C++

- by Harshil Bansal

Table of Contents

What is C ?

- C programming is a structural or procedural-oriented programming language.
- C is considered a middle-level programming language because it has the features of low-level language as well as high-level Language.
- Applications include:
  - Operating System
  - Language Compilers
  - Assemblers
  - Game Development
  - Text Editors

Disadvantages of C

- Does not support Object-Oriented Programming (OOP) features such as Polymorphism, Inheritance, and Encapsulation
- Inefficient Memory Management
- Does not allow Run Time Type Checking
- No support for Namespace
- Lacks Exception Handling
- Does not support the concept of constructors and destructors


You've got no class
C++
C

What is C++ ?

- C++ is a high-level computer programming language. It is an extension of the traditional C language with added support for object-oriented programming and other capabilities. Bjarne Stroustrup is the original creator of C++ in 1979 at AT&T Bell Labs.
- C++ is close to low-level languages and is considered one of the fastest programming languages. It provides complete control over memory allocation and management. C++ is used to develop complex and high-performance applications.
- Applications Include:
  - Operating Systems
  - GUIs
  - Games
  - Browsers
  - Database Engines
  - Cloud/Distributed Systems

Similarities in C & C++

- Both have a similar syntax.
- Their compilation is similar.
- Both languages follow the same concept of a stack, heap, and static variable.


WHEN YOU CAN FINALLY REST IN PEACE
BUT SOMEONE ON FB CLAIMS
THAT C++ IS BETTER THAN C
codeavail.com

C vs C++

C and C++ are programming languages that are used for developing applications, games, database systems, operating systems, and more.

- C is a procedural programming language and does not support objects and classes.
- C++ is an enhanced version of C programming with object-oriented programming support.

C vs C++

- C is a function-driven language while C++ is an object-driven language.
- In addition to Procedural C++ supports Object Oriented Programming as well.
- C++ supports User-defined data types in addition to Built-in data types.
- C doesn’t have access modifiers while C++ have them.
- C++ supports Exception Handling, Overloading, Reference variables, Data Hiding
- Data and functions are separated in C while in C++ they are encapsulated together.

-How long have you been programming **?**

-Like 5 **years.**

-So you're good at C++?

Procedural vs Object Oriented Programmming

- Example of Procedural Programming Languages:
  - Cobol
  - Pascal
  - C

- Example of Object Oriented Programming Languages:
  - C++
  - Java
  - C#
  - Python

High Level vs Low Level Programming

Levels of Programming Languages

High-level program

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

Low-level program

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

Executable Machine code

```
0001001001000101
0010010011101100
10101101001...
```

High Level vs Low Level Programming

High Level Language	Low Level Language
It is programmer friendly language.	It is a machine friendly language.
High level language is less memory efficient.	Low level language is high memory efficient.
It is easy to understand.	It is tough to understand.
It is simple to debug.	It is complex to debug comparatively.
It is simple to maintain.	It is complex to maintain comparatively.
It is portable.	It is non-portable.
It can run on any platform.	It is machine-dependent.
It needs compiler or interpreter for translation.	It needs assembler for translation.
It is used widely for programming.	It is not commonly used now-a-days in programming.

Features of C++

- Object Oriented Programming
- General Purpose Programming Language, supports features such as Polymorphism, Inheritance, and Encapsulation
- Extensible
- Recursion
- Portable or Machine Independent (but Platform Dependent)
- Rich Library
- Fast Execution Speed
- Compiled Language

Installation of C++

- Integrated Development Environments available for C++:
  - Visual Studio Code
  - Code Blocks
  - Clion
  - Eclipse
  - CodeLite
- MinGW Compiler Download

Starting with C++

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

int main() {
    cout << "Hello World";
    return 0;
}
```

python error messages: C++ error messages:

#include <iostream>

C++ comes with libraries that provide us with many ways for performing input and output.

In C++ input and output are performed in the form of a sequence of bytes or more commonly known as streams.

- **Input Stream:** If the direction of flow of bytes is from the device(for example, Keyboard) to the main memory then this process is called input.
- **Output Stream:** If the direction of flow of bytes is opposite, i.e. from main memory to device(display screen) then this process is called output.

iostream: iostream stands for standard input-output stream. This header file contains definitions of objects like cin, cout, etc.

using namespace std

- A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc.) inside it.
- Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries.
- cout, cin, endl are all defined in iostream file. If we try to use cout, endl in our code without specifying the namespace it will throw an error, because these are defined in the std namespace in the iostream.h file.

Scope Resolution Operator ::

In C++, scope resolution operator is ::. It is used for following purposes.

1) To access a global variable when there is a local variable with same name.

```
#include<iostream>
using namespace std;
int x; // Global x
int main()
{
      int x = 10; // Local x
      cout << "Value of global x is " << ::x;
      cout << "\nValue of local x is " << x;
      return 0;
}
```

Scope Resolution Operator ::

2) To define a function outside a class.

```
#include<iostream>
using namespace std;
class A
{
    public:
    void fun();
};
void A::fun()      // Definition outside class using ::
{
    cout << "fun() called";
}
int main()
{   A a;
    a.fun();
    return 0;}
```

Scope Resolution Operator ::

3) For namespace.

```
// Use of scope resolution operator for namespace.
#include<iostream>

int main()
{
    std::cout << "Hello" << std::endl;
}
```

Directly using namespace std

```cpp
#include <iostream>
int main()
{
        std::cout << “Enter any number:”;
        int n1 = 0;
        std::cin >> n1;
        std::cout << “The entered number is ”
        << n1;
        return 0;
}
```

Notice that we used `std::cin` and `std::cout` instead of simply `cin` or `cout`.

The prefix `std::` indicates that the names `cout` and `cin` are defined inside the namespace named std.

Calling cout from namespace std

```
#include <iostream>
using std::cout;
int main()
{
    cout << "Enter any number:";
    int n1 = 0;
    std::cin >> n1;
    cout << "The entered number is " << n1;
    return 0;
}
```

Calling names from namespace std

Example

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

int main()
{
        cout << "Enter any number:" <<endl;
        int n1 = 0;
        cin >> n1;
        cout << "The entered number is " << n1;
        return 0;
}
```

using namespace std

Example

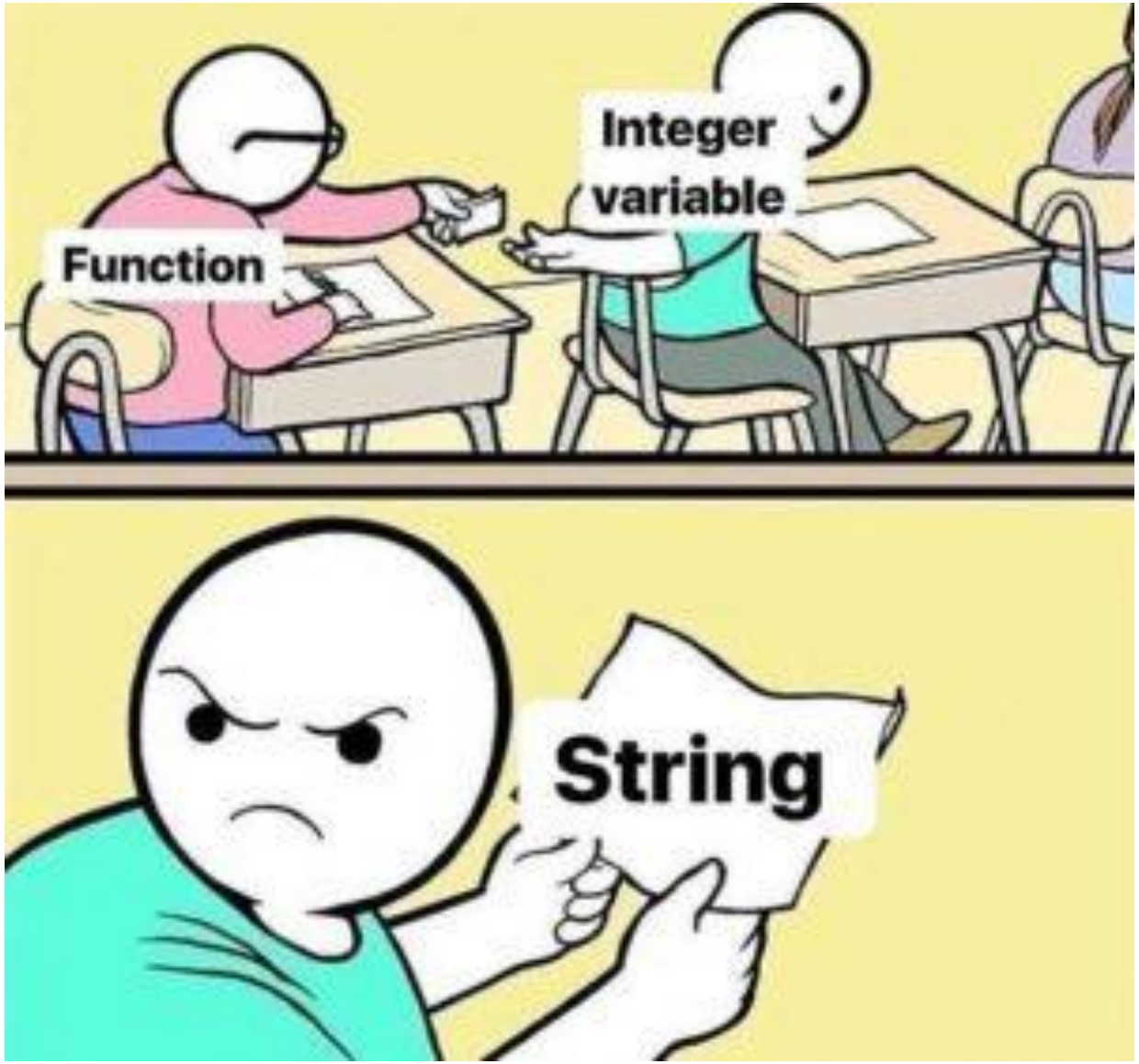
```
#include <iostream>
using namespace std;

int main()
{
        cout << "Enter any number:" <<endl;
        int n1 = 0;
        cin >> n1;
        cout << "The entered number is " << n1;
        return 0;
}
```

main()

- Every C/C++ program has at least one function that the name is main. The main function is called by the operating system by which our code is executed.
- We can make n number of function in a single program but we can make only one main function in a single program.
- When a C++ program is executed, the execution control goes directly to the main() function.

int main()

- A function may return some value.
- A return_type is the data type of the value the function returns.
- Some functions perform the desired operations and do not return any value. In this case, the return_type is specified by the keyword void.


Integer variable
Function
String

cout << "Hello World";

The "c" in cout refers to "character" and "out" means "output".

Hence cout means "character output".

The cout object is used along with the insertion operator << in order to display a stream of characters.

cin >> varName;

The "c" in cin refers to "character" and “in" means “input".

Hence cin means "character input".

The cin object is used along with the extraction operator >> in order to display a stream of characters.

endl

Endl in C++ is a manipulator or in simple terms a command. So when endl is encountered, the operating system will flush the output buffer and insert a new line.

cout << endl inserts a new line and flushes the stream(output buffer), whereas cout << “\n” just inserts a new line.

return 0

It is used to return a value from the function or stop the execution of the function.

- A return 0 means that the program will execute successfully and did what it was intended to do.
- A return 1 means that there is some error while executing the program, and it is not performing what it was intended to do.

This return value is called ‘Exit Value’.

This value is returned to the Operating System, which in turn determines if there was any error in the code or not.

Questions?

Variables in C++

Variables are containers for storing data values.

Syntax: *type variableName = value;*

where type is one of C++ types (such as int), and variableName is the name of the variable. The equal sign is used to assign values to the variable.

```
int myNum = 15;
cout << myNum;
```

```
int myNum;
myNum = 15;
cout << myNum;
```

Types of Variables in C++

In C++, there are different types of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as 123 or -123
- double - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- string - stores text, such as "Hello World". String values are surrounded by double quotes
- bool - stores values with two states: true or false

Types of Variables in C++

```
int myNum = 5;                  // Integer (whole number without decimals)
double myFloatNum = 5.99;       // Floating point number (with decimals)
char myLetter = 'D';            // Character
string myText = "Hello";        // String (text)
bool myBoolean = true;          // Boolean (true or false)
```

Data Types in C++

What is it?

Data Types in C++

All variables use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can store.

Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data type with which it is declared.

Every data type requires a different amount of memory.

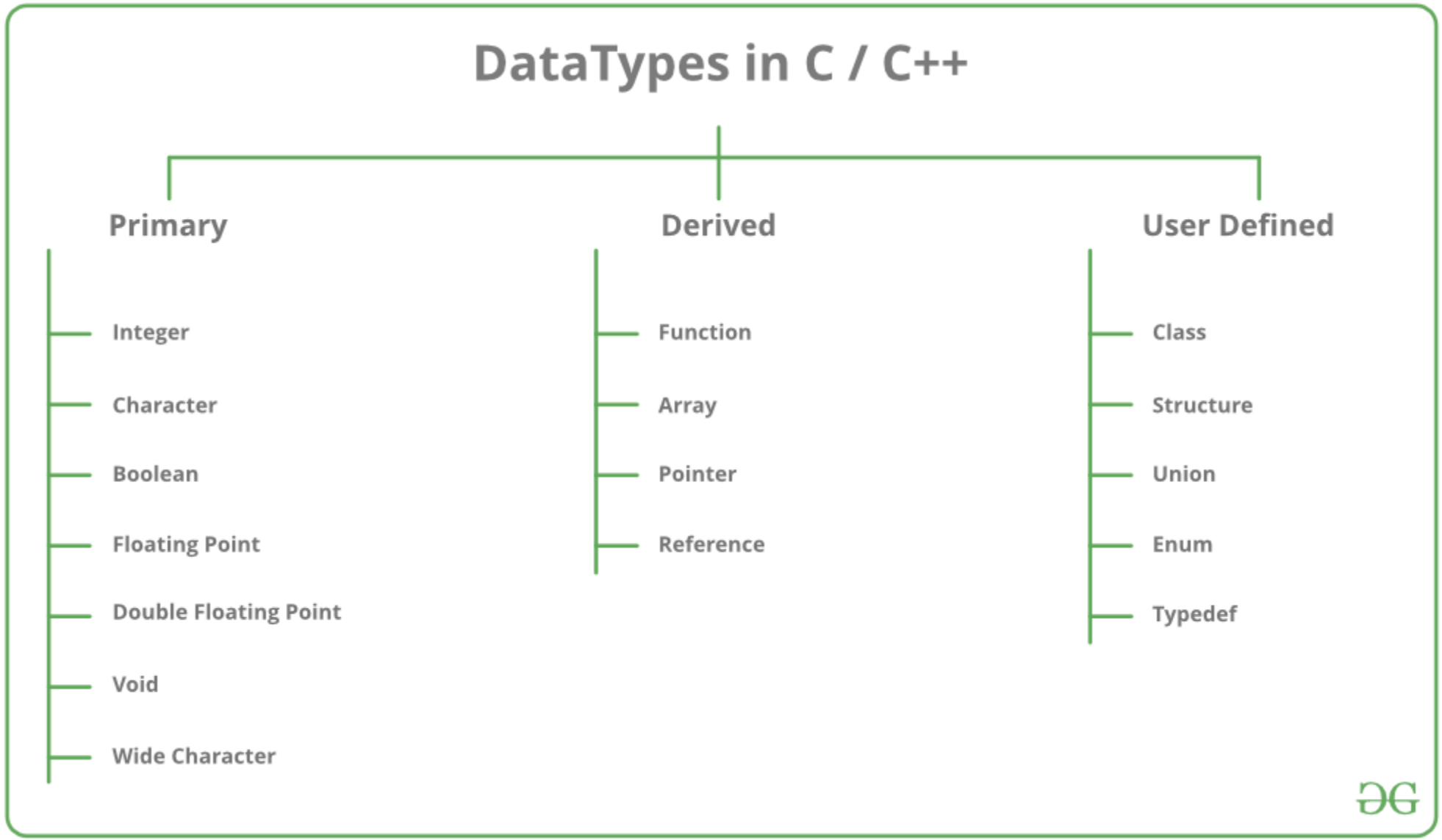
Data Types in C++

- All variables use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can store.
- Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data type with which it is declared.
- Every data type requires a different amount of memory.
- Data types specify the size and types of value to be stored.
- However, storage representation and machine instructions to manipulate each data type differ from machine to machine, although C++ instructions are identical on all machines.

Data Types in C++

C++ supports the following data types:

- Primary or Built in or Fundamental data type
- Derived data types
- User defined data types


DataTypes in C / C++
Primary
Integer
Character
Boolean
Floating Point
Double Floating Point
Void
Wide Character
Derived
Function
Array
Pointer
Reference
User Defined
Class
Structure
Union
Enum
Typedef

Data Types in C++

- Integer: The keyword used for integer data types is int. Integers typically require 4 bytes of memory space and range from -2147483648 to 2147483647.
- Character: Character data type is used for storing characters. The keyword used for the character data type is char. Characters typically require 1 byte of memory space and range from -128 to 127 or 0 to 255.
- Boolean: Boolean data type is used for storing Boolean or logical values. A Boolean variable can store either true or false. The keyword used for the Boolean data type is bool.
- Floating Point: Floating Point data type is used for storing single-precision floating-point values or decimal values. The keyword used for the floating-point data type is float. Float variables typically require 4 bytes of memory space.

Data Types in C++

- Double Floating Point: Double Floating Point data type is used for storing double-precision floating-point values or decimal values. The keyword used for the double floating-point data type is double. Double variables typically require 8 bytes of memory space.
- Void: Void means without any value. void data type represents a valueless entity. A void data type is used for those function which does not return a value.
- Wide Character: Wide character data type is also a character data type but this data type has a size greater than the normal 8-bit datatype. Represented by wchar_t. It is generally 2 or 4 bytes long.

Sizeof Function

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

int main()
{
    cout << "Size of char : " << sizeof(char) << endl;
    cout << "Size of int : " << sizeof(int) << endl;

    cout << "Size of long : " << sizeof(long) << endl;
    cout << "Size of float : " << sizeof(float) << endl;

    cout << "Size of double : " << sizeof(double) << endl;

    return 0;
}
```

Questions?

Operators in C++

- Arithmetic Operators
- Assignment Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Other Operators

Arithmetic Operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo Operation (Remainder after division)

`++` increases the value of the operand by **1**

`--` decreases it by **1**

Assignment Operators

`=`	`a = b;`	`a = b;`
`+=`	`a += b;`	`a = a + b;`
`-=`	`a -= b;`	`a = a - b;`
`*=`	`a *= b;`	`a = a * b;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`a %= b;`	`a = a % b;`

Relational Operators

`==`	Is Equal To	`3 == 5` gives us **false**
`!=`	Not Equal To	`3 != 5` gives us **true**
`>`	Greater Than	`3 > 5` gives us **false**
`<`	Less Than	`3 < 5` gives us **true**
`>=`	Greater Than or Equal To	`3 >= 5` give us **false**
`<=`	Less Than or Equal To	`3 <= 5` gives us **true**

Logical Operators

`&&`	expression1 **&&** expression2	Logical AND. True only if all the operands are true.
`||`	expression1 **||** expression2	Logical OR. True if at least one of the operands is true.
`!`	**!**expression	Logical NOT. True only if the operand is false.

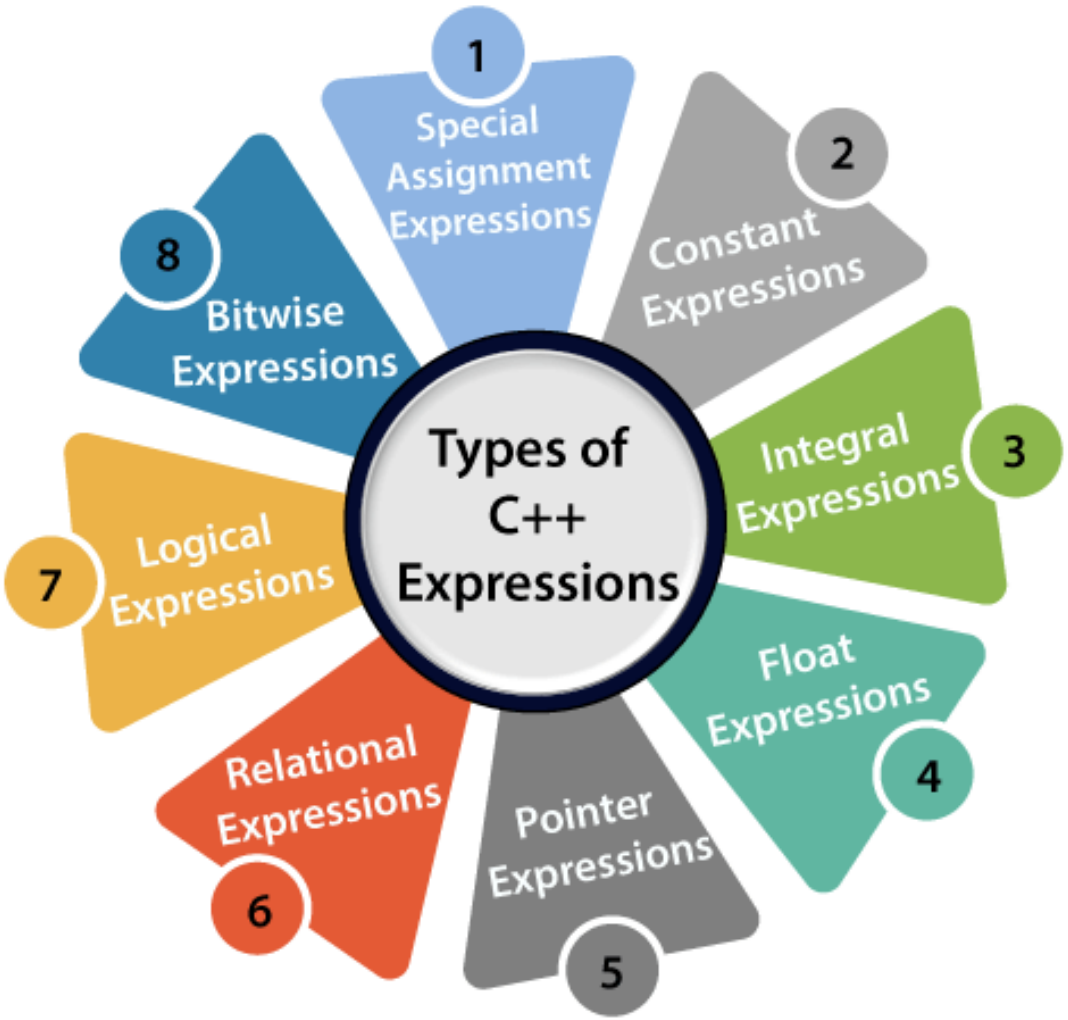
Bitwise Operators

&	Binary AND
|	Binary OR
^	Binary XOR
~	Binary One's Complement

Other Misc Operators

`sizeof`	returns the size of data type	`sizeof(int); // 4`
`?:`	returns value based on the condition	`string result = (5 > 0) ? "even" : "odd"; // "even"`
`&`	represents memory address of the operand	`&num; // address of num`
`.`	accesses members of struct variables or class objects	`s1.marks = 92;`
`->`	used with pointers to access the class or struct variables	`ptr->marks = 92;`
`<<`	prints the output value	`cout << 5;`
`>>`	gets the input value	`cin >> num;`

Expressions in C++

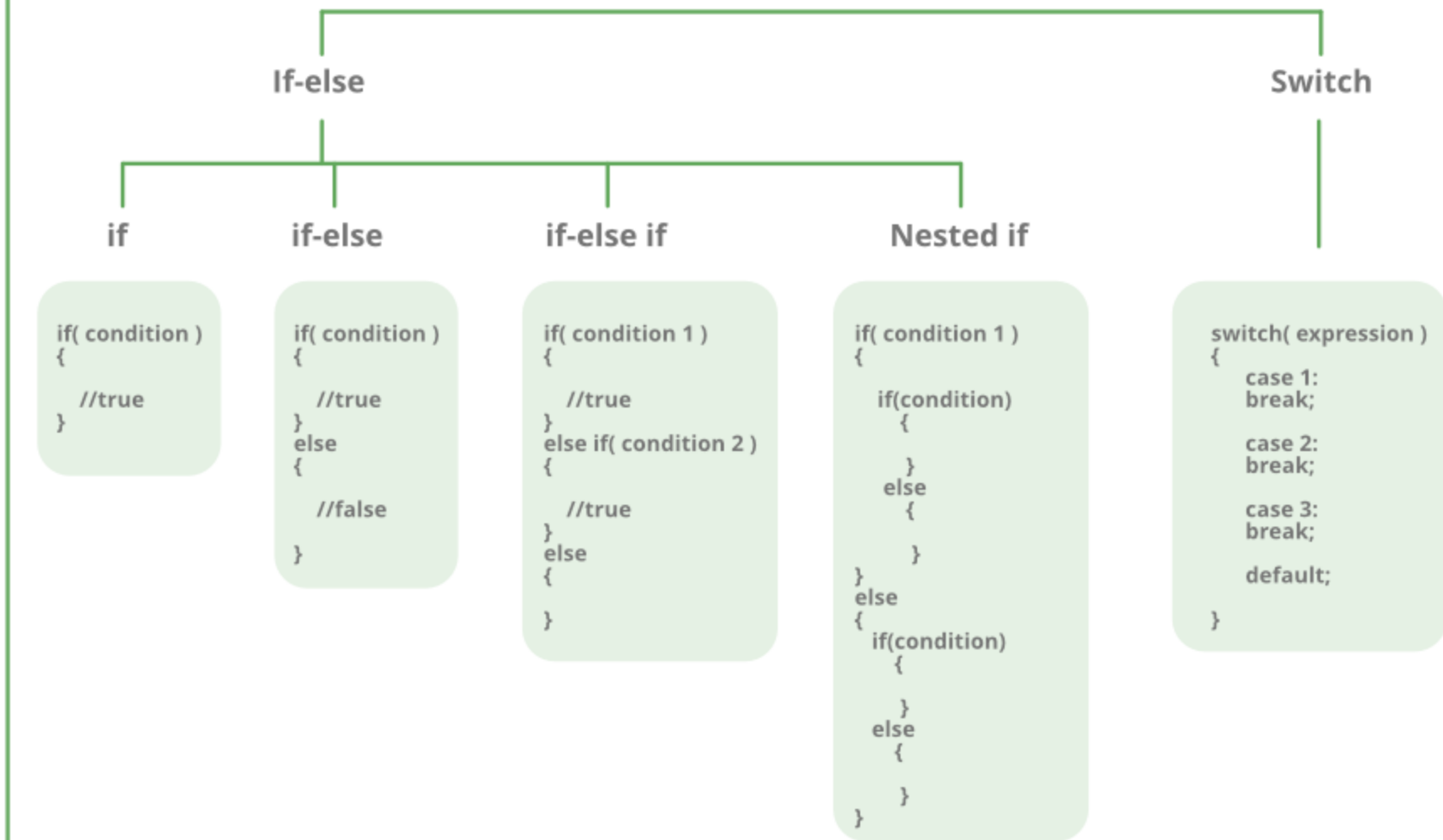


Decision Making Statements

There come situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code.

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Decision Making



If Statement

if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

```
if(condition)
{
    // Statements to execute if
    // condition is true
}
```

If-else Statement

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't.

But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with if statement to execute a block of code when the condition is false.

```
if (condition)
{
    // Executes this block if
    // condition is true
}
else
{
    // Executes this block if
    // condition is false
}
```

Nested-if Statement

A nested if in C is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement. Yes, both C and C++ allow us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

```
if (condition1)
{
   // Executes when condition1 is true
   if (condition2)
   {
      // Executes when condition2 is true
   }
}
```

if-else-if ladder Statement

Here, a user can decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions are true, then the final else statement will be executed.

```
if (condition)
    statement;
else if (condition)
    statement;
.
.
else
    statement;
```