NEXT, we will shift our attention to write **user-defined functions** for **implementing string-handling functions** supported by C language.

BUT keep in mind that we have to make use of pointers in writing function definition of each user-defined string-handling function.

Hence, it would better to discuss pointers and arrays (both Integer array and Character array) again with examples.

Let us declare an integer array **x** as follows –

int **x**[4] = {5, 9, 6, 7};

Now let us assume that the base address of array **x** is 1000 and **each integer** requires **two** bytes, then 4 elements of the array **x** will be stored in the memory as follows –

Elements	x[0]	x[1]	x[2]	x[3]
Value	**5**	**9**	**6**	**7**
Address	1000	1002	1004	1006

Remember that the array name **x** is a **constant pointer** pointing to the first element **x[0]** of the array. So **x** contains the **address** of the **x[0]** i.e. the **base address of the array x**.

Address of x[1] = base address + (1 x scale factor of int) type = 1000 + (1 x 2) = 1002.
Address of x[2] = base address + (2 x scale factor of int) type = 1000 + (2 x 2) = 1004.
Address of x[3] = base address + (3 x scale factor of int) type = 1000 + (3 x 2) = 1006.

[**Note**: **scale factor** of a data type is the **length** of the data type to which a pointer points to. Length of the data type is the number of bytes required to store a variable of that data type.]

Remember **we are not allowed to change the base address of an array**.

Now let us consider the following example to clear our understanding –

```
int main()
{
    int i;
    int  x[4] = {5, 9, 6, 7};
    int  *p;
    p = x;
    for(i = 0; i < 4; i++)
    {
       printf("\nx[%d]  is %d at address %u", i, *p, p);
       p++;
    }
    return(0);
}
```

Explanation

- Declaration of integer variable **i**
- Declaration & initialization of integer array **x**
- Declaration of integer pointer **p**
- Base address of array **x** is assigned to **p** i.e. **p** holds 1000

Itn no.	i	p	*p	Due to p++ content of p becomes
1	0	1000	5	1002
2	1	1002	9	1004
3	2	1004	6	1006
4	3	1006	7	1008
5	4	As i < 4 is false, control comes out of the loop		

OUTPUT:

x[0] is 5 at address 1000
x[1] is 9 at address 1002
x[2] is 6 at address 1004
x[3] is 7 at address 1006

Review Question 1

What will be the output of the following code if we consider base address of x is 1000 and scale factor of int data type is 2.

```
int main()
{
    int i;
    int  x[4] = {5, 9, 6, 7};
    for(i = 0; i < 4; i++)
        printf("\nx[%d]  is %d at address %u", i, *(x+i), x+i);
    return(0);
}
```

Review Question 2
What will be the output of the following code if we consider base address of x is 1000 and scale factor of int data type is 2.

```
int main()
{
      int i;
      int  x[4] = {5, 9, 6, 7};
      for(i = 1; i < 4; i++)
            printf("\nx[%d]  is %d at address %u", i, *(x+i), x+i);
       return(0);
}
```

Review Question 3
What will be the output of the following code if we consider base address of x is 1000 and scale factor of int data type is 2. Explain your answer.

```
int main()
{
      int i;
      int  x[4] = {5, 9, 6, 7};
      x++;
      printf("\n%d  ",  *x);
       return(0);
}
```

==========

Pointers and Strings

We know string can be treated as character array and it can be declared and initialized as follows –

char s[5] = "GOOD";

The compiler **automatically inserts the null character '\0' at the end of the string**. Here **s** is a character array with 5 elements.

If we assume **each character requires 1 byte** and **base address** of the array **s** is **1000**, then we can have the following representation –

Elements	s[0]	s[1]	s[2]	s[3]	s[4]
Value	G	O	O	D	\0
Address	1000	1001	1002	1003	1004

Now observe the following program carefully and try to understand each and every line of the program.

```
#include <stdio.h>
int main()
{
   char s[5] = "GOOD";
   int i = 0;
   while(*(s+i) != '\0')
   {
      printf("\n %c is stored at address  %u", *(s + i),  s + i);
      i++;
   }
   printf("\n Length of string %s is %d", s , i);
   return(0);
}
```

OUTPUT:

```
G is stored at address  1000
O is stored at address  1001
O is stored at address  1002
D  is stored at address 1003
Length of string GOOD is 4
```

Consider base address of array s is 1000 and character requires 1 byte.

Declaration of Character Array S of size 5 and initialization with string GOOD

Itn #	i	s+i	*(s+i)	Value of i due to i++
1	0	1000	G	1
2	1	1001	O	2
3	2	1002	O	3
4	3	1003	D	4
5	4	1004	\0	

Control comes out of the loop as the condition fails.

Review Question 4

What will be the **output** of the following code if we consider base address of **s** is **1000** and scale factor of **char data type is 1**.

```
#include <stdio.h>
int main()
{
        char s[5] = "good";
        char *str = s;
        while(*str != '\0')
        {
                printf("\n %c is stored at address  %u", *str, str);
                str++;
        }
        printf("\n Length of string %s is %d", s, str - s);
        return (0);
}
```

Now let us discuss **an alternative method to create string** using **pointer variable of type char**. C language allows us to declare and initialize string as follows –

char *str = “GOOD”;

This creates a string **GOOD** and then stores its address in the pointer variable **str**. Now the pointer **str** points to the first character of the string **GOOD** as follows –

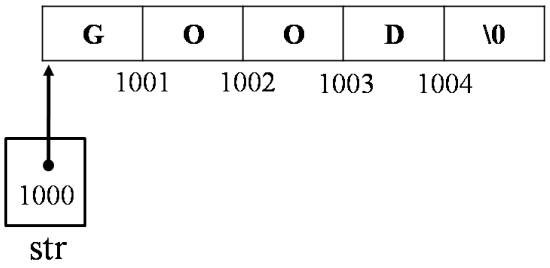

Figure – 1

Review Question 5

Consider the following C program.

```
#include <stdio.h>
int main()
{
        char *str = “GOOD";
        char *name = str;
        while(*str != '\0')
        {
                printf("\n %c is stored at address %u", *str, str);
                str++;
        }
        printf("\n Length of string %s is %d", name, str - name);
        return(0);
}
```

Assume the representation shown in the figure – 1 above, is also applicable for this program. Answer the following questions.

i) Is **str** a pointer to character?

ii) What is the content of **name** when it is initialized?

iii) What will be the contents of **name** and **str** after control comes out of the while loop?

iv) What will be the **output** of the program?