Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

SKU: 9780134498379-SOLUTIONS

Answers to Review Questions
Chapter 2

1. 1, 2, 3
2. `double temp, weight, age;`
3. `int months = 2, days, years = 3;`
4. A) `b = a + 2;`
   B) `a = b * 4;`
   C) `b = a / 3.14;`
   D) `a = b - 8;`
   E) `a = 27;`
   F) `c = 'K';`
   G) `c = 66;`
5. Multi-line comment
6. Single line comment
7.
```
#include <iostream>
int main()
{
      cout << "Two mandolins like creatures in the\n\n\n";
      cout << "dark\n\n\n";
      cout << "Creating the agony of ecstasy.\n\n\n";
      cout << " - George Barker\n\n\n";
      return 0;
}
```

8. A)
```
0
100
```
   B)
```
8
2
```
   C)
```
I am the incrediblecomputing
machine
and I will
amaze
you.
```
   D)
```
Be careful
This might/n be a trick question
```
   E)
```
23
1
```
9. C
10. C
11. B
12. A
13. B
14. A
15. B, C
16. B, C, D
17. A) 12 B) 4 C) 2 D) 6 E) 1

 ©2018 Pearson Education

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

18. A) 3.287E6 B) -978.65E12 C) 7.65491E-3 D)-58710.23E-4
19. A
20. C
21. A
22. true
23. false
24. true
25. true
26. false

27.
```
int speed, time, distance;
speed = 20;
time = 10;
distance = speed * time;
cout << distance << endl;
```

28.
```
double force, area, pressure;
force = 172.5;
area = 27.5;
pressure = force / area;
cout << pressure << endl;
```

29. The C-style comments symbols are backwards.
`iostream` should be enclosed in angle brackets.
There shouldn't be a semicolon after `int main`.
The opening and closing braces of function main are reversed.
There should be a semicolon after `int a, b, c`.
The comment `\\ Three integers` should read `// Three integers`.
There should be a semicolon at the end of the following lines:
```
a = 3
b = 4
c = a + b
```
`cout` begins with a capital letter.
The stream insertion operator (that appears twice in the `cout` statement) should read << instead of <.
The `cout` statement uses the variable `C` instead of `c`.

 ©2018 Pearson Education

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

LESSON SET

2 Introduction to the C++ Programming Language

PURPOSE

1. To briefly introduce the C++ programming language
2. To show the use of memory in programming
3. To introduce variables and named constants
4. To introduce various data types:
   a. Integer
   b. Character
   c. Floating point
   d. Boolean
   e. String
5. To introduce the assignment and `cout` statements
6. To demonstrate the use of arithmetic operators

PROCEDURE

1. Students should read the Pre-lab Reading Assignment before coming to lab.
2. Students should complete the Pre-lab Writing Assignment before coming to lab.
3. In the lab, students should complete Labs 2.1 through 2.4 in sequence. Your instructor will give further instructions as to grading and completion of the lab.

Contents	Prerequisites	Approximate completion time	Page number	Check when done
Pre-lab Reading Assignment		20 min.	14	
Pre-lab Writing Assignment	Pre-lab reading	10 min.	19	
Lesson 2A				
Lab 2.1 Working with the `cout` Statement	Pre-lab reading	20 min.	20	
Lab 2.2 Working with Constants, Variables, and Arithmetic Operators	Understanding of variables and operators	30 min.	21	

continues

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

Lesson 2B			
Lab 2.3 Rectangle Area and Perimeter	Understanding of basic components of a program	30 min.	22
Lab 2.4 Working with Characters and Strings	Completion of labs 2.1–2.3	30 min.	22

PRE-LAB READING ASSIGNMENT

The C++ Programming Language

Computer programming courses generally concentrate on program design that can be applied to any number of programming languages on the market. It is imperative, however, to apply that design to a particular language. This course uses C++, a popular object-oriented language, for that purpose.

For now, we can think of a C++ program as consisting of two general divisions: header and main. The **header**, or **global section**, gives preliminary instructions to the compiler. It consists of comments that describe the purpose of the program, as well as information on which library routines will be used by the program.

```
// This program prints to the screen the words:
// PI = 3.14
// Radius = 4
// Circumference = 25.12

#include <iostream>
using namespace std;

const double PI  = 3.14;

int main()
{
    float radius;
    radius = 4.0;

    cout << "PI = " << PI << endl;
    cout << "Radius = " << radius << endl;
    cout << "Circumference = " << 2 * PI * radius << endl;

    return 0;
}
```

Everything in bold (everything above the `int main()` statement) is considered the header or global section. Everything else is the main section.

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

Comments are included in every program to document what a program does and how it operates. These statements are ignored by the computer but are most valuable to the programmers who must update or fix the program. In C++, comments begin with `//` which is an indication to the compiler to ignore everything from the `//` to the end of the line. Comments can also cross line boundaries by beginning with `/*` and ending with `*/`. Notice that the first three lines of the previous program all begin with `//` and thus are comments. Those same lines could also have been written as the following:

```
/*   This program prints to the screen the words:
     PI = 3.14
     Radius = 4
     Circumference = 25.12
*/
```

The next statement, the `#include` statement, indicates which library will be needed by the program.

```
#include <iostream>
```

Recall from Lesson Set 1, that every program needs other modules attached so that it may execute properly. Your instructor will generally tell you which libraries are needed for each particular programming assignment; however, in time you will learn this task for yourself.

Every C++ program has a **`main`** function which indicates the start of the executable instructions. Every `main` must begin with a left brace `{` and end with a right brace `}`. The statements inside those braces will be explained as we progress through this lesson.

Memory

Memory storage is the collection of locations where instructions and data that are used by the program are temporarily stored. Recall from Lesson Set 1 that a computer only understands a sequence of 1s and 0s. These are binary digits or **bits (BInary digiTs)**. Eight of these brought together are called a **byte**, which is the most common unit of storage. These chunks of memory can be thought of as hotel mailboxes at the registration desk. The size of each of those boxes indicates the type of mail that can be stored there. A very small mailbox can only hold notes or postcards. Larger mailboxes can hold letters, while even larger ones can hold packages. Each mailbox is identified by a number or name of an occupant. We have identified two very important attributes of these mailboxes: the name or number, which indicates the mailbox that is being referenced, and the size, which indicates what type of "data" can be placed there.

Example: **postcards Jim** is an indication that the mailbox called Jim can only hold postcards, while the statement **packages Mary** indicates that the mailbox called Mary can hold large packages. Memory locations in a computer are identified by the same two attributes: data type and name.

Much of programming is getting data to and from memory locations and thus it is imperative that the programmer tell the computer the name and data type of each memory location that he or she intends to use. In the sample program the statement **`float radius`** does just that. `float` is a data type that indicates what kind of data can be stored and `radius` is the name for that particular memory location.

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

Variables and Constants

The ability to change or not change the data stored can be a third attribute of these memory locations. Components of memory in which data values stored can change during the execution of the program are called **variables**. These usually should not be defined in the header or global section of the program. In our sample program, `radius` is defined in the main function. Components of memory in which data values stored are initialized once and never changed during the execution of the program are called **constants**. They are often defined in the global section and are preceded (in C++) by the word **`const`**. PI, in the sample program, is an example of a named constant.

Identifiers in C++

Identifiers are used to name variables, constants and many other components of a program. They consist exclusively of letters, digits and the underscore _ character. They cannot begin with a digit and cannot duplicate reserved words used in C++ such as **`int`** or **`if`**. All characters in C++ are case sensitive; thus memory locations called `simple`, `Simple`, and `SIMPLE` are three distinct locations. It has become standard practice among programmers to make constants all uppercase and variables predominantly lowercase characters.

The statement **`const double PI = 3.14;`** in our sample program is contained in the global section. It defines a memory location called `PI` to be a constant holding a double (a data type discussed shortly) value equal to 3.14 which will *not* change during the execution of the program.

The statement **`float radius;`** in the sample program is contained in the main section. It defines a variable memory location called `radius` that holds a floating point data type (type discussed shortly) which can be changed during the execution of the program.

Both of these statements are called **definitions**. They reserve by name enough memory to hold the data type specified.

Variables, like constants, can be given an initial value when they are defined, but that value is not permanent and can be altered. For example:

```
int count = 7;       // Defines a variable memory location called count that
                     // initially has the value of 7
count = count + 1;   // count is now altered
```

Data Types

As noted earlier, computer memory is composed of locations identified by a data type and a name (like the room number of a hotel mailbox). The data type indicates what kind of data can be stored, thus setting the size of that location.

Integer Data Type

Integers are real numbers that do not contain any fractional component. They take up less memory than numbers with fractional components. C++ has three data types that are integers: **`short`**, **`int`** and **`long`**. The difference is strictly in the amount of memory (bytes) they reserve: `short` reserving the least and `long` reserving the most. Larger integers may need the `long` data type.

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

The following three statements define integer variables in C++:

```
short count;
int sum;
long total;
```

Floating Point Data Type

In computer science 3 = 3.0 is not a true statement. The number on the left is an integer and the number on the right is a real, or floating point, number (a number that has a fractional component). Although mathematically the two are equal, the computer stores them as different data types. C++ uses both **`float`** and **`double`** to indicate floating point numbers, with `double` using more memory than `float`.

The following two statements define floating point variables in C++.

```
float average;
double nationaldebt;
```

Character Data Type

Character data includes the letters of the alphabet (upper and lower cases), the digits 0–9 and special characters such as ! ? . , *. All these symbols combined are called **alphanumeric**. Each character data is enclosed with single quotes to distinguish it from other data types. Thus `'8'` is different than `8`. The first is a character while the second is an integer. The following statement defines a C++ character variable initialized to `'a'`.

```
char letter = 'a';
```

Boolean Data Type

The Boolean data type, named after the mathematician George Boole, allows only two values: true or false, which are internally represented as 0 and non-zero, respectively. The following statement defines a Boolean variable initialized to false.

```
bool found = false;
```

String Type: A variable defined to be character data can store only one character in its memory location, which is not very useful for storing names. The **string class** has become part of standard C++ and, although not a primitive type defined by the language, it can be used as a type for storing several characters in a memory location. We must "include" the string library (`#include <string>`) in the program header. The following statement defines a string initialized to "Daniel": `string name = "Daniel";` Note that a string is enclosed in double (not single) quotes. Thus the string `"a"` is not the same as the character `'a'`.

Assignment Operator

The = symbol in C++ is called the **assignment operator** and is read "is assigned the value of." It assigns the variable on its left the value on its right. Although this symbol looks like an equal sign, do not confuse it with equality. The left hand side must always be a variable. For example, `count = 8;` is a legitimate statement in C++, however `8 = count;` generates a syntax error.

Fundamental Instructions

Most programming languages, including C++, consist of five fundamental instructions from which all code can be generated.

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

1. **Assignment Statements**: These statements place values in memory locations. The left side of an assignment statement consists of one and only one variable. The right side consists of an **expression**. An expression can be any manipulation of **literal** numbers (actual numbers such as 7 or 38, etc.), or the contents of constants and/or variables, that will "boil down" to one value. That value is placed in the memory location of the variable on the left. C++ uses = as the separator between the left and right side of the assignment statement. Those new to programming often get this confused with equality; however = in C++ is not equality but rather the symbol to indicate assignment. The = in C++ is read as "is assigned the value of".

Example:

```
int count;

int total;

total = 10;       // 10 is a literal that is placed in the memory
                  // location called total

count = 3 + 4;    // The right hand side of the statement is evaluated to
                  // 7. count is assigned the value of 7.

total = total + count; // The right hand side is evaluated 10 + 7,
                       // and 17 is placed in the memory location called
                       // total.
```

This last statement may seem a bit confusing. Starting with the right side, it says to get the value that is in `total` (10 in this case), add it to the value that is in `count` (7 in this case), and then store that combined sum (17) in the memory location called `total`. Notice that `total`, which was initially 10, gets changed to 17.

2. **Output Statements**: These instructions send information from the computer to the outside world. This information may be sent to the screen or to some file. In C++ the **`cout <<`** statement sends information to the screen. The `#include <iostream>` directive must be in the header for `cout` to be used.

```
cout << total;
```

The above statement sends whatever value is stored in the variable `total` to the screen. C++ uses the semicolon as a statement terminator.

We can output literal strings (such as "Hello") by inclosing them in double quotes.

The << operator acts as a separator for multiple outputs.

```
cout << "The value of total is " << total << endl;
```

The `endl` statement causes the cursor to be moved to the beginning of the next line.

The remaining three fundamental instructions will be explained in future labs.

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

3. **Input Statements**: These statements bring in data to the computer. (Lesson Set 3)
4. **Conditional Statements**: These instructions test conditions to determine which path of instructions to execute. (Lesson Set 4)
5. **Loops**: These instructions indicate a repetition of a series of instructions. (Lesson Set 5)

Arithmetic Operators

Programming has the traditional arithmetic operators:

Operation	C++ Symbol
addition	+
subtraction	-
multiplication	*
division	/
modulus	%

Integer division occurs when both the numerator and denominator of a divide operation are integers (or numbers stored in variables defined to be integers). The result is always an integer because the decimal part is truncated or "chopped" from the number. Thus 9/2 will give the answer 4 not 4.5! For this reason there are two division operations for integer numbers. The modulus operator, (%) used only with integers, gives the remainder of a division operation. 9/2 gives 4 while 9 % 2 gives 1 (the remainder of the division).

Example:

```
int count = 9;
int div = 2;
int remainder;
int quotient;

quotient = count / div;         // quotient is assigned a 4
remainder = count % div;        // remainder is assigned a 1
```

You should go back and review the sample program on the first page of the Pre-lab Reading Assignment. By now you should understand most of the statements.

PRE-LAB WRITING ASSIGNMENT

Fill-in-the-Blank Questions

1. A _____is a memory location whose value cannot change during the execution of the program.
2. _____is a data type that only holds numbers with no fractional component.
3. _____is a data type that holds numbers with fractional components.
4. _____is an arithmetic operator that gives the remainder of a division problem.

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

5. `cout <<` is an example of the _____fundamental instruction.
6. _____data types only have two values: true and false.
7. One byte consists of _____bits.
8. // or /* in C++ indicates the start of a _____.
9. A _____is a memory location whose value can change during the execution of the program.
10. A _____can hold a sequence of characters such as a name.

LESSON 2A

LAB 2.1 Working with the `cout` Statement

Exercise 1: Retrieve program `name.cpp` from the Lab 2 folder.
Fill in the code so that the program will do the following:

Write your first and last name on one line.
Write your address on the next line (recall the function of the `endl` statement).
Write your city, state and zip on the next line.
Write your telephone number on the next line.
Remember that to output a literal, such as "Hello", you must use quotes.
Compile and run the program.

Example: Deano Beano
123 Markadella Lane
Fruitland, Md. 55503
489-555-5555

The code for `name.cpp` is as follows:

```
//  This program will write the name, address and telephone
//  number of the programmer.

// PLACE YOUR NAME HERE

#include <iostream>
using namespace std;

int main()
{

      // Fill in this space to write your first and last name
      // Fill in this space to write your address (on new line)
      // Fill in this space to write you city, state and zip (on new line)
      // Fill in this space to write your telephone number (on new line)

      return 0;

}
```

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

Exercise 2: Change the program so that three blank lines separate the telephone number from the address. Compile and run the program.

Exercise 3: Change the program so that the following (but with your name and address) is printed. Try to get the spacing just like the example. Compile and run the program.

```
************
    Programmer: Deano Beano
                123 Markadella Lane
                Fruitland, Md. 55503

     Telephone: 489-555-5555

************
```

LAB 2.2 Working with Constants, Variables and Arithmetic Operators

Exercise 1: Bring in the file `circlearea.cpp` from the Lab 2 folder.

The code of `circlearea.cpp` is as follows:

```
// This program will output the circumference and area
// of the circle with a given radius.

//  PLACE YOUR NAME HERE

#include <iostream>
using namespace std;

const  double PI = 3.14;
const  double RADIUS = 5.4;

int main()

{
       _____________area                 // definition of area of circle

       float  circumference;             // definition of circumference

       circumference = 2 * PI * RADIUS;  // computes circumference

       area =_____________;              // computes area

   // Fill in the code for the cout statement that will output (with description)
   // the circumference

   // Fill in the code for the cout statement that will output (with description)
   // the area of the circle

   return 0;
}
```

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

Exercise 2: Fill in the blanks and the `cout` statements so that the output will produce the following:

```
The circumference of the circle is 33.912
The area of the circle is 91.5624
```

Exercise 3: Change the data type of circumference from `float` to `int`. Run the program and record the results.

The circumference of the circle is _____.

The area of the circle is _____.

Explain what happened to get the above results.

LESSON 2B

LAB 2.3 Rectangle Area and Perimeter

Exercise 1: Using Lab 2.2 as an example, develop a program that will determine the area and perimeter of a rectangle. The length and width can be given as constants. (`LENGTH=8 WIDTH=3`)

Exercise 2: Compile and run your program. Continue to work on it until you get the following output.

```
The area of the rectangle is 24
The perimeter of the rectangle is 22
```

LAB 2.4 Working with Characters and Strings

Exercise 1: Retrieve program `stringchar.cpp` from the Lab 2 folder. This program illustrates the use of characters and strings. The `char` data type allows only one character to be stored in its memory location. The `string` data type (actually a class and not a true data type built into the language) allows a sequence of characters to be stored in one memory location. The code follows:

```
//  This program demonstrates the use of characters and strings

//  PLACE YOUR NAME HERE

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

// Definition of constants
const string FAVORITESODA = "Dr. Dolittle";  // use double quotes for strings
const char BESTRATING = 'A';                 // use single quotes for characters


int main()

{
```

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

```
    char rating;            // 2nd highest product rating
    string favoriteSnack;   // most preferred snack
    int numberOfPeople;     // the number of people in the survey
    int topChoiceTotal;     // the number of people who prefer the top choice

    //  Fill in the code to do the following:
    //  Assign the value of "crackers" to favoriteSnack
    //  Assign a grade of 'B' to rating
    //  Assign the number 250 to the numberOfPeople
    //  Assign the number 148 to the topChoiceTotal

    // Fill in the blanks of the following:
    cout << "The preferred soda is " <<_____________<< endl;
    cout << "The preferred snack is " <<_____________<< endl;
    cout << "Out of " <<_____________<< " people "
         <<_____________  << " chose these items!" << endl;
    cout << "Each of these products were given a rating of " <<_____________;
    cout << " from our expert tasters" << endl;
    cout << "The other products were rated no higher than a " << rating
         << endl;

    return 0;

}
```

Exercise 2: Fill in the indicated code, then compile and run the program. Continue to work on the program until you have no syntax, run-time, or logic errors.

The output should look similar to the following:

The preferred soda is Dr. Dolittle
The preferred snack is crackers
Out of 250 people 148 chose these items!
Each of these products were given a rating of A from our expert tasters
The other products were rated no higher than a B

Exercise 3: Is it possible to change the choice of FAVORITESODA by adding code within the main module of the program? Why or why not?

Exercise 4: Is it possible to change the choice of favoriteSnack by adding code within the program? Why or why not?

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual

Starting Out with C++ from Control Structures to Objects 9th edition Gaddis Solutions Manual