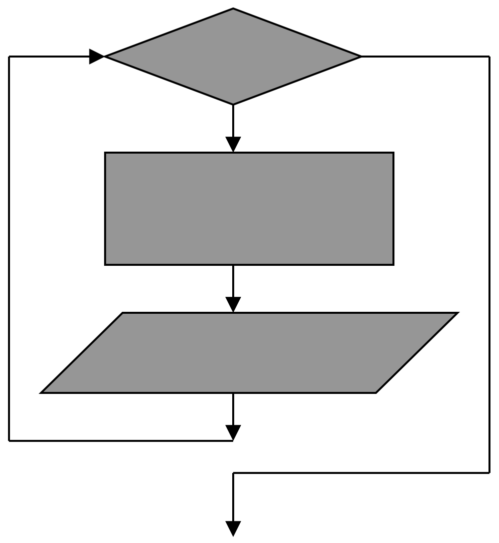
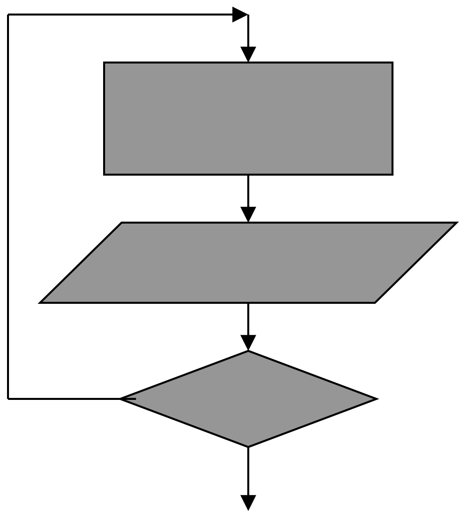
C++ Control Structures

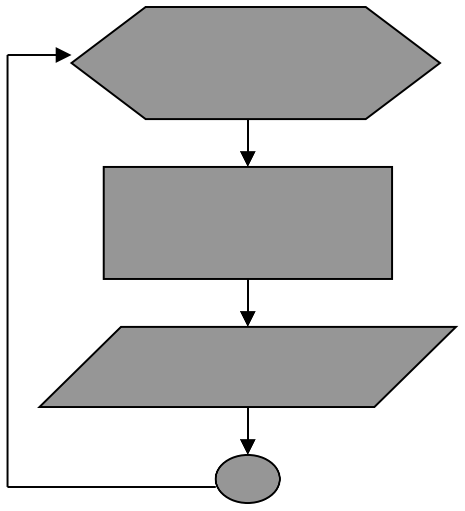
Part III - Loops

Repetition - Loops

- A set of commands need to executed multiple times.

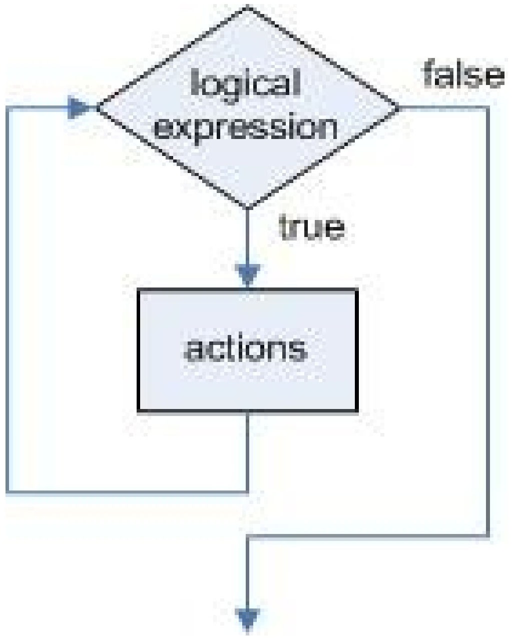






while loops

```
while ( logical expression ) {
  commands;
   // commands should
   // update the logical
   // expression
}
```



while loops

- The decision is made at the top of the loop
- While the expression is true, continue executing the body of the loop
- Once the condition is false, skip the body of the loop and continue with the commands that follow the loop

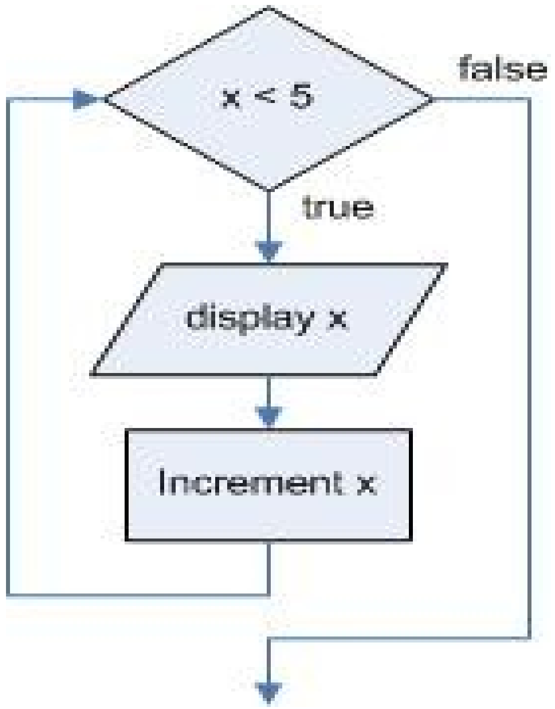
while loops

Syntax Notes:

- No semicolon
- Braces are not required if there is exactly one statement in the body of the loop
- Variables declared inside a block are local to that block!

Example

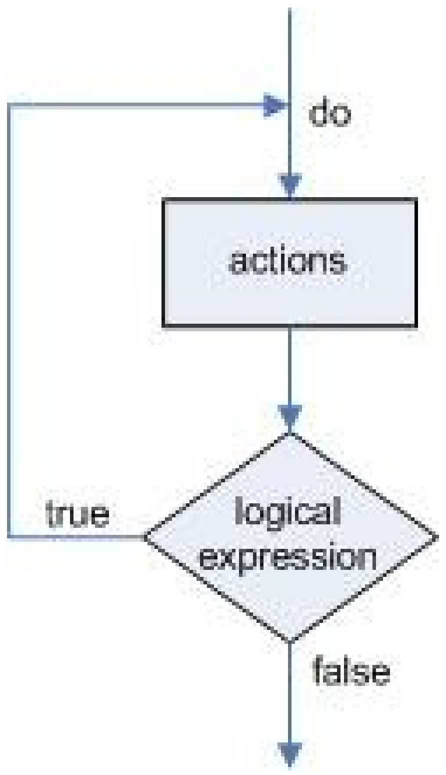
```
while ( x < 5 ) {
  cout << x << endl;
  x = x + 1;
} // end while loop
```



- Note: x must have an initial value
- Forgetting to increment x would cause an infinite loop!

do-while loops

```
do {
    commands;
} while( logical expression );
```



do-while loops

- The decision is made at the bottom of the loop – this ensures that the body of the loop will be executed at least once
- While the expression is true, continue executing the body of the loop. Once the condition is false, continue with the commands that follow the loop.

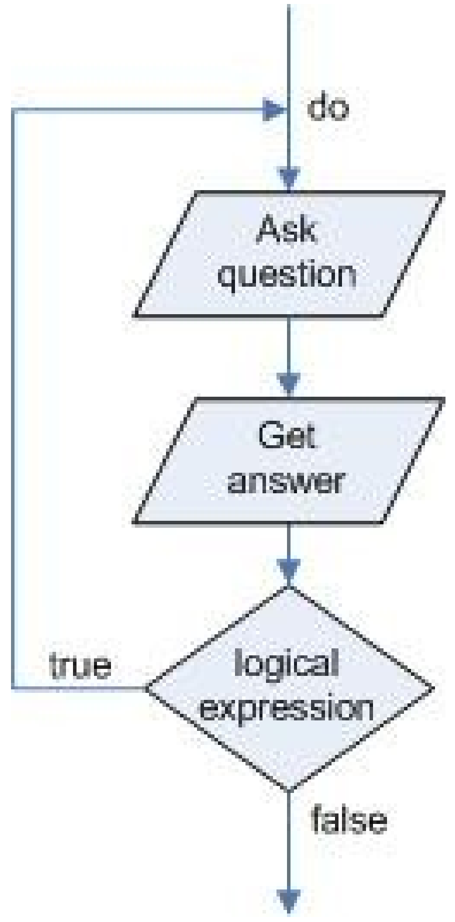
do-while loops

Syntax Notes:

- The semicolon <u>is</u> necessary
- Braces are not required if there is only one statement in the body of the loop

Example

```
do {
    cout << “Enter y for yes or n for no”;
    cin >> answer;
} while( answer != ‘y’ && answer != ‘n’ );
```

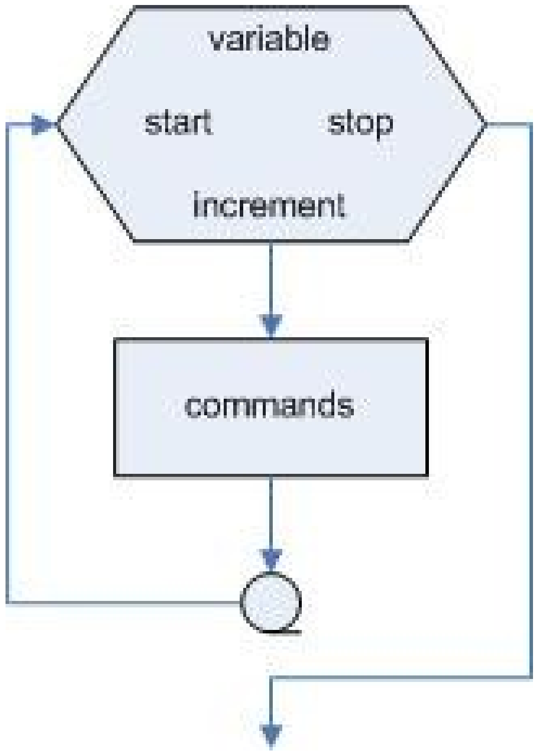


for loops

for (*before first iteration* **;** *logical expression* **;**
after each iteration) {

commands ;

}



for loops

- Method of writing counter-controlled loops in a more concise way
- Best for use when you know how many times the loop will execute before you enter the loop

for loops

Syntax Notes:

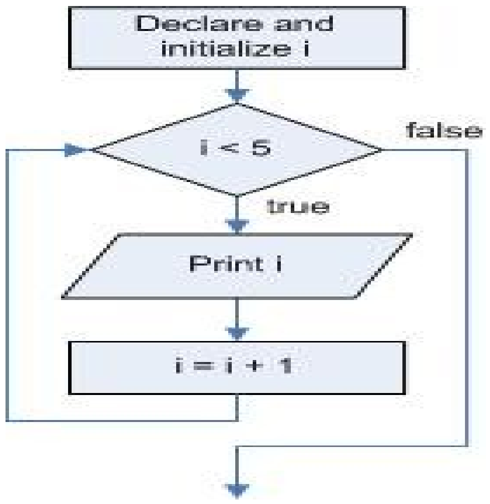
- No semicolon
- Braces are not required if there is only one statement in the body of the loop
- Variables declared in the parenthesis are local to the loop block

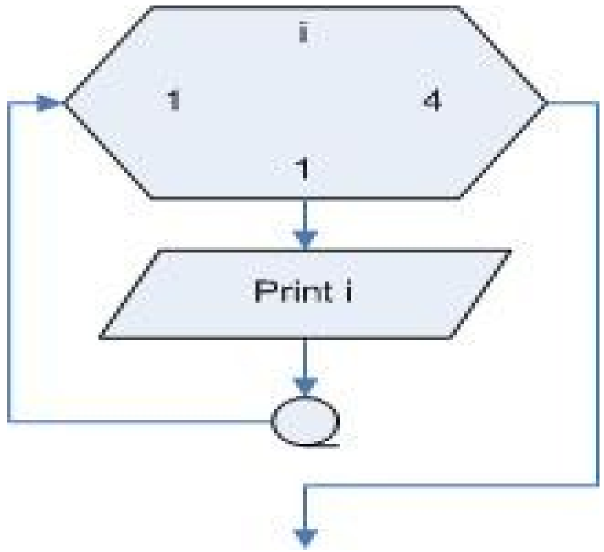
Example

```
int i = 1;
while ( i < 5 ) {
   cout << i << "\n";
   i = i + 1;
}
```

Will produce the same output as…

```
for ( int i = 1; i < 5; ++i )
   cout << i << "\n";
```





Note

- The body of any loop structure can contain any valid C++ command or control structure – decisions, switch statements, or other loops.

break and continue

- The break command exits the current code block.
- The continue command causes the program to skip the remaining commands in the body of the loop, and then continue with the next loop iteration.
- I suggest that you don’t use these except for breaks in switch statements