UNIT-5: Concepts of Arrays and pointer
5.1 Concepts of Single-dimensional Array
5.1.1 Numeric single dimensional Array
5.1.2 Numeric single dimensional array operations:
5.1.2.1 Sorting array in ascending or descending. (Bubble and selection)
5.1.2.2 Searching element from array (Linear Search)
5.1.3 Character Single dimensional Array
5.1.3.1 Character Single dimensional array operations:
5.1.3.2 Use of \0, \n and \t
5.2 Pointers:
5.2.1 Concepts of Pointers
5.2.2 Declaring and initializing int, float, char and void pointers
5.2.3 Pointer to single dimensional numeric array.

UNIT-5: Concepts of Arrays and pointer

5.1 Concepts of Single-dimensional Array

An array is a fixed size sequential group of elements of the same data types which are referred by common name.

There are three types of array in C language.

1.One dimension array

2. Two dimension array

3. Multi dimension array

5.1.1 Numeric single dimensional Array

The list of similar items can be given one variable name using only one subscript is called one dimension array.

Syntax: datatype arrayname[size]

Data type means any data types of c like float, int or char.

Size indicates the maximum number of elements that can be stored inside the array.

e.g. int number[5];

float square[50];

char name[9];

Number[0]=10;

Number[1]=25;

```
Number[2]=44;
Number[3]=2;
Number[4]=51;
```

5.1.2 Numeric single dimensional array operations:
/write a program to read and print 1-dim array.

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int i,a[3];
 clrscr();
 for(i=1;i<=3;i++)
 {
   printf("enter a[%d]=>",i);
   scanf("%d",&a[i]);
 }

 for(i=1;i<=3;i++)
 {
 printf("\n a[%d]=%d",i,a[i]);
 }
 getch();
}
```

Output:

```
Enter a1 1
Enter a2 2
Enter a3 3

a[1]=1
a[2]=2
a[3]=3
```

Initialization of one- dimension arrays

Initialized at compile time:

syntax datatype arrayname[size]={ list of elements };

List of elements must be separated by commas.

e.g. float a[3]={2.5,22,6.3};

float a[3]={2.5};

Remaining 2 elements will be set to zero automatically. And NULL for type character.

e.g.int f[]={1,2,3,4,5};

char n[]={'a','b','c','\0'};

In above e.g. size is blank so compiler allocates enough space for all initialized elements.

Initialized at run time:

This type of initialization is used for large arrays. Like you want to find more than 1000 students total.

e.g. float a;

```
for(a=0; a<1000; a++)
{
        If (a < 500)
        T[a]=0.0;
        Else
         T[a]=5.0;
 }
```

The first 500 elements of array total are initialized to zero while remaining other elements is initialized to 5.0 at run time.

5.1.2.1 Sorting array in ascending or descending. (Bubble and selection)

Array sorting requires data to be sort in ascending or descending order. One method to sort data is bubble sort.

Process is as follow

First compare first and second value in the list if second value is greater than first then interchange two value. This process continues till last value.

Each time interchanging within the array at the end largest element bubble up to top.

Because of this bubble up effect this process is known as bubble sort.

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

```
#include<conio.h>
void main()
{
 int n,a[50],i,j,t;
 clrscr();
 printf("\n Enter any number:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  printf("\n Enter a[%d]=",i);
  scanf("%d",&a[i]);
 }
 for(i=1;i<=n-1;i++)
 {
          for(j=1;j<=n-i;j++)
          {
          if(a[j]<a[j+1])
          {
                  t=a[j];
                  a[j]=a[j+1];
                  a[j+1]=t;
          }
          }
 }
 for(i=1;i<=n;i++)
 {
  //printf("\n sorting variable is:");
   printf("\n a[%d]=%d",i,a[i]);
 }
 getch();
}
```

Output

Enter any number: 3

Enter a[1]=12

Enter a[2]=3

Enter a[3]=30

sorting variable is:

a[1]=30

a[2]=12

a[3]=3

selection sort

1.Find the minimum element in the array and swap it with the element in the 1st position.
2.Find the minimum element again in the remaining array[2, n] and swap it with the element at 2nd position, now we have two elements at their correct positions.
3.We have to do this n-1 times to sort the array.

```
#include <stdio.h>
int main()
{
  int a[100], n, c, d, pos, t;
  clrscr();
  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &a[c]);

  for (c = 0; c < (n - 1); c++) // finding minimum element (n-1) times
  {
    pos = c;
    for (d = c + 1; d < n; d++)
    {
      if (a[pos] > a[d])
          pos = d;
    }
    if (pos != c)
    {
      t = a[c];
      a[c] = a[pos];
      a[pos] = t;
```

```
    }
  }

  printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
    printf("%d\n", a[c]);

  getch();
  return 0;
}
```

5.1.2.2 Searching element from array (Linear Search)

Linear search in C to find whether a number is present in an array. If it's present, then at what location it occurs. It is also known as a sequential search. It is straightforward and works as follows: we compare each element with the element to search until we find it or the list ends.

```
#include <stdio.h>
int main()
{
  int a[100], s, c, n;

  clrscr();
  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d integer(s)\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &a[c]);

  printf("Enter a number to search\n");
  scanf("%d", &s);

  for (c = 0; c < n; c++)
  {
    if (a[c] == s)    /* If required element is found */
    {
      printf("%d is present at location %d.\n", s, c+1);
      break;
    }
  }
  if (c == n)
    printf("%d isn't present in the array.\n", s);

  getch();
  return 0;
```

}

5.1.3 Character Single dimensional Array
5.1.3.1 Character Single dimensional array operations:
5.1.3.2 Use of \0, \n and \t

C does not support string as a data type. It allows us to store string as character array.

Declaration of string

Syntax char variable_name[size];

Variable name is any valid C variable name

Size contain number of character in the string

e.g. char s[20];

When the compiler assigns a character it automatically supplies a null (\0) at the end of string. Therefore size should be plus one.

Declare and initialize character array

e.g char s[12]=”hello world”;

//write a program to remove given character from given string also count that character within string

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

void main()
{
char s[70],r;
int i,c=0;
clrscr();
printf("\n enter string:");
gets(s);
printf("\n enter character to replace:");
scanf("%c",&r);
for(i=0;i<strlen(s);i++)
{
 if(s[i]==r)
 c++;
 else
printf("%c",s[i]);
```

```
}
printf("\n%c appear %d time",r,c);
getch();
}
```

Output:enter string: this is c language
Enter character to replace: g
This is c lanuae
g appear 2 time

Copy string into another string and display character in string

```
#include <stdio.h>
#include <conio.h>
void main()
{
 char s1[20],s2[20];
 int i;
 clrscr();
 printf("enter string:");
 gets(s2);
 for(i=0;s2[i]!='\0';i++)
       s1[i]=s2[i];

 s1[i]='\0';

 puts(s1);
 printf("\n no of character %d",i);
 getch();
}
```

Output
Enter string:ishvari patel
ishvari patel
no of character 13

5.2 Pointers:

5.2.1 Concepts of Pointers

A pointer is a derived data type in c. it is built from one of the fundamental data types available in c. Pointers contain memory address as their value.

& sign is used to return address of variable. It is known as address of operator.

* sign is used to return value at specified address and it is known as a pointer operator or indirection operator or value operator.

5.2.2 Declaring and initializing int, float, char and void pointers

Declaration of pointer variable

```
Datatype *pt_name;
```

The * indicates variable pt_name is pointer variable. Pt_name points to variable of datatype

Example
```
int *p;
char *c;
float *f;
void *ptr;
```

The process of assigning address of a variable to a pointer variable is known as initialization.

Example
```
int *p,a;
p=&a;
```

A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.

```
int x;
char b='C';
ptr=&x;
ptr=&b;
```

5.2.3 Pointer to single dimensional numeric array.

When array is declare compile allocate elements of array in contiguous memory location.

The base address is the location of the first element of array. The compiler also defines the array name as a constant pointer to the first element. We declare one integer pointer and assigned first element address to pointer variable.

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

Suppose base address of x is 1000 and each integer require two bytes so memory representation as follows.

X[0]	x[1]	x[2]	x[3]	x[4]

1	2	3	4	5
1000	1002	1004	1006	1008

We can access every value of x using p++ to move from one element to another.

p=&x[0] or p=x; (1000)
p+1 =&x[1](1002)
p+2 =&x[2](1004)
p+3 =&x[3](1006)
p+4 =&x[4](1008)

The address of an element is calculated using its index and the scale factor of the data type.
e.g.
address of x[3]= base address +(index x scale factor of datatype)
= 1000+(3 x 2)
=1000+6
=1006

We can use pointer to access array elements. Instead of writing x[3] we use *(p+3)
So in general, we use *(p+i) instead x[i].

Write a program to take array element and display elements using pointer. Also display sum of elements in array.

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a[5],*p,i,n,*s,sum=0;
 p=a;
 s=a;
 clrscr();
 printf("enter value of n:");
 scanf("%d",&n);

 for(i=0;i<n;i++)
```

```
{
printf("\n a[%d]=",i+1);
 scanf("%d",s+i);
 sum=sum+*(s+i);
 }

 for(i=0;i<n;i++)
 {
 printf("\n a[%d]=",i+1);
 printf("%d",*(p+i));
}
 printf("\n sum is %d",sum);
 getch();
}
```

Output

```
Enter value of n:3
A[1]=10
A[2]=11
A[3]=12

A[1]=10
A[2]=11
A[3]=12
Sum is 33
```