Determining Average Magnitude

- Suppose we want to calculate the average **apparent brightness** of a list of five star magnitude values
  - Can we do it?
    - **Yes, it would be easy**

- Suppose we want to calculate the average apparent brightness of a list of 8,479 stars visible from earth
  - Can we do it
    - Yes, but it would be gruesome without the use of iteration

C++ Iterative Constructs

- Three constructs
  - while statement
  - for statement
  - do-while statement

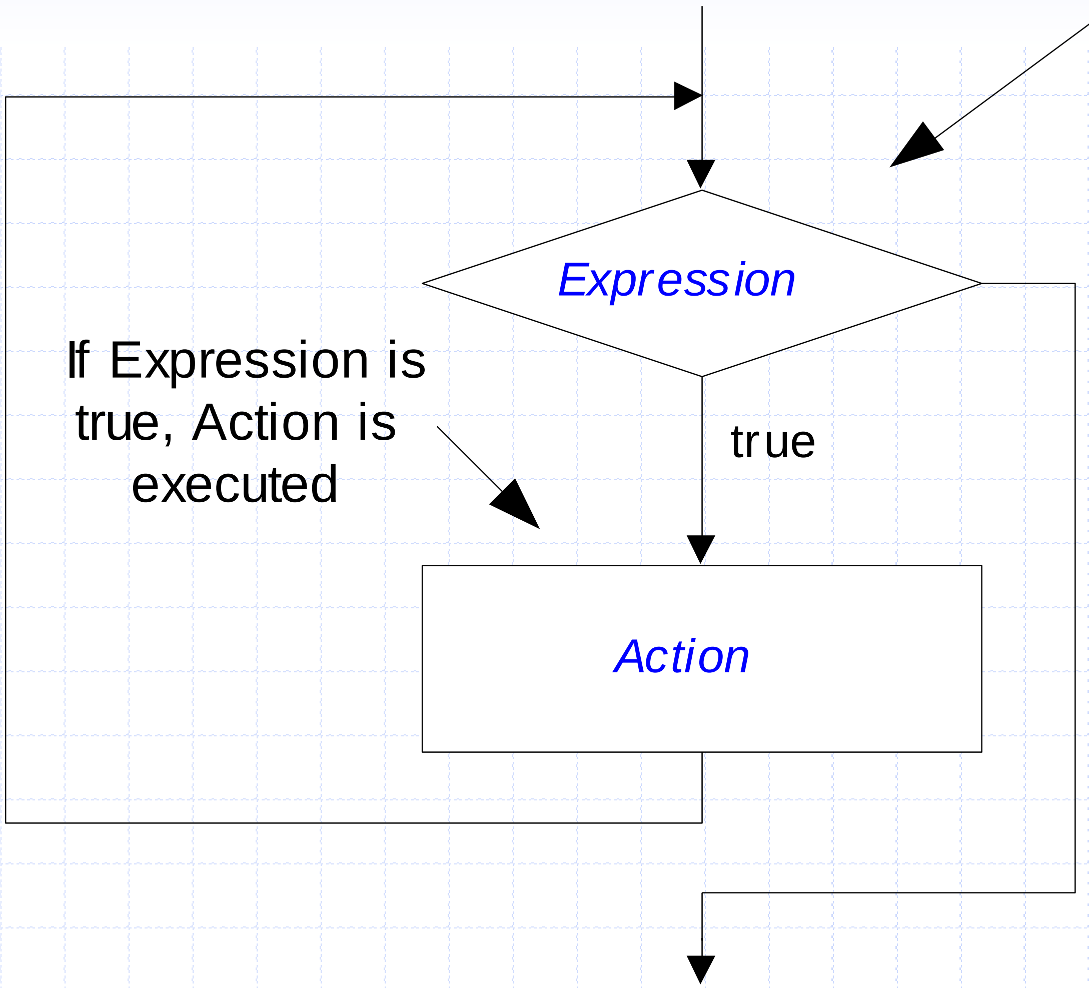
While Syntax

Logical expression that determines whether the action is to be executed

Action to be iteratively performed until logical expression is false

while (*Expression***)** *Action*

While Semantics



Expression is evaluated at the start of each iteration of the loop

If Expression is true, Action is executed

If Expression is false, program execution continues with next statement

Computing an Average

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0
sum	0

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0
sum	0

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0
sum	0
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0
sum	0
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0
sum	1
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	1
sum	1
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	1
sum	1
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	1
sum	1
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	1
sum	1
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	1
sum	6
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	6
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	6
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	6
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	6
value	3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	9
value	3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	9
value	3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	9
value	3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	9
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	9
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	10
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
average	2.5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
average	2.5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Suppose input contains: 1 5 3 1 6

Execution Trace

Stays in stream until extracted

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
   double value;
   cin >> value;
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

Power of Two Table

```
const int TableSize = 20;

int i = 0;
long Entry = 1;

cout << "i" << "\t\t" << "2 ** i" << endl;

while (i < TableSize) {
   cout << i << "\t\t" << Entry << endl;
   Entry = 2 * Entry;
   ++i;
}
```

Better Way of Averaging

```
int numberProcessed = 0;
double sum = 0;
double value;
while (  cin >> value  ) {
   sum += value;
   ++numberProcessed;
}
double average = sum / numberProcessed ;
cout  << "Average: " << average << endl;
```

The value of the input operation corresponds to true only if a successful extraction was made

What if list is empty?

Even Better Way of Averaging

```
int numberProcessed = 0;
double sum = 0;
double value;
while (  cin >> value  ) {
   sum += value;
   ++numberProcessed;
}
if ( numberProcessed > 0 ) {
   double average = sum / numberProcessed ;
   cout  << "Average: " << average << endl;
}
else {
   cout << "No list to average" << endl;
}
```

The For Statement

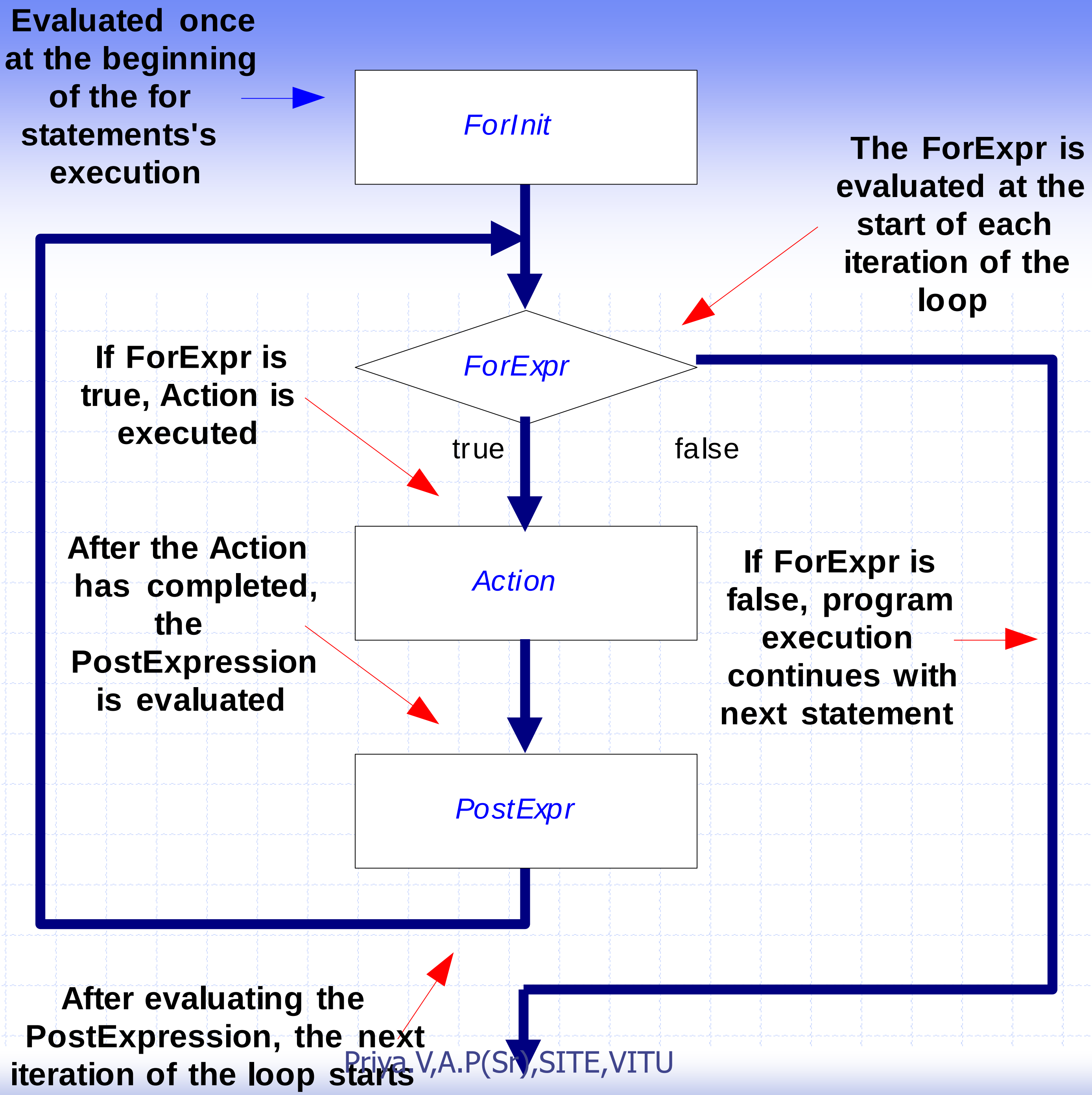
- Syntax

  for (*ForInit* ; *ForExpression*; *PostExpression*)

  Action

- Example

```
for (int i = 0; i < 3; ++i) {
    cout << "i is " << i << endl;
}
```



Evaluated once at the beginning of the for statements's execution

The ForExpr is evaluated at the start of each iteration of the loop

If ForExpr is true, Action is executed

After the Action has completed, the PostExpression is evaluated

If ForExpr is false, program execution continues with next statement

After evaluating the PostExpression, the next iteration of the loop starts

Execution Trace

i | 0

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

Execution Trace

i 0

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

Execution Trace

```
i  0
```

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
```

Execution Trace

i 0

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
```

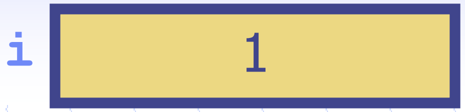
Execution Trace

i 1

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
```

Execution Trace



```
for (int i = 0; i < 3; ++i) {
  cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

Execution Trace

i 1

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
```

Execution Trace

i | 1

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
```

Execution Trace

i 2

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
```

Execution Trace

i 2

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
```

Execution Trace

i 2

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
i is 2
```

Execution Trace

i 2

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
i is 2
```

Execution Trace

i `3`

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
i is 2
```

Execution Trace

i $\boxed{3}$

```
for (int i = 0; i < 3; ++i) {
   cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
i is 2
```

Execution Trace

i 3

```
for (int i = 0; i < 3; ++i) {
  cout << "i is " << i << endl;
}
cout << "all done" << endl;
```

```
i is 0
i is 1
i is 2
all done
```

Table Revisiting

```
const int TableSize = 20;

long Entry = 1;

cout << "i" << "\t\t" << "2**i" << endl;

for (int i = 0; i <= TableSize; ++i) {
   cout << i << "\t\t" << Entry << endl;
   Entry *= 2;
}
```

Table Revisiting

```
const int TableSize = 20;

long Entry = 1;

cout << "i" << "\t\t" << "2**i" << endl;

for (int i = 0; i < TableSize; ++i) {
   cout << i << "\t\t" << Entry << endl;
   Entry = 2 * Entry;
}

cout << "i is" << i << endl; // illegal
```

The scope of i is limited to the loop!

```
int Counter1 = 0;
int Counter2 = 0;
int Counter3 = 0;
int Counter4 = 0;
int Counter5 = 0;

++Counter1;

for (int i = 1; i <= 10; ++i) {

    ++Counter2;

    for (int j = 1; j <= 20; ++j) {
        ++Counter3;
    }

    ++Counter4;
}

++Counter5;

cout << Counter1 << " " << Counter2 << " "

  << Counter3 << " " << Counter4 << " "

  << Counter5 << endl;
```

For Into While

- Observation
  - The for statement is equivalent to

```
{
  ForInit;
  while (ForExpression) {
        Action;
        PostExpression;
  }
}
```

Counting Characters

```
int NumberOfNonBlanks = 0;
int NumberOfUpperCase = 0;
char c;
while (cin >> c) {
   ++NumberOfNonBlanks;
   if ((c >= 'A') && (c <= 'Z')) {
       ++NumberOfUpperCase;
   }
}
cout << "Nonblank characters: " << NumberOfNonBlanks
     << endl << "Uppercase characters: "
     << NumberOfUpperCase << endl;
```

Only extracts nonblank characters

Counting All Characters

```
char c;
int NumberOfCharacters = 0;
int NumberOfLines = 0;
while ( cin.get(c) ) {
   ++NumberOfCharacters;
   if (c == '\n') {
       ++NumberOfLines
   }
}
cout << "Characters: " << NumberOfCharacters
     << endl << "Lines: " << NumberOfLines
     << endl;
```

Extracts all characters

Iteration Do's

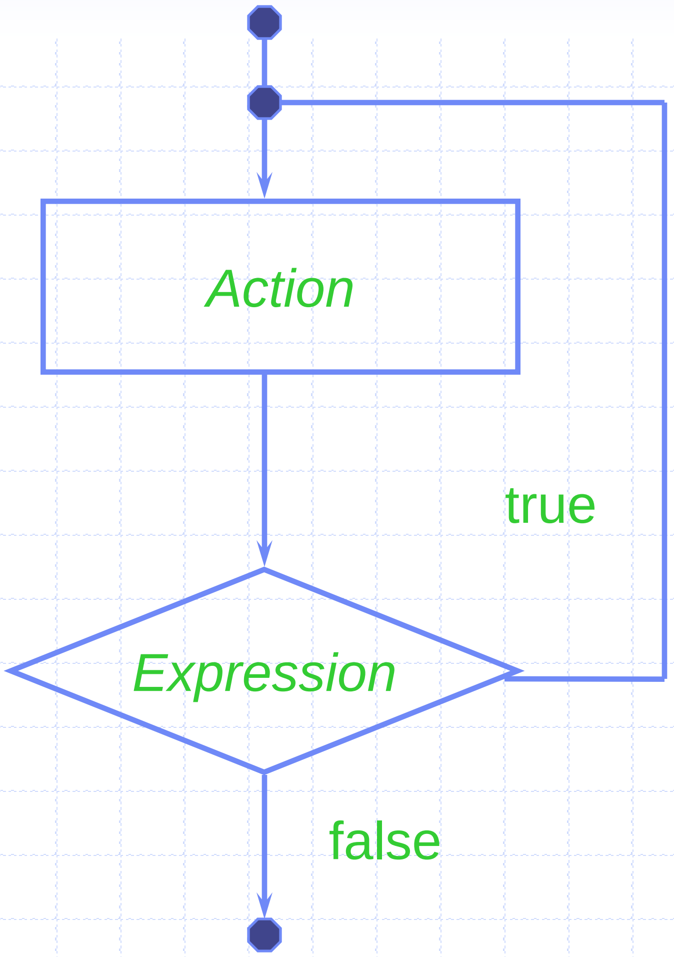
- Key Points
  - Make sure there is a statement that will eventually terminate the iteration criterion
    - The loop must stop!
  - Make sure that initialization of loop counters or iterators is properly performed
  - Have a clear purpose for the loop
    - Document the purpose of the loop
    - Document how the body of the loop advances the purpose of the loop

The Do-While Statement

- Syntax

  `do` *Action*

  `while` (*Expression*)
- Semantics
  - Execute *Action*
  - If *Expression* is true then execute *Action* again
  - Repeat this process until *Expression* evaluates to false
- *Action* is either a single statement or a group of statements within braces



Waiting for a Proper Reply

```
//This program averages 3 test scores. It repeats as many times as  the user wishes
        int score1, score2, score3;
        float average;
        char again;
        do
        {
                cout << "Enter 3 scores and I will average them: ";
                cin >> score1 >> score2 >> score3;
                average = (score1 + score2 + score3) / 3.0;
                cout << "The average is " << average << ".\n";
                cout << "Do you want to average another set? (Y/N) ";
                cin >> again;
        } while (again == 'Y' || again == 'y');
```