1) C++ es un lenguaje de programacion creado con la finalidad de ser una extension de C, orientado al manejo de objetos. Fue diseñado en 1979 y en los años venideros, se añadieron herramientas de programacion genericas, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos. Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

    a) C++ posee las siguientes estructuras de control de flujo:
        i) Secuenciación: Cada instruccion en C++ esta separadas por el caracter " ; ". Ademas, para separar bloques, se utili=zan los corchetes para definir un bloque de sentencias.

        ej:

```
{
    instrucción 1;
    instrucción 2;
    instrucción 3;
    .....
    instrucción N;
}
```

        ii) Seleccion: ofrece tanto seleccion simple con if-else, como seleccion multiple con switch.

- ej selección simple:

```
if (condicion)
    instrucción1;
else
    instrucción2;
```

Las instrucciones if-else se pueden anidar obteniéndose una estructura condicional múltiple:

```
if(condicion1)
  instrucción1;
else if(condicion2)
      instrucción2;
    else if(condicion3)
          instrucción3;
        else if(condicion4)
              instruccion4;
```

```
        else
         instrucción5;
```

- ej sleecion multiple (switch):

```
switch (expresión)
{
    case constante1:
        instrucciones;
        break;
    case constante 2:
        instrucciones;
        break;
    . . .
    default:
        instrucciones;
}
```

iii) Repeticion: dispone de tres estructuras reepetitivas, que son while, do-while y for.

- ej while:

```
while (condicion)
{
    instrucción 1;
    .............
    instrucción N;
}
```

- ej do-while:

```
do
{
    instrucción 1;
    .............
    instrucción N;
} while (condicion);
```

- ej for:

  ```
  for(inicialización; condicion; incremento/decremento)
  {
   instrucción 1;
   .....
   instrucción N;
  }
  ```

iv) Abstracción Procedural: C++ permite el uso de funciones

estas son definidas mediante la palabra clave function y debe tener el cuerpo encerrado entre llaves. Estan pueden o no, devolver un valor.

- ej funcion de suma:

  ```
  int x = 10;
  int y = 20;
  int z = suma(x, y);
  ```

v) Recursion: C++ permite la recursion. Esta sucede al llamar a una funcion dentro de si misma.

- ej funcion recursiva para calcular factorial:

  ```
  int factorial(int n) {
    if (n == 0) {
      return 1;
    } else {
      return n * factorial(n - 1);
    }
  }
  ```

vi) Concurrencia: En C++, la concurrencia se puede implementar mediante hilos. Los hilos son unidades de ejecución independientes que pueden ejecutarse simultáneamente.

Para crear un hilo en C++, se utiliza la función std::thread(). Esta función toma una función como parámetro, que se ejecutará en el hilo.

- ej funcion de hilos que imprime los numeros del 1 al 10:

```
#include <thread>
int main() {
  std::thread hilo1([&]() {
    for (int i = 1; i <= 10; i++) {
      cout << "Hilo 1: " << i << endl;
    }
  });
  std::thread hilo2([&]() {
    for (int i = 1; i <= 10; i++) {
      cout << "Hilo 2: " << i << endl;
    }
  });
  hilo1.join();
  hilo2.join();
  return 0;
}
```

vii) Excepciones: En C++, las excepciones se declaran utilizando la palabra clave throw. Para manejar una excepción, se utiliza la palabra clave try y catch.

- ej de un programa que lanza excepción si el valor de una variable es menor a cero:

```
#include <iostream>
int main() {
  int x = -1;
  try {
    if (x < 0) {
      throw x;
    } else {
```

```
    cout << "x es mayor o igual que cero" << endl;
  }
} catch (int e) {
  cout << "x es menor que cero: " << e << endl;
}
return 0;}
```

b) La evaluación de expresiones en C++ es normal, lo que significa que todas las expresiones se evalúan completamente antes de que se asigne un valor a una variable o se utilice una expresión en el lado derecho de una asignación.

   En C++, las expresiones se evalúan de izquierda a derecha, siguiendo las siguientes reglas de precedencia:

   - Operadores entre paréntesis: Los operadores entre paréntesis se evalúan primero, independientemente de su precedencia.
   - Potencias: Las potencias se evalúan a continuación.
   - Multiplicación, división y módulo: La multiplicación, la división y el módulo se evalúan a continuación, en orden de izquierda a derecha.
   - Suma y resta: La suma y la resta se evalúan a continuación, en orden de izquierda a derecha.
   - Operadores relacionales: Los operadores relacionales se evalúan a continuación.
   - Operadores lógicos: Los operadores lógicos se evalúan a continuación, en orden de izquierda a derecha.
   - Asignación: Los operadores de asignación se evalúan a continuación.
   - Las funciones se evalúan en el orden en que aparecen en el código.

Es importante tener en cuenta que las funciones pueden invocar otras funciones, lo que puede alterar el orden de evaluación.

c) Tipos de datos

Los tipos de datos simples de C++ son los siguientes:

- Char: Representa un carácter. El tamaño de un char es de 1 byte.
- Int: Representa un número entero. El tamaño de un int es de 4 bytes en la mayoría de los sistemas.
- Float: Representa un número de coma flotante. El tamaño de un float es de 4 bytes en la mayoría de los sistemas.
- Double: Representa un número de coma flotante de precisión doble. El tamaño de un double es de 8 bytes en la mayoría de los sistemas.
- Bool: Representa un valor booleano, que puede ser true o false. El tamaño de un bool es de 1 byte.
- Void: Representa un tipo sin valor. Se utiliza principalmente para declarar funciones que no devuelven ningún valor.

Los tipos de datos compuestos de C++ son los siguientes:

- Struct: Representa un tipo de datos compuesto que puede contener otros tipos de datos.
- Union: Representa un tipo de datos compuesto que puede contener un solo valor de un conjunto de tipos de datos.
- Class: Representa un tipo de datos compuesto que puede contener otros tipos de datos, así como funciones y métodos.
- Enum: Representa un conjunto de valores discretos.
- Tipos polimórficos

C++ tiene dos tipos de datos polimórficos:

- Punteros: Los punteros pueden apuntar a objetos de cualquier tipo, lo que permite el polimorfismo.
- Referencias: Las referencias pueden hacer referencia a objetos de cualquier tipo, lo que permite el polimorfismo.

d) El sistema de tipos de C++ es un sistema de tipos estático, lo que significa que el tipo de una variable o expresión se conoce en tiempo de compilación. Esto ayuda a garantizar que el código sea correcto y seguro.

El tipo de equivalencia para los tipos de datos básicos es la igualdad. Dos tipos de datos básicos son equivalentes si tienen el mismo tamaño y representación.

Por ejemplo, los tipos char, int y long int son todos equivalentes, ya que todos tienen un tamaño de 4 bytes.

Las reglas de compatibilidad para los tipos de datos básicos son las siguientes:

Operaciones aritméticas: Los tipos de datos compatibles para operaciones aritméticas son los tipos de datos básicos que tienen el mismo tamaño y representación.

Operaciones relacionales: Los tipos de datos compatibles para operaciones relacionales son cualquier tipo de datos básico.

Operaciones lógicas: Los tipos de datos compatibles para operaciones lógicas son cualquier tipo de datos básico.

Por ejemplo, los tipos int y double son compatibles para operaciones aritméticas, ya que ambos tienen un tamaño de 4 bytes.

Las reglas de compatibilidad para los tipos de datos compuestos son más complejas. En general, dos tipos de datos compuestos son compatibles si tienen la misma estructura.

C++ posee inferencia de tipos

Por ejemplo, la siguiente expresión declara una variable x y la asigna el valor 10:

C++

```
int x = 10;
```

En este caso, el compilador puede inferir que el tipo de la variable x es int.