CS102/IT102 Computer Programming I

Lecture 10: Repetition (Part 1)

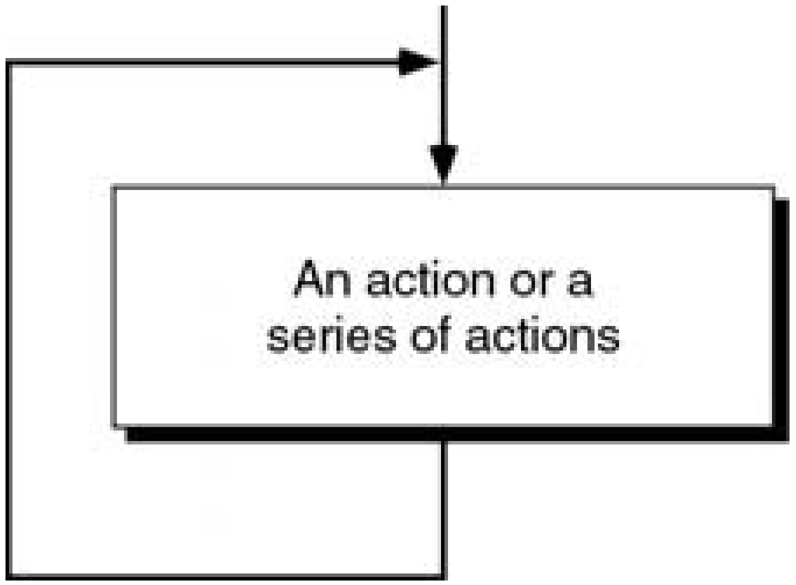
Bicol University College of Science
CSIT Department
1st Semester, 2023-2024

Topics

- Concept of a loop
- Event-controlled and counter-controlled loops
- Pre-test loops in C
  - **`while`** statement
  - **`for`** statement
- **`break`** statement
- Nested loops

Concept of a Loop

- Loop is a group of instructions computer executes repeatedly while some condition remains true
- To make sure that a loop ends, we must have a condition that controls it.

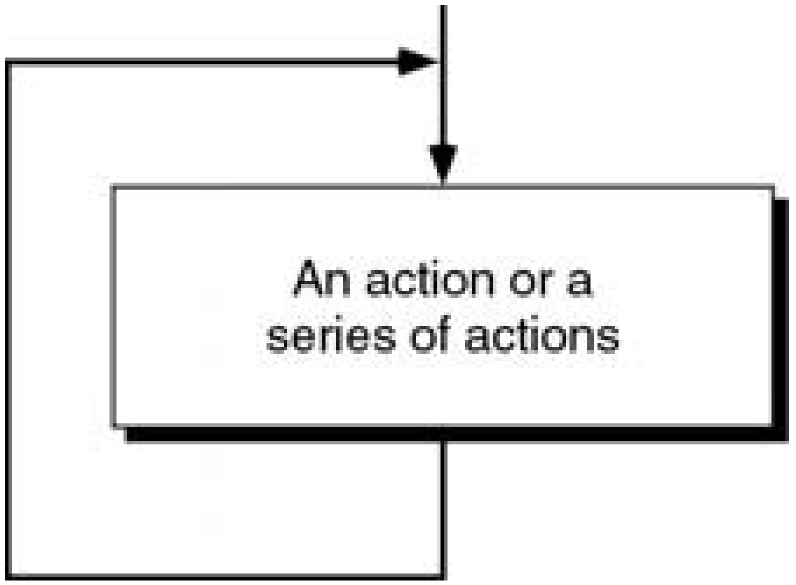


Concept of a Loop

The loop must be designed so that before or after each **iteration**, it checks to see if it is done

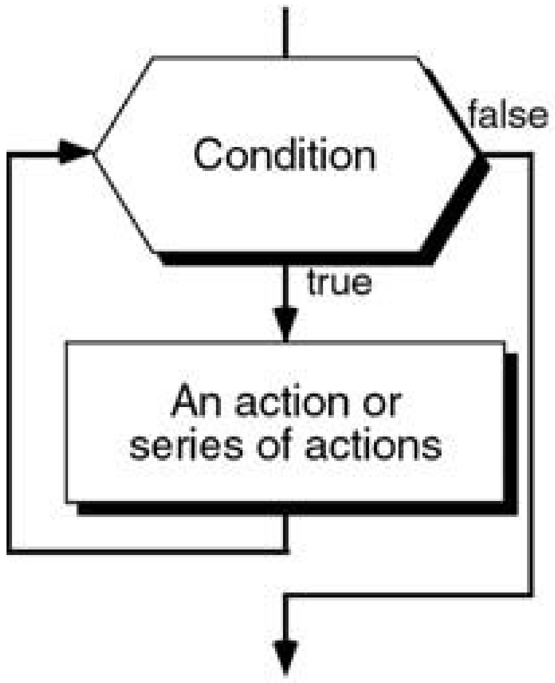
- If it is not done, it repeats one more time;
- If it is done, it exits the loop.

This test is known as a **loop control expression**.

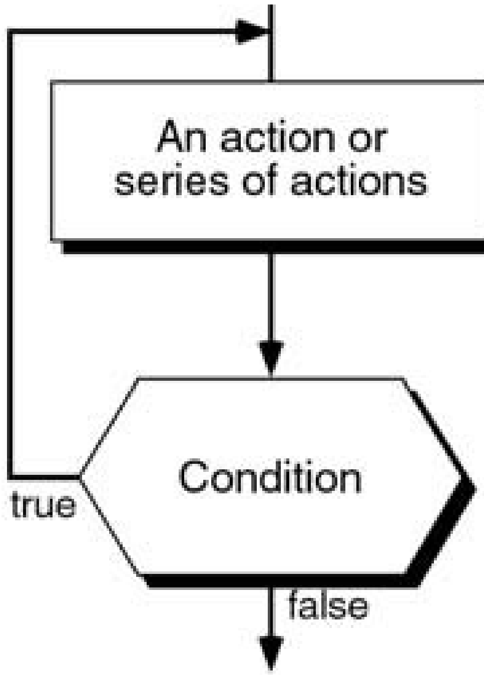


Pretest and Post-test Loops

Programming languages allow us to check the loop control expression either before or after each iteration of the loop.



(a) Pretest Loop



(b) Post-test Loop

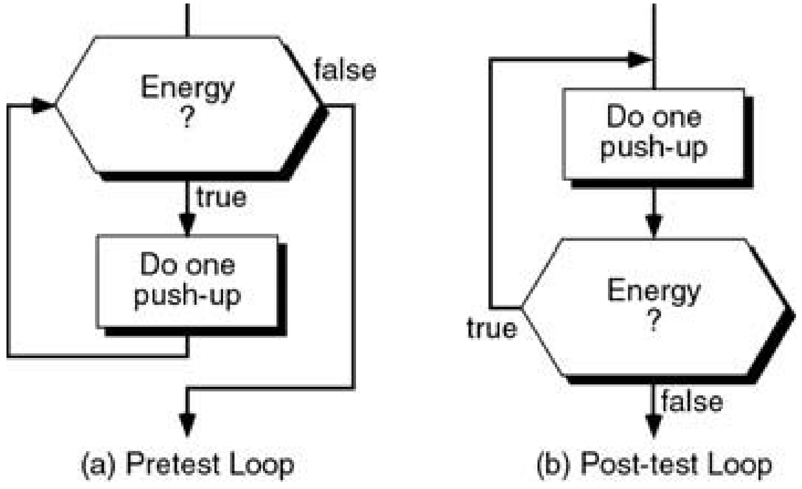
Pretest and Post-test Loops

Pretest loop- the condition is checked before we start and at the beginning of each iteration after the first.

– If the test condition is true, the code executes; if the test condition is false, the loop terminates.

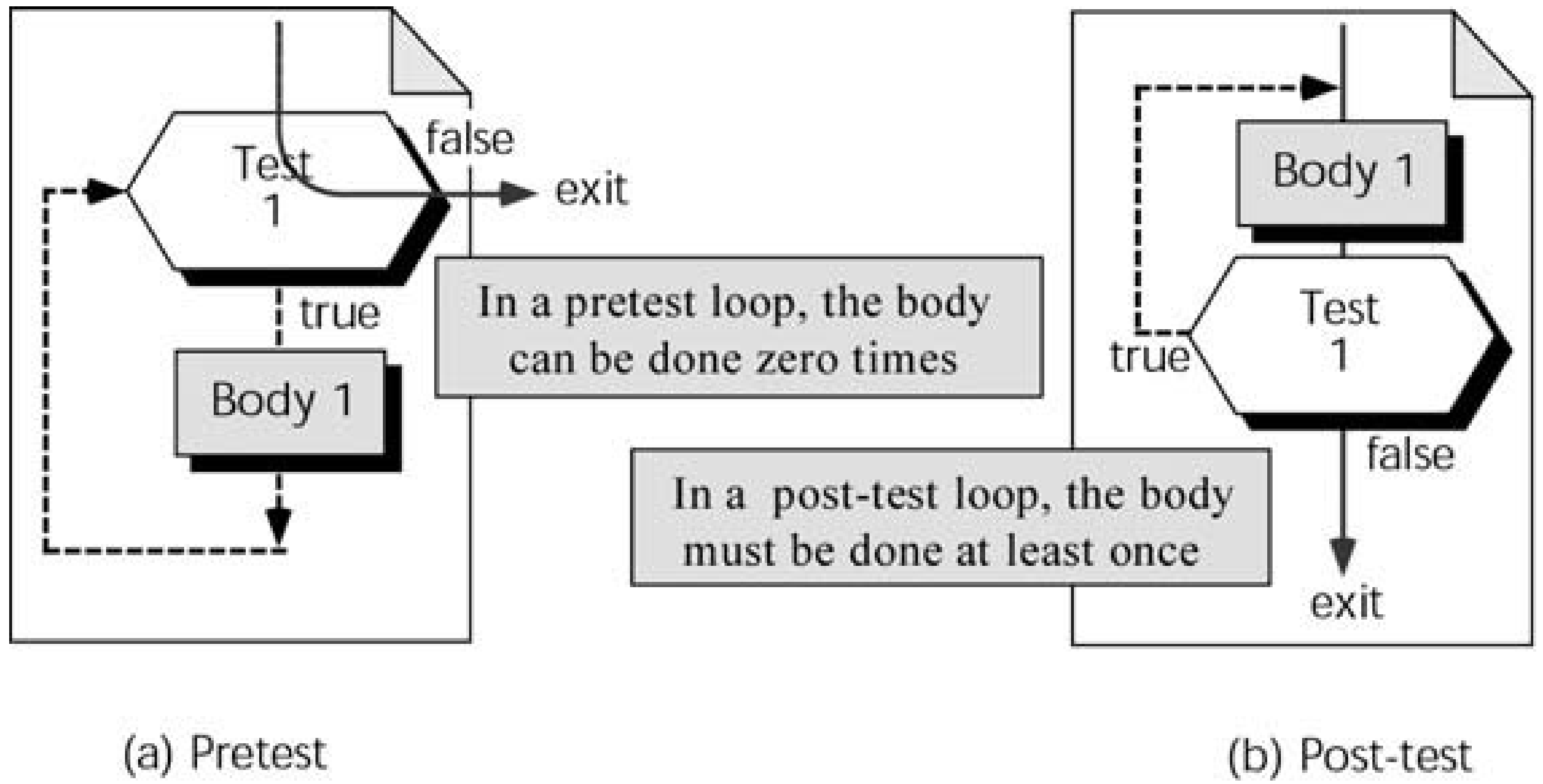
Post-test loop- the code is always executed at least once. At the completion of the loop code, the loop control expression is tested.

– If the expression is true, the loop repeats; if the expression is false, the loop terminates.



Two different strategies for doing exercises

Pretest and Post-test Loops



false
exit
Test 1
true
Body 1
In a pretest loop, the body can be done zero times
Body 1
Test 1
true
false
exit
In a post-test loop, the body must be done at least once
(a) Pretest
(b) Post-test

Minimum number of iterations in pretest and post-test loop

Initializing and Updating

- **Loop initialization**

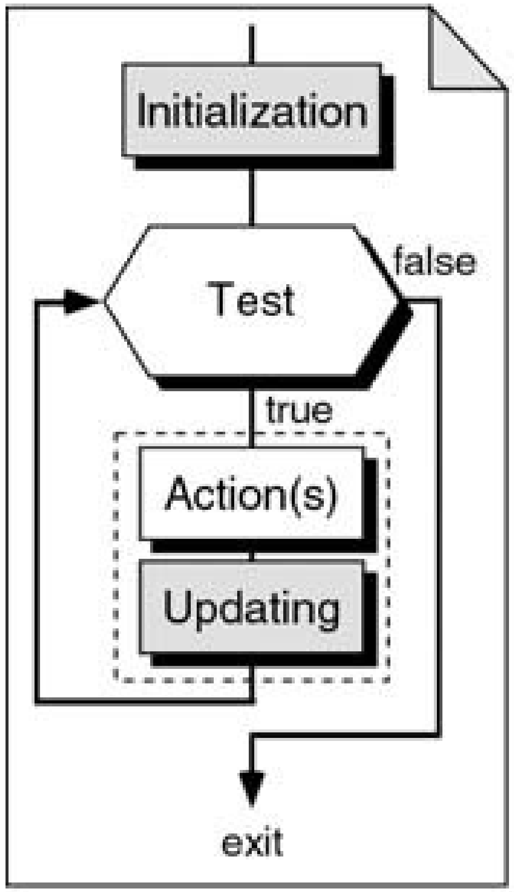
Before a loop can start, some preparation is usually required. Initialization must be done before the first execution of the body. It sets the stage for the loop actions. Initialization may be explicit and implicit.

  - Explicit initialization is much more common. You include code to set the beginning values of key loop variables.
  - Implicit initialization provides no direct code to set the starting values but, rather, relies on a pre-existing situation, such as values passed to the function that controls the loop.

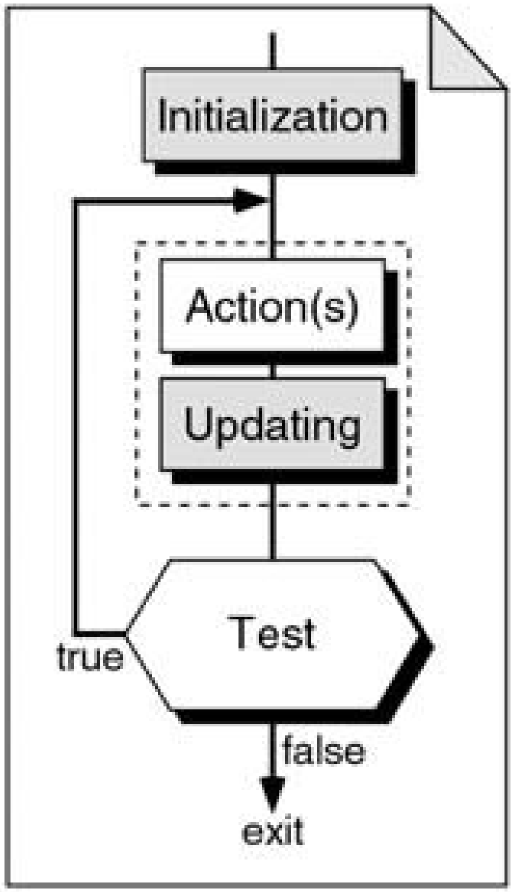
- **Loop update**

Something must happen inside the body of the loop to change the condition that controls the loop from true to false. Otherwise, we would have an infinite loop. The actions that cause these changes are known as loop update.

Initializing and Updating

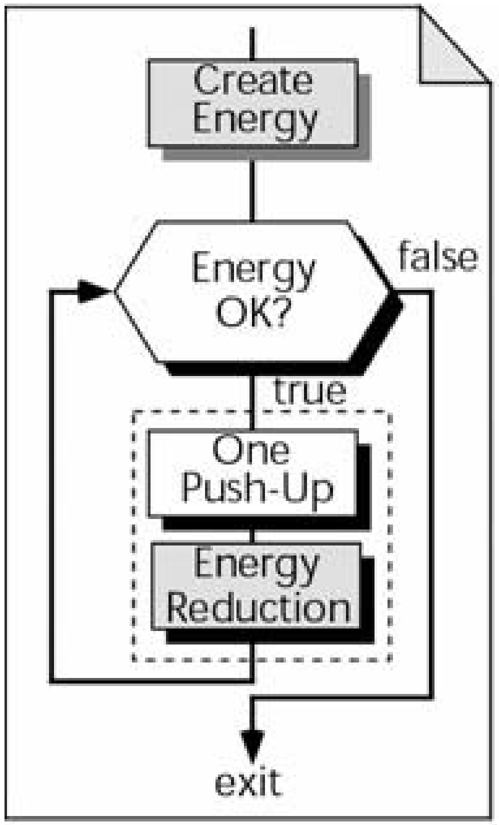


(a) Pretest Loop

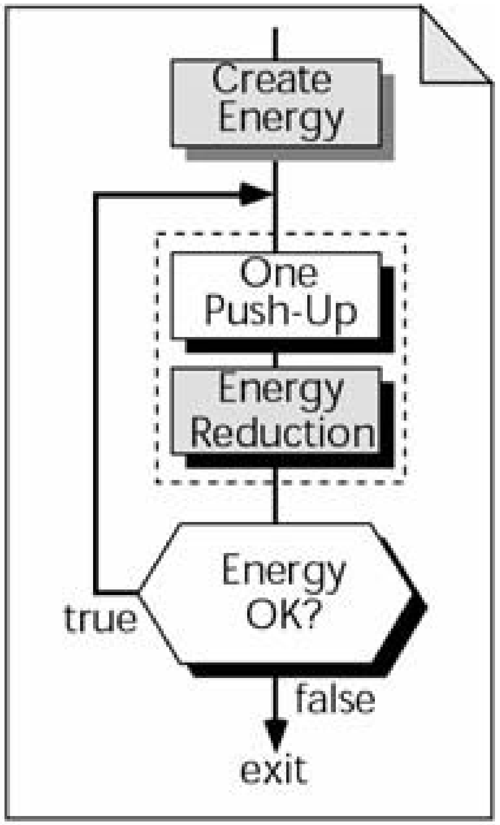


(b) Post-test Loop

Initializing and Updating



(a) Pretest Loop



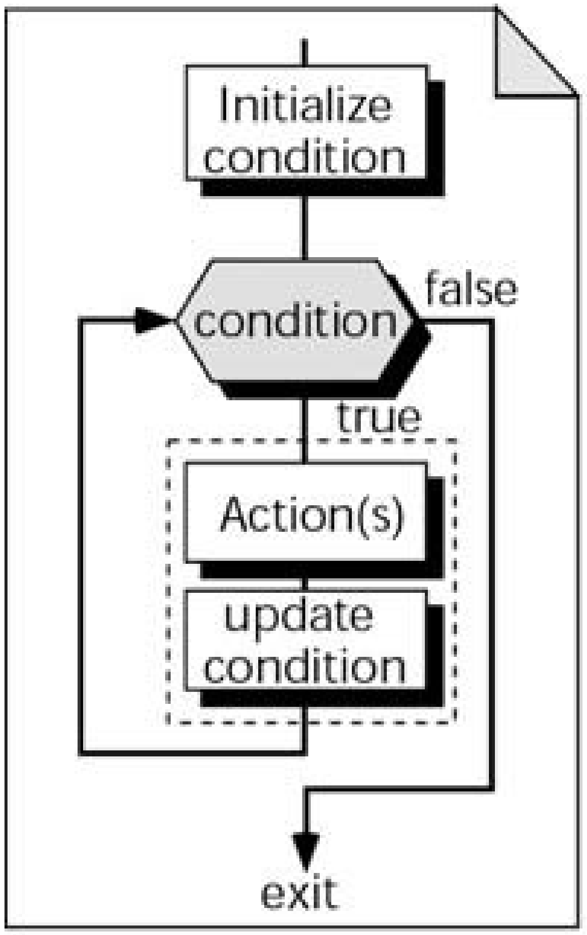
(b) Post-test Loop

Event-Controlled and Counter-Controlled Loops

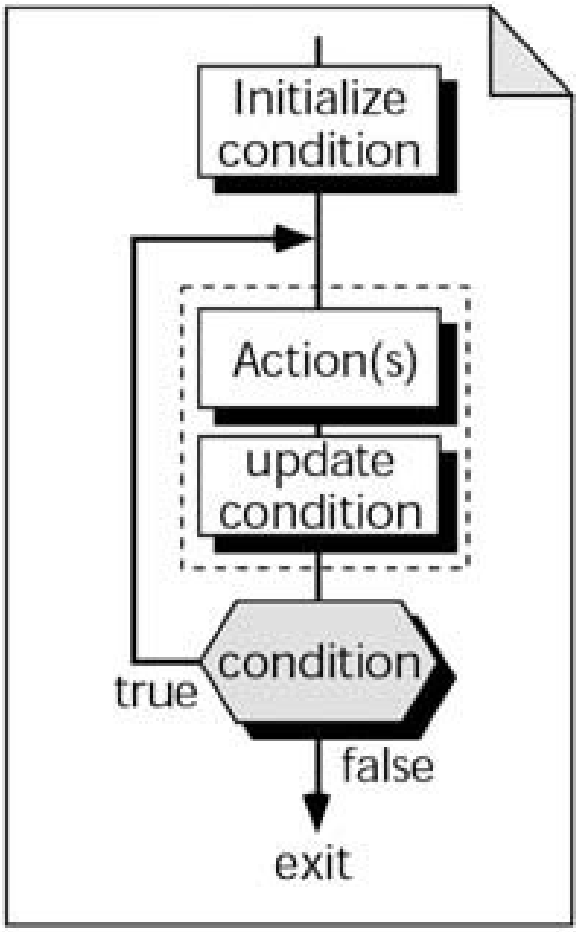
All the possible expression that can be used in a loop limit test can be summarized into two general categories:

- **Event-controlled loop** - an event changes the loop control expression from true to false.
  - Indefinite repetition
  - Used when number of repetitions not known
  - Sentinel value indicates "end of data"
  - Explicit (controlled by the loop) or implicit (controlled by some external condition) updating process
- **Counter-controlled loop** is used when we know the number of times an action is to be repeated.
  - Definite repetition: know how many times loop will execute
  - Control variable used to count repetitions

Event-Controlled Loop Concept



(a) Pretest Loop

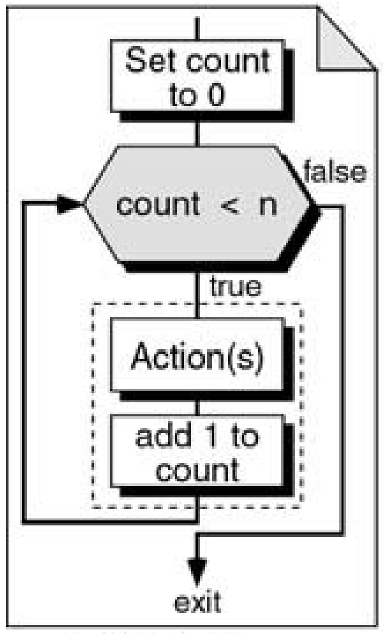


(b) Post-test Loop

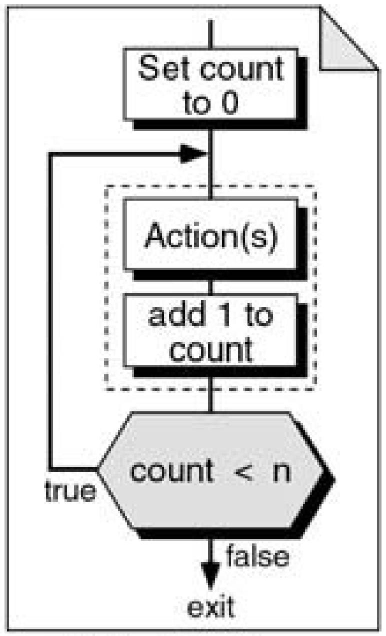
Counter-Controlled Loop Concept

Counter-controlled repetition requires

- the name of a control variable (or loop counter)
- the initial value of the control variable
- an increment (or decrement) by which the control variable is modified each time through the loop
- a condition that tests for the final value of the control variable (i.e., whether looping should continue)



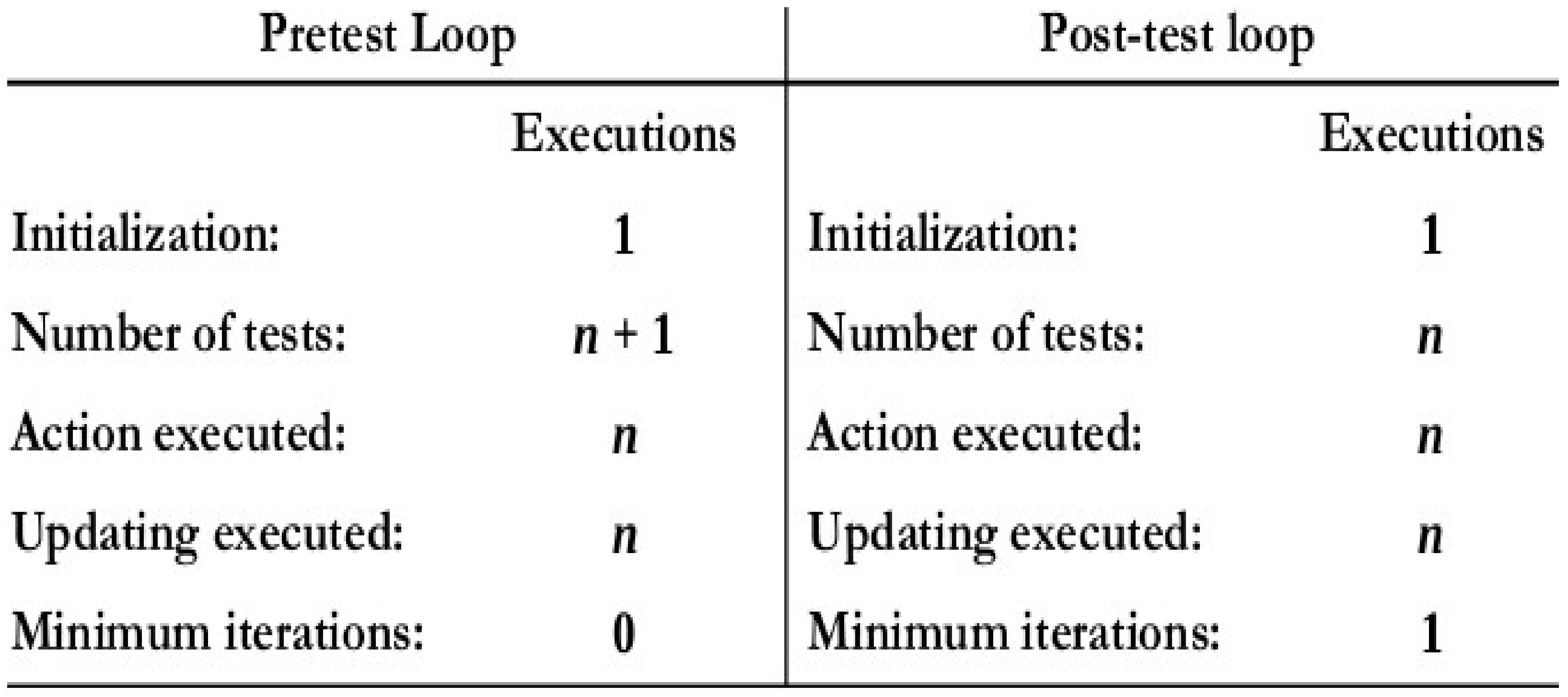
(a) Pretest Loop



(b) Post-test Loop

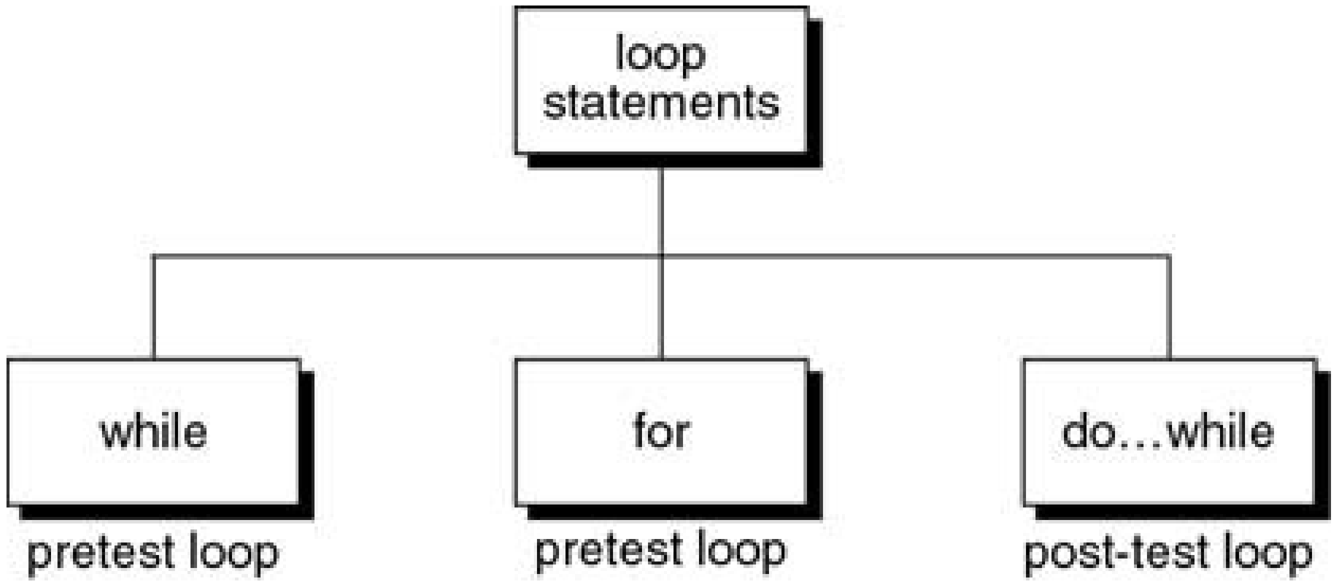
Loop Comparison

- In the pretest loop, when we come out of the test, the limit test has been done **n+1** times.
- In the post-test loop, when we come out of the loop, the limit test has been done only **n** times.



Pretest Loop		Post-test loop	
	Executions		Executions
Initialization:	1	Initialization:	1
Number of tests:	$n + 1$	Number of tests:	n
Action executed:	n	Action executed:	n
Updating executed:	n	Updating executed:	n
Minimum iterations:	0	Minimum iterations:	1

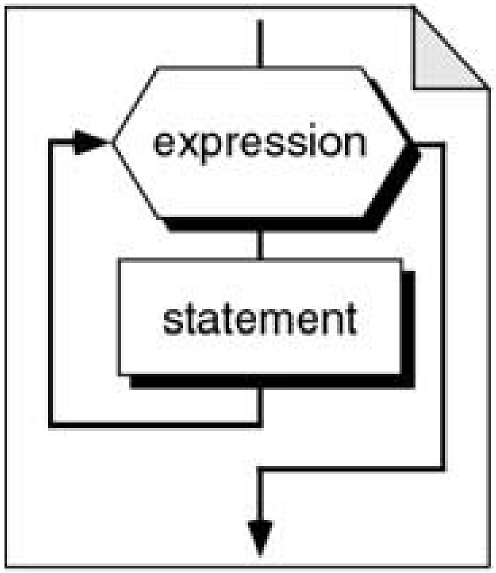
Loops in C



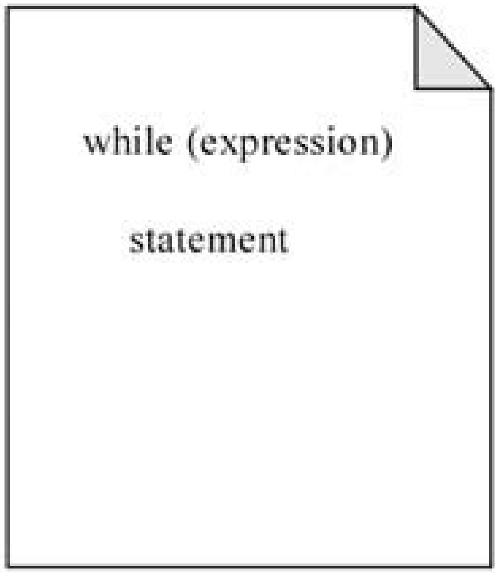
- The **`while`** and **`do…while`** are most commonly used for event-controlled loops.
- The **`for`** is usually used for counter-controlled loop.
- All loop constructs continue when the limit control test is true and terminate when it is false. This consistency of design makes it easy to write the limit test in C.

The `while` Loop

- The **while** statement is a pretest loop.
- It uses an expression to control the loop. Since it is a pretest loop, it tests the expression before every iteration of the loop.



(a) Flowchart



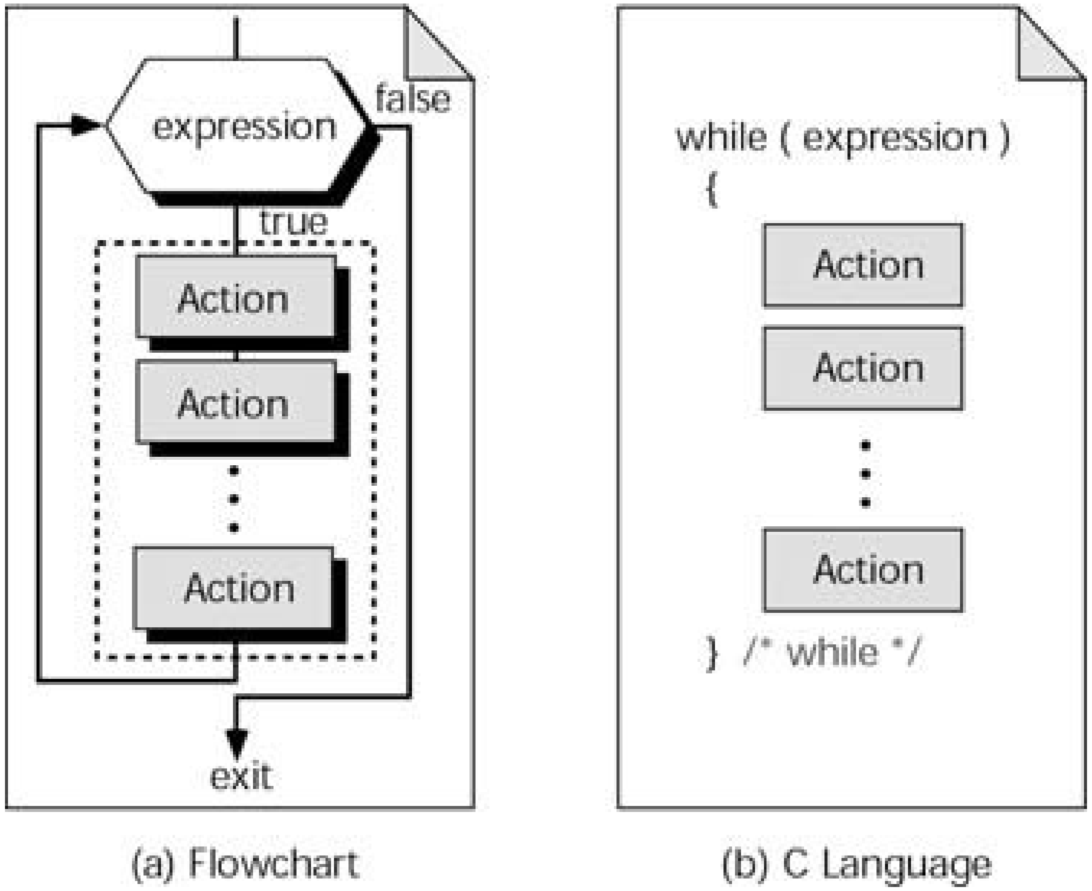
```
while (expression)

    statement
```

(b) Sample Code

Compound `while` Statement

- If we want to include multiple statements in the body, we must put them in a compound statement (block).



Example: **addnum.c**

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

Iteration Control

Initialize

Check condition

Update

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
   input nextNum
   add nextNum to sum
   add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{









  return 0;
}
```

Example: **addnum.c** (cont)

```
Read in numbers, add them, and
print their sum and average


set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;




  return 0;
}
```

only the variables sum and count are initialized to 0

Example: **addnum.c** (cont)

```
Read in numbers, add them, and
print their sum and average


set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);










  return 0;
}
```

Example: **addnum.c** (cont)

```
Read in numbers, add them, and
print their sum and average



set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/***********************************\
 Read in numbers and add them up
 Print out the sum and the average
\***********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {



  }




  return 0;
}
```

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }




  return 0;
}
```

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    co  ++;
  }
}
```

Same as: `sum = sum + nextNum;`

Others: `-=`, `*=`, `/=`, etc. (King, Table 4.2)

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum
```

```c
#include <stdio.h>
/***********************************\
 Read in numbers and add them up
 Print out the sum and the average
\***********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }
}
```

Same as: `count = count + 1;`

Decrement: `count--;` (King, Table 4.2)

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

Example: **addnum.c** (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

Example: **addnum.c** (cont)

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

totalNumbers	count	nextNum	sum
????	0	????	0.0

Example: **addnum.c** (cont)

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

totalNumbers	count	nextNum	sum
????	0	????	0.0
3			

Example: **addnum.c** (cont)

```
#include <stdio.h>
/***********************************\
 Read in numbers and add them up
 Print out the sum and the average
\***********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

totalNumbers	count	nextNum	sum
????	0	????	0.0
3			
	1	4	4.0

Example: **addnum.c** (cont)

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

totalNumbers	count	nextNum	sum
????	0	????	0.0
3			
	1	4	4.0
	2	-1	3.0

Example: **addnum.c** (cont)

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count = 0, totalNumbers;
  scanf("%d", &totalNumbers);

  while (count < totalNumbers)
  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);
  return 0;
}
```

totalNumbers	count	nextNum	sum
????	0	????	0.0
3			
	1	4	4.0
	2	-1	3.0
	3	6.2	9.2

Common Mistakes in **`while`** – *“one liners”*

```
while (num < minimum)
  scanf(“%d”, &num);
  printf(“Number must be greater than %d.\n”, minimum);
  printf(“Please try again.\n”);
```

```
while (num < minimum)
{
  scanf(“%d”, &num);
}

printf(“Number must be greater than %d.\n”, minimum);
printf(“Please try again.\n”);
```

Common Mistakes in **`while`** -- *“one liners”* (cont)

```
while (num < minimum)
  scanf(“%d”, &num);
  printf(“Number must be greater than %d.\n”, minimum);
  printf(“Please try again.\n”);
```

```
while (num < minimum)
{
  scanf(“%d”, &num);
  printf(“Number must be greater than %d.\n”, minimum);
  printf(“Please try again.\n”);
}
```

✓

Common Mistakes in **`while`** -- extra semi-colon;

```
while (num < minimum);
{
  scanf("%d", &num);
  printf("Number must be g[illegible]ater than %d.\n", minimum);
  printf("Please try again[illegible]);
}
```

Marks the end of the while-block -- usual cause of infinite loops

Checking for End-of-Input / End-of-File in `while`

Read in numbers, add them, and print their sum and average

```
set sum to 0

input nextNum
check if end of input
while  (not end of input)
{
    add nextNum to sum
    input nextNum
    check if end of input
 }
```

etc...etc...etc...

Checking for End-of-Input / End-of-File in `while` (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0

input nextNum
check if end of input
while  (not end of input)
{
   add nextNum to sum
   input nextNum
   check if end of input
 }
 etc...etc...etc...
```

etc...etc...etc...

```
float nextNum;
float sum = 0.0;

scanf("%f", &nextNum);

while ( ?????? )
{
   sum += nextNum;
   scanf("%f", &nextNum);

}
```

etc...etc...etc...

Checking for End-of-Input / End-of-File in `while` (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0

input nextNum
check if end of input
while  (not end of input)
{
   add nextNum to sum
   input nextNum
   check if end of input
}
```

etc...etc...etc...

etc...etc...etc...

```
float nextNum;
float sum = 0.0;

scanf("%f", &nextNum);
???????
while ( ?????? )
{
   sum += nextNum;
   scanf("%f", &nextNum);
???????
}
```

etc...etc...etc...

Checking for End-of-Input / End-of-File in `while` (cont)

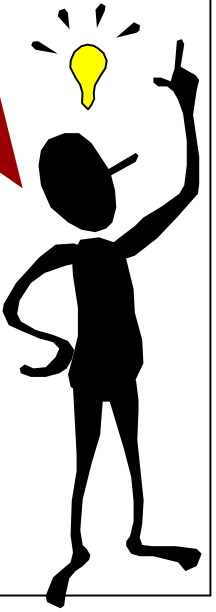
Read in numbers, add them, an
print their sum and average

```
set sum to 0

input nextNum
check if end of input
while  (not end of input)
{
   add nextNum to sum
   input nextNum
   check if end of input
}
etc...etc...etc...
```

Recall: **When the input ends, the scanf() function returns a special char value: EOF**

```
scanf("%f", &nextNu
???????
while ( ?????? )
{
   sum += nextNum;
   scanf("%f", &nextNum);
   ???????
}
etc...etc...etc...
```



Checking for End-of-Input / End-of-File in `while` (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0

input nextNum
check if end of input
while  (not end of input)
{
   add nextNum to sum
   input nextNum
   check if end of input
}
```

etc...etc...etc...

etc...etc...etc...

```
float nextNum;
float sum = 0.0;


while ( scanf("%f",&nextNum) != EOF )
{


  sum += nextNum;


}
```

etc...etc...etc...

Checking for End-of-Input / End-of-File in `while` (cont)

Read in numbers, add them, and print their sum and average

```
set sum to 0

input nextNum
check if end of input
while  (not end of input)
{
   add nextNum to sum
   input nextNum
   check if end of input
 }
```

etc...etc...etc...

etc...etc...etc...

```
float nextNum;
float sum = 0.0;


while ( scanf("%f",&nextNum) != EOF )
{

   sum += nextNum;

}
```

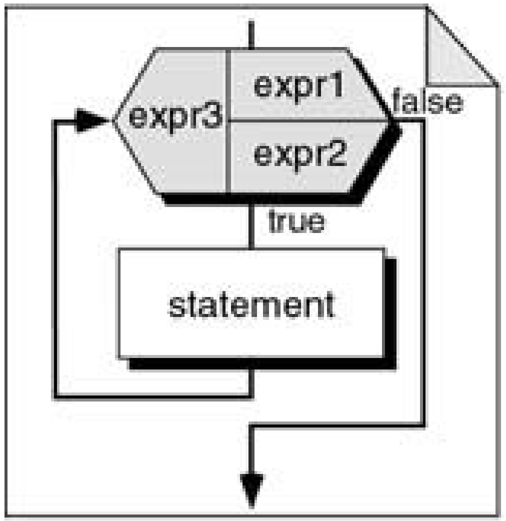
etc...etc...etc...

The `for` Loop

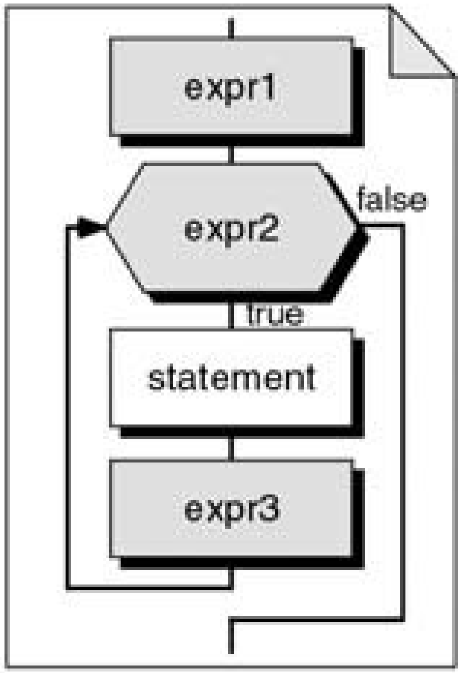
The **`for`** statement is a pretest loop that uses three expressions:

- **expr1**: contains any initialization statements. It is executed when the for starts.
- **expr2**: contains the limit-test expression or the condition. It is executed before every iteration.
- **expr3**: contains the updating expression. It is executed in the end of each loop.

- The code in the **`for`** statement must be expressions. You cannot use statements, such as **`return`**, in the **`for`** statement itself.



(a) Flowchart

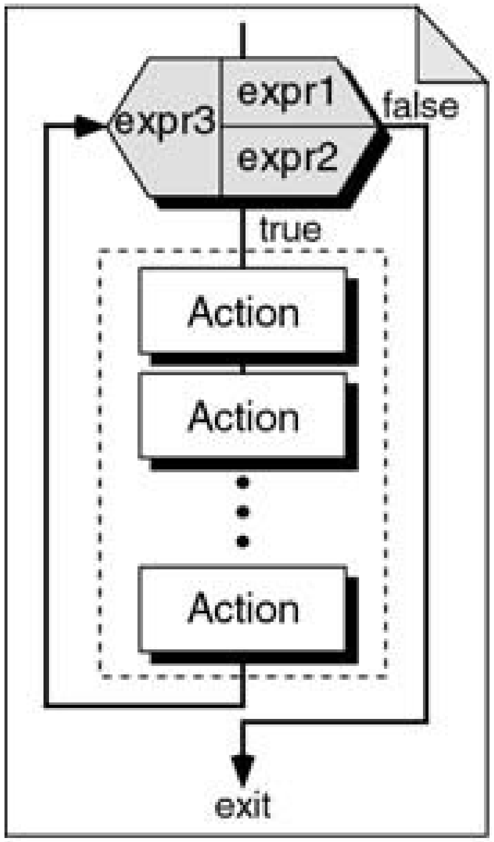


(b) Expanded Flowchart

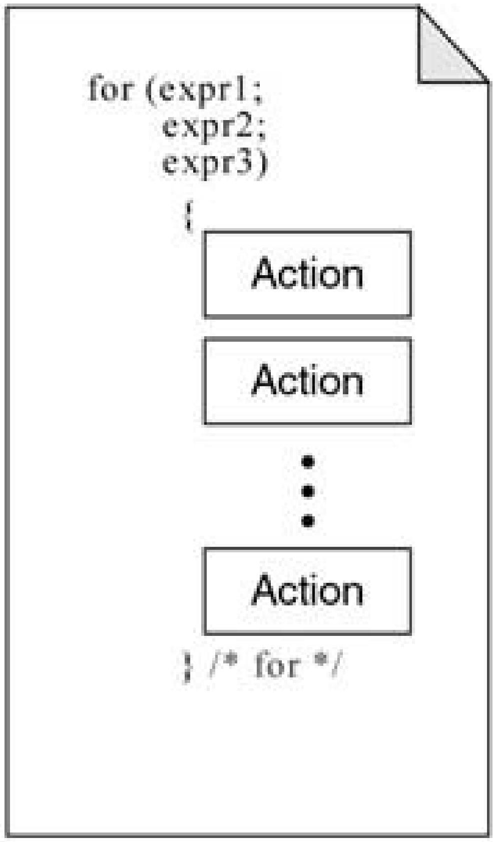
```
for (expr1; expr2; expr3)
      statement
```

Compound **`for`** Statement

- The **body** of the ***for*** loop must be one, and only one, statement.
- If we want to include more than one statement, we must code them in a compound statement.

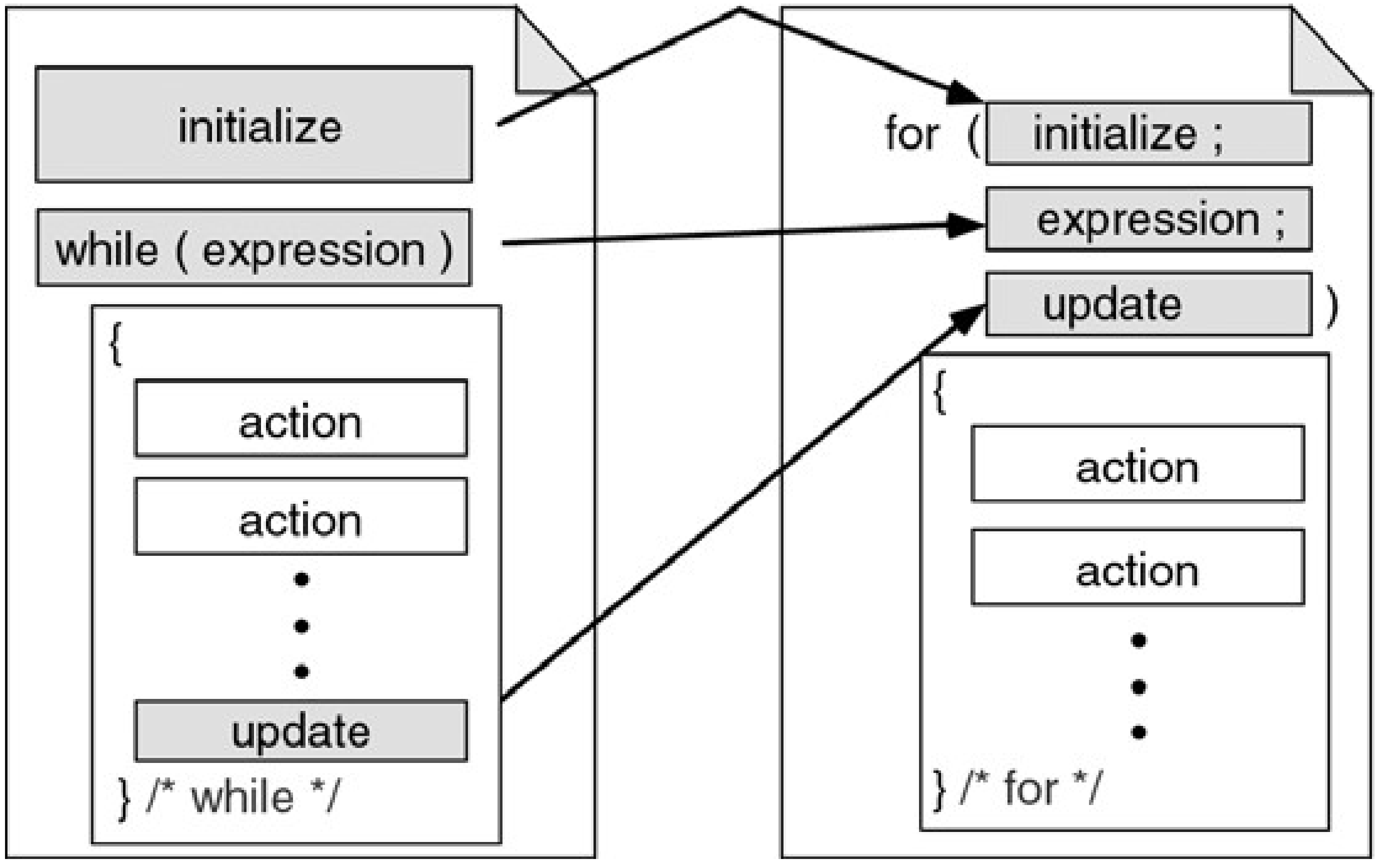


(a) Flowchart



(b) C Language

Comparing **`for`** and **`while`** Loops



A **`for`** loop is used when your loop is to be executed a known number of times. You can do the same thing with a **`while`** loop, but the **`for`** loop is easier to read and more natural for counting loops.

The **`for`** Statement

- Form of loop which allows for initialization and iteration control
- Syntax:

```
for ( initialization; condition; update )
{
    instruction block
}
```

Careful! A semi-colon here marks the end of the instruction block!

Example: addfor.c

Read in numbers, add them, and print the sum and the average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

Example: **addfor.c** (cont)

Read in numbers, add them, and print the sum and the average

```
set sum to 0
set count to 0
input totalNumbers

while  (count < totalNumbers)
{
   input nextNum
   add nextNum to sum
   add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```c
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count, totalNumbers;

  scanf("%d", &totalNumbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);

  return 0;
}
```

Example: **addfor.c** (cont)

Read in numbers, add them, and print the sum and the average

```
set sum to 0
set count to 0

input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```
#include <stdio.h>
/***********************************\
 Read in numbers and add them up
 Print out the sum and the average
\***********************************/
int main()
{
  float nextNum, sum
  int count, totalNu

  scanf("%d", &total   mbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);

  return 0;
}
```

Initialize

Example: **addfor.c** (cont)

Read in numbers, add them, and print the sum and the average

```
set sum to 0
set count to 0

input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int ma
{
  floa
  int

  scanf("%d", &totalNu     s);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);

  return 0;
}
```

Check condition

Example: **addfor.c** (cont)

```
Read in numbers, add them, and
print the sum and the average

set sum to 0
set count to 0

input totalNumbers

while  (count < totalNumbers)
{
   input
   add n
   add 1
}

output '
output '
```

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{

  scan      &totalNumbers);

  for ( cou    0;
        count < totalNumbers;
        count++ )

                                unt);

}
```

Update **(aka *Increment Step*)**

IMPORTANT!!
The Update is performed AFTER the body of the loop

Example: **addfor.c** (cont)

Read in numbers, add them, and print the sum and the average

```
set sum to 0
set count to 0

input totalNumbers

while  (count < totalNumbers)
{
    input nextNum
    add nextNum to sum
    add 1 to count
}

output "Sum was" sum
output "Mean was" sum/count
```

```
#include <stdio.h>
/**********************************\
 Read in numbers and add them up
 Print out the sum and the average
\**********************************/
int main()
{
  float nextNum, sum = 0.0;
  int count, totalNumbers;

  scanf("%d", &totalNumbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",sum/count);

  return 0;
}
```

while and for

```
#include <stdio.h>
int main()
{
  float nextNum, sum = 0.0;
  int count, totalNumbers;

  scanf("%d", &totalNumbers);

  count = 0;
  while (count < totalNumbers)

  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }
  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

```
#include <stdio.h>
int main()
{
  float nextNum, sum = 0.0;
  int count, totalNumbers;

  scanf("%d", &totalNumbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

while and for (cont)

Initialize

```
#include <std
int main()
{
  float nextN
  int count, tota         rs;

  scanf("%d"   totalNumbers);

  count = 0;
  while (count < totalNumbers)

  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

```
      ude <stdio.h>
      ain()

      at nextNum, sum = 0.0;
  nt count, totalNumbers;

  scan     %d", &totalNumbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

while and for (cont)

Check condition

```
#include <stdio.h>
int main()
{
  float nextNum, sum = 0.0
  int count, totalNumbers;

  scanf("%d", &totalNumbers

  count = 0;
  while (count < totalNumbers)

  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

```
#include <stdio.h>



                 um = 0.0;
  int co       talNumbers;

  scanf("%d", &to    lNumbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

while and for (cont)

```
#include <stdio.h>
int main()
{
  float nextNum, sum = 0.0;
  int count, totalNumbers;

  scanf("%d", &totalNumbers);

  count = 0;
  while (count < totalNumbers)

  {
    scanf("%f", &nextNum);
    sum += nextNum;
    count++;
  }

  printf("Sum    s %f\n",sum)
  printf("Mean w    %f\n",
          sum

  return 0;
}
```

Update

```
#include <stdio.h>
int main()
{
  float nextNum, sum = 0.0;
  int count, totalNumbers;

  scanf("%d", &totalNumbers);

  for ( count=0;
        count < totalNumbers;
        count++ )
  {
    scanf("%f", &nextNum);
    sum += nextNum;
  }

  printf("Sum was %f\n",sum);
  printf("Mean was %f\n",
          sum/count);

  return 0;
}
```

The **`break`** Statement

- Implements the "exit loop" primitive
- Causes flow of control to leave a loop block (**`while`** or **`for`**) immediately

Example: **recip.c**

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
   input nextNum
   if (nextNum is 0)
   {
      exit loop
   }
   else
   {
      output 1/nextNum
   }
}
```

Example: **recip.c** (cont)

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
   input nextNum
   if (nextNum is 0)
   {
      exit loop
   }
   else
   {
      output 1/nextNum
   }
}
```

```
#include <stdio.h>
/********************************\
  Print out the reciprocals of
  numbers entered. Quit when 0
  is entered
\********************************/

int main()
{
  float nextNum;










  return 0;
}
```

Example: **recip.c** (cont)

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
   input nextNum
   if (nextNum is 0)
   {
      exit loop
   }
   else
   {
      output 1/nextNum
   }
}
```

```
#include <stdio.h>
/********************************\
  Print out the reciprocals of
  numbers entered. Quit when 0
  is entered
\********************************/

int main()
{
  float nextNum;

  while (1)
  {





  }
  return 0;
}
```

"while (True)" infinite loop

Example: **recip.c** (cont)

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
  input nextNum
  if (nextNum is 0)
  {
    exit loop
  }
  else
  {
    output 1/nextNum
  }
}
```

```
#include <stdio.h>
/******************************\
  Print out the reciprocals of
  numbers entered. Quit when 0
  is entered
\******************************/

int main()
{
  float nextNum;

  while (1)
  {
    scanf("%f", &nextNum);









  }
  return 0;
}
```

Example: **recip.c** (cont)

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
  input nextNum
  if (nextNum is 0)
  {
    exit loop
  }
  else
  {
    output 1/nextNum
  }
}
```

```
#include <stdio.h>
/*******************************\
  Print out the reciprocals of
  numbers entered. Quit when 0
  is entered
\*******************************/

int main()
{
  float nextNum;

  while (1)
  {
    scanf("%f", &nextNum);
    if (nextNum == 0.0)
    {
      break;
    }
    else
    {
      printf("%f\n", 1/nextNum);
    }

  }
  return 0;
}
```

Example: **recip.c** (cont)

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
   input nextNum
   if (nextNum is 0)
   {
      exit loop
   }
   else
   {
      output 1/nextNum
   }
}
```

```c
#include <stdio.h>
/*******************************\
  Print out the reciprocals of
  numbers entered. Quit when 0
  is entered
\*******************************/

int main()
{
  float nextNum;

  while (1)
  {
    scanf("%f", &nextNum);
    if (nextNum==0.0)
    {
      break;
    }
    else
    {
      printf("%f\n", 1/nextNum);
    }
  }

  return 0;
}
```

Example: **recip.c** (cont)

Print out the reciprocals of numbers entered. Quit when 0 is entered

```
loop
{
  input nextNum
  if (nextNum is 0)
  {
    exit loop
  }
  else
  {
    output 1/nextNum
  }
}
```

```c
#include <stdio.h>
/*******************************\
  Print out the reciprocals of
  numbers entered. Quit when 0
  is entered
\*******************************/

int main()
{
  float nextNum;

  while (1)
  {
    scanf("%f", &nextNum);
    if (nextNum==0.0)
    {
      break;
    }
    else
    {
      printf("%f\n", 1/nextNum);
    }

  }
  return 0;
}
```

Example: **addpos.c**

Read in numbers, and add only the positive ones. Quit when input is 0

```
set sum to 0

loop
{
   input number
   if (number is zero)
   {
      exit loop
   }
   else if ( number is positive)
   {
      add number to sum
   }
}

output sum
```

Example: **addpos.c** (cont)

```
Read in numbers, and add
only the positive ones.  Quit
when input  is 0

set sum to 0

loop
{
   input number
   if (number is zero)
   {
       exit loop
   }
   else if ( number is positive)
   {
       add number to sum
   }
}

output sum
```

```c
include <stdio.h>

/*****************************
** Read in numbers, and add
** only the positive ones.
** Quit when input is 0
*****************************/

int main()
{
  float num, sum = 0.0;










  printf("sum = %f\n", sum);
  return 0;
}
```

Example: **addpos.c** (cont)

Read in numbers, and add only the positive ones. Quit when input is 0

```
set sum to 0


loop
{
   input number
   if (number is zero)
   {
      exit loop
   }
   else if ( number is positive)
   {
      add number to sum
   }

}

output sum
```

scanf returns EOF if an end of file occurs; otherwise it returns the number of items converted and assigned

```
{
   float nu      um = 0.0;

   while (scanf("%f", &num) > 0)
   {







      sum += num;
   }

   printf("sum = %f\n", sum);
   return 0;
}
```

Example: **addpos.c** (cont)

Read in numbers, and add only the positive ones. Quit when input is 0

```
set sum to 0


loop
{
  input number
  if (number is zero)
  {
    exit loop
  }
  else if ( number is positive)
  {
   add number to sum
  }
}

output sum
```

```c
include <stdio.h>

/*****************************
** Read in numbers, and add
** only the positive ones.
** Quit when input is 0
*****************************/


int main()
{
  float num, sum = 0.0;

  while (scanf("%f", &num) > 0)
  {
    if (num == 0)
      break;




    else if (num > 0)
      sum += num;
  }

  printf("sum = %f\n", sum);
  return 0;
}
```

Example: **addpos.c** (cont)

Read in numbers, and add only the positive ones. Quit when input is 0

```
set sum to 0

loop
{
  input number
  if (number is zero)
  {
     exit loop
  }
  else if ( number is positive)
  {
     add number to sum
  }
}

output sum
```

```
include <stdio.h>

/****************************
** Read in numbers, and add
** only the positive ones.
** Quit when input is 0
****************************/

int main()
{
  float num, sum = 0.0;

  while (scanf("%f", &num) > 0)
  {
    if (num == 0)
      break;




    else if (num > 0)
      sum += num;
  }

  printf("sum = %f\n", sum);
  return 0;
}
```

Example: **addpos.c** (cont)

These comparisons are OK despite `num` being of type `float`

```
include <stdio.h>

/****************************
** Read in numbers, and add
** only the positive ones.
** Quit when input is 0
****************************/


int main()
{
  float num, sum = 0.0;

  while (scanf("%f", &num) > 0)
  {
    if (num == 0)
      break;

    else if (num > 0)
      sum += num;
  }

  printf("sum = %f\n", sum);
  return 0;
}
```

Example: **addpos.c** (cont)

Read in numbers, and add only the positive ones. Quit when input is 0

```
set sum to 0


loop
{
   input number
   if (number is zero)
   {
      exit loop
   }
   else if ( number is positive)
   {
      add number to sum
  }
}

output sum
```

```c
include <stdio.h>

/****************************
** Read in numbers, and add
** only the positive ones.
** Quit when input is 0
*****************************/


int main()
{
  float num, sum = 0.0;

  while (scanf("%f", &num) > 0)
  {
    if (num == 0)
      break;


    else if (num > 0)
      sum += num;
  }


  printf("sum = %f\n", sum);
  return 0;
}
```

scanf and while -- Example 1

```
float num;

while (scanf("%f", &num) > 0)
{

    ...etc...etc...etc...

}
```

Input: 45.2

Result: 1

`scanf` and **`while`** -- Example 1 (cont)

```
float num;

while (scanf("%f", &num) > 0)
{

    ...etc...etc...etc...

}
```

Input: -5

Result: 1

`scanf` and **`while`** -- Example 1 (cont)

```
float num;

while (scanf("%f", &num) > 0)
{

    ...etc...etc...etc...

}
```

Input: **0**

Result: **1**

`scanf` and `while` -- Example 1 (cont)

```
float num;

while (scanf("%f", &num) > 0)
{

    ...etc...etc...etc...

}
```

Input: `c`

Result: `0`

scanf and **while** -- Example 1 (cont)

```
float num;

while (scanf("%f", &num) > 0)
{

    ...etc...etc...etc...

}
```

Input: Dog

Result: 0

scanf and **while** -- Example 1 (cont)

```
float num;

while (scanf("%f", &num) > 0)
{

    ...etc...etc...etc...

}
```

Input: **^Z or ^D**
(depending on the operating system)

Result: EOF
(usually has value **-1**, but it can be any negative number)

scanf -- Example 2

```
int val;
float x, y, z;

val = scanf(“%f %f %f”, &x, &y, &z);
printf(“%d\n”, val);
```

Input: **42.5 -5 23**

Output: **3**

scanf -- Example 2 (cont)

```
int val;
float x, y, z;

val = scanf(“%f %f %f”, &x, &y, &z);
printf(“%d\n”, val);
```

Input: 42.5 -5 c

Output: 2

scanf -- Example 2 (cont)

```
int val;
float x, y, z;

val = scanf(“%f %f %f”, &x, &y, &z);
printf(“%d\n”, val);
```

Input: `42.5 c 23`

Output: `1`

scanf -- Example 2 (cont)

```
int val;
float x, y, z;

val = scanf(“%f %f %f”, &x, &y, &z);
printf(“%d\n”, val);
```

Input: `man 2 wolf`

Output: `0`

Nested Loops

- Loops can be placed inside other loops
- The **`break`** statement applies to the innermost enclosing **`while`** or **`for`** statement

Example: **rect.c**

Print an m by n rectangle of asterisks

```
input width and height

for each row
{
   for each column in the current
     row
   {
      print an asterisk
   }
  start next row
}
```

Example: **rect.c** (cont)

Print an m by n rectangle of asterisks

input width and height

```
for each row
{
   for each column in the current
     row
   {
      print an asterisk
   }
  start next row
}
```

```
#include <stdio.h>

 /*  Print an m-by-n rectangle of
    asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  return 0;
}
```

Example: **rect.c** (cont)

Print an m by n rectangle of asterisks

```
input width and height

for each row
{
   for each column in the current
     row
   {
      print an asterisk
   }
  start next row
}
```

```c
#include <stdio.h>

/*  Print an m-by-n rectangle of
    asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {







  }
  return 0;
}
```

Example: **rect.c** (cont)

Print an m by n rectangle of asterisks

```
input width and height

for each row
{
  for each column in the current
    row
  {
    print an asterisk
  }
  start next row
}
```

```c
#include <stdio.h>

/*  Print an m-by-n rectangle of
    asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {
    for (colc=0; colc < numcol; colc++)
    {


    }

  }
  return 0;
}
```

Example: **rect.c** (cont)

```
Print an m by n rectangle of
asterisks


input width and height


for each row
{
  for each column in the current
    row
  {
    print an asterisk
  }
  start next row
}
```

```c
#include <stdio.h>

/*  Print an m-by-n rectangle of
    asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {
    for (colc=0; colc < numcol; colc++)

    {
      printf("*");
    }

  }
  return 0;
}
```

Example: **rect.c** (cont)

```
Print an m by n rectangle of
asterisks


input width and height


for each row
{
  for each column in the current
    row
  {
    print an asterisk
  }
  start next row
}
```

```c
#include <stdio.h>

/*  Print an m-by-n rectangle of
    asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {
    for (colc=0; colc < numcol; colc++)

    {
      printf("*");
    }

    printf("\n");
  }
  return 0;
}
```

Example: **rect.c** (cont)

```
Print an m by n rectangle of
asterisks


input width and height


for each row
{
   for each column in the current
     row
   {
      print an asterisk
   }
   start next row
}
```

```c
#include <stdio.h>

/*  Print an m-by-n rectangle of
    asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {
    for (colc=0; colc < numcol; colc++)

    {
      printf("*");
    }

    printf("\n");
  }
  return 0;
}
```

Example: **rect.c** (cont)

Print an m by n rectangle of asterisks

algorithm

```
input wi        d height

for each row
{
   for each column in the current
     row
   {
      print an asterisk
   }
  start next row
}
```

program

```
#include <stdio.h>

/*   Print an m-by-n rectangle of
     asterisks */

int main()
{
  int rowc, colc, numrow, numcol;

  pri                  th: ");
  scan                  );
  pri              ght: ");
  scanf   %d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {
    for (colc=0; colc < numcol; colc++)

    {
      printf("*");
    }

    printf("\n");
  }
  return 0;
}
```

Variation: **rect2.c**

Print an m by n rectangle of asterisks

```
input width and height

for each row
{
   for each column in the current
     row
   {
     print an asterisk
   }
  start next row

}
```

```c
#include <stdio.h>

/*  Print an m-by-n rectangle of
    asterisks */
int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  rowc = 0;
  while (rowc < numrow)
  {
    for (colc=0; colc < numcol; colc++)

    {
      printf("*");
    }

    printf("\n");
    rowc++;
  }
return 0;
}
```

Variation: **rect3.c**

Print an m by n rectangle of asterisks

```
input width and height

for each row
{
   for each column in the current
     row
   {
      print an asterisk
   }

  start next row
}
```

```
#include <stdio.h>
/* Print an m-by-n rectangle of
   asterisks */
int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

for (rowc=0; rowc < numrow; rowc++)
  {
    colc = 0;
    while (1)
    {
      printf("*");
      colc++;
      if (colc == numcol)
       { break; }
    }
    printf("\n");
  }
  return 0;
}
```

Variation: **rect3.c (cont)**

Print an m by n rectangle of asterisks

```
input width and height

for each row
{
                                        nt

  start next row
}
```

The innermost enclosing loop for this break is the while-loop

```
#include <stdio.h>
/* Print an m-by-n rectangle of
   asterisks */
int main()
{
  int rowc, colc, numrow, numcol;

  printf("\nEnter width: ");
  scanf("%d", &numcol);
  printf("\nEnter height: ");
  scanf("%d", &numrow);

  for (rowc=0; rowc < numrow; rowc++)
  {
    colc = 0;
    while (1)
    {
      printf("*");
      colc++;
      if (colc == numcol)
       { break; }
    }
    printf("\n");
  }
  return 0;
}
```

Reading

- King
  - Chapter 6, except Section 6.2
- Deitel and Deitel
  - Chapter 3, Section 3.7
  - Chapter 4,
    - Sections 4.1 to 4.6
    - Sections 4.8 to 4.11