# Chapter 1

## Short Answer

1. Why is the CPU the most important component in a computer?

Because it is the part of the computer that runs programs.

2. What number does a bit that is turned on represent? 1

What number does a bit that is turned off represent? 0

3. What would you call a device that works with binary data?

A digital device

4. What are the words that make up a high-level programming language called?

Key words, or reserved words

5. What are the short words that are used in assembly language called?

Mnemonics

6. What is the difference between a compiler and an interpreter?

Compilers translate code into machine language.

Interpreters translate into machine language and execute.

7. What type of software controls the internal operations of the computer’s hardware?

Operating System.

## Exercises

### 1. Convert the following decimal numbers to binary:

11

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 4 | 2 | 1 |
| 1 |  | 1 |  |

Answer: 1010

65

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 |  |  |  |  |  | 1 |

Answer: 1000001

100

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 1 |  |  | 1 |  |  |

Answer: 1100100

255

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Answer: 11111111

### 2. Use what you’ve learned about the binary numbering system in this chapter to convert the following binary numbers to decimal:

1101 = 13 = (1 × 2³) + (1 × 2²) + (0 × 2¹) + (1 × 2⁰)

1000 = 8 = (1 × 2³) + (0 × 2²) + (0 × 2¹) + (0 × 2⁰)

101011 = 43 = (1 × 2⁵) + (0 × 2⁴) + (1 × 2³) + (0 × 2²) + (1 × 2¹) + (1 × 2⁰)

### 3. Look at the ASCII chart in Appendix A and determine the codes for each letter of your first name.

Mark = 077 097 114 107

mark = 109 097 114 107

### 4. Use the Web to research the history of the BASIC, C++, Java, and Python programming languages, and answer the following questions:

* BASIC
  + John Kemeny, Tom Kurtz
  + 1964
  + To expand the use of computers beyond math/science
* C++
  + Bjarne Stroustrup
  + 1985
  + To improve the C language based on features from Simula
* Java
  + James Gosling
  + 1995
  + Intended to run anywhere that supported Java after being compiled. No idea what his motivation was.
* Python
  + Guido van Rossum
  + 1991
  + I’ve no idea what his motivation was. Probably just scratching an itch he had with existing languages.

# Chapter 2

## True or False

1. Programmers must be careful not to make syntax errors when writing pseudocode programs.

False - technically there’s no syntax to pseudocode.

2. In a math expression, multiplication and division take place before addition and subtraction.

True

3. Variable names can have spaces in them.

False

4. In most languages, the first character of a variable name cannot be a number.

True

5. The name gross\_pay is written in the camelCase convention.

False - snake\_case

6. In languages that require variable declarations, a variable’s declaration must appear before any other statements that use the variable.

True

7. Uninitialized variables are a common cause of errors.

True

8. The value of a named constant cannot be changed during the program’s execution.

True

9. Hand tracing is the process of translating a pseudocode program into machine language by hand.

False

10. Internal documentation refers to books and manuals that document a program, and are intended for use within a company’s programming department.

True

## Short Answer

1. What does a professional programmer usually do first to gain an understanding of a problem?

Discuss with client/customer

2. What is pseudocode?

A way to describe the way a program will work using plain English, without having to worry about the syntax of a programming language.

3. Computer programs typically perform what three steps?

Handle input, processing and output of data.

4. What does the term user-friendly mean?

That something is easy for the typical end-user to understand and operate.

5. What two things must you normally specify in a variable declaration?

Type and value.

6. What value is stored in uninitialized variables?

Depends. In some languages it will hold the value, if any, of the content of the stack space. For all intents and purposes though, unitialized variables have no value.

# Chapter 3

## True or False

1. The phrase “divide and conquer” means that all of the programmers on a team should be divided and work in isolation.

False

2. Modules make it easier for programmers to work in teams.

True - in general

3. Module names should be as short as possible.

False

4. Calling a module and defining a module mean the same thing.

False

5. A flowchart shows the hierarchical relationships between modules in a program.

False

6. A hierarchy chart does not show the steps that are taken inside a module.

True

7. A statement in one module can access a local variable in another module.

False

8. In most programming languages, you cannot have two variables with the same name in the same scope.

True

9. Programming languages typically require that arguments be of the same data type as the parameters that they are passed to.

True

10. Most languages do not allow you to write modules that accept multiple arguments.

False

11. When an argument is passed by reference, the module can modify the argument in the calling part of the program.

True

12. Passing an argument by value is a means of establishing two-way communication between modules.

False (unless it’s required for the module’s return. I guess that could be considered two-way communication)

## Short Answer

1. How do modules help you to reuse code in a program?

A module can perform a repetitive task removing the need to repeat code every time the task needs to be completed.

2. Name and describe the two parts that a module definition has in most languages.

* Header - includes the module name, accepted parameters and possibly the value type returned. It’s used to call the module.
* Body - includes any tasks the module performs.

3. When a module is executing, what happens when the end of the module is reached?

The computer returns to the section of the program where the module was called.

4. What is a local variable? What statements are able to access a local variable?

A local variable is limited to the scope in which it was created. It is only accessible by code in the same scope. If a local variable was created in a module, only other code within that module can access it.

5. In most languages, where does a local variable’s scope begin and end?

Its scope begins and ends with the module in which it was created. The variable does not exist outside of the module.

6. What is the difference between passing an argument by value and passing it by reference?

* Passing by value provides that module with a value to work with. The original variable remains unaffected.
* Passing by reference allows the module to change the value stored in the variable used to call the module.

7. Why do global variables make a program difficult to debug?

* It’s not always easy to tell where they are defined and what their expected value is.
* Any code anywhere can change the value of a global variable.

## Algorithm Workbench

1. Design a module named timesTen. The module should accept an Integer argument. When the module is called, it should display the product of its argument multiplied times 10.

Module timesTen(Integer n)  
 Declare Integer res  
 Set res = n \* 10  
 Display res  
End Module

# Chapter 4

## True or False

1. You can write any program using only sequence structures.

False

2. A program can be made of only one type of control structure. You cannot combine structures.

False

3. A single alternative decision structure tests a condition and then takes one path if the condition is true, or another path if the condition is false.

False

4. A decision structure can be nested inside another decision structure.

True

5. A compound Boolean expression created with the AND operator is true only when both subexpressions are true.

True

## Short Answers

1. Explain what is meant by the term conditionally executed.

Executed only when a condition is true.

2. You need to test a condition and then execute one set of statements if the condition is true. If the condition is false, you need to execute a different set of statements. What structure will you use?

Dual Alternative Decision

3. If you need to test the value of a variable and use that value to determine which statement or set of statements to execute, which structure would be the most straightforward to use?

Case Structure

4. Briefly describe how the AND operator works.

Compares the boolean value of two conditions and only returns true if both conditions are true.

5. Briefly describe how the OR operator works.

Compares the boolean value of two conditions and returns true if either of them are true.

6. When determining whether a number is inside a range, which logical operator is it best to use?

AND. As in IF x < 20 AND x > 10

7. What is a flag and how does it work?

A variable that signals when a condtion exists. It’s usually set to false until the condition exists.

## Algorithm Workbench #6

Rewrite the following If-Then-Else If statement as a Select Case statement.

If selection == 1 Then  
 Display "You selected A."  
Else If selection == 2 Then  
 Display "You selected 2."  
Else If selection == 3 Then  
 Display "You selected 3."  
Else If selection == 4 Then  
 Display "You selected 4."  
Else  
 Display "Not good with numbers, eh?"  
End If

==========================================

Continued on next page…

std::string output;  
  
switch(selection) {  
 case 1:  
 output = "You selected A.";  
 break;  
 case 2:  
 output = "You selected 2.";  
 break;  
 case 3:  
 output = "You selected 3.";  
 break;  
 case 4:  
 output = "You selected 4.";  
 break;  
 default:  
 output = "Not good with numbers, eh?";  
}  
  
std::cout << output << std::endl;

# Chapter 5

## True or False

1. A condition-controlled loop always repeats a specific number of times.

False

2. The While loop is a pretest loop.

True

3. The Do-While loop is a pretest loop.

False

4. You should not write code that modifies the contents of the counter variable in the body of a For loop.

True

5. You cannot display the contents of the counter variable in the body of a loop.

False

6. It is not possible to increment a counter variable by any value other than 1.

False

7. The following statement decrements the variable x: Set x = x - 1.

True

8. It is not necessary to initialize accumulator variables.

False

9. In a nested loop, the inner loop goes through all of its iterations for every single iteration of the outer loop.

True

10. To calculate the total number of iterations of a nested loop, add the number of iterations of all the loops.

False

## Short Answer

1. Why should you indent the statements in the body of a loop?

Makes it easier to read, debug

2. Describe the difference between pretest loops and posttest loops.

Pretest only performs the loop if test is true. Posttest always performs loop at least once.

3. What is a condition-controlled loop?

Loop is performed as long as a condition is true.

4. What is a count-controlled loop?

Loop is performed a certain number of times.

5. What three actions do count-controlled loops typically perform using the counter variable?

Initialize, test, increment

6. What is an infinite loop? Write the code for an infinite loop.

A loop that does not end, such as while(true){ do\_something() }

7. A For loop looks like what other loop in a flowchart?

It’s the same as a while loop, only one is condition based and one is count based.

8. Why is it critical that accumulator variables are properly initialized?

You can’t properly count if you don’t know where you’re starting from.

9. What is the advantage of using a sentinel?

Don’t have to know how many times the loop needs to iterate.

10. Why must the value chosen for use as a sentinel be carefully selected?

It’s useless if it can be mistaken for a valid value.

## Algorithm Workbench #7

Convert the While loop in the following code to a Do-While loop:

Declare Integer x = 1  
While x > 0  
 Display "Enter a number."  
 Input x  
End While

Converted:

Declare Integer x  
Do  
 Display "Enter a number:"  
 Input x  
While x > 0