University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Examples of C Programs with Loops

Time for Some Detailed Examples

Let's do some examples of program execution.

Before you can execute a program, you need to **learn how to compile**.

You will learn that **in the lab**.

You should also **take a look at the style guidelines** for the class (see the Wiki).

The examples obey most style rules, but space is tight in slides.

You may want to get out a sheet of paper…

Let's See How This Loop Works

```
/* Print 20 Fibonacci numbers. */
int A = 1; int B = 1; int C; int D;
for (D = 0; 20 > D; D = D + 1) {
    printf ("%d\n", A);
    C = A + B;
    A = B;
    B = C;
}
```

NOTE: Example programs are available online. Feel free to try them before/during/after class.

One Statement/Step at a Time…

comment	A	B	C	D	output
before loop	1	1	bits	bits	
init				0	
20 > D					
print A					1
C = A + B			2		
A = B	1				
B = C		2			
D = D + 1				1	

One Statement/Step at a Time…

comment	A	B	C	D	output
(previous slide)	1	2	2	1	
20 > D					
print A					1
C = A + B			3		
A = B	2				
B = C		3			
D = D + 1				2	

One Statement/Step at a Time…

comment	A	B	C	D	output
(previous slide)	2	3	3	2	
20 > D					
print A					2
C = A + B			5		
A = B	3				
B = C		5			
D = D + 1				3	

One Statement/Step at a Time…

comment	A	B	C	D	output
(previous slide)	3	5	5	3	
20 > D					
print A					3
C = A + B			8		
A = B	5				
B = C		8			
D = D + 1				4	

One Statement/Step at a Time…

comment	A	B	C	D	output
(previous slide)	5	8	8	4	
20 > D					
print A					5
C = A + B			13		
A = B	8				
B = C		13			
D = D + 1				5	

One Statement/Step at a Time…

comment	A	B	C	D	output
(previous slide)	8	13	13	5	
20 > D					
print A					8
C = A + B			21		
A = B	13				
B = C		21			
D = D + 1				6	

Each Loop Iteration Prints One Number

The output column on the last few slides **produces the first twenty numbers** in the Fibonacci sequence (on separate lines, without commas):

1, 1, 2, 3, 5, 8, 13, … , 6765

Steps for a Factorial Printing Program

Remember factorials?

$$N! = N \times (N - 1) \times \ldots \times 1$$

The next program…
- prints a welcome message,
- asks user to enter a number,
- uses **`scanf`** to get the number,
- checks that the user typed something valid,
- calculates the factorial of the user's number,
- and prints the factorial.

Recall that `main` is a Sequence of Statements

When we develop a program,
- we break down the problem into smaller steps,*
- and express each step with **C** statements.

The six steps on the previous slide
- Are written using **C** statements
- And appear in order in **`main`**.

* Part 4 of our class describes a systematic way to do so. Also see P&P Ch. 6.

Before Statements, We Declare Variables

We need two variables.
- In practice, a programmer may decide to declare more variables as they write statements.
- This program is already finished, so we know how many variables it needs…

```
int number;
/* number given by user        */
int factorial;
/* factorial of user's number */
```


How are Variable Names Chosen?

```
int number;
/* number given by user        */
int factorial;
/* factorial of user's number */
```

Variable names
- are **chosen to describe their meaning**,
- but we **use comments** to give further details.

These variable names are all lower-case.
Be consistent in how you use case with variable names in a program.

Use `printf` to Write to the Display

The **first two steps** use `printf`.

```
/* Print a welcome message,
   followed by a blank line. */
printf (">--- Welcome to the
factorial calculator! ---<\n\n");
/* A                          's
   n
printf ("What factorial shall I
calculate for you today? ");
```

Warning: On two lines only on slides.
Do not break format (between quotes) over multiple lines!

Next Step: Wait for the User to Type a Number

After asking the user to enter a number,
- the program **waits for the user**
- **to type a decimal value** using `scanf`.

```
scanf ("%d", &number)
```

The format specifier `%d` tells `scanf` to **convert decimal ASCII to 2's complement**.

The expression `&number` tells scanf to **store the result into the variable `number`**.

Always Check the Return Value!

```
scanf ("%d", &number)
```

Remember that scanf also
- returns **1 if successful** (# of conversions)
- returns **-1 if the user typed something that isn't a decimal number** (such as "hahahaha" … those humans!)

A program can **use the return value** (the value of the **scanf** expression) **to determine what has happened**…

Next Step: Quit if the User Doesn't Behave

```
if (1 != scanf ("%d", &number)) {
   printf ("Only integers, please.\n");
   return 3; /* Program failed. */
}
```

The program **uses an if statement to check the result** of **scanf**.

If the user doesn't type a number, the program…
- **prints an error message**, then
- **terminates** and tells the OS that something went wrong (non-zero by convention).

Time for Some Real Work!

```
for (factorial = number; 1 < number;
     number = number - 1) {
    factorial = factorial *
                (number - 1);
}
```

Note that **C allows you to add extra lines**
- in the middle of **for** loops
- and in expressions
- **to make the code more readable**.

Example: Factorial of 4

comment	factorial	number
before loop	bits	4
init	4	
1 < number		
loop body	12	
number = number - 1		3
1 < number		
loop body	24	
number = number - 1		2

Example: Factorial of 4

comment	factorial	number
(previous slide)	24	2
1 < number		
loop body	24	
number = number - 1		1
1 < number		
after loop	24	1

Second Example: Factorial of 7

comment	factorial	number
before loop	bits	7
init	7	
1 < number		
loop body	42	
number = number - 1		6
1 < number		
loop body	210	
number = number - 1		5

Second Example: Factorial of 7

comment	factorial	number
(previous slide)	210	5
1 < number		
loop body	840	
number = number - 1		4
1 < number		
loop body	2520	
number = number - 1		3

Second Example: Factorial of 7

comment	factorial	number
(previous slide)	2520	3
1 < number		
loop body	5040	
number = number - 1		2
1 < number		
loop body	5040	
number = number - 1		1

Second Example: Factorial of 7

comment	factorial	number
(previous slide)	5040	1
1 < number		
after loop	5040	1

Last Step: Print the Answer

```
printf ("\nThe factorial is %d.\n",
        factorial);
```

The format specifier **`%d`** tells **`printf`** to **convert 2's complement to decimal ASCII**.

The variable **`factorial` is the expression to be printed**.

Then the program **terminates (successfully)**: **`return 0;`**

