STRINGS

30 October 2024 15:56

STRINGS
a string is a character or symbol between a pair of quote characters.

C has no string type.

DECLARING STRING:

char my_full_name[25]; /* When you want to specifically reserve a certain amount of memory to store the string */

char my_full_name[] = "This is a string."; /*The length will be automatically defined based on the initialised string */

String is always followed by a termination or null character \0.

Get string input :

Gets() input function used to read strings or text line from stdin

```
#include<stdio.h>
int main()
{
    char string[10];
    puts("Enter the String: ");
    gets(string);
    printf("\n%s",string);
    return 0;
}
```

```
INPUT ---------> hello word
OUTPUT----------> hello word
```

Fgets() input function used to read string or text line from a file or stdin.
reads data untill it either encounters a newline or exceeds the specified buffer length

fgets(char *str, int n, FILE *stream)

- **str** - It is the variable in which the string is going to be stored
- **n** - It is the maximum length of the string that should be read
- **stream** - It is the filehandle, from where the string is to be read.

READ FROM STDIN WITH FGETS()

```
#include<stdio.h>
int main()
{
    char string[20];
    printf("Enter the string: ");
    fgets(string,20,stdin);        #input from stdin stream
    printf("\nThe string is: %s",string);
    return 0;
}
```

```
Input ----------> god is good
Output -----> god is good
```

Read from a given file using fgets()

```
#include<stdio.h>
int main()
{
    char string[20];
    FILE *fp;
    fp=fopen("file.txt","r");
    fgets(string,20,fp);
    printf("The string is: %s",string);
    fclose(fp);
    return 0;
}
```

String.h
 c library containing functions that help with string.
#<include string.h>

Strlen() gives you length of string
Strcpy(to where, from where)
Strcat(source, destination); concatena due stringhe, dove viene salvata la stringa? In the first argument of the function, so make sure you have enough space in the first parametro per mettere entrambi

String comparison: [most common application is checking if result is 0]
Strcmp() compares strings based on ASCII values (so if they have same number) basically a - a = 0

Char str1[] = A
Char str2[] = A
Char str3[] = a

Compare = strcmp(str1,str2);   A-A = 0
Print(%s,compare) ------> this returns 0, because they are the same
When it encounters the first difference it stops.
Get ascii :
Print(%d,"A")

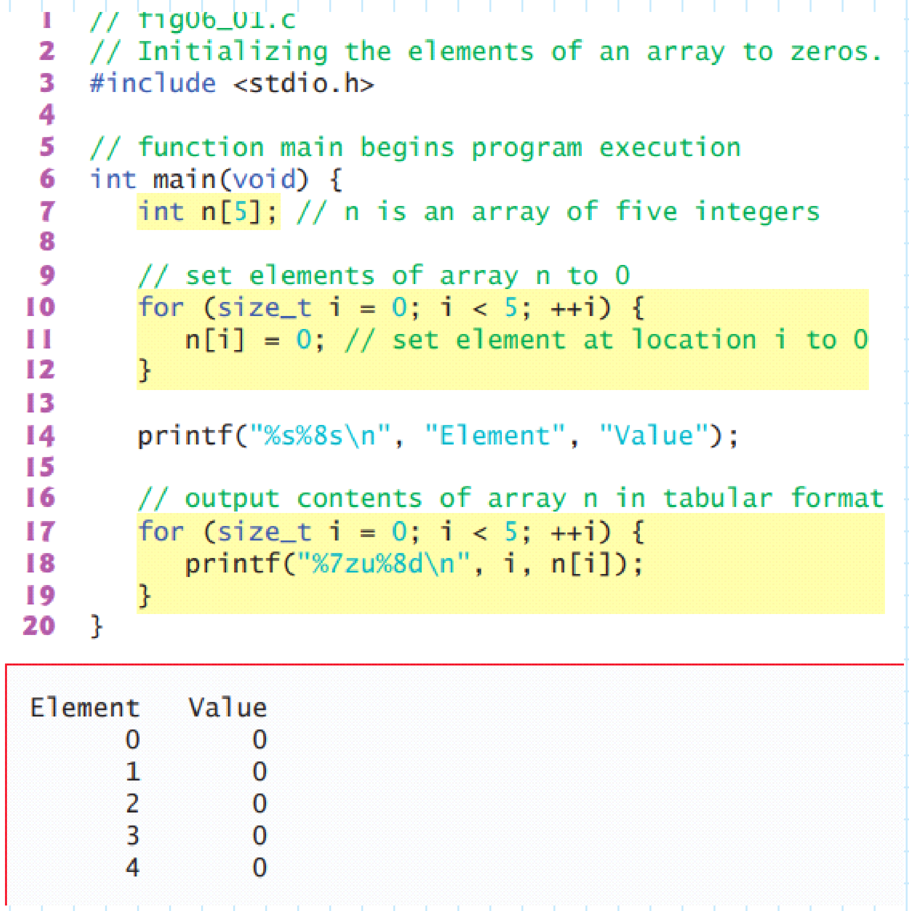
CTYPE.H
Another c library that helps you transform the characters.

Math library. <Math.h>

pow(u,m)   power operator   [u to the power of m]

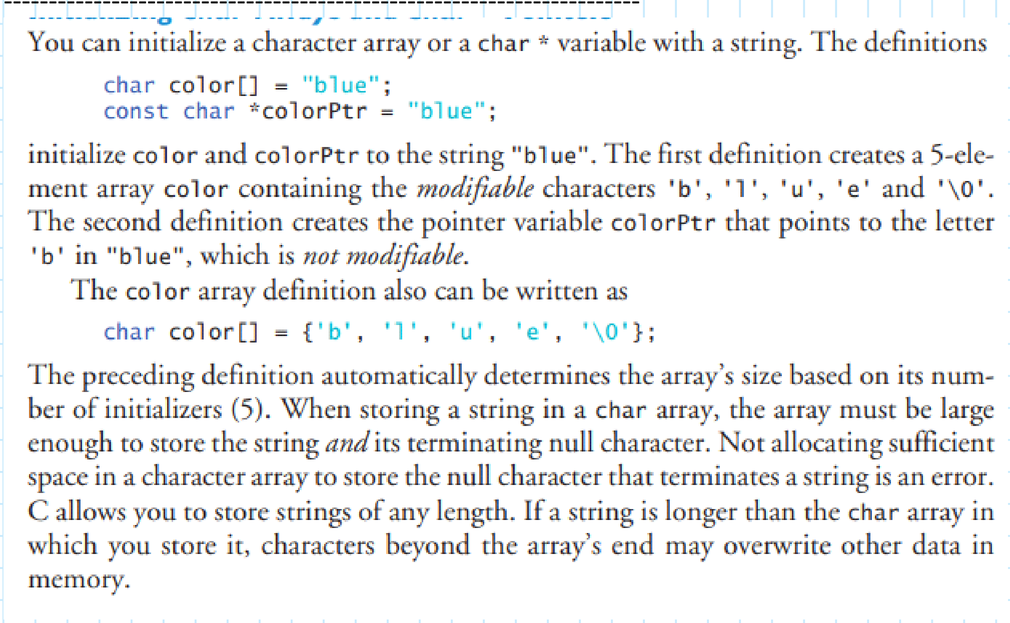
FORMATTING:
NUM:
%.numF precision point %.2f == 2.65
%num.numF field width %2.2f == (one leading splace)2.65

STRING:
("%S%8s\n, 'elements', 'value');
Will print elements then print value separated from element by 8 spaces
So % + number + type sepcification without % put spaces.
%7zu\n -> prints type size_t with 7 spaces before it



```c
// fig06_01.c
// Initializing the elements of an array to zeros.
#include <stdio.h>

// function main begins program execution
int main(void) {
   int n[5]; // n is an array of five integers

   // set elements of array n to 0
   for (size_t i = 0; i < 5; ++i) {
      n[i] = 0; // set element at location i to 0
   }

   printf("%s%8s\n", "Element", "Value");

   // output contents of array n in tabular format
   for (size_t i = 0; i < 5; ++i) {
      printf("%7zu%8d\n", i, n[i]);
   }
}
```

```
Element   Value
      0       0
      1       0
      2       0
      3       0
      4       0
```



You can initialize a character array or a char * variable with a string. The definitions

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

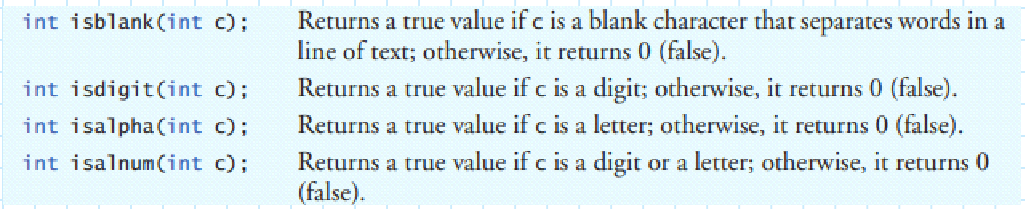
initialize color and colorPtr to the string "blue". The first definition creates a 5-element array color containing the *modifiable* characters 'b', 'l', 'u', 'e' and '\0'. The second definition creates the pointer variable colorPtr that points to the letter 'b' in "blue", which is *not modifiable*.

The color array definition also can be written as

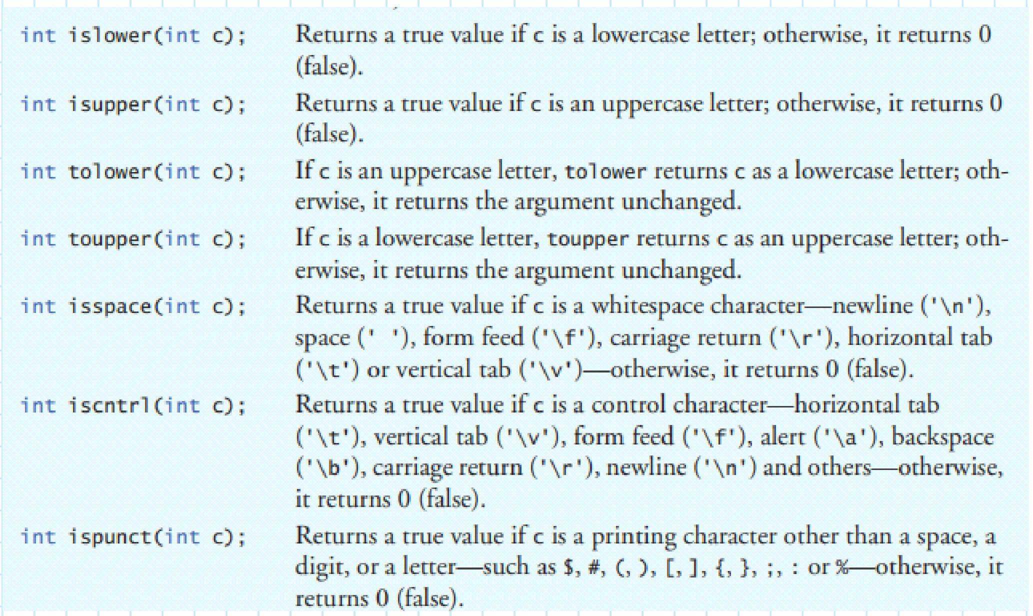
```c
char color[] = {'b', 'l', 'u', 'e', '\0'};
```

The preceding definition automatically determines the array's size based on its number of initializers (5). When storing a string in a char array, the array must be large enough to store the string *and* its terminating null character. Not allocating sufficient space in a character array to store the null character that terminates a string is an error. C allows you to store strings of any length. If a string is longer than the char array in which you store it, characters beyond the array's end may overwrite other data in memory.

Library ctype.h



`int isblank(int c);`	Returns a true value if c is a blank character that separates words in a line of text; otherwise, it returns 0 (false).
`int isdigit(int c);`	Returns a true value if c is a digit; otherwise, it returns 0 (false).
`int isalpha(int c);`	Returns a true value if c is a letter; otherwise, it returns 0 (false).
`int isalnum(int c);`	Returns a true value if c is a digit or a letter; otherwise, it returns 0 (false).



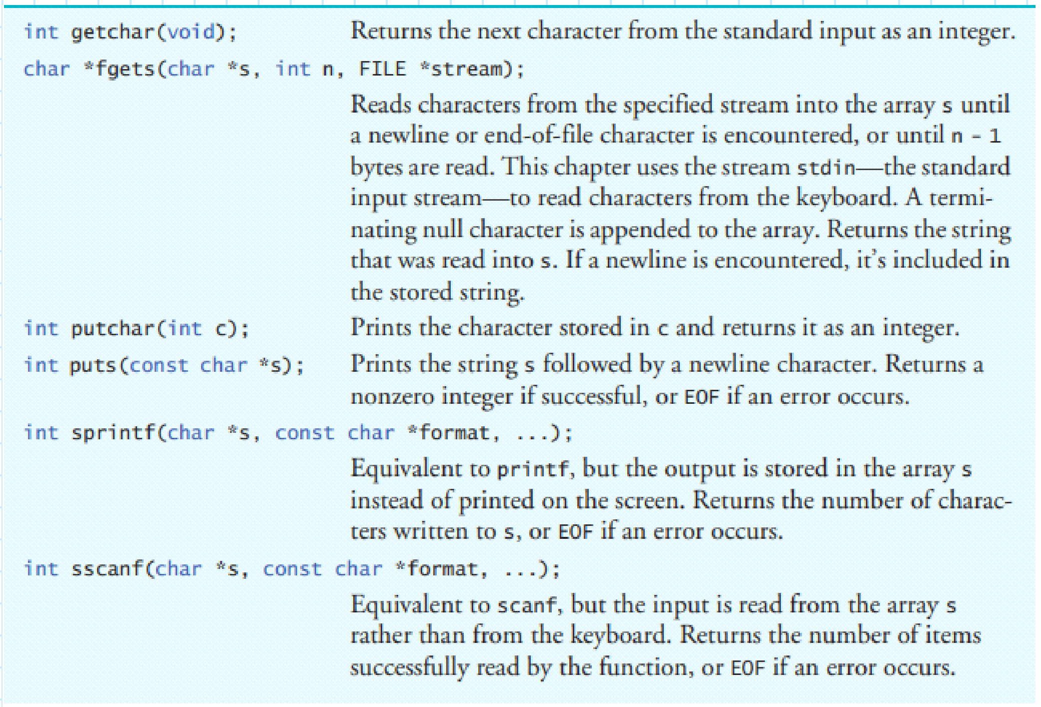
`int islower(int c);`	Returns a true value if `c` is a lowercase letter; otherwise, it returns 0 (false).
`int isupper(int c);`	Returns a true value if `c` is an uppercase letter; otherwise, it returns 0 (false).
`int tolower(int c);`	If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter; otherwise, it returns the argument unchanged.
`int toupper(int c);`	If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter; otherwise, it 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'`)—otherwise, it returns 0 (false).
`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—otherwise, it returns 0 (false).
`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 `%`—otherwise, it returns 0 (false).

String conversions: STDLIB.H
String to double   double strtod(const char *nPtr, char **endPtr);

String to long int   long strtol(const char *nPtr, char **endPtr, int base);

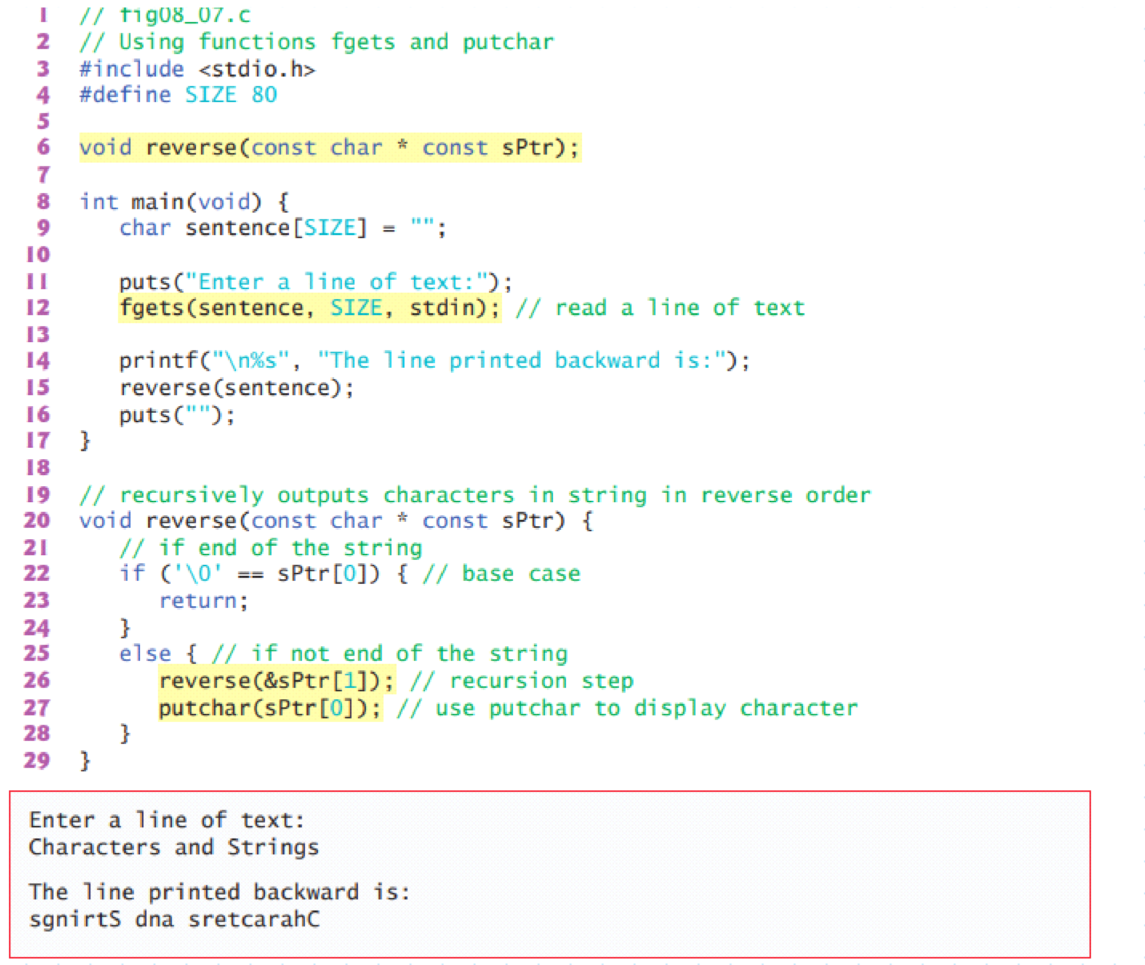
String to unsigned long   unsigned long strtoul(const char *nPtr, char **endPtr, int base);

STDIO.H



`int getchar(void);`	Returns the next character from the standard input as an integer.
`char *fgets(char *s, int n, FILE *stream);`	Reads 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. This chapter uses the stream `stdin`—the standard input stream—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 stored string.
`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 nonzero integer if successful, or `EOF` if an error occurs.
`int sprintf(char *s, const char *format, ...);`	Equivalent to `printf`, but 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.
`int sscanf(char *s, const char *format, ...);`	Equivalent to `scanf`, but 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.

s fgets to read characters into its char array argument until it encounters a newline or the end-of-file indicator, or until the maximum number of characters is read.
The maximum number of characters is one fewer than fgets's second argument. The third argument is the stream from which to read characters—in this case, the standard input stream (stdin). When reading terminates, fgets appends a null character ('\0') to the array



```c
// fig08_07.c
// Using functions fgets and putchar
#include <stdio.h>
#define SIZE 80

void reverse(const char * const sPtr);

int main(void) {
   char sentence[SIZE] = "";

   puts("Enter a line of text:");
   fgets(sentence, SIZE, stdin); // read a line of text

   printf("\n%s", "The line printed backward is:");
   reverse(sentence);
   puts("");
}

// recursively outputs characters in string in reverse order
void reverse(const char * const sPtr) {
   // if end of the string
   if ('\0' == sPtr[0]) { // base case
      return;
   }
   else { // if not end of the string
      reverse(&sPtr[1]); // recursion step
      putchar(sPtr[0]); // use putchar to display character
   }
}
```

```
Enter a line of text:
Characters and Strings

The line printed backward is:
sgnirtS dna sretcarahC
```

SEARCH FUNCTION

Strchr() search for the first occurrence of a character in a string, returns a pointer to the char or null

Strpbrk() searches its first string argument for the first occurrence of any character in its second string argument. If a character from the second argument is found, strpbrk returns a pointer to the character in the first argument; otherwise, it returns NULL.

Strrchr() searches for the last occurrence of the specified character in a string. If the character is found, strrchr returns a pointer to the character in the string; otherwise, it returns NULL.