The **while** loop

```c
while ( condition )
    statement_to_repeat;
```

OR

```c
while (condition)
{
    statement_to_repeat1;
    . . .
    statement_to_repeatN;
}
```

**condition** – a Boolean expression

*Just a reminder from our earlier if-else slides*

- Boolean expressions are either true or false
- Conditions often consist of **comparisons**
  - age ≥ 21  // can buy drinks
  - age < 4   // can ride subway for free
  - year = 2   // you are a sophomore

How can we output “Hello world” 4 times?

```c
int x=4;
while ( x>0 )
{
    cout << "Hello world.\n";
    x--;
}
```

Remember `x--;` same as `x=x-1;`
Repeats until `x≤0`

Alternatives to “linear execution”

Repeated actions

```c
> ./myProgram
```

Execution of while loop

- If condition is **true**, enter while loop
  - Complete all statements in block
  - Return to top (re-evaluate condition)
- Otherwise, continue to statements beyond loop
Execution of while loop

- If condition is true, enter while loop
  - Complete all statements in block
  - Return to top (re-evaluate condition)
- Otherwise, continue to statements beyond loop

```cpp
int x = 2;
while (x > 0) {
    x--; // How many "Hello world"'s are output?
    cout << "Hello world." << endl;
}
```

What does this code do?

```cpp
int b = 3;
while (b < 6) {
    cout << b;
    b += 2;
}
```

What code will do this for us?

```
> ./myProgram.out
1 mississippi
2 mississippi
3 mississippi
4 mississippi
5 mississippi
>
```

What does this code do?

```cpp
int b = 6;
while (b != 3) {
    cout << b;
    b -= 2;
}
```

Beware infinite loops!

- Loops that never stop are called infinite loops
- Typically, write code so each loop will stop

```cpp
int i = 1;
while (i < 10) {
    cout << "Hello\n";
    i++; // Infinite loop
}
```

Beware the misplaced ;

Placing a semicolon after the parentheses of a while loop causes an empty statement as the body of the loop

```
int i = 1;
while (i < 10);
{
    cout << "Hello\n";
    i++; // Infinite loop
}
Counters and accumulators

How can we write a program to compute 1+2+3+4+5?

Use two variables:
- **Counter**: Keep track of number of loop repeats
- **Accumulator**: Keep track of running sums

---

do-while loop

- while evaluates condition, then performs statements if condition is **true**
- do-while performs statements, then evaluates condition to determine whether to perform statements again

```java
do {
  statement1;
  ...  
  statement N;
} while ( condition );
```

---

What does this code do?

```java
int main () {
  int a;
  cout << "Input a number: ";
  cin >> a;
  do {
    cout << "one ";
    cout << "two\n";
    a-=2;
  } while ( a > 0);
  return 0;
}
```

---

**for loop**

a while loop alternative

```java
for ( init; condition; update )
{
  statement1;
  ...  
  statement N;
}
```

**typical example:**

```java
int i, product=1;
for ( i=1; i<=5; i++ )
{
  product = product*i;
}
```

---

**init** – initializes variable

**condition** – statement about variable, must stay true for loop to keep running

**update** – updates the variable after each loop execution

---

**What does this code do?**

```java
int main () {
  int i, product=1;
  for ( i=1; i<=5; i++ )
  {
    product = product*i;
    cout << i << "! = " << product << endl;
  }
  return 0;
}
```
Picking a loop

- **do-while** if you need to perform the action at least once
- **for** if there is a standard repeated mathematical update to your loop variable (e.g., count++)
- **while** loop for less-standard loop variable updates

“loop variable” is the variable tested by the condition in your given loop