

Lesson 13



Modules and Documentation

Terms to Know

documentation. Descriptive phrases in computer programs, ignored by the computer but helpful to humans.

module. A building block of a computer program; it performs a specific task and can be used in programs other than the one for which it was originally written.

This lesson briefly explores two techniques that make computer programs better. The first is the **module** concept.

Software is written to do a wide variety of things, such as print reports, calculate payroll, operate robots, and regulate temperatures, to mention only a few. But within any given piece of software—whether for business, manufacturing, or health care environments—blocks of code are written to perform specific functions. These blocks, called modules, carry out specific functions that may need to be done again somewhere else in the program or even in another program.

Programmers capitalize on these modules. A module is a great time-saver. A module written to perform a particular task can be written once and then used repeatedly. There is less possibility for error, too. If it has been done correctly once, new errors are not introduced into it by trying to write it again. The challenge comes in writing a module that is general enough to be usable in other places yet specific enough to perform a useful task.

Documentation is another technique that makes computer programming easier. Every computer language provides some way that the programmer can write comments or explanations into the program for human beings to read that computers will ignore. Usually a comment is preceded by a special character or characters that tell the computer to ignore what follows. It could be an apostrophe as in Visual Basic or a / and an

asterisk as in C. The example below shows a comment in Visual Basic. The first line—describing the section of code—is preceded by an apostrophe and will be ignored by the computer.

This module enters a subscription.

```
'If Format.Text = "subscription" Then
Text13.SetFocus
field1 = Text13.Text
stDocName = "Subscription Order"
DoCmd.OpenForm stDocName
DoCmd.GoToRecord , , acNewRec
[Forms]![Subscription Order].SetFocus
[Forms]![Subscription Order]![Title
Number].Text = field1
Endif
```

The pieces of programs you have seen so far in this course have been printed without accompanying documentation. However, in a real programming environment, documentation is important for a couple of reasons. First, the programmers themselves will use it to help them understand what their program is doing during a certain stage.

Secondly, programs are often very large, making it difficult to find a specific segment or look at only a part of a program to see how it affects the whole. Programmers may sometimes forget how their programs work,

and if they need to study the programs later to fix them or to make enhancements, the whole process is much easier if they were careful to enter explanations through the program when it was first written. Otherwise they need to think through the whole process again.

Another reason for documentation is that programmers are not indispensable. The author of a particular program will not always be there to make the needed changes

a week, a month, or a year down the road. Therefore, they must document their writing so that someone else can maintain the software they originally wrote. Plus, complex software is written by teams of people, and one person on the team must have an understanding of what work another person has done so that all the pieces will fit together at the end. This communication can be partially accomplished through documentation.



Answer the questions.

1. What is a module? _____

2. Why are they such great time-savers? _____

3. How can they help to reduce errors in a program? _____

4. What is documentation? _____



Give four reasons documentation is important.

5. a. _____
b. _____
c. _____
d. _____

LOOKING BACK . . .



List the arithmetic operators.

6. Four common ones: _____

7. Three less common ones (tell what each does):

Lesson 13



List six relational operators.

8. _____



List three logical operators.

9. _____



Complete these sentences.

10. The logical operator _____ will cause something to happen only if both conditions are true.
11. The logical operator _____ will cause something to happen if one of several conditions is true.
12. Three basic structures used in computer programming are _____ , _____ , and _____ .
13. The _____ structure tells the computer in what order to do certain steps.
14. The _____ structure tells the computer to keep performing a certain action _____ a certain condition is met or _____ a certain condition remains true.
15. The _____ structure instructs the computer to perform a test and to take one action if the test result is _____ , and another action if the test result is _____ .
16. A _____ is a picture or diagram showing the structures of a computer program.



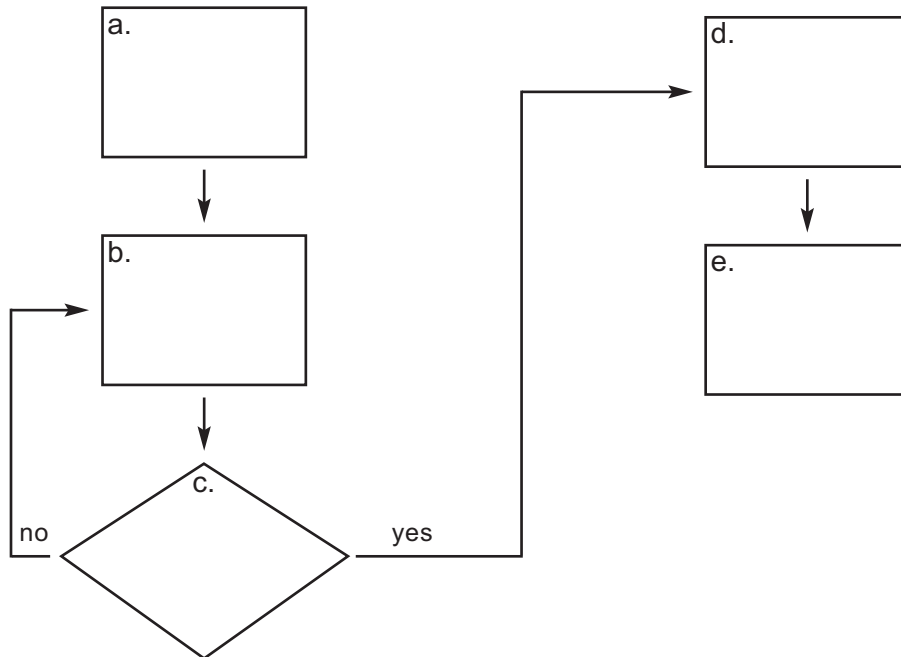
Write *sequence, selection, or iteration.*

17. _____ Do these ten steps in this specific order.
18. _____ Keep repeating an action until a given event happens.
19. _____ If one things happens, do this. If another thing, do that.
20. _____ If it snows tonight, we will go sledding tomorrow.
21. _____ Instructions for building a birdhouse.
22. _____ *Until* and *while* are the signals for this action.
23. _____ Directions for getting to someone's house.

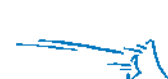


Complete the flowchart for this set of directions.

24. Turn left onto Rt.16 as you leave the interstate.
Go through several towns—I'm not sure how many.
If you cross under I-95, you'll know you've gone through the last town.
Turn right onto Jemison Drive.
Find the school on the left after you cross Maple Street.



Lesson 14



Debugging and Upgrades



Terms to Know

debugging. The process of finding and fixing errors in a computer program.

upgrade. An improved version of software. Also used as a verb to describe what a user does to improve his computer system by buying new software and/or hardware.

Lesson 14

These two concepts are also related to producing software that will run hardware. The first is critical before the software is useful at all, and the second is a part of improving the software and adding features that make the software more useful or user-friendly.

Lets look at **debugging** first. Debugging is looking for mistakes in a computer program. Since the instructions given to a computer must be extremely meticulous and specific, an error may be as small as a punctuation mark or the lack thereof. Finding those errors can be a tedious process, although not as tedious as it was a few decades ago when people programmed in zeros and ones.

Today, computer languages include debugging tools. These additional pieces of software help the programmer pinpoint the problem. They can sometimes show the point at which the computer quit being able to understand the instructions given to it. But even then, the debugging tools don't do all the work.

Errors are usually of two major types: typographical and logical. The typographical ones are easier to find. They are most often caused by simply misspelling words, such as variable names or commands, or failing to use the proper syntax. Logical errors can be very difficult to find, largely because we humans do not think like computers. We think we have given the computer the correct steps in the correct order, but we have inadvertently overlooked a key step or arranged the steps in the wrong order. The computer may still operate and even give an answer, but it's not the right one!

Think of a cooking example. The recipe says to mix the cookies and drop by teaspoonfuls onto a baking sheet two inches apart. What the recipe did not say was to get the baking sheet out of the cupboard first. With people, that is understood; however a

computer would lock up because it would not know how to get the sheet unless it was part of its instructions.

A programmer can use several tricks to help in the debugging process. One is to force the computer to quit following instructions before the program is actually over. If it runs correctly up to that point, he knows the problem lies later in the program. Then he could move the checkpoint further down in the instructions to see if it still runs correctly. He then repeats the process as needed. Another debugging trick is to make the computer print out what it thinks the values of its variables are at any given time. That will give a clue as to why the computer is doing what it does. The third trick is to make the statement or group of statements that seems to be causing the problem into comments. Then the computer will ignore the statements, and the programmer can judge from how the computer then reacts as to whether the problem exists there or somewhere else. Fixing the problem will still be up to the programmer to figure out, but these tricks are helpful in isolating where the problem is.

What about **upgrades**? There are several reasons to upgrade. First, when people use software for a while, they sometimes uncover bugs that no one knew existed, even though the software has been tested beforehand. Secondly, software companies are busy writing new software or improving what they have already written. Users often have suggestions about how the software could be made easier to use or how it would be nice if the software included another feature. These ideas are often included in upgrades. And finally, upgrades are a great selling point. If the software marketers can convince a potential customer that he "needs" the upgrade, that's another sell and another dollar in the company's pocket.



Answer the questions.

1. What is debugging? _____
-

2. Why is debugging not as difficult as it once was? _____

3. What are the two main types of bugs? _____
4. Which is more difficult to find? Why? _____

5. Where do ideas for new features in upgrades often come from? _____

6. Why do software companies like upgrades? _____



Give two reasons to upgrade.

7. a. _____
- b. _____



Complete these sentences about debugging methods.

8. Stopping a computer _____ the program is done may reveal where the bug is.
9. Making the computer _____ what it thinks the variables are at any given time can help to find the problem.
10. Changing _____ or groups of them to documentation so that the computer will ignore it may help the programmer decide if a particular section is causing the problem.

LOOKING BACK . . .



Write true or false.

11. _____ Computer programmers like to use versatile modules as much as possible.
12. _____ Computers use documentation to help them correctly run programs.
13. _____ Modules and documentation both save programmers time.
14. _____ Computers ignore documentation unless it is incorrectly identified.
15. _____ Computer programmers use documentation only to help themselves remember what they meant for their program to do.
16. _____ A set of code for addressing a label could not be used as a module.
17. _____ Modules may sometimes be used in programs besides the ones for which they were originally written.