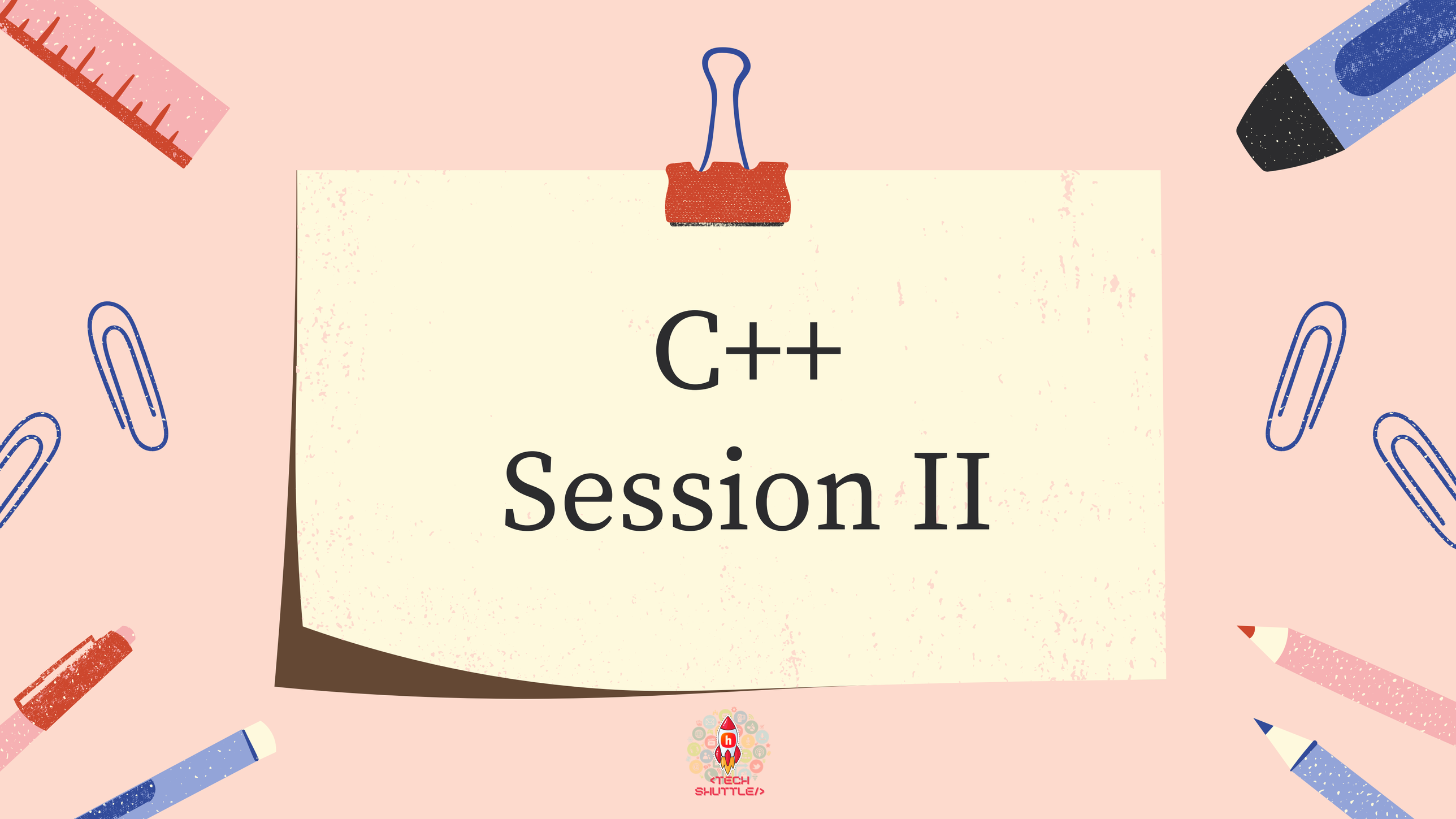
C++

Session II



To be covered:

- Data types & Variables
- Valid Range of Data Types
- Basic I/O
- Manipulators
- Data Operators

Data Types & variables

Variables

A variable is a name given to a memory location. It is the basic unit of storage in a program.

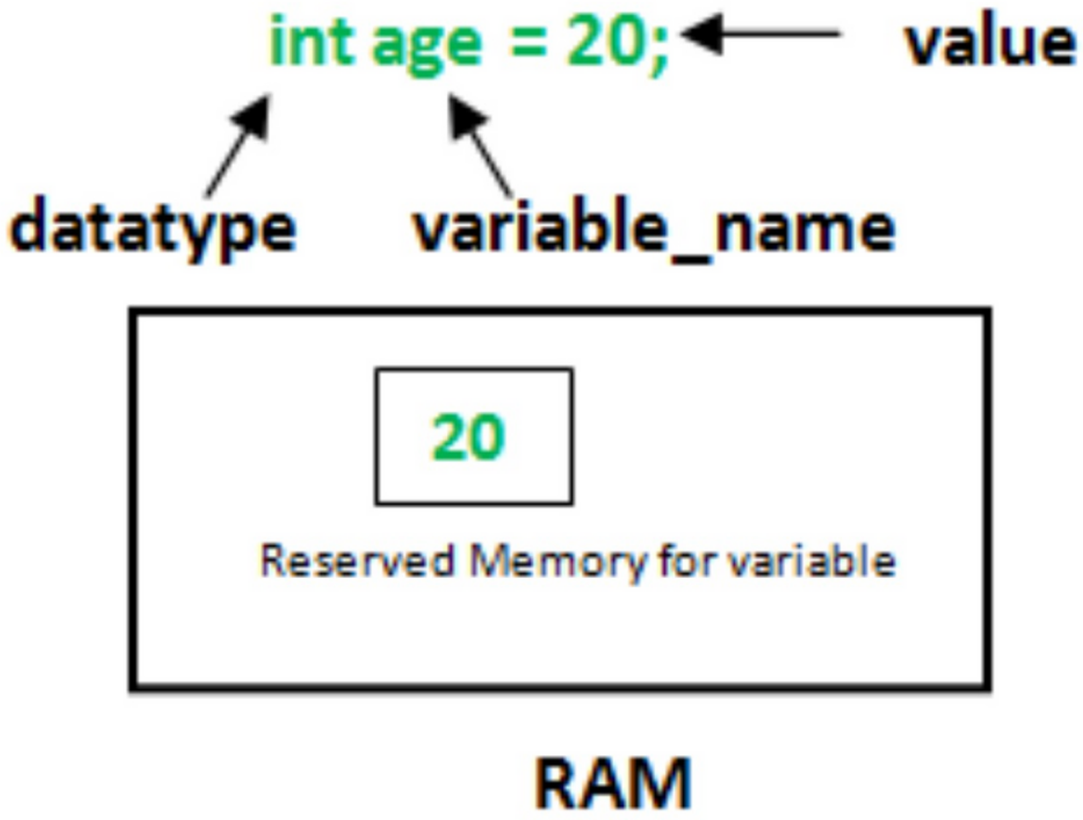
- In C++, all the variables must be declared before use.

A typical variable declaration is of the form:

```
// Declaring a single variable
type variable_name;

// Declaring multiple variables:
type variable1_name, variable2_name, variable3_name;
```

Variables in C++

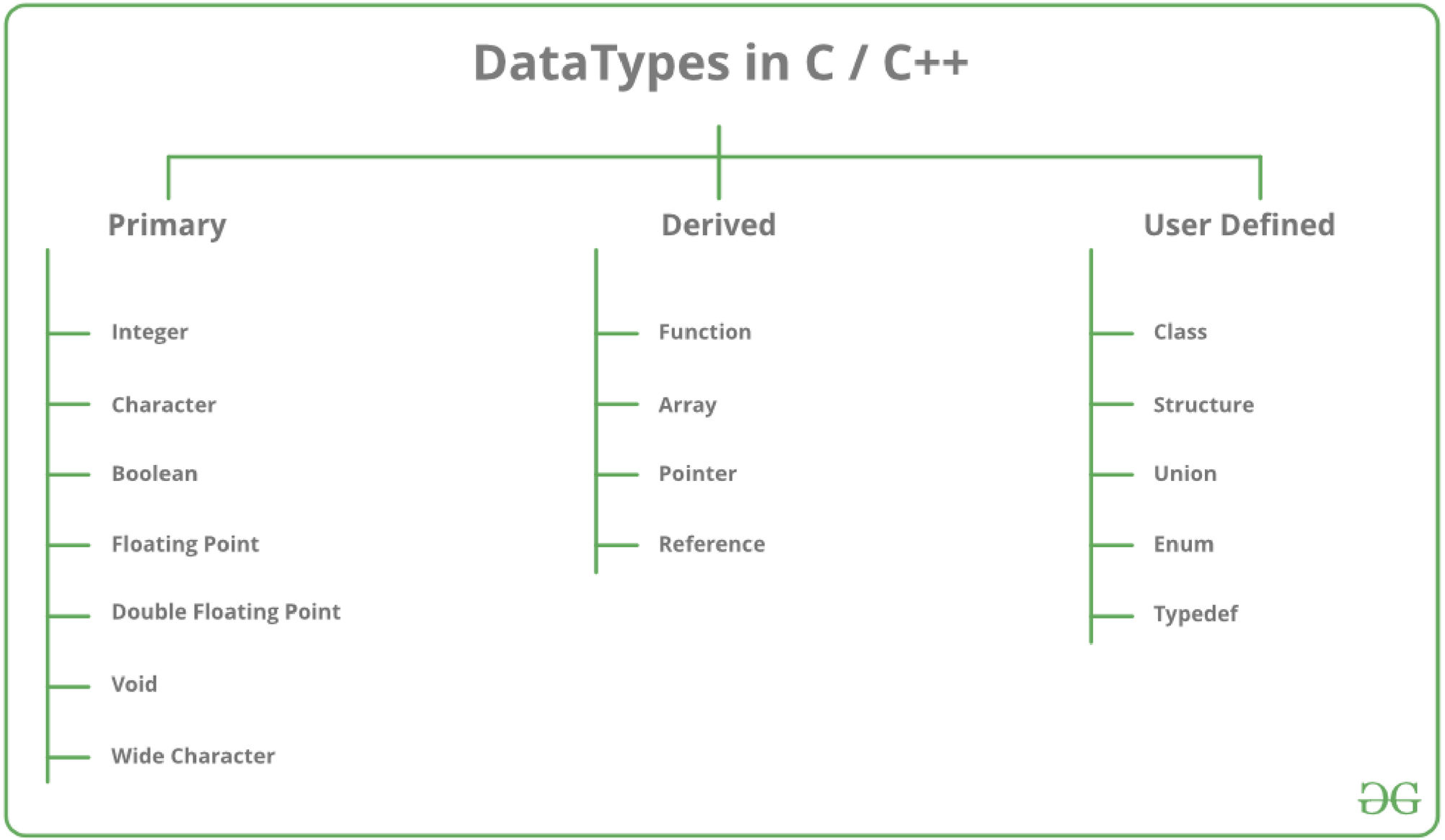


Data Types

Data types are used to tell the variables the type of data it can store. All variables use data-type during declaration to restrict the type of data to be stored.

Various datatypes are:

1. Primitive datatype
2. Derived Data Types
3. Abstract or 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

1. **Primitive Data Types:**

These data types are built-in or predefined data types and can be used directly by the user to declare variables.

- Integer
- Character
- Boolean
- Floating Point
- Double Floating Point
- Valueless or Void
- Wide Character

2. Derived Data Types:

The data-types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:

- Function
- Array
- Pointer
- Reference

3. Abstract or User-Defined Data Types:

These data types are defined by user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:

Class

Structure

Union

Enumeration

Typedef defined DataType

Integer: Keyword used for integer data types is int. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.

Character: Character data type is used for storing characters. Keyword used for character data type is char. Characters typically requires 1 byte of memory space and ranges from -128 to 127.

Boolean: Boolean data type is used for storing boolean or logical values. A boolean variable can store either true or false. Keyword used for boolean data type is bool.

Floating Point: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is float. Float variables typically requires 4 byte of memory space

Double Floating Point: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is double. Double variables typically requires 8 byte of memory space.

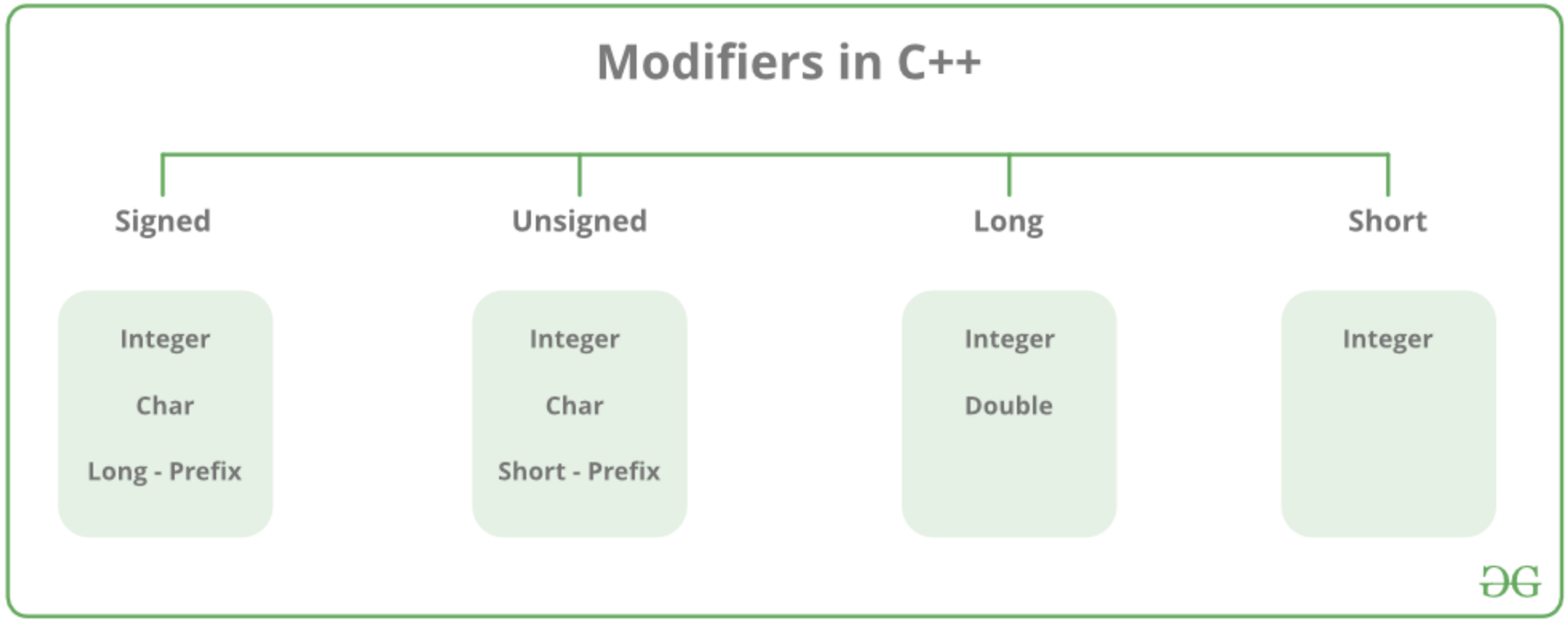
void: Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.

Wide Character: Wide character data type is also a character data type but this data type has size greater than the normal 8-bit datatype. Represented by wchar_t. It is generally 2 or 4 bytes long.

Datatype

Modifiers

As the name implies, datatype modifiers are used with the built-in data types to modify the length of data that a particular data type can hold.



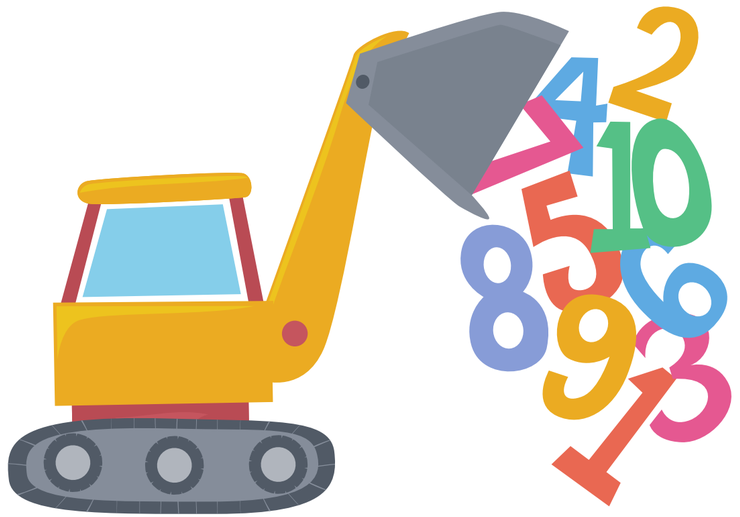
Data type modifiers available in C++ are:

Signed

Unsigned

Short

Long



Data Type	Size (in bytes)	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	8	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

Basic

I/O

I/O STREAM

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.

Header files available in C++ for Input/Output operations are:

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

- iomanip: iomanip stands for input output manipulators. The methods declared in this files are used for manipulating streams. This file contains definitions of setw, setprecision, etc.

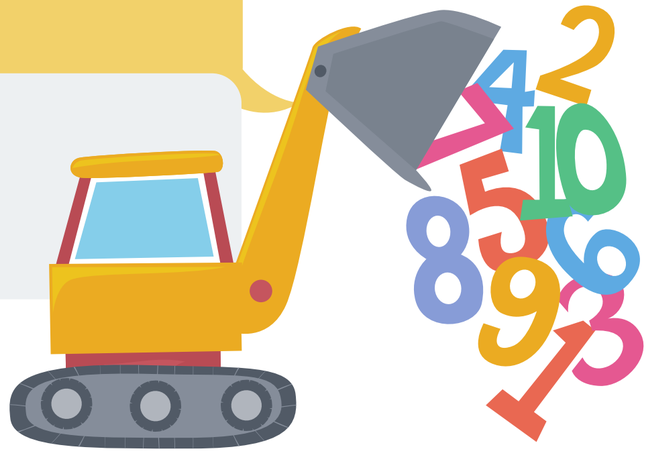
- fstream: This header file mainly describes the file stream. This header file is used to handle the data being read from a file as input or data being written into the file as output.

cout and cin

Standard output stream (cout): Usually the standard output device is the display screen. The C++ cout statement is the instance of the ostream class. It is used to produce output on the standard output device which is usually the display screen. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator(<<).

standard input stream (cin): Usually the input device in a computer is the keyboard. C++ cin statement is the instance of the class iostream and is used to read input from the standard input device which is usually a keyboard.
The extraction operator(>>) is used along with the object cin for reading inputs. The extraction operator extracts the data from the object cin which is entered using the keyboard.

Manipulators



Manipulators

Manipulators are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<<) and extraction (>>) operators.

- endl: It is defined in ostream. It is used to enter a new line and after entering a new line it flushes (i.e. it forces all the output written on the screen or in the file) the output stream.
- setw (val): It is used to set the field width in output operations.
- setprecision (val): It sets val as the new value for the precision of floating-point values.
- setbase(val): It is used to set the numeric base value for numeric values.

Data

Operators

Operators

Operators are the foundation of any programming language. Thus the functionality of C/C++ programming language is incomplete without the use of operators. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands.

```
c = a + b;
```

Here, ‘+’ is the operator known as addition operator and ‘a’ and ‘b’ are operands. The addition operator tells the compiler to add both of the operands ‘a’ and ‘b’.

Types of Operators

C/C++ has many built-in operator types and they are classified as follows:

1. Arithmetic Operators: These are the operators used to perform arithmetic/mathematical operations on operands. Examples: (+, -, *, /, %,++,––). Arithmetic operator are of two types:
    a. Unary Operators: Operators that operates or works with a single operand are unary operators. For example: (++ , ––)
    b. Binary Operators: Operators that operates or works with two operands are binary operators. For example: (+ , – , * , /)
2. Relational Operators: These are used for comparison of the values of two operands. For example, checking if one operand is equal to the other operand or not, an operand is greater than the other operand or not etc. Some of the relational operators are (==, >= , <=).

Types of Operators

C/C++ has many built-in operator types and they are classified as follows:

- Logical Operators: Logical Operators are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration. The result of the operation of a logical operator is a boolean value either true or false.

- Bitwise Operators: The Bitwise operators is used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. The mathematical operations such as addition, subtraction, multiplication etc. can be performed at bit-level for faster processing. For example, the bitwise AND represented as & operator in C or C++ takes two numbers as operands and does AND on every bit of two numbers.

Types of Operators

C/C++ has many built-in operator types and they are classified as follows:

Assignment Operators: Assignment operators are used to assign value to a variable. The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value. The value on the right side must be of the same data-type of variable on the left side otherwise the compiler will raise an error.

```
a = 10;
b = 20;
ch = 'y';
```

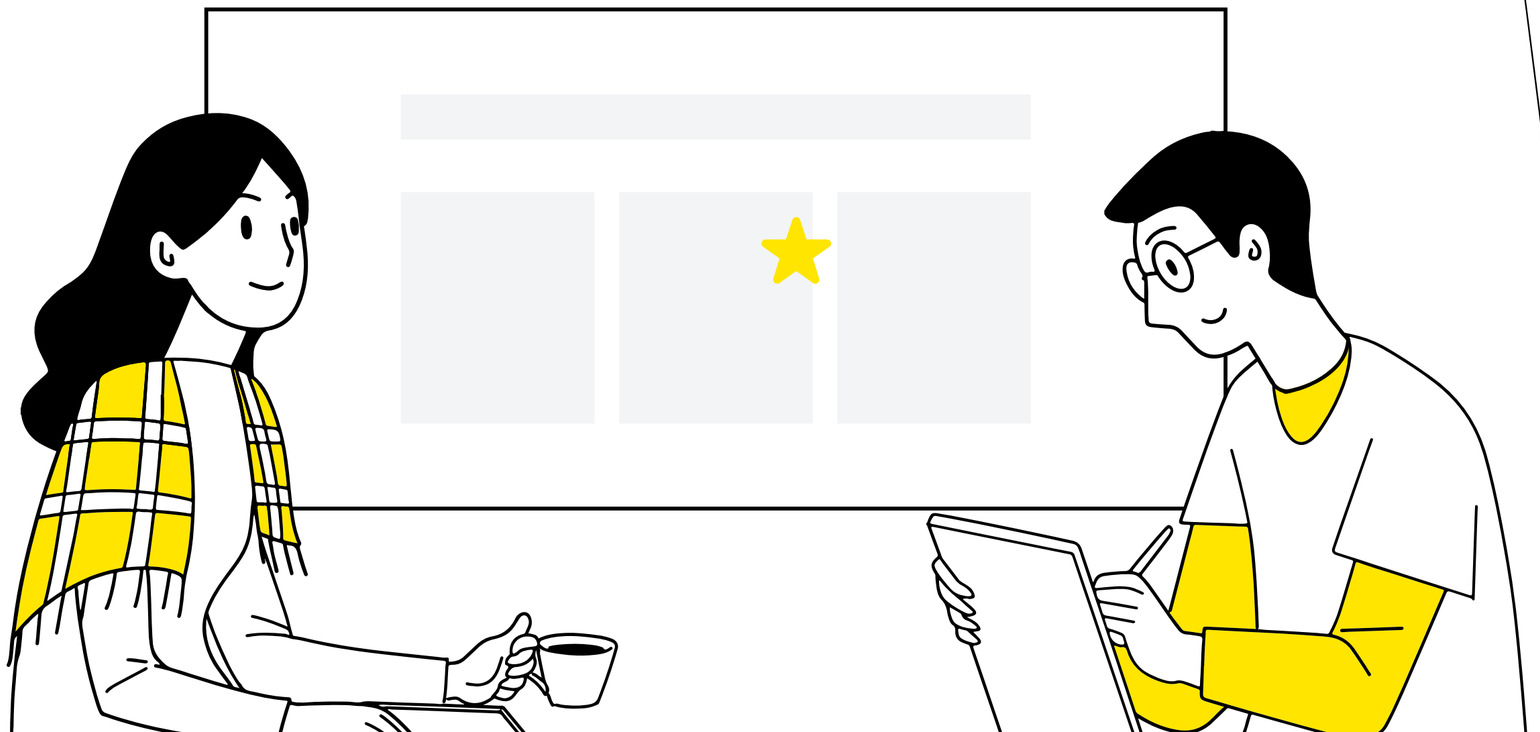
```
(a += b) can be written as (a = a + b)
```

QUESTION TIME

Let's make the most out of our learning sessions.

<TECH SHUTTLE/>

Quiz Time!



The purpose of the quiz is not to shame or embarrass anyone, but to make sure everyone is on the same page.


See you in the next session
Hope you had fun!