

# Loops

CSE 130: Introduction to C Programming  
Spring 2005

1

# What We've Covered

- Algorithms
- Simple programs
- Variables and data types
- Expressions

2

# What's Coming Up

- Conditional (decision) statements
- Arrays
- Functions and recursion

3

# Iterative Programming

- Many programs perform the same task many times
  - Operations are repeated on different data
- Ex. Adding a list of numbers
- Ex. Displaying a menu of options
- Repetitive tasks are specified using loops

4

# Loop Elements

- All loop constructs share four basic elements:
  1. Initialization
  2. Testing the loop condition
  3. Loop body (the task to be repeated)
  4. Loop update
- The order of these elements may vary

5

# Initialization

- This section of code is used to set starting values
- For example, setting a total to 0 initially
- This can be done as part of the loop, or separately before the loop code begins

6

## Loop Tests

- Test expressions are used to determine whether the loop should execute (again)
- Tests compare one value/variable with another
- If the test evaluates to TRUE, then the loop will execute another time

7

## Test Operators

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Is equal to
!=	Is not equal to

8

## Loop Update

- This step changes the value(s) of the loop variable(s) before the loop repeats
- Ex. moving to the next item to process
- This can be done explicitly as part of the loop, or it can be done inside the loop body

9

## Choosing a Loop Type

- For a fixed number of iterations:
  - for loops are the way to go
- For a variable number of iterations:
  - while loops can execute 0 or more times
  - do...while loops execute at least once
- Any loop can be rewritten as another type

10

## for Loops

- for loops execute a fixed number of times
- Order of execution:
  1. Initialization
  2. Loop condition test
  3. Loop body
  4. Loop update

11

## General Form

```
for ( initialization ;  
      loop condition test ;  
      loop update )  
{  
    loop body  
}
```

12

## for Loop Example

```
int i;
for (i = 0; i < 10; i++)
{
    printf("%d ", i);
}
```

13

## Loop Output

0 1 2 3 4 5 6 7 8 9

14

## Another Example

```
int nextNumber, i, sum = 0;
for (i = 0; i < 5; i++)
{
    printf("\nEnter a number: ");
    scanf("%d ", nextNumber);
    sum += nextNumber;
}
```

15

i	nextNumber	sum
-	-	0
0	2	2
1	15	17
2	5	22
3	7	29
4	3	32
5	-	32

16

## Fancier Loop Headers

- We can include multiple initialization or update statements in a loop header
  - Separate each statement with a comma
- Ex. for (i = 0, j = 1; i < 5; i++, j--) { }
- Loop headers can also include calls to functions
- A for loop can also omit part of the header
  - Ex. for (; i < 5; i++) { }

17

## while Loops

- while loops execute any number of times
- Order of execution:
  1. Initialization
  2. Loop condition test
  3. Loop body
  4. Loop update

18

## General Form

```
initialization
while ( loop condition test )
{
    loop body
    loop update
}
```

19

## while Loop Example

```
int countDown = 5;
while (countDown >= 0)
{
    printf("%d...", countDown);
    countDown--;
}
```

20

## Loop Output

5...4...3...2...1...0...

21

## Another Example

```
int root = 0;
while (root < 10)
{
    root += 1;
    printf("%d * %d = ", root, root);
    printf("%d\n", root * root);
}
```

22

root	output
0	1 * 1 = 1
1	2 * 2 = 4
2	3 * 3 = 9
3	4 * 4 = 16
4	5 * 5 = 25
5	6 * 6 = 36
6	7 * 7 = 49
7	8 * 8 = 64
8	9 * 9 = 81
9	10 * 10 = 100

23

## do...while Loops

- Like while loops, but execute at least once
- Order of execution:
  1. Initialization
  2. Loop body
  3. Loop update
  4. Loop condition test

24

## General Form

```
initialization
do
{
    loop body
    loop update
} while ( loop condition test ) ;
```

25

## do...while Example

```
/* print numbers from 1-15 */

int counter = 1;
do
{
    printf("%d\n", counter);
    counter++;
} while (counter < 16);
```

26

## Loop Output

```
1
2
3
4
5
...
13
14
15
```

27

## do...while Example

```
int sum = 0, value = 0;
do
{
    sum += value;
    printf("\nEnter a # (-1 to quit): ");
    scanf(" %d", &value);
} while (value != -1);
```

28

sum	value	value (at end)
0	0	5
5	5	3
8	3	7
15	7	-1

29

## Nested Loops

- The body of a loop can contain any other type of statement(s)
  - This includes other loops
- If the outer loop executes  $n$  times, and inner loop executes  $m$  times, then inner loop body executes  $n \times m$  times

30

## An Example

```
int i, j;
for (i = 0; i < 4; i++)
{
    for (j = 0; j < 4; j++)
    {
        printf("*");
    }
    printf("\n");
}
```

31

## Example Output

```
****
****
****
****
```

32

## Another Example

```
int i, numStars = 1;
do
{
    for (i = 0; i < numStars; i++)
    { printf("*"); }
    printf("\n");
    numStars++;
} while (numStars < 10);
```

33

## Example Output

```
*
**
***
****
*****
******
*******
*****
****
***
**
*
```

34

## Next Time

- Conditional statements
- Arrays

35