

Chapter 2:

Introduction

to

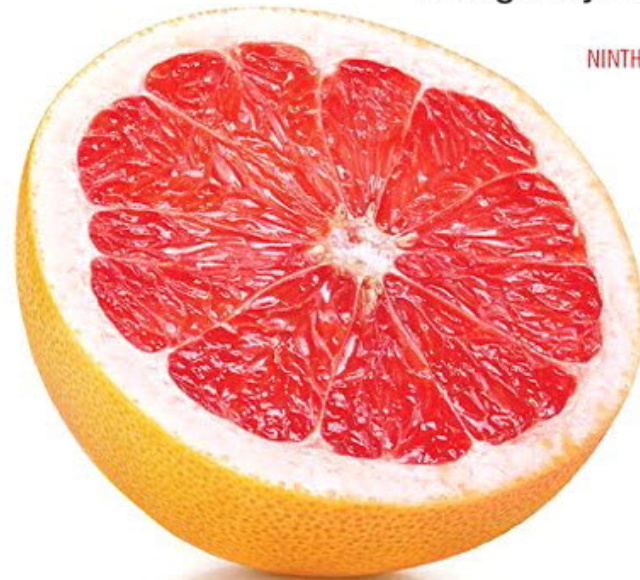
C++

starting out with >>>

C++

From Control Structures
through Objects

NINTH EDITION



TONY GADDIS

Hexa-decimal, octal, and decimal

- All numbers are stored as binary numbers.
- Data stored in an integer variable can be displayed in
 - Octal
 - Hexadecimal
 - Decimal
- Hexadecimal and octal values can be stored in an integer variable directly
 - `int h_value = 0xA;`
 - `int o_value = 010;`

Hexa-decimal, octal, and decimal

0 _{hex} = 0 _{dec} = 0 _{oct}	0	0	0	0
1 _{hex} = 1 _{dec} = 1 _{oct}	0	0	0	1
2 _{hex} = 2 _{dec} = 2 _{oct}	0	0	1	0
3 _{hex} = 3 _{dec} = 3 _{oct}	0	0	1	1
4 _{hex} = 4 _{dec} = 4 _{oct}	0	1	0	0
5 _{hex} = 5 _{dec} = 5 _{oct}	0	1	0	1
6 _{hex} = 6 _{dec} = 6 _{oct}	0	1	1	0
7 _{hex} = 7 _{dec} = 7 _{oct}	0	1	1	1
8 _{hex} = 8 _{dec} = 10 _{oct}	1	0	0	0
9 _{hex} = 9 _{dec} = 11 _{oct}	1	0	0	1
A _{hex} = 10 _{dec} = 12 _{oct}	1	0	1	0
B _{hex} = 11 _{dec} = 13 _{oct}	1	0	1	1
C _{hex} = 12 _{dec} = 14 _{oct}	1	1	0	0
D _{hex} = 13 _{dec} = 15 _{oct}	1	1	0	1
E _{hex} = 14 _{dec} = 16 _{oct}	1	1	1	0
F _{hex} = 15 _{dec} = 17 _{oct}	1	1	1	1

2.9

- Floating-Point Data Types

Floating-Point Data Types

- The floating-point data types are:

`float`
`double`

- floating point will ensure that the 7 decimal places after the decimal point is correct
- double point will ensure that the 15 decimal places after the decimal point is correct.
- They can hold real numbers such as:
12.45 -3.8
- Stored in a form similar to scientific notation
- All floating-point numbers are signed (they can store both negative and positive values)

Floating-Point Literals

- Can be represented in

- Fixed point (decimal) notation:

`31.4159`

`0.0000625`

- E notation:

`3.14159E1`

`6.25e-5`

- Are `double` by default

- To indicate floating point, use the `f` at the end, `3.14159f`

Floating-Point Data Types in Program 2-16

Program 2-16

```
1  // This program uses floating point data types.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      float distance;
8      double mass;
9
10     distance = 1.495979E11;
11     mass = 1.989E30;
12     cout << "The Sun is " << distance << " meters away.\n";
13     cout << "The Sun\'s mass is " << mass << " kilograms.\n";
14     return 0;
15 }
```

Program Output


The Sun is 1.49598e+011 meters away.

The Sun's mass is 1.989e+030 kilograms.

2.10

The `bool` Data Type

The `bool` Data Type

 Represents values that are `true` or `false`

 `bool` variables are stored as small integers

 `false` is represented by 0, `true` by 1:

```
bool allDone = true;    allDone finished
bool finished = false;  

|   |
|---|
| 1 |
|---|



|   |
|---|
| 0 |
|---|


```

Boolean Variables in Program 2-17

Program 2-17

```
1  // This program demonstrates boolean variables.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      bool boolValue;
8
9      boolValue = true;
10     cout << boolValue << endl;
11     boolValue = false;
12     cout << boolValue << endl;
13     return 0;
14 }
```

Program Output

1
0

2.11

- Determining the Size of a Data Type

Determining the Size of a Data Type

 The `sizeof` operator gives the size of any data type or variable:

```
double amount;  
cout << "A double is stored in "  
      << sizeof(double) << "bytes\n";  
cout << "Variable amount is stored in "  
      << sizeof(amount)  
      << "bytes\n";
```

2.12

- Variable Assignments and Initialization

Variable Assignments and Initialization

- An assignment statement uses the `=` operator to store a value in a variable.

```
item = 12;
```

- This statement assigns the value 12 to the `item` variable.

- This will NOT work:

```
// ERROR!  
12 = item;
```

Variable Initialization in Program 2-19

Program 2-19

```
1  // This program shows variable initialization.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      int month = 2, days = 28;
8
9      cout << "Month " << month << " has " << days << " days.\n";
10     return 0;
11 }
```



Program Output

Month 2 has 28 days.

2.13

- Scope

Scope

-  The scope of a variable: the part of the program in which the variable can be accessed
-  A variable cannot be used before it is defined

Variable Out of Scope in Program 2-20

Program 2-20

```
1  // This program can't find its variable.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << value; // ERROR! value not defined yet!
8
9      int value = 100;
10     return 0;
11 }
```

Variable Scope (cont)

Scope is defined by using the curly brackets.

The variables are only available to use within the scope, where they are defined.

Let us study the following code:

```
{  
    {  
        int value = 100;  
    }  
    cout<< value;  
}
```

2.14

- Arithmetic Operators

Arithmetic Operators

- Used for performing numeric calculations

- C++ has unary, binary, and ternary operators:

 - unary (1 operand) -5


 - binary (2 operands) $13 - 7$

 - ternary

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	VALUE OF ans
+	addition	<code>ans = 7 + 3;</code>	10
-	subtraction	<code>ans = 7 - 3;</code>	4
*	multiplication	<code>ans = 7 * 3;</code>	21
/	division	<code>ans = 7 / 3;</code>	2
%	modulus	<code>ans = 7 % 3;</code>	1

A Closer Look at the / Operator

 / (division) operator performs integer division if both operands are integers

```
cout << 13 / 5;      // displays 2  
cout << 91 / 7;      // displays 13
```

 If either operand is floating point, the result is floating point

```
cout << 13 / 5.0;    // displays 2.6  
cout << 91.0 / 7;    // displays 13.0
```

A Closer Look at the % Operator

 % (modulus) operator computes the remainder resulting from integer division

```
cout << 13 % 5;    // displays 3
```

 % requires integers for both operands

```
cout << 13 % 5.0;  // error
```


2.16

- Named Constants

Named Constants

- 🍊 Named constant (constant variable):
variable whose content cannot be changed during program execution
- 🍊 Used for representing constant values with descriptive names:

```
const double TAX_RATE = 0.0675;  
const int NUM_STATES = 50;
```
- 🍊 Always named in uppercase letters

Named Constants in Program 2-28

Program 2-28

```
1 // This program calculates the circumference of a circle.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     // Constants
8     const double PI = 3.14159;
9     const double DIAMETER = 10.0;
10
11     // Variable to hold the circumference
12     double circumference;
13
14     // Calculate the circumference.
15     circumference = PI * DIAMETER;
16
17     // Display the circumference.
18     cout << "The circumference is: " << circumference << endl;
19     return 0;
20 }
```

Program Output

The circumference is: 31.4159

2.17

- Programming Style

Programming Style

- The visual organization of the source code
- Includes the use of spaces, tabs, and blank lines
- Does not affect the syntax of the program
- Affects the readability of the source code