Unit 5

5.1 Concept of array

Array is a collection of similar type of data type in contiguous memory locations. It allows storing multiple values using a single variable name.

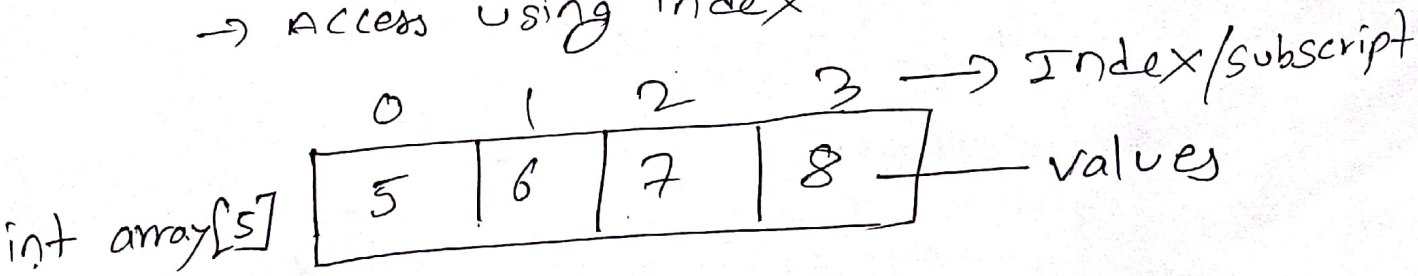
Syntax:

```
datatype array_Name[size];
```

eg

```
int numbers[5];
```

→ Array stores multiple values
→ All elements are same datatype.
→ Stored in continuous memory
→ Access using index



5.2 Array Declaration, Access and Initialization

Array Declaration

```
datatype array_name[size];
```

eg

```
int marks[5];
char name[20];
float price[10];
```

Array Initialization

```
int a[5] = { 10, 20, 30, 40, 50};  → defined array

int a[] = { 10, 20, 30, 40};  → undefined array
```

Compiler automatically counts size.

3. Accessing Array Elements

Syntax

```
array_name[index];
```

eg

```
printf("%d", a[0]); // first element print.
```

Program in c to demonstrate Array

```
#include<stdio.h>
int main()
{
    int a[3] = { 10, 20, 30};
    printf("%d\n", a[0]);
    printf("%d\n", a[1]);
    printf("%d\n", a[2]);
    return 0;
}
```

Output:

```
10
20
30
```

9. Write a C program to sum of all the array elements?

```
#include <stdio.h>
int main()
{
    int a[5];
    int i, sum=0;
    printf("Enter 5 numbers: \n");
    for(i=0; i<5; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0; i<5; i++)
    {
        sum = sum + a[i];
    }
    printf("Sum of array elements is %d",
                sum);
    return 0;
}
```

output:

```
Enter 5 numbers:
1
2
3
4
5
Sum of array elements is 15
```

Types of Array

1. Single Dimensional Array
2. Multi Dimensional Array

(1) Single Dimensional Array :- Array having one dimension is 1D Array.

eg int array[5] = { 1, 2, 3, 4, 5 };

(2) Multi Dimensional Array :- Array having two or more dimensions is 2D / multi Dimensional Array :-

Syntax datatype array_Name [row size] [column size];

eg: int a[3][3];

Write a C program to take user input of matrix elements and display them.

```
#include <stdio.h>
int main()
{
    int a[2][2], i, j;
    printf("Enter elements :\n");
    for(i=0; i<2; i++)
    {
        for(j=0;j<2;j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
```

```
printf("Display matrix:\n");
for(i=0; i<2; i++)
{
    for(j=0; j<2; j++)
    {
        printf("%d\t", a[i][j]);
    }
    printf("\n");
}
return 0;
}
```

Q. Write a C program to add two matrices?

```
#include<stdio.h>
int main()
{
    int i, j, m1[2][2], m2[2][2], sum[2][2];
    printf("Enter first matrix elements:");
    for(i=0; i<2; i++)
    {
        for(j=0; j<2; j++)
        {
            scanf("%d", &m1[i][j]);
        }
    }
    printf("Enter second matrix elements:");
    for(i=0; i<2; i++)
    {
        for(j=0; j<2; j++)
        {
            scanf("%d", &m2[i][j]);
        }
    }
    printf("\n Perform Addition:");
    for(i=0; i<2; i++)
    {
        for(j=0; j<2; j++)
        {
```

~~scanf("%d",~~

~~sum[i][j]~~

```
        sum[i][j] = m1[i][j] + m2[i][j];
        printf("%d\t", sum[i][j]);
    }
    printf("\n");
}
return 0;
}
```

Q. write a c program to perform matrix multiplication for m1[3][4] and m2[4][2].

```
#include<stdio.h>
int m1[3][4], m2[4][2], mul[3][2];
int i, j, k, sum=0;
printf("Enter the values of first matrix:\n");
for(i=0; i<3; i++)
  {for(j=0; j<4; j++)
    {scanf("%d", &m1[i][j]);
    }}
printf("\n Enter the values of second matrix: ");
for(i=0; i<4; i++)
  {for(j=0; j<2; j++)
    {scanf("%d", &m2[i][j]);
    }}
printf("\n perform multiplication: ");
for(i=0; i<3; i++)
  {
```

```
for(j=0; j<2; j++)
{
    sum=0;
    {for(k=0; k<4; k++)
    {
        sum = sum + m1[i][k] * m2[k][j];
    }
    mul[i][j] = sum;
}
}
printf("\nDisplay result:");
for(i=0; i<3; i++)
{for(j=0; j<2; j++)
    {printf("%d", mul[i][j]);
    }printf("\n");
}
return 0;
}
```

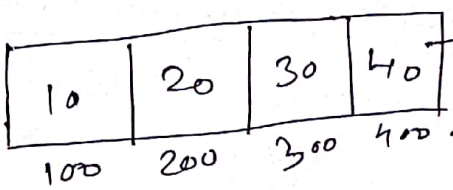
5.4 Concept of pointer

A pointer is a variable that store the address of another variable.

Syntax

```
int x =10;
int *p=&x;
```

let us consider,



print x returns 10
print *p returns 10
print p returns address of x (100)
print &x returns address of x (100)

5.5 pointer Address, Deference, Declaration, Assignment, Initialization.

1. point Declaration

```
datatype *pointer_name;
```

eg

```
int *p;
float *ptr;
```

2. pointer Address operator (&)

used to get the address of variable

Example

```
int x = 10;
printf ("%.p", &x);
```

3. pointer Initialization

```
int x = 20;
int *ptr = &x;
```

where pointer ptr stores address of x.

4. Deference operator (*) :- It is also called Indirection operator.

used to access value stored at address means return value not address.

```
printf ("%.d", *p);
```

Pointer program Example1

```
#include <stdio.h>
int main()
{ int x = 10;
  int *ptr;
  p = &x
```

OR

```
int *ptr = &x;
```

```
printf("Address of x is %p\n", &x);
printf("pointer value is %p\n", p);
printf("value of x is %d\n", *p);
printf("value of x is %d\n", x);
return 0;
}
```

output: Address of x is some memory address value,
pointer value is same memory address value.
value of x is 10
value of x is 10

Q. write a c program to add two numbers using pointer?

```
#include<stdio.h>
int main()
{
int num1, num2, sum;
printf("enter num1 and num2:");
scanf("%d %d", &num1, &num2);
int *p1 = &num1;
int *p2 = &num2;
sum = *p1 + *p2;
printf("sum is %d", sum);
return 0;
}
```

output: enter num1 and num2: 5 10
sum is 15

Q.2 write a c program to swap two numbers using pointers

```
#include <stdio.h>
void swap(int *a, int *b);
Void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
    printf("\nvalues after swap a=%d and
                b=%d", *a, *b);
}
int main()
{
    int p;
    int q;
    printf("Enter p and q values:");
    scanf("%d %d", &p, &q);
    printf("values before swap p=%d and
                q=%d", p, q);
    swap(&p, &q);
    return 0;
}
```

output:

```
Enter p and q values: 20 30
values before swap p=20 and q=30
values after swap a=30 and b=20
```

5.6 pointer Arithmetic

Pointer arithmetic means performing mathematical operations on pointers.

In C programming, pointers store memory addresses, and arithmetic operations allows us to move between memory Locations.

Some pointer arithmetic operations are :-

(i) Increment (++)
(ii) Decrement (--)
(iii) Addition (+)
(iv) Subtraction (-), these operators helps to change the memory Locations,

C program Pointer Arithmetic with Array

```
#include <stdio.h>
int main()
{ int a[5] = { 10, 20, 30, 40, 50};

    int *p = a; // same as int *p = &a[0]
    printf("forward traversal using p++ :\n");
    printf("%d\n" *p); // 10
    p++;
    printf("%d\n" *p); // 20
    p++;
    printf("%d\n" *p); // 30
    printf("moving baekward using p--\n");
    p--;
    printf("%d\n" *p); // 20
```

```
printf("use p+n and p-n:\n");
printf("%d\n", *(p+2)); //40
printf("%d\n", *(p-1)); //10
return 0;
}
```

5. Array and Pointer (Array Pointer)

↳ An array pointer is a pointer that points to an element of an array.

↳ The name of an array acts like a pointer to the first element.

↳ You can use a pointer arithmetic operators to access or traverse array elements.

```
int a[5] = {1,2,3,4,5};
int *p = &a[0];
```

OR

```
int *p = a;
```

where array name acts as the address of the first element. Now p points to the first element.

Program example

```
#include<stdio.h>
int main()
{ int a[2] = {1,2};
  int *p = a;
  printf("%d", *p); //1
  p++;
  printf("%d", *p); //2
  return 0;
}
```

5.8 String in C

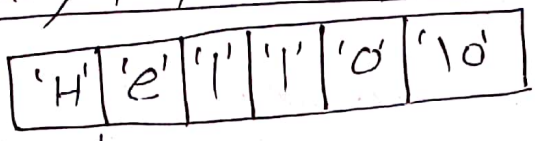
A sequence of characters / Array of characters that are terminated by '\0' is called String.

↳ Terminated by a null character \0.

Example

```
char str[] = "Hello";
```

memory Representation:



Program Example

① Direct

```
#include <stdio.h>
int main()
{
printf("Rajesh");
return 0;
}
```

output: Rajesh

using variable

```
#include<stdio.h>
int main()
{
char name[10]="Rajesh";
printf("%s", name);
}
```

output: Rajesh

① Declaration of string in two ways

① using character Array

```
char str[6]={'H','e','l','l','o',
'\0'};
```

② using string Literal

```
char str[] = "Hello";
```

To get the string from the user input, there is a problem with scanf("%s", ...); when you use:

scanf("%s", name); it stops reading input when it finds a space. for example

Rajesh parajuli

output: Rajesh so, text after the space is not printed.

To get the string from the user input and print a full string with spaces, we use:-

gets() → reads the whole line including spaces.

puts() → prints the string.

Example program

```
#include<stdio.h>
int main()
{
char name[50];
printf("Enter your full name:");
gets(name);
printf("your name is:");
puts(name); // Always on new line
return 0;
}
```

output:

Enter your full name: Rajesh parajuli

your name is:

Rajesh Parajuli

c program to print full line string with spaces.

```
#include <stdio.h>
int main()
{
    char name[50];
    printf("enter your name:");
    scanf("%[^\n]", name);
    printf("your name is %s", name);
    return 0;
}
```

output:

Enter your name: Rajesh parajuli
your name is Rajesh parajuli

String functions in C

String functions are built-in functions used to manipulate strings. use header file string.h.

There are several string functions :-

1. strlen() - find length of string not including '\0' null character.
2. strcpy() - copy one string to another.
3. strcat() - concatenate (join) the strings.
5. strcmp() - compare two strings if equal return 0.
6. strncmp() - compare first n characters
7. strcmpi() - case-insensitive string compare
8. strlwr() - convert string to lowercase
9. strupr() - convert string to uppercase
10. strrev() - reverse the string.

```
#include<stdio.h>
#include<string.h>
  int main()
{
  char str1[50] = "rajesh";
  char str2[50] = "parajuli";
  Char str3[50];
  printf("Length of str1 is %d \n", strlen(str1));
  strcpy(str3, str1);
  printf("copied str1 to str3: %s \n", str3);
  strcat(str1, str2);
  printf("concatenated str1 and str2 : %s \n", str1);
  printf("compare first 3 str1 and str2: %d \n",
        strncmp(str1, str2, 3));
  strrev(str2);
  printf("Reversed str2: %s \n", str2);
  strlwr(str1);
  printf("Lowercase str1: %s \n", str1);
  strupr(str2);
  printf("uppercase str2: %s \n", str2);
  return 0;
}
```

output:

```
Length of str1 is 6
copied str1 to str3 : rajesh
Concatenated str1 and str2 : rajeshparajuli
compare first 3 str1 and str2 : -1
Reversed str2 : iljarap
Lowercase str1 : rajeshparajuli
uppercase str2 : ILUJARAP
```