CS12 CH:02

C++ Introduction & Integer Datatypes

Mr. Gullo

September 10, 2025

Learning Objectives

- Understand the purpose of comments in code.

Learning Objectives

- Understand the purpose of comments in code.
- Learn the syntax for single-line (`//`) and multi-line (`/* ... */`) comments in C++.

Learning Objectives

- Understand the purpose of comments in code.
- Learn the syntax for single-line (`//`) and multi-line (`/* ... */`) comments in C++.
- Use comments to document code and temporarily disable code for testing.

What Are Comments?

For Human Eyes Only

Comments are text in your code that is completely ignored by the compiler. They are meant for you and other programmers to read and understand the code better.

What Are Comments?

For Human Eyes Only

Comments are text in your code that is completely ignored by the compiler. They are meant for you and other programmers to read and understand the code better.

Single-Line Comments

Start with //. The compiler ignores everything from // to the end of the line.

What Are Comments?

For Human Eyes Only

Comments are text in your code that is completely ignored by the compiler. They are meant for you and other programmers to read and understand the code better.

Single-Line Comments

Start with //. The compiler ignores everything from // to the end of the line.

Multi-Line Comments

Start with /* and end with */. Can span multiple lines. Useful for longer explanations or "commenting out" large blocks of code.

Code Example: Comments

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

// Use a double forward slash to create a single line comment
/*
    Use a forward slash and a star to make a multiline comment
    these are useful for large pieces of text or to remove
    ↪ sections
    from your code without deleting it.
*/
int main()
{
  // cout << "Hello World" << endl; // This line is disabled by a
  ↪ comment
    cout << "goodbye comments" << endl;
    return 0;
}
```

Learning Objectives

- Identify the fundamental integer-based datatypes in C++.

Learning Objectives

- Identify the fundamental integer-based datatypes in C++.
- Differentiate between `int`, `char`, and `bool`.

Learning Objectives

- Identify the fundamental integer-based datatypes in C++.
- Differentiate between `int`, `char`, and `bool`.
- Understand the kind of data each type is designed to store.

Integer Datatypes (The Basics)

Today we will look at three fundamental types that store whole numbers or concepts based on them.

Integer Datatypes (The Basics)

Today we will look at three fundamental types that store whole numbers or concepts based on them.

int

Short for 'integer'.
Stores positive and negative whole numbers.
Examples: 23, 19, -3,

0

Integer Datatypes (The Basics)

Today we will look at three fundamental types that store whole numbers or concepts based on them.

int

Short for 'integer'. Stores positive and negative whole numbers. **Examples:** 23, 19, -3, 0

char

Short for 'character'. Stores a *single* character. **Examples:**

`'a'`, `'Z'`, `'\n'`

Integer Datatypes (The Basics)

Today we will look at three fundamental types that store whole numbers or concepts based on them.

int

Short for 'integer'. Stores positive and negative whole numbers. **Examples:** `23`, `19`, `-3`, `0`

char

Short for 'character'. Stores a *single* character. **Examples:** `'a'`, `'Z'`, `'\n'`

bool

Short for 'Boolean'. Stores logical values. **Examples:** `true`, `false`

Learning Objectives: `int`

- Recognize valid mathematical and comparison operators for the `int` datatype.

Learning Objectives: `int`

- Recognize valid mathematical and comparison operators for the `int` datatype.
- Predict the outcome of integer division (/) and modulo (%) operations.

Learning Objectives: `int`

- Recognize valid mathematical and comparison operators for the `int` datatype.
- Predict the outcome of integer division (/) and modulo (%) operations.
- Declare and initialize integer variables.

Operations on `ints`

Binary Operations

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Integer Division - no remainder!)
- % (Modulo - gives the remainder)

Operations on `ints`

Binary Operations

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Integer Division - no remainder!)
- % (Modulo - gives the remainder)

Comparison Operations

- == (Equal to)
- != (Not equal to)
- < (Less than)
- <= (Less than or equal to)
- > (Greater than)
- >= (Greater than or equal to)

I Do: Integer Operations Demo

Let's walk through this code and see what it does.

I Do: Integer Operations Demo

Let's walk through this code and see what it does.

```
int main()
{
    int x = 34;
    int y = 5;

    // Integer addition
    cout << "x + y = " << x + y << endl;
    // Integer subtraction
    cout << "x - y = " << x - y << endl;
    // Integer multiplication
    cout << "x * y = " << x * y << endl;
    // Integer division (rounds down)
    cout << "x / y = " << x / y << endl;
    // Integer modulo division (remainder)
    cout << "x % y = " << x % y << endl;
    // Integer comparison ==
    cout << "x == y is " << (x == y) << endl;
    return 0;
}
```

We Do: Predict the Output!

What will the output be for each question?

We Do: Predict the Output!

What will the output be for each question?

```
int main()
{
    int x;
    // Question 1
    x = 99 / 20;
    cout << "Question 1:\t" << x << endl;
    // Question 2
    x = 27 % 10;
    cout << "Question 2:\t" << x << endl;
    // Question 3
    x = 100 / 10 / 2;
    cout << "Question 3:\t" << x << endl;
    return 0;
}
```

You Do: Integer Exercises

Your Turn

Complete the integer exercises on Schoology. Use the following template to organize your solutions. Remember to declare a variable before you use it!

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

int main()
{
    int x;    // only use "int x =" once, after that only use "x ="
    char c;

    // Question 0 (example)
    x = 1 + 1;
    cout << "Question 0:\t" << x << endl;

    // Your code for other questions goes here...
```

Exercises: INT

```cpp
#include <iostream>

using namespace std;

int main()
{
    int x;     // only use "int x =" once, after that only use "x = "
    char c;

    // Question 0 (example)
    x = 1 + 1;
    cout << "Question 0:\t" << x << endl;

    return 0;
}
```

Learning Objectives: `char`

- Understand that `char` variables store single characters using single quotes.

Learning Objectives: `char`

- Understand that `char` variables store single characters using single quotes.
- Recognize that characters are represented by numerical ASCII values.

Learning Objectives: `char`

- Understand that `char` variables store single characters using single quotes.
- Recognize that characters are represented by numerical ASCII values.
- Identify common special characters (escape sequences) like `\n` and `\t`.

Characters and ASCII

Characters are Numbers!

A `char` stores a single character like `'a'` or `'5'`. But behind the scenes, the computer stores it as an integer code. The most common system is **ASCII** (American Standard Code for Information Interchange).

Characters and ASCII

Characters are Numbers!

A `char` stores a single character like `'a'` or `'5'`. But behind the scenes, the computer stores it as an integer code. The most common system is **ASCII** (American Standard Code for Information Interchange).

Example

The character `'A'` is stored as the number 65.
The character `'a'` is stored as the number 97.

Characters and ASCII

Characters are Numbers!

A `char` stores a single character like `'a'` or `'5'`. But behind the scenes, the computer stores it as an integer code. The most common system is **ASCII** (American Standard Code for Information Interchange).

Example

The character `'A'` is stored as the number 65.
The character `'a'` is stored as the number 97.
This means we can perform math on characters! `'a' - 32` would result in `'A'`.

Common Special Characters

Some characters are not printable. We use an "escape sequence" (a backslash followed by a letter) to represent them.

Common Special Characters

Some characters are not printable. We use an "escape sequence" (a backslash followed by a letter) to represent them.

Escape Character	Character Represented	Meaning
\n	Newline	Move to a new line
\t	Horizontal Tab	Move to the next horizontal tab setting
\a	Alert	Issue an alert (annoying bell sound)
\\	Backslash	Since \ is an escape character we need to use two of them if we actually want an \ to show up
\?	Question Mark	? Character
\'	Single quote	' character
\"	Double quote	" character
\0	Null character	Insert the null character; zero or 00000000 in ASCII. We will use this a lot later in the course.

ASCII Codes (For Reference Only)

Lowercase Letter	Binary Code	Lowercase Letter	Binary Code	Uppercase Letter	Binary Code	Uppercase Letter	E
a	01100001	n	01101110	A	01000001	N	01
b	01100010	o	01101111	B	01000010	O	01
c	01100011	p	01110000	C	01000011	P	01
d	01100100	q	01110001	D	01000100	Q	01
e	01100101	r	01110010	E	01000101	R	01
f	01100110	s	01110011	F	01000110	S	01
g	01100111	t	01110100	G	01000111	T	01
h	01101000	u	01110101	H	01001000	U	01
i	01101001	v	01110110	I	01001001	V	01
j	01101010	w	01110111	J	01001010	W	01
k	01101011	x	01111000	K	01001011	X	01
l	01101100	y	01111001	L	01001100	Y	01
m	01101101	z	01111010	M	01001101	Z	01

Converting from Binary to Decimal

Example: Character 'M'

'M' is represented by: 01001101 in binary

Let's convert this to decimal

Binary Powers Calculation

$$0 \times 2^7 = 0$$

$$1 \times 2^6 = 64$$

$$0 \times 2^5 = 0$$

$$0 \times 2^4 = 0$$

$$1 \times 2^3 = 8$$

$$1 \times 2^2 = 4$$

$$0 \times 2^1 = 0$$

$$1 \times 2^0 = 1$$

Result

$$64 + 8 + 4 + 1 = 77$$

So 'M' = ASCII 77

We Do: Character Predictions

Remember, characters are just numbers. What is the output here?

We Do: Character Predictions

Remember, characters are just numbers. What is the output here?

```
int main()
{
    int x;
    char c;

    // Question 7
    x = 'a'; // 'a' has ASCII value 97
    cout << "Question 7:\t" << x << endl;

    // Question 8
    x = 'z' - 'a'; // (122 - 97)
    cout << "Question 8:\t" << x << endl;

    // Question 9
    c = 100; // ASCII 100 is 'd'
    cout << "Question 9:\t" << c << endl;
    return 0;
}
```

Learning Objectives: `bool`

- Understand the purpose of the `bool` datatype for representing logical states.

Learning Objectives: `bool`

- Understand the purpose of the `bool` datatype for representing logical states.
- Know the relationship between `true`/`false` and their integer representations `1`/`0`.

Boolean Logic

`bool`: True or False

A Boolean variable can only hold one of two values: `true` or `false`. They are the foundation of decision-making in programs.

Boolean Logic

`bool`: True or False

A Boolean variable can only hold one of two values: `true` or `false`. They are the foundation of decision-making in programs.

Booleans are also Numbers!

In C++, `true` is represented by the integer `1`, and `false` is represented by the integer `0`.

Boolean Logic

`bool`: True or False

A Boolean variable can only hold one of two values: `true` or `false`. They are the foundation of decision-making in programs.

Booleans are also Numbers!

In C++, `true` is represented by the integer `1`, and `false` is represented by the integer `0`.

Important Note

When evaluating a condition, C++ considers any non-zero integer to be `true` and only `0` to be `false`.

We Do: Boolean Predictions

What will the output be? Remember `true` is 1, `false` is 0.

We Do: Boolean Predictions

What will the output be? Remember `true` is 1, `false` is 0.

```
int main()
{
    int x;

    // Question 4
    x = true;
    cout << "Question 4:\t" << x << endl;

    // Question 5
    x = false;
    cout << "Question 5:\t" << x << endl;

    // Question 6
    x = (99 == 99); // Is 99 equal to 99? This is true.
    cout << "Question 6:\t" << x << endl;
    return 0;
}
```

Common Errors and How to Fix Them

I'm going to compile these as we come across common errors.

Error 1: Missing Semicolon

Error: `expected ';' before '}' token`

One of your lines is likely missing the semicolon at the end.
Look at the lines before the line number of the error. (in this case just before 59)

- **Meaning:** You missed a semicolon at the end of a line.
- **Fix:** Look at the line *before* the error number and add a ;

Error 2: Variable Redeclaration

Error: `redeclaration of 'int x'`

You have used 'int x =' more than once. After the declaration you only need 'x ='

```
int x = 0;
.
.
.
x = 100;
```

- **Meaning:** You declared the same variable twice.
- **Fix:** Declare the type only once. After that, just use the variable name.

Error 3: Undeclared Variable

Error: `'y' was not declared in this scope`

You have used 'y =' without declaring a type. Fix this by adding a type (int, float, char, etc.)

```
int y = 100;
```

- **Meaning:** You tried to use a variable before creating it.
- **Fix:** Make sure you have a declaration line like `int y;` before you try to use `y`.

Error 4: Wrong Quote Type for Char

Error: `invalid conversion from 'const char*' to 'char'`

You have used double quotes "a" instead of 'a' for a char datatype.

```
char c = 'a';
```

- **Meaning:** You used double quotes (`"a"`) for a `char`.
- **Fix:** `char` types must use single quotes: `char c = 'a';`

Homework

Complete the Exercises on Schoology

- I have included a short sample program on Schoology showing how you can organize your solutions.
- I am purposefully not giving you text you can copy/paste for the first few assignments.
- The goal is to improve your typing, especially finding the special characters like {}, ;, <, >, etc. on the keyboard.

Challenge Question

Find the Range!

Write C++ code to determine the range (smallest and largest values) for each of the following integer types:

1. `unsigned int`
2. `int`
3. `short int`

Note: These questions are intended as a way to challenge students who already have some programming experience. Don't worry if you have no idea how to do this, we'll get there.

Extension Question (Challenging)

Here's a hint for the first one... what happens when you subtract 1 from 0 with an unsigned integer?

Extension Question (Challenging)

Here's a hint for the first one... what happens when you subtract 1 from 0 with an unsigned integer?

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

int main()
{
    unsigned int a = 0;
    int b;
    short int c;

    cout << "The range of an unsigned int is: [0, " << a - 1 << "]\n"

    // these are incomplete, you will likely need loops to find thes
    cout << "The range of an int is: [" << 0 << ", " << 0 << "]\n";
    cout << "The range of a short int is: [" << 0 << " , " << 0 << "]
    return 0;
}
```

Loop Structures for Advanced Students

For the challenge question, you may need these loop structures:

```
// Different loop structures
while(<test>)
{
}
//-------------------------------------------------
do
{
}while(<test>); // this statement has the semicolon
//-------------------------------------------------
for(int i = 0; i < 100; i = i + 1)
{
}
```