Tony Gaddis 5th Ed
Starting Out with C++

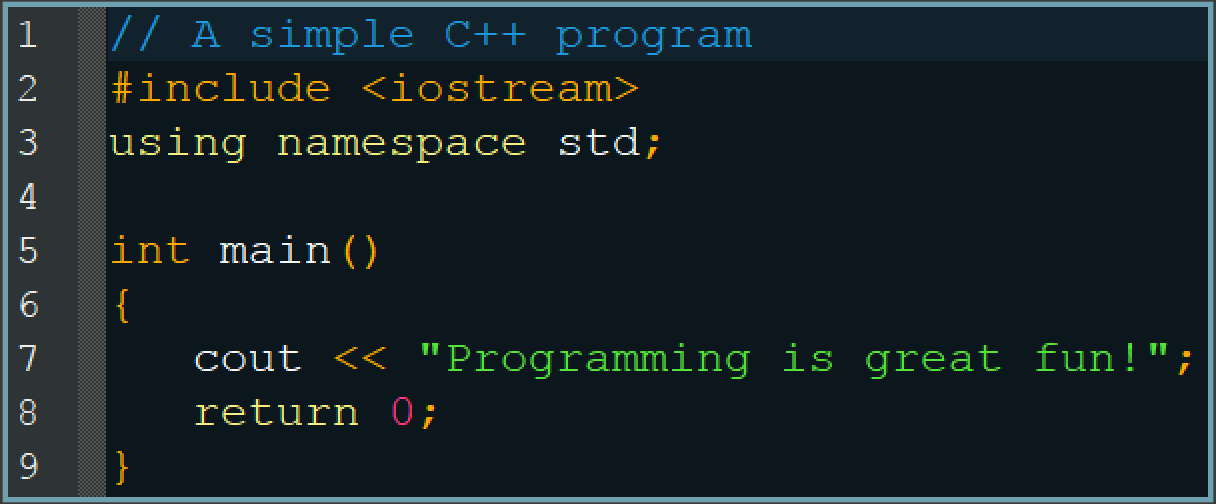
COMPUTER SCIENCE

Chapter 2

Introduction to C++

Mohammad Al-Husseini

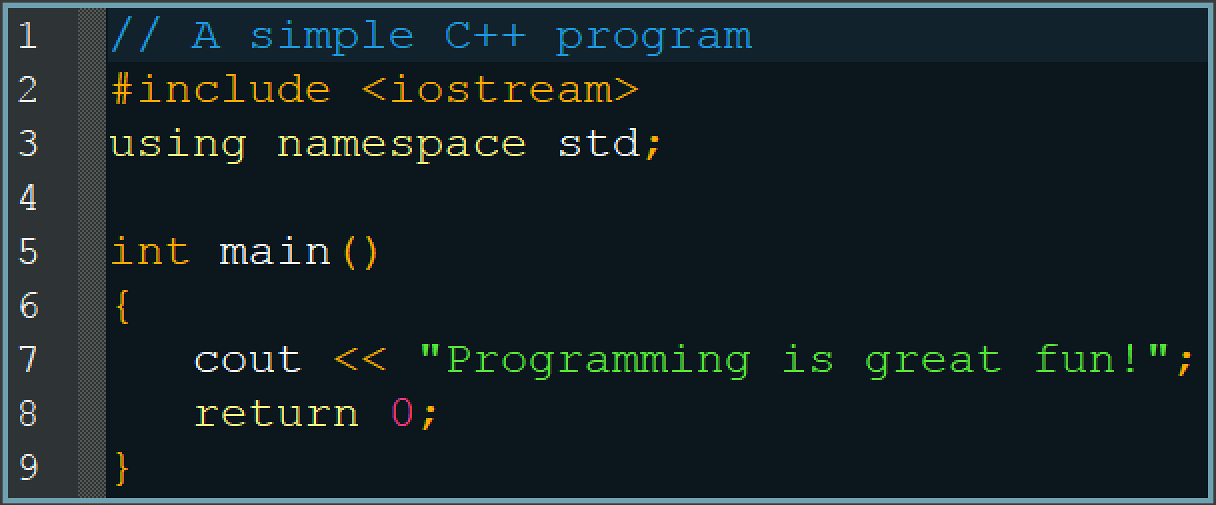
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is great fun!";
8      return 0;
9  }
```

- Comments
  - Line 1, blue in this editor/layout (notepad++)
  - Preceded by //
  - Not part of the code, assists the programmers and readers of the code in knowing what the program and or code does

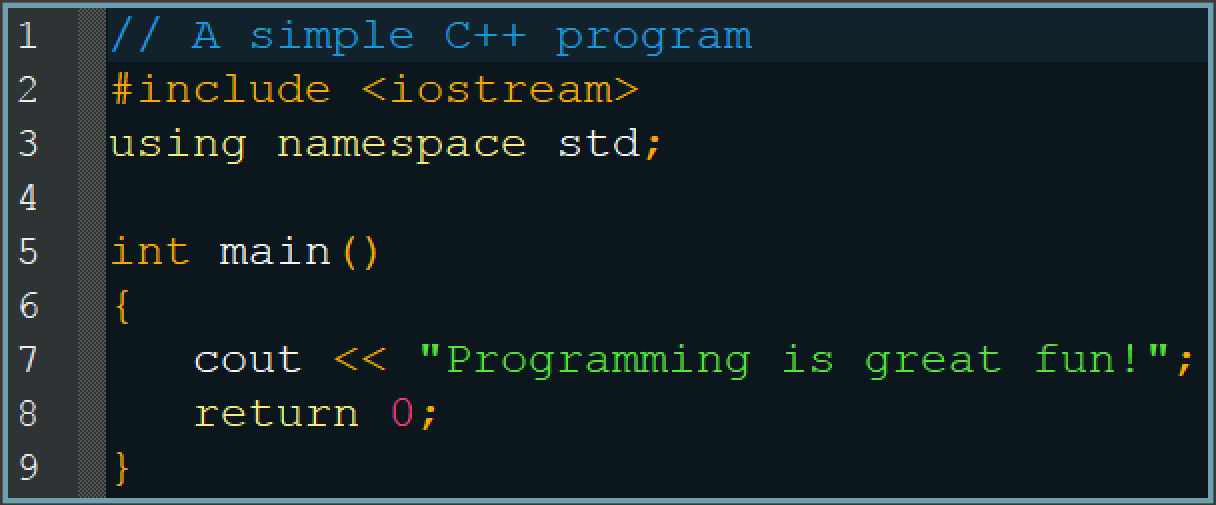
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is great fun!";
8      return 0;
9  }
```

- Preprocessor Directives
  - Line 2, orange
  - Preceded by #
  - Sets up code for the compiler
    - Can create macros, constants, and more
    - In this case includes a library of code

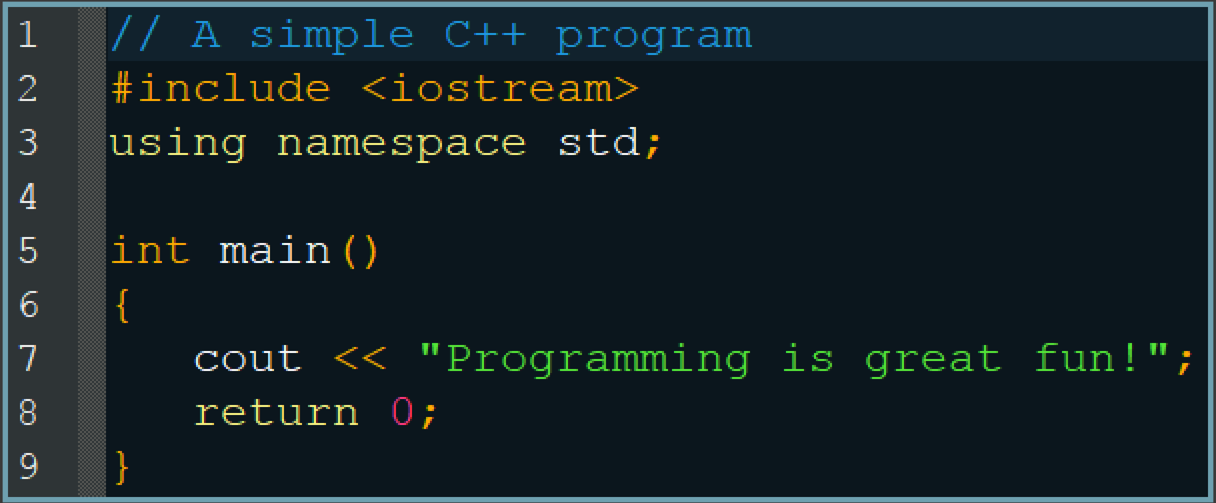
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is great fun!";
8      return 0;
9  }
```

- Namespaces
  - Line 3, yellow
  - Organizes the names of variables, functions, and objects much like a folder in an operating system
  - “Using” a namespace is generally frowned upon most programmers prefer std::cout << “etc.”;

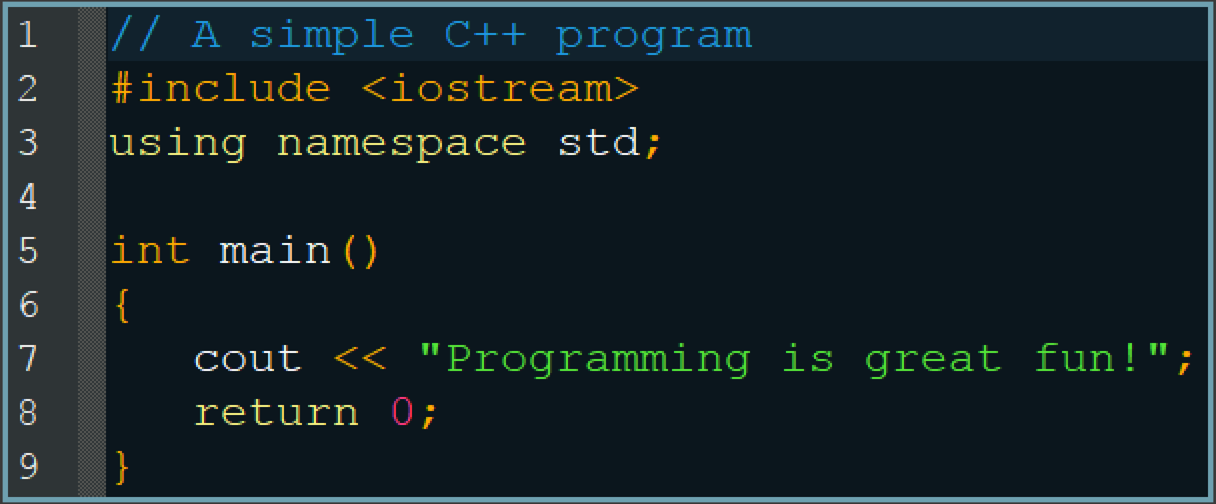
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is great fun!";
8      return 0;
9  }
```

- Functions
  - Line 5
  - All must have a return type (int), a name (main) and parameters contained within parenthesizes ()
  - All programs must contain a main (and only one)

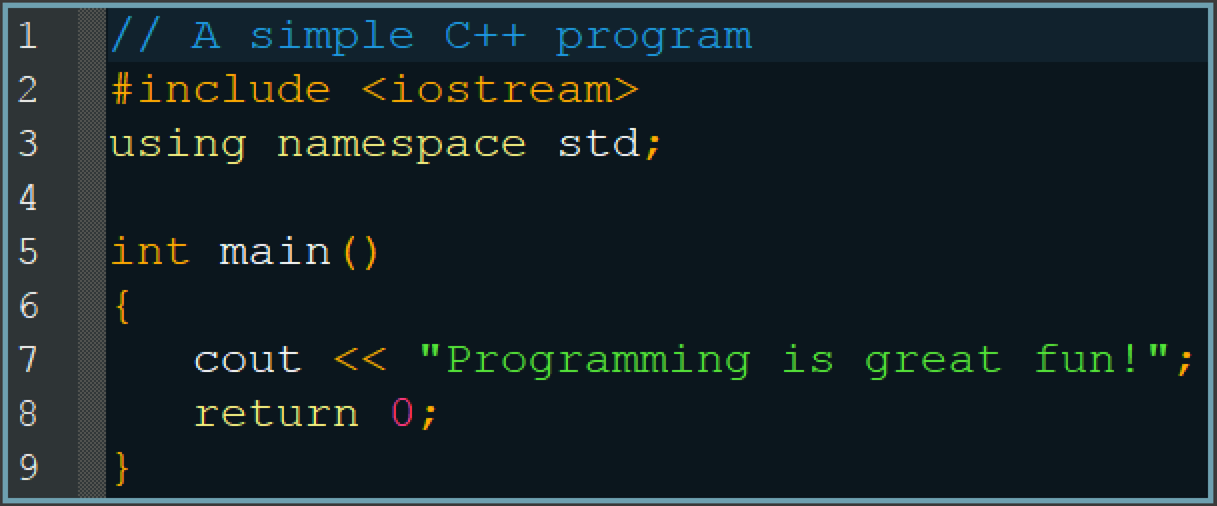
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     cout << "Programming is great fun!";
8     return 0;
9  }
```

- Braces
  - Lines 6 and 9
  - Marks the start and end of a function
  - Make sure that every brace, parenthesis, and bracket has a closing one for every starting one.

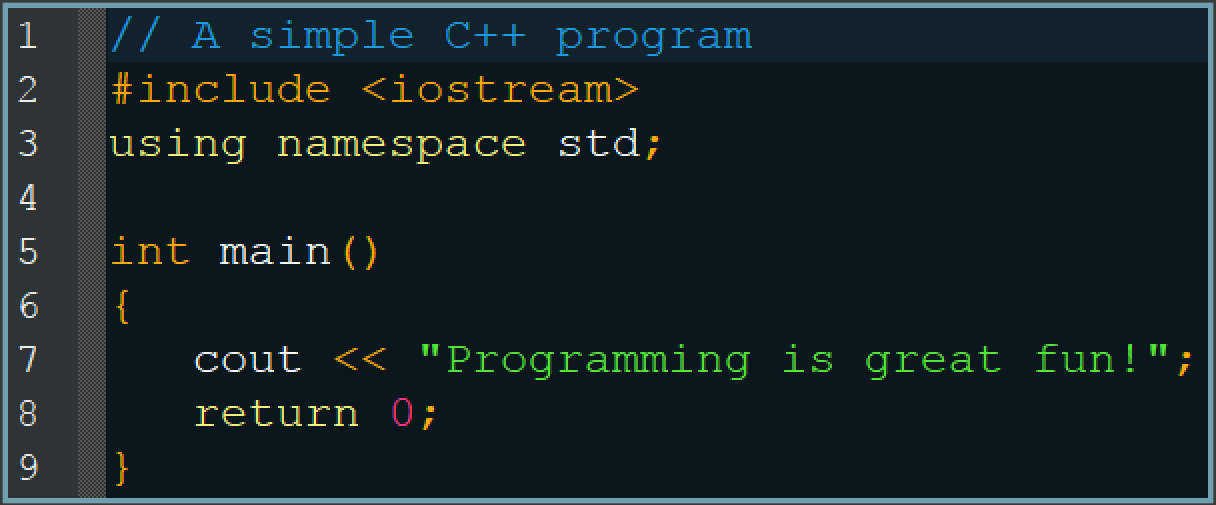
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is great fun!";
8      return 0;
9  }
```

- Output
  - Line 7
  - cout displays text to the command prompt
  - << (stream insertion operator) is used in conjunction with cout
  - “” contains the string to be outputted

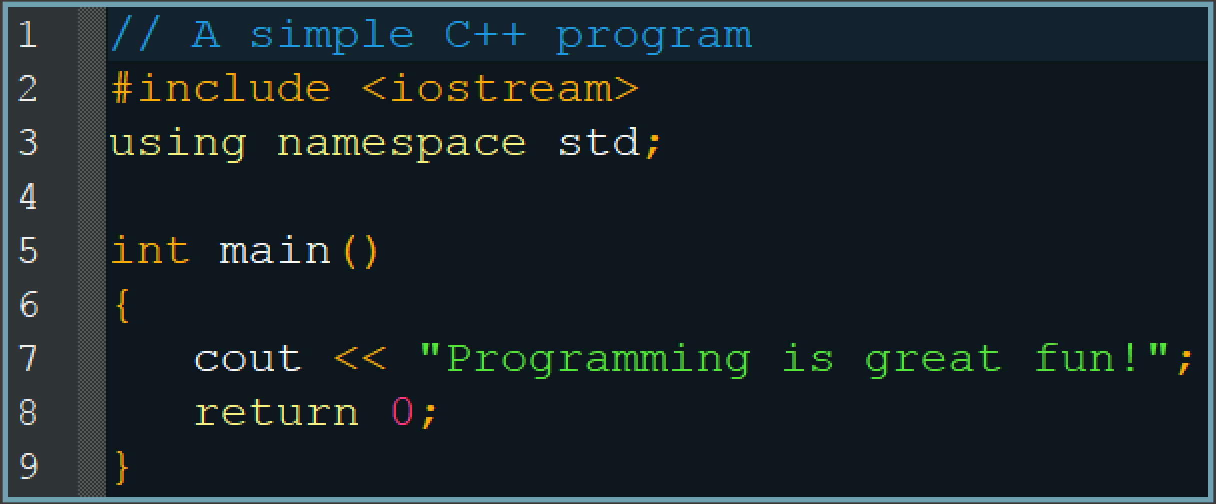
The Parts of a Program



```
1  // A simple C++ program
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Programming is great fun!";
8      return 0;
9  }
```

- Return
  - Line 8
  - The information to be returned out of the function
    - The value 0 matches the int return type on line 5
    - For main, this is typically an exit code indicating if the program completed successfully

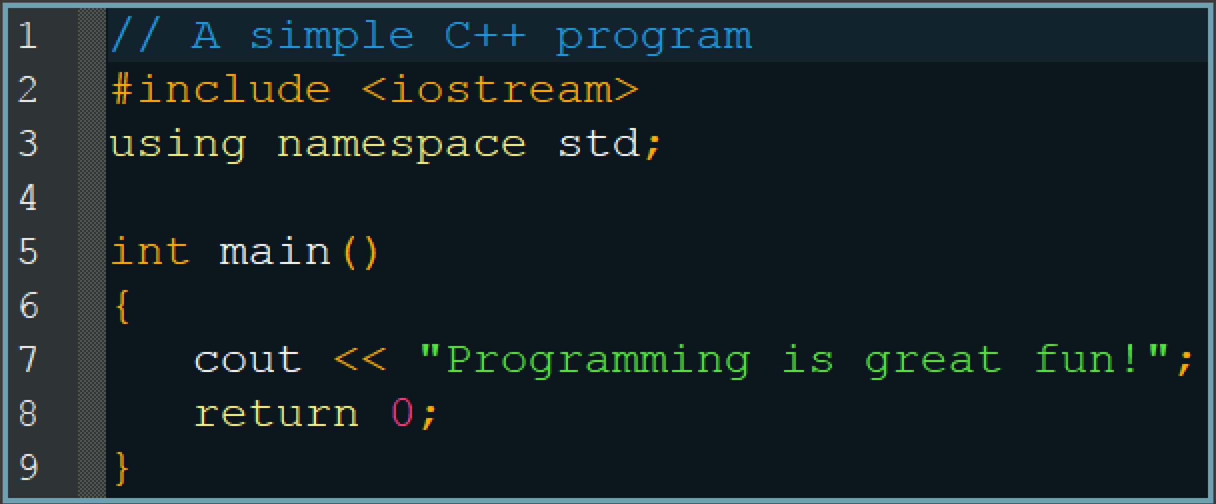
The Parts of a Program



```cpp
// A simple C++ program
#include <iostream>
using namespace std;

int main()
{
    cout << "Programming is great fun!";
    return 0;
}
```

- Notes
  - Not all lines are terminated with a semicolon (;)
    - Preprocessor directives, braces, function headers, and comments (generally) never use semicolons
    - Although not the case in this example, most lines will be terminated with semicolons

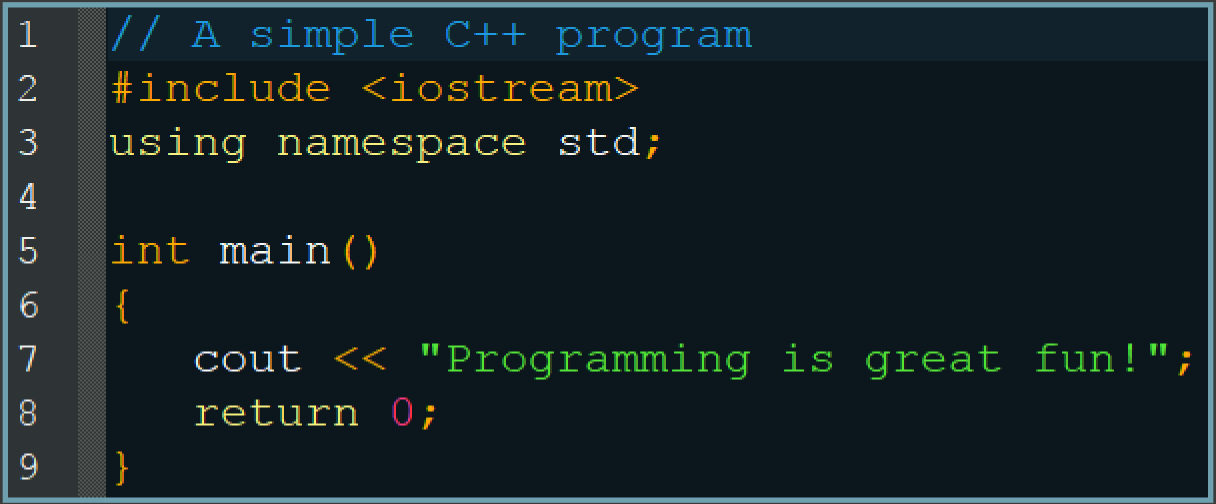
The Parts of a Program



```
// A simple C++ program
#include <iostream>
using namespace std;

int main()
{
   cout << "Programming is great fun!";
   return 0;
}
```

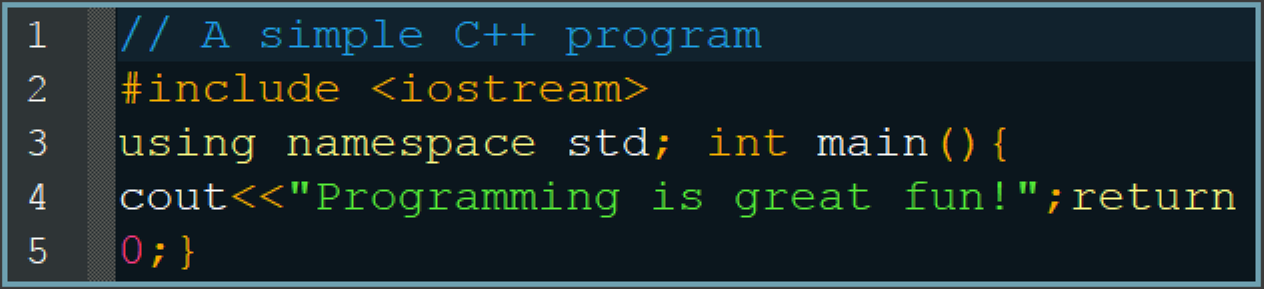
- Notes
  - C++ is case-sensitive
    - iostream is different from Iostream or IOSTREAM or any such combination
    - Reserved words such as main, include, using, return, cout, and more must be all lowercase

The Parts of a Program



```
// A simple C++ program
#include <iostream>
using namespace std;

int main()
{
    cout << "Programming is great fun!";
    return 0;
}
```

- Spacing doesn’t matter but is important

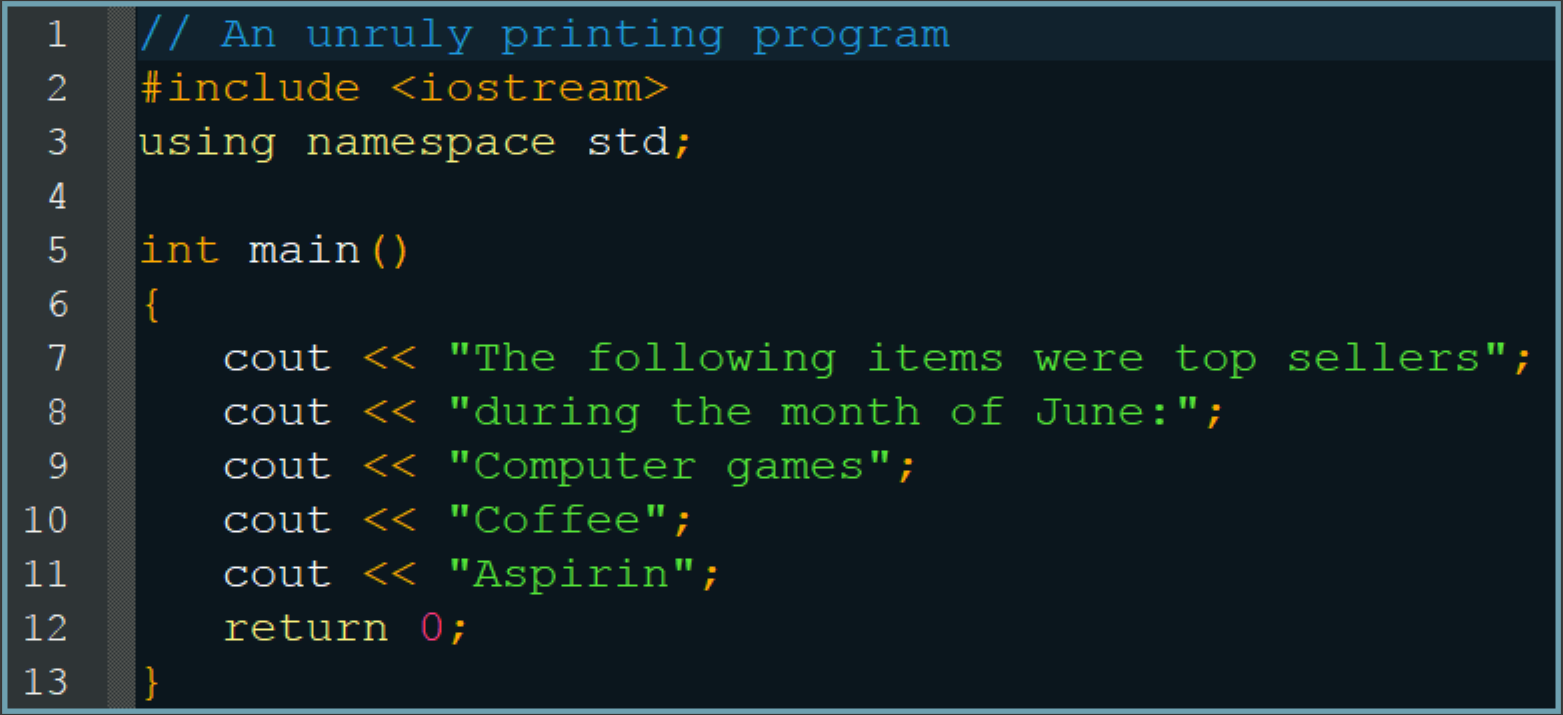


```
// A simple C++ program
#include <iostream>
using namespace std; int main(){
cout<<"Programming is great fun!";return
0;}
```

The cout Object

- Outputs to the console
- Classified as a stream object as it works with streams of data
- Used with << to insert data into the stream (hence the stream insertion operator)
- Accepts strings (“ex”), numbers, variables, certain keywords (endl), and more
- Outputs exactly what you tell it to and does not automatically add whitespace

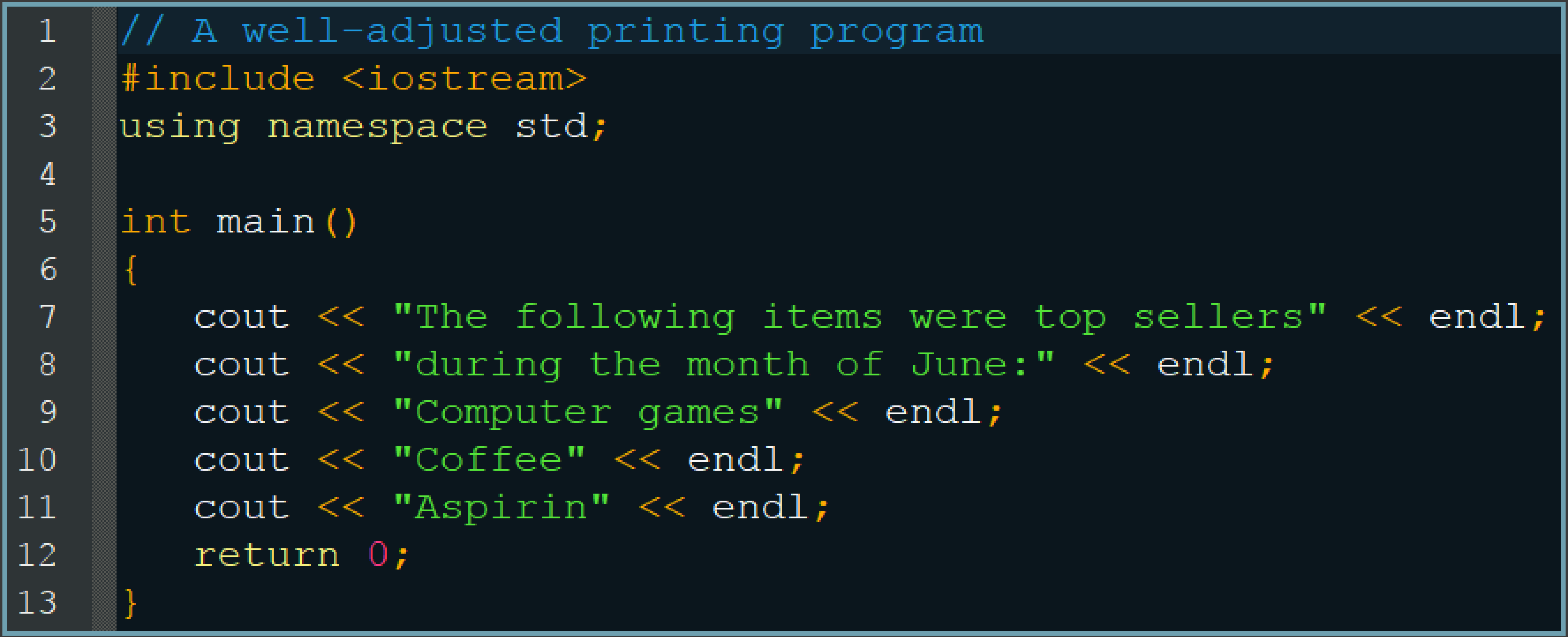
The cout Object



```cpp
// An unruly printing program
#include <iostream>
using namespace std;

int main()
{
    cout << "The following items were top sellers";
    cout << "during the month of June:";
    cout << "Computer games";
    cout << "Coffee";
    cout << "Aspirin";
    return 0;
}
```

- The following items were top sellersduring the month of June:Computer gamesCoffeeAspirin

The cout Object



```cpp
// A well-adjusted printing program
#include <iostream>
using namespace std;

int main()
{
   cout << "The following items were top sellers" << endl;
   cout << "during the month of June:" << endl;
   cout << "Computer games" << endl;
   cout << "Coffee" << endl;
   cout << "Aspirin" << endl;
   return 0;
}
```

- The following items were top sellers
  During the month of June:
  Computer games
  Coffee
  Aspirin

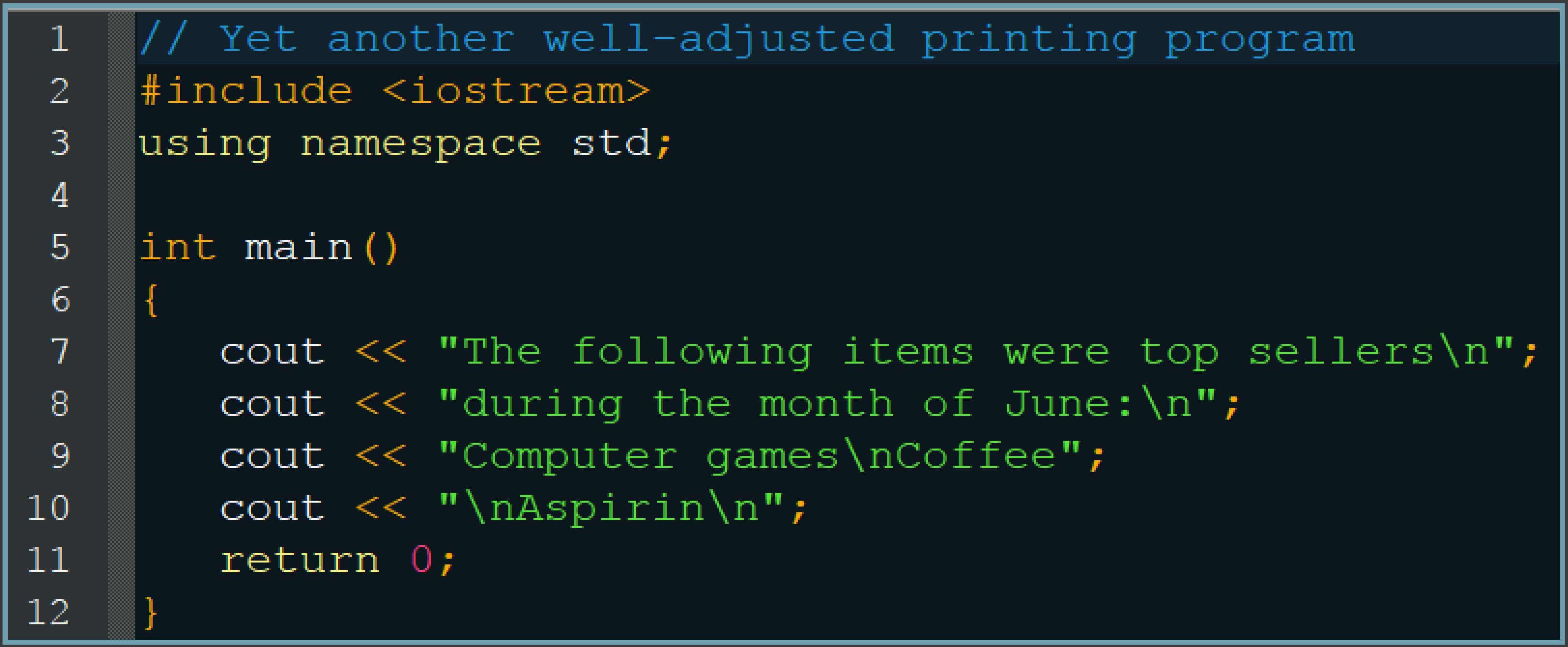
The cout Object

- Common Escape Sequences

\n	Newline
\t	Horizontal Tab
\a	Alarm
\b	Backspace
\r	Return
\\	Backslash
\’	Single Quote
\”	Double Quote

- Note: Do not put a space between the backslash and the character

The cout Object



```cpp
// Yet another well-adjusted printing program
#include <iostream>
using namespace std;

int main()
{
    cout << "The following items were top sellers\n";
    cout << "during the month of June:\n";
    cout << "Computer games\nCoffee";
    cout << "\nAspirin\n";
    return 0;
}
```

- The following items were top sellers
  During the month of June:
  Computer games
  Coffee
  Aspirin

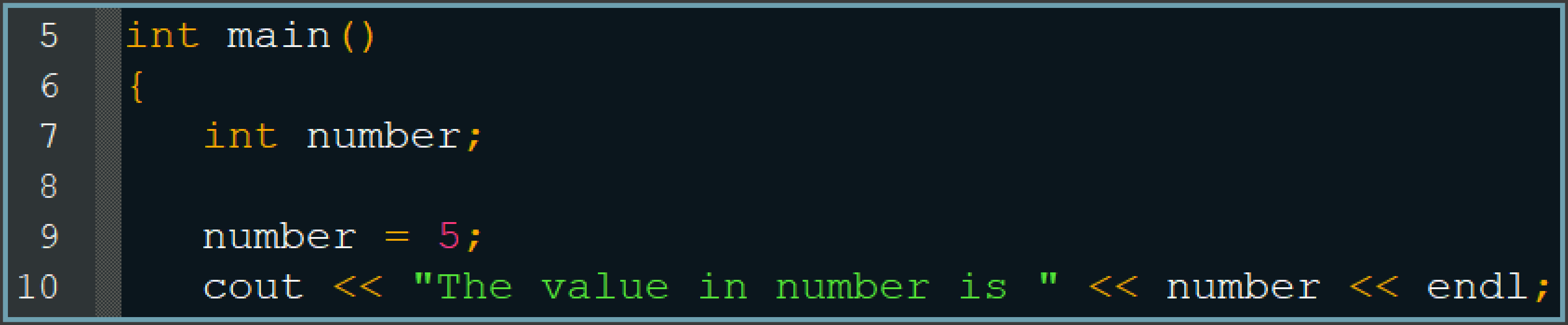
The #include Directive

- Includes libraries of code that would be too difficult / repetitious to retype each time
- Must include the name of a file
- iostream, for example, is the name of the file containing the code that defines the cout object, which is not part of the core of C++
- These preprocessor directives are not part of the code and, as such, are not terminated by a semicolon

```
#include <iostream>
```

Variables

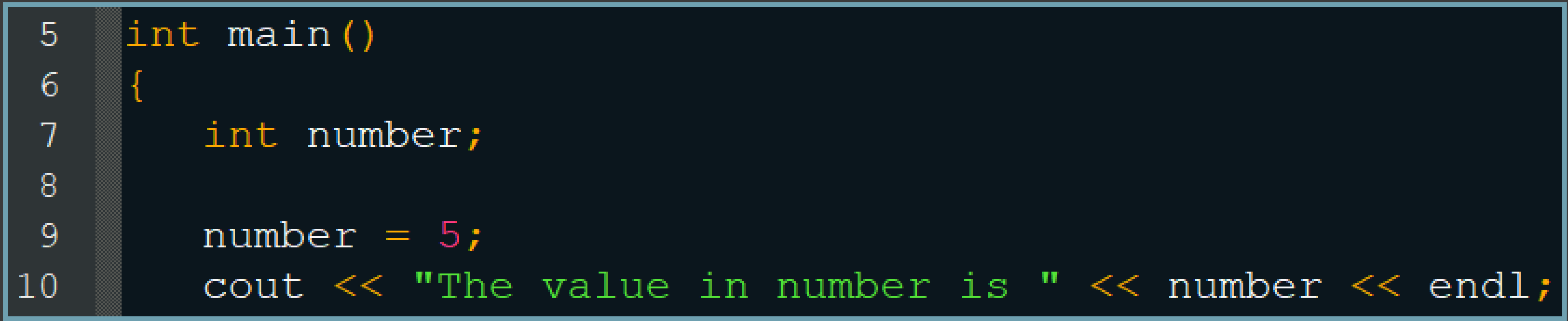
- Allow you to work with data in memory, giving you an interface to RAM
- Must be defined before trying to use them
- Upon declaration, variables contain garbage data that is unusable
  - Until they are given a value by the code, their values cannot be accessed



```
int main()
{
    int number;

    number = 5;
    cout << "The value in number is " << number << endl;
```

Variables

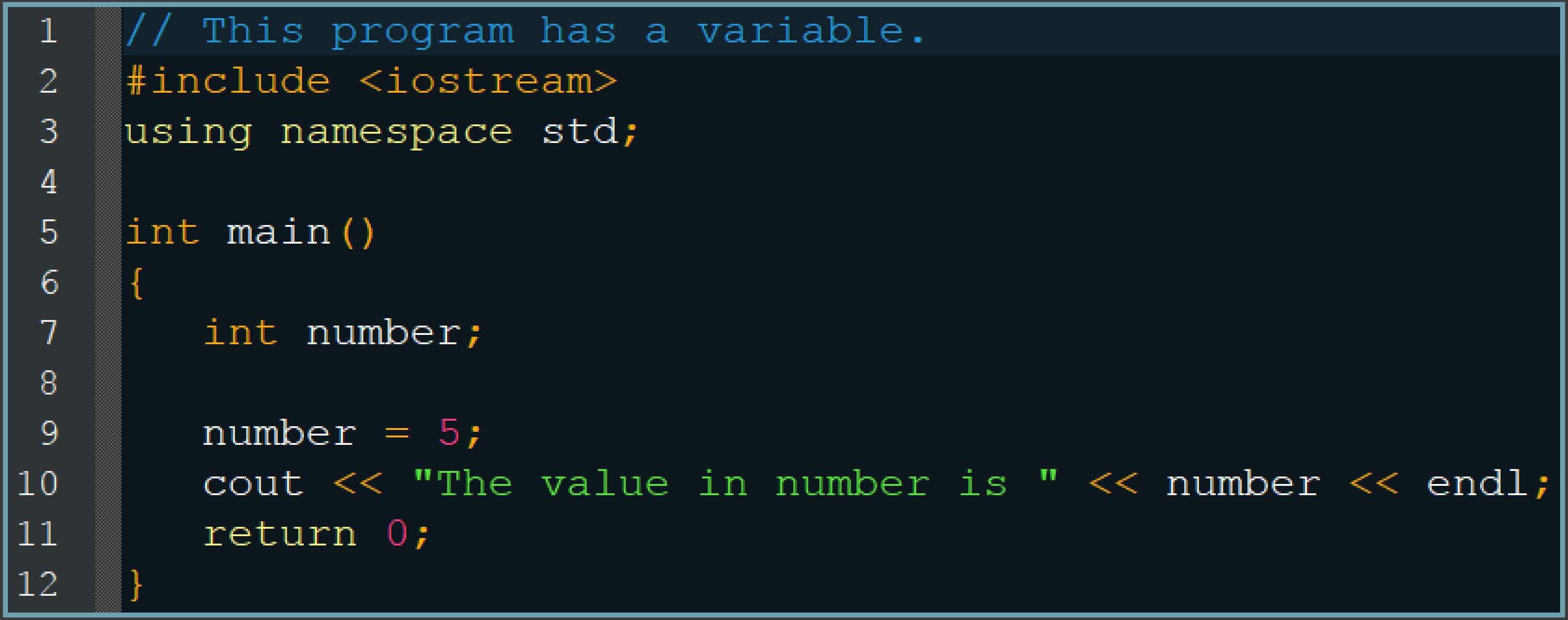
- Line 7 declares and defines the variable
- Line 9 redefines the variable with the value 5
- Line 10 accesses the value of the number variable to return the value 5 and pass the string “The value in number is 5” to cout
- Note: Although line 7 defines the variable, its value is garbage and cannot be accessed (as in line 10) until it is redefined (as in line 9)



```
 5  int main()
 6  {
 7      int number;
 8
 9      number = 5;
10      cout << "The value in number is " << number << endl;
```

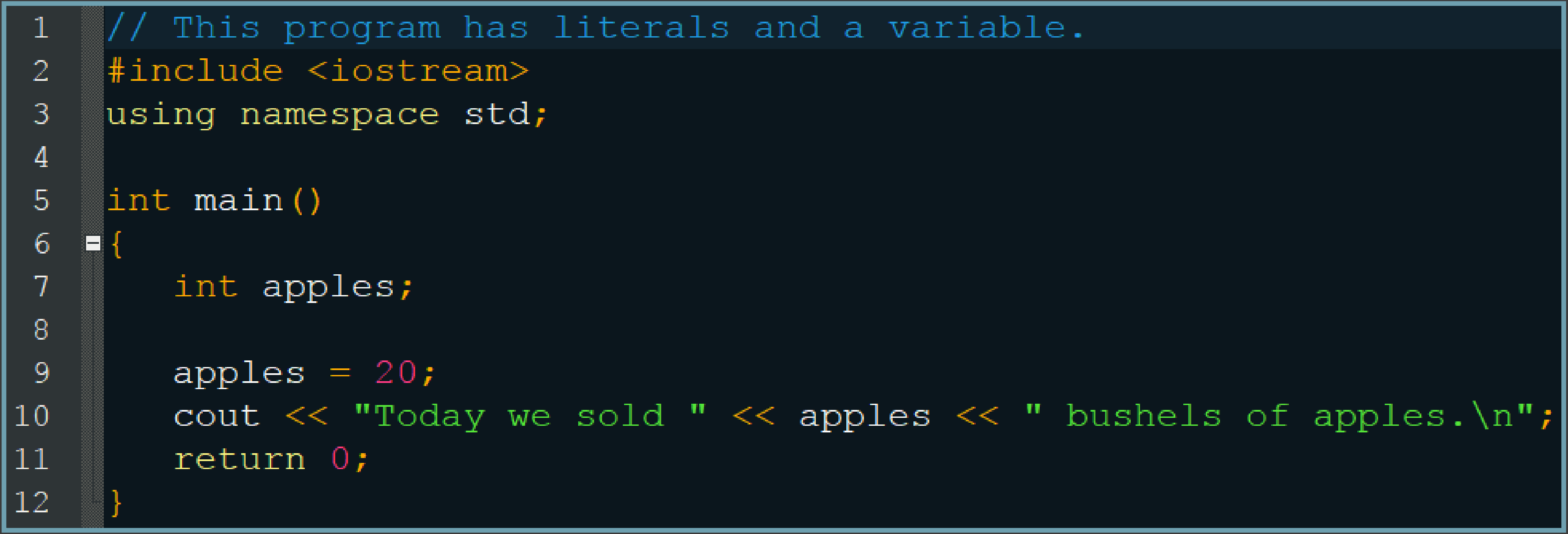
Variables

- When declaring a variable, as on line 7, you must precede its name with a data type (int in this example)
- Do not put quotations around the name of the variable when you try to access it (line 10)



```
 1  // This program has a variable.
 2  #include <iostream>
 3  using namespace std;
 4
 5  int main()
 6  {
 7      int number;
 8
 9      number = 5;
10      cout << "The value in number is " << number << endl;
11      return 0;
12  }
```

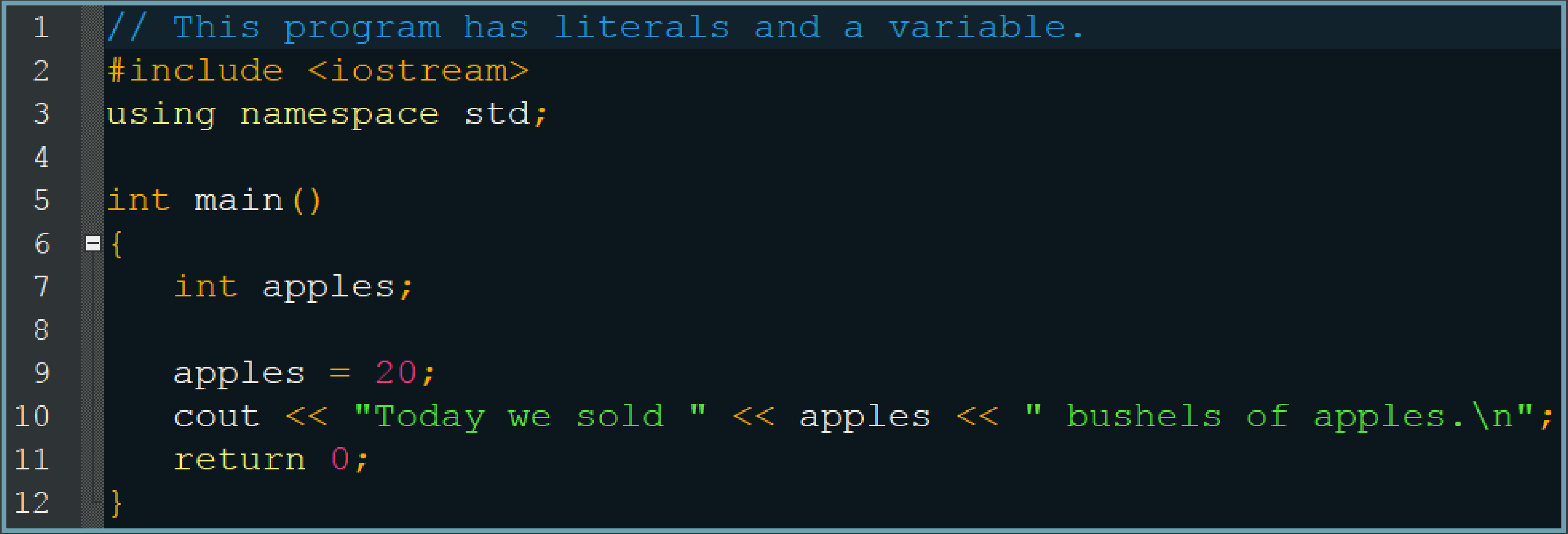
Literals



```cpp
// This program has literals and a variable.
#include <iostream>
using namespace std;

int main()
{
   int apples;

   apples = 20;
   cout << "Today we sold " << apples << " bushels of apples.\n";
   return 0;
}
```

- Values that do not change during the course of the program, unlike variables, also referred to as constants
- Hold values without the need to reserve space in memory

Literals



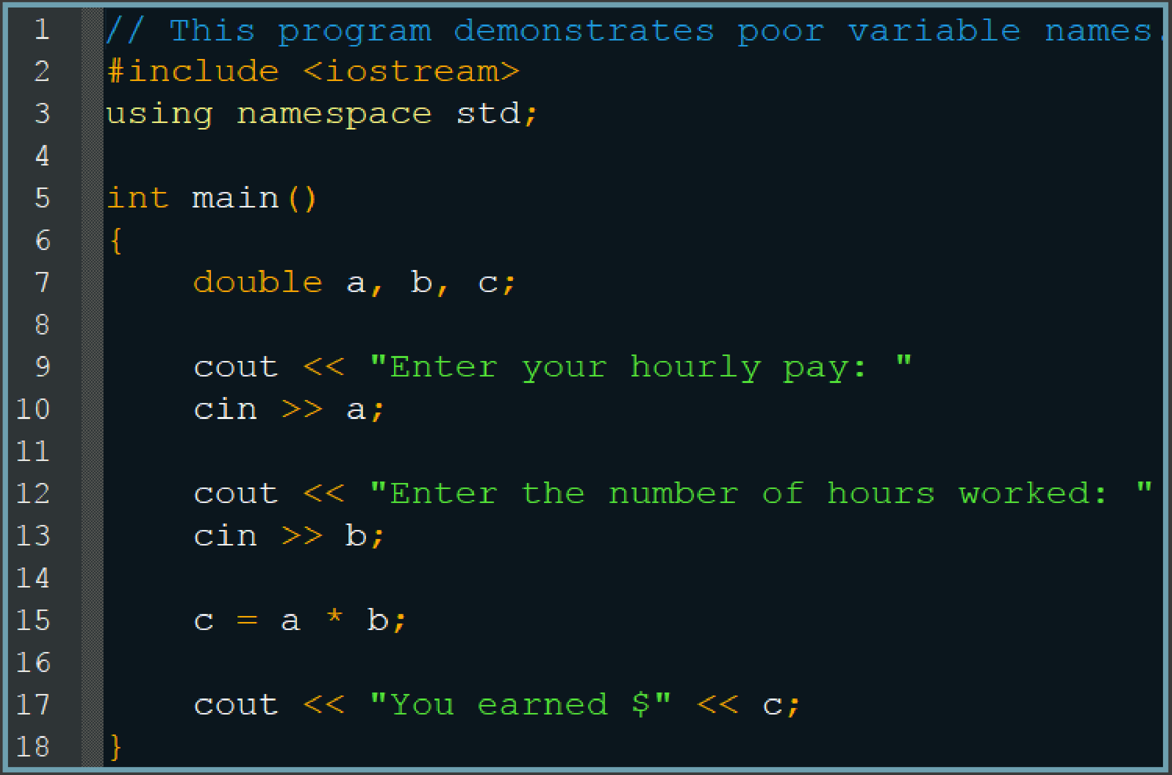
```cpp
// This program has literals and a variable.
#include <iostream>
using namespace std;

int main()
{
    int apples;

    apples = 20;
    cout << "Today we sold " << apples << " bushels of apples.\n";
    return 0;
}
```

- The literals in this program are:

20	-	Integer Literal
“Today we sold “	-	String Literal
“Bushes of apples.\n”	-	String Literal
0	-	Integer Literal

Identifiers

- You are free to choose whatever names you want for your variables, with a few recommendations and exceptions
  - Pick meaningful names



```
// This program demonstrates poor variable names.
#include <iostream>
using namespace std;

int main()
{
    double a, b, c;

    cout << "Enter your hourly pay: ";
    cin >> a;

    cout << "Enter the number of hours worked: ";
    cin >> b;

    c = a * b;

    cout << "You earned $" << c;
}
```

Identifiers

- You cannot use a key word (IDE will warn you)

asm	auto	break	bool	case
catch	char	class	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
false	float	for	friend	goto
if	inline	int	long	mutable
namespace	new	operator	private	protected
public	register	reinterpret_cast	return	short
signed	sizeof	static	static_cast	struct
switch	template	this	throw	true
try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t	while			

IDENTIFIERS

- You must obey the following rules:

1. The first character must be a letter or underscore
2. After the first, you can use letters, underscores, or numbers
3. They are case-sensitive

dayOfWeek	-	Legal
_employee_num	-	Legal
June1997	-	Legal
3dGraph	-	Illegal, begins with a number
Mixture#3	-	Illegal, contains a character that is not a letter, number, or underscore

Identifiers

- Conventions:
  - Begin with a lowercase letter
    - Ex: apples
  - If two words, either separate with an underscore
    - Ex: total_pay
  - Or make the second word a capital letter
    - Ex: totalPay
  - This all improves readability
    - Ex: totalpay
    - Ex: TotalPay
    - Ex: avoidreallylongvariablenames

Data Types

- Different containers for different data
- Two real types
  - Characters
    - Letters
    - Strings of letters with meaning
  - Numbers
    - Whole
    - Decimal
    - Signed
    - Numbers of varying sizes
    - True/False logic

Char Data

- This data type only holds a single character
- char only takes 1 byte of memory
- Must use single quotation marks with the character
  - char letter = ‘a’;
- Can also use a whole number (integer)
  - char letter = 97;
  - This is the ASCII representation of the character a

Char Data

- You have also seen character literals as strings
  - cout << “The letter is: “ << letter;
  - You will learn how to store these in variables later
  - These strings contain multiple characters and are encased with double quotation marks
  - They automatically contain a null terminator character that signifies their end
    - Hence they always contain multiple characters >= 2

T	h	e		l	e	t	t	e	r		i	s	:		\0

Integer Data

- Only holds whole numbers
- Size of the numbers storable varies
    - short
    - int
    - long
    - unsigned

Data Type	Size	Range
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
int	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned int	4 bytes	0 to 4,294,967,295
long	>= 4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	>= 4 bytes	0 to 4,294,967,295

Integer Data

- C++ will generally store long ints as regular ints
- You must force this behavior using the L tag
- long int number = 64L;
- You may also create long long variables to be at least 8 bytes
- There are other number systems as well
  - Hexadecimal 0xF4
  - Octal 031
  - Binary 01110100

Floating-Point Data

- Holds fractional or decimal number values

Data Type	Size	Range
float	4 bytes	+/- 3.4E-38 to +/- 3.4E38
double	8 bytes	+/- 1.7E-308 to +/- 1.7E308
long double	>= 8 bytes	+/- 1.7E-308 to +/- 1.7E308

- There are no unsigned floating point types
- float will automatically be promoted to double and long double will be demoted to double
  - float number = 1.5F;
  - long double number = 1.5L;

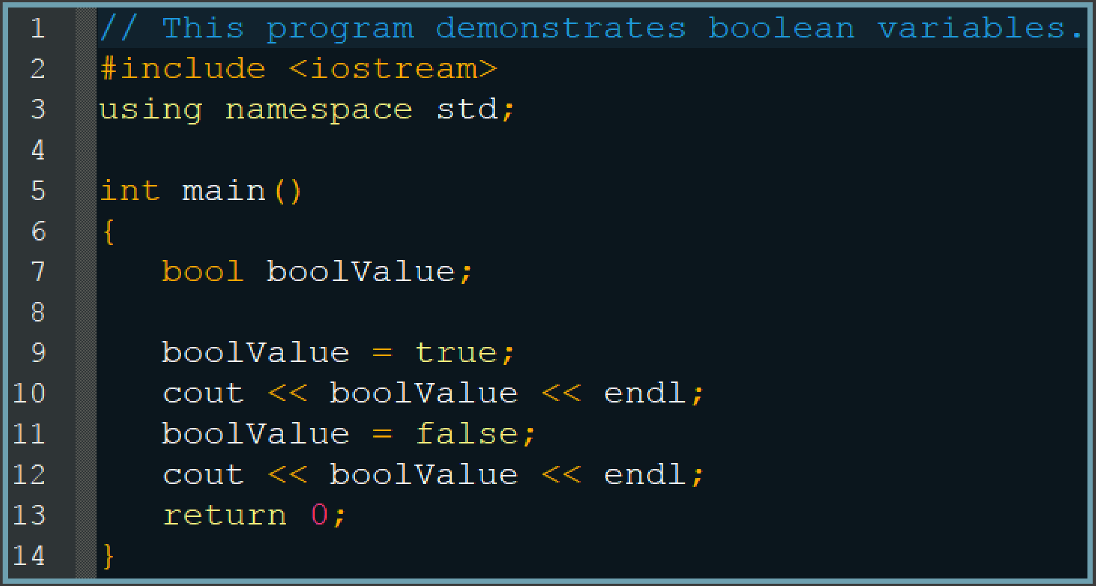
Floating-Point Data

- You may assign a floating point number to an integer, but the results may seem unexpected
  - The decimal portion will be truncated (lost) NOT rounded

    -
      ```
      int i;
      float f = 7.5;
      i = f;            // i will equal 7
      ```
- You may also assign integers to floating points without issue
- Note that floating point numbers take much more memory than integers
  - Don’t use a double when you only need an int

Bool Data

- Represents true and false
- Really stored as a number
  - false = 0
  - true = any non-zero value, usually 1



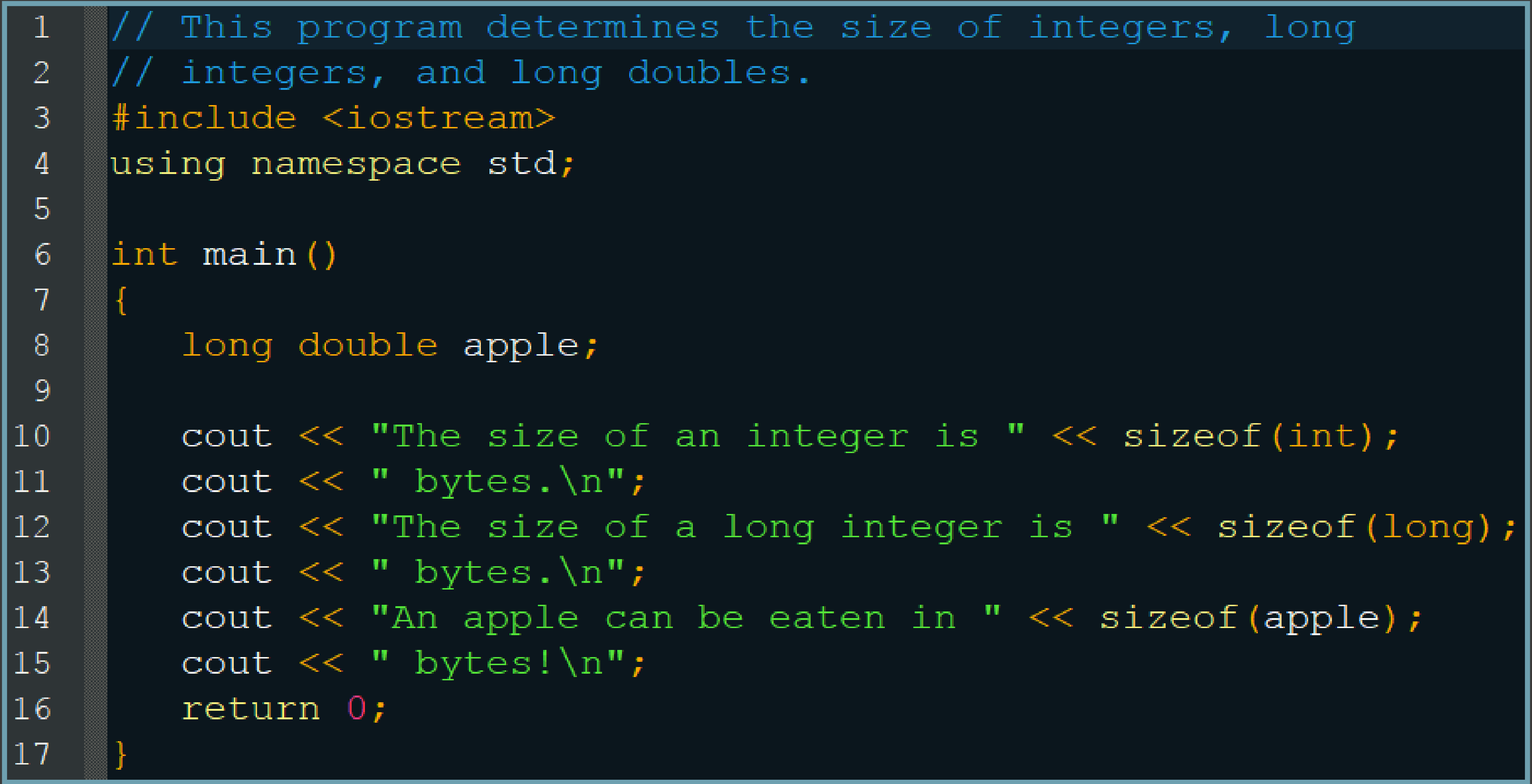
```cpp
// This program demonstrates boolean variables.
#include <iostream>
using namespace std;

int main()
{
   bool boolValue;

   boolValue = true;
   cout << boolValue << endl;
   boolValue = false;
   cout << boolValue << endl;
   return 0;
}
```

- Outputs a 1 and then a 0

Data Type Size

- All of the sizes listed before are for the average system
- These sizes vary depending on your operation system
- You can determine the size of a data type on your system using the sizeof operator
- sizeof(variable or datatype)

Data Type Size



```cpp
// This program determines the size of integers, long
// integers, and long doubles.
#include <iostream>
using namespace std;

int main()
{
    long double apple;

    cout << "The size of an integer is " << sizeof(int);
    cout << " bytes.\n";
    cout << "The size of a long integer is " << sizeof(long);
    cout << " bytes.\n";
    cout << "An apple can be eaten in " << sizeof(apple);
    cout << " bytes!\n";
    return 0;
}
```

- The size of an integer is 4 bytes.
- The size of a long integer is 4 bytes.
- An apple can be eaten in 8 bytes!

Variable Assignments

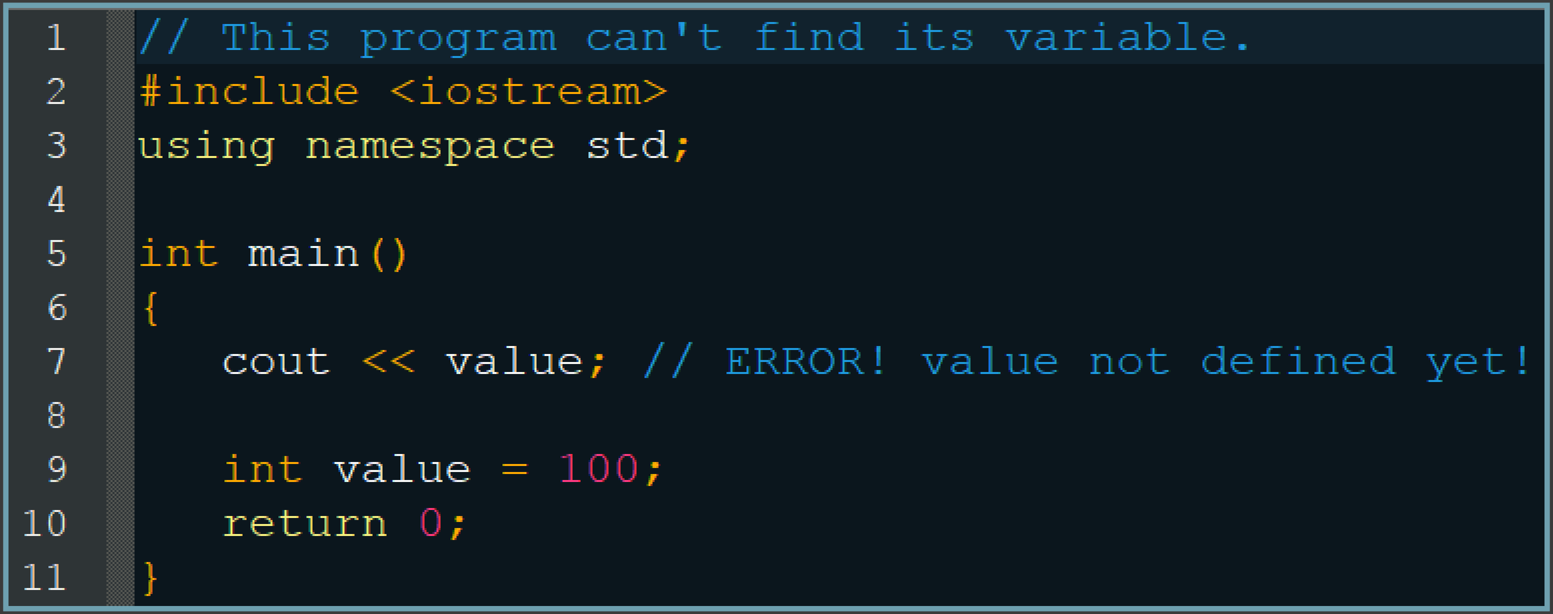
- Assignment operator =
  - Works with two operands
  - Right operand must be an rvalue (expression that has a value)
  - Left operand or lvalue must have a location in memory
    - unitsSold = 12;
    - 12 = unitsSold; // Wrong

Variable Initializations

- You may declare and initialize a variable in a single statement
  - double interestRate = 12.9;
- You may also declare/initialize multiple variables in one statement
  - int month = 2, days = 28;
  - int flightNum = 89, travelTime, departure = 10, distance;

Scope

- You cannot use a variable (and many other things) before you have declared them



```
// This program can't find its variable.
#include <iostream>
using namespace std;

int main()
{
   cout << value; // ERROR! value not defined yet!

   int value = 100;
   return 0;
}
```

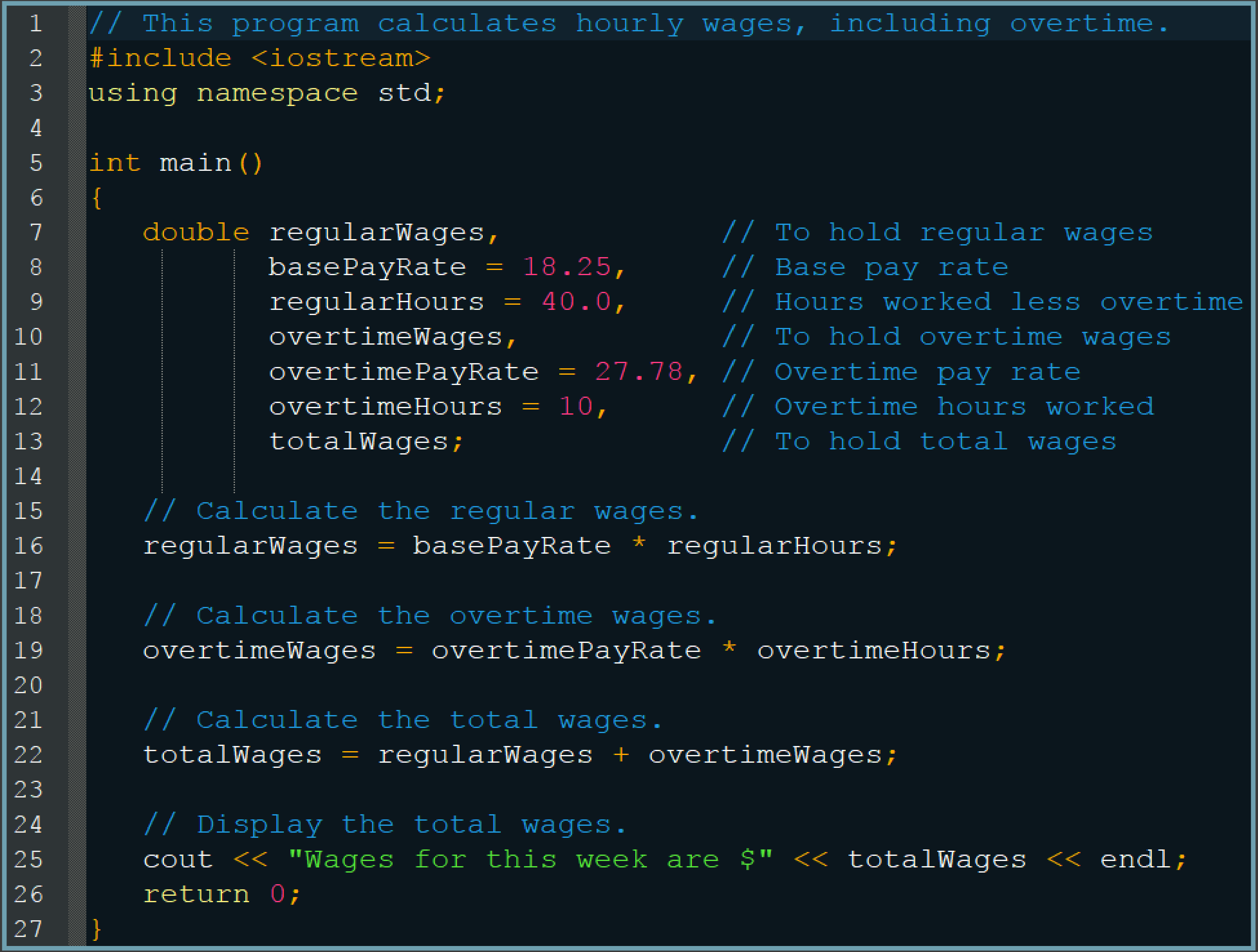
Arithmetic Operators

- Three varieties
  - Unary
  - Binary
  - Trinary
- This number of operands sometimes determines which operator you are using
  - Ex. -
    - number = -5;   // Unary negation operator
    - number = 5 - 1; // Binary subtraction operator

Arithmetic Operators

- + Addition
- - Subtraction/Negation
- * Multiplication
- / Division
- % Modulus
- Order of operations just like math
- Be careful with division
  - double number = 21 / 2; // The result is 10
  - 21 and 2 are integers, thus integer division
  - double number = 21 / 2.0; // Correct 10.5

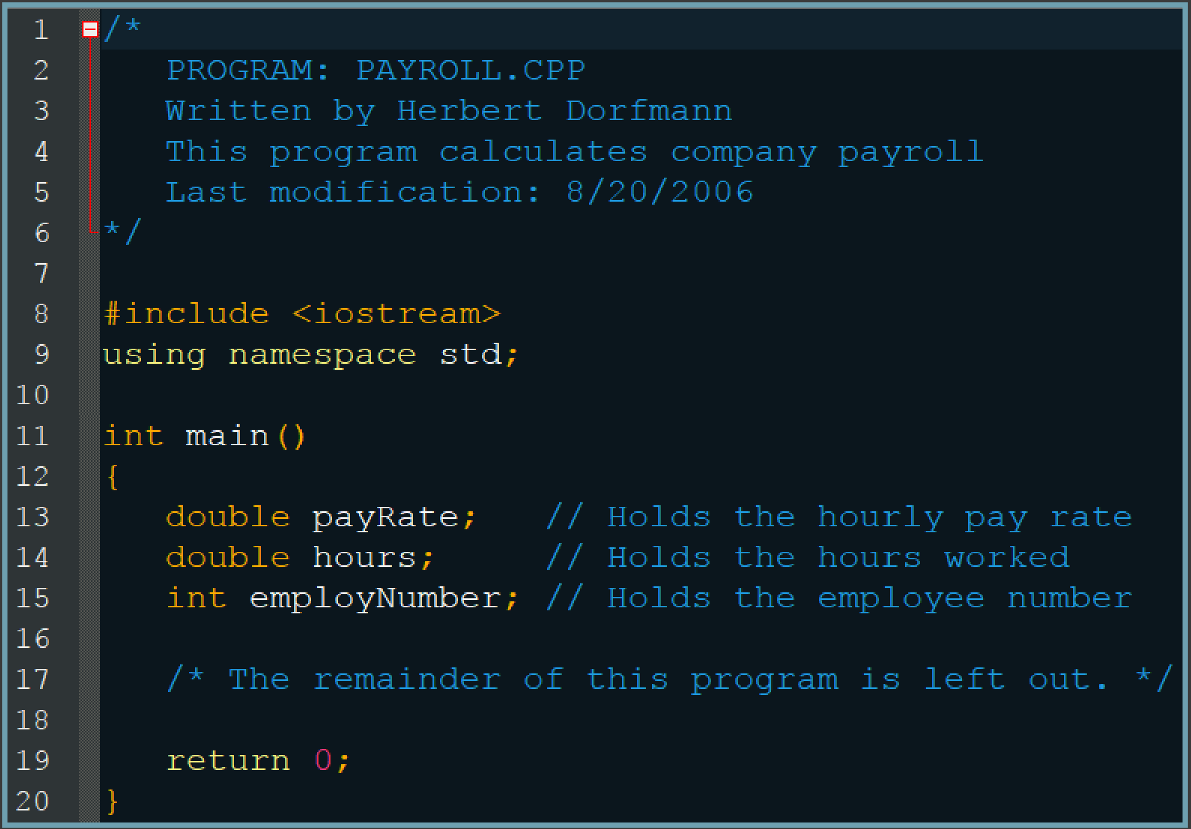
Arithmetic Operators



```
// This program calculates hourly wages, including overtime.
#include <iostream>
using namespace std;

int main()
{
    double regularWages,            // To hold regular wages
           basePayRate = 18.25,     // Base pay rate
           regularHours = 40.0,     // Hours worked less overtime
           overtimeWages,           // To hold overtime wages
           overtimePayRate = 27.78, // Overtime pay rate
           overtimeHours = 10,      // Overtime hours worked
           totalWages;              // To hold total wages

    // Calculate the regular wages.
    regularWages = basePayRate * regularHours;

    // Calculate the overtime wages.
    overtimeWages = overtimePayRate * overtimeHours;

    // Calculate the total wages.
    totalWages = regularWages + overtimeWages;

    // Display the total wages.
    cout << "Wages for this week are $" << totalWages << endl;
    return 0;
}
```

Comments

- Single line comments are proceeded by //
- Multiline comments start with
  /* and end with */



```
/*
   PROGRAM: PAYROLL.CPP
   Written by Herbert Dorfmann
   This program calculates company payroll
   Last modification: 8/20/2006
*/

#include <iostream>
using namespace std;

int main()
{
   double payRate;   // Holds the hourly pay rate
   double hours;     // Holds the hours worked
   int employNumber; // Holds the employee number

   /* The remainder of this program is left out. */

   return 0;
}
```