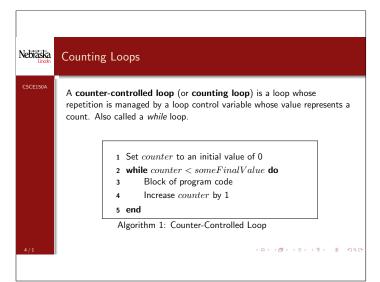


Nebraska Lincoln	Chapter 5
CSCE150A	
	5.1 Repetition in Programs
	5.2 Counting Loops and the While Statement
	5.3 Computing a Sum or a Product in a Loop
	5.4 The for Statement
	5.5 Conditional Loops
	5.6 Loop Design
	5.7 Nested Loops
	5.8 Do While Statement and Flag-Controlled Loops
	5.10 How to Debug and Test
	5.11 Common Programming Errors
2/1	(D) (B) (E) (E) E NQ(

Nebraska Lincoln	Repetition in Programs
CSCE150A	
	Just as the ability to make decisions (<i>if-else</i> selection statements) is an important programming tool, so too is the ability to specify the repetition of a group of operations.
	When solving a general problem, it is sometimes helpful to write a solution to a specific case. Once this is done, ask yourself:
	 Were there any steps that I repeated? If so, which ones? Do I know how many times I will have to repeat the steps? If not, how did I know how long to keep repeating the steps?
3/1	くロ・(ヴ・(ミ・(ミ・ ミ・ウス))

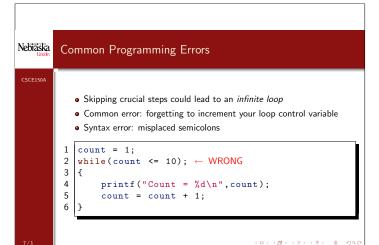
-			
			_
			_



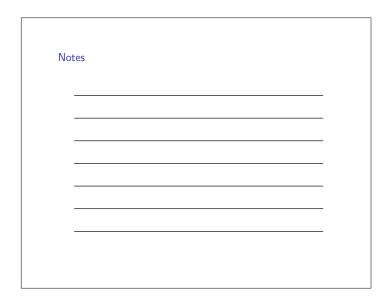
Nebraska Lincoln	WI	nile Loop Syntax
CSCE150A	Sy	ntax of the while Statement:
	ľ	Initialize the loop control variable
		Without initialization, the loop control variable value is meaningless.
		Test the loop control variable before the start of each loop repetition
		Update the loop control variable during the iteration
		 Ensures that the program progresses to the final goal
	1	count = 1;
	2	while(count <= 10)
	3	{
	4	<pre>printf("Count = %d\n", count);</pre>
	5 6	count = count + 1;
	0	}
6/1		<□>→ (B)→ (E)→ (E)→ (E)→ (O)

Not	es	

Not	es	
		-
		-
		- -
		-



Notes



While Loop Exercise Exercise Write a while loop to compute the sum of natural numbers 1 to 100: $\sum_{i=1}^{100} i = 1 + 2 + \dots + 100$ Generalize the loop so that the sum from 1 to any n can be computed. Steps to design: • Identify and define a loop control variable. • Write the syntax for the loop control structure • Fill in the code used within the loop to compute the sum

Not	es	
		-
		-
		-

```
Notes ______
```

Notes			

Nebrasia While Loop Example II Instead of the sum of integers 1 to n, compute the product: $\prod_{i=1}^{100} i = 1 \times 2 \times \ldots \times 100$ What changes need to be made? • Variable names? • Initialized variable value? • Operators? Note: this is the factorial function, $n! = \prod_{i=1}^{n} i$

Notes		
		_
		_
		_
		_
		_
		_

```
While Loop Example II
Nebraska
                                                                                        int product = 1;
                                                                                      int n = 100; /* general variable, may be
                                                                                        * changed or read from input */
int i = 1; /* our loop control variable */
                                                                 3
                                                                 4
                                                                 5 | while (i <= n)
                                                                 6
                                                                                           {
                                                                 7
                                                                                                           product = product * i;
                                                                                                             i = i + 1;
                                                                                           }
                                                                 9
                                                              10 printf("Product 1 to %d is %d\n", n, product);
                                                                                                                                                                                                                                                                                                                                                                4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m > 4 m >
```

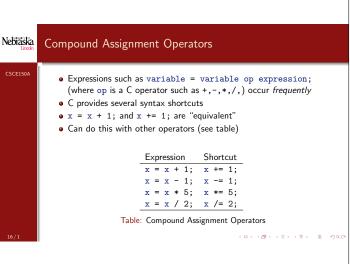
_

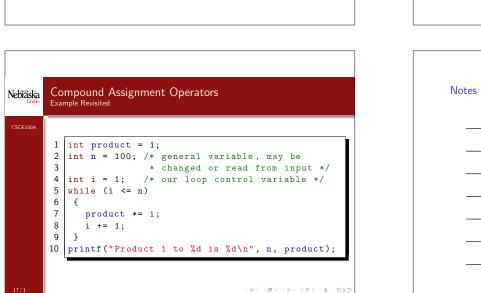
Run the previous program: it gives an answer of 0—why? • Debug your code: use a printf statement in the loop to see what intermediate values are computed: printf("i = %3d product = %d\n",i,product); • Check the answers with a calculator • For what i does this program fail?

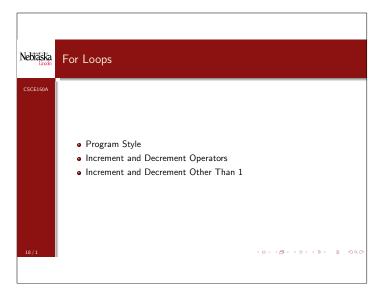
Not	res	

Overflow • We got the wrong answer for i=13, 13!=6,227,020,800• We used a 32-bit integer to store product • Maximum representable value is $2^{31}=2,147,483,648$ • When a number is too large (or too small!) to be represented by its type, overflow occurs (or underflow) • More sophisticated solutions are available, but outside this course's scope

-			
			_
			_

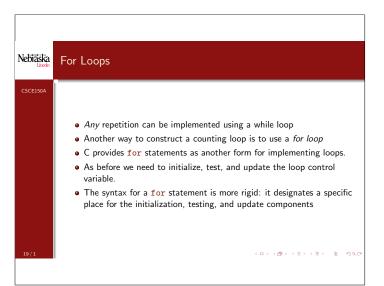






		
-		

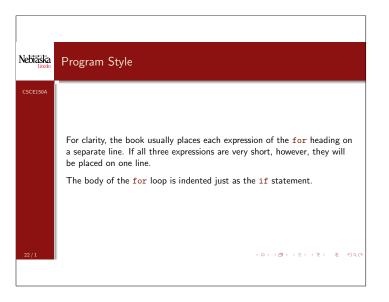
Notes		
		_
		_
		_
		_



Not	res	
		-
		-
		-
		-
		-

Nebraska Lincoln	Increment Operators
CSCE150A	 New syntax: i++ Known as a (postfix) increment "Equivalent" to i = i + 1 Also available: (postfix) decrement: i ("equivalent" to i = i - 1)
21/1	くロン(ラ)(さ)(き) そ かなの

-			



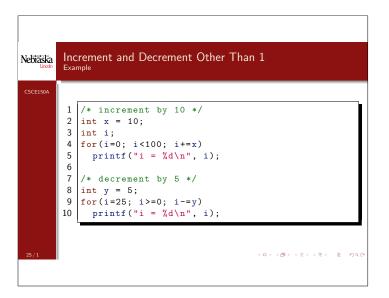
Notes

The counting loops that we have seen have all included assignment expressions of the form • counter = counter + 1 • counter++ • counter += 1 This will add 1 to the variable counter. Using -- will subtract one from the counter.

No	tes			

i	
Nebraska Lincoln	Increment and Decrement Other Than 1
CSCE150A	We can use the "shortcut" compound assignment operators with values other than 1 • Increment operations: sum = sum + x or sum += x, will take the value of sum, add x to it, and then assign the new value to sum • Decrement operations: temp = temp - x or temp -= x, will take the value of temp, subtract x from it and then assign the new value to temp
24/1	(D) (Ø) (E) (E) E HOQC

Not	tes	
		_
		_
		-
		-
		-
		_
		-

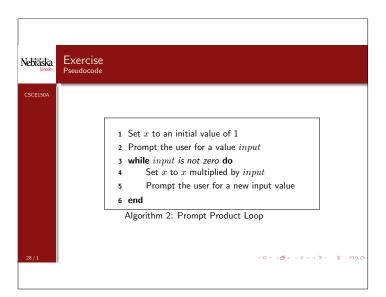


		_
		_
		_

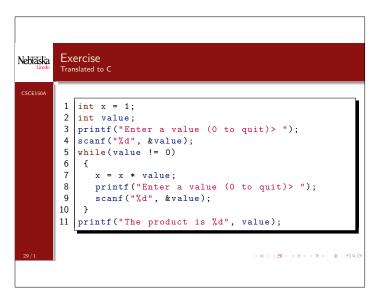
Conditional Loops The exact number of loop repetitions we need to run for a loop will not always be known before loop execution begins. Initialization step? Test? Update action?

Notes		

Exercise
Exercise Create a program that prompts the user for a value x and multiplies it by the previous value of x , storing the result in x , until the user enters a 0 .
(ロ) (B) (본) (본) 본 火 (()

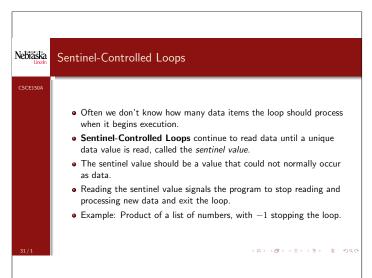


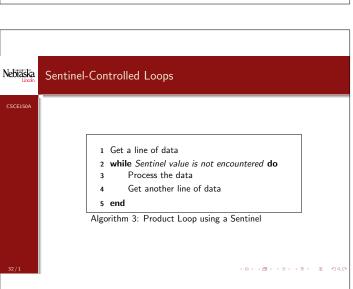


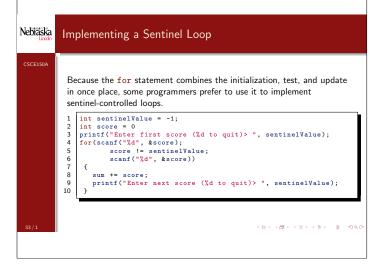


Notes		
		_
		_
		_
		_
		_

Nebraska Lincoln	Loop Design
CSCE150A	
	To this point, we have been analyzing the actions a loop performs. Now, we also want to design our own loops:
	 Sentinel-Controlled Loops Using a for Statement to Implement a Sentinel Loop
30/1	(D) (B) (E) (E) E D)(C



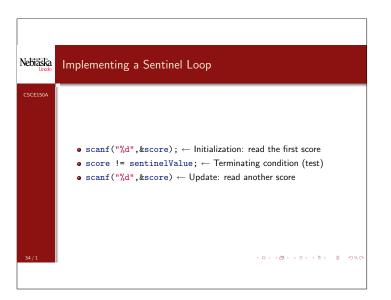


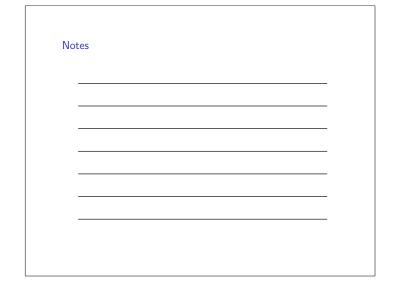


_			_
_			_
_			_
_			_
_			_

Notes		

_			
_			
_			
_			
_			
_			
_			



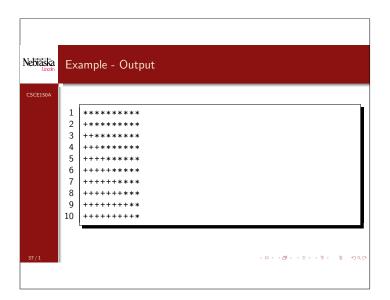


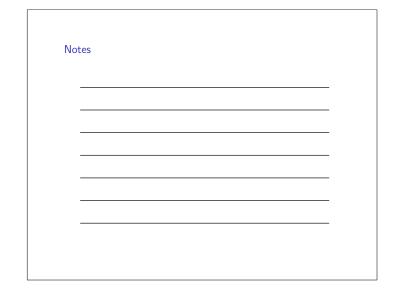
Nebiaska Nested Loops Like if statements, loops can also be nested. • Nested loops consist of an outer loop with or more inner loops. • Each time the outer loop is repeated, the inner loops are reentered. • The inner loop control expressions are reevaluated, and all required iterations are performed.

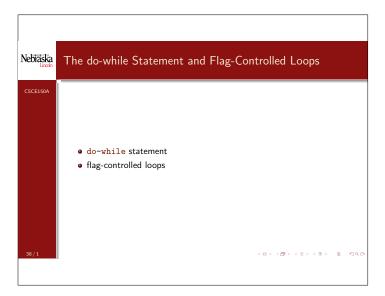
Notes		
Notes		
		_
		_
		_

Nebraska	Ex	ample
CSCE150A	1 2 3 4 5 6 7 8 9 10 11 12	<pre>int i, j; for(i=1; i<=10; i++) { for(j=1; j<=10; j++) { if(j<i) else="" pre="" printf("*");="" printf("+");="" printf("\n");="" }="" }<=""></i)></pre>
36 / 1		(ロ)(費)(き)(き) を かん(や

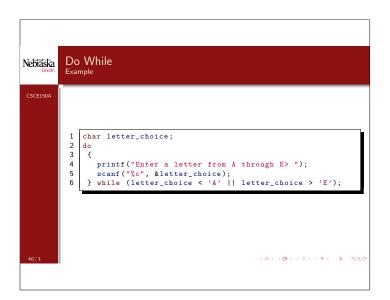
Notes		







Nebřaška Lincoln	Do-While Statement
CSCE150A	
	 The for statement and the while statement evaluate conditions before the first execution of the loop body. In most cases, this pretest is desirable; Prevents the loop from executing when there are no data items to process Prevents execution when the initial value of the loop control variable is outside the expected range.
	 Situations involving interactive input, when we know that a loop must execute at least one time, often use a do-while loop.
39 / 1	(B) (B) (E) (E) E DQC

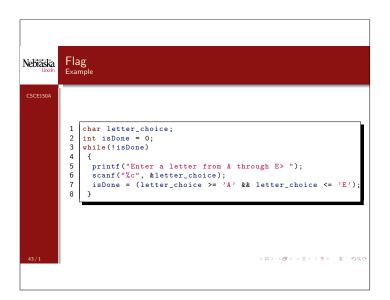


Nebřaska Lincoln	Do While
CSCE150A	
	a Loon begins with do
	Ends with while Careful: Conditional expression does end with a semicolon!
	Conditional is checked at the end of each loop (versus the beginning)
41/1	(D) (B) (E) (E) E MQ(
41/1	 Careful: Conditional expression does end with a semicolon! Conditional is checked at the end of each loop (versus the beginning)

Nebřaška Lincoln	Flag-Controlled Loops
CSCE150A	 Sometimes a loop repetition condition becomes so complex that placing the full expression in its usual spot is awkward. In many cases, the condition may be simplified by using a flag. A flag is a type int variable used to represent whether or not a certain event has occurred. A flag has one of two values: 1 (true) and 0 (false).
42/1	・(B) (B) (B) (B) (B) (B) (B) (B) (B) (B)

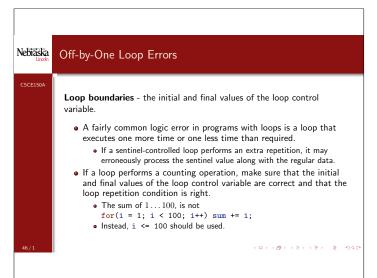
Notes		

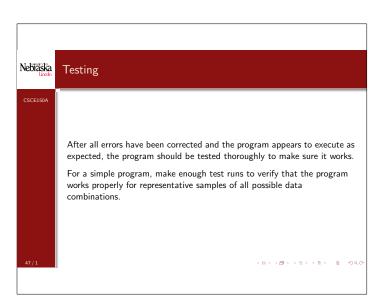


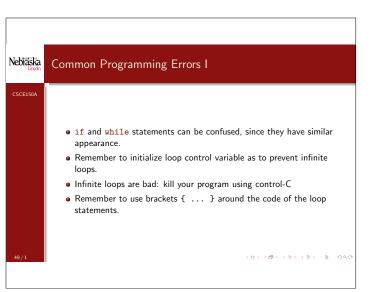
Nebřaska Lincoln	How to Debug and Test Programs
CSCE150A 44/1	 Debugging using printf Off-by-One Loop Errors Testing

Nebraska Lincoln	Debugging using printf
CSCE150A	 Use several printf statements to output the values of your variables to make sure they have the correct value in them as your program executes. It is often helpful to print out the value of your loop control variable to make sure you are incrementing it and will not enter an infinite loop.
45/1	(5) (5) \$ 9Q()

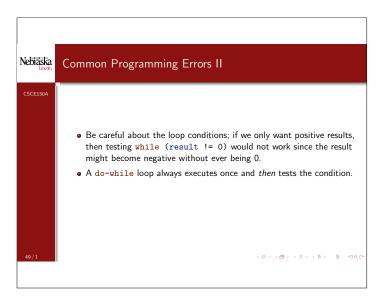
-			

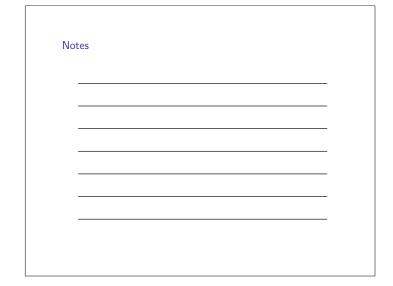






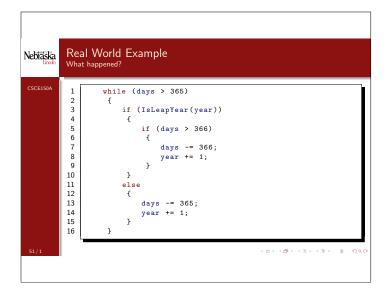
Notes	i	
_		
_		
_		
_		
_		
Notes		
_		
_		
_		
- -		
- - -		
- - - -		
- - - -		



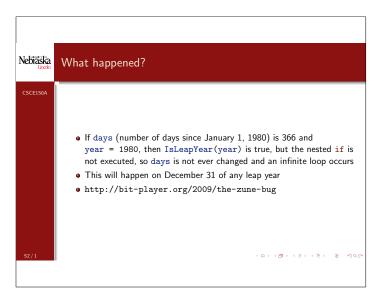


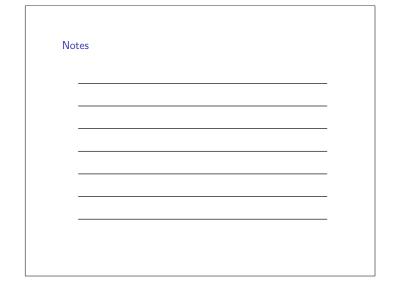
Real World Example SCELISON • Zune Bug: December 31st, 2008 • 2008 was a leap year: 366 days • Thousands of Zunes froze for 24 hours • An embedded module in the Zune contained the following (actual) code

Notes



-		





Nebraska Lincoln	Other Examples
CSCE150A	
	 September 30, 1999: \$125 million Mars orbiter crashes due to lack of converting pounds of thrust to Newtons September 26, 1983: Stanislav Petrov averts nuclear war
	 Russian missle early detection system alerts Petrov of incoming US missles; he correctly assumes that it's a bug since it's a new system and the size of the supposed attack is small
	• Wired Article: History's Worst Software Bugs (http: //www.wired.com/software/coolapps/news/2005/11/69355)
53/1	<□> ⟨♂> ⟨₹> ⟨₹> ₹ • √0,00

Nebraska Lincoln	Questions
CSCE150A	Questions?
54/1	(ロ) (巻) (さ) (き) ぎ かりで

-			