Programming Fundamentals with C++

Lecture 2 – Fundamentals - 1

Dr. Muhammad Sajjad

R.A Asad Ullah

Overview

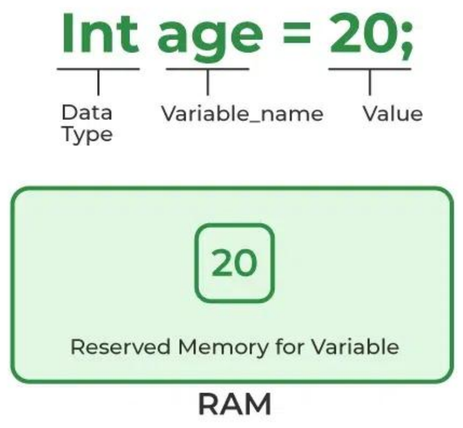
- **Variables**
  - Definition
  - Purpose
  - Rules for writing Variables Names
- **Data Types in C++**
  - Purpose of Data Types
  - Pre-Defined Data Types in C++
- **Declaration & Initialization of Variables**
  - Declaration
  - Initialization
  - Default Value

Variables

- **Definition**
  - Variables are like storage containers in a program where you can save data.
  - Think of a variable as a labeled box where you can store and retrieve information.
  - For example, if you want to store a person's age, you might create a variable called age.

- **Purpose**
  - Variables allow us to save values temporarily, making it possible to reuse data and perform operations throughout a program.

Variable
• Variable: imagine like a box to store one thing (data)
• Eg: int age;
age = 5;
int - Size of variable must be big enough to store Integer
age - Name of variable
age



Variables

- **Rules for writing Variables Names**
  - The first character of variable name must be an alphabetic character.
  - Underscore can be used as first character of variable name.
  - Blank spaces are not allowed in a variable name.
  - Special character, such as arithmetic operators, #, ^, etc. cannot be used in a variable name.
  - Reserved words (keywords) cannot be used as variable names.
  - The maximum length of a variable name depends upon the compiler of C++.
  - A variable name declared for one data type cannot be used to declare another data type.

- **Note**
  - C++ is a case-sensitive language. Thus variable names with same spellings but different cases are treated as different variables names.
  - For example, variables "Pay" and "pay" are two different variables.

Variables

Following are some examples of the valid and invalid variable names.

Variable Name	Valid / Invalid	Remarks
Nadeem	valid	
perform	valid	
double	invalid	C++ reserved word
foxpro	valid	
switch	invalid	C++ reserved word
Marrie	valid	
int	invalid	C++ reserved word
3rd	invalid	Start with a numeral
unsigned	invalid	C++ reserved word
x-y	invalid	Special character used
Taq Ahd	invalid	Spaces not allowed

Data Types in C++

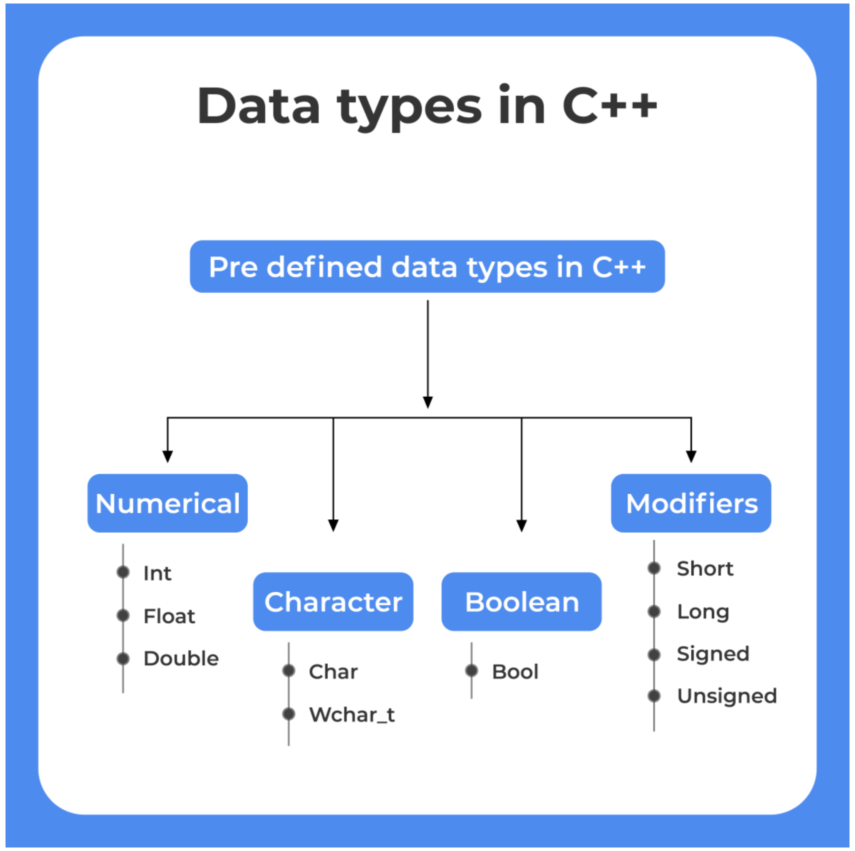
Purpose of Data Types

- Data types specify the kind of data a variable can hold.
- This is important because different types of data (like whole numbers, decimal numbers, characters) are handled differently by the computer.

Pre-Defined Data Types in C++

- int: Used for integer numbers (e.g., int age = 20;).
- float and double: Used for numbers with decimals. float has lower precision compared to double. Example: float price = 19.99;
- char: Stores a single character, like a letter or symbol. Example: char grade = 'A';
- bool: Stores a boolean value, which can be either true or false.

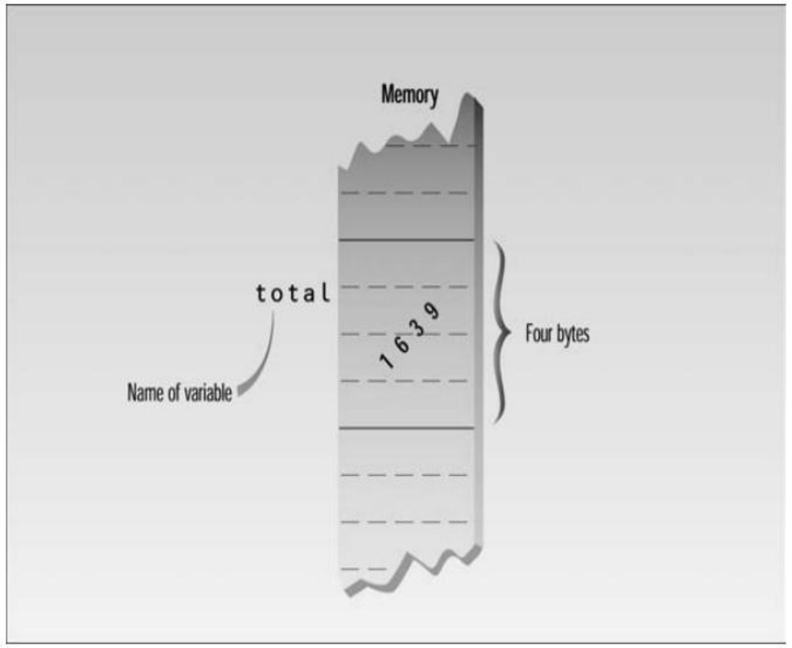
  Example: bool is_student = true;



Data Types in C++

The int Data Type

- **Definition:** int is short for "integer," representing whole numbers (both positive and negative) without any decimal point.
- **Usage**: It is used to store values like counts, ages, IDs, etc.
- **Memory:** The amount of memory occupied by the integer types is system dependent. On a 32-bit system such as Windows, an int occupies **4 bytes** (which is 32 bits) of memory, allowing a range of about -2 billion to 2 billion (-2,147,483,648 to 2,147,483,647).



- **Example**

int total = 1639;

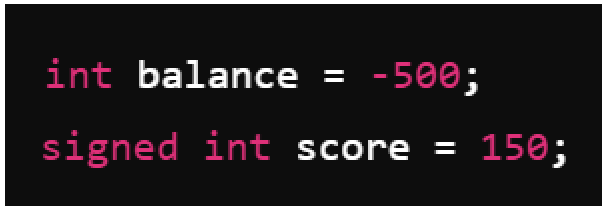
Data Types in C++

The int Data Type

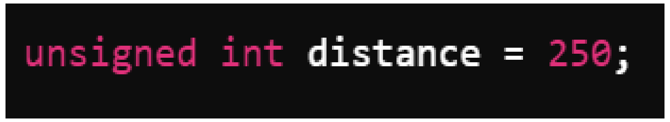
- **Qualifiers**

  C++ provides several qualifiers that can be used with int to modify its properties, mainly to control size and sign.

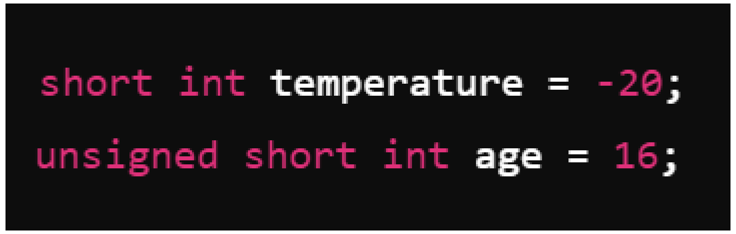
  - **signed int**
    - **Description:** The default int type is signed, meaning it can store both positive and negative values.
    - **Range:** Typically, from -2,147,483,648 to 2,147,483,647.
  - **unsigned int**
    - **Description:** An unsigned int can only store non-negative values, effectively doubling the maximum positive range.
    - **Range:** 0 to 4,294,967,295 (for 4 bytes).
  - **short int**
    - **Description:** A short int uses less memory, usually 2 bytes (16 bits), which means a smaller range.
    - **Range:** -32,768 to 32,767 for signed and 0 to 65,535 for unsigned.
  - **Long int**
    - **Description:** A long int allows for a larger range of values, typically using 8 bytes (64 bits) on most systems.
    - **Range:** -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 for signed and 0 to 18,446,744,073,709,551,615 for unsigned.



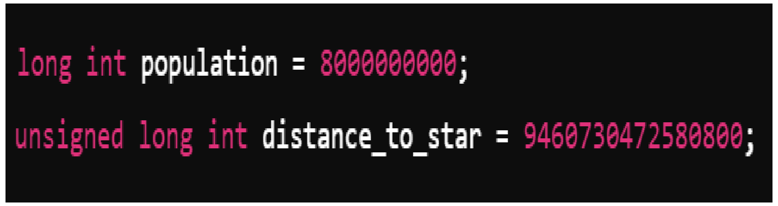
```
int balance = -500;
signed int score = 150;
```



```
unsigned int distance = 250;
```



```
short int temperature = -20;
unsigned short int age = 16;
```

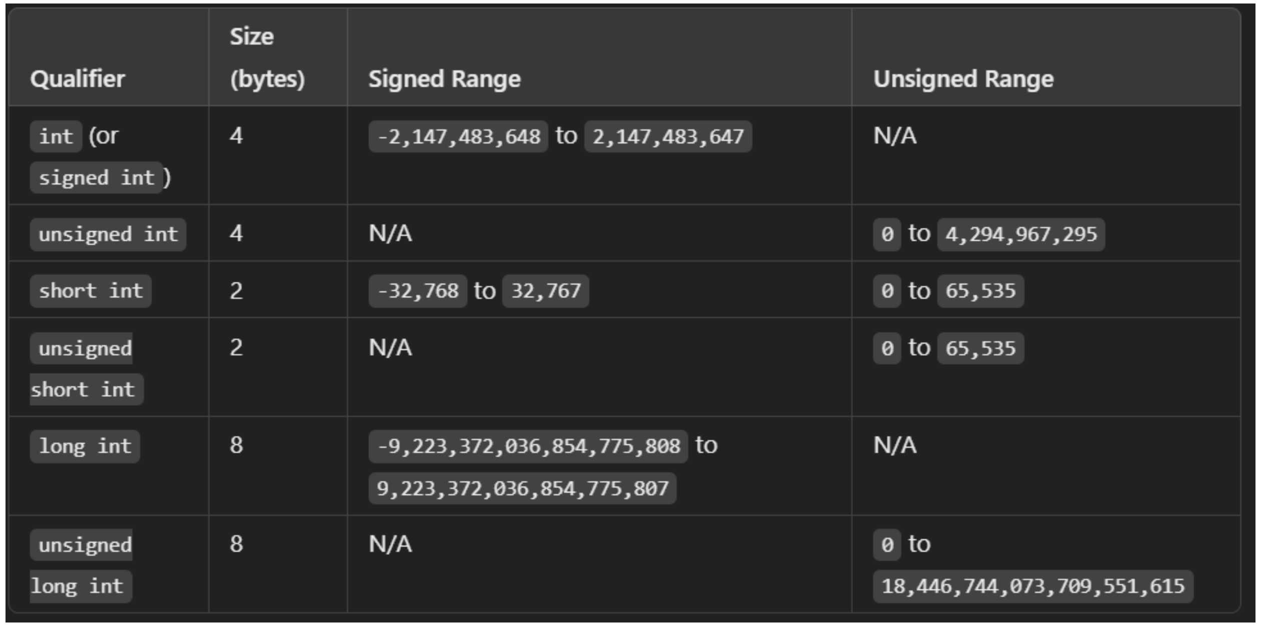


```
long int population = 8000000000;
unsigned long int distance_to_star = 9460730472580800;
```

Data Types in C++

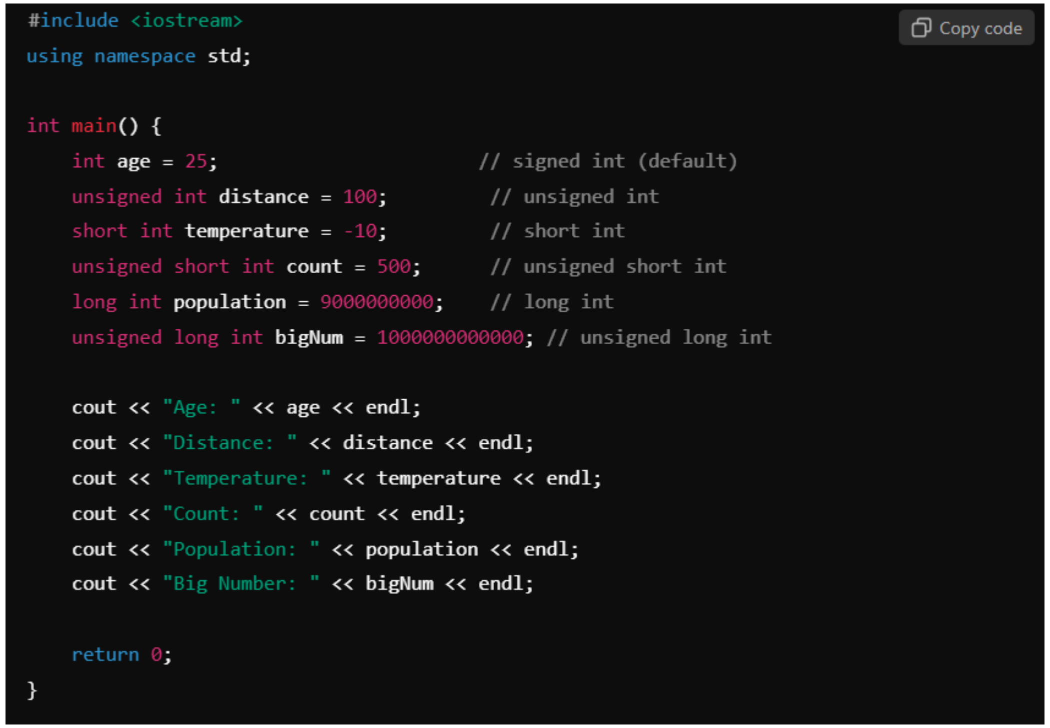
The int Data Type

Summary and Example Code



Qualifier	Size (bytes)	Signed Range	Unsigned Range
`int` (or `signed int`)	4	`-2,147,483,648` to `2,147,483,647`	N/A
`unsigned int`	4	N/A	`0` to `4,294,967,295`
`short int`	2	`-32,768` to `32,767`	`0` to `65,535`
`unsigned short int`	2	N/A	`0` to `65,535`
`long int`	8	`-9,223,372,036,854,775,808` to `9,223,372,036,854,775,807`	N/A
`unsigned long int`	8	N/A	`0` to `18,446,744,073,709,551,615`

Summary



Copy code

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

int main() {
    int age = 25;                            // signed int (default)
    unsigned int distance = 100;              // unsigned int
    short int temperature = -10;              // short int
    unsigned short int count = 500;           // unsigned short int
    long int population = 9000000000;         // long int
    unsigned long int bigNum = 1000000000000; // unsigned long int

    cout << "Age: " << age << endl;
    cout << "Distance: " << distance << endl;
    cout << "Temperature: " << temperature << endl;
    cout << "Count: " << count << endl;
    cout << "Population: " << population << endl;
    cout << "Big Number: " << bigNum << endl;

    return 0;
}
```

Code

int_data_type.cpp

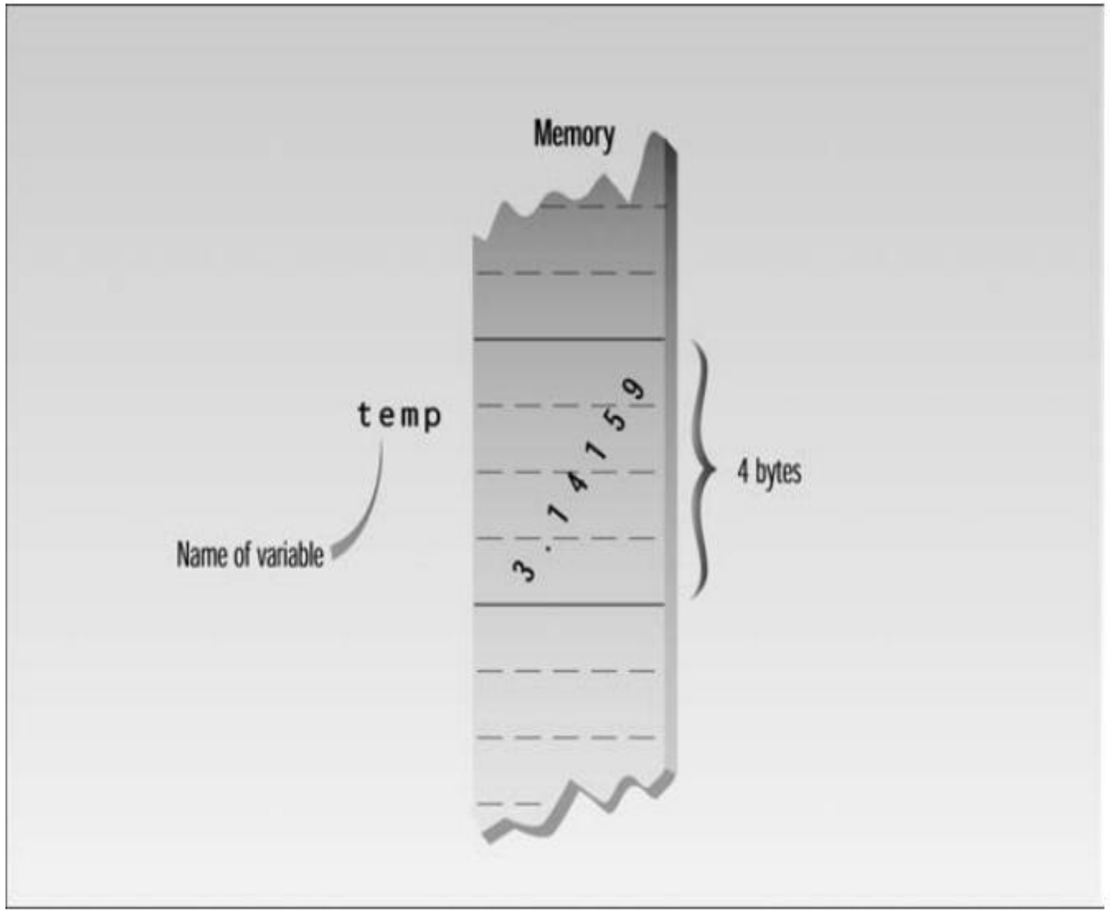
Data Types in C++

The float **Data Type**

- **Definition:** float is a data type used to store single-precision floating-point numbers, which are numbers with a decimal point.
- **Usage**: Ideal for storing real numbers in applications where high precision is not essential, such as graphical or scientific applications.
- **Memory:** Typically, a float uses 4 bytes (32 bits), providing a range of approximately ±3.4 x 10^38 and a precision of 6-7 decimal places.

- **Example**

float temp = 3.14159;



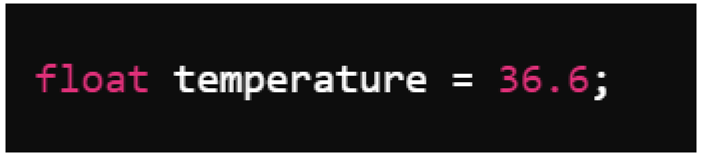
Data Types in C++

The float Data Type

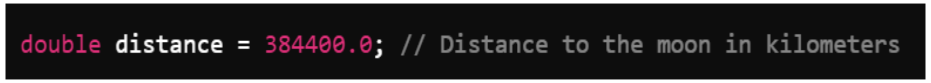
- **Qualifiers**

  In addition to float, C++ offers other floating-point types to handle different levels of precision and range.

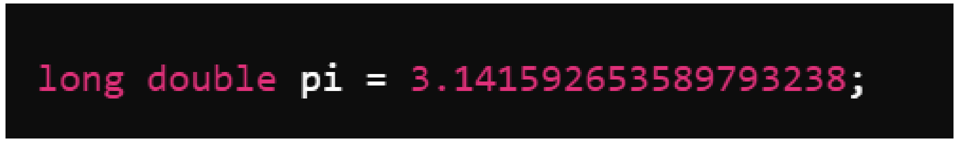
  - **float**
    - **Description**: Single-precision floating-point, typically 4 bytes.
    - **Precision**: ~6-7 decimal digits.
  - **double**
    - **Description**: Double-precision floating-point, generally using 8 bytes.
    - **Precision:** ~15-16 decimal digits.
  - **long double**
    - **Description**: Extended-precision floating-point, size can vary (often 10-16 bytes depending on the compiler).
    - **Precision:** More precise than double, often up to 18-19 decimal digits.



```cpp
float temperature = 36.6;
```



```cpp
double distance = 384400.0; // Distance to the moon in kilometers
```

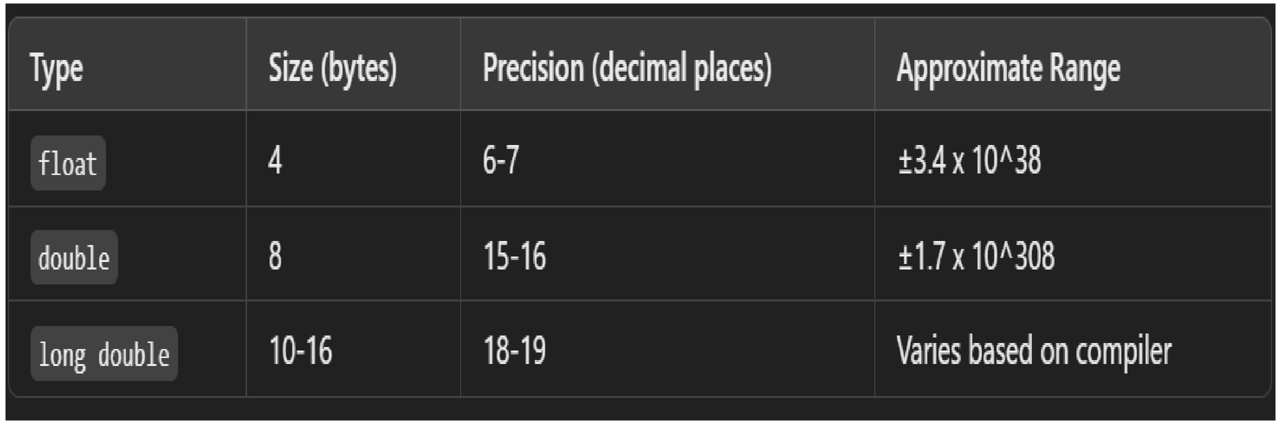


```cpp
long double pi = 3.141592653589793238;
```

Data Types in C++

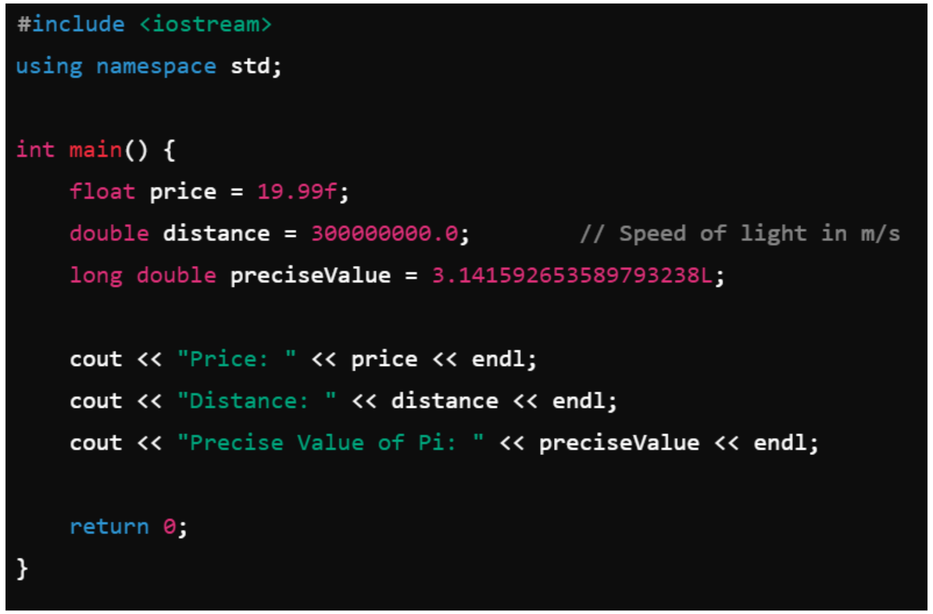
The float Data Type

Summary and Example Code



Type	Size (bytes)	Precision (decimal places)	Approximate Range
`float`	4	6-7	±3.4 x 10^38
`double`	8	15-16	±1.7 x 10^308
`long double`	10-16	18-19	Varies based on compiler

Summary



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

int main() {
    float price = 19.99f;
    double distance = 300000000.0;        // Speed of light in m/s
    long double preciseValue = 3.141592653589793238L;

    cout << "Price: " << price << endl;
    cout << "Distance: " << distance << endl;
    cout << "Precise Value of Pi: " << preciseValue << endl;

    return 0;
}
```

Code

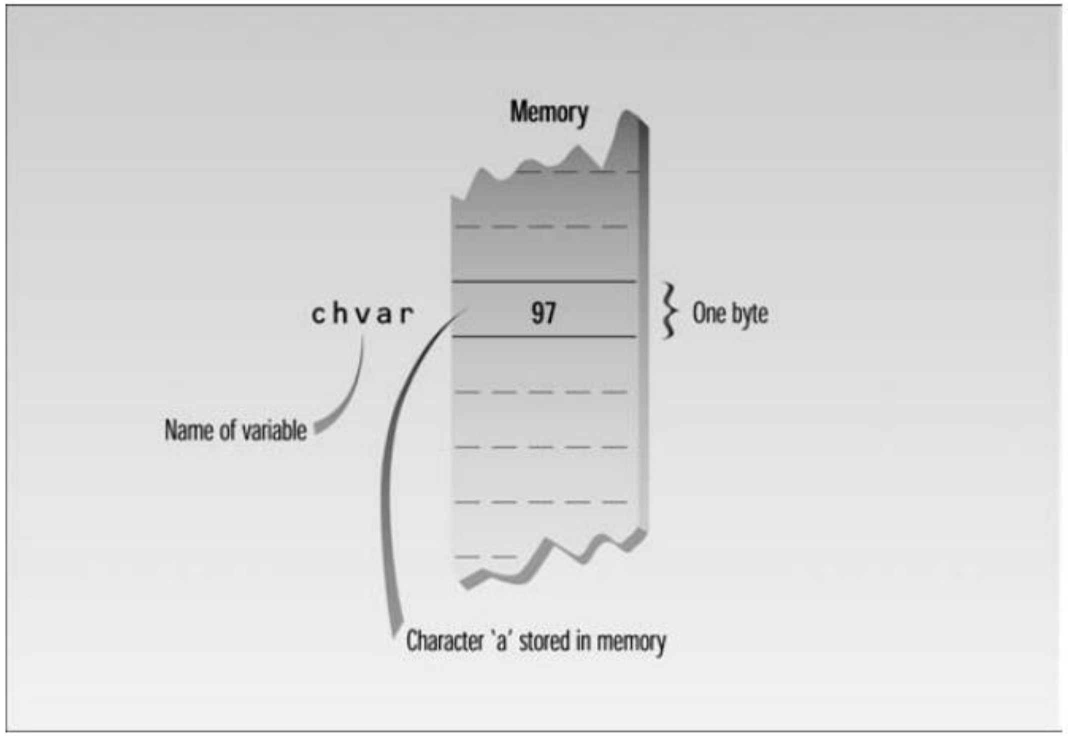
float_and_double.cpp

Note: The suffix f is used to specify float, and L for long double to avoid any ambiguity with the compiler.

Data Types in C++

The char Data Type

- **Definition:** char is a data type used to store individual characters (letters, numbers, symbols).
- **Usage**: Commonly used to store single characters like 'A', 'b', '3', or symbols like '$'.
- **Memory**: Typically uses **1 byte** (8 bits), enough to store 256 different values (for ASCII characters).
- **Range:** char can store values from -128 to 127 if it's signed, and 0 to 255 if it's unsigned.
- **ASCII Representation:** Each char variable holds an integer value corresponding to the ASCII code of the character.



- **Example**

char a = 97; // 97 is ASCII for 'a'

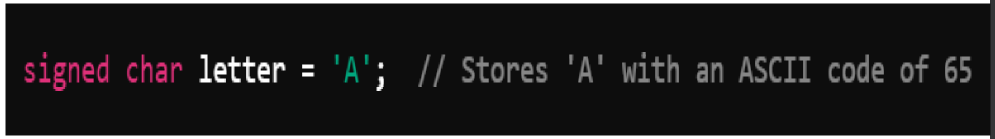
Or char a = 'a';

Note: Strictly speaking type char is an integer type as well

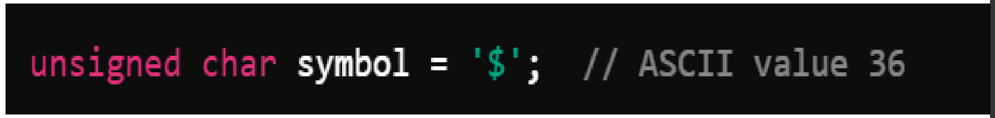
Data Types in C++

The char Data Type

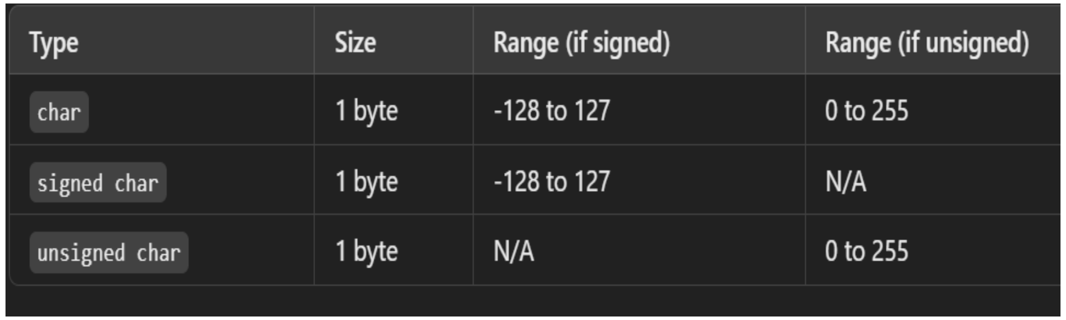
- **Qualifiers**
  - **Signed char:** Can hold values from -128 to 127, allowing negative values, which are sometimes used to represent extended ASCII characters.
  - **Unsigned char:** Ranges from 0 to 255, which is useful if you only need positive values (standard ASCII values).



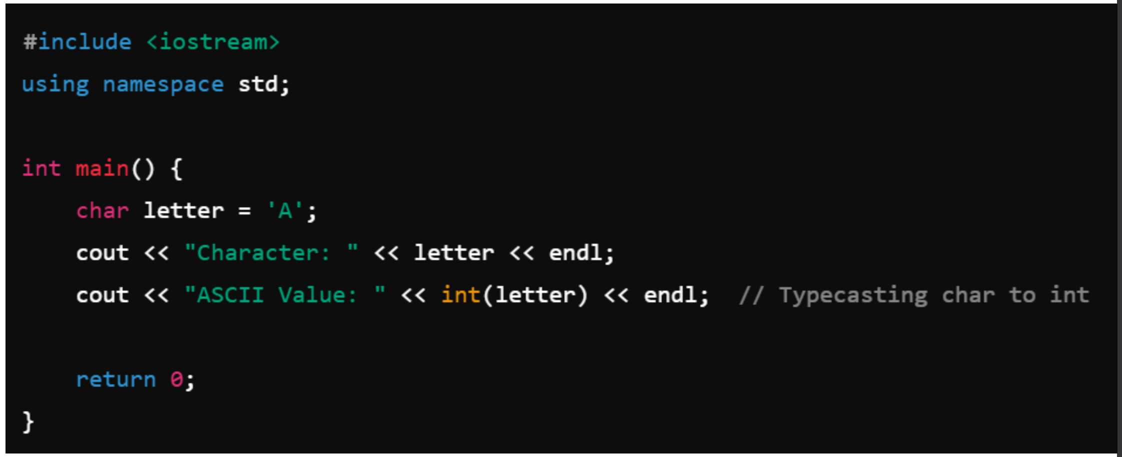
```cpp
signed char letter = 'A';  // Stores 'A' with an ASCII code of 65
```



```cpp
unsigned char symbol = '$';  // ASCII value 36
```



Type	Size	Range (if signed)	Range (if unsigned)
`char`	1 byte	-128 to 127	0 to 255
`signed char`	1 byte	-128 to 127	N/A
`unsigned char`	1 byte	N/A	0 to 255



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

int main() {
    char letter = 'A';
    cout << "Character: " << letter << endl;
    cout << "ASCII Value: " << int(letter) << endl;  // Typecasting char to int

    return 0;
}
```

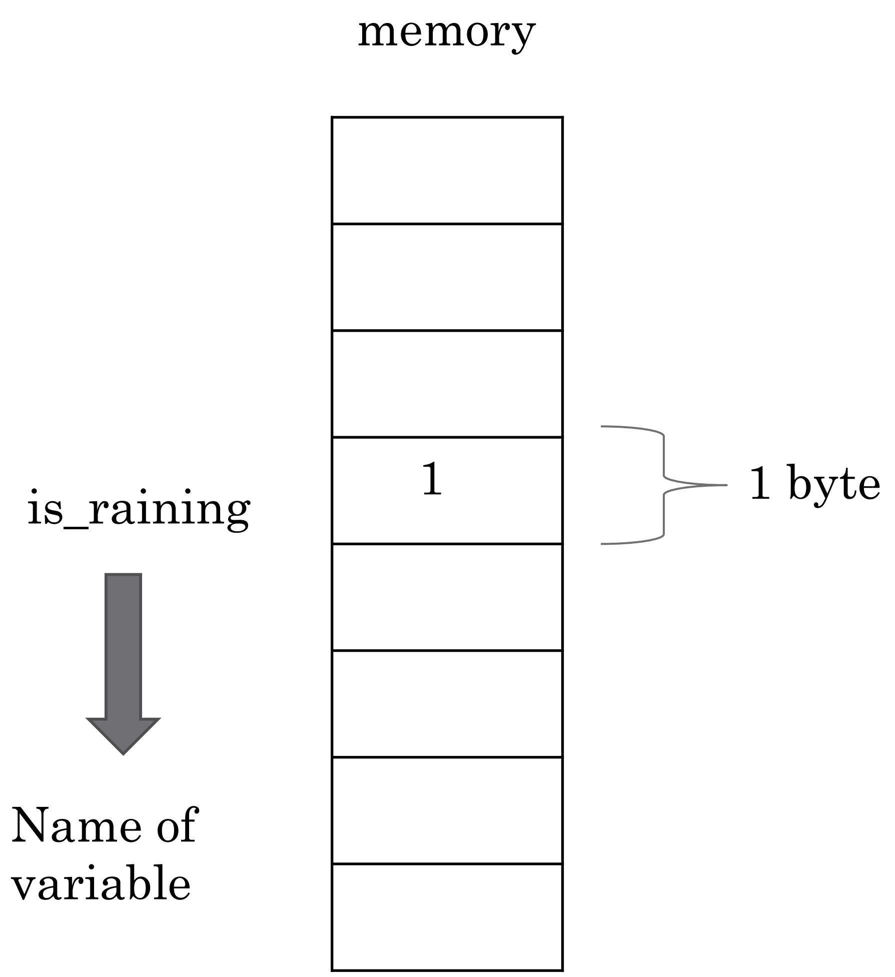
Output:
Character : **A**
ASCII Value: **65**

char_data_type.cpp

Data Types in C++

The bool **Data Type**

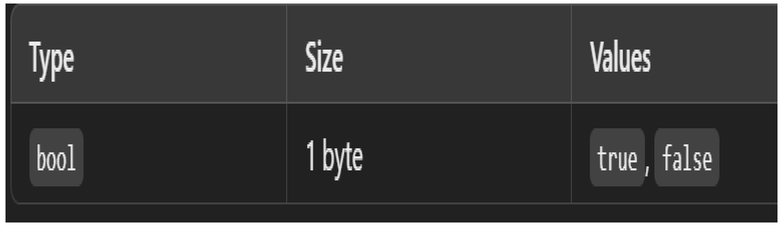
- **Definition:** bool is a data type used to represent Boolean values: true and false.
- **Purpose:** Primarily used in conditional statements and logic operations to control the flow of the program.
- **Memory:** Generally uses **1 byte*** of memory, although the exact size can vary depending on the compiler.
- **Boolean Values:** In C++, bool variables can have only two values i.e. true(1) or false (0).
- **Example**
  - **bool is_raining = true;**



Data Types in C++

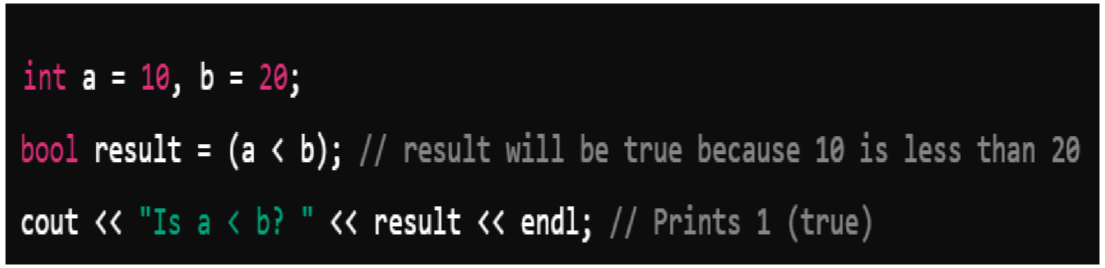
The bool Data Type

Summary and Example Code



Type	Size	Values
bool	1 byte	true, false

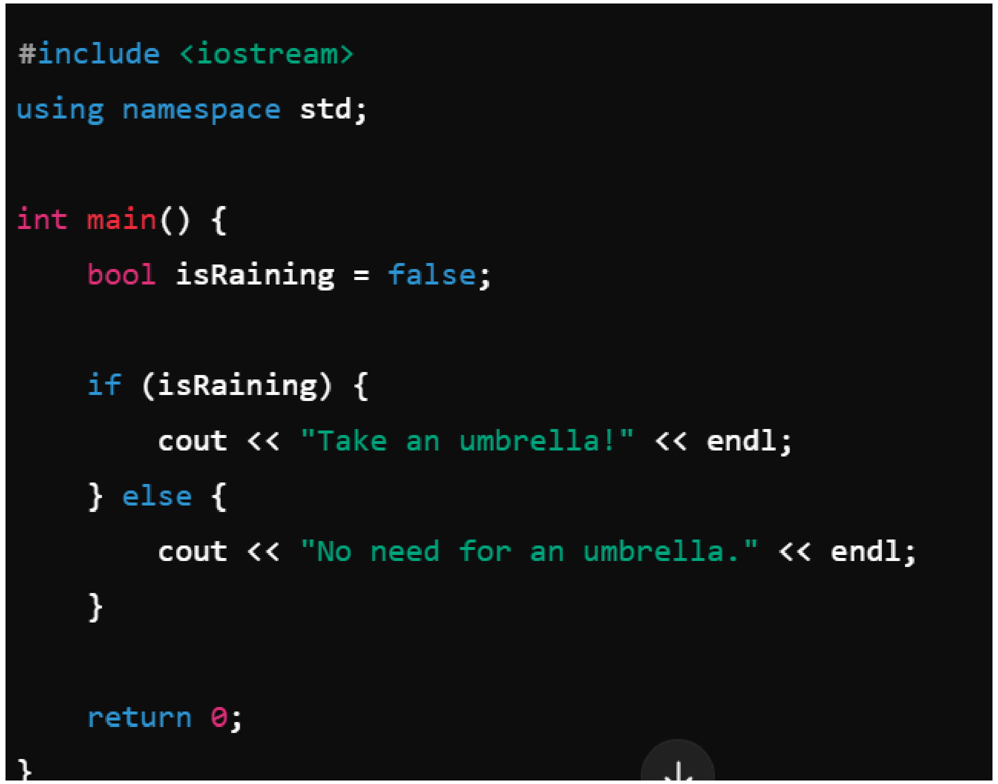
Summary



```
int a = 10, b = 20;
bool result = (a < b); // result will be true because 10 is less than 20
cout << "Is a < b? " << result << endl; // Prints 1 (true)
```

Using bool in comparison

bool2_data_type.cpp



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

int main() {
    bool isRaining = false;

    if (isRaining) {
        cout << "Take an umbrella!" << endl;
    } else {
        cout << "No need for an umbrella." << endl;
    }

    return 0;
}
```

Using bool in condition

bool_data_type.cpp

Declaration & Initialization of Variables

Declaration

- Declaring a variable means informing the compiler about the variable's name and type, reserving space in memory for it.
- Example: int age; — Here, int specifies the data type (integer), and age is the variable name.
- When a variable is declared, its memory is allocated, but it doesn't necessarily contain a meaningful value until it is initialized.

Initialization

- Initialization occurs when you assign a value to a variable at the time of its declaration.
- Example: int age = 20; — This both declares age as an int and initializes it with the value 20.
- Initializing variables ensures they have a known starting value, avoiding potential bugs caused by "garbage values" (random values in memory).

Default Value :
```
int a;
cout<<a;
output: Some random value
```

Declaration Examples:
```
int age;
float salary, bonus;
Char grade;
bool isOpen;
```

Initializing:
```
age = 18;
salary = 2000000;
bonus = 10000;
grade = 'A';
isOpen = false;
```

Declaring & Initializing at same time:
```
int age = 22;
char grade = 'B'
```

Thank You