UNIT 4

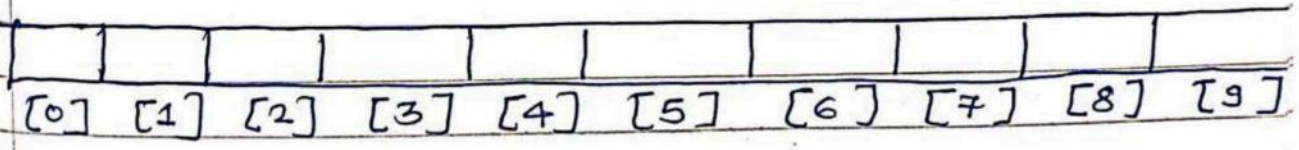
Page No.
Date:

* ARRAYS :-
- It can be used to handle large amount of data at a time.
- It is a group of same data type.
- It is a collection of data that holds fixed number of values of same type.
-
Eg:- Suppose we have to store marks of 50 students One way to do this is alloting 50 variables. So, it will be typical & hard to manage: We cannot access the value of these variable with only 1 or 2 lines of code. But by using array, we can access the elements easily. Only few lines of code is required to access elements of array.

- USES :- - to store list of employee or student names.
- to store marks of students.
- to store list of numbers or characters

* DECLARATION OF AN ARRAY :
- data-type Variable name [size / length of array] ;

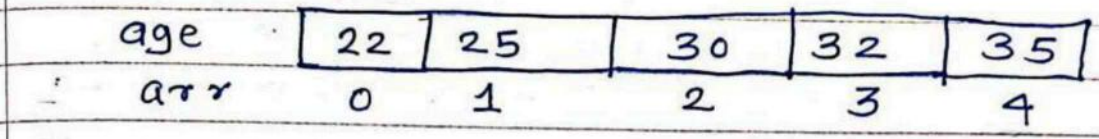
Eg:- int arr [10];



* INITIALIZATION OF AN ARRAY :
1. COMPILE TIME ~~OR RUN TIME~~ : ARRAY INITIALN.
SYNTAX :-
data type array name [size] = {List of values} ;

Page No.
Date:

Eg:- int age [5] = {22, 25, 30, 32, 35}



2. INITIALISING ALL SPECIFIED MEMORY LOCATION

- Same as above
- No. of values to be initialised is equal to size of array.
- Array elements can be initialised with data items of type int, float, char.

Eg:- char b [8] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R'};

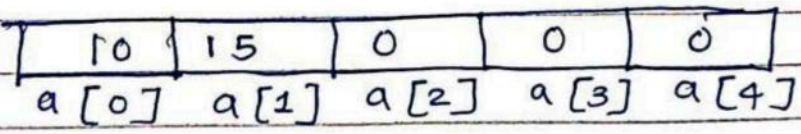
Eg. char b [5] = {'J', 'B', 'R', 'E', 'C', 'B'};

// error as no. of initial values > size of array.

3. PARTIAL ARRAY INITIALISATION :-

- If the number of values to be initialised
- is less than size of array then elements are initialised in the order from 0th location. The remaining locations will be initialised to zero automatically.

- Eg:- int a [5] = {10, 15};



Page No.
Date:

4. RUN TIME ARRAY INITIALISATION:-

- An Array can also be initialised at runtime using scanf() function.
- It is used for initialising large arrays or to initialize array with user specified values.

Page No.
Date:

5. ARRAY INITIALIZATION WITH DECLARATION WITHOUT SIZE :-

- If we initialize an array using an initialiser list we can skip declaring the size of array as compiler can automatically deduce the size of array in these case.

- SYNTAX.

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

6. ARRAY INITIALIZATION AFTER DECLARATION :- [USING LOOPS]

- We can initialize array after declaration by assigning initial value to each element individually.

- SYNTAX :

```
for (int i=0 ; i<N ; i++)
    {
        array_name [i] = value i ;

    }
```

* EXAMPLES :

```
# include <stdio.h>
# include <conio.h>
  Void main ()
{
    int i ;
    int arr [ ] = { 2, 3, 4 } ;
    for ( i=0 ; i<3 ; i++ )
```

```
{
    printf("%d\t", arr[i]);
}
getch();
}
```

- OUTPUT

2 3 4

* PROGRAM TO PRINT ARRAY IN REVERSE ORDER:-

```
#include <stdio.h>
int main()
{
    int array[10] = {1,2,3,4,5,6,7,8,9,0};
    int loop;
    for(loop=9; loop>=0; loop--)
    printf("%d", array[loop]);
    return 0;
}
```

- OUTPUT:-

0 9 8 7 6 5 4 3 2 1

* PROGRAM TO FIND SUM OF ARRAY:-

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

Page No.
Date:

```
int main ( )
{  int array [10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
   int sum , loop;
   sum = 0;
for (loop = 9 ; loop >= 0 ; loop - -)
  {
     sum = sum + array [loop ];
  }
printf ( "sum of array is %d :", sum ) ;
return 0;
}
```

OUTPUT.

sum of array is. 45

Page No.
Date :

* STRINGS :

- The group of characters, digits & symbols enclosed within double quotation (" ") marks are called as string.
- It is an array of characters that is terminated by \0 (null character). This null character indicates end of string.
- Strings are always enclosed by double quotes (" "). whereas character is enclosed by single quotes.

* DECLARATION OF STRING :

- C does not support string as data type. It allows us to represent strings as character arrays.
- String variable is any valid C variable name & it is always declared as an array of characters.

- SYNTAX : Char string_name [size];
- Size determines number of characters in string name. In declaration of string, size must be required to mention otherwise it gives an error.

Eg:-
```
Char Str [ ] ;   // Invalid
Char Str [0] ;   // Invalid
Char Str [-1] ;  // Invalid
Char str [10] ;  // valid
Char a [9] ;     // valid
```

Page No.
Date:

* INITIALIZING ARRAY STRING :-

- SYNTAX

```
char string_name[size] = {"string"};
```

- In Initialization of string if specific number of character is not initialized then that character will be initialized with NULL.

Eg:- `Char str[5] = {'5', '+', 'A'};`

```
str[0]; → 5
str[1]; → +
str[2]; → A
str[3]; → NULL
str[4]; → NULL
```

- In Initialization of string we cannot initialize more than size of string elements.

Eg:- `char str[2] = {'5', '+', 'A', 'B'};`
// Invalid.

* DIFFERENT WAYS OF INITIALIZATION OF STRING:

1. INITIALIZING LOCATION CHARACTER BY CHARACTER

Eg: `char b[9] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R'};`

C	O	M	P	U	T	E	R	\0
0	1	2	3	4	5	6	7	8

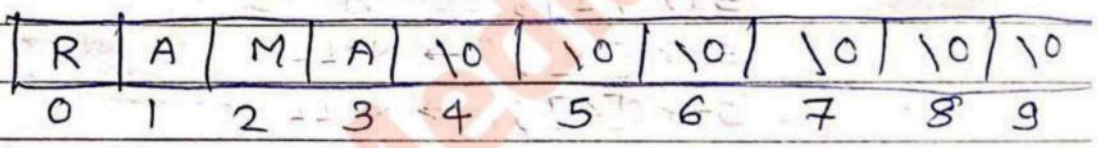
The compiler allocates 9 memory locations ranging from 0 to 8 & these locations are initialized with characters in order specified The remaining locations are automatically

initialized to null characters

2. PARTIAL INITIALIZATION :-

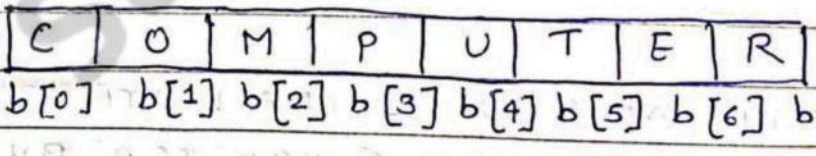
- If the characters to be initialized is less than the size of the string, then characters are stored sequentially from left to right.
- The remaining Locations will be initialized to NULL characters automatically.

Eg: Char a [10] = {'R', 'A', 'M', 'A'}



3. INITIALIZATION WITHOUT SIZE :

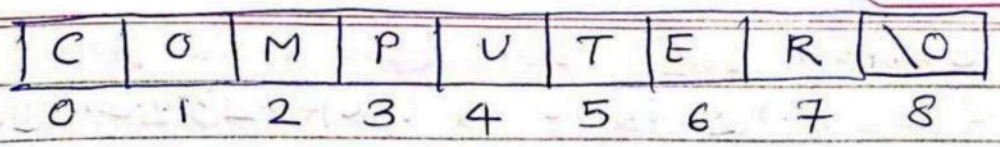
Eg: Char b [ ] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R'};



4. ARRAY INITIALISATION WITH A STRING :

Eg :- Char b [ ] = "COMPUTER";

Here string length is 8 bytes. But string size is 9 bytes. So compiler reserves 8+1 memory Location & these locations are initialized with characters in the order specified. The string is terminated by \0 by the compiler



* READING & WRITING STRINGS :

- The '%s' control string can be used in scanf() statement to read a string from the terminal & same may be used to write string to the terminal in printf() statement.

- Eg: Char name [10];
  scanf ("%s", name);
  printf ("%s", name);

-
```
# include <stdio.h>
void main ()
{
  char ch[13] = {'C', 'P', 'r', 'o', 'g', 'r'
                 'a', 'm', 'm', 'i', 'n',
                 'g', '\0'};

  char ch2 [13] = "Cprogramming";

  printf ("char Array value is : %s \n",
                                  ch);

  printf ("string literal value is : %s \n",
                                  ch2);

}
```

OUTPUT

Char Array value is : cprogramming

String Literal value is : cprogramming

Eg:-

```
#include <stdio.h>
int main()
{
    char name[] = "klinsman";
    int i = 0;

    while (i<=7)
    {
        printf("%c", name[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

OUTPUT

klinsman

Page No.
Date :

* COMPARISON OF TWO STRINGS :

- strcmp() is a built in Library function that is used for string comparison
- This function takes two strings as arguments, compares these two strings & then returns 0, 1 or -1 as result.
- SYNTAX :-

```
strcmp (first_str , second_str) ;
```

- first_str :- First string is taken as pointer to the constant character. i.e immutable string.
- second_str :- second string is taken as pointer to a constant character.

- It returns three different values :-

1. ZERO (0) :-
   - When both strings are found to be identical.
   - All of characters in both strings are same.

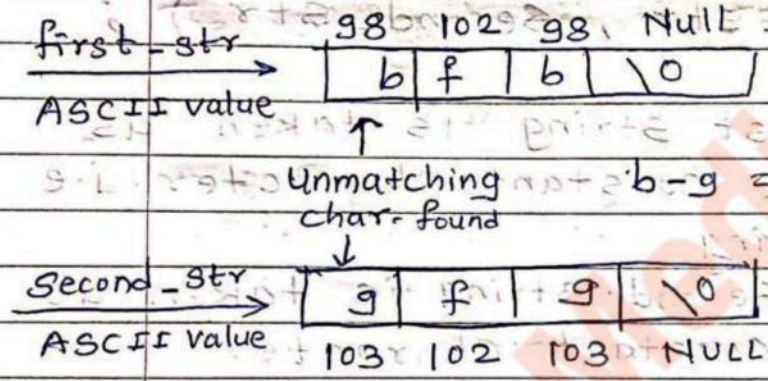
2. GREATER THAN ZERO : (>0)

- When the first not matching character in first_str has a greater ASCII value than Corresponding character in second_str.

- ASCII means American Standard code for Information Interchange. It is used to represent alphanumeric data. i.e 26 upper case

letters A-Z, 26 lower case letters a-z, 10 numerals (0 to 9) & 33 special characters..

3. LESSER THAN ZERO (<0):
- A value less than zero is returned when first non-matching character in first_str has lesser ASCII value than correspondi character in second_str.




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

  int res = strcmp (first_str, second_str);
  if (res == 0)
  printf (" Strings are equal ") ;
  else
  printf (" Strings are Unequal ");
  printf (" \n value returned by strcmp ()
```

```
is : %d", res);
return 0;
}
```

OUTPUT.
strings are equal.
Value returned by strcmp() is : 0.

* STRING HANDLING FUNCTIONS :-

1. strlen(str) :- STRING LENGTH:

- It returns the length of string str.
- It is used to count & return the number of characters present in a string.

- SYNTAX :- var = strlen (string);

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
Void main()
{
    char name[] = "JBREC";
    int len1, len2;

    Clrscr();
    len1 = Strlen (name);
    len2 = Strlen ("JBREC ECE");

printf("The string length of %s is : %d\n",
                        name, len1);
```

```
printf ("The string length of %s is : %d",
        "JBRECECE", len 2);

getch ();

}
```

OUTPUT :

The string length of JBREC is : 5
The string length of JBRECECE is : 8

2] STRING COPY :

- This function is used to copy the contents of one string to another string.
- SYNTAX : strcpy (string 1 , string 2);
  string 1 : is the destination string
  string 2 : is the source string
  i.e contents of string 2 is assigned to contents of string 1.

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

int main ()
{
    Char source [] = "SKNSITS";
    Char dest [20];

    Strcpy (dest , source);

  printf ("Source : %s\n", source);
```

```
    printf (" Destination : %s\n", dest);
    return 0;
}
```

OUTPUT:
Source : SKNSITS
Destination : SKNSITS

3] STRING LOWER CASE :-

- SYNTAX :- strlwr (string);

- This function is used to convert upper case letter of string into lower case letter

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
 void main ( )
{
    Char str [ ] = "GIRISH";
    clrscr ( );

    strlwr (str);

   printf (" The lowercase is : %s\n", str);

    getch ( );
}
```

OUTPUT
The lowercase is : girish

4] STRING UPPERCASE :-

- It is used to convert lower case letters of the string into upper case letters

- SYNTAX :- strupr (string) ;

```
# include <stdio.h>
# include <conio.h>
# include <string.h>
Void main ()
{
Char str [] = "krishna" ;
strupr (str) ;

  printf ("upper case is : %s\n", str);

 getch ();
}
```

OUTPUT :-

upper case is : KRISHNA

5] STRING CONCATENATION :-

- It is used to combine two strings together & form a new concatenated string.

- SYNTAX :- strcat (string 1 , string 2) ;
- When the above function is executed string 2 is combined with string 1 & it removes the null character (\0) of string 1 & places string 2 from there

Page No.
Date:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str1[10] = "TOM";
    char str2[] = "JERRY";

    strcat(str1, str2);
    printf("%s\n", str1);
    printf("%s\n", str2);

    getch();

}
```

OUTPUT :

TOMJERRY

JERRY.

6] STRING REVERSE :.

- It is used to reverse the string.
- It takes only one argument & returns one argument.

SYNTAX :. strrev(string);

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char str[20];
```

```
printf("Enter the string : ");
scanf("%s", str);

printf("The string reversed is : %s",
        strrev(str) );

getch();

}
```

OUTPUT :

Enter the String : GIRISH

The string reversed is : HSIRIG