Loops
Reading data from file

Increment(++), Decrement (--)

- `n++;` means increase n by 1
- `++n;` means the same
- `n--;` means decrease n by 1
- `--n;` also means decrease n by 1

Increment(++), Decrement (--)

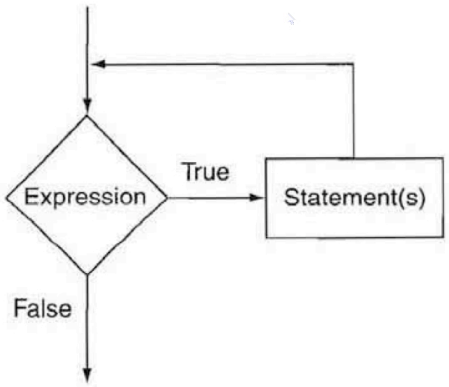
- PREFIX increment : `++n`. Happens before n is evaluated in the expression
  - `int n=1; int m=++n + 3;`
- POSTFIX increment: `n++`. Happens after n is evaluated in the expression
  - `int n=1; int m=n++ + 3;`
- same for decrement: PREFIX and POSTFIX

Loops - `while`

- a block of statements that repeats.

```
while (expr_cond){
   statement;
   statement;
   .....
}
```

- no "else"
- `expr_cond` evaluated before each iteration of the loop
- if false, the loop does not execute, and the program jumps at the end of the loop



Loops - while

- the loops should change/update the expr_cond
- might never execute
- might never finish

```
int counter=0; while (counter<10){
   counter++;
   cout<<" \n at iteration "<<counter;

}
```

- pay special attention to the first and the last expr_cond
- also figure out expr_cond after while exits

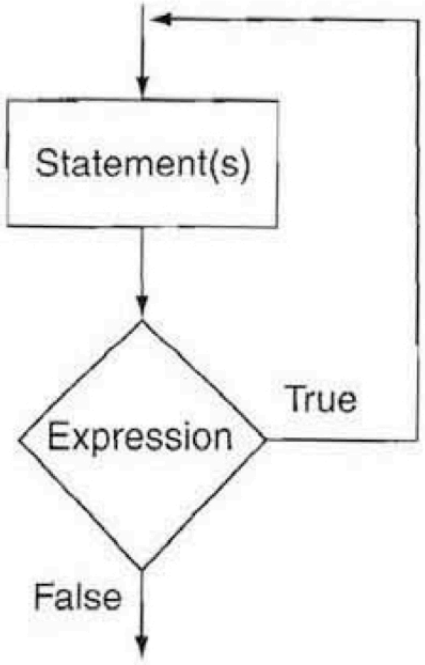
while loop for input validation

```
n=-1;
while (n<0 || n>=10){
    cout<<"\n\n give me a non-negative integer less than 10>";
    cin>>n;
}
cout<<"\n you gave me n="<<n;
cout << "\n";
```

do - while

```
do {
     statement;
     statement;
     ...

}while(expr_cond);
```



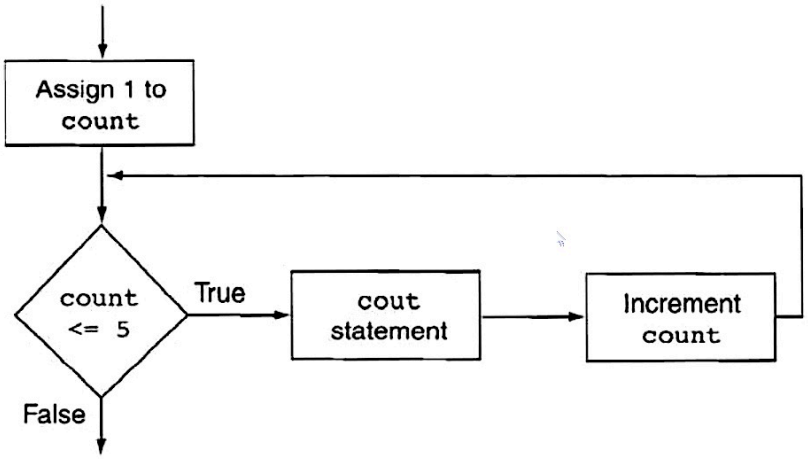
- post-test loop = condition is evaluated at the end
  - opposite to the while loop, which is pre-test loop
- the statements block is executed first time before evaluating the `expr_cond`

do-while menu

- display menu, ask user for a choice
- read user choice, validate, check for sentinel (exit option)
- do some operations given the user choice
- repeat

for

```
for (expr_init; expr_cond; expr_update){
    statement;
    statement;
    ...
}
```



for

```
for (expr_init; expr_cond; expr_update){
    statement;
    statement;
    ...
}
```

- usually know how many times want to execute the loop
  - or know the the iterations i=1,2,3…,10 that might be executed
- while, do-while : dont know how many times the loop will execute, waiting for expr_cond to be false

`for` loop

```
for (expr_init; expr_cond; expr_update){
   ...
}
```

- 1) `expr_init` : done only once, first thing
- 2) `expr_cond` : evaluated at the beginning of each iteration
  - if false, loop ends
- 3) body of `statements` inside the loop: each iteration
- 4) `expr_update`: last operation in each iteration
  - avoid modifying `expr_update` in the loop body

break, continue

- `break` : exit the loop (jumps right after the loop/block)
- `continue` : jump to the next iteration of the loop
  - does not finish the current iteration

Nested loops

```
for (i=0;i<N;i++)

  for(j=0;j<M;j++){
    cout << "element at row="<<i<<" and column =";
    cout<<j<<"is"<<A[i][j];

  }
```

Nested loops

```
//matrix multiplication A(N,P); B(P,M); C=A*B

for (int i=0;i<N;i++){
   cout <<"\nrow="<<i<<". columns:";
   for(int j=0;j<M;j++){
      cout << j<<" ";
      for(int k=0;k<P;k++){
         C[i][j]+=A[i][k]*B[k][j];
      }
   }
}
```

multiple/no statements expressions

- multiple statements in `expr_init, expr_update`

```
for (i=0,j=0;i<N && j<M;i++,j++){
     cout << "diagonal element at"<<i<<","<<j;;
     cout<<" is"<<A[i][j];
}
```

- no statements in `expr_init, expr_update`

```
int i=0,j=0;
for (;i<N && j<M;){
     cout << "diagonal element at"<<i<<","<<j;;
     cout<<" is"<<A[i][j];
     i++; j=i;
}
```

Data from Files

Files

- use class ifstream/ofstream/fstream
  - `#include <fstream>`
  - `fstream filehandle ;`
- use class methods to manipulate data
  - `filehandle.open()`
  - `filehandle.close()`
  - `filehandle.get()`
  - `filehandle.getline()`
  - `filehandle >> a`
  - `filehandle << a`
  - …
- data has to be well formatted

Reading from file

```
int a;
file.open ("myfile.txt");
while (file>>a){
    cout<<" read form file a="<<a<<"\n";
}
file.close();
```

- after `file>>a` fails, not clear where the `get` pointer is
  - for now : only works for well formatted files

reading from file

- `filehandle.get()` : get a character
- `filehandle.peek()` : look up a character
  - `get` (read) pointer does not advance
- `filehandle.getline(char*, int)` : get a line
- `filehandle.ignore(int, int)` : ignore a sequence of chars

Files : change pointer location

- look/move the get pointer (reading location)
  - `filehandle.tellg()`
  - `filehandle.seekg()`
- look/move the put pointer (writing location)
  - `filehandle.tellp()`
  - `filehandle.seekp()`

Writing to file

```
fstream file;

file.open("writingfile.txt", fstream::in | fstream::out |
fstream::trunc);

cout <<"\n\n\n WRITING DATA TO FILE  filehandle="<<file<<endl;

for (int i=0;i<10;i++){

   file<<"writing now "<<i<<"\n";

}

file.close();
```