Characters and Strings

- Character-handling library `ctype.h`
- String-conversion functions `stdlib.h`
- String and memory-processing functions `string.h`
- Fundamentals of strings and characters
- Standard input/output library functions
- String manipulation functions
- Comparison and search functions

Fundamentals of Characters and Strings

- Techniques used to develop editors, word processors, page-layout software, computerized typesetting systems and of the text-processing software
- Text manipulations with formatted input/output
- Character Constants have an `int` value in ASCII
- String is a series of characters in double quotes
- Terms used are **strings literals** or **string constants**
- Strings end with the **null character** '\0'

Fundamentals of Characters and Strings

- Strings are access via a **pointer**
- Points to the first element of the character array
- Character array `char *` can initialize a string

```
char color[] = "blue";
const char *colorPtr = "blue";
```

Fundamentals of Strings

- Previous definition could have also been

```
char[] = {'b','l','u','e','\0'};
```

- Input a string to the program

```
char word[20];
scanf("%s", word);
```

- String must always have '\0' or an error will occur
- Printing a string will continue until '\0' is reached

Fundamentals of Strings

- Using the conversion specifier can help avoid problems

```
scanf("%19s", word)
```

- Ensures that `scanf`` reads a maximum of 19 characters, saving the last character for '\0'
- Reason for using a field width to read in a char array
- For reading input lines of arbitrary length, there is a nonstandard function `readline`, usually included in stdio.h.

Character Handling Library `ctype.h`

Prototype	Function Description
`int isblank(int c);`	Returns a true value if c is a blank character. (' ')
`int isdigit(int c);`	Returns a true value if c is a digit.
`int isalpha(int c);`	Returns a true value if c is a letter.
`int isalnum(int c);`	Returns a true value if c is a digit or a letter.
`int isxdigit(int c);`	Returns a true value if c is a hexadecimal digit character.
`int islower(int c);`	Returns a true value if c is a lowercase letter.
`int isupper(int c);`	Returns a true value if c is an uppercase letter.
`int tolower(int c);`	Returns c as a lowercase letter.
`int toupper(int c);`	Returns c as an uppercase letter.

```c
printf("isdigit function\n%s%s\n%s%s\n\n",
isdigit('8')  ? "8 is" : "8 is not", " a digit",
isdigit('#')  ? "# is" : "# is not", " a digit");

printf("isalpha function\n%s%s\n%s%s\n\n",
isalpha('A')  ? "A is" : "A is not", " a letter",
isalpha('b')  ? "b is" : "b is not", " a letter",
isalpha('&')  ? "& is" : "& is not", " a letter",
isalpha('4')  ? "4 is" : "4 is not", " a letter",);

printf("isalnum function\n%s%s\n%s%s\n\n",
isalnum('A')  ? "A is" : "A is not", " a digit or letter",
isalnum('8')  ? "8 is" : "8 is not", " a digit or letter",
isalnum('#')  ? "# is" : "# is not", " a digit or letter");
```

```c
printf("isxdigit function\n%s%s\n%s%s\n\n",
isxdigit('F')  ? "F is" : "F is not", " a hexadecimal",
isxdigit('7')  ? "7 is" : "7 is not", " a hexadecimal",
isxdigit('j')  ? "j is" : "j is not", " a hexadecimal");

printf("islower function\n%s%s\n%s%s\n\n",
islower('p')  ? "p is" : "p is not", " a lowercase letter",
islower('P')  ? "P is" : "P is not", " a lowercase letter",
islower('5')  ? "5 is" : "5 is not", " a lowercase letter");

printf("isupper function\n%s%s\n%s%s\n\n",
isupper('D')  ? "D is" : "D is not", " an uppercase letter",
isupper('d')  ? "d is" : "d is not", " an uppercase letter",
isupper('$')  ? "$ is" : "$ is not", " an uppercase letter");

printf("%s%cs\n%s%c\n%s%c\n\n", "u convert to
uppercase is", toupper('u'),
"7 convert to uppercase is" toupper('7'),
"S convert to lowercase is" tolower('S'),
"2 convert to lowercase is" tolower('2'));
```

Prototype	Function Description
`int isspace(int c);`	Returns a true value if c is a whitespace character : space (' '), form feed ('\f'), newline ('\n'), carriage return ('\r') , horizontal tab ('\t') or vertical tab ('\v').
`int iscntrl(int c);`	Returns a true value if c is a control character : horizontal tab ('\t'), vertical tab ('\v'), form feed ('\f'), alert ('\a'), backspace ('\b'), carriage return ('\r') or newline ('\n').
`int ispunct(int c);`	Returns a true value if c is a printing character other than a space, a digit, or a letter — such as $, #, (,), [,], {, }, ;, :, or %.
`int isprint(int c);`	Returns a true value if c is a character that is visible on the screen, including a space.
`int isgraph(int c);`	Returns a true value if c is a character that is visible on the screen, other than a space.

```
printf("isspace function\nNewline %s%s\n
  Horizontal Tab %s%s\n%s%s\n\n",
isspace('\n')  ? "is" : "is not", " whitespace",
isspace('\t')  ? "is" : "is not", " whitespace",
isspace('#')  ? "% is" : "% is not", " whitespace");

printf("iscntrl function\nNewline %s%s\n%s%s\n\n",
iscntrl('\n')  ? "is" : "is not", " a control character",
iscntrl('$')  ? "$ is" : "$ is not", " a control character");

printf("ispunct function\n%s%s\n%s%s\n\n",
ispunct(';')  ? "; is" : "; is not", " a punctuation character",
ispunct('Y')  ? "Y is" : "Y is not", " a punctuation character",
ispunct('#')  ? "# is" : "# is not", " a punctuation character");

printf("isprint function\n%s%s\nAlert %s%s\n\n",
isprint('$')  ? "$ is" : "$ is not", " a print character",
isprint('\a')  ? "is" : "is not", " a print character");

printf("isgraph function\n%s%s\n%s%s\n\n",
isgraph('Q')  ? "Q is" : "Q is not", " a graph character",
isgraph(' ')  ? "  is" : "  is not", " a graph character");
```

Character Handling Library `stdlib.h`

Prototypes and Function Descriptions
`double strtod(const char *nPtr, char **endPtr);`
Converts the string nPtr to double.
`double strtol(const char *nPtr, char **endPtr, int base);`
Converts the string nPtr to long.
`unsigned long strtoul(const char *nPtr, char **endPtr, int base);`
Converts the string nPtr to unsigned long.

- `strtoll` and `strtoull` for strings to long long int and unsigned long long int

Arguments for the functions

- `char *` is a string to be converted
- `char **` is a pointer to the string
    - assigned the remainder of string
    - NULL causes the remainder for string ignored
- `int base` is the value base to use
    - 0 means octal, decimal, or hexadecimal
    - 2 to 36 represents the base selected
- Function returns 0 if unable to convert any portion of its first argument

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
  const char *string1 = "-1234567abc";
  char *remainder1Ptr;
  long x = strtol(string1, &remainder1Ptr, 10);

  const char *string2 = "45670988 xyz";
  char *remainder2Ptr;
  unsigned long int y = strtol(string2, &remainder2Ptr, 10);

  printf("String : %s, %ld, %s", string1, x, remainder1Ptr);
  printf("String : %s, %ld, %s", string2, x, remainder2Ptr);

  return 0;
}
```