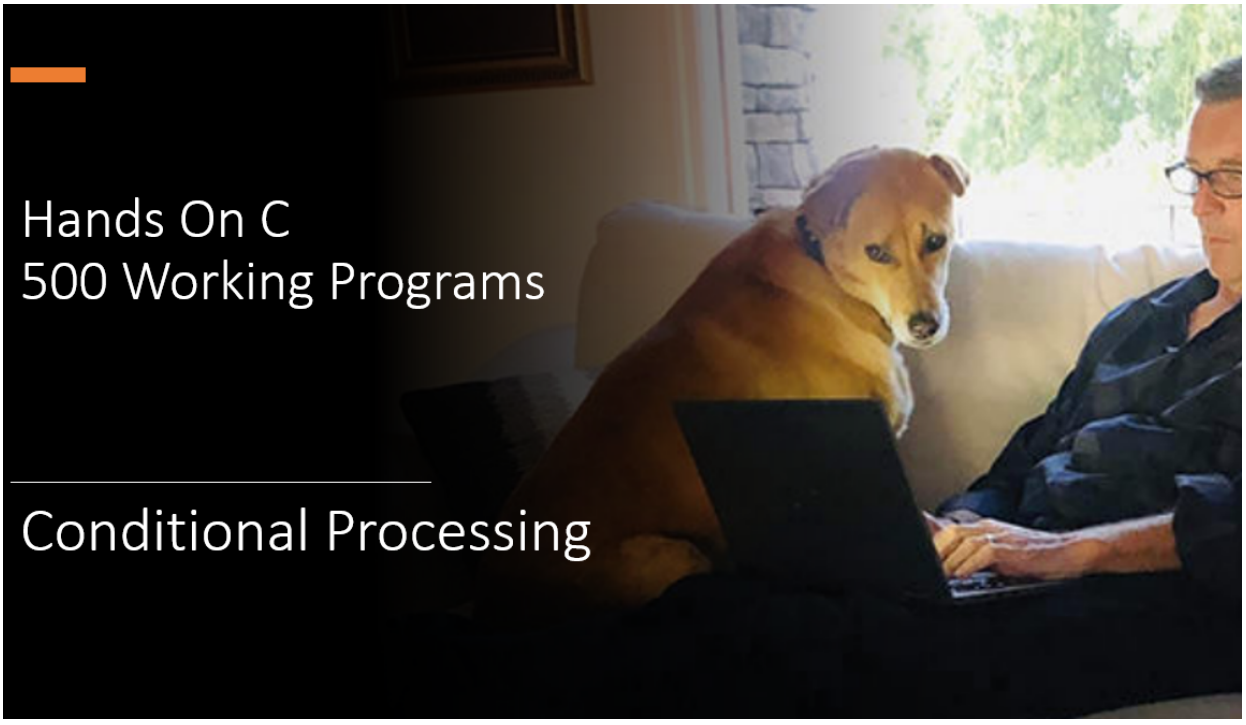

Understanding Conditional Processing

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

int main(void)
{
   int age = 22;

   if (age >= 21)
     printf("Going to Vegas!\n");
   else
     printf("Going to Disneyland!\n");
}
```

```
Going to Vegas!
```

Understanding the C Represents True Values

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

int main(void)
{
  if (1)
    printf("1 is always true\n");

  if (1-0 == 1)
    printf("The expression equals 1 and 1 is true\n");

  if (1-0)
    printf("The expression by itself equals 1 and 1 is true\n");

  if (0)
    printf("This statement will not appear\n");
}
```

```
1 is always true
The expression equals 1 and 1 is true
The expression by itself equals 1 and 1 is true
```

Understanding Simple and Compound Statements

```
Simple statement is a single statement:

   age = 21;

Compound statements appear within braces
```

```
{
  int age = 21;

  printf("She is %d\n" age);
}
```

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

int main(void)
{
    int a = 0, b;  // this is a simple statement

    {
      b = a++;     // this is a compound statement
      printf("a %d b %d", a, b);
    }
}
```

```
a 1 b 0
```

C's if Statement uses Simple and Compound Statements

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

int main(void)
{
    int pet = 1;  // 1 for dog 0 for cat

    if (pet)
      printf("Dogs are great!");
}
```

```
Dogs are great!
```

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

int main(void)
{
    int pet = 1;

    if (pet)
      {
         printf("Dogs are great\n");
         printf("I have 3 dogs\n");
      }
}
```

```
Dogs are great
I have 3 dogs
```

Testing for Equality

```
The single equal sign = is the C assignment operator

The double equal signs == is the test for equality
```

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

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

   if (a == b)
     printf("a and b are equal\n");
}
```

```
a and b are equal
```

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

int main(void)
{
  int a = 0;

  if (a = 2)                        // Bug
    printf("a is equal to 2\n");
}
```

```
a is equal to 2
```

Understanding Relational Tests

```
Operator        Purpose
>               Greater than
<               Less than
>=              Greater than or equal to
<=              Less than or equal to
```

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

int main(void)
{
   int a = 2, b = 3;

   if (a >= b)
       printf("a %d is bigger than b %d\n", a, b);

   if (a < b)
       printf("a %d is less than b %d\n", a, b);
}
```

```
a 2 is less than b 3
```

Performing a Logical AND Operation to Test Two Conditions

```
// the double ampersand && is the C AND operator

if ((today == Friday) && (hour > 5))
  printf("Happy Hour");
```

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

int main(void)
{
   int day = 5;  // Sunday 0, Monday 1, Tuesday 2, Wednesday 3, Thursday 4, Friday
   int hour = 6;


   if ((day == 5) && (hour == 6))
     printf("It\'s Happy Hour");
}
```

```
It's Happy Hour
```

Performing a Logical AND Operation to Test Two Conditions

```
// the double pipe symbol || is the C OR operator

if ((today == Saturday) || (today == Sunday))
  printf("Turn on the football game");
```

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

int main(void)
{
   int day = 0;  // Sunday 0, Monday 1, Tuesday 2, Wednesday 3, Thursday 4, Friday

   if ((day == 0) || (day == 6))
     printf("It\'s time for football");
}
```

Understanding C's Logical Not Operator

```c
// The exclamation mark ! is C's Not Operator

if (! happy)
  printf("Cheer up\n");
```

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

int main(void)
{
  int happy = 0;  // 0 is sad, 1 is happy

  if (! happy)
    printf("Don\'t worry, be happy!\n");

  if (! 0)
    printf("Remember 0 is false in C\n");
}
```

```
Don't worry, be happy!
Remember 0 is false in C
```

Testing the Result of an Assignment

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

int main(void)
{
  int pet, dog = 1, cat = 0;

  if ((pet = dog) == 1)
    printf("You got a dog!\n");

  if (pet = dog)  // This can be a tricky error
    printf("Did you test for a dog or assign the value dog to pet?\n");

  if (pet == dog)
    printf("Testing the value of pet says dog!");
}
```

```
You got a dog!
Did you test for a dog or assign the value dog to pet?
Testing the value of pet says dog!
```

Revisiting Compound Statements

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

int main(void)
{
  int pet = 1;  // 1 for dog, 0 for cat

  if (pet == 1)
    {
      printf("You got a dog\n");
      printf("Dogs are great\n");
    }

  if (pet == 0)
    {
      printf("You got a cat\n");
      printf("Cats are great\n");
    }

}
```

```
You got a dog
Dogs are great
```

Delcaring Variables within Compound Statements

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

int main(void)
{
  int pet = 1;  // 1 for dogs, 0 for cat

  if (pet == 1)
    {
       int numberOfDogs = 3;

       printf("You have a dog! Actually you have %d\n", numberOfDogs);
    }
}
```

```
You have a dog! Actually you have 3
```

Using Indentation to Improve Readability

In [15]:
```
#include <stdio.h>
int main(void)
{
int pet = 1;  // 1 for dogs, 0 for cat

if (pet == 1)
{
int numberOfDogs = 3;

printf("You have a dog! Actually you have %d\n", numberOfDogs);
}
}
```

```
You have a dog! Actually you have 3
```

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

int main(void)
{
  int pet = 1;  // 1 for dogs, 0 for cat

  if (pet == 1)
    {
       int numberOfDogs = 3;

       printf("You have a dog! Actually you have %d\n", numberOfDogs);
    }
}
```

```
You have a dog! Actually you have 3
```

Be Careful Testing Floating-Point Values

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

int main(void)
{
   float taxRate = 0.65;

   if (taxRate == 0.65)
     printf("Tax rate is low %f\n", taxRate);

   printf("Tax Rate %8.10lf\n", taxRate);
}
```

```
Tax Rate 0.6499999762
```

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

int main(void)
{
   float taxRate = 0.65;

   if ((0.65- taxRate) < 0.001)
     printf("Tax rate is low %f\n", taxRate);

   printf("Tax Rate %8.10lf\n", taxRate);
}
```

```
Tax rate is low 0.650000
Tax Rate 0.6499999762
```

Understanding if-else Statements

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

int main(void)
{
  int pet = 1;  // 1 for dogs, 0 for cats

  if (pet == 1)
    printf("You have a dog!\n");
  else
    printf("You have a cat\n");
}
```

```
You have a dog!
```

Using Compound Statements with if-else

In [20]:
```c
#include <stdio.h>

int main(void)
{
  int pet = 1;  // 1 for dogs, 0 for cats

  if (pet == 1)
    {
       printf("You have a dog!\n");
       printf("Dogs are great\n");
    }
  else
    {
       printf("You have a cat\n");
       printf("Cats are great!\n");
    }
}
```

```
You have a dog!
Dogs are great
```

Be Careful of if-if-else Statements

In [21]:
```c
#include <stdio.h>

int main(void)
{
  int pet = 1;  // 1 for dogs, 0 for cats
  int numberOfDogs = 3;

  if (pet == 1)
    if (numberOfDogs > 5)
      printf("You have many dogs\n");
  else
    printf("You have a cat\n");
}
```

```
You have a cat
```

In [22]:
```c
#include <stdio.h>

int main(void)
{
  int pet = 1;  // 1 for dogs, 0 for cats
  int numberOfDogs = 3;

  if (pet == 1)
    {
      if (numberOfDogs > 5)
        printf("You have many dogs\n");
    }
  else
    printf("You have a cat\n");
}
```

Testing Multiple Conditions Using if-else if

In [23]:
```
#include <stdio.h>

int main(void)
{
  int pet = 1; // 0 for cats, 1 for dogs, 2 for horses, 3 for birds

  if (pet == 0)
    printf("Cats!\n");
  else if (pet == 1)
    printf("Dogs!\n");
  else if (pet == 2)
    printf("Horses!\n");
  else if (pet == 3)
    printf("Birds!\n");
  else
    printf("You should get a pet\n");
}
```

```
Dogs!
```

Testing Multiple Consditions Using switch

In [24]:
```c
#include <stdio.h>

int main(void)
{
  char letter = 'A';

  switch (letter) {
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U': printf("%c is a vowel!", letter);
  }
}
```

```
A is a vowel!
```

Understanding the break Statement

In [25]:
```c
#include <stdio.h>

int main(void)
{
  char letter = 'A';

  switch (letter) {
    case 'A': printf("The letter %c matches A\n", letter);
    case 'E': printf("The letter %c matches E\n", letter);
    case 'I': printf("The letter %c matches I\n", letter);
    case 'O': printf("The letter %c matches O\n", letter);
    case 'U': printf("The letter %c matches U\n", letter);
  }
}
```

```
The letter A matches A
The letter A matches E
The letter A matches I
The letter A matches O
The letter A matches U
```

In [26]:
```c
#include <stdio.h>

int main(void)
{
  char letter = 'A';

  switch (letter) {
    case 'A': printf("The letter %c matches A\n", letter);
              break;
    case 'E': printf("The letter %c matches E\n", letter);
              break;
    case 'I': printf("The letter %c matches I\n", letter);
              break;
    case 'O': printf("The letter %c matches O\n", letter);
              break;
    case 'U': printf("The letter %c matches U\n", letter);
              break;
  }
}
```

```
The letter A matches A
```

Understanding the default Statement in switch

In [27]:
```c
#include <stdio.h>

int main(void)
{
  char letter = 'A';

  switch (letter) {
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U': printf("%c is a vowel!", letter);
         break;
    default: printf("%c is a consonant!", letter);
  }
}
```

```
A is a vowel!
```

What You will Learn Next

```
Programs often must repeat a set of instructions, such as displaying a menu
until the user selects quit or looping for a set of values. Such processing is
called iterative processing.

while (fgets(buffer, sizeof(buffer), fp) != NULL)
  puts(fgets);

for (int i = 0; i < 10; i++)
  printf("%d ", i);

do {
  printf("Hello, world\n");
} while (forever);
```