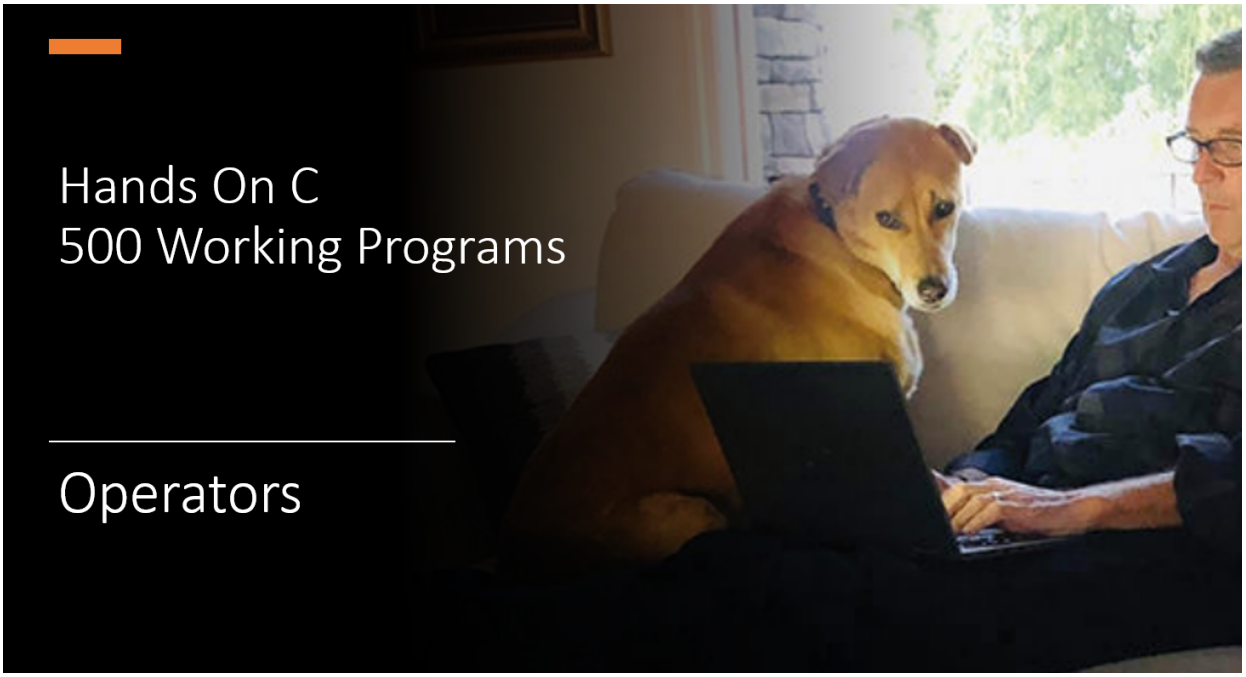

Performing Basic Math Operations

```
Operator     Operation
+            Addition
-            Subtraction
*            Multiplcation
/            Division
```

In [1]:
```c
#include <stdio.h>

int main(void)
{
  int y, slope = 3, intercept = 2, x = 4;

  y = slope * x + intercept;

  printf("Y = %d\n", y);
}
```

```
Y = 14
```

In [2]:
```c
#include <stdio.h>

int main(void)
{
   long seconds_in_a_day = 60 * 60 * 24;

   int hours_in_a_day = seconds_in_a_day / (60 * 60);  // Note parentheses to for

   printf("Seconds in a day %ld\n", seconds_in_a_day);
   printf("Hours in a day %d\n", hours_in_a_day);
}
```

```
Seconds in a day 86400
Hours in a day 24
```

Understanding the C Modulo (Remainder) Operator

In [3]:
```c
#include <stdio.h>

int main(void)
{
    int result, remainder;

    result = 7 / 2;
    remainder = 7 % 2;

    printf("Result %d Remainder %d", result, remainder);
}
```

```
Result 3 Remainder 1
```

Understanding the C Increment Operator

```
variable++ is the same as variable = variable + 1

++variable is also the same as variable = variable + 1

The difference occurs when the expression is used in an assignment
```

In [4]:
```c
#include <stdio.h>

int main(void)
{
    int a = 0, b;

    b = a++;
    printf("a %d b %d", a, b);
}
```

```
a 1 b 0
```

In [5]:
```c
#include <stdio.h>

int main(void)
{
    int a = 0, b;

    b = ++a;
    printf("a %d b %d", a, b);
}
```

```
a 1 b 1
```

Understanding the C Decrement Operator

```
variable-- is the same as variable = variable - 1

--variable is also the same as variable = variable - 1

The difference occurs when the expression is used in an assignment
```

In [6]:
```c
#include <stdio.h>

int main(void)
{
    int a = 0, b;

    b = a--;
    printf("a %d b %d", a, b);
}
```

```
a -1 b 0
```

In [7]:
```c
#include <stdio.h>

int main(void)
{
    int a = 0, b;

    b = --a;
    printf("a %d b %d", a, b);
}
```

```
a -1 b -1
```

In [8]:
```c
#include <stdio.h>

int main(void)
{
  char letter;

  for (letter = 'A'; letter <= 'Z'; letter++)
     putchar(letter);
}
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

In [9]:
```c
#include <stdio.h>

int main(void)
{
   int count = 100;

   while (count >= 0)
     printf("%d ", count--);
}
```

```
100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77 76 75
74 73 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 4
8 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

Understanding a Bitwise Or Operation

```
2     0000 0010
5     0000 0101
      --------- Bitwise OR
7     0000 0111
```

In [10]:
```c
#include <stdio.h>

int main(void)
{
   unsigned int a = 2, b = 5;

   printf("Bitwise Or Result %d", a | b);
}
```

```
Bitwise Or Result 7
```

Understanding a Bitwise And Operation

```
2     0000 0010
5     0000 0101
      --------- Bitwise And
7     0000 0000
```

In [11]:
```c
#include <stdio.h>

int main(void)
{
   unsigned int a = 2, b = 5;

   printf("Bitwise And Result %d", a & b);
}
```

```
Bitwise And Result 0
```

Understanding a Bitwise Exclusive-Or Operation

```
2     0000 0010
7     0000 0111
      --------- Bitwise Exclusive Or
5     0000 0101
```

In [12]:
```c
#include <stdio.h>

int main(void)
{
   unsigned int a = 2, b = 7;

   printf("Bitwise And Result %d", a ^ b);
}
```

```
Bitwise And Result 5
```

Using Shorthand Notation to Apply an Operation to a Variable

```
a += 5 is the same a = a + 5
a *= 2 is the same as a = a * 2
a /= 3 is the same as a = a / 3
```

In [13]:
```c
#include <stdio.h>

int main(void)
{
    int a = 5;
    a += 5;

    printf("Value of a is %d\n", a);

    a *= 2;
    printf("Value of a is %d\n", a);
}
```

```
Value of a is 10
Value of a is 20
```

Understanding the C Conditional Operator

```c
a = 1;              is the same as      a = 1;
b = 2;                                  b = 2;

if (a < b)                              min = (a < b) ? a: b;
  min = a;
else
  min = b;
```

In [14]:
```c
#include <stdio.h>

int main(void)
{
  int a = 1, b = 2, min;

  min = (a < b) ? a: b;
  printf("Minimum value is %d", min);
}
```

```
Minimum value is 1
```

Understanding the C sizeof Operator

In [15]:
```c
#include <stdio.h>

int main(void)
{
  printf("The size of an int is %ld bytes\n", sizeof(int));
  printf("The size of an float is %ld bytes\n", sizeof(float));
  printf("The size of an double is %ld bytes\n", sizeof(double));
  printf("The size of an char is %ld bytes\n", sizeof(char));
}
```

```
The size of an int is 4 bytes
The size of an float is 4 bytes
The size of an double is 8 bytes
The size of an char is 1 bytes
```

Using the C Bitwise Shift Operators

In [16]:
```c
#include <stdio.h>

int main(void)
{
  unsigned int a = 1, b = 1, c = 1;

  int value = a << 1;
  printf("Value %u\n", value);

  value = b << 2;
  printf("Value %u\n", value);

  value = c << 3;
  printf("Value %u\n", value);
}
```

```
Value 2
Value 4
Value 8
```

In [17]:
```c
#include <stdio.h>

int main(void)
{
  unsigned int a = 16, b = 16, c = 16;

  int value = a >> 1;
  printf("Value %u\n", value);

  value = b >> 2;
  printf("Value %u\n", value);

  value = c >> 3;
  printf("Value %u\n", value);
}
```

```
Value 8
Value 4
Value 2
```

Understanding Operator Precidence

In [18]:
```c
#include <stdio.h>

int main(void)
{
  int result = 1 + 2 * 3;
  printf("Result %d\n", result);
}
```

```
Result 7
```

In [19]:
```c
#include <stdio.h>

int main(void)
{
  int result = (1 + 2) * 3;
  printf("Result %d\n", result);
}
```

```
Result 9
```

```
C Operator Precidence High to Low
()   []   .   ->
++   --   +   -   *   &   !   ~   (type)   sizeof
*    /    %
+    -
<<   >>
==   !=
&
^
|
&&
||
?:
=   +=   -=   *=   /=   %=   &=   ^=   |=   <<=   >>=
,
```

What You will Learn Next

```
To perform meaningful work, programs must make decisions.

if (condition)
  statement

if (condition)
  statement
else
  statement
```