# <u>Datatypes and Qualifiers</u>

In C++, **fundamental data types** (also called **built-in types**) are the basic types provided by the language to store and manipulate data. Along with them, **type qualifiers** provide additional control over how these data types behave.

---

## 🔹 Fundamental Data Types in C++

### 1. Integer Types

Used to store whole numbers.

| Type | Size (Typically) | Description |
|---|---|---|
| `int` | 4 bytes | Standard integer |
| `short` | 2 bytes | Shorter integer |
| `long` | 4 or 8 bytes | Larger integer |
| `long long` | 8 bytes | Even larger integer |
| `unsigned` versions | Same as signed | Only non-negative values |

### 2. Floating-Point Types

Used to store numbers with fractional parts.

| Type | Size | Precision |
|---|---|---|
| `float` | 4 bytes | ~6 digits |
| `double` | 8 bytes | ~15 digits |
| `long double` | 8-16 bytes | Highest precision (platform-dependent) |

### 3. Character Type

Used to store individual characters.

| Type | Size | Description |
|---|---|---|

|  |  |  |
|---|---|---|
| `char` | 1 byte | Single character |

### 4. Boolean Type

| Type | Description |
|---|---|
| `bool` | `true` or `false` |

---

## 🔹 Type Qualifiers in C++

Type qualifiers change the behavior of variables.

### 1. `const`

- Makes a variable **read-only**.

```
const int x = 10;
x = 20; // ❌ Error: cannot modify a const variable
```

### 2. `volatile`

- Tells the compiler that a variable **can be changed unexpectedly**, often used with hardware or multithreading.

```
volatile int status;
```

### 3. `static`

- For global or local scope:
    - Inside a function: retains value between calls.
    - Outside a class: limits scope to file.

```
void counter() {
    static int count = 0;
    count++;
    std::cout << count << "\n";
}
```

### 4. `extern`

- Declares a variable defined in another file or scope.

```
extern int sharedValue;
```

### 5. `mutable`

- Allows modification of a class member even if the object is `const`.

```
class Example {
    mutable int counter;
public:
    void update() const {
        counter++; // OK due to mutable
    }
};
```

---

## 🔹 Examples

### ✅ Using All Basic Types

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

int main() {
    int age = 30;
    float height = 5.9f;
    double pi = 3.14159;
    char grade = 'A';
    bool isPassed = true;
    unsigned int distance = 250;

    cout << "Age: " << age << endl;
    cout << "Height: " << height << endl;
    cout << "Pi: " << pi << endl;
    cout << "Grade: " << grade << endl;
    cout << "Passed: " << isPassed << endl;
    cout << "Distance: " << distance << endl;
```

```
  return 0;
}
```

---

## 🔹 Summary Table

| Qualifier | Purpose |
|---|---|
| `volatile` | Tells compiler not to optimize access |
| `const` | Makes variable read-only |
| `static` | Keeps value between calls or limits scope |
| `extern` | Links to variable declared elsewhere |
| `mutable` | Allows modification in const objects |

---