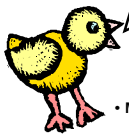
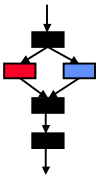
We're Doomed! Doomed!





- Machine's execution speed
  - 10,000,000 lines of C per second
- Programmer speed
  - 12 lines of working C per day
- How many days will it take to write a program that runs for 1 second?

H1-1

CSE 142
Computer Programming I

Iteration

or… How we *really* get the computer to do our work for us.

H1-2

(in which we catch up with Turing!)

Outline

Iteration: why do we need it?
What are loops?
How do we write loops in C?
How do we go about writing loops?
Some examples
Nested loops
Other ways to write loops
Dangers and devices

H1-3

Chapter 5

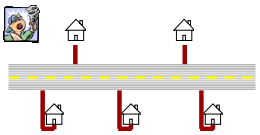
Read Sections 5.1-5.6, 5.10

- **5.1 Introduction**
- **5.2-5.3 While statement**
- **5.4 For statement**
- **5.5-5.6 Loop design**
- **5.7 Nested Loops**
- **5.11 Common errors**

H1-4

Revisiting Our Paper Route

Go to house #1
Deliver a paper to #1
Go to house #2
Deliver a paper to #2
Go to house #3
Deliver a paper to #3
…



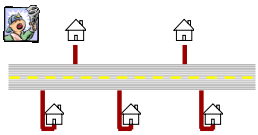
This is better than before we had functions.

But, it's still cumbersome. Why?

H1-5

Revisiting Our Paper Route

For every house on the block:
  Go to the house
  Deliver a paper to it



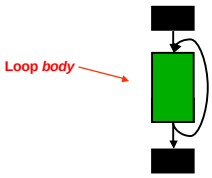
We really want to repeat the same process for each house in order.

We want to ***loop*** over the houses.

H1-6

What are Loops?
One More Type of Control Flow

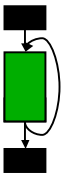
When we want to repeat a block of code, we use a *loop*.



Help! I'm going round and round. How do I get out?

H1-7

Loops



- A "loop" is a repeated ("iterated") sequence of statements
- Like conditionals, loops let us control the flow of our program in powerful ways.
- Like functions, loops take code that has been *generalized* and execute it many times.

H1-8

(More) Robust Input

```
char choice = 'x';
printf("Do you want to switch doors? (y/n) ");
scanf (" %c", &choice);
```

H1-9

Test: If it didn't work, try again *until it does*.

Getting Loopy

initialization

loop condition

loop body (with "update")

```
char choice;
printf("Do you want to switch doors? (y/n) ");
scanf (" %c", &choice);
while choice is neither 'y' nor 'n'
printf("Do you want to switch doors?\n");
printf("Please just enter 'y' or 'n'! ");
scanf (" %c", &choice);
```

H1-10

How do we write loops in C?
The while statement.

```
char choice;
printf("Do you want to switch doors? (y/n) ");
scanf (" %c", &choice);
while (choice != 'y' && choice != 'n') {
        printf("Do you want to switch doors?\n");
        printf("Please just enter 'y' or 'n'! ");
        scanf (" %c", &choice);
}
```

H1-11

while Statement Syntax

Loop condition

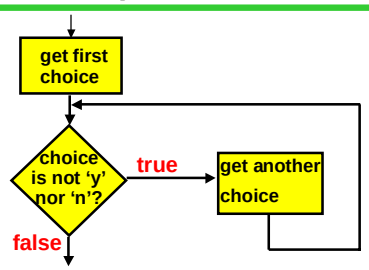
```
while ( condition ) {
   statement1;
   statement2;
   …
}
```

Loop body: Any statement, or a compound statement

H1-12

Psst… just like with conditionals, a while's body is really a statement. So, why are there two statements? And, why are those braces there?

while Loop Control Flow



H1-13

A Loopless Problem (?)

Problem: add 4 numbers entered at the keyboard.

```
int sum;
int x1, x2, x3, x4;

printf("Enter 4 numbers: ");
scanf("%d%d%d%d", &x1, &x2, &x3, &x4);
sum = x1 + x2 + x3 + x4;
```

This works perfectly!
But... what if we had 14 numbers? or 40? or 4000?

H1-14

How do we go about writing loops? *Generalizing!*

Problem: read a series of numbers entered at the keyboard and add all of them.

The key to solving problems with loops is to figure out how to do one or a few concrete steps… then generalize.

Our algorithm for adding four numbers was concrete. It had no repeated statements at all…

But it did have some repetition buried in it.

H1-15

Let's rework the algorithm to make the repetition more explicit… then, we can solve the general problem.

Add 4 Numbers, Repetitively

```
int sum,   x;
sum = 0;
printf("Enter 4 numbers: ");

scanf("%d", &x);
sum = sum + x;

scanf("%d", &x);
sum = sum + x;

scanf("%d", &x);
sum = sum + x;
```

H1-16

```
scanf("%d", &x);
sum = sum + x;
```

Loop to Add 4 Numbers

```
int sum,   x;
sum = 0;
printf("Enter 4 numbers:");

scanf("%d", &x);
sum = sum + x;

scanf("%d", &x);
sum = sum + x;

scanf("%d", &x);
sum = sum + x;

scanf("%d", &x);
sum = sum + x;
```

```
int sum,  x;
int count;

sum = 0;
printf("Enter 4 numbers:");

count = 1;
while (count <= 4) {
    scanf("%d", &x);
    sum = sum + x;
    count = count + 1;
}
```

H1-17

More General Loop to Add Numbers

```
int sum, x, count;
int number_inputs;          /* Number of inputs */

sum = 0;
printf("How many numbers? ");
scanf("%d", &number_inputs);
printf("Enter %d numbers: ", number_inputs);
count = 1;                                  <- initialization
while ( count <= number_inputs ) {          <- condition
   scanf("%d", &x);
   sum = sum + x;
   count = count + 1;                       <- update
}
```

H1-18

Examples: Compute 7!

What is 1 * 2 * 3 * 4 * 5 * 6 * 7? ("seven factorial")

```
x = 1 * 2 * 3 * 4 * 5 * 6 * 7;
printf ( "%d", x ) ;
```

Bite size pieces:	More Regular:	As a loop:
x = 1;	x = 1; i = 2;	x = 1;
x = x * 2;	x = x * i; i = i + 1;	i = 2;
x = x * 3;	x = x * i; i = i + 1;	while (i <= 7) {
x = x * 4;	x = x * i; i = i + 1;	x = x * i;
x = x * 5;	x = x * i; i = i + 1;	i = i + 1;
x = x * 6;	x = x * i; i = i + 1;	}
x = x * 7;	x = x * i; i = i + 1;	

H1-19

Tracing the Loop

```
/* What is 1 * 2 * 3 * ...*7  */
x = 1 ;              /* A */
i = 2 ;              /* B */
while ( i <= 7 ) {   /* C */
    x = x * i ;      /* D */
    i = i + 1 ;      /* E */
}                    /* F */
printf ( "%d", x ) ; /* G */
```

line	i	x	i≤7?
A	?	1	
B	2	1	
C	2	1	T
D	2	2	
E	3	2	
C	3	2	T
.....	.....	.....	
C	6	120	T
D	6	720	
E	7	720	
C	7	720	T
D	7	5040	
E	8	5040	
C	8	5040	F
G	(Print 5040)		

H1-20

Double Your Money

/* Suppose your $1,000 is earning interest at 5% per year. How many years until you double your money? */

```
my_money = 1000.0;
n = 0;
while ( my_money  <  2000.0 ) {
    my_money  =  my_money *1.05;
    n = n + 1;
}
printf( "My money will double in %d years.", n);
```

H1-21

Average Inputs

```
printf ( "Enter values to average, end with -1.0 \n") ;
sum  =  0.0 ;
count  =  0 ;
scanf ( "%lf", &next ) ;
while ( next  !=  -1.0 )  {
    sum   = sum + next ;
    count = count + 1;
    scanf ( "%lf", &next ) ;
}
if  (count  >  0)
    printf( "The average is %f. \n",
            sum / (double) count );
```

sentinel

H1-22

Printing a 2-D Figure

How would you print the following diagram?

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

repeat 3 times
print a row of 5 stars

repeat 5 times
print *

It seems as if a loop within a loop is needed

H1-23

Is that allowed?
What can go inside loops?

Nested Loop

```
#define ROWS    3
#define COLS    5
...
row = 1;
while ( row  <=  ROWS ) {
   /* print a row of COLS *'s */
      ...
   row =  row + 1;
}
```

H1-24

Nested Loop

```
row = 1;
while ( row  <=  ROWS ) {
   /* print a row of COLS *'s */
   col = 1;
   while (col <= COLS) {
      printf("*");
      col = col + 1;
   }
   printf( "\n" );
   row =  row + 1;
}
```

outer loop: print 3 rows

inner loop: print one row

H1-25

Trace

row: 1 2 3 4

col: 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6

output:

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

```
row = 1;
while ( row  <=  ROWS ) {
   /* print a row of 5 *'s */
   col = 1;
   while (col <= COLS) {
      printf("*");
      col = col + 1;
   }
   printf( "\n" );
   row =  row + 1;
}
```

H1-26

Print a Multiplication Table

	1	2	3
1	1	2	3
2	2	4	6
3	3	6	9
4	4	8	12

	1	2	3
1	1 * 1	1 * 2	1 * 3
2	2 * 1	2 * 2	2 * 3
3	3 * 1	3 * 2	3 * 3
4	4 * 1	4 * 2	4 * 3

H1-27

How should we start?

First, let's try a particular row…

	1	2	3
1	1	2	3
2	2	4	6
3	3	6	9
4	4	8	12

Print Row 2

```
col = 1;
while (col <= 3) {
   printf("%4d", 2 * col );
   col = col + 1;
}
printf("\n");
```

row number

H1-28

Now, Generalize!

Print 4 rows

Print one row

```
row = 1;
while (row <= 4)  {
   col = 1;
   while (col <= 3) {
      printf("%4d",  row  * col );
      col = col + 1;
   }
   printf("\n");
   row = row + 1;
}
```

H1-29

Loop Trace

row	col	
1	1	print 1
	2	print 2
	3	print 3
		print \n
2	1	print 2
	2	print 4
	3	print 6
		print \n

row	col	
3	1	print 3
	2	print 6
	3	print 9
		print \n
4	1	print 4
	2	print 8
	3	print 12
		print \n

H1-30

Notes About Loop Conditions

- They offer all the same possibilities as conditions in *if*-statements
  - Can use &&, ||, !
- Condition is reevaluated each time through the loop
- A common loop condition: checking the number of times through the loop

H1-31

Counting Loops

- A common loop condition: checking the number of times through the loop
- Requires keeping a "counter"
- This pattern occurs so often there is a separate statement type based on it: the *for*-statement

H1-32

A *for* Loop

```
/* What is 1 * 2 * 3 * ... * n ? */

x = 1 ;
i = 2 ;
while ( i <= n ) {
    x = x * i ;
    i = i+1;
}
printf ( "%d", x ) ;
```

```
x = 1 ;

for ( i = 2 ; i <= n ; i = i+1 ) {
    x = x * i ;
}

printf ( "%d", x) ;
```

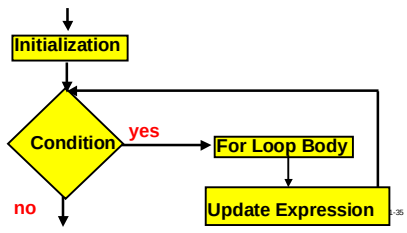
H1-33

for Statement Syntax

```
for ( initialization;
      condition;
      update expression) {
    statement1;
    statement2;
    ...
}
```

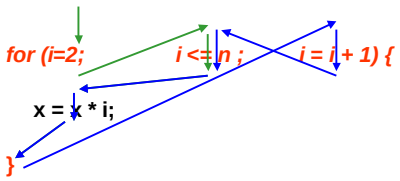
H1-34

for Loop Control Flow



H1-35

Control Flow: *for*



H1-36

for Loops vs *while* Loops

Any for loop can be written as a while loop

These two loops mean exactly the same thing:

```
for (initialization; condition; update)
        statement;

initialization;
while (condition) {
        statement;
        update;
}
```


Counting in *for* Loops

```
/* Print n asterisks */
for ( count = 1 ; count <= n ;  count = count + 1 ) {
    printf ( "*" ) ;
}
```

```
/* Different style of counting */
for (  count = 0 ; count < n ;  count = count + 1 ) {
    printf ( "*" );
}
```


"3 Rows of 5" as a Nested *for* Loop

```
#define ROWS    3
#define COLS    5
...
for ( row = 1;  row  <=  ROWS ;  row =  row + 1 ) {
    for ( col = 1 ;  col <= COLS ;  col = col + 1 ) {
        printf( "*");
    }
    printf( "\n" );
}
```

inner loop: print one row

outer loop: print 3 rows

Yet Another 2-D Figure

How would you print the following diagram?

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

For every row (row = 1, 2, 3, 4, 5)

Print row stars

Solution: Another Nested Loop

```
#define ROWS 5
...
int  row, col ;
for ( row = 1 ;  row <= ROWS ;  row = row + 1 ) {
    for ( col = 1 ;  col <= row ;  col = col + 1) {
        printf( "*" ) ;
    }
    printf( "\n" );
}
```


Yet One More 2-D Figure

How would you print the following diagram?

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

For every row (row = 0, 1, 2, 3, 4)

Print row spaces followed by (5 - row) stars

Yet Another Nested Loop

```
#define ROWS 5
...
int  row, col ;
for ( row = 1 ;  row <= ROWS;  row = row + 1 ) {
    for ( col = 1 ;  col <= row - 1;  col = col + 1)
        printf( "  " ) ;
    for ( col = row ;  col <= ROWS;  col = col + 1)
        printf( "*" ) ;
    printf( "\n" );
}
```

H1-43

Generalizing Ever More

```
/* Print character ch n times */
void RepeatChars ( int n, char ch) {
    int i ;
    for ( i = 0 ;  i < n ;  i = i + 1 )
        printf ( "%c", ch) ;
}
...
for ( row = 1 ;  row <= ROWS ;  row = row + 1 ) {
    RepeatChars ( row - 1,  ' ' ) ;
    RepeatChars ( ROWS – row + 1, '*' ) ;
    printf( "\n" );
}
```

H1-44

Remember PrintBannerLines?

```
/* Print N rows of asterisks */
void PrintBannerLines ( int lines ) {
    int i ;
    for ( i = 0 ;  i < lines ;  i = i + 1 ) {
        RepeatChars(20, '*');
    }
}
```

H1-45

Some Loop Pitfalls

```
while ( sum < 10 ) ;
    sum = sum + 2;

                for ( i = 1; i <= 10; i = i + 1);
                    sum = sum + i ;

for ( i = 1;  i != 10 ; i = i + 2 )
    sum = sum + i ;
```

H1-46

Double Danger

```
double x ;
for ( x = 0.0 ; x < 10.0 ; x = x + 0.2 )
    printf("%.18f", x) ;
```

Seems harmless...

H1-47

Double Danger

What you expect:	What you might get:
0.000000000000000000	0.000000000000000000
0.200000000000000000	0.200000000000000000
0.400000000000000000	0.400000000000000000
...	...
9.000000000000000000	8.999999999999999997
9.200000000000000000	9.199999999999999996
9.400000000000000000	9.399999999999999996
9.600000000000000000	9.599999999999999996
9.800000000000000000	9.799999999999999996
	9.999999999999999996

H1-48

Use *int*s as Loop Counters

```
int i ;
double x ;
for ( i = 0 ; i < 50 ; i = i + 1 )
{
    x = (double) i / 5.0 ;
    printf("%.18f", x) ;
}
```

H1-49

Counting in Loops

Counting up by one or down by one:

```
for ( i = 1 ;  i <= limit ;  i = i+1 ) { . . . }

times_to_go = limit;
while ( times_to_go > 0 ) {
    • • •
    times_to_go = times_to_go - 1;
}
```

H1-50

Counting Up or Down by 1

This pattern is so common there is special jargon and notation for it

To "increment:" increase (often by 1)
To "decrement:" decrease (often by 1)

C operators:
- Post-increment (x++): add 1
- Post-decrement (x--): subtract 1

H1-51

Handy Shorthand x++ x--

Used by itself,
- x++ means the same as x = x+1
- x-- means the same as x = x-1

Very often used with loop counters:

```
for ( i=1 ;  i <= limit ;  i++ ) { . . . }

times_to_go = limit;
while ( times_to_go > 0 ) {
    . . .
    times_to_go--
```

H1-52

...

Surgeon General's Warning

++ and -- are unary operators.

Pre-increment (++x) and pre-decrement (--x) exist, too.

In this course, use ++ and -- only in isolation.

Don't combine these with other operators in expressions! E.g., don't try

```
x = y++ / (3 * --x--)
```

H1-53

Iteration Summary

General pattern:
Initialize, test, do stuff, repeat . . .

"while" and "for" are equally general in C

Use "for" when initialize/test/update are closely related and simple, especially when counting

H1-54

Looking Ahead

We'll talk more about how to design loops

We'll discuss complex conditional expressions

- Can be used with loops as well as in conditional statements

We'll see "arrays", a powerful new way of organizing data

- Very often used with loops

H1-55

QOTD: Counting Crows

A vital part of using loops is to understand the structure of the data you're looping over.

In particular, the loops we look at require "serializing" the data: giving it an order and going through it one at a time.

Serialize the following data:

- A murder of crows on a wire
- The rooms on your house (for vacuuming!)
- The integers (to check for primes!)
- All integral, positive (x, y) coordinates

H1-56

There's something tricky about the last one. What is it???