Flow of Control – Part 2
Repetition Statements

Repetitive Control Flow in C

- Programs often must repeat different instructions in a variety of situations
  - sometimes, code must be repeated a determinate number of times
  - other times, code must be repeated an indeterminate number of times based on a condition

The `while` Statement

- Indeterminate Loop
  - Repeat While A Condition Is True

```
while ( logical-expression ) {
   ...block of statements...
}
```

The `while` Statement

- Indeterminate Loop

```
while (x < y) {
   printf(“x < y”);
   x++;
}
```

The `while` Statement

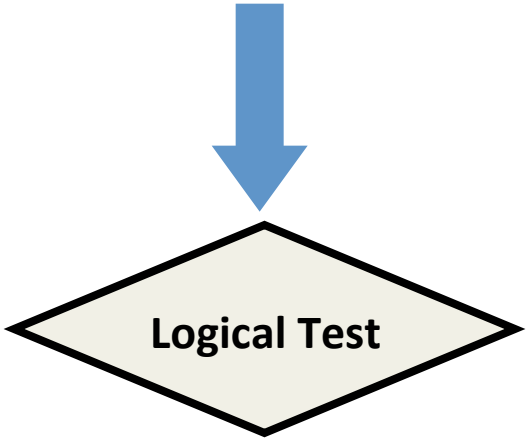
- Indeterminate Loop

```
while (x < y) {
   printf(“x < y”);
   x++;
}
```

The `while` Statement

- Indeterminate Loop

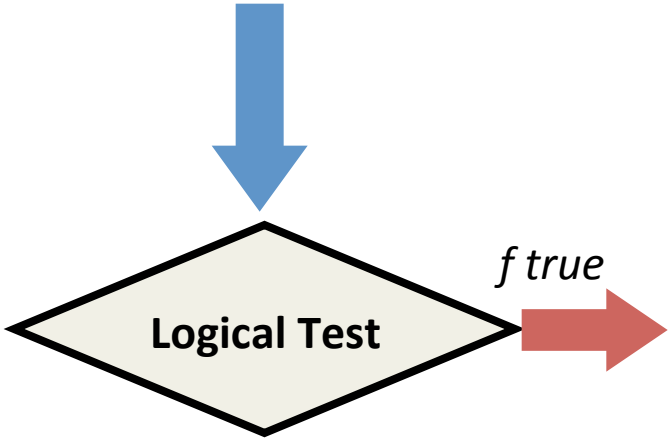
```
while (x < y) {
   printf("x < y");
   x++;
}
```



The `while` Statement

- Indeterminate Loop

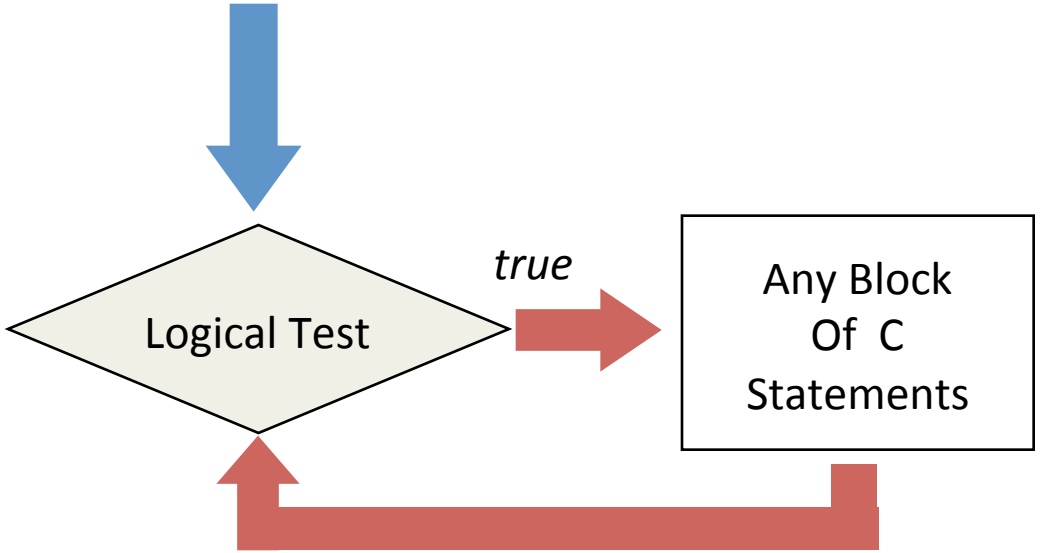
```
while (x < y) {
    printf("x < y");
    x++;
}
```



The `while` Statement

- Indeterminate Loop

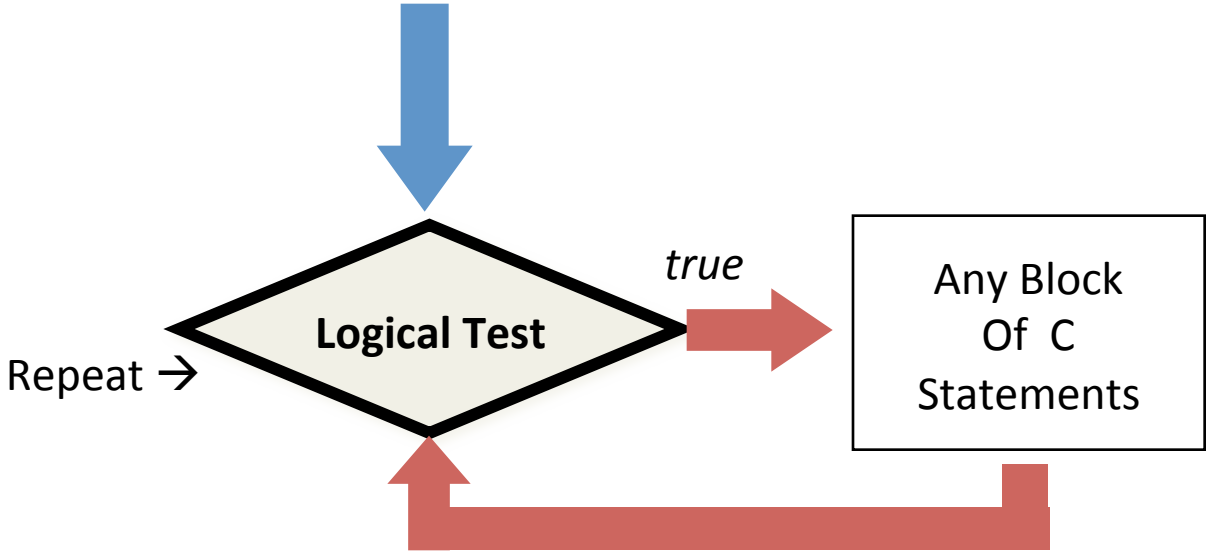
```
while (x < y) {
    printf("x < y");
    x++;
}
```



The `while` Statement

- Indeterminate Loop

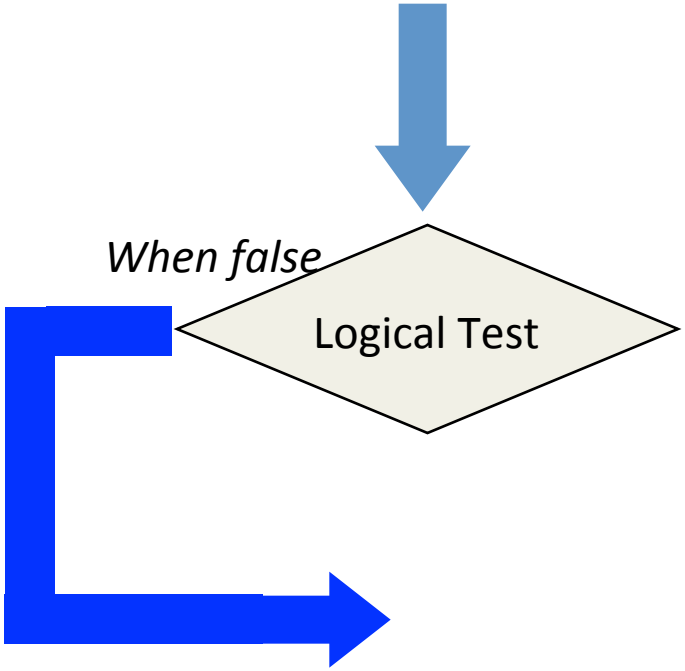
```
while (x < y) {
    printf("x < y");
    x++;
}
```



The `while` Statement

- Indeterminate Loop

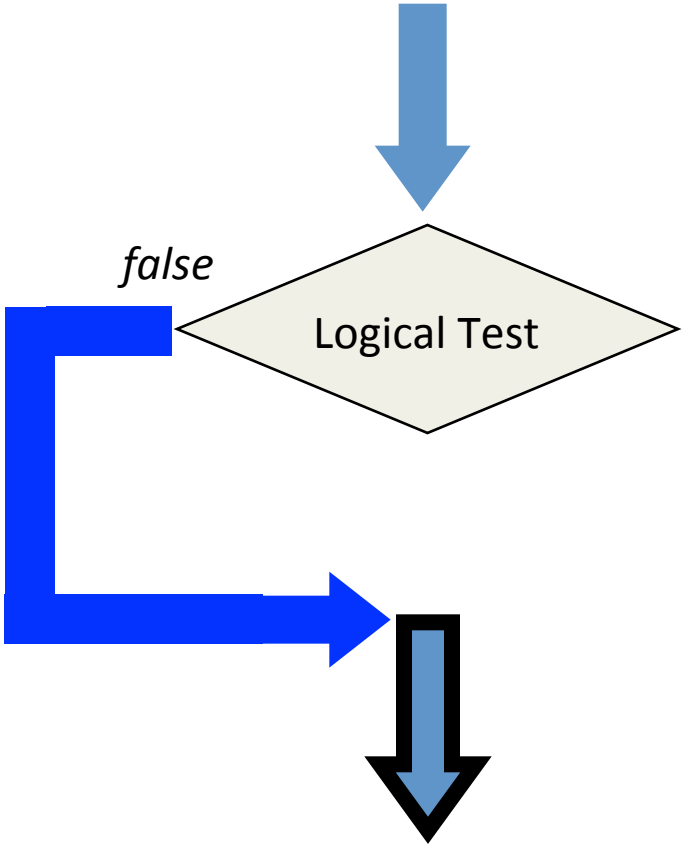
```
while (x < y) {
    printf("x < y");
    x++;
}
```



The `while` Statement

- Indeterminate Loop

```
while (x < y) {
   printf("x < y");
   x++;
}
// when x<y is false
// compiler moves
// to this line
```



The `do...while` Statement

- Indeterminate Loop
  - Repeat While A Condition Is True

```
do {
   ...block of statements...
} while ( logical-expression );
```

The `do...while` Statement

- Indeterminate Loop

```
do {
   printf(“x < y”);
   x++;
} while (x < y);
```

The `do...while` Statement

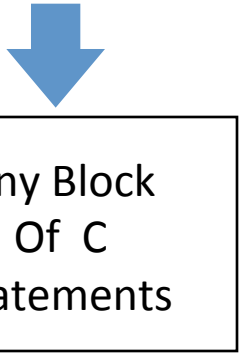
- Indeterminate Loop

```
do {
   printf(“x < y”);
   x++;
} while (x < y);
```

The `do...while` Statement

- Indeterminate Loop

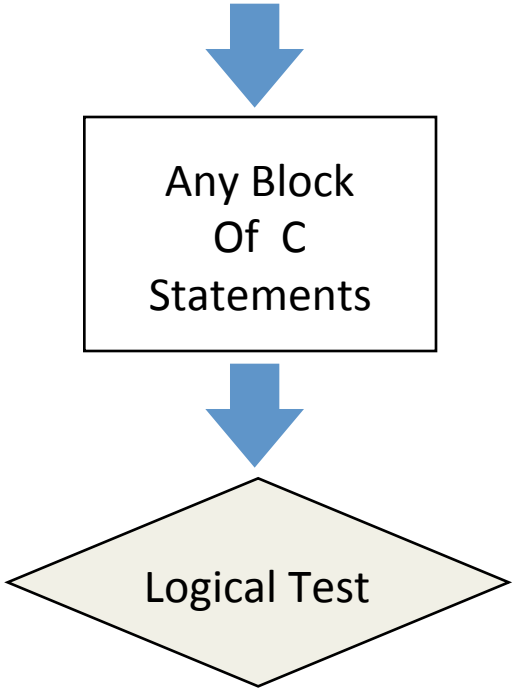
```
do {
   printf(“x < y”);
   x++;
} while (x < y);
```



The `do...while` Statement

- Indeterminate Loop

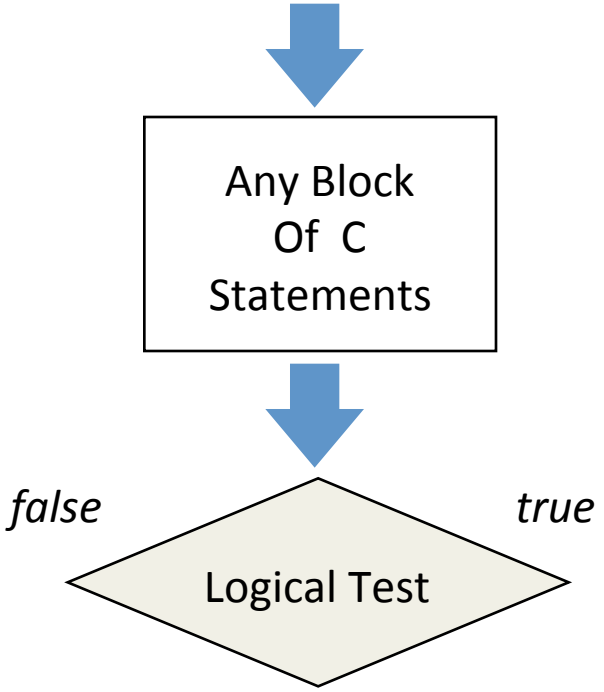
```
do {
   printf("x < y");
   x++;
} while (x < y);
```



The `do...while` Statement

- Indeterminate Loop

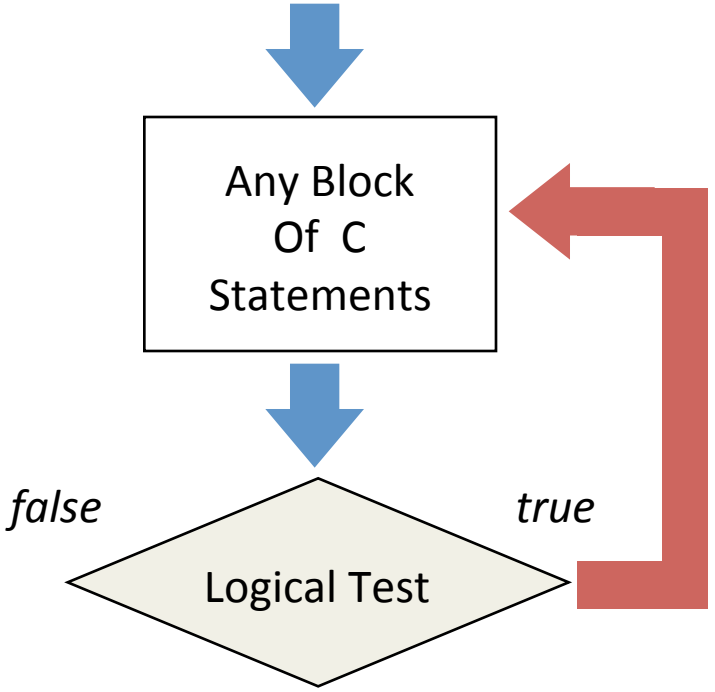
```
do {
   printf("x < y");
   x++;
} while (x < y);
```



The `do...while` Statement

- Indeterminate Loop

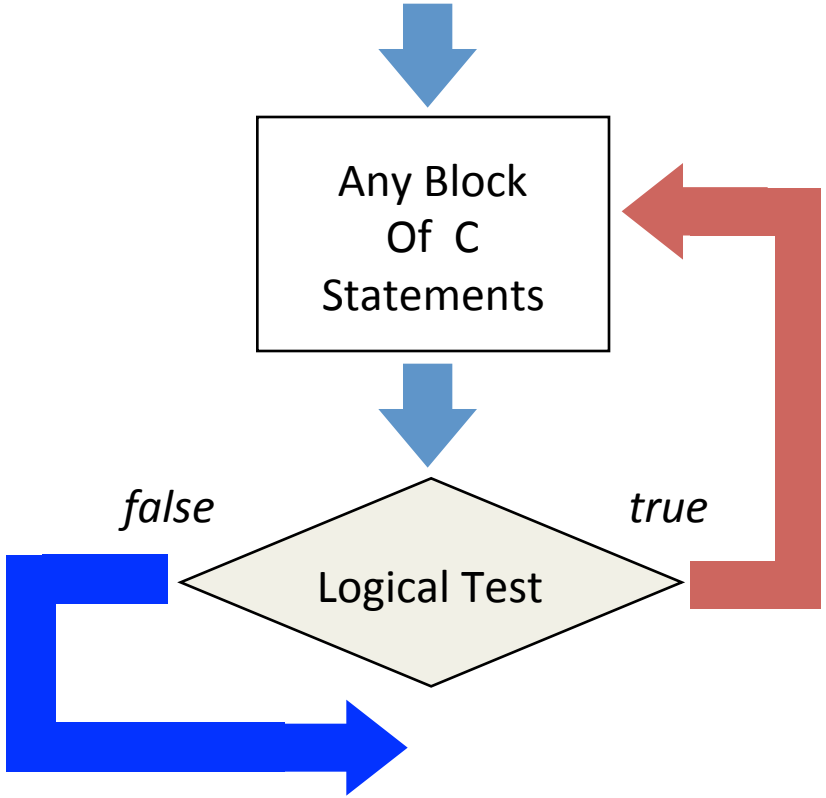
```
do {
   printf("x < y");
   x++;
} while (x < y);
```



The `do...while` Statement

- Indeterminate Loop

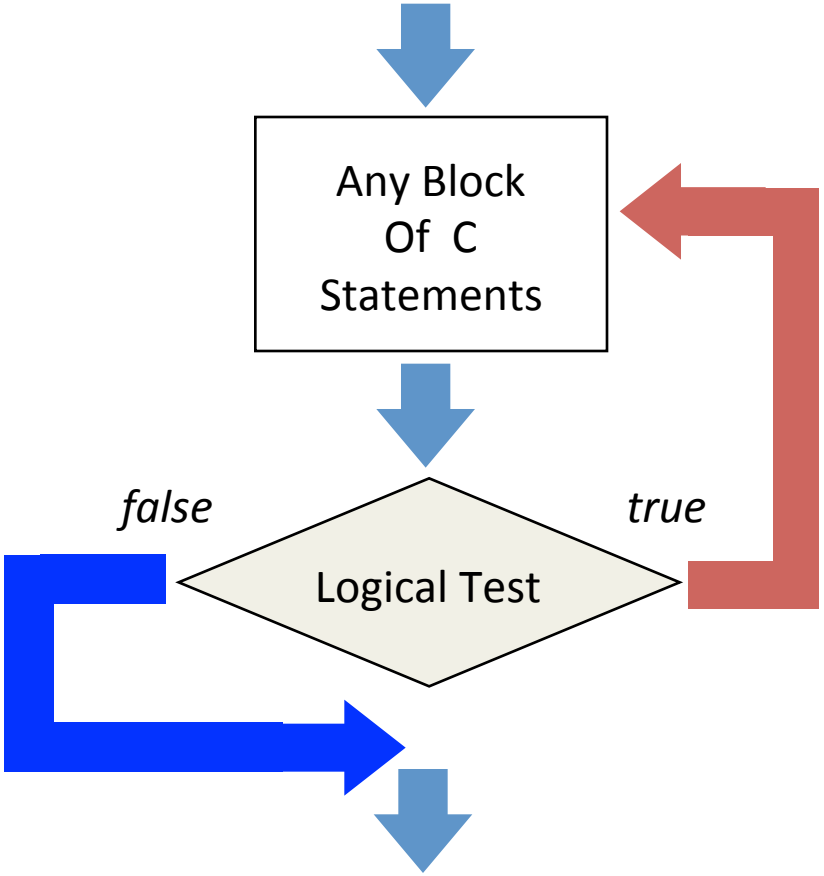
```
do {
   printf("x < y");
   x++;
} while (x < y);
```



The `do...while` Statement

- Indeterminate Loop

```
do {
   printf("x < y");
   x++;
} while (x < y);
```



Loops Demo!

```
/*
  Let's try using finding odd numbers...
 */
#include <stdio.h>

int main() { // main brace
        int start = 0, stop = 0; //demo multi var dec
        int value = 0;
        /*
          For testing purposes, output a prompt
           for the data you are trying to read.
    */
         printf( “finding odd number between\n" );
        printf( "the two values you enter\n" );
        printf( ”enter your first value:" );
        scanf( "%d", &start );
        printf( ”enter your second value:" );
        scanf( "%d", &stop );
        /*
          Depending on the values you enter, the loop
          may run, or not...
      */
        value = start;
        while (value <= stop)
         {  // brace 1 open
                /* is the value odd?? */
                if (value %2 == 1) { //brace 2 open
                        printf( “odd number: %d\n", value );
                } //closing brace 2
                value = value + 1;
        }  //closing brace 2
        return( 0 );
} //closing main brace
```

Summarizing Loops Demo!

- Typically, one of the loop forms fits your problem better than the other
- However, any loop written in one form can be re-written in the other
- Coming up with the code for a loop comes with practice, so practice many examples
- The next few slides have the logic − you write the code

Examples 1

Loop to add entries from user then showing their average; user enters 0 to mark the end of data:

do

//get entry

//add entry to running total: t=t+entry

//count the number of entries: c=c+1 OR c++

//loop while entry <> 0

//after loop show total/c

Example 2

User enters a positive number, show the factorial:

//get input into n

//multiply n by factorialn

// subtract one from n

// if n is 1 end loop

//show factorialn

Examples 3-1

To show a table of temperature conversions from Celsius to Fahrenheit, where C = 5/9(F-32), you will need a starting value of F, and and end value of F, and the increment. Let's use start=0, end=300 and the step is 20. The output should look something like:

0	-17
20	-6
40	4
…….	

Example 3-2

The code would be something like:

```
lower=0;
upper=300;
step=20;
fahr=lower;
while (fahr <= upper) {
    cel=(5/9)*(fahr-32) //this line causes a problem. What is it?
     printf(%3.0f  %6.1f", fahr, cel);
     fahr = fahr + step
   …. Close all closing braces
```

`while` versus `do...while`

- `while` loop may never execute
- `do...while` loop will always execute at least once

When To Use Loops

- Whenever you have a task to do repeatedly
  - "As long as some condition is true, do some action..."
  - "Do some action until some condition is no longer true..."
- Sometime, looping is harder to recognize
  - For a given value in cents (0 to 99), calculate how many quarters, dimes, nickels and pennies are required to represent that value

How To Use Loops

- Identify the terminating condition
  - how will the loop stop?
- Identify the initial condition
  - what is true before the loop ever executes?
- How is progress made toward the terminating condition
  - something must guarantee progress toward the terminating condition
  - without progress, you will have an infinite loop

Repetitive Control Flow in C

- Programs often must repeat different instructions in a variety of situations
  - sometimes, code must be repeated a determinate number of times
  - this calls for a for loop

The `for` Statement

- Determinate loop
  - Do something exactly ***n*** times, where ***n*** is known in advance

```
for ( int i = 1; i <= n; i++ ) {
   …block of statements...
}
```

The `for` Statement

- Determinate Loop

```
for (int i = 1;
    i < n;
    i++) {
   printf(“i=%d\n”, i);
}
```

The `for` Statement

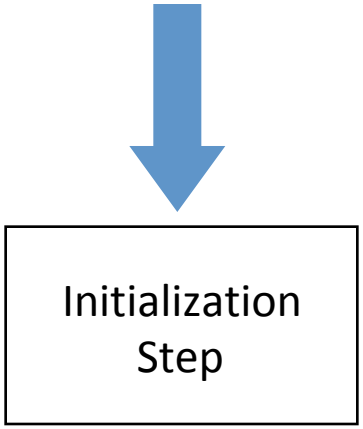
- Determinate Loop

```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```

The `for` Statement

- Determinate Loop

```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```

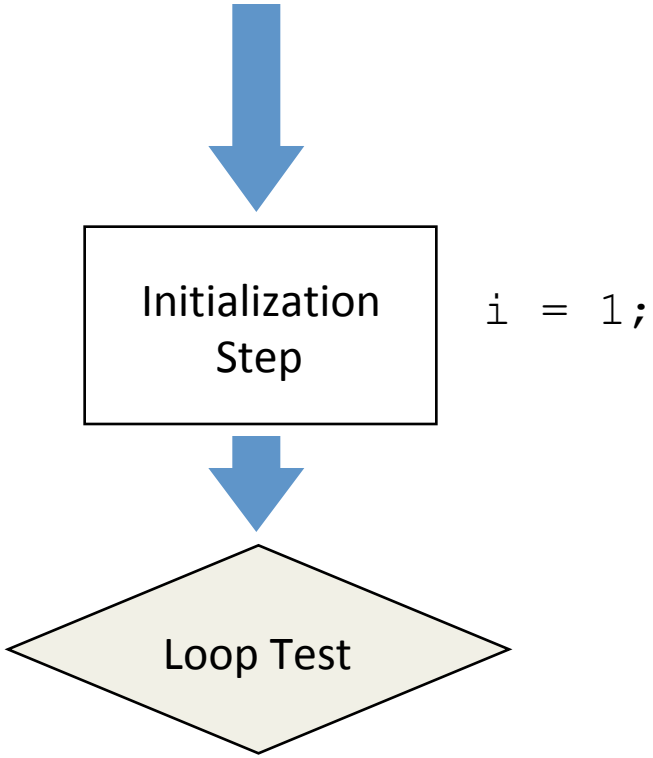



```
i = 1;
```

The `for` Statement

- Determinate Loop

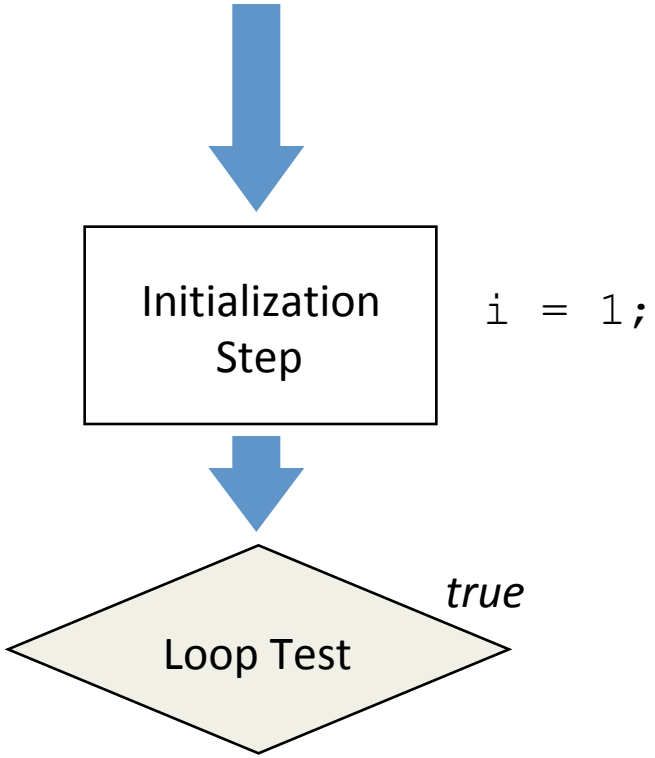
```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```



The `for` Statement

- Determinate Loop

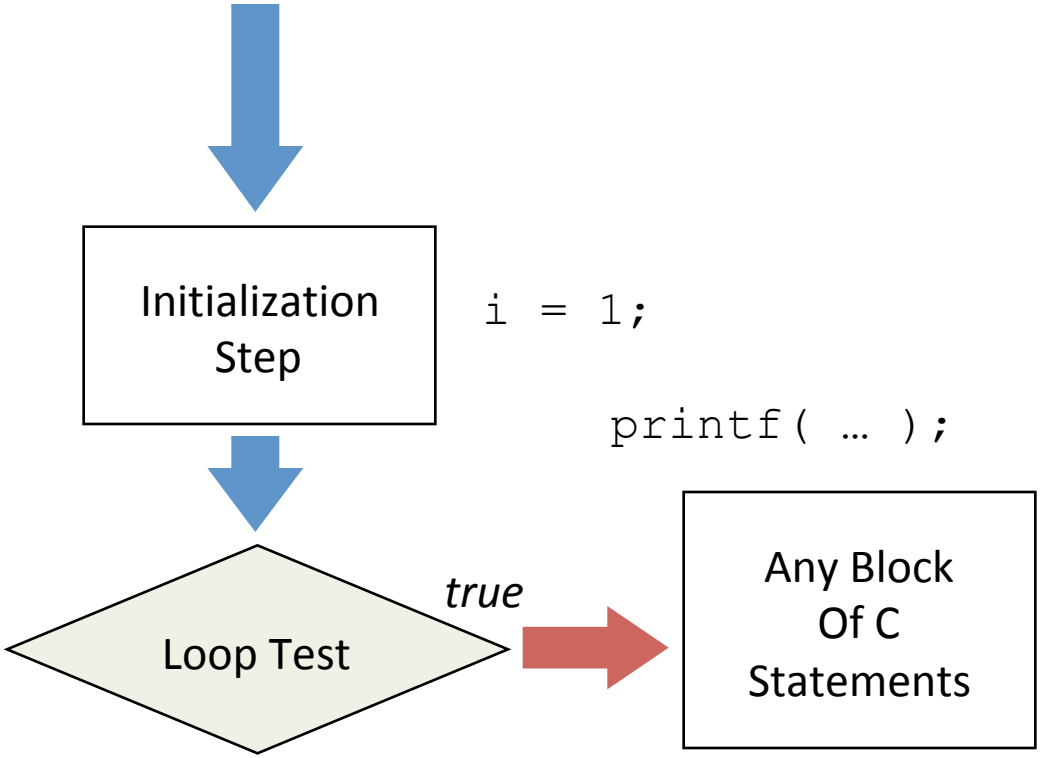
```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```



The `for` Statement

- Determinate Loop

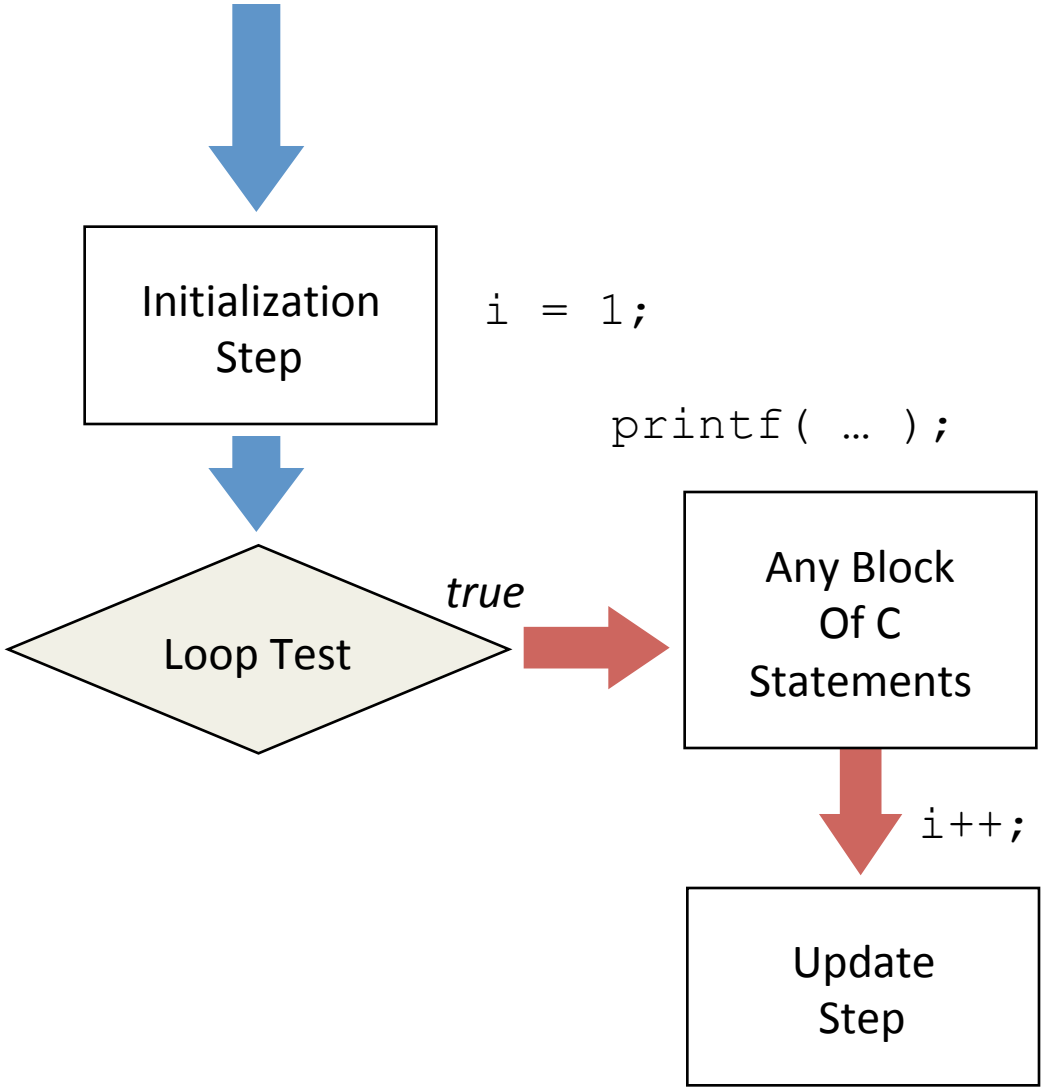
```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```



The `for` Statement

- Determinate Loop

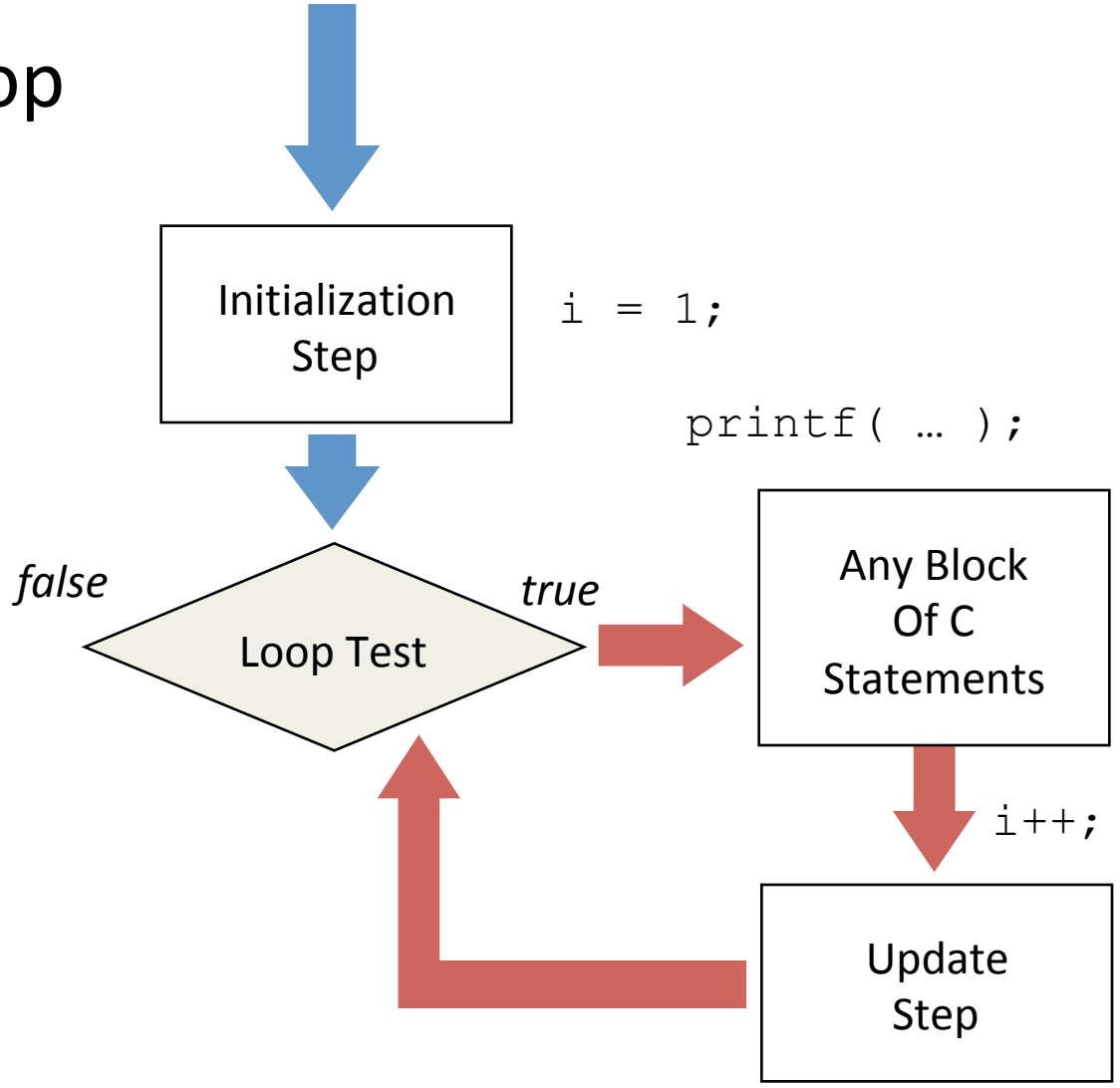
```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```



The `for` Statement

- Determinate Loop

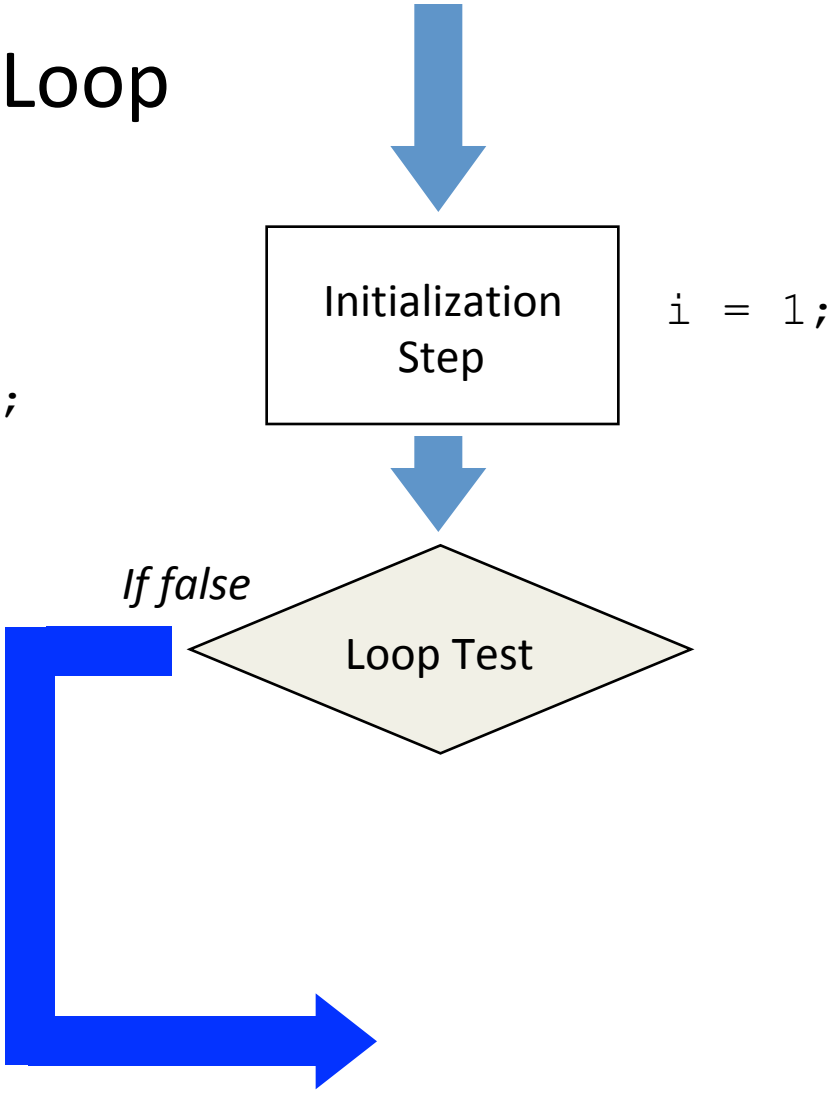
```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```



The `for` Statement

- Determinate Loop

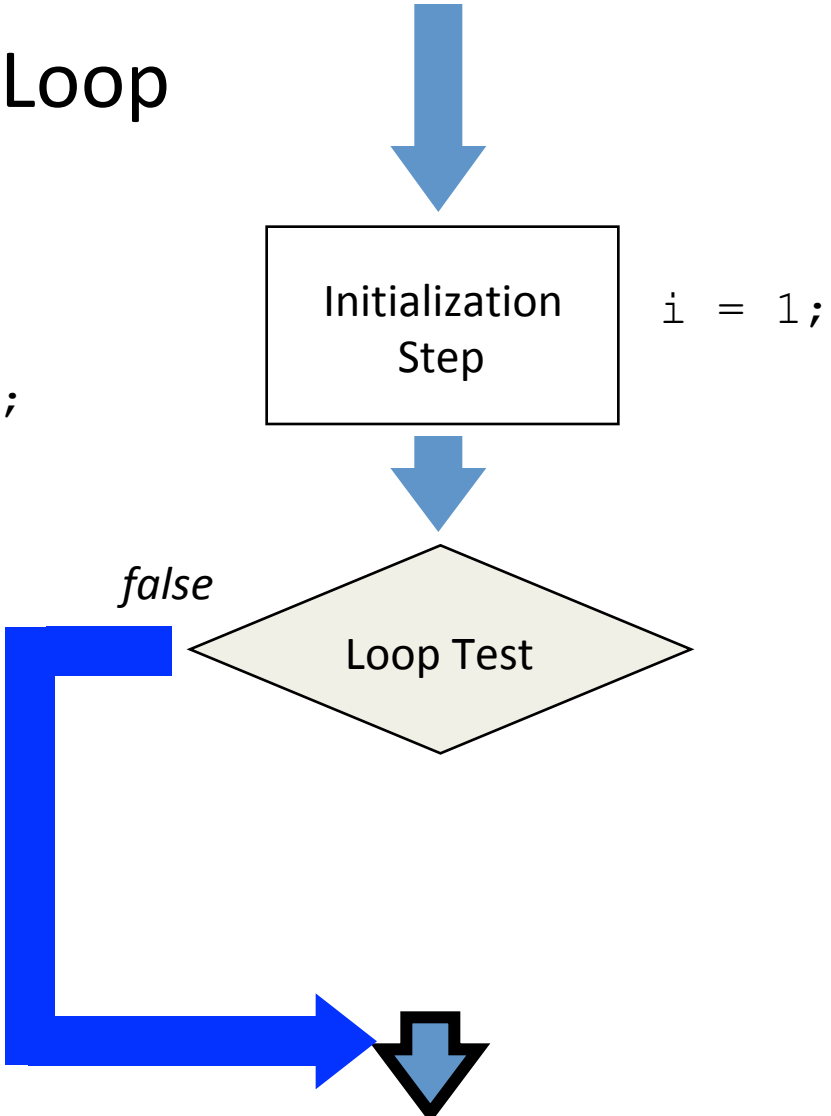
```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}
```



The `for` Statement

- Determinate Loop

```
for (int i = 1;
    i < n;
    i++) {
   printf("i=%d\n", i);
}//loop ends
```



Nested for-loop Demo!

```
/*
 Using loops...
*/
#include <stdio.h>

int main() {
        int start1 = 0, stop1 = 0;
        int start2 = 0, stop2 = 0;
        int i = 0, j = 0;
        /*
          It is always a good idea to output a prompt
          for the data you are trying to read.
    */
        printf( "I'm going to run nested loops\n" );
        printf( "based on values you enter now\n" );
        printf( ”enter your first start value:" );
        scanf( "%d", &start1 );
        printf( ”enter your first stop value:" );
        scanf( "%d", &stop1 );
        printf( ”enter your second start value:" );
        scanf( "%d", &start2 );
        printf( ”enter your second stop value:" );
        scanf( "%d", &stop2 );
        /*
          Depending on the values you enter, the
    loop may run, or not...
        */
        for (i = start1; i <= stop1; i++) {
                for (j = start2; j <= stop2; j++) {
                        printf( "i = %d and j = %d\n", i, j );
                }
        }
        return( 0 );
}
```

Summarizing Our for-loop Demo!

- Pick the control flow that most naturally fits your intentions
- It pays to invest the time to figure out the logic (your intentions)
- A `for` loop may never execute at all

Book Reading

- Check out section 15.2
- Go over section 15.3 goto. Avoid is a small word to describe how much you should NOT use goto.
  Same applies to break ; **note break is needed in switch statements** −but can be avoided in repetition statements.

Summary

- Various kinds of repetition flow of control
  - Do while
  - While
  - For