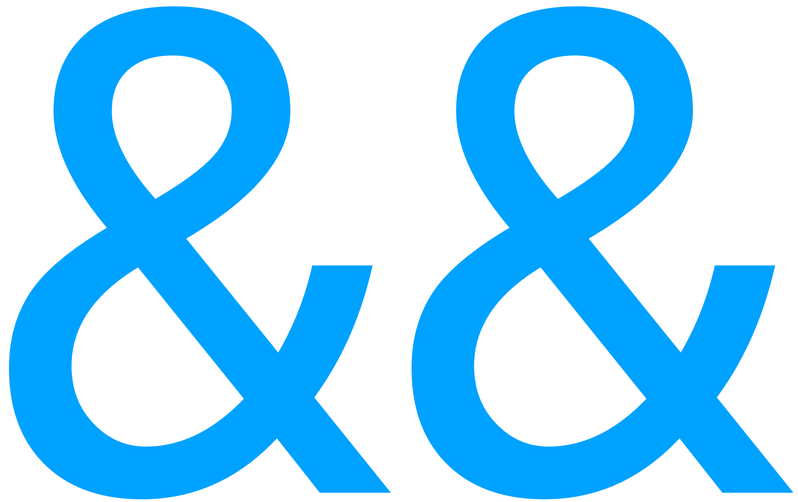
Essential Computing 1

Logical operators



And operator

```
boolean isSunny = true;
boolean isRainy = true;

if( isSunny && isRainy ){
   // show rainbow!
}
```

Or operator

```
boolean isAlarmDead = false;
boolean isBusLate = true;

if( isAlarmDead || isBusLate ){
  // will be late for work
}
```

Negation

```
boolean isLionClose = true;
boolean isCarClose = false;

if( isLionClose && !isCarClose ){
   // eaten alive by lion
}
```

De Morgan's laws

Negating a logical expression is the same as
negating each term and changing the operator.

```
!A || !B   same as  !(A && B)

!A && !B   same as  !(A || B)
```

Negation

```
boolean isLionClose = true;
boolean isCarClose = false;

if( isLionClose && !isCarClose ){
   // eaten alive by lion
}
```

Negation

De Morgan's law

```
boolean isLionClose = true;
boolean isCarClose = false;

if( !( !isLionClose || isCarClose ) ){
  // still eaten alive by lion
}
```

De Morgan's laws

Also true for relational operators

```
A>=B || C!=D    same as    !( A<B && C==D )

E<F && G==H     same as    !( E>=F || G!=H )
```

Short circuit evaluation

Silent Java optimisation

```
int a = 1;
int b = 2;
int c = 3;

if( a > 0 || b > 0 || c > 0 ){
   // code
}
```

Short circuit evaluation

Silent Java optimisation. Only first relational operator is evaluated.

```java
int a = 1;
int b = 2;
int c = 3;

if( a > 0 || b > 0 || c > 0 ){
   // code
}
```

Short circuit evaluation

If calA() returns a number greater than zero then calcB() and calC() will never be called.

```
if( calcA() > 0 || calcB() > 0 || calcC() > 0 ){
   // code
}
```