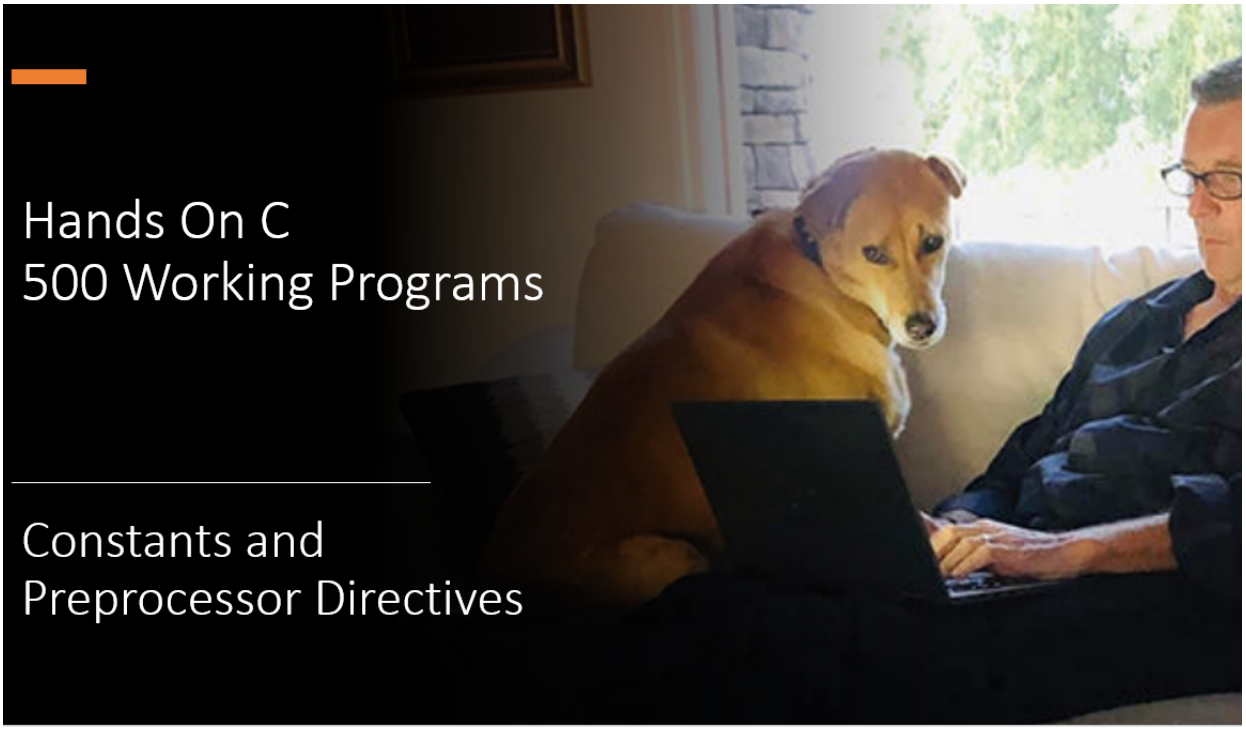

Understanding Preprocssing

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

In [1]:
```
int main(void)
{
   printf("Hello, world!");
}
```

```
/tmp/tmp5q0zaqbn.c: In function ‘main’:
/tmp/tmp5q0zaqbn.c:3:4: warning: implicit declaration of function ‘printf’ [-Wi
mplicit-function-declaration]
    printf("Hello, world!");
    ^~~~~~
/tmp/tmp5q0zaqbn.c:3:4: warning: incompatible implicit declaration of built-in
function ‘printf’
/tmp/tmp5q0zaqbn.c:3:4: note: include ‘<stdio.h>’ or provide a declaration of
‘printf’

Hello, world!
```

Defining a Constant

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

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

  if (pet == 1)
    printf("Dogs are great\n");
  else
    printf("Cats are great\n");
}
```

```
Dogs are great
```

In [6]:
```c
#include <stdio.h>
#define DOG 1       // No equal sign, no semicolon
#define CAT 0

int main(void)
{
  int pet = DOG;

  if (pet == DOG)
    printf("Dogs are great\n");
  else
    printf("Cats are great\n");
}
```

```
Dogs are great
```

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

#define BUFFER_SIZE 128   // No equal sign, no semicolon

int main(void)
{
  char title[BUFFER_SIZE] = "C Programming Complete 500 Complete Programs";

  printf("%s", title);
}
```

```
C Programming Complete 500 Complete Programs
```

```

```

```
# Using the __FILE__ Constant
```

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

int main(void)
{
   printf("This file is %s\n", __FILE__);
}
```

```
This file is /tmp/tmpbumpjxtl.c
```

```
# Using the __LINE__ Constant
```

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

int main(void)
{
  int count;

  printf("About to start loop line %d\n", __LINE__);
  for (count = 0; count < 10; count++)
    {
      if (count % 2 == 1)
        printf("%d is odd\n", count);
      else
        printf("%d is even\n", count);
    }
  printf("Ended loop line %d\n", __LINE__);
}
```

```
About to start loop line 7
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
Ended loop line 15
```

Controlling LINE and FILE

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

int main(void)
{

#line 100 "Demo.c"

  int count;

  printf("In %s About to start loop line %d\n", __FILE__, __LINE__);
  for (count = 0; count < 10; count++)
    {
      if (count % 2 == 1)
        printf("%d is odd\n", count);
      else
        printf("%d is even\n", count);
    }
  printf("Ended loop line %d\n", __LINE__);
}
```

```
In Demo.c About to start loop line 103
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
Ended loop line 111
```

Ending Compilation Using #error

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

int main(void)
{
  int count = 0;

#error Code is not complete -- still has bug

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

```
/tmp/tmpjd1n0our.c: In function ‘main’:
/tmp/tmpjd1n0our.c:7:2: error: #error Code is not complete -- still has bug
 #error Code is not complete -- still has bug
  ^~~~~
[C kernel] GCC exited with code 1, the executable will not be executed
```

Noting the Preprocessor's Date and Time

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

int main(void)
{
  printf("Date %s and Time %s compiled\n", __DATE__, __TIME__);
}
```

```
Date Jan 20 2021 and Time 20:04:30 compiled
```

Testing If a Constant is Defined

```
#ifdef CONSTANT
  // statement
#endif

#ifdef CONSTANT
  // statements
#else
  // statements
#endif
```

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

int main(void)
{
  #ifdef __STDC__
    printf("ANSI C Compliant Compiler\n");
  #else
    printf("Compiler NOT ANSI C Compliant");
  #endif
}
```

```
ANSI C Compliant Compiler
```

Testing for C Versus C++

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

int main(void)
{
  #ifdef __cplusplus
    printf("C++ Compiler\n");
  #else
    printf("C Compiler");
  #endif

}
```

```
C Compiler
```

Undefining a Constant

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

#define BIG 10

int main(void)
{
    #define BIG 100

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

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 5
6 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
```

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

#define BIG 10

int main(void)
{
    #undef BIG
    #define BIG 100

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

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 5
6 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
```

Understanding Compiler Directives (Pragmas)

In [26]:
```c
#pragma pragma_name value
```

```
/tmp/tmplssfrcv5.out: /tmp/tmpnr5ogjny.out: undefined symbol: main
[C kernel] Executable exited with code 1
```

Creating a Macro

In [33]:
```
#include <stdio.h>
#define SQUARE(x) ((x)*(x))

int main(void)
{
  for (int count = 1; count <= 10; count++)
    printf("%d squared is %d\n", count, SQUARE(count));
}
```

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
6 squared is 36
7 squared is 49
8 squared is 64
9 squared is 81
10 squared is 100
```

Be Careful with Macros

In [36]:
```
#include <stdio.h>
#define SQUARE(x) x*x

int main(void)
{
  printf("5 squared is %d\n", SQUARE(5));
  printf("3+2 squared is %d\n", SQUARE(3+2));
}
```

```
5 squared is 25
3+2 squared is 11
```

In [ ]:
```
  printf("5 squared is %d\n", SQUARE(5));

  becomes

  printf("5 squared is %d\n", 5*5);

  printf("3+2 squared is %d\n", SQUARE(3+2));

  becomes

  printf("3+2 squared is %d\n", 3+2*3+2);  // 3+6+2 => 11
```

Creating Some Sample Macros

In [44]:
```c
#include <stdio.h>

#define MIN(x, y) ((x) < (y) ? (x): (y))   // No semicolons
#define MAX(x, y) ((x) > (y) ? (x): (y))
#define SUM(x, y) ((x)+(y))
#define CUBE(x) ((x)*(x)*(x))

int main(void)
{
    printf("Minimum of %d and %d is %d\n", 5, 3, MIN(5, 3));
    printf("Maximum of %d and %d is %d\n", 5+2, 3+3, MAX(5+2, 3+3));

    printf("Cube of %d is %d\n", 9, CUBE(9));
    printf("Cube of %d is %d\n", 9+1, CUBE(9+1));

    printf("Sum of %d and %d is %d\n", 9, 7+1, SUM(9, 7+1));
}
```

```
Minimum of 5 and 3 is 3
Maximum of 7 and 6 is 7
Cube of 9 is 729
Cube of 10 is 1000
Sum of 9 and 8 is 17
```

Macros Are Typeless

In [47]:
```
#include <stdio.h>

#define CUBE(x) ((x)*(x)*(x))

int main(void)
{
    printf("Cube of %d is %d\n", 9, CUBE(9));
    printf("Cube of %f is %f\n", 7.7, CUBE(7.7));
}
```

```
Cube of 9 is 729
Cube of 7.700000 is 456.533000
```

What You will Learn Next

As you have learned, programs store data in variables when they run. A variable normally stores only one value of a specific type, such as an int or a float. Often, our programs must work with multiple values of the same type, such as a class's test scores, a list of product prices, and so on. An array is a data structure that stores multiple values of the same type.

```
int array[5] = { 10, 20, 30, 40, 50 };
```