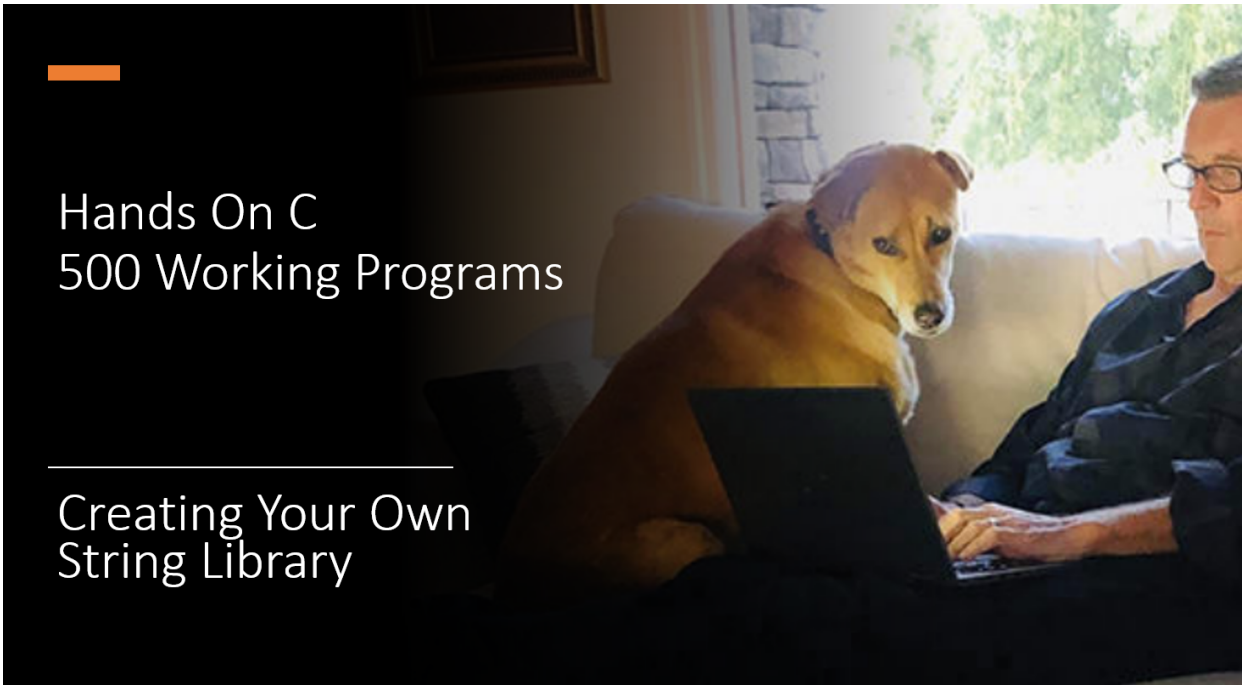

Determining a String's Length

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

int stringlength(const char *string)
{
    int i = 0;
    while (*string++)
     i++;
    return(i);
}

int main(void)
{
   char stringOne[] = "Hello";
   char stringTwo[] = "Hello, World";

   printf("%s contains %d characters\n", stringOne, stringlength(stringOne));
   printf("%s contains %d characters\n", stringTwo, stringlength(stringTwo));
}
```

```
Hello contains 5 characters
Hello, World contains 12 characters
```

Copying One String to Another

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

char *stringcopy(char *source, char *destination)
{
    char *originalDestination = destination;

    while (*destination++ = *source++)
        ;

    return(originalDestination);
}

int main(void)
{
   char firstName[64] = "Kris";
   char copy[64];

   stringcopy(firstName, copy);
   printf("%s\n", copy);
}
```

```
Kris
```

Appending One String's Contents to Another

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

char *stringappend(char *source, char *destination)
{
    char *originalDestination = destination;

    // find end of first string
    while (*destination)
        destination++;

    while (*destination++ = *source++)
        ;

    return(originalDestination);
}

int main(void)
{
  char string[64] = "Hello, ";
  char stringTwo[64] = "World";

  stringappend(stringTwo, string);
  printf("%s\n", string);
}
```

```
Hello, World
```

Appending n Characters of One String to Another

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

char *stringappendN(char *source, char *destination, int n)
{
    char *originalDestination = destination;

    int i = 0;

    // find end of first string
    while (*destination)
        destination++;

    // append up to n characters
    while((i++ < n) && (*destination++ = *source++))
        ;

    if (i > n)
        *destination = '\0';

    return(originalDestination);
}

int main(void)
{
  char string[64] = "Hello, ";
  char stringTwo[64] = "World";

  stringappendN(stringTwo, string, 3);
  printf("%s\n", string);
}
```

```
Hello, Wor
```

Determining if Two Strings are the Same

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

int stringsequal(char *s1, char *s2)
{
    while ((*s1 == *s2) && *s1)
    {
        s1++;
        s2++;
    }

    return ((*s1 == '\0') && (*s2 == '\0'));
}

int main(void)
{
   char stringOne[] = "Hello, world!";
   char stringTwo[] = "Hello, world!";
   char stringThree[] = "Hello, mundo!";

   if (stringsequal(stringOne, stringTwo))
       printf("%s and %s are the same\n", stringOne, stringTwo);

   if (stringsequal(stringThree, stringTwo))
       printf("%s and %s are the same\n", stringThree, stringTwo);
}
```

```
Hello, world! and Hello, world! are the same
```

Converting Strings to Upper and Lowercase

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

char *stringupper(char *string)
{
    char *original_string = string;
    printf("%s\n", string);
    while (*string)
    {
        if ((*string >= 'a') && (*string <= 'z'))
            *string = *string - ('a' - 'A');

        string++;
    }

    return(original_string);
}

char *stringlower(char *string)
{
    char *original_string = string;

    while (*string)
    {
        if ((*string >= 'A') && (*string <= 'Z'))
            *string = *string + ('a' - 'A');

        string++;
    }

    return(original_string);
}

int main(void)
{
   char string[64] = "Hello, world!";

   stringupper(string);
   printf("%s\n", string);

   stringlower(string);
   printf("%s\n", string);
}
```

```
Hello, world!
HELLO, WORLD!
hello, world!
```

Returning a Pointer to the First Occurrence of a Character within a String

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

char *stringcharacter(char *string, const char letter)
{
   while ((*string != letter) && *string)
       string++;

   return(*string ? string : 0);
}

int main(void)
{
   char string[64] = "Hello, world!";

   char *ptr = stringcharacter(string, 'l');

   if (ptr)
     printf("Found %c in %s at offset %ld\n", 'l', string, ptr-string);
   else
     printf("Did not find %c in %s\n", 'l', string);

   ptr = stringcharacter(string, 'm');

   if (ptr)
     printf("Found %c in %s at offset %ld\n", 'm', string, ptr-string);
   else
     printf("Did not find %c in %s\n", 'm', string);

}
```

```
Found l in Hello, world! at offset 2
Did not find m in Hello, world!
```

Finding the Rightmost Occurrence of a Character within a String

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

char *stringcharacterR(char *string, const char letter)
{
   char *ptr = NULL;

   while (*string)
   {
      if (*string == letter)
          ptr = string;

       string++;
   }

   return(ptr);
}

int main(void)
{
   char string[64] = "Hello, world!";

   char *ptr = stringcharacterR(string, 'l');

   if (ptr)
     printf("Found %c in %s at offset %ld\n", 'l', string, ptr-string);
   else
     printf("Did not find %c in %s\n", 'l', string);

   ptr = stringcharacterR(string, 'm');

   if (ptr)
     printf("Found %c in %s at offset %ld\n", 'm', string, ptr-string);
   else
     printf("Did not find %c in %s\n", 'm', string);

}
```

```
Found l in Hello, world! at offset 10
Did not find m in Hello, world!
```

Counting the Number of Occurrence of a Character within a String

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

int charactercount(const char *string, const char letter)
{
  int count = 0;

  while (*string)
    if (*string++ == letter)
      count++;

  return(count);
}

int main(void)
{
  char string[] = "Hello, world!";
  char letter = 'l';
  printf("The letter %c occurs %d times in %s\n", letter, charactercount(string,

  letter = 'm';
  printf("The letter %c occurs %d times in %s\n", letter, charactercount(string,
}
```

```
The letter l occurs 3 times in Hello, world!
The letter m occurs 0 times in Hello, world!
```

Comparing the Contents of Two Strings

```
0 if equal
-1 if string1 < string2
```

```
1 if string1 > string2
```

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

int stringcompare(const char *s1, const char *s2)
{
  while ((*s1 == *s2) && (*s1))
    {
      s1++;
      s2++;
    }

  if ((*s1 == *s2) && (! *s1))
    return(0);   // same
  else if ((*s1) && (!*s2))
    return(1);   // s1 longer
  else if ((*s2) && (!*s1))
    return(-1);  // s2 longer
  else if ((*s1) != (*s2))
    return((*s1 > *s2) ? -1: 1); // different
}

int main(void)
{
  char stringOne[] = "Hello, World";
  char stringTwo[] = "Hello, World";
  char stringThree[] = "Hello, World!";
  char stringFour[] = "hello, world";

  printf("Comparing %s and %s returns %d\n", stringOne, stringTwo, stringcompare(
  printf("Comparing %s and %s returns %d\n", stringTwo, stringThree, stringcompar
  printf("Comparing %s and %s returns %d\n", stringOne, stringFour, stringcompare
}
```

```
Comparing Hello, World and Hello, World returns 0
Comparing Hello, World and Hello, World! returns -1
Comparing Hello, World and hello, world returns 1
```

Finding a Substring within a String

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

char *substring(char *string, const char *substring)
{
   int i, j, k;

   for (i = 0; string[i]; ++i)
     for (j = i, k = 0; string[j] == substring[k]; j++, k++)
         if (! substring[k+1])
             return(string+i);

   return(NULL);
}

int main(void)
{
   char string[64] = "Hello, World!";
   char *ptr;

   ptr = substring(string, "World");
   if (ptr)
     printf("Found World in %s at offset %ld\n", string, ptr-string);

   ptr = substring(string, "!");
   if (ptr)
     printf("Found ! in %s at offset %ld\n", string, ptr-string);

   ptr = substring(string, "mundo");
   if (ptr)
     printf("Found mundo in %s at offset %ld", string, ptr-string);
 }
```

```
Found World in Hello, World! at offset 7
Found ! in Hello, World! at offset 12
```

Removing a Substring from a String

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

char *removesubstring(char *string, const char *substring)
{
   int i, j, k, location = -1;
   char *original_string = string;
   int oldLength = strlen(string);

   for (i = 0; string[i] && location == -1; ++i)
     for (j = i, k = 0; string[j] == substring[k]; j++, k++)
         if (! substring[k+1])
             location = i;

   if (location != -1)
   {
     for (k = 0; substring[k]; k++)
       ;

     for (j = location, i = location + k; string[i]; j++, i++)
         string[j] = string[i];
     string[j] = '\0';
   }

   return(string);
}

int main(void)
{
   char string[64] = "Hello, World!";
   char stringTwo[64] = "Hello, World!";

   printf("Original %s without %s yields %s\n", "Hello, World!", "World!", remove
   printf("Original %s without %s yields %s\n", "Hello, World!", "WORLD", removes
   printf("Original %s without %s yields %s\n", "Hello, World!", "!", removesubst
}
```

```
Original Hello, World! without World! yields Hello,
Original Hello, World! without WORLD yields Hello, World!
Original Hello, World! without ! yields Hello, World
```

Removing Every Occurrence of a Substring from a String

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

char *removesubstringAll(char *string, const char *substring)
{
   int i, j, k, location = -1;
   char *original_string = string;
   int oldLength = strlen(string);

   for (i = 0; string[i] && location == -1; ++i)
     for (j = i, k = 0; string[j] == substring[k]; j++, k++)
         if (! substring[k+1])
             location = i;

   if (location != -1)
   {
     for (k = 0; substring[k]; k++)
       ;

     for (j = location, i = location + k; string[i]; j++, i++)
         string[j] = string[i];
     string[j] = '\0';
   }

   if (location != -1)
     return(removesubstringAll(string, substring));
   else
     return(string);
}

int main(void)
{
   char string[64] = "Hello, World!";
   char stringTwo[64] = "Hello, World!";
   char stringThree[64] = "Hello, World!";

   printf("Original %s without %s yields %s\n", "Hello, World!", "World!", remove
   printf("Original %s without %s yields %s\n", "Hello, World!", "l", removesubst
   printf("Original %s without %s yields %s\n", "Hello, World!", "!", removesubst
}
```

```
Original Hello, World! without World! yields Hello,
Original Hello, World! without l yields Heo, Word!
Original Hello, World! without ! yields Hello, World
```

Replacing One Substring with Another

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

char *replacesubstring(char *string, const char *substring, char *new)
{
   int i, j, k, location = -1;
   char *original_string = string;
   int old_length = strlen(substring);
   char temp[256];

   for (i = 0; string[i] && location == -1; ++i)
     for (j = i, k = 0; string[j] == substring[k]; j++, k++)
         if (! substring[k+1])
             location = i;

   if (location != -1)
   {
     for (j = 0; j < location; j++)
       temp[j] = string[j];

     for (i = 0; new[i]; j++, i++)
       temp[j] = new[i];

     for (k = location + old_length; string[k]; j++, k++)
       temp[j] = string[k];

     temp[j] = '\0';

     for (i = 0; string[i] = temp[i]; i++)
         ;
   }

  return(string);
}

int main(void)
{
   char string[64] = "Hello, World!";
   char stringTwo[64] = "Hello, World!";
   char stringThree[64] = "Hello, World!";

   printf("Original %s replacing %s with %s yields %s\n", "Hello, World!", "World
   printf("Original %s replacing %s with %s yields %s\n", "Hello, World!", "Mundo
   printf("Original %s replacing %s with %s yields %s\n", "Hello, World!", "World
   printf("Original %s replacing %s with %s yields %s\n", "Hello, World!", "Hello
}
```

```
Original Hello, World! replacing World with WORLD yields Hello, WORLD!
Original Hello, World! replacing Mundo with WORLD yields Hello, World!
Original Hello, World! replacing World with WORLD yields Hello, WORLD!
Original Hello, World! replacing Hello, with  yields World!
```

Using C's strxfrm Function to Copy n Characters from One String to Another

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

int main(void)
{
    char string[64] = "Hello";
    char secondString[64] = "World Peace";

    strxfrm(string, secondString, 5);
    printf("%s\n", string);
}
```

```
World
```

Determining if Two Strings are the Same

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

int main(void)
{
  char stringOne[] = "Hello, World";
  char stringTwo[] = "Hello, World";
  char stringThree[] = "Hello, World!";

  int i;

  for (i = 0; stringOne[i] == stringTwo[i] && stringOne[i]; i++)
    ;
  if (stringOne[i] == '\0' && stringTwo[i] == '\0')
    printf("%s and %s are the same\n", stringOne, stringTwo);

  for (i = 0; stringTwo[i] == stringThree[i] && stringTwo[i]; i++)
    ;
  if (stringTwo[i] == '\0' && stringThree[i] == '\0')
    printf("%s and %s are the same\n", stringTwo, stringThree);
}
```

```
Hello, World and Hello, World are the same
```

What You will Learn Next

C programs use variables to store data as they execute. A variable normally stores one value of a specific type. You have learned that using arrays, a variable can store multiple values of the same type.Using a structure in C, you can create a variable that can hold multiple values of different types.

```
struct Employee {
  char name[256];
  int age;
  float salary;
} worker;

worker.age = 50;
worker.salary = 100000;
```