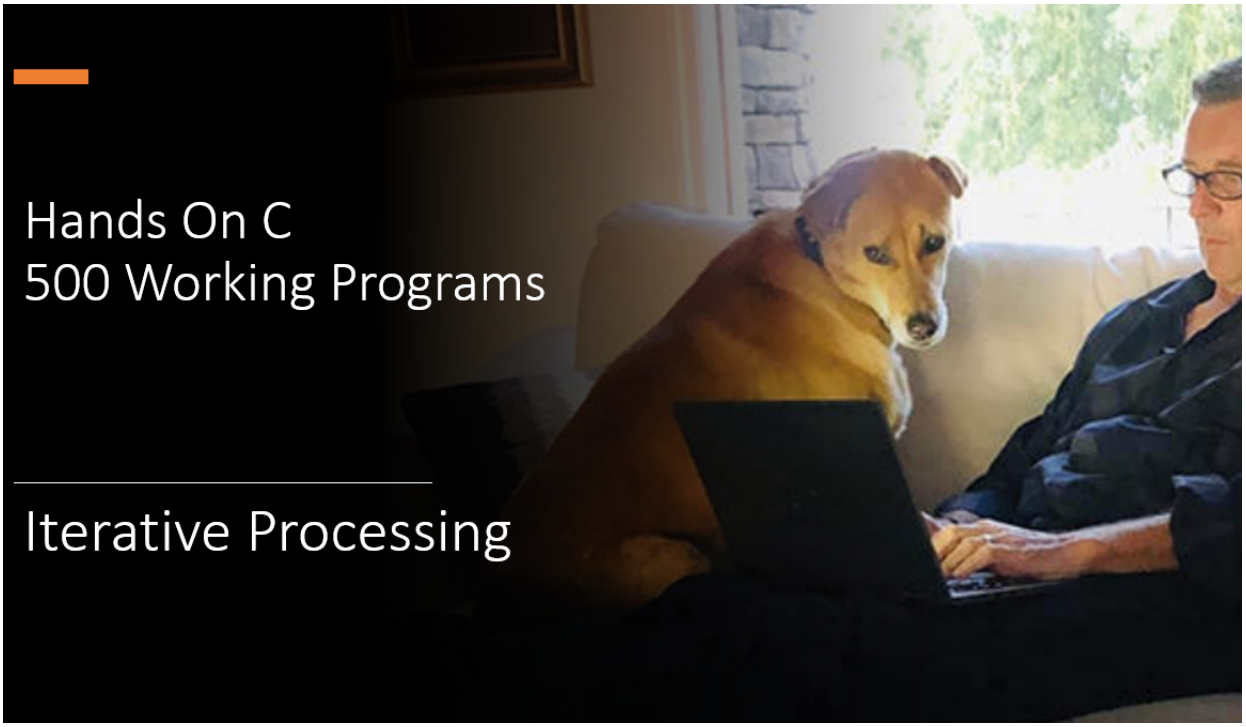

Understanding Iterative Processing

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

int main(void)
{
   int age = 1;

   while (age < 21)
     ++age;           // same as a = a + 1

   printf("Ending age %d\n", age);
}
```

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

int main(void)
{
   int age = 1;

   while (age < 21)
     printf("Age %d\n", age++);          // note postfix increment

   printf("Ending age %d\n", age);
}
```

C's Iterative Operators

```
while

for

do while
```

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

int main(void)
{
  int count = 0;

  while (count < 100)
    printf("%d ", count++);

  for (count = 0; count < 100; count++)
    printf("%d ", count);

  count = 0;

  do {
   printf("%d ", count++);

 } while (count < 100);
}
```

Performing Statements a Specific Number of Times

```
for (initialization; condition test; increment)
  statements to execute
```

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

int main(void)
{
    int count;

    for (count = 1; count <= 10; count++)
      printf("%d\n", count);
}
```

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

int main(void)
{
    int count;

    for (count = 100; count != 0; count--)
      printf("%d ", count);
}
```

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

int main(void)
{
    char letter;

    for (letter = 'A'; letter <= 'Z'; letter++)
      printf("%c", letter);
}
```

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

int main(void)
{
    int count;

    for (count = 100; count != 0; count--)
      printf("%d ", count);
}
```

In [ ]:
```c

```

Using a Compound Statement within for

```
a = b;  // simple statement

{       // compound statement within {}
  a = b;
  printf("%d %d\n", a, b);
}
```

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

int main(void)
{
  int count;

  for (count = 0; count < 10; count++)
    {
      if (count % 2 == 1)
        printf("%d is odd\n", count);
      else
        printf("%d is even\n", count);
    }
}
```

Understanding Infinite Loops

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

int main(void)
{
    int count;

    for (count = 1; count >= 0; count++)  // loop never ends
      printf("%d ", count);
}
```

Parts of the for Loop are Optional

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

int main(void)
{
  int count = 0;

  for ( ; count < 10; count++)
    printf("%d ", count);
}
```

```
0 1 2 3 4 5 6 7 8 9
```

Changing the Count Increment

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

int main(void)
{
   int count;

   for (count = 0; count <= 10; count += 2)
    printf("%d ", count);

   for (count = 0; count <= 100; count += 5)
    printf("%d ", count);
}
```

```
0 2 4 6 8 10 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
```

Understanding the Null Statement

```
; // a semicolon by itself is a Null statement
```

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

int main(void)
{
  int count;

  printf("Starting loop\n");
  for (count = 0; count < 1000; count++)
      ;
  printf("Ending loop");
}
```

```
Starting loop
Ending loop
```

Using a for Loop with a Floating-Point Value

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

int main(void)
{
  float value;

  for (value = 0.0; value <= 10.0; value += 0.5)
    printf("%f ", value);
}
```

```
0.000000 0.500000 1.000000 1.500000 2.000000 2.500000 3.000000 3.500000 4.00000
0 4.500000 5.000000 5.500000 6.000000 6.500000 7.000000 7.500000 8.000000 8.500
000 9.000000 9.500000 10.000000
```

Delcaring the Loop Variable within for

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

int main(void)
{
    for (int count = 0; count < 10; count++)
        printf("%d ", count);
}
```

```
0 1 2 3 4 5 6 7 8 9
```

Introduction to Variable Scope (Where the Variable is Known)

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

int main(void)
{
  for (int count = 0; count < 10; count++)
    printf("%d ", count);

  printf("\nEnding Value %d", count);
}
```

```
/tmp/tmpgptjx76i.c: In function ‘main’:
/tmp/tmpgptjx76i.c:8:31: error: ‘count’ undeclared (first use in this function)
   printf("\nEnding Value %d", count);
                               ^~~~~
/tmp/tmpgptjx76i.c:8:31: note: each undeclared identifier is reported only once
for each function it appears in
[C kernel] GCC exited with code 1, the executable will not be executed
```

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

int main(void)
{
  int count = 100;

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

  printf("\nEnding Value %d", count);
}
```

```
0 1 2 3 4 5 6 7 8 9
Ending Value 100
```

Using C's Comma Operator within a for Loop

```
int a, b, c;

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

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

int main(void)
{
   for (int a = 0, b = 100; a <= 10; a++, b++)
     printf("a %d b %d\n", a, b);
}
```

```
a 0 b 100
a 1 b 101
a 2 b 102
a 3 b 103
a 4 b 104
a 5 b 105
a 6 b 106
a 7 b 107
a 8 b 108
a 9 b 109
a 10 b 110
```

Using the break Statement within a for loop

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

int main(void)
{
  for (int count = 0; count <= 10; count++)
  {
      printf("%d ", count);
      if (count == 5)
          break;
  }
}
```

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

int main(void)
{
  for (int count = 0; count <= 5; count++)
      printf("%d ", count);
}
```

Using the continue Statement within for

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

int main(void)
{
  for (int count = 0; count <= 10; count++)
    {
       if ((count % 2) == 1)
         continue;
       printf("%d is even\n", count);
    }
}
```

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

int main(void)
{
  for (int count = 0; count <= 10; count++)
    {
       if ((count % 2) == 0)
         printf("%d is even\n", count);
    }
}
```

In [ ]:
```c

```

Using Indentation and Whitespace to Improve Readability

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

int main(void)
{
for (int count = 0; count <= 10; count++)
{
if ((count % 2) == 0)
printf("%d is even\n", count);
}
}
```

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

int main(void)
{
  for (int count = 0; count <= 10; count++)
    {
       if ((count % 2) == 0)
         printf("%d is even\n", count);
    }
}
```

In [ ]:
```c

```

Using the C while Loop

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

int main(void)
{
   int count = 0;

   while (count < 10)
     {
        if (count % 2)
          printf("%d is odd\n", count);

        count++;
     }
}
```

Preventing Infinite Loops Using ITEM

```
I Initialize   count = 0;
T Test         while (count < 10) {
E Execute        printf("%d ", count);
M Modify         count ++
               }
```

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

int main(void)
{
  int count = 0;             // Initialize

  while (count < 10)         // Test
    {
       printf("%d ", count); // Execute
       count++;              // Modify
    }

}
```

Looping Using the C do Statement

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

int main(void)
{
   char response;   // user's menu response

   do {
       printf("A Say Hello\n");
       printf("B Say Goodbye\n");
       printf("Q Quit\n");

       response = getchar();    // Read a letter from the keyboard

       switch (response) {
           case 'A': printf("Hello, world\n");
                 break;
           case 'B': printf("Goodbye, world\n");
                 break;
       }
   } while (response != 'Q');
}
```

Using the C goto Statement

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

int main(void)
{
  int count = 0;

  label:
    printf("%d ", count++);
    if (count <= 10)
      goto label;
}
```

What You will Learn Next

```
To improve the readability of their code and to reduce changes they must make
in the future, C programmers often define constants and macros.

#define SIZE 25

#define SQUARE(x) ((x)*(x))
```