Week 6- Lecture 1,2

Characters and Strings

Edited by: Dr. Saeid Pourroostaei Ardakani

Autumn 2021

Overview

- Character-handling library
- String-conversion functions
- Standard input and output functions
- Search string

Character-Handling Library

- #include <ctype.h>

Prototype	Function description
int isblank(int c);	Returns a true value if c is a *blank character* that separates words in a line of text and 0 (false) otherwise. [*Note:* This function is not available in Microsoft Visual C++.]
int isdigit(int c);	Returns a true value if c is a *digit* and 0 (false) otherwise.
int isalpha(int c);	Returns a true value if c is a *letter* and 0 (false) otherwise.
int isalnum(int c);	Returns a true value if c is a *digit* or a *letter* and 0 (false) otherwise.
int isxdigit(int c);	Returns a true value if c is a *hexadecimal digit character* and 0 (false) otherwise. (See Appendix C for a detailed explanation of binary numbers, octal numbers, decimal numbers and hexadecimal numbers.)
int islower(int c);	Returns a true value if c is a *lowercase letter* and 0 (false) otherwise.
int isupper(int c);	Returns a true value if c is an *uppercase letter* and 0 (false) otherwise.
int tolower(int c);	If c is an *uppercase letter*, tolower returns c as a *lowercase letter*. Otherwise, tolower returns the argument unchanged.
int toupper(int c);	If c is a *lowercase letter*, toupper returns c as an *uppercase letter*. Otherwise, toupper returns the argument unchanged.
int isspace(int c);	Returns a true value if c is a *whitespace character*—newline ('\n'), space (' '), form feed ('\f'), carriage return ('\r'), horizontal tab ('\t') or vertical tab ('\v')—and 0 (false) otherwise.
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'), newline ('\n') and others—and 0 (false) otherwise.
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 %—and returns 0 otherwise.
int isprint(int c);	Returns a true value if c is a *printing character* (i.e., a character that's visible on the screen) *including a space* and returns 0 (false) otherwise.
int isgraph(int c);	Returns a true value if c is a *printing character other than a space* and returns 0 (false) otherwise.

Source: Deitel and Deiltel (2016). C How to Program with an Introduction to C++ (8th Ed.). Pearson.

Example

```
3    #include <stdio.h>
4    #include <ctype.h>
5
6    int main(void)
7    {
8        printf("%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
9            isdigit('8')   ? "8 is a " : "8 is not a ", "digit",
10           isdigit('#')   ? "# is a " : "# is not a ", "digit");
11
12       printf("%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
13           "According to isalpha:",
14           isalpha('A')   ? "A is a " : "A is not a ", "letter",
15           isalpha('b')   ? "b is a " : "b is not a ", "letter",
16           isalpha('&')   ? "& is a " : "& is not a ", "letter",
17           isalpha('4')   ? "4 is a " : "4 is not a ", "letter");
18
```

Source: Dola saha, C programming for engineer, 2017.

Example (2)

```
19      printf("%s\n%s%s\n%s%s\n%s%s\n\n",
20          "According to isalnum:",
21          isalnum('A')  ? "A is a " : "A is not a ",
22          "digit or a letter",
23          isalnum('8')  ? "8 is a " : "8 is not a ",
24          "digit or a letter",
25          isalnum('#')  ? "# is a " : "# is not a ",
26          "digit or a letter");
27
28      printf("%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
29          "According to isxdigit:",
30          isxdigit('F')  ? "F is a " : "F is not a ",
31          "hexadecimal digit",
32          isxdigit('J')  ? "J is a " : "J is not a ",
33          "hexadecimal digit",
34          isxdigit('7')  ? "7 is a " : "7 is not a ",
35          "hexadecimal digit",
36          isxdigit('$')  ? "$ is a " : "$ is not a ",
37          "hexadecimal digit",
38          isxdigit('f')  ? "f is a " : "f is not a ",
39          "hexadecimal digit");
40  }
```

University of Nottingham
UK | CHINA | MALAYSIA

Example- Output

```
According to isdigit:
8 is a digit
# is not a digit

According to isalpha:
A is a letter
b is a letter
& is not a letter
4 is not a letter

According to isalnum:
A is a digit or a letter
8 is a digit or a letter
# is not a digit or a letter

According to isxdigit:
F is a hexadecimal digit
J is not a hexadecimal digit
7 is a hexadecimal digit
$ is not a hexadecimal digit
f is a hexadecimal digit
```

Source: Dola saha, C programming for engineer, 2017.

String-Conversion Functions

- #include <stdlib.h>

Strtod(): converts string to double

Strtol(): converts string to long

Strtoul(): converts string to unsigned long

Example: strtod

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

int main () {
  char str[30] = "20.30300 This is test";
  char *ptr;
  double ret;

  ret = strtod(str, &ptr);
  printf("The number(double) is %f\n", ret);
  printf("String part is |%s|", ptr);

  return(0);
}
```

Example: strtod (1)

The string must begins with a valid floating point number.

- The pointer receives the memory address of the character after floating point value.

- On error, point to the beginning of the string.

```
C:\Users\z2017233\Desktop>char_str
double value is 51.200000, and the string is % are admitted
double value is 41.500000, and the string is
double value is 0.000000, and the string is My number is 1.23 not 4.56
double value is 10.200000, and the string is

C:\Users\z2017233\Desktop>
```

```
23   #include <stdio.h>
24   #include <stdlib.h> // strtod
25
26   // Instruction: try to use strtol and strtoul
27
28   int main()
29   {
30       const char *str = "51.2% are admitted";
31       const char *str2 = "41.5";
32       const char *str3 = "My number is 1.23 not 4.56";
33       char arr[10] = "10.2";
34
35       char *sPtr;
36
37       double d = 0.0;
38       d = strtod(str, &sPtr);
39       printf("double value is %f, and the string is %s\n", d, sPtr);
40
41       d = strtod(str2, &sPtr);
42       printf("double value is %f, and the string is %s\n", d, sPtr);
43
44       d = strtod(str3, &sPtr);
45       printf("double value is %f, and the string is %s\n", d, sPtr);
46
47       d = strtod(arr, &sPtr);
48       printf("double value is %f, and the string is %s\n", d, sPtr);
49
50       return 0;
51   }
```

Nottingham
UK | CHINA | MALAYSIA

Example: atof, atol

```c
const char *str = "51.2% are admitted";
const char *str2 = "41.5";
const char *str3 = "My number is 1.23 not 4.56";
char arr[10] = "10.2";
```

- Converts string to double /long interger.

```c
77    float f = 0.0;
78    f = atof("51.2");
79    printf("float value is %f\n", f);
80
81    f = atof(str);
82    printf("float value is %f\n", f);
83
84    f = atof(str2);
85    printf("float value is %f\n", f);
86
87    f = atof(str3);
88    printf("float value is %f\n", f);
89
90    f = atof("1.23");
91    printf("float value is %f\n", f);
```

Float value is: 51.200000
Float value is: 51.200000
Float value is: 41.500000
Float value is: 0.000000
Float value is: 1.2300000

Overview

- Character-handling library
- String-conversion functions
- **Standard input and output functions**
- Search string

Standard Input/Output Functions

- #include <stdio.h>

Function prototype	Function description
int getchar(void);	Inputs the next character from the standard input and returns it as an integer.
char *fgets(char *s, int n, FILE *stream);	
	Inputs characters from the specified stream into the array s until a *newline* or *end-of-file* character is encountered, or until n - 1 bytes are read. In this chapter, we specify the stream as stdin—the *standard input stream*, which is typically used to read characters from the keyboard. A *terminating null character* is appended to the array. Returns the string that was read into s. If a newline is encountered, it's included in the string stored in s.
int putchar(int c);	Prints the character stored in c and returns it as an integer.
int puts(const char *s);	Prints the string s followed by a *newline* character. Returns a non-zero integer if successful, or EOF if an error occurs.
int sprintf(char *s, const char *format, ...);	
	Equivalent to printf, except the output is stored in the array s instead of printed on the screen. Returns the number of characters written to S, or EOF if an error occurs. [*Note:* We mention the more secure related functions in the Secure C Programming section of this chapter.]
int sscanf(char *s, const char *format, ...);	
	Equivalent to scanf, except the input is read from the array s rather than from the keyboard. Returns the number of items successfully read by the function, or EOF if an error occurs. [*Note:* We mention the more secure related functions in the Secure C Programming section of this chapter.]

Source: Deitel and Deiltel (2016). C How to Program with an Introduction to C++ (8th Ed.). Pearson.

University of Nottingham
UK | CHINA | MALAYSIA

puts, sprintf

- puts add '\n' automatically.

- sprintf writes to a string e.g. array, instead of screen i.e. printf.

- This example prints 3 rows of Hello World!

```
86     // different functions for output
87     printf("%s", "Hello World!\n");
88     puts("Hello World!");
89     // note there is no \n in puts, but it gets added automatically
90
91     char arr[15] = {'\0'};
92     sprintf(arr, "Hello World!\n");
93     printf("%s", arr);
94     // note that without printf, the string is stored only in the array but
95     // not display
```

sscanf

- sscanf reads from a string e.g. array, instead of keyboard input i.e. scanf.

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

int main () {
  int day, year;
  char weekday[20], month[20];
  char dtm[100] = "Friday October 29 2021";

  sscanf( dtm, "%s %s %d  %d", weekday, month, &day, &year );

  printf("%s %d, %d = %s\n", month, day, year, weekday );

  return(0);
}
```

Output:
October 29, 2021 = Friday

fgets, putchar

- fgets takes specified number of characters and put it in the array.

```
116        fgets(arr, 15, stdin);
117        printf("%s\n", arr);
118        // note that without printf, the string is stored only in the array
119        // when fgets is used
120
121        putchar('a');
```

- putchar displays the character.

Overview

- Character-handling library
- String-conversion functions
- Standard input and output functions
- **Search string**

Basic String Functions: strcpy

strcpy : dest <- src

```c
#include <stdio.h>
#include <string.h>
int main()
{
    char src[] = "geeksforgeeks";

    char dest[14];

    // copying src into dest.
    strcpy(dest, src);
    printf("Copied string: %s\n", dest);

    return 0;
}
```

Output:

Copied string: geeksforgeeks

University of Nottingham
UK | CHINA | MALAYSIA

Basic String Functions: strncpy

Strncpy: dest <- src for n character.

```c
#include <stdio.h>
#include <string.h>
int main()
{
  char src[] = "geeksforgeeks";

  char dest[8];

  strncpy(dest, src, 8);

  int len = strlen(dest);

  printf("Copied string: %s\n", dest);
  printf("Length of destination string: %d\n", len);

  return 0;
}
```

Output:

Copied string: geeksfor
Length of destination string: 8

strcpy and strncpy: risks

- The strcpy() function does not specify the size of the destination array, so buffer overrun is often a risk.
  - Using strcpy() function to copy a large character array into smaller one is dangerous.
  - If destination string is not large enough to store the source string then the behavior of strcpy() is unspecified or undefined.
- The strncpy() function is similar to strcpy() function
  - If there is no NULL character among the first n character of src, the string placed in dest will not be NULL-terminated.
  - If the length of src is less than n, strncpy() writes additional NULL character to dest to ensure that a total of n character are written.

Basic String Functions: strcat

- strcat() function joins two strings.

```c
#include <stdio.h>
#include <string.h>
int main() {
    char str1[100] = "This is ", str2[] =
"programiz.com";

    // concatenates str1 and str2
    // the resultant string is stored in str1.
    strcat(str1, str2);

    puts(str1);
    puts(str2);

    return 0;
}
```

Output:

This is programiz.com
programiz.com

Basic String Functions: strncat

- strncat() function joins two strings for n char.

```c
#include <stdio.h>
#include <string.h>
int main()
{

char src[50] = "efghijkl";
  char dest[50]= "abcd";

strncat(dest, src, 5);

  // Prints the string
  printf("Source string : %s\n", src);
  printf("Destination string : %s", dest);

  return 0;
}
```

Output:

Source string : efghijkl
Destination string : abcdefghi

Basic String Functions: strcmp/strncmp

- Compares two strings (n char for strncmp)

- Return zero if it is the same string.

- n specifies the maximum number of characters to compare.

Basic String Functions: strcmp/strncmp (2)

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char leftStr[] = "g f g";
    char rightStr[] = "g f g";

int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue returned by strcmp() is:  %d" , res);
    return 0;
}
```

Output:

Strings are equal
Value returned by strcmp() is: 0

University of
Nottingham
UK | CHINA | MALAYSIA

Basic String Functions: strcmp/strncmp (3)

```c
#include <stdio.h>
#include <string.h>
int main () {
  char str1[15];
  char str2[15];
  int ret;

  strcpy(str1, "abcdef");
  strcpy(str2, "mnpqrst");

  ret = strncmp(str1, str2, 4);

  if(ret  = 0) {
    printf("four first character of str1 are equal to str2");
  } else {
    printf(" four first character of str1 are not equal to str2 ");
  }
  return(0);
}
```

Output:

four first character of str1 are not equal to str2

strchr vs. strrchr

- **char *strchr(const char *str, int c)** searches for the first occurrence of the character c (an unsigned char) in the string pointed to by the argument str.
  - This returns a pointer to the first occurrence of the character c in the string str, or NULL if the character is not found.

- **char *strrchr(const char *str, int c)** searches for the last occurrence of the character c (an unsigned char) in the string pointed to, by the argument str.
  - This function returns a pointer to the last occurrence of character in str. If the value is not found, the function returns a null pointer.

strchr

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

int main () {
   const char str[] = "http://www.tutorialspoint.com";
   const char ch = '.';
   char *ret;

   ret = strchr(str, ch);

   printf("String after |%c| is - |%s|\n", ch, ret);

   return(0);
}
```

Output:

String after |.| is - |.tutorialspoint.com|

strrchr

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

int main () {
  int len;
  const char str[] = "http://www.tutorialspoint.com";
  const char ch = '.';
  char *ret;

  ret = strrchr(str, ch);

  printf("String after |%c| is - |%s|\n", ch, ret);

  return(0);
}
```

Output:

String after |.| is - |.com|

University of
Nottingham
UK | CHINA | MALAYSIA

strstr

- **char *strstr(const char *A, const char *B)** function finds the first occurrence of the substring "B" in the string "A". The terminating '\0' characters are not compared.
  - This function returns a pointer to the first occurrence in A of any of the entire sequence of characters specified in B, or a null pointer if the sequence is not present in A.

strstr (2)

```c
#include <string.h>
#include <stdio.h>
int main()
{

    char s1[] = "GeeksforGeeks";
    char s2[] = "for";
    char *p;

     p = strstr(s1, s2);

    if (p) {
        printf("String found\n");
        printf("First occurrence of string '%s' in '%s' is '%s'", s2, s1, p);
    } else
        printf("String not found\n");

    return 0;
}
```

Output:

String found
First occurrence of string 'for' in 'GeeksforGeeks' is 'forGeeks'

University of
Nottingham
UK | CHINA | MALAYSIA

strstr (3)

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


int main () {
   const char haystack[20] = "TutorialsPoint";
   const char needle[10] = "Point";
   char *ret;

   ret = strstr(haystack, needle);

   printf("The substring is: %s\n", ret);

   return(0);
}
```

Output:

The substring is: Point

Question: what is the output?

```c
#include <string.h>
#include <stdio.h>
int main()
{

  char s1[] = "Fun with Coding";
  char s2[] = "Coding";
  char* p;
  p = strstr(s1, s2);


  if (p) {
    strcpy(p, "Strings");
    printf("%s", s1);
  } else
    printf("String not found\n");


  return 0;
}
```

strspn

- **strspn (const char * str1, const char * str2)** returns the length of the initial portion of str1 which consists only of characters that are part of str2.

```c
#include <stdio.h>
#include <string.h>
int main ()
{
  int i;
  char str1[] = "129th";
  char str2[] = "1234567890";

  i = strspn (str1,str2);
  printf ("The initial number has %d digits.\n",i);
  return 0;
}
```

Output:

The initial number has 3 digits.

University of Nottingham
UK | CHINA | MALAYSIA

Search String

- #include <string.h>

Function prototypes and descriptions

```
char *strchr(const char *s, int c);
```
Locates the first occurrence of character c in string s. If c is found, a pointer to c in s is returned. Otherwise, a NULL pointer is returned.

```
size_t strcspn(const char *s1, const char *s2);
```
Determines and returns the length of the initial segment of string s1 consisting of characters *not* contained in string s2.

```
size_t strspn(const char *s1, const char *s2);
```
Determines and returns the length of the initial segment of string s1 consisting *only* of characters contained in string s2.

```
char *strpbrk(const char *s1, const char *s2);
```
Locates the first occurrence in string s1 of any character in string s2. If a character from string s2 is found, a pointer to the character in string s1 is returned. Otherwise, a NULL pointer is returned.

```
char *strrchr(const char *s, int c);
```
Locates the last occurrence of c in string s. If c is found, a pointer to c in string s is returned. Otherwise, a NULL pointer is returned.

```
char *strstr(const char *s1, const char *s2);
```
Locates the first occurrence in string s1 of string s2. If the string is found, a pointer to the string in s1 is returned. Otherwise, a NULL pointer is returned.

```
char *strtok(char *s1, const char *s2);
```
A sequence of calls to strtok breaks string s1 into *tokens*—logical pieces such as words in a line of text—separated by characters contained in string s2. The first call contains s1 as the first argument, and subsequent calls to continue tokenizing the same string contain NULL as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, NULL is returned.

Source: Deitel and Deiltel (2016). C How to Program with an Introduction to C++ (8th Ed.). Pearson.

University of Nottingham
UK | CHINA | MALAYSIA

Summary

- Character-handling library
- String-conversion functions
- Standard input and output functions
- Search string