Week 6 - Lecture 1, 2

Characters and Strings

Edited by: Dr. Wooi Ping Cheah

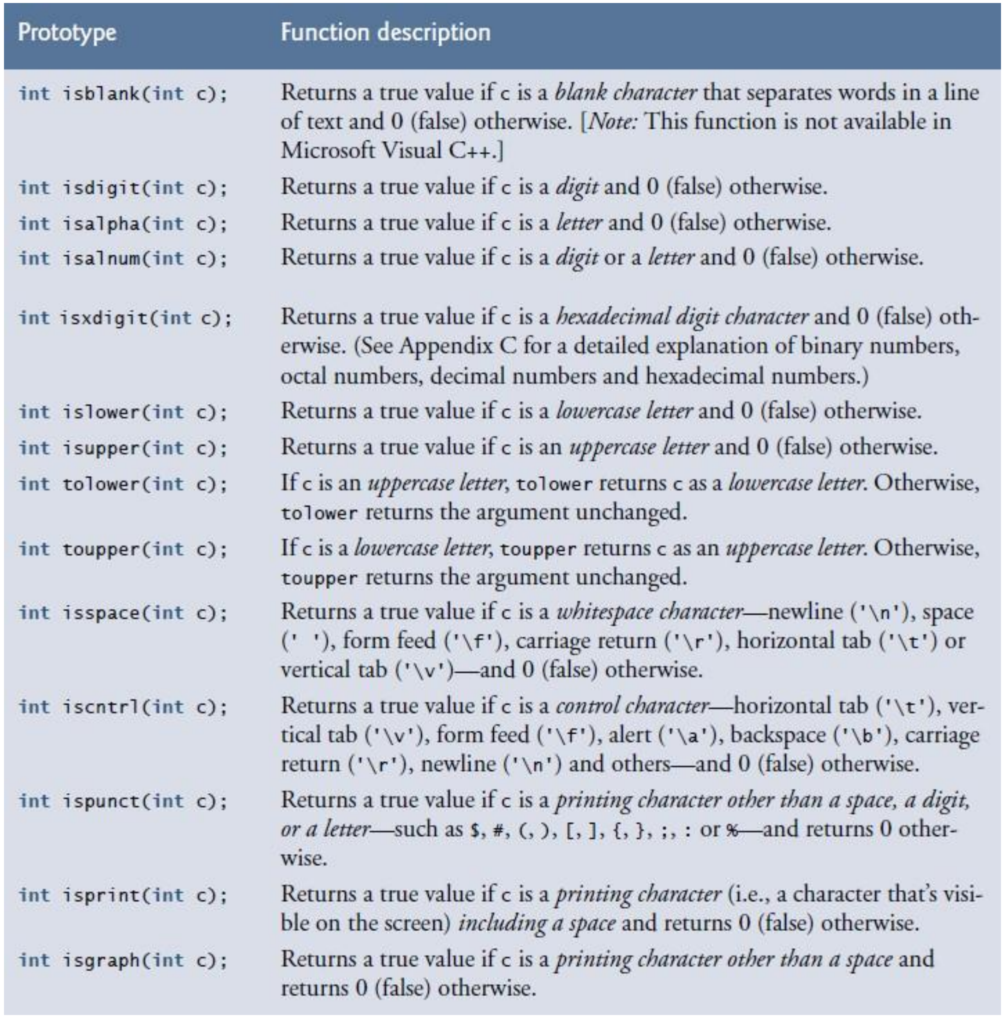
Autumn 2022

Overview

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

Character-Handling Library

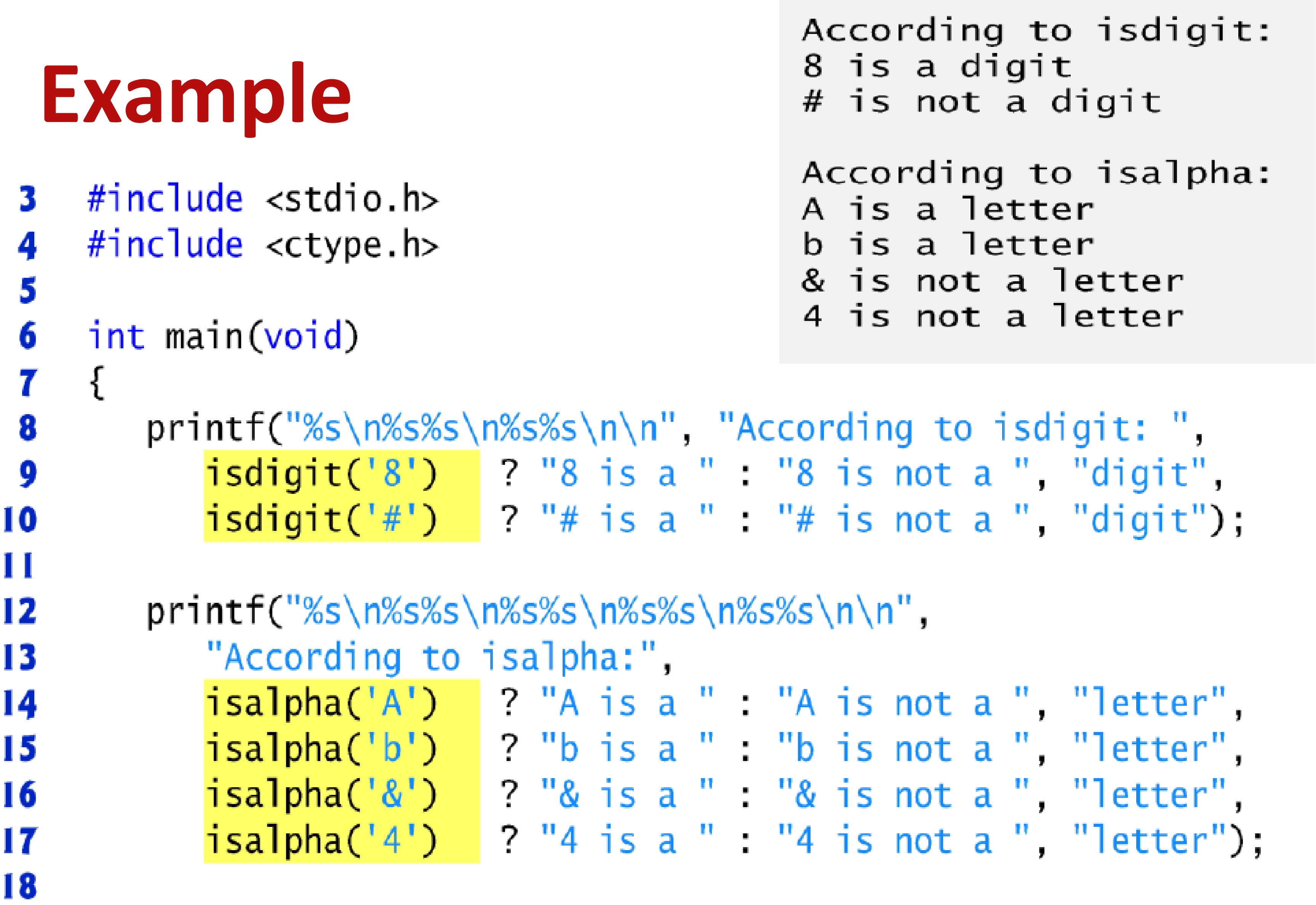
- #include <ctype.h>



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

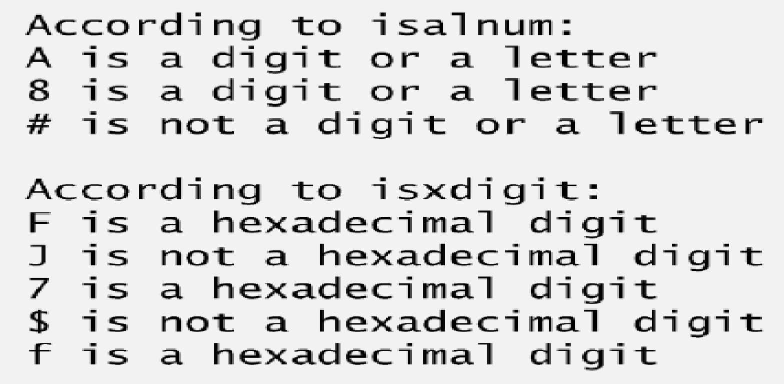


```
According to isdigit:
8 is a digit
# is not a digit

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

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

int main(void)
{
   printf("%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
      isdigit('8')   ? "8 is a " : "8 is not a ", "digit",
      isdigit('#')   ? "# is a " : "# is not a ", "digit");

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

```

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

Example (2)

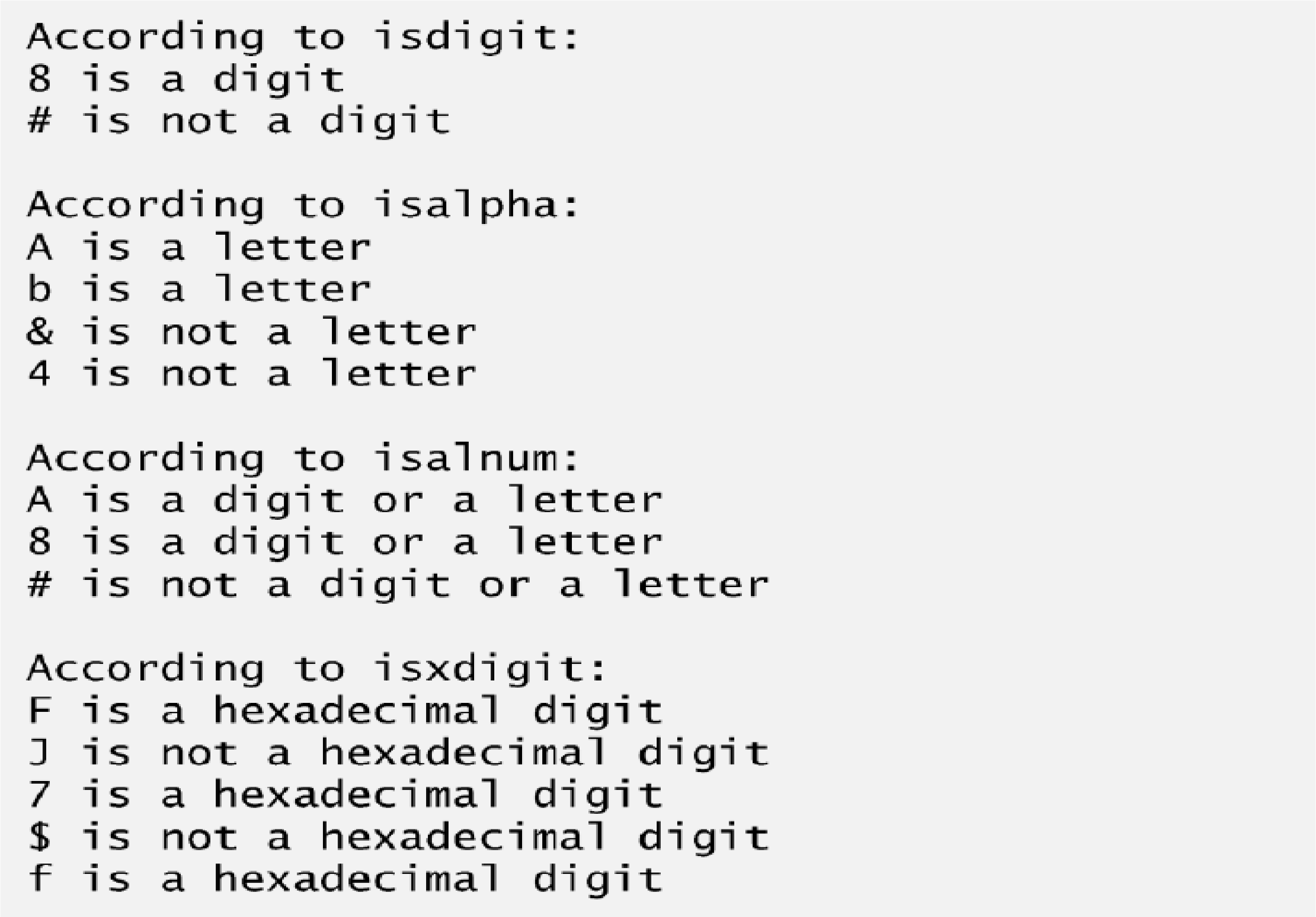


```
According to isalnum:
A is a digit or a letter
8 is a digit or a letter
# is not a digit or a letter

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
```

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

   printf("%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
      "According to isxdigit:",
      isxdigit('F')   ? "F is a " : "F is not a ",
      "hexadecimal digit",
      isxdigit('J')   ? "J is a " : "J is not a ",
      "hexadecimal digit",
      isxdigit('7')   ? "7 is a " : "7 is not a ",
      "hexadecimal digit",
      isxdigit('$')   ? "$ is a " : "$ is not a ",
      "hexadecimal digit",
      isxdigit('f')   ? "f is a " : "f is not a ",
      "hexadecimal digit");
}
```

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

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

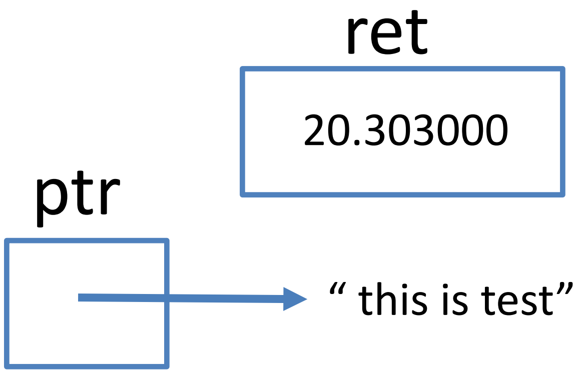
Strtoul(): converts string to unsigned long

Example: strtod

```
#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);
}
```

Output:

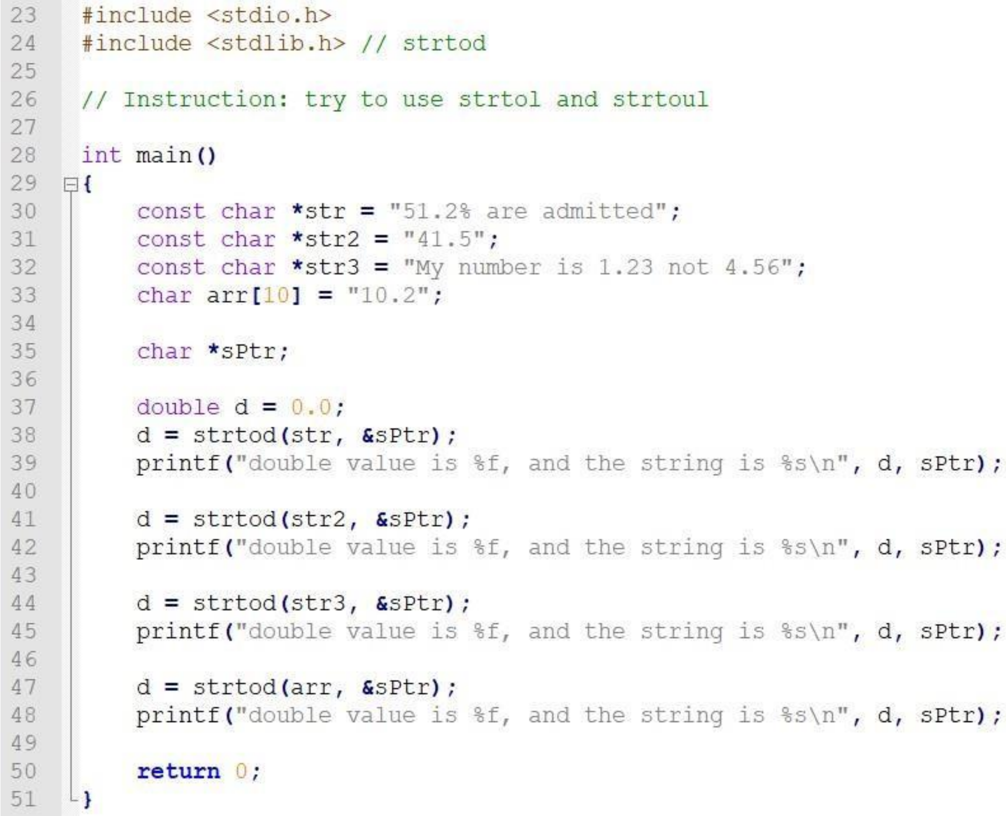
```
The number(double) is 20.303000
String part is | this is test|
```



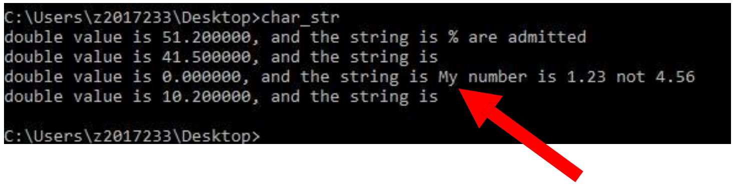
Example: strtod (1)

The string must begins with a valid floating point number.

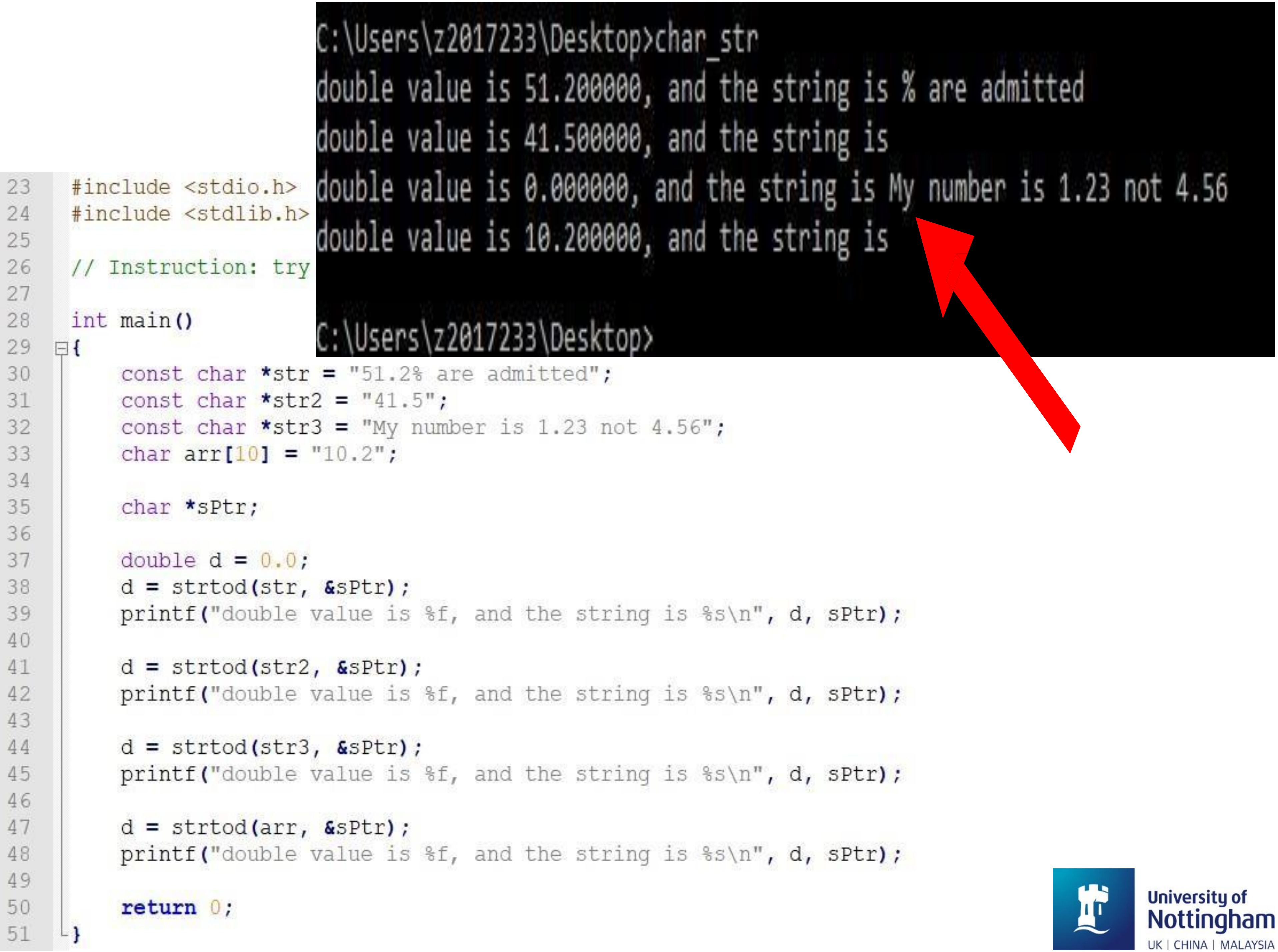
- The pointer receives the memory address of the character after floating point value.
- On error, point to the beginning of the string.



```
#include <stdio.h>
#include <stdlib.h> // strtod

// Instruction: try to use strtol and strtoul

int main()
{
    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";

    char *sPtr;

    double d = 0.0;
    d = strtod(str, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    d = strtod(str2, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    d = strtod(str3, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    d = strtod(arr, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    return 0;
}
```



```
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>
```

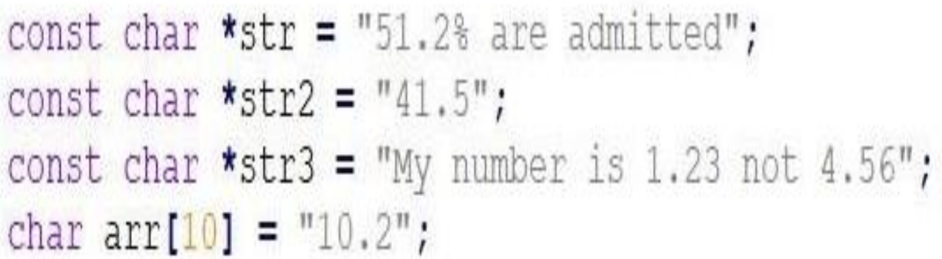


```
C:\Users\z2017233\Desktop>char_str
double value is 51.200000, and the string is % are admitted
double value is 41.500000, and the string is
double value is 0.000000, and the string is My number is 1.23 not 4.56
double value is 10.200000, and the string is

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

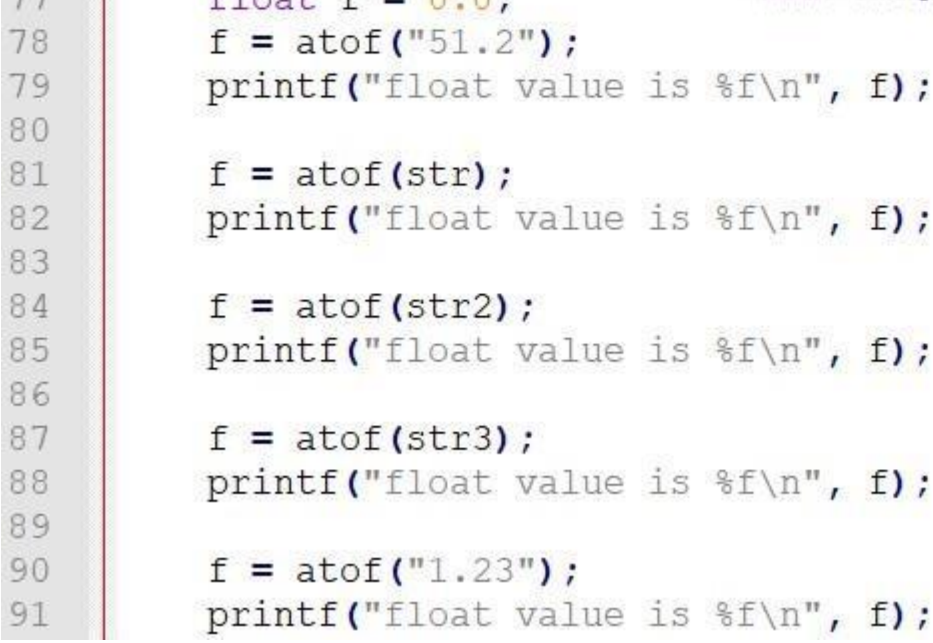
```c
#include <stdio.h>
#include <stdlib.h>

// Instruction: try

int main()
{
    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";

    char *sPtr;

    double d = 0.0;
    d = strtod(str, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    d = strtod(str2, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    d = strtod(str3, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    d = strtod(arr, &sPtr);
    printf("double value is %f, and the string is %s\n", d, sPtr);

    return 0;
}
```

Example: atof, atol

- Converts string to float /long interger.



```
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";
```



```
        float f = 0.0;
        f = atof("51.2");
        printf("float value is %f\n", f);

        f = atof(str);
        printf("float value is %f\n", f);

        f = atof(str2);
        printf("float value is %f\n", f);

        f = atof(str3);
        printf("float value is %f\n", f);

        f = atof("1.23");
        printf("float value is %f\n", f);
```

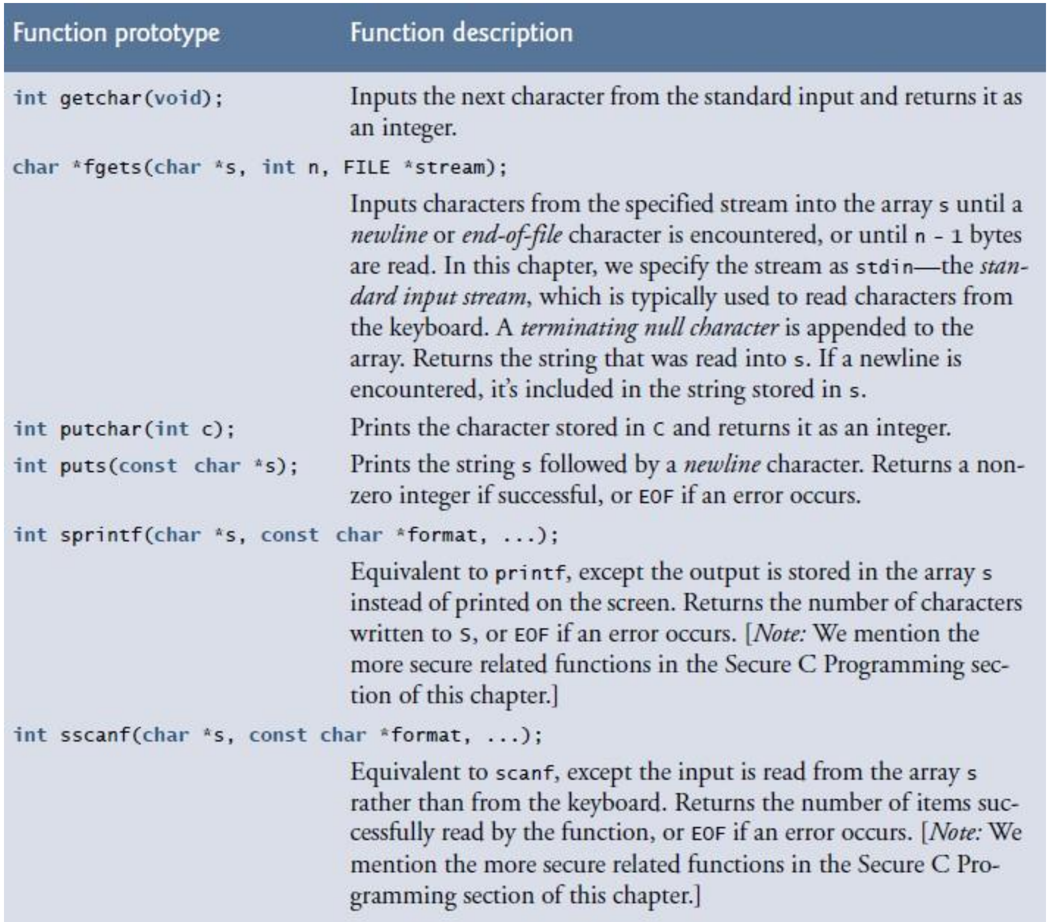
```
Float value is: 51.200000
Float value is: 51.200000
Float value is: 41.500000
Float value is: 0.000000
Float value is: 1.2300000
```

Overview

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

Standard Input/Output Functions

- #include <stdio.h>

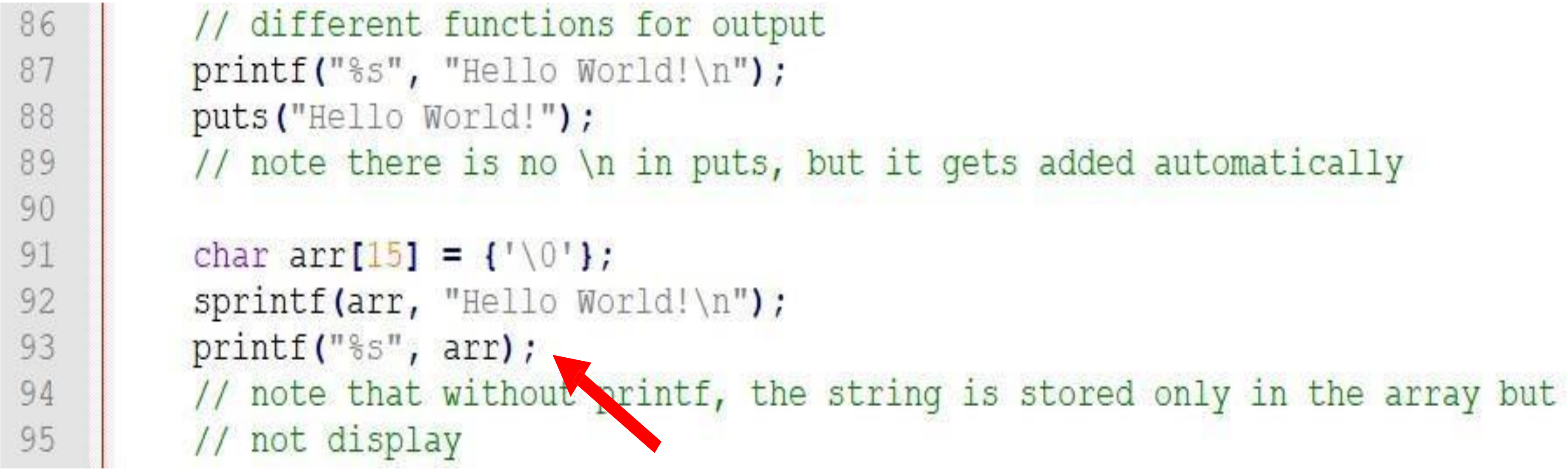


Function prototype	Function description
`int getchar(void);`	Inputs the next character from the standard input and returns it as an integer.
`char *fgets(char *s, int n, FILE *stream);`	Inputs characters from the specified stream into the array `s` until a *newline* or *end-of-file* character is encountered, 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.

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!



```c
// different functions for output
printf("%s", "Hello World!\n");
puts("Hello World!");
// note there is no \n in puts, but it gets added automatically

char arr[15] = {'\0'};
sprintf(arr, "Hello World!\n");
printf("%s", arr);
// note that without printf, the string is stored only in the array but
// not display
```

sscanf

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

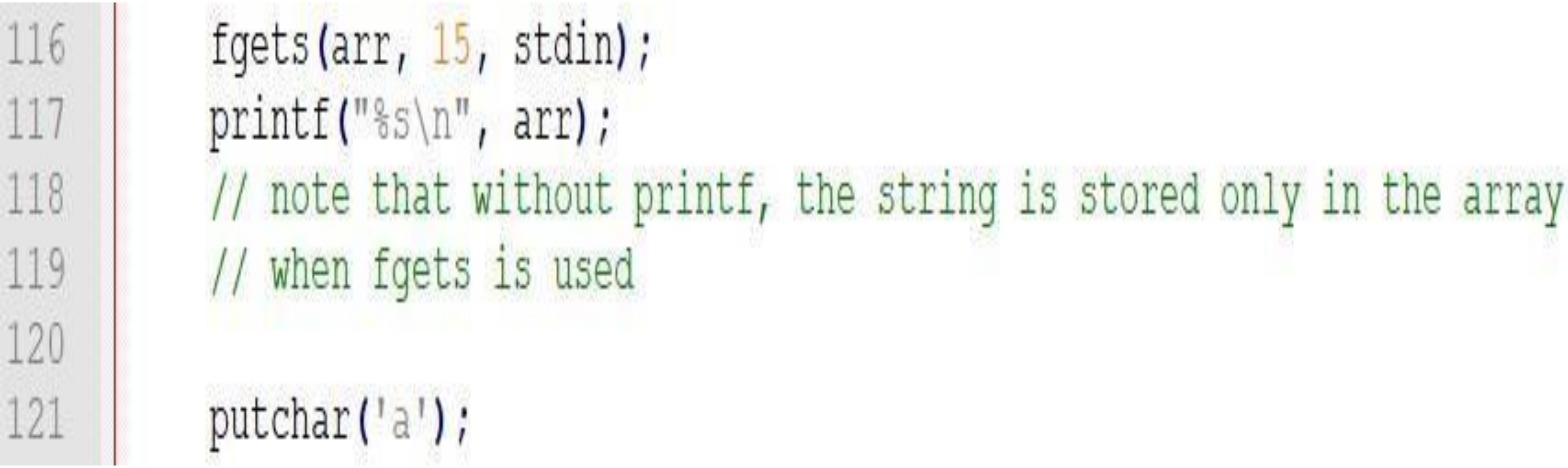
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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.



```
fgets(arr, 15, stdin);
printf("%s\n", arr);
// note that without printf, the string is stored only in the array
// when fgets is used

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

```
#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
```

Basic String Functions: strncpy

Strncpy: dest <- src for n character.

```
#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.

```
#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.

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

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

  strncat(dest, src, 5);

  // Prints the string

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

  return 0;
}
```

Output:

```
Destination string : abcdefghi
Source string : efghijkl
```

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)

```
#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
```

Basic String Functions: strcmp/strncmp (3)

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

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

  ret = strncmp(str1, str2, 4);

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


  return(0);
}
```

Output:

```
four first characters of str1 are 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

```
#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

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

int main () {
   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|
```

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)

```
#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'
```

strstr (3)

```
#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
```

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.

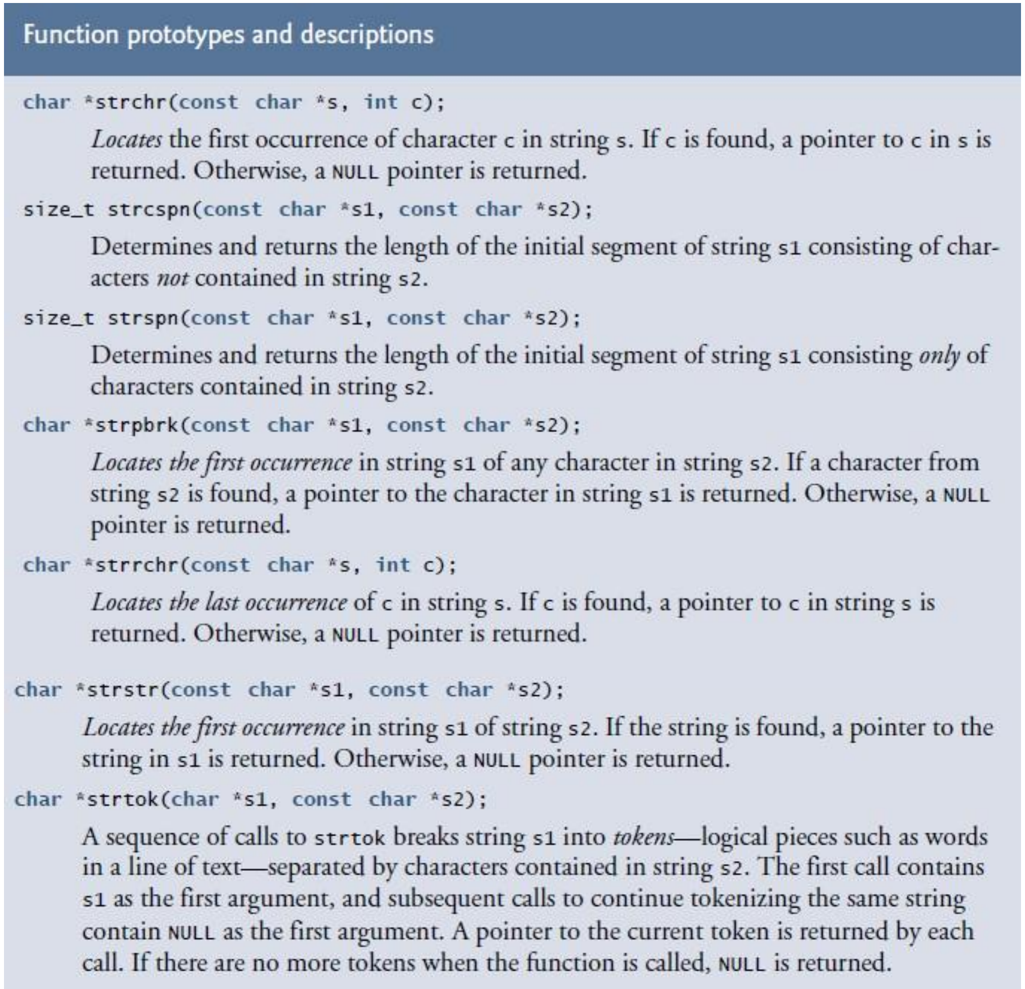
```
#include <stdio.h>
#include <string.h>
int main ()
{
  int i;
  char str1[] = "12t9h8";
  char str2[] = "t1234567890";

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

Output:

```
The initial number has 4 digits.
```

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.

Summary

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