# Prelude to programming

Technology

As you took your first step you had to figure out how to execute the following process: Put one foot in front of the other At some point you did just that, and it was a major accomplishment. But this didn't get you very far. If you wanted to walk across the room, you needed to extend this process to the following: put the left foot in front of the right foot Put the right foot in front of the left foot Put the left foot in front of the right foot Put the right foot in front of the left foot and so forth This is not a very efficient way to describe what you did.

A detailed list of your actions as you ambled all over the house would be very long. Because you did the same thing over and over, the following is a much better way to describe your actions: Repeat Put the right foot in front of the left foot until you get across the room This way is short, convenient, and just as descriptive. Even if you want to take hundreds or thousands of steps, the process can still be described in four lines. This is the basic idea off loop. Walking is just one of many examples of loops in your daily life.

For example, if you have a largefamilyand need to prepare lunches in the morning for everyone, you can do the following: Make a sandwich Wrap the sandwich Place the sandwich in a lunch bag place an apple in the lunch bag Place a drink in the lunch bag Continue until lunches have been made for everyone in the family Where else do you encounter a looping process? How about eating a sandwich (one bite at a time) or brushing your teeth? If you have a programming class on Tuesdays at 1 1100 a. M. , you go to class every Tuesday at 1 1 a. M. Until the end of the semester.

You do the " go to programming class" loop until a certain day. After you read this chapter (one word at a time), you'll be ready to place loops in your programs as well. 4. 1 An Introduction to Repetition Structures: Computers Never Get Bored! 165 4. 1 An Introduction to Repetition Structures: Computers Never Get Bored! You have already learned that all computer programs are created from three basic constructs: sequence, decision, and repetition. This chapter discusses repetition, which in many ways is the most important construct of all. We are lucky that computers don't find repetitious tasks boring.

Regardless of what task we ask a computer to perform, the computer is virtually useless if it can perform that task only once. The ability to repeat the same actions over and over is the most basic requirement in programming. When you use any software application, you expect to be able to open the application and do certain tasks. Imagine if your word processor Was programmed to make your text bold only once or if your operating system allowed you to use the copy command only once. Each computer task you perform has been coded into the software by a programmer and each task must have the ability to be used over and over.

In this chapter, we will examine how to program a computer to repeat one or more actions many times. Pop Basics All programming languages provide statements to create a loop. The loop is the basic component of the repetition structure. These statements are a block of code, which under certain conditions, will be executed repeatedly. In this section, we will introduce some basic ideas about these structures. We will start with a simple illustration of a loop shown in Example 4. 1 . This example uses a

type of loop called a Repeat... Until loop. Other types of loops are discussed throughout the chapter. Example 4.

Simply Writing Numbers This program segment repeatedly inputs a number from the user and displays that number until the user enters O. The program then displays the words List Ended. 2 3 5 6 7 Declare Number As Integer Write " Please enter a number: ' Input Number Write Number Until Number - Write " List Ended" In the pseudopodia, the loop begins on line 2 with the word Repeat and ends on line 6 with Until Number O. The loop body is contained in lines 3, 4, and 5. These are the statements that will be executed repeatedly. The body of a loop is executed until the test condition following the word Until on line 6 becomes true.

In this case, the test condition becomes true when the user types a O. At that point, the loop is exited and the statement on line 7 is executed. 1 66 What Happened? Let's trace the execution of this program, assuming that the user enters the numbers 1, 3, and O, in that order: When execution begins the loop is entered, the number 1 is input, and this number is displayed. These actions make up the first pass through the loop. The test condition, " Number - = O? " is now " tested" on line 6 and found to be false because at this point, Number =-? Therefore, the loop is entered again.

The program execution returns to line 2 ND the body of the loop is executed again. (Recall that the double equals sign, -? 2, is a comparison operator and asks the question, " Is the value of the variable Number the same as O? ) On the second pass through the loop, the number 3 is input (line 4) and displayed (line 5), and once again the condition (line 6), Number = = O is

false. So the program returns to line 2. On the third pass through the loop, the number O is input and displayed. This time the condition Number == O is true, so the loop is exited and execution transfers to line 7, the statement after the loop.

The words List Ended are displayed and the program is complete. Iterations We have said that the loop is the basic component of the repetition structure. One of the main reasons a computer can perform many tasks efficiently is because it can quickly repeat tasks over and over. The number of times a task is repeated is always a significant part of any repetition structure, but a programmer must be aware of how many times a loop will be repeated to ensure that the loop performs the task correctly. In computer lingo a single pass through a loop is called a loop iteration.

A loop that executes three times goes through three iterations. Example 4. 2 presents the iteration process. Example 4. 2 How Many Iterations? This program segment repeatedly asks the user to input a name until the user enters " Done. " Declare Name As String Write " Enter the name Of your brother or sister: " Input Name Write Name until Name " Done" and Elizabeth Drake. Published by Addison-Wesley. Copyright C 2011 by This pseudopodia is almost the same as shown in Example 4. 1 except that the input in this example is string data instead Of integer data.

The loop begins on line 2 with the word Repeat and ends on line 6 with until Name " Done". The loop body is contained in lines 3, 4, and 5. How are the iterations counted? Each time these statements are executed, the loop is said to have gone through one iteration. 167 Let's assume this program

segment is used to enter a list of a user's brothers and sisters. If Hector has two brothers named Joe and Jim and one sister named Ellen, the loop would complete four iterations. Joe would be entered on the first iteration, Jim on the next iteration, Ellen on the third iteration, and the word Done would be entered on the fourth iteration.

If Marie, on the other hand, had only one sister named Anne, the program would go through two iterations-? one to enter the name Anne and one to enter the word Done. And if Bobby were an only child, the program would only complete one iteration since Bobby would enter Done on the first iteration. Later in this chapter, We will see how to create a loop that does not require that the test condition count as one of the iterations. Beware of the Infinite Loop! In Example 4. 1 , we saw that the user was prompted to enter any number and that number would be displayed on the screen.

Fifth user started with the number 234789 and worked his way down, entente 234, 788, then 234, 787, and so forth, the computer would display 234, 790 numbers (including the O that terminates the loop). However, after the user entered the last number, O, the loop would end. It would be a lot of numbers, but it would end. On the other hand, what would happen if the loop was written as shown in Example 4. 3? Example 4. 3 The Dangerous Infinite Loop In this example, we change the test condition of Example 4. 1 to a condition that is impossible to achieve. The user is asked to enter a number on line 2 and line 3 takes in the user's input.

Line 4 sets a new variable, Computerized equal to that number plus one. The loop will continue to ask for and display numbers until the value of Number is

greater than Computerized. That condition will never be met because on each pass through the loop, regardless of what number the user enters, Computerized will always be one greater. Thus, the loop will repeat and repeat, continually asking for and displaying numbers. 8 Declare Number, Computerized As Integer Write " Please enter a number: " Computerized = Number + 1 Until Number > Computerized Write " The End" When will it end?

Never. The words The End will never be displayed. If, as shown in Example 4. 3, a loop's test condition is never satisfied, then the loop will never be exited and it will become an infinite loop. Infinite loops can reek and Elizabeth Drake. Published by Addison-Wesley. Copyright @ 2011 by 168 havoc on a program, so when you set up a loop and put in a test condition, be sure that the test condition can be met. Computers don't mind doing a task many times, but forever is simply too many! Don't Let the User Get Trapped in a Loop There is one more important point to mention about Examples 4. And 4. 2. In both of these examples, we have test conditions that can easily be met. As soon as a user enters O for the number in Example 4. 1, the loop ends. As soon as the user enters the word Done in Example 4. 2, the loop ends. But owe would the user know that O or Done is the cue for the program segment to end? It is important for the programmer to make it clear, by means of a suitable prompt, how the user will terminate the action of the loop. In Example 4. 1, the following would be a suitable prompt: Write " Enter a number; enter O to quit. In Example 4. 2, the following would be a suitable prompt: Write " Enter the name of your brother or sister:" Write " Enter the word Done to quit. " In the type of loops we used in these two examples, the

loop continues until the user ends it. Other loops end without user input. Regardless of what type f loop you write, you always want to avoid the possibility that the loop will continue without end. Therefore, you must ensure that the test condition can be met and, if the user must enter something special to end the loop, be sure it's clear.

Relational and Logical Operators The condition that determines whether a loop is reentered or exited is usually constructed with the help Of relational and logical operators. We will briefly review these operators here. The following are the six standard relational operators and the programming symbols we will use in this book to represent them: equal to (or " is the same as"): - to equal to: less than: < less than or equal to: greater than: > greater than or equal to: >= prelude to Programming: Concepts and Design, Fifth Edition, by Stewart Event All six operators can be applied to either numeric or character string data.

Note that the double equals sign, the comparison operator (==) is different from the assignment operator While the assignment operator assigns the value on the right side of the equals sign to the variable on the left side, the comparison operator compares the values of the variable or expression on the left side of the operator to the value of the variable, expression, number, r text on the right side. It returns only a value of false (if the two values are different) or true (if the two values are the same).