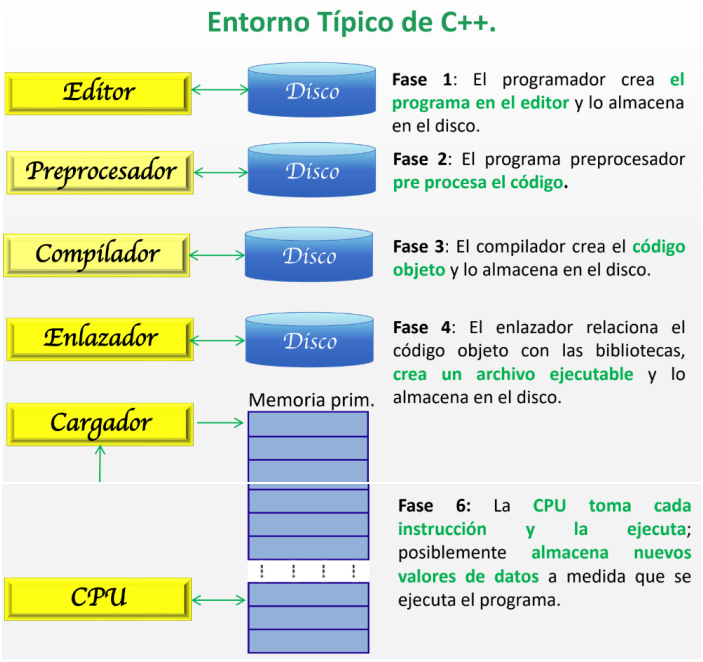
Código de colores:

- Resumen del tema
- Notas importantes.

1. Entorno típico.



2. Compiladores.

Es un lenguaje de programación que traduce puto el que lo lea código fuente escrito en código máquina, para ser ejecutados por la máquina.

2.1 Proceso de traducción.

Existen 2 etapas al momento de traducción.

a) Análisis. Se analiza la entrada para entender que se intenta comunicar, lo cual arroja una representación de entrada y permite que la siguiente etapa se desarrolle con facilidad.
   a. Análisis léxico: El compilador lee el código fuente caracter por caracter y lo agrupa en "tokens" o "palabras" significativas.
   b. Análisis Sintáctico: Una vez identificados los tokens, el compilador verifica que la estructura y sintaxis del código sea correcta de acuerdo a las reglas del lenguaje.
   c. Análisis semántico: Verifica que el código tenga significado y sea correcto en cuanto al significado de las palabras.

El análisis léxico identifica los tokens, el sintáctico verifica la estructura gramatical y el semántico verifica que tenga sentido y sea correcto en significado. Juntos aseguran que el código esté bien escrito y sea entendible por la computadora.

b) Síntesis. Toma la representación, el resultado del análisis y la transforma en equivalente pero en el lenguaje destino.
    a. Generación de código intermedio: El código fuente se convierte a una representación intermedia más simple, independiente de cualquier plataforma/arquitectura específica.
    b. Optimizaciones de código: Se aplican optimizaciones al código intermedio para que sea más eficiente y rápido, por ejemplo reordenando calculaciones o reduciendo operaciones redundantes.
    c. Generación de código objetivo: El código intermedio optimizado se convierte/traduce finalmente a código máquina u objeto que sí depende de la plataforma/arquitectura objetivo.

La síntesis convierte y optimiza el código analizado para generar código máquina nativo ejecutable por la computadora, realizando diferentes transformaciones y enlazándolo con bibliotecas necesarias para su ejecución.

3. Errores.

Sintaxis	Ejecución	Lógica
Estructura del código, el programa no puede compilar y se marca el error.	El programa se ejecuta pero no le das un dato correcto y se rompe.	Por pendejo, los resultados no son los correctos.

4. Principales librerías.

#include <iostream>// Librería que permite al programa imprimir datos en pantalla (input output stream, entrada y salida de datos).

#include <cstdlib> /* Contiene prototipos defunción para manipuladores de flujo que dan

formato a flujos de datos. Permitirá formatear y organizar la salida de datos.

#include <cmath>// Librería permite realizar operaciones matemáticas complejas.

using namespace std; //para tener acceso al paquete std donde hay funciones de C y no tener que escribir std::cout por ejemplo.

5. Función principal.

int main( ) {//comienza ejecución del programa las llaves encierran el cuerpo o definición de la función.

Decalracion de variables

Procedimientos.

return 0; /* como main es función tipo entero, debe retornar algo y con 0 indica que el programa terminó con éxito.*/

}

6. Reglas para identificadores.

Solo letras, números, algunos símbolos puto si lo volviste a leer no alfabéticos, case sentitive, se_utiliza_sanake_para_separar_palabras

Primer carácter no numero.

7. Declaración de variables.

tipo_de_dato nombre_de_variable[=valor_inicial];

Si es una constante (no se puede cambiar el valor durante la ejecución) agrega const al inicio.

8. Secuencias especiales de caracteres.

Nombre	Carácter
Fin de línea	\n
Tabulador	\t
Sonido	\b
Retroceso	\a
Comilla doble	\"
Comilla simple	\'
Signo de interrogación	\?
Diagonal invertida	\\

Nombre	Operador
Suma/Concatenación	+
Producto	*
Diferencia	-
Módulo	%
Incremento	++
Decremento	--
División	/

Nombre	Operador
Asignación	=
Asignación y suma	+=
Asignación y resta	-=
Asignación y multiplicación	*=
Asignación y división	/=
Asignación y módulo	%=

Nombre	Operador
Igual	==
Diferente	!=
Mayor que	>
Mayor o igual que	>=
Menor que	<
Menor o igual que	<=

Nombre	Operador
Or	||
And	&&
Not	!

9. Funciones.

conjunto de instrucciones que realizan una tarea específica. toman unos valores de entrada, llamados parámetros y proporcionan un valor (a que puto, lo sigue leyendo) de salida o valor de retorno; aunque tanto unos como el otro pueden no existir.

Necesitas especificar el tipo de datoq eu va a retornar, en caso de que no retorne nada, colocar void:

Void Nombre_funcion(parámetro){

Es una buena practica declarar la función en la sección superior del código (antes del main) pero generar el procedimiento después del main.

Ejemplo declaración: int Mayor(int , int);

10. Paso por referencia y paso por valor.

a) Paso por valor:

Al llamar a una función, se pasa una copia del valor de la variable. Cualquier cambio a la variable dentro de la función no afecta el valor de la variable original.

```
void funcion(int x) {
  x = 5;
}

int main() {
  int a = 3;
  funcion(a);
  // a sigue siendo 3
}
```

b) Paso por referencia:

Se pasa un "alias" o referencia a la variable original. Los cambios hechos dentro de la función SI se aplican a la variable original.

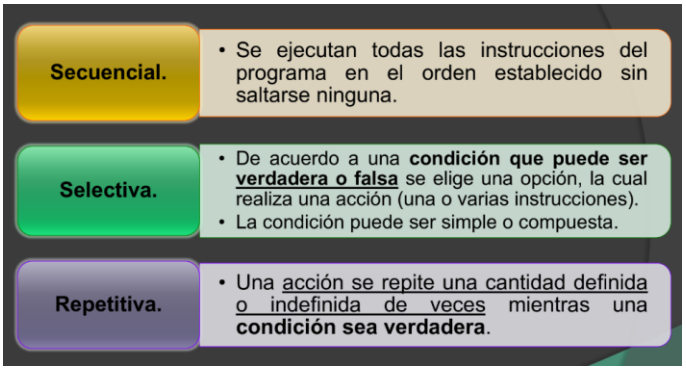
```
void funcion(int& x) {
  x = 5;
}

int main() {
  int a = 3;
  funcion(a);
  // a ahora vale 5
}
```

Paso por valor: se pasa una copia, no se modifica el original.

Paso por referencia: se pasa un alias, si se modifica el original.

11. Estructuras de control.



12. Fstream

Para trabajar con archivos se necesita incluir la biblioteca <fstream>. Esta permite asociar "streams" a los archivos para operar con ellos.

- Las principales operaciones sobre archivos son: creación, apertura, lectura, escritura y desplazamiento a lo largo del contenido.
- Se abren archivos con constructores como "ofstream" para escritura o "ifstream" para lectura. También se pueden abrir en modo lectura/escritura.
- Es importante cerrar los archivos con ".close()" cuando ya no se utilizan para liberar recursos del sistema.
- La escritura y lectura se realiza de forma similar a la consola, con los operadores "<<" para escribir y ">>" para leer, pasando el "stream" asociado.

```cpp
int main(){
    int outval = 15;
    //Escritura de archivo
    ofstream WriteFile("Mitexto.txt");
    WriteFile << outval << "\n";
    WriteFile.close();

    //Lectura de un archivo
    int inval;
    ifstream ReadFile("Mitexto.txt");
    ReadFile >> inval;
    ReadFile.close();

    cout << "El valor escrito en archivo es: " << inval;
    cout << endl << endl;

    return 0;
}
```