

2.1

The Parts of a C++ Program

The Parts of a C++ Program



```
// sample C++ program ← comment
#include <iostream> ← preprocessor directive
using namespace std; ← which namespace to use
int main() ← beginning of function named main
{ ← beginning of block for main
    cout << "Hello, there!"; ← output statement
    return 0; ← Send 0 to operating system
} ← end of block for main
```

Special Characters



Character	Name	Meaning
//	Double slash	Beginning of a comment
#	Pound sign	Beginning of preprocessor directive
< >	Open/close brackets	Enclose filename in #include
()	Open/close parentheses	Used when naming a function
{ }	Open/close brace	Encloses a group of statements
" "	Open/close quotation marks	Encloses string of characters
;	Semicolon	End of a programming statement

2.2

The `cout` Object

The cout Object



- ❖ Displays output on the computer screen
- ❖ You use the stream insertion operator << to send output to cout:

```
cout << "Programming is fun!";
```

The cout Object



- ❖ Can be used to send more than one item to cout:

```
cout << "Hello " << "there!";
```

Or:

```
cout << "Hello ";
```

```
cout << "there!";
```

The cout Object



- ❖ This produces one line of output:

```
cout << "Programming is ";  
cout << "fun!";
```

The endl Manipulator



- ❖ You can use the **endl** manipulator to start a new line of output. This will produce two lines of output:

```
cout << "Programming is" << endl;  
cout << "fun!";
```


The endl Manipulator



```
cout << "Programming is" << endl;  
cout << "fun!";
```



The `endl` Manipulator



- ❖ You do NOT put quotation marks around `endl`
- ❖ The last character in `endl` is a lowercase L, not the number 1.

`endl` ← This is a lowercase L

The `\n` Escape Sequence



- ❖ You can also use the `\n` escape sequence to start a new line of output. This will produce two lines of output:

```
cout << "Programming is\n";  
cout << "fun!";
```

Notice that the `\n` is INSIDE
the string.

The \n Escape Sequence



```
cout << "Programming is\n";  
cout << "fun!";
```



2.3

The `#include` Directive

The `#include` Directive



- ❖ Inserts the contents of another file into the program
- ❖ This is a preprocessor directive, not part of C++ language
- ❖ `#include` lines not seen by compiler
- ❖ Do not place a semicolon at end of `#include` line

2.4

Variables and Literals

Variables and Literals



- ❖ Variable: a storage location in memory
 - Has a name and a type of data it can hold
 - Must be defined before it can be used:

```
int item;
```

int \textcircled{x} = 5
 $x = 10$

A hand-drawn red table representing memory storage. The first row is labeled 'item' and contains the value '5'. The second row contains the value '2'. The remaining three rows are empty.

item
5
2

Variable Definition in Program 2-7



Program 2-7

```
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 }
```

← Variable Definition

Program Output

The value in number is 5

Literals



- ❖ Literal: a value that is written into a program's code.

"hello, there" (string literal)

12 (integer literal)

Integer Literal in Program 2-9



Program 2-9

```
1 // This program has literals and a variable.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int apples;
8
9     apples = 20;
10    cout << "Today we sold " << apples << " bushels of apples.\n";
11    return 0;
12 }
```

20 is an integer literal

Program Output

Today we sold 20 bushels of apples.

String Literals in Program 2-9



Program 2-9

```
1 // This program has literals and a variable.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int apples;
8
9     apples = 20;
10    cout << "Today we sold " << apples << " bushels of apples.\n";
11    return 0;
12 }
```

These are string literals

Two blue arrows originate from the text "These are string literals". One arrow points to the string literal "Today we sold " which is circled in blue. The other arrow points to the string literal " bushels of apples.\n" which is also circled in blue.

Program Output

Today we sold 20 bushels of apples.

2.5

Identifiers

Identifiers



- ❖ An identifier is a programmer-defined name for some part of a program: variables, functions, etc.

C++ Key Words



Table 2-4 The C++ Key Words

alignas	const	for	private	throw
alignof	constexpr	friend	protected	true
and	const_cast	goto	public	try
and_eq	continue	if	register	typedef
asm	decltype	inline	reinterpret_cast	typeid
auto	default	int	return	typename
bitand	delete	long	short	union
bitor	do	mutable	signed	unsigned
bool	double	namespace	sizeof	using
break	dynamic_cast	new	static	virtual
case	else	noexcept	static_assert	void
catch	enum	not	static_cast	volatile
char	explicit	not_eq	struct	wchar_t
char16_t	export	nullptr	switch	while
char32_t	extern	operator	template	xor
class	false	or	this	xor_eq
compl	float	or_eq	thread_local	

You cannot use any of the C++ key words as an identifier. These words have reserved meaning.

Variable Names



- ❖ A variable name should represent the purpose of the variable. For example:

`itemsOrdered`

The purpose of this variable is to hold the number of items ordered.

Identifier Rules



- ❖ The first character of an identifier must be an alphabetic character or an underscore (_),
- ❖ After the first character you may use alphabetic characters, numbers, or underscore characters.
- ❖ Upper- and lowercase characters are distinct

TaxRate
example

Valid and Invalid Identifiers



IDENTIFIER	VALID?	REASON IF INVALID
totalSales	Yes	
total_Sales	Yes ✓	
total.Sales	No	Cannot contain .
4thQtrSales	No	Cannot begin with digit
totalSale\$	No	Cannot contain \$

Valid and Invalid Identifiers



❖ _boxVolume	❖ yes
❖ 4thShipment	❖ No
❖ \$TaxRate	❖ No
❖ Square!Area	❖ No
❖ Square_Area	❖ Yes
❖ Square-Area	❖ No
❖ Square4Area	❖ Yes

2.6

Integer Data Types

Integer Data Types



- Integer variables can hold whole numbers such as 12, 7, and -99.

Table 2-6 Integer Data Types

Data Type	Typical Size	Typical Range
short int	2 bytes	-232,768 to 32,767
unsigned short int	2 bytes	0 to +65,535
int	4 bytes	-22,147,483,648 to 12,147,483,647
unsigned int	4 bytes	0 to 4,294,967,295
long int	4 bytes	-22,147,483,648 to 12,147,483,647
unsigned long int	4 bytes	0 to 4,294,967,295
long long int	8 bytes	-29,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long int	8 bytes	0 to 18,446,744,073,709,551,615

-32768 to 32767

Defining Variables



- ❖ Variables of the same type can be defined
 - On separate lines:

```
int length;  
int width;  
unsigned int area;
```
 - On the same line:

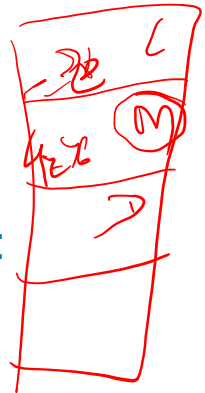
```
int length, width;  
unsigned int area;
```
- ❖ Variables of different types must be in different definitions

Integer Types in Program 2-10



```
1 // This program has variables of several of the integer types.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     int checking;
8     unsigned int miles;
9     long diameter;
10
11     checking = -20;
12     miles = 4276;
13     diameter = 100000;
14     cout << "We have made a long journey of " << miles;
15     cout << " miles.\n";
16     cout << "Our checking account balance is " << checking;
17     cout << "\nThe galaxy is about " << diameter;
18     cout << " light years in diameter.\n";
19     return 0;
20 }
```

This program has three variables:
checking, miles, and diameter



are checking
the balance
of miles
-20
ACCS

Integer Literals



- ❖ An integer literal is an integer value that is typed into a program's code. For example:

```
itemsOrdered = 15;
```

In this code, 15 is an integer literal.

Integer Literals in Program 2-10



1 // This program has variables of several of the integer types.

2 #include <iostream>

3 using namespace std;

4

5 int main()

6 {

7 int checking;

8 unsigned int miles;

9 long diameter;

10

11 checking = -20;

12 miles = 4276;

13 diameter = 100000;

14 cout << "We have made a long journey of " << miles;

15 cout << " miles.\n";

16 cout << "Our checking account balance is " << checking;

17 cout << "\nThe galaxy is about " << diameter;

18 cout << " light years in diameter.\n";

19 return 0;

20 }


Integer Literals

A blue text label "Integer Literals" is positioned to the right of the code. Three blue arrows originate from this label and point to the integer literals in the code: -20 on line 11, 4276 on line 12, and 100000 on line 13. Each of these literals is also circled in blue.

long int var 1234L



Integer Literals

- ❖ Integer literals are stored in memory as ints by default 
- ❖ To store an integer constant in a long memory location, put 'L' at the end of the number: 1234L
- ❖ To store an integer constant in a long long memory location, put 'LL' at the end of the number: 324LL
 $x = 075$
- ❖ Constants that begin with '0' (zero) are base 8: 075
 $x = 75$
- ❖ Constants that begin with '0x' are base 16: 0x75A

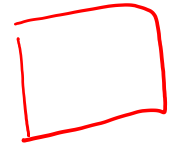
2.7

The `char` Data Type

The char Data Type



- ❖ Used to hold characters or very small integer values
- ❖ Usually 1 byte of memory
- ❖ Numeric value of character from the character set is stored in memory:



CODE:
`char letter;`
`letter = 'C';`

MEMORY:
letter



Character Literals



- ❖ Character literals must be enclosed in single quote marks. Example:

'A'

Character Literals in Program 2-14



Program 2-14

```
1  // This program uses character literals.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      char letter;
8
9      letter = 'A';
10     cout << letter << '\n';
11     letter = 'B';
12     cout << letter << '\n';
13     return 0;
14 }
```



Program Output

A



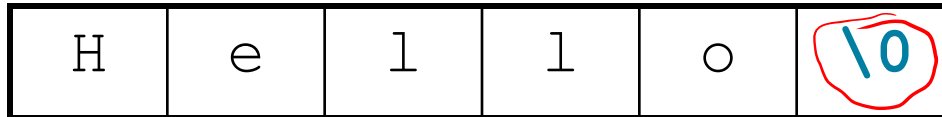
B



Character Strings



- ❖ A series of characters in consecutive memory locations:
"Hello"
- ❖ Stored with the null terminator, `\0`, at the end:
- ❖ Comprised of the characters between the " "



2.8

The C++ `string` Class

The C++ `string` Class



- ❖ Special data type supports working with strings
`#include <string>`

- ❖ Can define `string` variables in programs:
`string firstName, lastName;`

- ❖ Can receive values with assignment operator:
`firstName = "George";`
`lastName = "Washington";`

- ❖ Can be displayed via `cout`
`cout << firstName << " " << lastName;`
`stringTest.cpp`

The string class in Program 2-15



Program 2-15

```
1  // This program demonstrates the string class.
2  #include <iostream>
3  #include <string> // Required for the string class.
4  using namespace std;
5
6  int main()
7  {
8      string movieTitle;
9
10     movieTitle = "Wheels of Fury";
11     cout << "My favorite movie is " << movieTitle << endl;
12     return 0;
13 }
```

Program Output

My favorite movie is Wheels of Fury

2.9

Floating-Point Data Types

Floating-Point Data Types



- The floating-point data types are:
`float`
`double`
`long double`
- 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

Floating-Point Data Types



Table 2-8 Floating-Point Data Types on PCs

Data Type	Key Word	Description
Single precision	float	4 bytes. Numbers between $\pm 3.4\text{E}-38$ and $\pm 3.4\text{E}38$
Double precision	double	8 bytes. Numbers between $\pm 1.7\text{E}-308$ and $\pm 1.7\text{E}308$
Long double precision	long double	8 bytes*. Numbers between $\pm 1.7\text{E}-308$ and $\pm 1.7\text{E}308$

*Some compilers use 10 bytes for long doubles. This allows a range of $\pm 3.4\text{E}-4932$ to $\pm 1.1\text{E}4832$.

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
- ❖ Can be forced to be float (3.14159f) or long double (0.0000625L)

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

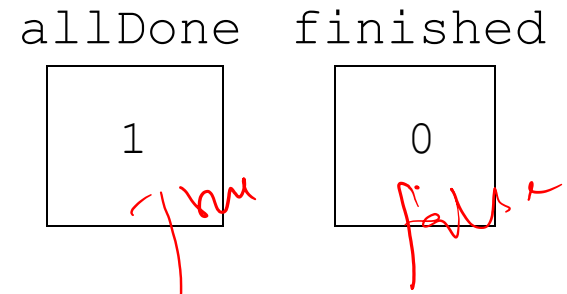


- ❖ Represents values that are true or false
- ❖ bool variables are stored as small integers

- ❖ false is represented by 0, true by 1:

```
bool allDone = true;
```

```
bool finished = false;
```

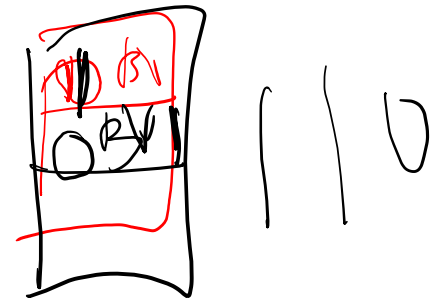


Boolean Variables in Program 2-17



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";
```

2.18

2.12

Variable Assignments and Initialization

Variables



- ❖ In order to store the values, the program needs to declare a symbol called a *variable*, to be used later in a program.
- ❖ They are called variables because their values can be changed.
- ❖ We need to tell the compiler the name of the variable.
- ❖ Choose descriptive names for variables.
 - *radius* for radius
 - *area* for area

Variable Names



- ❖ Descriptive names allow the code to be more readable
- ❖ Which of the following is more descriptive?
double **tr** = 0.0725;
double **salesTaxRate** = 0.0725; ✓

Variables



```
int main()
```

```
{
```

```
    double radius;
```

radius

allocate memory
for radius

no value

```
    // Step 1: Read in radius
```

```
    // Step 2: Compute area
```

```
    // Step 3: Display the area
```

```
}
```


Variables



```
int main()
```

```
{
```

```
    double radius;
```

```
    double area;
```

values

```
    // Step 1: Read in radius
```

```
    // Step 2: Compute area
```

```
    // Step 3: Display the area
```

```
}
```

radius

no value *5*

area

no value

allocate memory
for area

Declaring Variables



- ❖ We need to tell the compiler what type of data a variable can store.
- ❖ Specify the type of data stored in a variable (integers, real numbers (i.e., numbers with a decimal point), characters, and Boolean types).
- ❖ The syntax for declaring a variable is

```
datatype  variableName;
```

Declaring Variables



- **int** count; // Declare count to be an integer variable; 4
- **double** radius; // Declare radius to be a double variable 8
- **char** alpha; // Declare a to be a character variable; ↓

Declaring Variables



- ❖ If variables are of the same type, they can be declared together, as follows:

```
datatype variable1, variable2, ..., variablen;
```

- ❖ For example,

```
int i, j, k;
```

Assignment Statements



- ❖ After a variable is declared, we can assign a value to it by using an *assignment statement*.
- ❖ In C++, the equal sign (=) is used as the *assignment operator*.
- ❖ The syntax for assignment statements is as follows:

Variable = expression;

- ❖ An *expression* represents a computation involving values, variables, and operators that taking them together, evaluates to a value.

Assignment Statements



❖ **int** count;

```
count = 30;
```

// Assign 30 to count;

Handwritten note: A red box containing "30" and "count" written next to it.

❖ **double** radius;

```
radius = 1.5;
```

// Assign 1.5 to radius;

Handwritten note: A red box containing "radius" written above "1.5".

❖ **char** alpha;

```
alpha = 'A';
```

// Assign 'A' to alpha;

Handwritten note: A red box containing "alpha" written above "A".

Assignment Statements



❖ **int** beta;

beta = count + 2;

// Assign addition of count and 2 to beta;

❖ **int** count;

count = count + 1;

// Assign addition of count and 1 to count;



Assignment Statements



- ❖ To assign a value to a variable, you must place the variable name to the left of the assignment operator.

`1 = x;`

Not Correct

Declaring and Initializing in One Step



❖ `int count = 1;`

❖ `double radius = 1.4;`

Note



- ❖ A variable must be declared before it can be assigned a value.
- ❖ A variable declared in a method must be assigned a value before it can be used.

1. Declare the variable
2. Assign a value to the variable
3. Use the variable

Variables



```
int main()
```

```
{
```

```
    double radius = 10.33;
```

```
    // Step 1: Read in radius
```

```
    // Step 2: Compute area
```

```
    // Step 3: Display the area
```

```
}
```

allocate memory for
radius and value is
assigned

radius

10.33

Variable Initialization



- ❖ To initialize a variable means to assign it a value when it is defined:

```
int length = 12;
```

- ❖ Can initialize some or all variables:

```
int length = 12, width = 5, area;
```

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.

Declaring Variables With the auto Key Word



- ❖ C++ 11 introduces an alternative way to define variables, using the `auto` key word and an initialization value. Here is an example:

```
auto amount = 100; ← int
```

- ❖ The `auto` key word tells the compiler to determine the variable's data type from the initialization value.

```
auto interestRate = 12.0; ← double
```

```
auto stockCode = 'D'; ← char
```

```
auto customerNum = 459L; ← long
```

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 }
```

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 (3 operands) $\text{exp1} ? \text{exp2} : \text{exp3}$

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

Arithmetic Operators in Program 2-21



Program 2-21

```
1 // This program calculates hourly wages, including overtime.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     double regularWages,           // To hold regular wages
8           basePayRate = 18.25,     // Base pay rate
9           regularHours = 40.0,     // Hours worked less overtime
10          overtimeWages,           // To hold overtime wages
11          overtimePayRate = 27.78, // Overtime pay rate
12          overtimeHours = 10,      // Overtime hours worked
13          totalWages;              // To hold total wages
14
15     // Calculate the regular wages.
16     regularWages = basePayRate * regularHours;
17
18     // Calculate the overtime wages.
19     overtimeWages = overtimePayRate * overtimeHours;
20
21     // Calculate the total wages.
22     totalWages = regularWages + overtimeWages;
23
24     // Display the total wages.
25     cout << "Wages for this week are $" << totalWages << endl;
26     return 0;
27 }
```

Program Output

Wages for this week are \$1007.8

Sales Prediction



- A East coast division of a company generates 58% of total sales. Write a program that will predict how much the east coast division will make if the company generates \$8.6 million in sales

EC 58% ? \$8.6

Writing a Program



Step 1: Designing An Algorithm

- ❖ The algorithm for calculating the sales of East coast division:
 1. Read in percentage of sales generated by the East Coast.
 2. Read in the predicted amount of sales
 3. Calculate and store the sales prediction for the East Coast sales division by using formula
$$\text{eastCoastSales} = \text{YearlySalesPrediction} * \text{eastCoastPercentage}$$
 4. Display the result.

Sales Tax – Inclass1



- Write a program that will compute the total sales tax on a \$95 purchase. The state sales tax is 4% and county sales tax is 2%

S

Writing a Program

Step 1: Designing An Algorithm

- ❖ The algorithm to write a program that will compute the total sales tax on a \$95 purchase :
 1. Read in state sales tax and county sales tax into constants.
 2. Store the purchase amount.
 3. Calculate and store the amount of state sales tax by
 $\text{stateSalesTax} = \text{purchase} * \text{STATE_SALES_TAX};$
 4. Calculate and store the amount of county sales tax by
 $\text{countySalesTax} = \text{purchase} * \text{COUNTY_SALES_TAX};$
 5. Calculate and store the total amount of sales tax by
 $\text{totalSalesTax} = \text{stateSalesTax} + \text{countySalesTax};$

Retirement Plan



- ❖ Write a program that calculates the amount of pay that will be contributed to a retirement plan if 5%, 7%, or 10% of monthly pay of \$6000 is withheld.

Sale Price



- ❖ Write a program that calculates the sale price of an item that is regularly priced at \$59.95, with a 20 percent discount subtracted.

Integer Division



- ❖ Division can be tricky.
 - ❖ In a C++ program, what is the value of $1/2$?
- ❖ You might think the answer is 0.5...But, that's wrong.
- ❖ The answer is simply 0.
- ❖ Integer division will truncate any decimal remainder.

Integer Division



- ❖ $5 / 2$ yields an integer 2.
- ❖ $5.0 / 2$ yields a double value 2.5.
- ❖ $5 / 2.0$ yields a double value 2.5
- ❖ $5 \% 2$ yields 1 (the remainder of the division)

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
```

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
```

- ❖ **divisionExamples**

THE PARTS OF A C++ PROGRAM - Modulus Operator



- ❖ Normally, the division $17 / 3$ would produce a result of 5.67. This corresponds to a result (quotient) of 5 and a remainder of 2.
 - Reminder: In integer division, the remainder is discarded.
- ❖ The *modulus operator* (%) is a binary operator that returns the remainder after an integer division.
- ❖ Examples:
 - `x = 17 % 3; // x contains 2`
 - `x = 23 % 4; // x contains 3`
- ❖ How could you use this operator to determine if a number is even or odd?

A Closer Look at the % Operator



- ❖ What if we are taking a smaller number modulo a larger number? For example, let's say we want to calculate $5 \% 13$.
 - **Quotient is 0:** $5 / 13 = 0$, since 5 is smaller than 13
 - **Modulus is always the smaller number:** $5 \% 13 = 5$, since that is the remainder after integer division.

- ❖ If we are taking a smaller number modulo a larger number, the quotient will be 0, but the modulus is always the smaller number.

Remainder or Modulus Operator



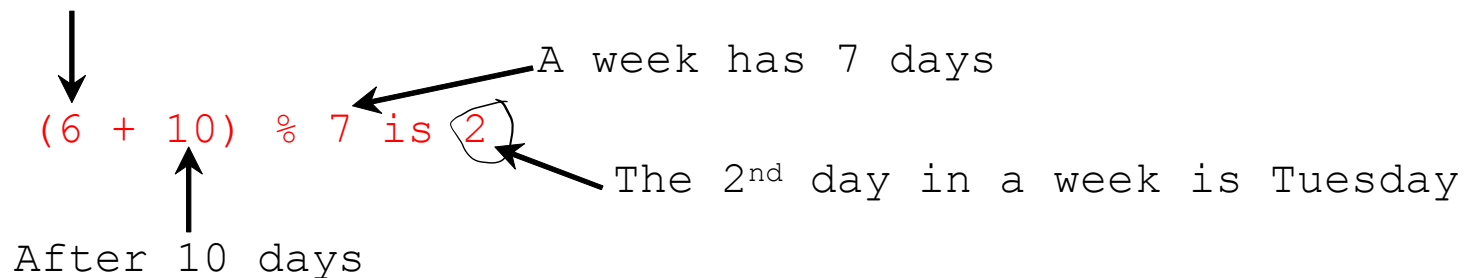
- ❖ Remainder is very useful in programming.
- ❖ For example, an even number $\% 2$ is always 0 and an odd number $\% 2$ is always 1.
- ❖ So you can use this property to determine whether a number is even or odd.

Remainder Operator



- ❖ Suppose today is Saturday and it is 6th day in the week and you, and your friends are going to meet in 10 days. What day is in 10 days?
- ❖ You can find that day is Tuesday using the following expression:

Saturday is the 6th day in a week



Swapping the Digits in an Integer

Write a program that reads a two digit integer and swap its digits to create a new integer. For example, if an integer is 65, after swapping it becomes 56.

Hint: $65 \% 10 = 5$ and $65 / 10 = 6$
 $5 * 10 + 6 = 56$

Writing a Program



Step 1: Designing An Algorithm

- ❖ The algorithm to swap the digits in a two digit number:
 1. Read in number.
 2. Compute the units and tens digits by using division and modulo operators $d1 = \text{num}/10$, $d2 = \text{num} \% 10$
 3. Now compute swapped number as $d2*10 + d1$
 4. Display the result.

Last digit



- ❖ Write a program to extract the rightmost digit of a number.

❖ 2.24

2.15

Comments

Comments



- ❖ Used to document parts of the program
- ❖ Intended for persons reading the source code of the program:
 - Indicate the purpose of the program
 - Describe the use of variables
 - Explain complex sections of code
- ❖ Are ignored by the compiler

Single-Line Comments



- ❖ Begin with `//` through to the end of line:

```
int length = 12; // length in inches
int width = 15;  // width in inches
int area;        // calculated area

// calculate rectangle area
area = length * width;
```

Multi-Line Comments



- ❖ Begin with `/*`, end with `*/`

- ❖ Can span multiple lines:

```
/* this is a multi-line  
   comment  
*/
```

- ❖ Can begin and end on the same line:

```
int area;    /* calculated area */
```

2.16

Named Constants

Named Constants



- ❖ A *named constant* is an identifier that represents a permanent value.
- ❖ The value of a variable may change during the execution of a program, but a *named constant*, or simply *constant*, represents permanent data that never changes.
- ❖ A constant is also known as a *const variable* in C++.

Named Constants



- ❖ The syntax for declaring a constant is as below:

```
const datatype CONSANTNAME = value;
```

- ❖ For example,

```
const double PI = 3.14159;
```

Named Constants



- ❖ Used for representing constant values with descriptive names:

```
const double TAX_RATE = 0.0675;  
const int NUM_STATES = 50;
```

- ❖ Often 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