Chapter 3 Expressions and Interactivity

3.1 The `cin` Object

Concept:

The `cin` object can be used to read data typed at the keyboard.

- Programs are limited when variables are only initialized with built-in data. To change values, the program must be modified and recompiled.

- Most programs are interactive, asking users for data. This allows a program to be run multiple times with different data sets without modification.

- Just as `cout` is the standard output object, `cin` is the standard input object, used for reading input from the keyboard.

Reading Input with cin

Program 3-1

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

int main()
{
    int length, width, area;

    cout << "This program calculates the area of a ";
    cout << "rectangle.\n";
    cout << "What is the length of the rectangle? ";
    cin >> length;
    cout << "What is the width of the rectangle? ";
    cin >> width;
    area = length * width;
    cout << "The area of the rectangle is " << area << ".\n";
    return 0;
}
```

Program Output

- In the program above, the `length` and `width` variables are assigned values entered by the user.

- The statement `cout << "What is the length of the rectangle? ";` is a **prompt**. It directs the user to enter specific data.

- A prompt should always be displayed before using `cin` so the user knows they need to provide input.

- The statement `cin >> length;` uses the **stream extraction operator** (`>>`). It takes characters from the `cin` object (the keyboard) and stores them in the `length` variable.

- Gathering user input is a two-step process:

  1. Use `cout` to display a prompt.
  2. Use `cin` to read a value from the keyboard.

- The `<<` and `>>` operators indicate the direction of data flow.

  - In a `cout` statement, `<<` points toward `cout`, showing data flowing out to the screen.
  - In a `cin` statement, `>>` points toward the variable, showing data flowing from `cin` into the variable.

- `cin` causes the program to wait for the user to type data and press the Enter key.

- `cin` automatically converts keyboard input to the data type of the variable it's being stored in. For example, it converts the characters '1' and '0' to the integer value 10.

- If a user enters a floating-point number for an integer variable, `cin` will only read the portion before the decimal point.

Note:

You must include the `<iostream>` header file in any program that uses `cin`.

Entering Multiple Values

- The `cin` object can gather multiple values in a single statement.

Program 3-2

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

int main()
{
    int length, width, area;

    cout << "This program calculates the area of a ";
    cout << "rectangle.\n";
    cout << "Enter the length and width of the rectangle ";
    cout << "separated by a space.\n";
    cin >> length >> width;
    area = length * width;
    cout << "The area of the rectangle is " << area << endl;
    return 0;
}
```

Program Output

- The statement `cin >> length >> width;` waits for the user to enter two values. The first is stored in `length`, and the second in `width`.

- When entering multiple numbers, the user must separate them with spaces. `cin` uses these spaces to distinguish between values.

Note:

The Enter key is pressed after the last number is entered.

- `cin` can also read multiple values of different data types in one statement.

Program 3-3

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

int main()
{
    int whole;
    double fractional;
    char letter;

    cout << "Enter an integer, a double, and a character: ";
    cin >> whole >> fractional >> letter;
    cout << "Whole: " << whole << endl;
    cout << "Fractional: " << fractional << endl;
    cout << "Letter: " << letter << endl;
    return 0;
}
```

💻 **Program Output**

- When the user types input, the values are first stored in memory in an area called the **keyboard buffer**.

- After the user presses Enter, `cin` reads from this buffer.

- If the user enters values in an order that doesn't match the variables in the `cin` statement, the data may be read incorrectly. It is important that the user enters values in the correct order.

Checkpoint

3.1 What header file must be included in programs using `cin`?

3.2 True or false: `cin` requires the user to press the Enter key when finished entering data.

3.3 Assume `value` is an integer variable. If the user enters 3.14 in response to the following programming statement, what will be stored in `value`?

```cpp
cin >> value;
```

3.14

3

0

Nothing. An error message is displayed.

3.4 A program has the following variable definitions.

```
long miles;
int feet;
float inches;
```

Write one `cin` statement that reads a value into each of these variables.

3.5 The following program will run, but the user will have difficulty understanding what to do. How would you improve the program?

```
#include <iostream>
using namespace std;
int main()
{
   double first, second, product;

   cin >> first >> second;
   product = first * second;
   cout << product;
   return 0;
}
```

3.6 Complete the following program skeleton so it asks for the user's weight (in pounds) and displays the equivalent weight in kilograms.

```
#include <iostream>
using namespace std;
int main()
{
   double pounds, kilograms;


   kilograms = pounds / 2.2;
```

```
    return 0;
}
```

3.2 Mathematical Expressions

Concept:

C++ allows you to construct complex mathematical expressions using multiple operators and grouping symbols.

- An **expression** is a programming statement that has a value.
- It usually consists of an operator and its operands (e.g., `21 + 3`).
- A value by itself (like `3`) or a variable can also be an expression.
- The `cout` object can display the value of any valid C++ expression.

Program 3-4

```

#include <iostream>
using namespace std;

int main()
{
    double numerator, denominator;

    cout << "This program shows the decimal value of ";
    cout << "a fraction.\n";
    cout << "Enter the numerator: ";
    cin >> numerator;
    cout << "Enter the denominator: ";
    cin >> denominator;
    cout << "The decimal value is ";
    cout << (numerator / denominator) << endl;
    return 0;
}
```

Program Output

Note:

The example input for Program 3-4 shows the user entering 3 and 16. Since these values are assigned to `double` variables, they are stored as the `double` values 3.0 and 16.0.

Note:

When sending an expression that consists of an operator to `cout`, it is always a good idea to put parentheses around the expression. Some advanced operators will yield unexpected results otherwise.

Operator Precedence

- When mathematical expressions have multiple operators, their order of evaluation is determined by operator precedence.
- Expressions are generally evaluated from left to right.
- When two operators share an operand, the operator with the highest **precedence** works first.
- Multiplication (`*`), division (`/`), and modulus (`%`) have higher precedence than addition (`+`) and subtraction (`-`).

Table 3-1 Precedence of Arithmetic Operators (Highest to Lowest)

(unary negation) `-`
`* / %`
`+ -`

- Operators with the same precedence (like `*`, `/`, and `%`) are evaluated based on associativity.

Table 3-2 Some Simple Expressions and Their Values

EXPRESSION	VALUE
`5 + 2 * 4`	`13`

EXPRESSION	VALUE
`10 / 2 - 3`	`2`
`8 + 12 * 2 - 4`	`28`
`4 + 17 % 2 - 1`	`4`
`6 - 3 * 2 + 7 - 1`	`6`

Associativity

- An operator's **associativity** is the direction (left-to-right or right-to-left) in which it works.
- If two operators with the same precedence share an operand, they are evaluated according to their associativity.
- Most arithmetic operators have left-to-right associativity.

Table 3-3 Associativity of Arithmetic Operators

OPERATOR	ASSOCIATIVITY
(unary negation) `-`	Right to left
`* / %`	Left to right
`+ -`	Left to right

Grouping with Parentheses

- Parentheses can be used to group parts of an expression, forcing those operations to be performed first, overriding the standard precedence rules.

Table 3-4 More Simple Expressions and Their Values

EXPRESSION	VALUE
`(5 + 2) * 4`	`28`
`10 / (5 - 3)`	`5`
`8 + 12 * (6 - 2)`	`56`

EXPRESSION	VALUE
`(4 + 17) % 2 - 1`	`0`
`(6 - 3) * (2 + 7) / 3`	`9`

Converting Algebraic Expressions to Programming Statements

- Unlike in algebra, C++ requires an explicit operator for all mathematical operations, including multiplication.
- You may need to insert parentheses that are not present in the original algebraic expression to ensure the correct order of operations.

Table 3-5 Algebraic and C++ Multiplication Expressions

ALGEBRAIC EXPRESSION	OPERATION	C++ EQUIVALENT
6*B*	`6 times B`	`6 * B`
(3)(12)	`3 times 12`	`3 * 12`
4*xy*	`4 times x times y`	`4 * x * y`

Table 3-6 Algebraic and C++ Expressions

ALGEBRAIC EXPRESSION	C++ EXPRESSION
$y = 3\frac{x}{2}$	`y = x / 2 * 3;`
$z = 3bc + 4$	`z = 3 * b * c + 4;`
$a = \frac{3x+2}{4a-1}$	`a = (3 * x + 2) / (4 * a - 1)`

No Exponents Please!

- C++ does not have an exponentiation operator (like `^`).

- To raise a number to a power, you must use the `pow` **library function**.

- The `pow` function requires two **arguments**: `pow(base, exponent)`. It returns the value of the base raised to the power of the exponent.

- Using the `pow` function:

- You must include the `<cmath>` header file.
- The arguments passed to `pow` should be `double`s.
- The variable receiving the return value should also be a `double`.

Program 3-5

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

int main()
{
    const double PI = 3.14159;
    double area, radius;

    cout << "This program calculates the area of a circle.\n";
    cout << "What is the radius of the circle? ";
    cin >> radius;
    area = PI * pow(radius, 2.0);
    cout << "The area is " << area << endl;
    return 0;
}
```

Program Output

Note:

Program 3-5 is presented as a demonstration of the `pow` function. In reality, there is no reason to use the `pow` function in such a simple operation. The math statement could just as easily be written as

```
area = PI * radius * radius;
```

The `pow` function is useful, however, in operations that involve larger exponents.

In the Spotlight:

Calculating an Average

- To calculate an average, you add all the values and then divide by the number of values.
- It is a common mistake to forget parentheses when coding this calculation. For example, `average = a + b + c / 3.0;` is incorrect because division happens before addition.
- The correct statement is `average = (a + b + c) / 3.0;`.
- The following program demonstrates the correct way to calculate the average of three test scores.

Program 3-6

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

int main()
{
    double test1, test2, test3;
    double average;

    cout << "Enter the first test score: ";
    cin >> test1;
    cout << "Enter the second test score: ";
    cin >> test2;
    cout << "Enter the third test score: ";
    cin >> test3;

    average = (test1 + test2 + test3) / 3.0;

    cout << "The average score is: " << average << endl;
    return 0;
}
```

Program Output

Checkpoint

3.7 Complete the table below by determining the value of each expression.

EXPRESSION	VALUE
`6 + 3 * 5`	
`12 / 2 - 4`	
`9 + 14 * 2 - 6`	
`5 + 19 % 3 - 1`	
`(6 + 2) * 3`	
`14 / (11 - 4)`	
`9 + 12 * (8 - 3)`	
`(6 + 17) % 2 - 1`	
`(9 - 3) * (6 + 9) / 3`	

3.8 Write C++ expressions for the following algebraic expressions:

$$\begin{aligned} y &= 6x \\ a &= 2b + 4c \\ y &= x^2 \\ g &= \frac{x+2}{z^2} \\ y &= \frac{x^2}{z^2} \end{aligned}$$

3.9 Study the following program and complete the table.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double value1, value2, value3;

    cout << "Enter a number: ";
    cin >> value1;
    value2 = 2 * pow(value1, 2.0);
    value3 = 3 + value2 / 2 - 1;
    cout << value3 << endl;
    return 0;
}
```

IF THE USER ENTERS...	THE PROGRAM WILL DISPLAY WHAT NUMBER (STORED IN `VALUE3`)?
`2`	
`5`	
`4.3`	
`6`	

3.10 Complete the following program skeleton so it displays the volume of a cylindrical fuel tank. The formula for the volume of a cylinder is

- Volume 5 πr^{2h}

where

- π is 3.14159,
- r is the radius of the tank, and
- h is the height of the tank.

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

int main()
{
    double volume, radius, height;
    cout << "This program will tell you the volume of\n";
    cout << "a cylinder-shaped fuel tank.\n";
    cout << "How tall is the tank? ";
    cin >> height;
    cout << "What is the radius of the tank? ";
    cin >> radius;
}
```

3.3 When You Mix Apples and Oranges: Type Conversion

Concept:

When an operator’s operands are of different data types, C++ will automatically convert them to the same data type. This can affect the results of mathematical expressions.

- C++ follows specific rules when performing operations on variables of different data types.
- Data types are ranked based on the size of the numbers they can hold. A `float`, for example, outranks an `int`.

Table 3-7 Data Type Ranking

`long double`
`double`
`float`
`unsigned long long int`
`long long int`
`unsigned long int`
`long int`
`unsigned int`
`int`

- This automatic conversion is known as **type coercion**.

- Converting a value to a higher data type is called **promotion**.

- Converting a value to a lower data type is called **demotion**.

- **Rule 1:** `char`, `short`, and `unsigned short` are automatically promoted to `int` when used in a mathematical expression.

- **Rule 2:** When an operator has two operands of different data types, the lower-ranking value is promoted to the type of the higher-ranking value.

- **Rule 3:** When the final result of an expression is assigned to a variable, the value is converted to the data type of that variable. Be cautious, as this can cause a loss of data (e.g., assigning a `float` to an `int` truncates the fractional part).

Integer Division

- When an integer is divided by another integer, the result is always an integer. Any remainder or fractional part is discarded.
- For a division operation to produce a floating-point result, at least one of the operands must be a floating-point type (e.g., `15.0 / 6`).

3.4 Overflow and Underflow

Concept:

When a variable is assigned a value that is too large or too small in range for that variable's data type, the variable overflows or underflows.

- **Overflow** occurs when a variable is assigned a number that is too large for its data type.
- **Underflow** occurs when a variable is assigned a number that is too small for its data type.
- When an integer overflows, its value typically wraps around to the lowest possible value for its data type, without any error message. This can lead to incorrect program results.

Program 3-7

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

int main()
{
    short testVar = 32767;

    cout << testVar << endl;

    testVar = testVar + 1;
    cout << testVar << endl;

    testVar = testVar - 1;
    cout << testVar << endl;
    return 0;
}
```

Program Output

- The result of floating-point overflow or underflow depends on the compiler configuration. The program might produce an incorrect result, stop with an error, or exhibit other behaviors.

Program 3-8

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

int main()
{
    float test;

    test = 2.0e38 * 1000;
    cout << test << endl;
    test = 2.0e-38 / 2.0e38;
    cout << test << endl;
    return 0;
}
```

3.5 Type Casting

Concept:

Type casting allows you to perform manual data type conversion.

- A **type cast expression** manually converts a value from one data type to another.
- The format is `static_cast<DataType>(Value)`.
- The expression returns a temporary copy of the value, converted to the specified `DataType`. The original variable's value is not changed.
- Type casting is useful for preventing integer division by converting one of the integer operands to a `double` before the division occurs.

Program 3-9

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

int main()
{
    int books;
    int months;
    double perMonth;

    cout << "How many books do you plan to read? ";
    cin >> books;
    cout << "How many months will it take you to read them? ";
    cin >> months;
    perMonth = static_cast<double>(books) / months;
    cout << "That is " << perMonth << " books per month.\n";
    return 0;
}
```

Program Output

Warning!

In Program 3-9, the following statement would still have resulted in integer division:

```
perMonth = static_cast<double>(books / months);
```

The result of the expression `books / months` is 4. When 4 is converted to a `double`, it is 4.0. To prevent the integer division from taking place, one of the operands should be converted to a `double` prior to the division operation. This forces C++ to automatically convert the value of the other operand to a `double`.

- Type casting can also be used to view the integer ASCII code of a `char` as a character, or vice-versa.

Program 3-10

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

int main()
{
    int number = 65;

    cout << number << endl;

    cout << static_cast<char>(number) << endl;
    return 0;
}
```

💻 **Program Output**

Note:

C++ provides several different type cast expressions. `static_cast` is the most commonly used type cast expression, so we will primarily use it in this book.

Checkpoint

3.11 Assume the following variable definitions:

```cpp
int a = 5, b = 12;
double x = 3.4, z = 9.1;
```

What are the values of the following expressions?

`b / a`

`x * a`

`static_cast<double>(b / a)`

`static_cast<double>(b) / a`

`b / static_cast<double>(a)`

`static_cast<double>(b) / static_cast<double>(a)` `b / static_cast<int>(x)`

`static_cast<int>(x) * static_cast<int>(z)`

`static_cast<int>(x * z)`

`static_cast<double>(static_cast<int>(x) * static_cast<int>(z))`

3.12 Complete the following program skeleton so it asks the user to enter a character. Store the character in the variable `letter`. Use a type cast expression with the variable in a `cout` statement to display the character’s ASCII code on the screen.

```
#include <iostream>
using namespace std;
int main()
{
    char letter;

    return 0;
}
```

3.13 What will the following program display?

```
#include <iostream>
using namespace std;
int main()
{
    int integer1, integer2;
    double result;
    integer1 = 19;
    integer2 = 2;
    result = integer1 / integer2;
    cout << result << endl;
    result = static_cast<double>(integer1) / integer2;
    cout << result << endl;
    result = static_cast<double>(integer1 / integer2);
    cout << result << endl;
    return 0;
}
```

3.6 Multiple Assignment and Combined Assignment

Concept:

Multiple assignment means to assign the same value to several variables with one statement.

- C++ allows you to assign a single value to multiple variables in one statement. For example: `a = b = c = d = 12;`.

Combined Assignment Operators

- Statements that modify a variable's own value (e.g., `number = number + 1;`) are very common.
- C++ provides **combined assignment operators** (also called **compound operators**) as a shorthand for these operations.

Table 3-8 (Assume `x` = 6)

STATEMENT	WHAT IT DOES	VALUE OF `x` AFTER THE STATEMENT
`x = x + 4;`	Adds 4 to `x`	10
`x = x - 3;`	Subtracts 3 from `x`	3
`x = x * 10;`	Multiplies `x` by 10	60
`x = x / 2;`	Divides `x` by 2	3
`x = x % 4`	Makes `x` the remainder of `x` / 4	2

Table 3-9 Combined Assignment Operators

OPERATOR	EXAMPLE USAGE	EQUIVALENT TO
`+=`	`x += 5;`	`x = x + 5;`
`-=`	`y -= 2;`	`y = y - 2;`
`*=`	`z *= 10;`	`z = z * 10;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`c %= 3;`	`c = c % 3;`

- These operators are convenient and can make code clearer.

Program 3-11

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

int main()
{
  int begInv,
      sold,
      store1,
      store2,
      store3;

  cout << "One week ago, 3 new widget stores opened\n";
  cout << "at the same time with the same beginning\n";
  cout << "inventory. What was the beginning inventory? ";
  cin >> begInv;

  store1 = store2 = store3 = begInv;

  cout << "How many widgets has store 1 sold? ";
  cin >> sold;
  store1 -= sold;

  cout << "How many widgets has store 2 sold? ";
  cin >> sold;
  store2 -= sold;

  cout << "How many widgets has store 3 sold? ";
  cin >> sold;
  store3 -= sold;

  cout << "\nThe current inventory of each store:\n";
  cout << "Store 1: " << store1 << endl;
  cout << "Store 2: " << store2 << endl;
  cout << "Store 3: " << store3 << endl;
  return 0;
}
```

Program Output

- The precedence of combined assignment operators is lower than that of regular arithmetic operators. For example, `result *= a + 5;` is equivalent to `result = result * (a + 5);`.

Table 3-10 Example Usage of the Combined Assignment Operators

EXAMPLE USAGE	EQUIVALENT TO
`x += b + 5;`	`x = x + (b + 5);`
`y -= a * 2;`	`y = y - (a * 2);`
`z *= 10 - c;`	`z = z * (10 - c);`
`a /= b + c;`	`a = a / (b + c);`
`c %= d - 3;`	`c = c % (d - 3);`

Checkpoint

3.14 Write a multiple assignment statement that assigns 0 to the variables `total`, `subtotal`, `tax`, and `shipping`.

3.15 Write statements using combined assignment operators to perform the following:

Add 6 to `x`.

Subtract 4 from `amount`.

Multiply `y` by 4.

Divide `total` by 27.

Store in `x` the remainder of `x` divided by 7.

Add `y * 5` to `x`.

Subtract `discount` times 4 from `total`.

Multiply `increase` by `salesRep` times 5.

Divide `profit` by `shares` minus 1000.

3.16 What will the following program display?

```cpp
#include <iostream>
using namespace std;
int main()
{
    int unus, duo, tres;

    unus = duo = tres = 5;
    unus += 4;
    duo *= 2;
    tres -= 4;
    unus /= 3;
    duo += tres;
    cout << unus << endl;
    cout << duo << endl;
    cout << tres << endl;
    return 0;
}
```

3.7 Formatting Output

Concept:

The `cout` object provides ways to format data as it is being displayed. This affects the way data appears on the screen.

- The way a value is displayed is called its **formatting**.
- The `cout` object has default formatting, but you can use **stream manipulators** to control how data appears.
- This is useful for aligning output, such as numbers in columns.

Program 3-12

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

int main()
{
    int num1 = 2897, num2 = 5,    num3 = 837,
        num4 = 34,   num5 = 7,    num6 = 1623,
        num7 = 390,  num8 = 3456, num9 = 12;

    cout << num1 << " " << num2 << " " << num3 << endl;

    cout << num4 << " " << num5 << " " << num6 << endl;

    cout << num7 << " " << num8 << " " << num9 << endl;
    return 0;
}
```

Program Output

- The `setw` manipulator establishes a print field of a specified width.
- The value passed to `setw` (e.g., `setw(5)`) specifies the minimum number of character positions, or **field width**, to print the next value in.
- If the value requires fewer characters than the field width, it is padded with spaces. By default, the value is **right-justified** (aligned to the right side of the field).
- `setw` only applies to the very next value being printed. You must use it before each value you want to format.
- If a value is too large to fit in the specified field width, `cout` will print the entire value, overriding the `setw` setting.
- The `<iomanip>` header file is required to use `setw` and other formatting manipulators.

Program 3-13

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

int main()
{
    int num1 = 2897, num2 = 5,    num3 = 837,
        num4 = 34,   num5 = 7,    num6 = 1623,
        num7 = 390,  num8 = 3456,  num9 = 12;

    cout << setw(6) << num1 << setw(6)
         << num2 << setw(6) << num3 << endl;

    cout << setw(6) << num4 << setw(6)
         << num5 << setw(6) << num6 << endl;

    cout << setw(6) << num7 << setw(6)
         << num8 << setw(6) << num9 << endl;
    return 0;
}
```

Program Output

Program 3-14

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

int main()
{
    int intValue = 3928;
    double doubleValue = 91.5;
    string stringValue = "John J. Smith";

    cout << "(" << setw(5) << intValue << ")" << endl;
    cout << "(" << setw(8) << doubleValue << ")" << endl;
    cout << "(" << setw(16) << stringValue << ")" << endl;
    return 0;
}
```

Program Output

The `setprecision` Manipulator

- **Precision** is the total number of significant digits displayed for a floating-point number.
- The `setprecision` manipulator controls this. `cout << setprecision(5);` sets the precision to 5 digits.
- Unlike `setw`, the precision setting remains in effect for all subsequent floating-point output until it is changed again.

Formatting Numbers with `setprecision`

🗐 **Program 3-15**

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

int main()
{
    double quotient, number1 = 132.364, number2 = 26.91;

    quotient = number1 / number2;
    cout << quotient << endl;
    cout << setprecision(5) << quotient << endl;
    cout << setprecision(4) << quotient << endl;
    cout << setprecision(3) << quotient << endl;
    cout << setprecision(2) << quotient << endl;
    cout << setprecision(1) << quotient << endl;
    return 0;
}
```

💻 **Program Output**

Table 3-11 The `setprecision` Manipulator

NUMBER	MANIPULATOR	VALUE DISPLAYED
`28.92786`	`setprecision(3)`	`28.9`
`21`	`setprecision(5)`	`21`
`109.5`	`setprecision(4)`	`109.5`
`34.28596`	`setprecision(2)`	`34`

🗐 Program 3-16

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

int main()
{
    double day1, day2, day3, total;

    cout << "Enter the sales for day 1: ";
    cin >> day1;
    cout << "Enter the sales for day 2: ";
    cin >> day2;
    cout << "Enter the sales for day 3: ";
    cin >> day3;

    total = day1 + day2 + day3;

    cout << "\nSales Amounts\n";
    cout << "-------------\n";
    cout << setprecision(5);
    cout << "Day 1: " << setw(8) << day1 << endl;
    cout << "Day 2: " << setw(8) << day2 << endl;
    cout << "Day 3: " << setw(8) << day3 << endl;
    cout << "Total: " << setw(8) << total << endl;
    return 0;
}
```

💻 Program Output

The `fixed` Manipulator

- By default, `setprecision` can cause large floating-point numbers to be displayed in scientific notation.
- The `fixed` manipulator forces `cout` to display numbers in **fixed-point notation** (decimal).
- When `fixed` and `setprecision(n)` are used together, `n` specifies the number of digits to display *after the decimal point*.

Program 3-17

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

int main()
{
    double day1, day2, day3, total;

    cout << "Enter the sales for day 1: ";
    cin >> day1;
    cout << "Enter the sales for day 2: ";
    cin >> day2;
    cout << "Enter the sales for day 3: ";
    cin >> day3;

    total = day1 + day2 + day3;

    cout << "\nSales Amounts\n";
    cout << "-------------\n";
    cout << setprecision(2) << fixed;
    cout << "Day 1: " << setw(8) << day1 << endl;
    cout << "Day 2: " << setw(8) << day2 << endl;
    cout << "Day 3: " << setw(8) << day3 << endl;
    cout << "Total: " << setw(8) << total << endl;
    return 0;
}
```

💻 **Program Output**

The `showpoint` Manipulator

- By default, `cout` does not display trailing zeros or a decimal point for numbers without a fractional part (e.g., `456.0` is displayed as `456`).
- The `showpoint` manipulator forces `cout` to display the decimal point and trailing zeros, according to the current precision setting.

Note:

`With most compilers, trailing zeros are displayed when the setprecision and fixed manipulators are used together.`

The `left` and `right` Manipulators

- The `left` manipulator causes all subsequent output within a field to be left-justified.
- The `right` manipulator returns to the default right-justification.
- Both `left` and `right` are “sticky” manipulators; they remain in effect until the other is used.

Table 3-12 Stream Manipulators

STREAM MANIPULATOR	DESCRIPTION

STREAM MANIPULATOR	DESCRIPTION
`setw(`*n*`)`	Establishes a print field of *n* spaces.
`fixed`	Displays floating-point numbers in fixed-point notation.
`showpoint`	Causes a decimal point and trailing zeros to be displayed, even if there is no fractional part.
`setprecision(` *n* `)`	Sets the precision of floating-point numbers.
`left`	Causes subsequent output to be left-justified.
`right`	Causes subsequent output to be right-justified.

Checkpoint

3.17 Write `cout` statements with stream manipulators that perform the following:

Display the number 34.789 in a field of nine spaces with two decimal places of precision.

Display the number 7.0 in a field of five spaces with three decimal places of precision.

The decimal point and any trailing zeros should be displayed.

Display the number 5.789e`+`12 in fixed-point notation.

Display the number 67 left-justified in a field of seven spaces.

3.18 The following program will not compile because the lines have been mixed up:

```
#include <iomanip>
}
cout << person << endl;
string person = "Wolfgang Smith";
int main()
cout << person << endl;
{
```

```
#include <iostream>
return 0;
cout << left;
using namespace std;
cout << setw(20);
cout << right;
```

When the lines are properly arranged, the program should display the following:

```
    Wolfgang Smith
Wolfgang Smith
```

Rearrange the lines in the correct order. Test the program by entering it on the computer, compiling it, and running it.

3.19 The following program skeleton asks for an angle in degrees and converts it to radians. The formatting of the final output is left to you.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    const double PI = 3.14159;
    double degrees, radians;

    cout << "Enter an angle in degrees and I will convert it\n";
    cout << "to radians for you: ";
    cin >> degrees;
    radians = degrees * PI / 180;
    return 0;
}
```

3.8 Working with Characters and `string` Objects

Concept:

Special functions exist for working with characters and `string` objects.

- Using `cin` with the `>>` operator to read string input can be problematic.
- `cin >>` ignores leading **whitespace** (spaces, tabs, newlines) and stops reading at the next whitespace character.
- This means it can only read single words, not entire sentences or names with spaces.

Program 3-18

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

int main()
{
    string name;
    string city;

    cout << "Please enter your name: ";
    cin >> name;
    cout << "Enter the city you live in: ";
    cin >> city;

    cout << "Hello, " << name << endl;
    cout << "You live in " << city << endl;
    return 0;
}
```

Program Output

- To read an entire line of input, including spaces, use the `getline` function.
- The format is `getline(cin, stringVariable);`.

Program 3-19

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

int main()
{
    string name;
    string city;

    cout << "Please enter your name: ";
    getline(cin, name);
    cout << "Enter the city you live in: ";
    getline(cin, city);

    cout << "Hello, " << name << endl;
    cout << "You live in " << city << endl;
    return 0;
}
```

Program Output

Inputting a Character

- The simplest way to read a single character is with `cin >>`.

Program 3-20

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

int main()
{
    char ch;

    cout << "Type a character and press Enter: ";
    cin >> ch;
    cout << "You entered " << ch << endl;
    return 0;
}
```

Program Output

Using `cin.get`

- Because `cin >>` skips whitespace, it cannot be used to read a space or the press of the Enter key.
- The `cin` object has a **member function** called `get` that reads a single character, including whitespace.
- It can be called in several ways:
  - `cin.get(ch);` // Reads a character and stores it in ch.
  - `ch = cin.get();` // Reads a character and returns it to be stored in ch.
  - `cin.get();` // Reads a character but does not store it (useful for pausing).

Program 3-21

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

int main()
{
    char ch;

    cout << "This program has paused. Press Enter to continue.";
    cin.get(ch);
    cout << "It has paused a second time. Please press Enter again.";
    ch = cin.get();
    cout << "It has paused a third time. Please press Enter again.";
    cin.get();
    cout << "Thank you!";
    return 0;
}
```

Program Output

Mixing `cin >>` and `cin.get`

- Mixing `cin >>` with `cin.get` can cause problems.
- When `cin >> number;` reads an integer, the user types the number and presses Enter. `cin` reads the number but leaves the newline character (`\n`) in the keyboard buffer.

- A subsequent call to `cin.get()` will immediately read that leftover newline character, skipping over any intended user input.

Program 3-22

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

int main()
{
    char ch;
    int number;

    cout << "Enter a number: ";
    cin >> number;
    cout << "Enter a character: ";
    ch = cin.get();
    cout << "Thank You!\n";
    return 0;
}
```

Program Output

Using `cin.ignore`

- The `cin.ignore()` member function can be used to solve this problem.
- It tells `cin` to skip (ignore) characters in the keyboard buffer.
- `cin.ignore();` skips the next single character.
- `cin.ignore(n, c);` skips `n` characters or until character `c` is found, whichever comes first.
- Placing `cin.ignore();` after a `cin >>` statement will consume the leftover newline character, allowing subsequent `cin.get()` or `getline` calls to work correctly.

Program 3-23

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

```

```
int main()
{
    char ch;
    int number;

    cout << "Enter a number: ";
    cin >> number;
    cin.ignore();
    cout << "Enter a character: ";
    ch = cin.get();
    cout << "Thank You!\n";
    return 0;
}
```

Program Output

`string` Member Functions and Operators

- `string` objects have member functions, such as `length()`, which returns the number of characters in the string.
- The `+` operator, when used with strings, performs **concatenation** (joins them together).
- The `+=` combined assignment operator can also be used for string concatenation.

3.9 More Mathematical Library Functions

Concept:

The C++ runtime library provides several functions for performing complex mathematical operations.

- The `<cmath>` header file provides access to numerous mathematical functions in addition to `pow`.
- These functions are useful for scientific and specialized programs.

Table 3-13 `<cmath>` Library Functions

FUNCTION	EXAMPLE	DESCRIPTION

FUNCTION	EXAMPLE	DESCRIPTION
`abs`	`y = abs(x);`	Returns the absolute value of the argument. The argument and the return value are integers.
`cos`	`y = cos(x);`	Returns the cosine of the argument. The argument should be an angle expressed in radians. The return type and the argument are `double`s.
`exp`	`y = exp(x);`	Computes the exponential function of the argument, which is `x`. The return type and the argument are `double`s.
`fmod`	`y = fmod(x, z);`	Returns, as a `double`, the remainder of the first argument divided by the second argument. Works like the modulus operator, but the arguments are `double`s. (The modulus operator only works with integers.) Take care not to pass zero as the second argument. Doing so would cause division by zero.
`log`	`y = log(x);`	Returns the natural logarithm of the argument. The return type and the argument are `double`s.
`log10`	`y = log10(x);`	Returns the base-10 logarithm of the argument.

FUNCTION	EXAMPLE	DESCRIPTION
		The return type and the argument are `double`s.
round	y = round(x)	The argument, x, can be a double, a float, or a long double. Returns the value of x rounded to the nearest whole number. For example, if x is 2.8, the function returns 3.0, or if x is 2.1, the function returns 2.0. The return type is the same as the type of the argument.
`sin`	`y = sin(x);`	Returns the sine of the argument. The argument should be an angle expressed in radians. The return type and the argument are `double`s.
`sqrt`	`y = sqrt(x);`	Returns the square root of the argument. The return type and argument are `double`s.
`tan`	`y = tan(x);`	Returns the tangent of the argument. The argument should be an angle expressed in radians. The return type and the argument are `double`s.

- Functions can be nested; the return value of one function can be used as an argument to another.

Program 3-24

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

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

    cout << "Enter the length of side a: ";
    cin >> a;
    cout << "Enter the length of side b: ";
    cin >> b;
    c = sqrt(pow(a, 2.0) + pow(b, 2.0));
    cout << "The length of the hypotenuse is ";
    cout << setprecision(2) << c << endl;
    return 0;
}
```

Program Output

Random Numbers

- Random numbers are useful in games, simulations, statistics, and computer security.
- The `rand()` function (in `<cstdlib>`) returns a pseudorandom integer. “Pseudorandom” means the function produces the same sequence of numbers every time the program runs.
- To get different random numbers on each run, you must “seed” the random number generator using the `srand()` function.
- A common practice is to seed `srand()` with the current time using `srand(time(0));`. This requires the `<ctime>` header file.

Program 3-25

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
```

```
using namespace std;

int main()
{
    unsigned seed = time(0);

    srand(seed);

    cout << rand() << endl;
    cout << rand() << endl;
    cout << rand() << endl;
    return 0;
}
```

Program Output

- To limit a random number to a specific range, use the modulus operator and addition. The formula is:
- `y = (rand() % (maxValue - minValue + 1)) + minValue;`

In the Spotlight:

Using Random Numbers

- This section provides a practical example of generating random numbers to simulate the rolling of two dice, each with a value from 1 to 6. The program uses the range-limiting formula with `minValue = 1` and `maxValue = 6`.

Program 3-26

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

int main()
{
    const int MIN_VALUE = 1;
    const int MAX_VALUE = 6;

    int die1;
    int die2;

    unsigned seed = time(0);

    srand(seed);

    cout << "Rolling the dice...\n";
    die1 = (rand() % (MAX_VALUE − MIN_VALUE + 1)) + MIN_VALUE;
    die2 = (rand() % (MAX_VALUE − MIN_VALUE + 1)) + MIN_VALUE;
    cout << die1 << endl;
    cout << die2 << endl;
    return 0;
}
```

💻 **Program Output**

💻 **Program Output**

💻 **Program Output**

Checkpoint

3.20 Write a short description of each of the following functions:

```
cos     log     sin
exp     log10   sqrt
fmod    pow     tan
```

3.21 Assume the variables `angle1` and `angle2` hold angles stored in radians. Write a statement that adds the sine of `angle1` to the cosine of `angle2` and stores the result in the

variable `x`.
3.22 To find the cube root (the third root) of a number, raise it to the power of ⅓. To find the fourth root of a number, raise it to the power of ¼. Write a statement that will find the fifth root of the variable `x` and store the result in the variable `y`.

3.23 The cosecant of the angle *a* is

$$\frac{1}{\sin a}$$

Write a statement that calculates the cosecant of the angle stored in the variable `a`, and stores it in the variable `y`.

3.10 Focus on Debugging: Hand Tracing a Program

- **Hand tracing** is a debugging method where you execute a program on paper, pretending to be the computer.
- You step through the code statement by statement.
- You create a chart with a column for each variable and record its value after each statement executes.
- This process is effective for finding mathematical mistakes and other logic errors that might not be obvious from just reading the code.
- By tracking the state of each variable, you can pinpoint the exact line where an error occurs.

Program 3-27

```cpp
#include <iostream>
using namespace std;
int main()
{
    double num1, num2, num3, avg;
    cout << "Enter the first number: ";
    cin >> num1;
    cout << "Enter the second number: ";
    cin >> num2;
    cout << "Enter the third number: ";
    cin >> num3;
    avg = num1 + num2 + num3 / 3;
    cout << "The average is " << avg << endl;
    return 0;
}
```

NUM1	NUM2	NUM3	AVG

Program Output

Program 3-27 (with hand trace chart filled)

```
#include <iostream>
using namespace std;
int main()
{
    double num1, num2, num3, avg;
    cout << "Enter the first number: ";
    cin >> num1;
    cout << "Enter the second number: ";
    cin >> num2;
    cout << "Enter the third number: ";
    cin >> num3;
    avg = num1 + num2 + num3 / 3;
    cout << "The average is " << avg << endl;
    return 0;
}
```

NUM1	NUM2	NUM3	AVG
?	?	?	?

NUM1	NUM2	NUM3	AVG
?	?	?	?
10	?	?	?
10	?	?	?
10	20	?	?
10	20	?	?
10	20	30	?
10	20	30	40
10	20	30	40

3.11 Focus on Problem Solving: A Case Study

- This case study involves writing a program for General Crates, Inc.
- The program must calculate the volume, cost, customer price, and profit for a custom wooden crate based on user-provided dimensions.
- The cost to build is $0.23 per cubic foot.
- The charge to the customer is $0.50 per cubic foot.

Variables

Table 3-14 Named Constants and Variables

CONSTANT OR VARIABLE	DESCRIPTION
`COST_PER_CUBIC_FOOT`	A named constant, declared as a `double` and initialized with the value 0.23. This represents the cost to build a crate, per cubic foot.
`CHARGE_PER_CUBIC_FOOT`	A named constant, declared as a `double` and initialized with the value 0.5. This

CONSTANT OR VARIABLE	DESCRIPTION
	represents the amount charged for a crate, per cubic foot.
`length`	A `double` variable to hold the length of the crate, which is input by the user.
`width`	A `double` variable to hold the width of the crate, which is input by the user.
`height`	A `double` variable to hold the height of the crate, which is input by the user.
`volume`	A `double` variable to hold the volume of the crate. The value stored in this variable is calculated.
`cost`	A `double` variable to hold the cost of building the crate. The value stored in this variable is calculated.
`charge`	A `double` variable to hold the amount charged to the customer for the crate. The value stored in this variable is calculated.
`profit`	A `double` variable to hold the profit GCI makes from the crate. The value stored in this variable is calculated.

Program Design

- The program follows three general steps:

  1. Ask the user for the crate’s dimensions (length, width, height).
  2. Calculate the volume, cost, charge, and profit.
  3. Display the calculated data.

- The design can be broken down using hierarchy charts.

- The program logic in pseudocode:

```
Ask the user to input the crate's length.
Ask the user to input the crate's width.
Ask the user to input the crate's height.
Calculate the crate's volume.
Calculate the cost of building the crate.
Calculate the customer's charge for the crate.
Calculate the profit made from the crate.
Display the crate's volume.
Display the cost of building the crate.
Display the customer's charge for the crate.
Display the profit made from the crate.
```

Calculations

- The formulas used in the program are:
    - `volume = length * width * height`
    - `cost = volume * 0.23`
    - `charge = volume * 0.5`
    - `profit = charge - cost`

The Program

- The final step is to write the C++ code based on the design and calculations.

Program 3-28

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

int main()
{
    const double COST_PER_CUBIC_FOOT = 0.23;
    const double CHARGE_PER_CUBIC_FOOT = 0.5;

    double length,
           width,
           height,
           volume,
           cost,
           charge,
           profit;

    cout << setprecision(2) << fixed << showpoint;

    cout << "Enter the dimensions of the crate (in feet):\n";
    cout << "Length: ";
    cin >> length;
    cout << "Width: ";
    cin >> width;
    cout << "Height: ";
    cin >> height;

    volume = length * width * height;
    cost = volume * COST_PER_CUBIC_FOOT;
    charge = volume * CHARGE_PER_CUBIC_FOOT;
    profit = charge -  cost;

    cout << "The volume of the crate is ";
    cout << volume << " cubic feet.\n";
    cout << "Cost to build: $" << cost << endl;
    cout << "Charge to customer: $" << charge << endl;
    cout << "Profit: $" << profit << endl;
    return 0;
}
```

Program Output

Program Output

Review Questions and Exercises

Short Answer

1. Assume the following variables are defined:

   ```
   int age;
   double pay;
   char section;
   ```

   Write a single `cin` statement that will read input into each of these variables.

2. Assume a `string` object has been defined as follows:

   ```
   string description;
   ```

   1. Write a `cin` statement that reads in a one-word string.

   2. Write a statement that reads in a string that can contain multiple words separated by blanks.

3. What header files must be included in the following program?

   ```
   int main()
   {
       double amount = 89.7;
       cout << showpoint << fixed;
       cout << setw(8) << amount << endl;
       return 0;
   }
   ```

4. Complete the following table by determining the value of each expression.

EXPRESSION	VALUE
`28 / 4 - 2`	
`6 + 12 * 2 - 8`	
`4 + 8 * 2`	
`6 + 17 % 3 - 2`	

EXPRESSION	VALUE
`2 + 22 * (9 - 7)`	
`(8 + 7) * 2`	
`(16 + 7) % 2 - 1`	
`12 / (10 - 6)`	
`(19 - 3) * (2 + 2) / 4`	

5. Write C++ expressions for the following algebraic expressions:

$$\begin{aligned} a &= 12x \\ z &= 5x + 14y + 6k \\ y &= x^4 \\ g &= \frac{h+12}{4k} \\ c &= \frac{a^3}{b^2k^4} \end{aligned}$$

6. Assume a program has the following variable definitions:

```
int units;
float mass;
double weight;
```

and the following statement:

```
weight = mass * units;
```

Which automatic data type conversion will take place?

   1. `mass` is demoted to an `int`, `units` remains an `int`, and the result of `mass * units` is an `int`.
   2. `units` is promoted to a `float`, `mass` remains a `float`, and the result of `mass * units` is a `float`.
   3. `units` is promoted to a `float`, `mass` remains a `float`, and the result of `mass * units` is a `double`.

7. Assume a program has the following variable definitions:

```
int a, b = 2;
float c = 4.2;
```

and the following statement:

```
a = b * c;
```

What value will be stored in `a`?

8.4

8

0

None of the above

8. Assume `qty` and `salesReps` are both integers. Use a type cast expression to rewrite the following statement so it will no longer perform integer division.

```
unitsEach = qty / salesReps;
```

9. Rewrite the following variable definition so that the variable is a named constant.

```
int rate;
```

10. Complete the following table by providing statements with combined assignment operators for the right-hand column. The statements should be equivalent to the statements in the left-hand column.

STATEMENTS WITH ASSIGNMENT OPERATOR	STATEMENTS WITH COMBINED ASSIGNMENT OPERATOR
`x = x + 5;`	
`total = total + subtotal;`	
`dist = dist / rep;`	
`ppl = ppl * period;`	
`inv = inv - shrinkage;`	

STATEMENTS WITH ASSIGNMENT OPERATOR	STATEMENTS WITH COMBINED ASSIGNMENT OPERATOR
`num = num % 2;`	

11. Write a multiple assignment statement that can be used instead of the following group of assignment statements:

```
east = 1;
west = 1;
north = 1;
south = 1;
```

12. Write a `cout` statement so the variable `divSales` is displayed in a field of 8 spaces, in fixed-point notation, with a precision of 2 decimal places. The decimal point should always be displayed.

13. Write a `cout` statement so the variable `totalAge` is displayed in a field of 12 spaces, in fixed-point notation, with a precision of 4 decimal places.

14. Write a `cout` statement so the variable `population` is displayed in a field of 12 spaces, left-justified, with a precision of 8 decimal places. The decimal point should always be displayed.

Fill-in-the-Blank

1. The ________ library function returns the cosine of an angle.

2. The ________ library function returns the sine of an angle.

3. The ________ library function returns the tangent of an angle.

4. The ________ library function returns the exponential function of a number.

5. The ________ library function returns the remainder of a floating-point division.

6. The ________ library function returns the natural logarithm of a number.

7. The ________ library function returns the base-10 logarithm of a number.

8. The ________ library function returns the value of a number raised to a power.

9. The ________ library function returns the square root of a number.

10. The ________ file must be included in a program that uses the mathematical functions.

Algorithm Workbench

1. A retail store grants its customers a maximum amount of credit. Each customer's available credit is his or her maximum amount of credit minus the amount of credit used. Write a pseudocode algorithm for a program that asks for a customer's maximum amount of credit and amount of credit used. The program should then display the customer's available credit.

   After you write the pseudocode algorithm, convert it to a complete C++ program.

2. Write a pseudocode algorithm for a program that calculates the total of a retail sale. The program should ask for the amount of the sale and the sales tax rate. The sales tax rate should be entered as a floating-point number. For example, if the sales tax rate is 6 percent, the user should enter 0.06. The program should display the amount of sales tax and the total of the sale.

   After you write the pseudocode algorithm, convert it to a complete C++ program.

3. Write a pseudocode algorithm for a program that asks the user to enter a golfer's score for three games of golf, and then displays the average of the three scores.

   After you write the pseudocode algorithm, convert it to a complete C++ program.

Find the Errors

Each of the following programs has some errors. Locate as many as you can.

1.
```
using namespace std;
int main ()
{
    double number1, number2, sum;
    Cout << "Enter a number: ";
    Cin << number1;
    Cout << "Enter another number: ";
    Cin << number2;
    number1 + number2 = sum;
    Cout "The sum of the two numbers is " << sum
    return 0;
}
```

2.
```
#include <iostream>
using namespace std;
int main()
{
    int number1, number2;
    float quotient;
    cout << "Enter two numbers and I will divide\n";
    cout << "the first by the second for you.\n";
    cin >> number1, number2;
    quotient = float<static_cast>(number1) / number2;
    cout << quotient
    return 0;
}
```

3.
```
#include <iostream>;
using namespace std;
int main()
{
    const int number1, number2, product;
    cout << "Enter two numbers and I will multiply\n";
    cout << "them for you.\n";
    cin >> number1 >> number2;
    product = number1 * number2;
    cout << product
    return 0;
}
```

4.
```
#include <iostream>;
using namespace std;
main
{
    int number1, number2;
    cout << "Enter two numbers and I will multiply\n"
    cout << "them by 50 for you.\n"
    cin >> number1 >> number2;
    number1 =* 50;
    number2 =* 50;
    cout << number1 << " " << number2;
    return 0;
}
```

5.
```
#include <iostream>;
using namespace std;
main
{
    double number, half;
    cout << "Enter a number and I will divide it\n"
    cout << "in half for you.\n"
    cin >> number1;
    half =/ 2;
    cout << fixedpoint << showpoint << half << endl;
    return 0;
}
```

6.
```
#include <iostream>;
using namespace std;
int main()
{
    char name, go;
    cout << "Enter your name: ";
    getline >> name;
    cout << "Hi " << name << endl;
    return 0;
}
```

Predict the Output

What will each of the following programs display? (Some should be hand traced and require a calculator.)

1. (*Assume the user enters 38700. Use a calculator.*)

```
#include <iostream>
using namespace std;
int main()
{
    double salary, monthly;
    cout << "What is your annual salary? ";
    cin >> salary;
    monthly = static_cast<int>(salary) / 12;
    cout << "Your monthly wages are " << monthly << endl;
    return 0;
}
```

2.

```
#include <iostream>
using namespace std;
int main()
{
    long x, y, z;

    x = y = z = 4;
    x += 2;
    y -= 1;
    z *= 3;
    cout << x << " " << y << " " << z << endl;
    return 0;
}
```

3. (*Assume the user enters George Washington.*)

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
int main()
{
    string userInput;
    cout << "What is your name? ";
    getline(cin, userInput);
    cout << "Hello " << userInput << endl;
    return 0;
}
```

4. (*Assume the user enters 36720152. Use a calculator.*)

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    long seconds;
    double minutes, hours, days, months, years;

    cout << "Enter the number of seconds that have\n";
    cout << "elapsed since some time in the past and\n";
    cout << "I will tell you how many minutes, hours,\n";
    cout << "days, months, and years have passed: ";
    cin >> seconds;
    minutes = seconds / 60;
    hours = minutes / 60;
    days = hours / 24;
    years = days / 365;
    months = years * 12;
    cout << setprecision(4) << fixed << showpoint << right;
    cout << "Minutes: " << setw(6) << minutes << endl;
    cout << "Hours: " << setw(6) << hours << endl;
    cout << "Days: " << setw(6) << days << endl;
    cout << "Months: " << setw(6) << months << endl;
    cout << "Years: " << setw(6) << years << endl;
    return 0;
}
```

Programming Challenges

1. Miles per Gallon

   Write a program that calculates a car’s gas mileage. The program should ask the user to enter the number of gallons of gas the car can hold, and the number of miles it can be driven on a full tank. It should then display the number of miles that may be driven per gallon of gas.

   VideoNote

   Solving the Stadium Seating Problem

2. Stadium Seating

   There are three seating categories at a stadium. For a softball game, Class A seats cost $15, Class B seats cost $12, and Class C seats cost $9. Write a program that asks how many tickets for each class of seats were sold, then displays the amount of income generated from ticket sales. Format your dollar amount in fixed-point notation, with two decimal places of precision, and be sure the decimal point is always displayed.

3. Test Average

   Write a program that asks for five test scores. The program should calculate the average test score and display it. The number displayed should be formatted in fixed-point notation, with one decimal point of precision.

4. Average Rainfall

   Write a program that calculates the average rainfall for three months. The program should ask the user to enter the name of each month, such as June or July, and the amount of rain (in inches) that fell each month. The program should display a message similar to the following:

   The average rainfall for June, July, and August is 6.72 inches.

5. Male and Female Percentages

   Write a program that asks the user for the number of males and the number of females registered in a class. The program should display the percentage of males and females in the class.

   Hint: Suppose there are 8 males and 12 females in a class. There are 20 students in the class. The percentage of males can be calculated as 8 ÷ 20 = 0.4, or 40 percent. The percentage of females can be calculated as 12 ÷ 20 = 0.6, or 60 percent.

6. Ingredient Adjuster

   A cookie recipe calls for the following ingredients:

   - 1.5 cups of sugar
   - 1 cup of butter
   - 2.75 cups of flour

   The recipe produces 48 cookies with this amount of the ingredients. Write a program that asks the user how many cookies he or she wants to make, then displays the number of cups of each ingredient needed for the specified number of cookies.

7. Box Office

   A movie theater only keeps a percentage of the revenue earned from ticket sales. The remainder goes to the movie distributor. Write a program that calculates a theater’s gross and net box office profit for a night. The program should ask for the name of the movie, and how many adult and child tickets were sold. (The price of an adult ticket is $10.00 and a child’s ticket is $6.00.) It should display a report similar to:

MOVIE NAME:	**“WHEELS OF FURY”**
Adult Tickets Sold:	382
Child Tickets Sold:	127
Gross Box Office Profit:	$ 4,582.00
Net Box Office Profit:	$ 916.40
Amount Paid to Distributor:	$ 3,665.60

Note:

Assume the theater keeps 20 percent of the gross box office profit.

8. How Many Widgets?

   The Yukon Widget Company manufactures widgets that weigh 12.5 pounds each. Write a program that calculates how many widgets are stacked on a pallet, based on the total weight of the pallet. The program should ask the user how much the pallet weighs by itself and with the widgets stacked on it. It should then calculate and display the number of widgets stacked on the pallet.

9. How Many Calories?

   A bag of cookies holds 30 cookies. The calorie information on the bag claims there are 10 “servings” in the bag and that a serving equals 300 calories. Write a program that asks the user to input how many cookies he or she actually ate, then reports how many total calories were consumed.

10. How Much Insurance?

    Many financial experts advise that property owners should insure their homes or buildings for at least 80 percent of the amount it would cost to replace the structure. Write a program that asks the user to enter the replacement cost of a building, then displays the minimum amount of insurance he or she should buy for the property.

11. Automobile Costs

Write a program that asks the user to enter the monthly costs for the following expenses incurred from operating his or her automobile: loan payment, insurance, gas, oil, tires, and maintenance. The program should then display the total monthly cost of these expenses, and the total annual cost of these expenses.

12. Celsius to Fahrenheit

    Write a program that converts Celsius temperatures to Fahrenheit temperatures. The formula is

    $$F = \frac{9}{5}C + 32$$

    F is the Fahrenheit temperature, and *C* is the Celsius temperature.

13. Currency

    Write a program that will convert U.S. dollar amounts to Japanese yen and to euros, storing the conversion factors in the constants `YEN_PER_DOLLAR` and `EUROS_PER_DOLLAR`. To get the most up-to-date exchange rates, search the Internet using the term "currency exchange rate". If you cannot find the most recent exchange rates, use the following:

    - 1 Dollar = 98.93 Yen
    - 1 Dollar = 0.74 Euros

    Format your currency amounts in fixed-point notation, with two decimal places of precision, and be sure the decimal point is always displayed.

14. Monthly Sales Tax

    A retail company must file a monthly sales tax report listing the sales for the month and the amount of sales tax collected. Write a program that asks for the month, the year, and the total amount collected at the cash register (i.e. sales plus sales tax). Assume the state sales tax is 4 percent, and the county sales tax is 2 percent.

    If the total amount collected is known and the total sales tax is 6 percent, the amount of product sales may be calculated as:

    $$S = \frac{T}{1.06}$$

    S is the product sales and *T* is the total income (product sales plus sales tax).

    The program should display a report similar to:

```
Month: October
--------------------
Total Collected:     $ 26572.89
Sales:               $ 25068.76
County Sales Tax:    $   501.38
State Sales Tax:     $  1002.75
Total Sales Tax:     $  1504.13
```

15. Property Tax

    A county collects property taxes on the assessment value of property, which is 60 percent of the property’s actual value. If an acre of land is valued at $10,000, its assessment value is $6,000. The property tax is then 75¢ for each $100 of the assessment value. The tax for the acre assessed at $6,000 will be $45. Write a program that asks for the actual value of a piece of property, then displays the assessment value and property tax.

16. Senior Citizen Property Tax

    Madison County provides a $5,000 homeowner exemption for its senior citizens. For example, if a senior’s house is valued at $158,000, its assessed value would be $94,800, as explained above. However, he would only pay tax on $89,800. At last year’s tax rate of $2.64 for each $100 of assessed value, the property tax would be $2,370.72. In addition to the tax break, senior citizens are allowed to pay their property tax in four equal payments. The quarterly payment due on this property would be $592.68. Write a program that asks the user to input the actual value of a piece of property and the current tax rate for each $100 of assessed value. The program should then calculate and report how much annual property tax a senior homeowner will be charged for this property, and what the quarterly tax bill will be.

17. Math Tutor

    Write a program that can be used as a math tutor for a young student. The program should display two random numbers to be added, such as

    $$\begin{array}{r} 247 \\ +\underline{129} \end{array}$$

    The program should then pause while the student works on the problem. When the student is ready to check the answer, he or she can press a key and the program will display the correct solution:

$$\begin{array}{r} 247 \\ +129 \\ \hline 376 \end{array}$$

18. Interest Earned

Assuming there are no deposits other than the original investment, the balance in a savings account after one year may be calculated as

$$\text{Amount} = \text{Principal} \times \left(1 + \frac{\text{Rate}}{\text{T}}\right)^{\text{T}}$$

`Principal` is the balance in the savings account, `Rate` is the interest rate, and `T` is the number of times the interest is compounded during a year (`T` is 4 if the interest is compounded quarterly).

Write a program that asks for the principal, the interest rate, and the number of times the interest is compounded. It should display a report similar to:

```
Interest Rate:          4.25%
Times Compounded:         12
Principal:           $ 1000.00
Interest:            $   43.34
Amount in Savings:   $ 1043.34
```

19. Monthly Payments

The monthly payment on a loan may be calculated by the following formula:

$$\text{Payment} = \frac{\text{Rate} \times (1 + \text{Rate})^{\text{N}}}{\left((1 + \text{Rate})^{\text{N}} - 1\right)}$$

`Rate` is the monthly interest rate, which is the annual interest rate divided by 12. (12 percent annual interest would be 1 percent monthly interest.) `N` is the number of payments, and `L` is the amount of the loan. Write a program that asks for these values then displays a report similar to:

```
Loan Amount:            $ 10000.00
Monthly Interest Rate:          1%
Number of Payments:             36
Monthly Payment:        $   332.14
```

```
Amount Paid Back:        $ 11957.15
Interest Paid:           $  1957.15
```

20. Pizza Pi

    Joe’s Pizza Palace needs a program to calculate the number of slices a pizza of any size can be divided into. The program should perform the following steps:

    1. Ask the user for the diameter of the pizza in inches.

    2. Calculate the number of slices that may be taken from a pizza of that size.

    3. Display a message telling the number of slices.

    To calculate the number of slices that may be taken from the pizza, you must know the following facts:

    - Each slice should have an area of 14.125 inches.

    - To calculate the number of slices, simply divide the area of the pizza by 14.125.

    - The area of the pizza is calculated with this formula:

    Area = πr^2

    Note:

    π is the Greek letter pi. 3.14159 can be used as its value. The variable *r* is the radius of the pizza. Divide the diameter by 2 to get the radius.

    Make sure the output of the program displays the number of slices in fixed-point notation, rounded to one decimal place of precision. Use a named constant for pi.

21. How Many Pizzas?

    Modify the program you wrote in Programming Challenge 20 (Pizza Pi) so it reports the number of pizzas you need to buy for a party if each person attending is expected to eat an average of four slices. The program should ask the user for the number of people who will be at the party, and for the diameter of the pizzas to be ordered. It should then calculate and display the number of pizzas to purchase.

22. Angle Calculator

Write a program that asks the user for an angle, entered in radians. The program should then display the sine, cosine, and tangent of the angle. (Use the `sin`, `cos`, and `tan` library functions to determine these values.) The output should be displayed in fixed-point notation, rounded to four decimal places of precision.

23. Stock Transaction Program

    Last month Joe purchased some stock in Acme Software, Inc. Here are the details of the purchase:

    - The number of shares that Joe purchased was 1,000.
    - When Joe purchased the stock, he paid $45.50 per share.
    - Joe paid his stockbroker a commission that amounted to 2 percent of the amount he paid for the stock.

    Two weeks later, Joe sold the stock. Here are the details of the sale:

    - The number of shares that Joe sold was 1,000.
    - He sold the stock for $56.90 per share.
    - He paid his stockbroker another commission that amounted to 2 percent of the amount he received for the stock.

    Write a program that displays the following information:

    - The amount of money Joe paid for the stock.
    - The amount of commission Joe paid his broker when he bought the stock.
    - The amount that Joe sold the stock for.
    - The amount of commission Joe paid his broker when he sold the stock.
    - Display the amount of profit that Joe made after selling his stock and paying the two commissions to his broker. (If the amount of profit that your program displays is a negative number, then Joe lost money on the transaction.)

24. Planting Grapevines

    A vineyard owner is planting several new rows of grapevines, and needs to know how many grapevines to plant in each row. She has determined that after measuring the length of a future row, she can use the following formula to calculate the number of vines that will fit in the row, along with the trellis end-post assemblies that will need to be constructed at each end of the row:

$$V = \frac{R - 2E}{S}$$

The terms in the formula are:

- *V* is the number of grapevines that will fit in the row.
- *R* is the length of the row, in feet.
- *E* is the amount of space, in feet, used by an end-post assembly.
- *S* is the space between vines, in feet.

Write a program that makes the calculation for the vineyard owner. The program should ask the user to input the following:

- The length of the row, in feet
- The amount of space used by an end-post assembly, in feet
- The amount of space between the vines, in feet

Once the input data has been entered, the program should calculate and display the number of grapevines that will fit in the row.

25. Word Game

    Write a program that plays a word game with the user. The program should ask the user to enter the following:

    - His or her name
    - His or her age
    - The name of a city
    - The name of a college
    - A profession
    - A type of animal
    - A pet’s name

After the user has entered these items, the program should display the following story, inserting the user's input into the appropriate locations:

```
There once was a person named NAME who lived in CITY. At the age of AGE, NAME we
```