Character Strings - searching, tokenizing, and Analyzing Strings

- Let's discuss some more string functions
- Searching a string
    - ➔ The string.h header file declares several string-searching functions for finding a single character or a substring.
        - ★ strchr() and strstr()
- Tokenizing a string
    - ➔ A token is a sequence of characters within a string that is bounded by delimiter (space, comma, period, ect)
    - ➔ Breaking a sentence into words is called tokenizing .
        - ★ strtok()
- Analyzing strings
    - ➔ Islower(), Isupper(), Isalpha(), isdigit() ect..

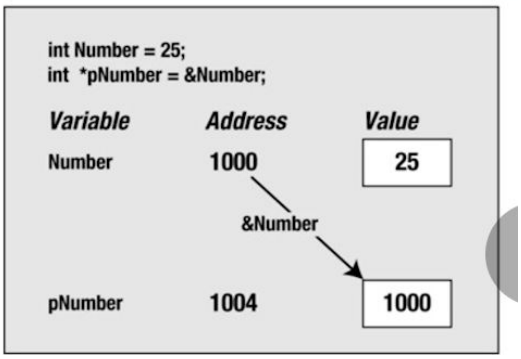
Concept of a pointer

- We are going to discuss in detail the concept of a pointer in an upcoming section.
    - ➔ However, in order to understand some of these string functions, i want to give you a quick peek on this concept
- C provides a remarkably useful type called a pointer in an upcoming section.
    - ➔ A variable that stores an address.
    - ➔ Its value is the address of another location in memory that can contain a value
    - ➔ We have used addresses in the past with the scanf() function

```
int Number = 25;
int *pNumber = &Number;
```

- Above, we declared a variable , Number , with the value 25
- We declared a pointer, pNumber, which contains the address of Number
    - ➔ Asterisk used in declaring a pointer
- To get the value of the variable pNumber , you can use the asterisk to dereference the pointer .
    - ➔ *pNumber =25
    - ➔ * is the dereference operator, and its effect is to access the data stored at the address specified by a pointer.

pointer (cont'd)



(taken from Beginning C, Horton)

- The value of &Number is the address where Number is located.
- This value is used to initialize pNumber in the second statement.
- Many of the string functions return pointers
    - ➔ This is why I wanted to briefly mention them.
    - ➔ Do not worry if this concept does not sink in right now, we are going to cover points in a ton of detail in an upcoming section.

Searching a string for a character

- The strchr () function searches a given string for a specified character
    - ➔ First argument to the function is the string to be searched (which will be the address of a char array)
    - ➔ Second argument is the character that you are looking for.
- The function will search the string starting at the beginning and return a pointer to the first position in the string where the character is found.
- The address of this position in memory.
- Is of type char* described as the "pointer to char".
- To store the value that's returned, you must create a variable that can store the address of a character .
- If the character is not found , the function returns a special value NULL
    - ➔ NULL is the equivalent of) for a point and represents a pointer that does not point to anything.

```
#include <stdio.h>
#include <string.h>

int main ()

{
    char str[] = "The quick brown fox"; //the string to be searched
    char ch = 'q';                      // the character we are looking for
    char *pGot_char = NULL;             //Pointer initialized to NULL

    pGot_char = strchr(str, ch);        //Store address where ch is found
```

```
}
```

- first argument to strchr() is the address of the first location to be searched.
- - second argument is the character that is sought (ch, which is of type char)
- -expects its second argument to be of type int , so the compiler will convert the value of ch to this type
- -could just as well define ch as type int (int ch ='q';)
- -pGot_char will point to the value ("quick brown fox")

Searching for a substring

- The strstr() function is probably the most useful of all the search functions
  - ➔ Searches one string for the first occurence of a substring
  - ➔ Returns a pointer to the position in the first string where the substring is found.
  - ➔ If no match, returns NULL
- The first argument is the string that is to be searched
- The second argument is the substring you're looking for.

```
#include <string.h>
#include <stdio.h>

int main ()
{
    char text [] = "Every dog has his day";
    char word [] = "dog";
    char *pFound = NULL;
    pFound = strstr(text , word);

    printf("%s", pFound);

}
```

- Searches text for the first occurence of the string stored in word
  - ➔ The string "dog" appears starting at the seventh character in text
  - ➔ pFound will be set to the address text + 6 ("dog has his day")
  - ➔ Searches is cases sensitive, "Dog" will not found.

Tokenizing a string

- A token is a sequence of characters within a string that is bound by a delimiter
- A delimiter can be anything , but , should be unique to the string
  - ➔ Spaces , commas, and a period are good examples.
- Breaking a sentence into words is called tokenizing.
- The Strtok () function is used for tokenizing a string.
- It requires two arguments .
  - ➔ String to be tokenized.
  - ➔ A string containing all possible delimiter characters .

Analyzing strings

Function	Tests for
`islower()`	Lowercase letter
`isupper()`	Uppercase letter
`isalpha()`	Uppercase or lowercase letter
`isalnum()`	Uppercase or lowercase letter or a digit
`iscntrl()`	Control character
`isprint()`	Any printing character including space
`isgraph()`	Any printing character except space
`isdigit()`	Decimal digit (`'0'` to `'9'`)
`isxdigit()`	Hexadecimal digit (`'0'` to `'9'`, `'A'` to `'F'`, `'a'` to `'f'`)
`isblank()`	Standard blank characters (space, `'\t'`)
`isspace()`	Whitespace character (space, `'\n'`, `'\t'`, `'\v'`, `'\r'`, `'\f'`)
`ispunct()`	Printing character for which `isspace()` and `isalnum()` return `false`

- The argument to each of these functions is the character to be tested
- All these functions return a non zero value of type int if the character is within the set that's being tested for
- These return values convert to true and false, respectively, so you can use them as Boolean values.