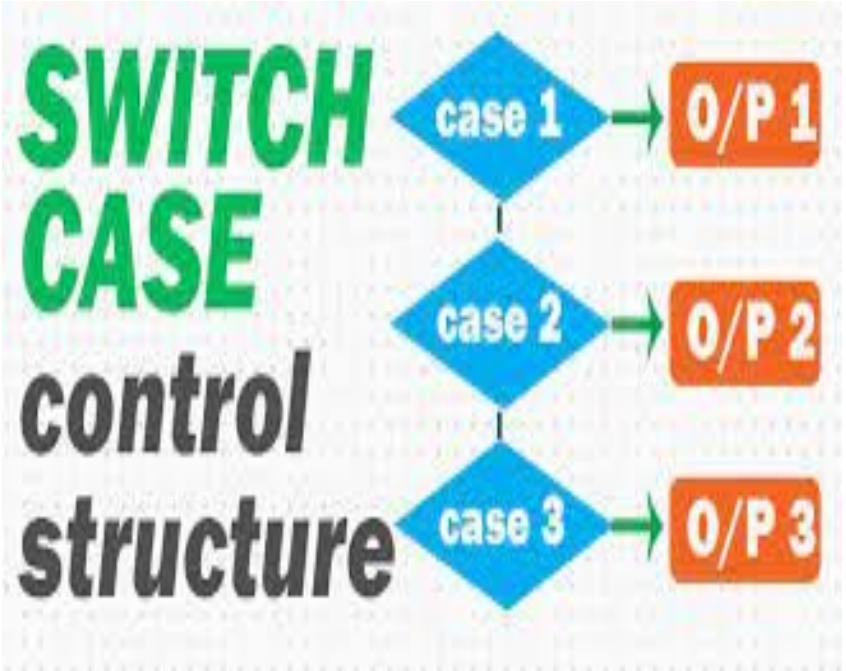
```
if(a>b && a>c && a>d)
{
    cout<<"El numero mayor es el guardado en la variable a\n";
}
else
{
    cout<<"El numero mayor NO es el guardado en la variable a\n";
}
```

INSTRUCCIÓN SWITCH – CASE EN C++

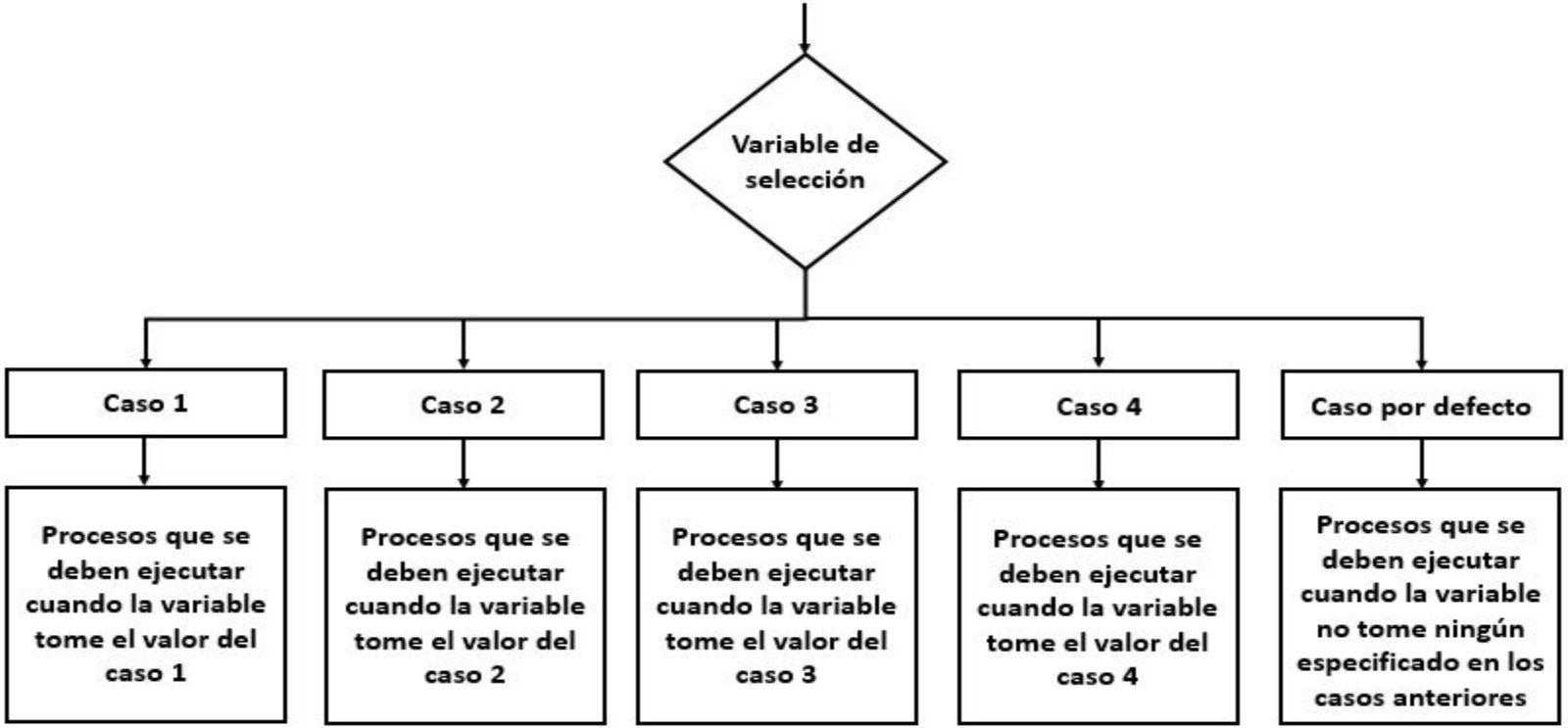
La instrucción switch – case, es utilizada como método de selección entre varios valores predeterminados que puede llegar tomar el programa en medio de su ejecución. Con esto se logra que dependiendo del valor que tome una variable, el programa siga por determinado camino especificado para dicho valor.

INSTRUCCIÓN SWITCH – CASE EN C++

Para el uso de esta instrucción, la variable que se utiliza como variable de selección debe ser de **tipo int o tipo char,** ya que no funciona con variables de tipos diferentes a estos dos. La estructura básica de un switch – case es la siguiente.



INSTRUCCIÓN SWITCH – CASE EN C++



INSTRUCCIÓN SWITCH – CASE EN C++

Entonces se puede observar, como dependiendo del valor que tome la variable de selección, el programa tomará uno u otro camino. Por último existe un caso especial, que es el caso por defecto, este último caso será tomado si el valor de la variable de selección no es igual al valor de ninguno de los casos anteriores.

INSTRUCCIÓN SWITCH – CASE EN C++

```
int opcion=0; //Declaración de varible para guardar la opción seleecionada por el usuario

//Impresión del menú y solicitud de datos
cout<<"Ingrese el numero de la opción del menú que quiere elegir:\n1- Opcion 1\n2- Opcion 2\n3- Opcion 3\nIngrese Opcion: ";
cin>>opcion; //Recepción y guardado del dato ingresado por el usuario

switch(opcion) //Se inicia la sentencia switch-case con la variable opción como variable de selección
{
case 1: //En caso que el valor de la variable opcion sea 1 entonces
    cout<<"Usted selecciono la opcion 1\n\n";
    break; //instrucción break para romper la ejecución de la instrucción switch-case y salir de la misma
case 2://En caso que el valor de la variable opcion sea 2 entonces
    cout<<"Usted selecciono la opcion 2\n\n";
    break;
case 3://En caso que el valor de la variable opcion sea 3 entonces
    cout<<"Usted selecciono la opcion 3\n\n";
    break;
default: //Si la variable no toma ningún valor de los antes declarado (1, 2 o 3)
    cout<<"Usted no ha seleccionado ningún valor valido\n\n";
    break;
}
```

INSTRUCCIÓN SWITCH – CASE EN C++

```
char opcion=' '; //Declaración de varible para guardar la opción seleecionada por el usuario

//Impresión del menú y solicitud de datos
cout<<"Ingrese el numero de la opción del menú que quiere elegir:\na- Opcion 1\nb- Opcion 2\nc- Opcion 3\nIngrese Opcion: ";
cin>>opcion; //Recepción y guardado del dato ingresado por el usuario

switch(opcion) //Se inicia la sentencia switch-case con la variable opción como variable de selección
{
case 'a': //En caso que el valor de la variable opcion sea a entonces
    cout<<"Usted selecciono la opcion 1\n\n";
    break; //instrucción break para romper la ejecución de la instrucción switch-case y salir de la misma
case 'b'://En caso que el valor de la variable opcion sea b entonces
    cout<<"Usted selecciono la opcion 2\n\n";
    break;
case 'c'://En caso que el valor de la variable opcion sea c entonces
    cout<<"Usted selecciono la opcion 3\n\n";
    break;
default: //Si la variable no toma ningún valor de los antes declarado (a, b o c)
    cout<<"Usted no ha seleccionado ningun valor valido\n\n";
    break;
}
```

Ciclos-Bucles

Ciclos

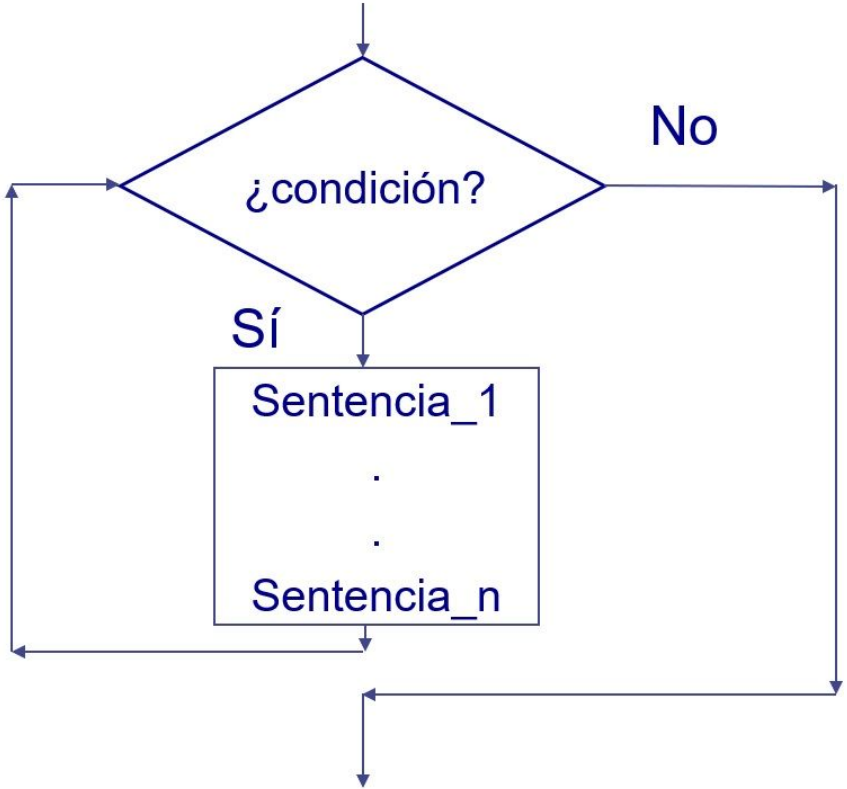
Estructuras de control que se emplean para repetir un conjunto conocido de instrucciones .

Las mismas circunscriben instrucciones por medio de llaves, determinan el conjunto de instrucciones que conforman el bloque de código que se repite.

La repetición es evaluada por una estructura condicional que opera bajo la lógica de Verdadero o Falso.

Ciclo while

Un ciclo while realiza un conjunto de instrucciones mientras una condición sea cierta (sea diferente de 0). Cuando la instrucción es 0 (o falso) se suspende la ejecución del ciclo.



Ciclo while

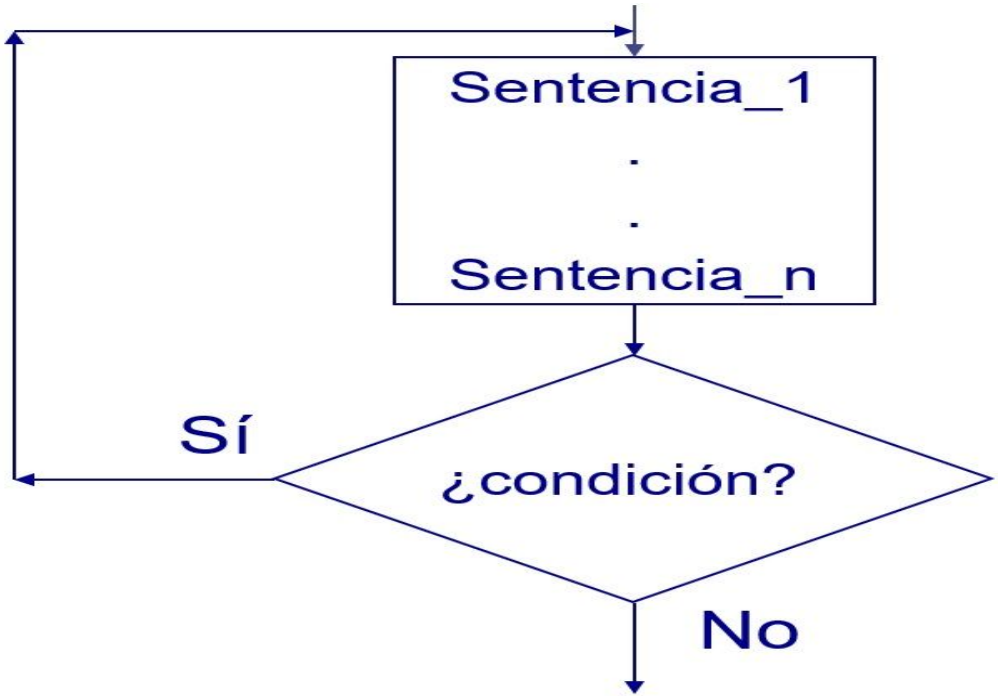
```
int numero;
cin >> numero;
while(numero <= 100)
{
    cout <<  "Ingrese un numero ";
    cin >> numero;
}
```

Ciclo do while

Es una variante especial del bucle while. Al contrario que el bucle while, que comprueba la condición antes de entrar en el bucle, el bucle do - while la evalúa al final del bucle. Esto implica que el bucle se ejecutará al menos una vez.

```
do
{
   bloque de sentencias;
}
while (condicion);
```

Ciclo do while



Ciclo For

El bucle for está concebido fundamentalmente para ejecutar sus sentencias asociadas un número fijo de veces. Por tanto, cuando el número de iteraciones necesarias para realizar una tarea en una parte del programa se conoce de antemano, lo más lógico es el empleo de una estructura tipo for, que controla automáticamente el número de repeticiones.

```
for (inicializacion; condicion; incremento)
{
   bloque de sentencias;
}
```

Ciclo For

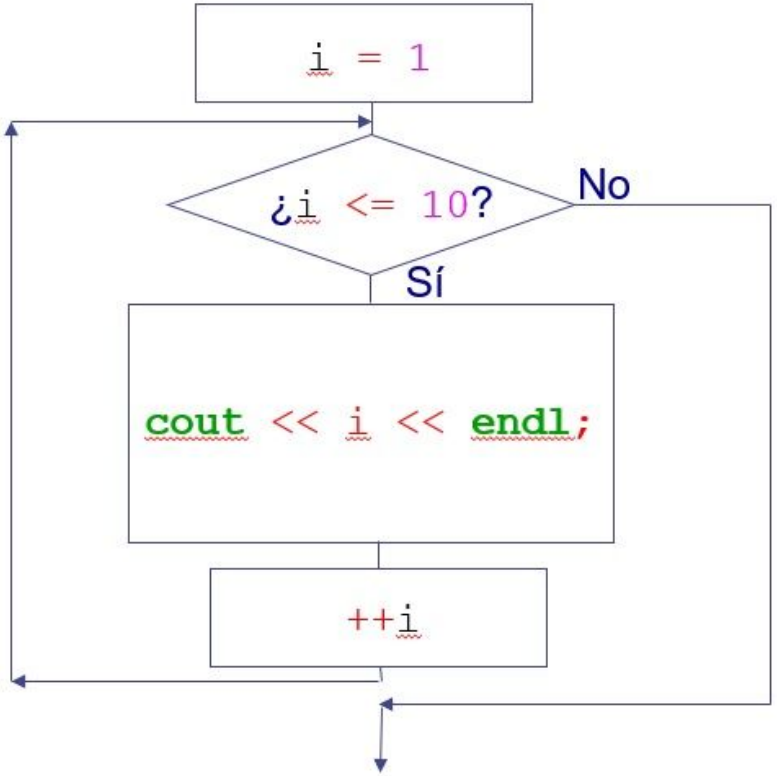
```
int i;

for (i = 1; i <= 10; ++i)

{

   cout << i << endl;

}
```



Operadores incremento y decremento

Son operadores unarios, es decir, actúan sobre un único operando

Operador	Descripción
`++x`	operador **preincremento**, cuando precede al operando
`x++`	operador **postincremento**, cuando sucede al operando
`--x`	operador **predecremento**, cuando precede al operando
`x--`	operador **postdecremento**, cuando sucede al operando

Operadores incremento y decremento

Si el operador antecede al operando, C++ utiliza el valor ya incrementado o decrementado.

```
int x = 10;

int y = ++x;
```

pone y a 11, X sigue valiendo 10.

Operadores incremento y decremento

Sin embargo, si se escribe el código como:

```
int x = 10;

int y = x++;
```

entonces toma el valor 10. En ambos casos **x** queda con valor **11**. La diferencia estriba en cuando cambia el valor.

Operadores incremento y decremento

Advertencia

Estos operadores no pueden utilizarse sobre un valor por la derecha.

```
x = (z+3)++;  //¡ERROR!
```

Operadores incremento y decremento

Opciones para incrementar/decrementar en una unidad, tenemos 3 posibilidades:

- x = x + 1;
- x += 1;
- ++x; (o x++;)

Operadores incremento y decremento

Analicemos qué instrucciones deben realizarse en la CPU para cada una de ellas.

Supongamos que la variable x se encuentra almacenada en un registro R de la CPU y sea A otro registro de la CPU, típicamente conocido como acumulador, donde ésta guarda el resultado de una operación.

Operadores incremento y decremento

x = x + 1;

Resolución :

MOVR,A: Mueve a A el contenido del registro R.

ADDA,1: Suma 1 (podría ser otro valor). Se ha evaluado la expresión x + 1 a la derecha del operador de asignación =.

MOVA,R: Mueve el resultado desde A a la posición R. Es el efecto del operador =.

Operadores incremento y decremento

x += 1;

ADDR,1: Suma 1 (podría ser otro valor) directamente al contenido del registro R.

++x;

INCR: No realiza una suma a nivel hardware. Un circuito especial de la ALU permite incrementar (o decrementar) una variable en una unidad de forma más rápida.

Conversión (un fenómeno que ocurre cuando los datos están sujetos a cambio de tipo).

Lo que significa es que podemos ver el código ASCII de cualquier carácter almacenado dentro de una char variable y viceversa, o ver un carácter cuyo código ASCII está colocado dentro de una int variable.

static_cast<newtype>(expr)

https://replit.com/join/xafxkovxhu-maximo-arielari

Anomalías numéricas

la suma computacional no siempre es conmutativa . ¿Sabes por qué? Imagine que tiene que agregar una gran cantidad de valores de punto flotante: algunos de ellos son muy grandes, otros muy pequeños (cercanos a cero). Si se suma un valor flotante muy pequeño a otro que es muy grande, el resultado puede ser bastante sorprendente.

Eche un vistazo a la imagen: asumimos que nuestra computadora solo guarda 8 dígitos precisos de cualquier archivo float. Si sumamos estos dos flotantes, probablemente obtendremos:

Anomalía numérica

No podemos evitar estos efectos cuando sumamos/restamos los números de tipo float(y también de sus primos, porque también se ven afectados por este problema). El fenómeno aquí descrito es lo que llamamos anomalía numérica .

https://replit.com/join/rcijbygqhu-maximo-arielari

Bibliografía

https://www.include-poetry.com/Code/C++/Introduccion/Variables/#:~:text=Las%20variables%20b%C3%A1sicas%20en%20C,char%20Para%20valores%20tipo%20car%C3%A1cter.

https://es.wikipedia.org/wiki/Algoritmo

https://geekelectronica.com/condicionales-en-c/

https://geekelectronica.com/instruccion-switch-case-en-c/

https://www2.eii.uva.es/fund_inf/cpp/temas/6_control_flujo_iterativo/for.html

https://www2.eii.uva.es/fund_inf/cpp/temas/6_control_flujo_iterativo/operadores_incremento_decremento.html

https://wilsonquispe.vercel.app/posts/variables

https://es.wikipedia.org/wiki/Coma_flotante