Array - String

Structured Programming Language (CSE-1271)

Course Instructor : Mohammed Mamun Hossain
Assistant Professor, Dept. of CSE, BAUST

Outline

1. Array
2. String

Array

Arrays is the collection of elements of the same data types. or

A collection of objects of the *same type* stored contiguously in memory under one name

Some Applications:

- Given a list of test scores, determine the maximum and minimum scores.
- Read in a list of student names and rearrange them in alphabetical order (sorting).
- Given the height measurements of students in a class, output the names of those students who are taller than average.

Types of Array

array

One dimensional Array:

```
int my_array [100];
```

These variables are:
my_array[0],
my_array[1],
my_array[2],
…
my_array[99].

Let see the addressing/indexing :

0, 1, 2, … 99.

Multi dimensional Array:

```
int my_array [10][10];
```

```
int my_array [10][100][30];
```

my_array[0][1],
my_array[0][2],
…
my_array[9][9].

00, 01, 02, … 09,
10, 11, 12, … 19,
…
90, 91, 92, … 99.

Array indexes *always* start at zero in *C*

One Dimensional Array

- ✓ In C, a one dimensional array is a list of variables that are all of the same type and are accessed through a common name.
- ✓ An individual variable in the array is called an array element.
- ✓ It is useful to handle groups of related data.

General form- type var_name[size]

```
int my_array [100];
```

Type

Array name

Size –
can hold 100 elements

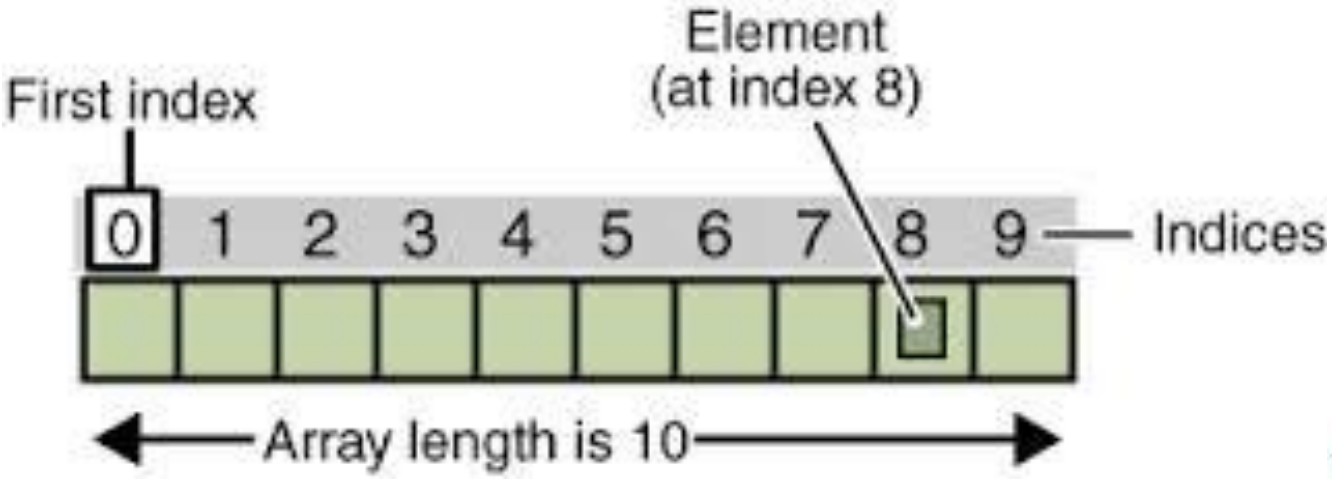
- ➢ type is a valid C data type.
- ➢ var_name is the name of the array.
- ➢ size specifies the number of elements in the array.

One Dimensional Array

An array element is accessed by indexing the array using the number of the element.

In C, all arrays begin at zero.

```
int my_array [10];
```




```
int main()
{
    int i, v=2, my_array[10];
    for(i=0; i<=9; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=9; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
UK	2	UK	UK	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
0	2	UK	UK	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
0	2	2	UK	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
0	4	2	UK	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
1	4	2	UK	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
1	4	2	4	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
1	8	2	4	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
2	8	2	4	UK	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
2	8	2	4	8	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
2	16	2	4	8	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
3	16	2	4	8	UK	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
3	16	2	4	8	16	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

3	32	2	4	8	16	UK
		0	1	2	3	4
i	v	my_array				

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
4	32	2	4	8	16	UK
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
4	32	2	4	8	16	32
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
4	64	2	4	8	16	32
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

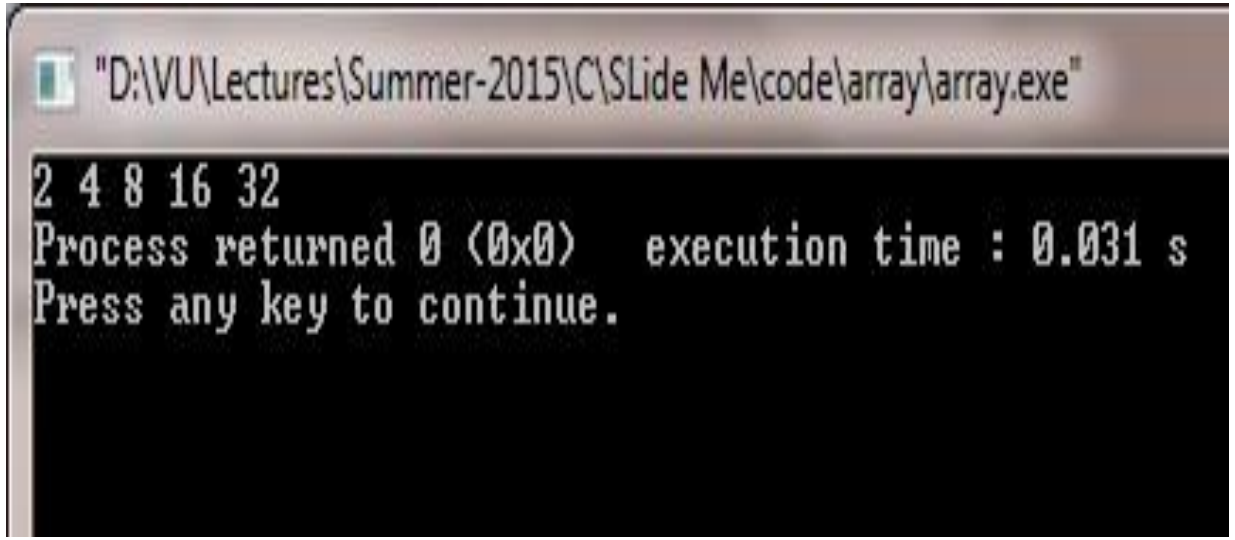
```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

i	v	my_array				
5	64	2	4	8	16	32
		0	1	2	3	4

One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=0; i<=4; i++)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

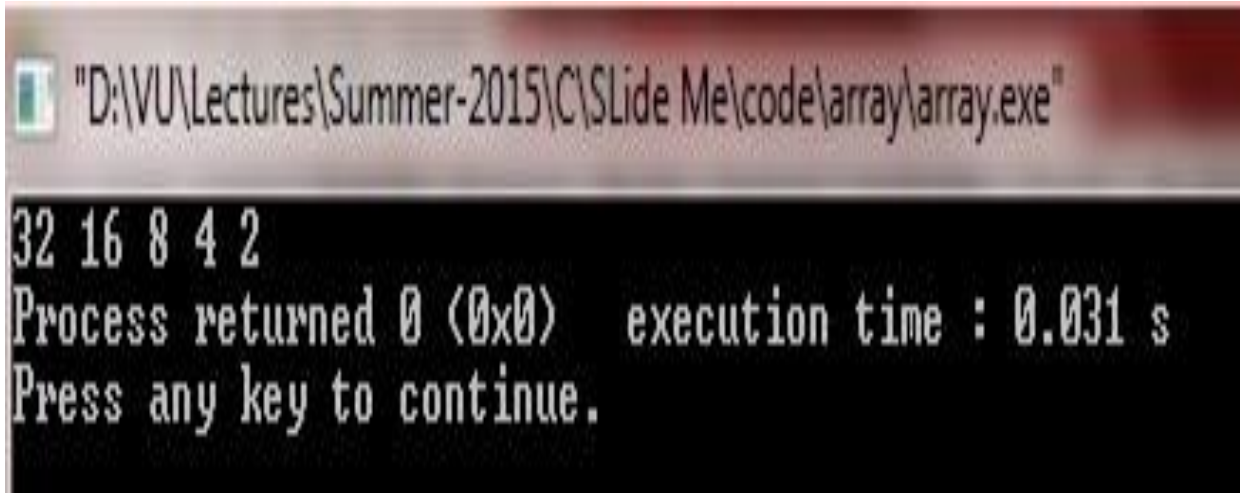
i	v	my_array				
0	64	2	4	8	16	32
		0	1	2	3	4



One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    for(i=4; i>=0; i--)
    {
        printf("%d ",my_array[i]);
    }

    return 0;
}
```

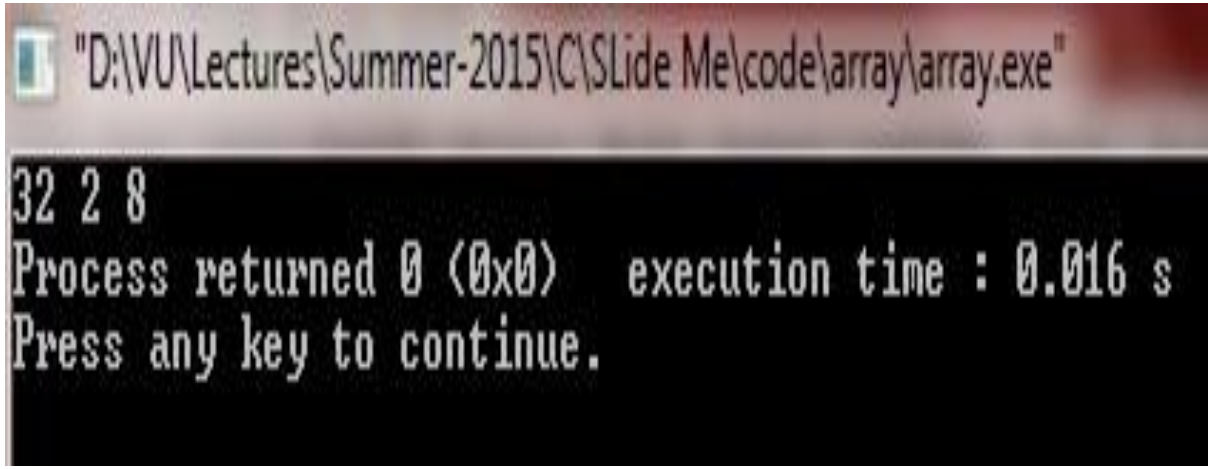
i	v	my_array				
0	64	2	4	8	16	32
		0	1	2	3	4



One Dimensional Array (Input from Keyboard)

```c
int main()
{
    int i, v=2, my_array[5];
    for(i=0; i<=4; i++)
    {
        my_array[i]=v;
        v=v*2;
    }
    printf("%d ",my_array[4]);
    printf("%d ",my_array[0]);
    printf("%d ",my_array[2]);

    return 0;
}
```

i	v	my_array				
0	64	2	4	8	16	32
		0	1	2	3	4



One Dimensional Array (Initializing)

Variable Initialization:

```
int sum = 0;
```

```
int sum;

sum = 0;
```

Array Initialization:

```
int a[5] = {1, 2, 3, 4, 5};
```

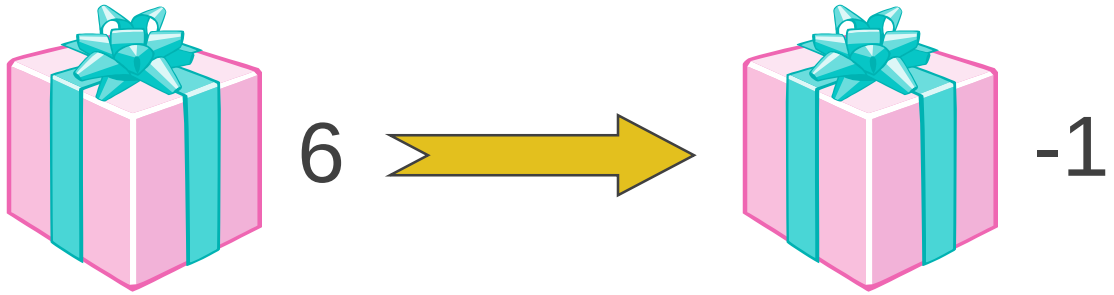
```
int a[5];

a[5] = {1, 2, 3, 4, 5};
```

One Dimensional Array (Initializing)

```
int Ar[10] = {9, 8, 7, 6, 5, 12, 3, -2, 1, 0};
```

	0	1	2	3	4	5	6	7	8	9
Ar	9	8	7	6	5	12	3	-2	1	0

```
Ar[3] = -1;
```



	0	1	2	3	4	5	6	7	8	9
Ar	9	8	7	-1	5	12	3	-2	1	0

Array (Histogram)

```c
#include <stdio.h>
#define SIZE 10

int main()
{
    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
    int i, j;

    printf( "%7s%13s%17s\n", "Index", "Value", "Histogram" );

    for ( i = 0; i <= SIZE - 1; i++ )
    {
        printf( "%7d%13d        ", i, n[ i ]) ;
        for ( j = 1; j <= n[ i ]; j++ )    /* print one bar */
        {
            printf( "%c", '*' );
        }
        printf( "\n" );
    }
return 0;
}
```

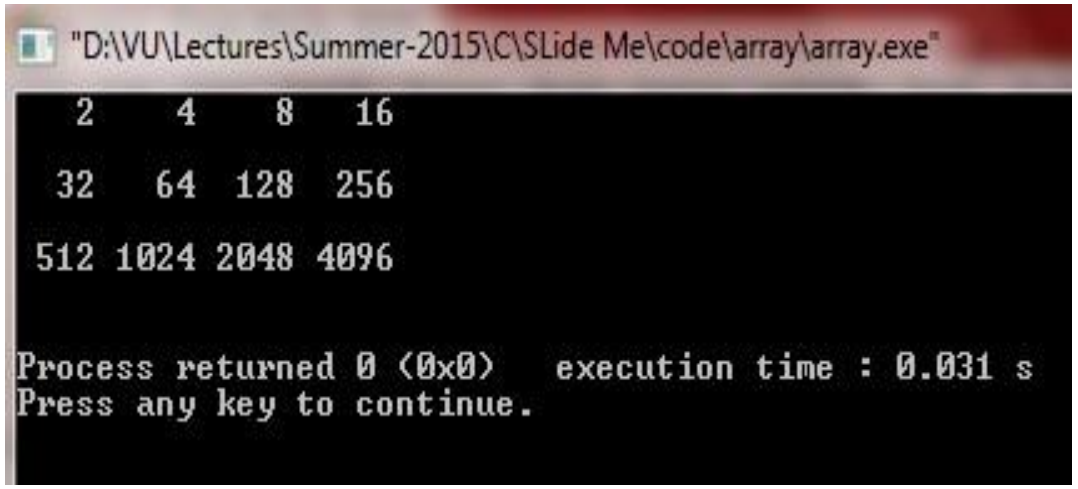
What will be the output?

Two Dimensional Array

```c
int main()
{
    int r, c, v=2, my_array[3][4], ROW=3,COL=4;
    for(r=0; r<=ROW-1; r++)
    {
        for(c=0; c<=COL-1; c++)
        {
            my_array[r][c]=v;
            v=v*2;
        }
    }
    for(r=0; r<=ROW-1; r++)
    {
        for(c=0; c<=COL-1; c++)
        {
            printf("%4d ",my_array[r][c]);
        }
        printf("\n\n");
    }

    return 0;
}
```

	0	1	2	3
0	2	4	8	16
1	32	64	128	256
2	512	1024	2048	4096

0,3



Two Dimensional Array

Please try to manipulate two dimensional array.

For matrix addition

For matrix subtraction

Access diagonal values of a matrix etc

String

String(character) Array

How to manipulate names to your software (program)?

- ✓ Google Inc.
- ✓ Yahoo Inc.
- ✓ Samsung
- ✓ Apple Inc.
- ✓ Microsoft Corporation
- ✓ Adobe Systems Incorporated
- ✓ Twitter, Inc.
- ✓ Facebook, Inc.

These are character data. But `char` data type only store one character.

The solution is string (character array).

- The string is actually a one-dimensional array of characters which is terminated by a **null** character '\0'.
- To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."
  - `char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};`

String(character) Array

How to declare?

```
type name_of_araay[size];
```

n is how many character needed for representing name.

Example: `char company_name[100];`

It's 100 element array.

We can use each element same as normal char variable.

```c
#include<stdio.h>

int main()
{
    char company_name[100];

    scanf("%s",company_name);
    printf("%s",company_name);

    //gets(company_name);
    //puts(company_name);

    return 0;
}
```

These variables are:

company_name[0],
company_name[1],
company_name[2],
…
company_name[99].

String(character) Array

- ✓ Character arrays
  - String "**first**" is really a static array of characters
  - Character arrays can be initialized using string literals

```
char string1[] = "first";
```

    - Null character '**\0**' terminates strings
    - **string1** actually has 6 elements
      - It is equivalent to

```
char string1[] = {'f', 'i', 'r', 's', 't', '\0'};
```

- ✓ Can access individual characters

  string1[3] is character 's'

- ✓ Array name is address of array, so & not needed for scanf ()

```
scanf( "%s", string2 );
```

    - Reads characters until whitespace encountered

String(character) Array

- A string is defined as null terminated character array.
- A string must be terminated by a null – means that we need one byte extra for holding null character.
- A string constant is null-terminated by the compiler automatically.
- Each character takes one byte in string.

Read String From Keyboard

- There are several ways-
  - ✓ Using scanf() function-for string C use “%s” format specifier
  - ✓ Most common is: gets()

`gets()`

- C’s standard library function
- uses the `STDIO.H` header file
- To use – call it using the name of a character array without any index.

Read String From Keyboard -gets()

```
char str[100];
gets(str);
```

- gets() function reads character until we press ENTER
- The ENTER key (carriage return - \r) is not stored, but is replaced by a null, which terminates the string.

Be aware!

- The gets() function performs no bounds checking.
- It is possible to enter more characters than the array receiving them can hold – **unwanted result!**
- For example, if we call gets() with an array that is 20 characters long, there is no mechanism to stop us from entering more than 20 character.

Write String to Monitor-puts()

- C's standard library function
- uses the STDIO.H header file
- To use – call it using the name of a ch

```
char str[100] = "This is it";
puts(str);
```

```
int main()
{
    char str[80];
    int i;

    printf("Enter a string: ");
    gets(str);

    puts(str);

    return 0;
}
```

String – STRING.H

- The most string related library functions are

1) strcpy() – for copy

```
strcpy(to, from);
```

2) strcat() – for concatenation

```
strcat(to, from);
```

3) strcmp() – for compare two strings

```
strcmp(s1,s2);
```

4) strncmp() - for compare two strings
(only first n characters)

```
strncmp(s1,s2,n);
```

5) strlen() – for finding length of a string

```
strlen(str);
```

6) strrev()- reverse the string

```
strrev(str);
```

String – strcmp()

```
strcmp(s1,s2);
```

- It returns zero – if s1 and s2 same (s1 == s2)
- It returns less than zero – if s1 less than s2 (s1 < s2)
- It returns greater than zero – if s1 greater than s2 (s1 > s2)

String

Please try to understand the string functions with example …

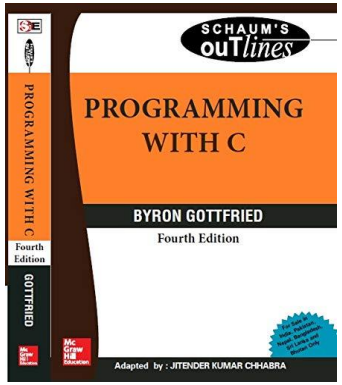
Thank You.

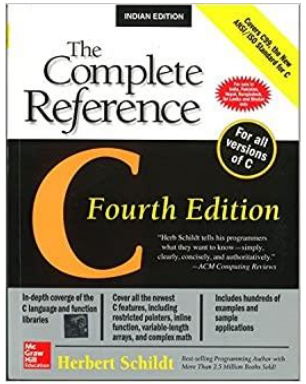
Questions and Answer

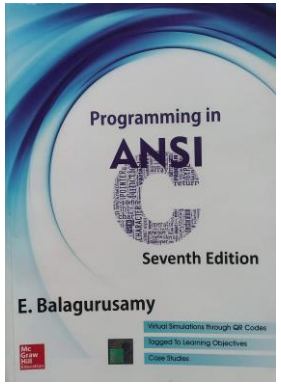
References

Books:

1. Programming With C. *By Byron Gottfried*
2. The Complete Reference C. *By Herbert Shield*
3. Programming in ANSI C *By E. Balagurusamy*
4. Teach yourself C. *By Herbert Shield*







Web:

1. www.wikbooks.org

and other slide, books and web search.