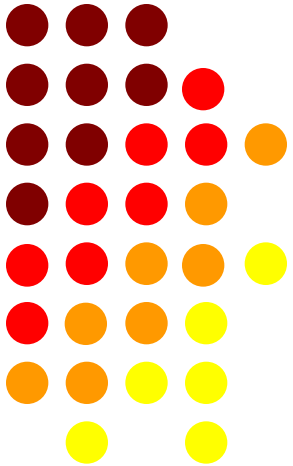
Programming For Problem Solving

GROUP OF INSTITUTIONS

Lecture 17



Looping Control Instructions

- The statement which are used to execute statement or set of statements repeatedly until a specific condition is satisfies is known as looping statement/instruction.

Type of looping control instructions

- for statement
- while statement
- do while statement

Looping Control Instructions

- **for statement:** It is used to repeat the block of code, on basis of some specific condition.

Syntax:

```
        (1)          (2)          (3)
for(initialization ; condition ; inc/dec)
{
   true block statement;    (4)
}
   statement x:     (5)
```

Order Of Execution

T T F

1 2 4 3 2 4 3 2 5(out of the loop)

Looping Control Instructions

/*Print the character ‘a’ 10 times*/

```
int i;

for(i=1 ; i<=10 ; i=i+1)

{

printf(“a”);

}
```

Output

aaaaaaaaaa

/*Print the counting 1 to 10*/

```
int i;

for(i=1 ; i<=10 ; i=i+1)

{

printf(“%d”, i);

}
```

Output

12345678910

Example-1 (for loop)

```
/*WAP to find factorial of a number*/
#include<stdio.h>
#include<conio.h>
void main()
{
 int n, f=1, i;
 clrscr();
 printf("\n Enter Number: \n");
 scanf("%d", &n);
 for (i=n; i>1; i=i-1)
 {
 f = f * i;
 }
 printf(" Factorial of %d=%d", n, f);
 getch();
 }
```

Example-2 (for loop)

/*WAP to construct a Fibonacci Series*/

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n, a=-1, b=1, c, i;
clrscr();
printf("\n Enter number of terms: \n");
scanf("%d", &n);
for(i=1 ; i<=n ; i++)
{
c = a + b;
printf("%d\t", c);
a = b;
b = c;
}
getch();
}
```


Example-3 (for loop)

/*WAP to calculate sum of Fibonacci Series*/

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n, a=-1, b=1, c, i, s=0;
clrscr();
printf("\n Enter number of terms: \n");
scanf("%d", &n);
for(i=1 ; i<=n ; i++)
{
c = a + b;
s = s + c;
a = b;
b = c;
}
printf("Sum of Series=%d", s);
getch();
}
```

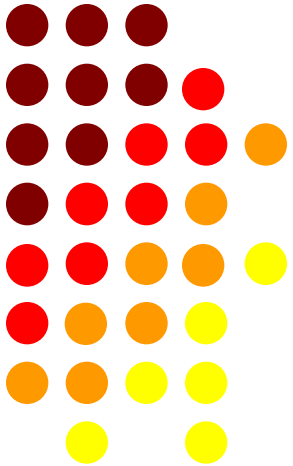

Example-4 (for loop)

```
/*WAP to check number is prime or not*/
#include<stdio.h>
#include<conio.h>
void main()
{
int n, i, j=0;
clrscr();
printf("\n Enter Number \n");
scanf("%d", &n);
for(i=1 ; i<=n ; i=i+1)
{
if ( n % i == 0)
      j = j + 1;
}
if (j == 2)
printf("\n Prime No.\n");
else
printf("\n Not  Prime \n");
getch();
}
```

Programming For Problem Solving

GROUP OF INSTITUTIONS

Lecture 18



Looping Control Instructions

- **while statement:** It is used to repeat the block of code, on basis of some specific condition.

Syntax:

```
initialization     (1)
while(condition)     (2)
{
   true block statement;    (4)
   increment/decrement;   (3)
}
   statement x:      (5)
```

Order Of Execution

T T F

1 2 4 3 2 4 3 2 5(out of the loop)

Looping Control Instructions

/*Print the character ‘a’ 10 times*/

```
int i=1;

while(i<=10)

{

printf(“a”);

i = i + 1;

}
```

Output

aaaaaaaaaa

/*Print the counting 1 to 10*/

```
int i=1;

while(i<=10)

{

printf(“%d”, i);

i = i + 1;

}
```

Output

12345678910

Example-5 (while loop)

```
/*WAP to find the sum of digits*/
#include<stdio.h>
#include<conio.h>
void main()
{
int  n, d, s=0, m;
clrscr();
printf("\n Enter Number: \n");
scanf("%d", &n);
m=n;
while (m > 0)
{
d = m % 10;
s = s + d;
m = m / 10;
}
printf(" Sum of %d=%d", n, s);
getch();
}
```

Example-6 (while loop)

```
/*WAP to find the reverse of digits*/
#include<stdio.h>
#include<conio.h>
void main()
{
int  n, d, r=0, m;
clrscr();
printf("\n Enter Number: \n");
scanf("%d", &n);
m=n;
while (m > 0)
{
d = m % 10;
r = r * 10 + d;
m = m / 10;
}
printf(" Sum of %d=%d", n, r);
getch();
}
```

Example-7 (while loop)

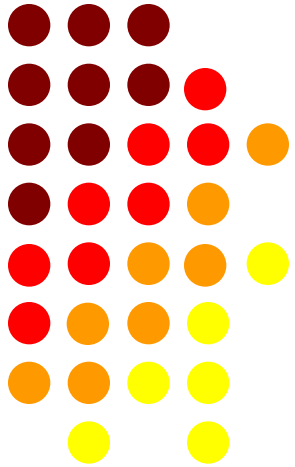
```
/*WAP to print Armstrong or not*/
#include<stdio.h>
#include<conio.h>
void main()
{
int  n, d, s=0, m;
clrscr();
printf("\n Enter Number: \n");
scanf("%d", &n);
m=n;
while (m > 0)
{
d = m % 10;
s = s + d * d * d;
m = m / 10;
}
if(n == s)
printf("\n Armstrong \n");
else
printf("\n Not Armstrong \n");
getch();
}
```


Example-8 (while loop)

```
/*WAP to print Palindrome or not*/
#include<stdio.h>
#include<conio.h>
void main()
{
int  n, d, r=0, m;
clrscr();
printf("\n Enter Number: \n");
scanf("%d", &n);
m=n;
while (m > 0)
{
d = m % 10;
r = r * 10 + d;
m = m / 10;
}
if(n == r)
printf("\n Palindrome \n");
else
printf("\n Not Palindrome \n");
getch();
}
```

Programming For Problem Solving

Lecture 19



DO WHILE LOOPING CONTROL INSTRUCTION

DO WHILE STATEMENT

Syntax-> initialization; //1

```
do
{
 true block st ; //4
 inc/dec ;//3
} while(condition);//2
   statement x;//5
```

Order Of Execution

T F

1 4 3 2 4 3 2 5

EXAMPLE:-

```
int I =1;
 do
{  printf( " a");
     i=i+1;
} while(i<=10);
```


17

While Vs Do While Statement

while	do while
It is an example of entry control loop.	It is an example of exit control loop.
In while the condition is checked first & then block is executed.	In do while the block is executed first & then the condition is checked.
In while if condition is true for n times, the block will execute n times.	In do while if condition is true for n times, the block will execute n+1 times.

Syntax:
initialization (1)
while(condition) (2)
{
true block statement; (4)
increment/decrement; (3)
}
statement x: (5)
Order Of Execution
T T F
1 2 4 3 2 4 3 2 5
	Syntax:
	initialization (1)
	do
	{
	true block statement; (4)
	increment/decrement; (3)
	} while(condition) (2);
	statement x: (5)
	Order Of Execution
	T F
	1 4 3 2 4 3 2 5

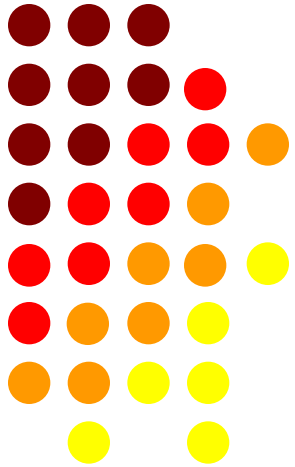
Break Vs Continue Statement

break	continue
It is used to come out of the loop.	It is used to come in the beginning of the loop.
It will skip all the coming rounds after its execution.	It will skip only the current rounds after its execution.
It is used to come out of switch statement.	It is not used to reach beginning of switch statement.

int i;
for(i=1 ; i<=3 ; i++)
{
if(i==2)
break;
printf("%d", i);
}
 	
Output
1
	int i;
	for(i=1 ; i<=3 ; i++)
	{
	if(i==2)
	continue;
	printf("%d", i);
	}
	
	Output
	1 3

Programming For Problem Solving

Lecture 20



NESTING OF LOOP

- When one loop is placed in block of another loop & the iteration of inner loop is based outer loop. It is termed as nested of loop…..Like
- int i,j;

  1 2 3 order of execution
- For(i=1;i<=3;i=i++) // T T T F T T T F
- { 4 5 6 // 1 2 4 5 7 6 5 7 6 5 8 3 2 4 5------5 8 3 2 9
- For(j=1;j<=3;j=j++)
- { printf("%d",j);//7
- }
- printf("%d",i);//8
- }9

21

Example-9 (nested for loop)

/*WAP to print the sum of series 1! + 2!+ n!*/

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int   n, j, i, f, s=0;
 clrscr();
 printf (“\n Enter Number of Terms: \n”);
 scanf(“%d”, &n);
 for( j=1; j<=n; j=j+1)
 {
 f=1;
for (i=j; i>1; i=i-1)
{
f = f * i;
}
s = s + f;
}
printf(“ Sum of Series=%d”, s);
getch();
}
```

Example-10 (nested for loop)

```
/*WAP to print the sum of series x^1 / 1!
+ x^2 / 2!+    x^n / n!*/
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
 int   n, j, i, f, s=0, x=2;
 clrscr();
 printf (“\n Enter Number of Terms: \n”);
 scanf(“%d”, &n);
 for( j=1; j<=n; j=j+1)
 {
 f=1;
for (i=j; i>1; i=i-1)
{
f = f * i;
}
s = s + pow(x, j) / f;
}
printf(“ Sum of Series=%d”, s);
getch();
}
```

Example-11 (nested for loop)

```
/*WAP to print the sum of series -1^4 +

3^4 – 5^4 + ------*/

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

 int   n, s=0, i, j=1;

 clrscr();

 printf (“\n Enter Number of Terms: \n”);

 scanf(“%d”, &n);

 for( i=1; i<=(2*n-1); i=i+2)

 {

s = s + pow(i, 4) * pow(-1,j);

j++;

}

printf(“ Sum of Series=%d”, s);

getch();

}
```

Example-12 (nested for loop)

/*WAP to print the following pattern*/

@

@ @

@ @ @

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int   n, i, j;
 clrscr();
 printf (“\n Enter Number of Rows: \n”);
 scanf(“%d”, &n);
for(i=1; i<=n; i=i+1)
{
 for( j=1; j<=i; j=j+1)
 {
  printf(“ @ “);
 }
 printf (“\n”);
}
getch();
}
```

Example-13 (nested for loop)

/*WAP to print the following pattern*/

1

1 2

1 2 3

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int   n, i, j;
 clrscr();
 printf (“\n Enter Number of Rows: \n”);
 scanf(“%d”, &n);
for(i=1; i<=n; i=i+1)
{
 for( j=1; j<=i; j=j+1)
 {
  printf(“%d“, j);
 }
 printf (“\n”);
}
getch();
}
```

Example-14 (nested for loop)

/*WAP to print the following pattern*/

a

a b

a b c

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int   n, i, j;
 clrscr();
 printf (“\n Enter Number of Rows: \n”);
 scanf(“%d”, &n);
for(i=1; i<=n; i=i+1)
{
 for( j=97; j<=96+i; j=j+1)
 {
  printf(“%c“, j);
 }
 printf (“\n”);
}
getch();
}
```


Example-15 (nested for loop)

/*WAP to print the following pattern*/

```
        *
    *   *
*   *   *
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int   n, i, j, k;
 clrscr();
 printf (“\n Enter Number of Rows: \n”);
 scanf(“%d”, &n);
for(i=1; i<=n; i=i+1)
{
 for( k=1; k<=n-i; k=k+1)
 {
  printf(“  “);
 }
 for( j=1; j<=i; j=j+1)
 {
  printf(“ * “);
 }
 printf (“\n”);
}
getch();
}
```

Example-16 (nested for loop)

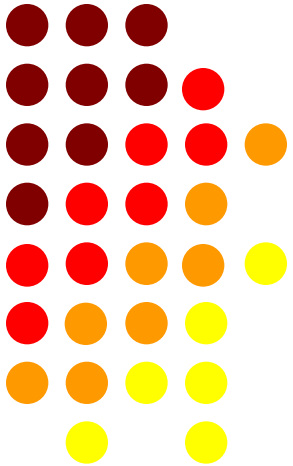
/*WAP to print the following pattern*/

```
* * * *
* * *
* *
*
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int   n, i, j, k;
 clrscr();
 printf (“\n Enter Number of Rows: \n”);
 scanf(“%d”, &n);
for(i=4; i>=1; i=i-1)
{
for( j=1; j<=i; j=j+1)
 {
  printf(“* “);
 }
 printf (“\n”);
}
getch();
}
```

Programming For Problem Solving

LECTURE-21



ARRAY

An array is a collection of similar data type elements stored in contiguous memory locations.

An array is also termed as subscript variable. Here first element is stored as 0th location.

Till the array is not initialized it contains garbage value while if partially initialized it contains zero.

One of the limitation of array is, it does not provide bound checking.

Array name is base address of array (base address is the address of first element of array). And You cannot change the base address of array.

Application of Array

Searching- It is the process of finding the location of specific element in an array of n elements. e.g. linear search, binary search

Sorting- It is the process of arranging the elements in specific order in an array of n elements. e.g. bubble sort

CLASSIFICATION OF ARRAY

1-D Array : A one-dimensional array (or single dimension array) is a type of linear array. Accessing its elements involves a single subscript which can either represent a row or column index

2-D Array : An array of arrays is known as 2D array. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns.

Multi-dimensional Array : C allows for arrays of two or more dimensions. A two-dimensional (2D) array is an array of arrays. A three-dimensional (3D) array is an array of arrays of arrays. In C programming an array can have two, three, or even ten or more dimensions.

1-D ARRAY

An array having one subscript or dimension is termed as 1-D Array.

Declaration of Array :

int a[5]; here, 5 represents number of elements.

Initialization of Array:

int a[5]={1,12,33,40,5,};// so the value of a[0]=1,a[1]=12,a[2]=33,a[3]=40,a[4]=5

Recieving array elements

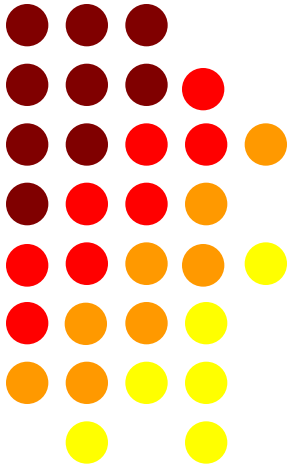
```
int a[5],i;
for(i=0;i<5;i++)
{
 printf(“enter value”);
scanf(“%d”,&a[i]);
}
```

Printing array elements

```
int a[5],i;
for(i=0;i<5;i++)
printf(“%d\t”,a[i]) ;
```


Programming For Problem Solving

LECTURE-22



1-D ARRAY(CONT…)

/* WAP to find largest element in one dimensional array */

```
#include<stdio.h>
int main()
{
int a[100];
int i,lar,n;
clrscr();
printf("enter size of array");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter value");
scanf("%d",&a[i]);
}
lar=a[0];
for(i=0;i<n;i++)
{
if(a[i]>lar)
lar=a[i];
 }
printf("%d is largest",lar);
return 0;
}
```

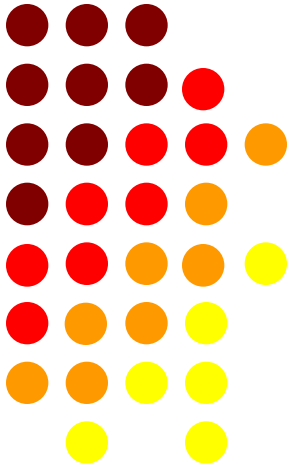

1-D ARRAY (CONT…)

/* WAP to find average of n numbers in one dimensional array */

```
#include<stdio.h>
int main()
{
int a[100];
int i,,n;
float s=0,avg;
clrscr();
printf("enter size of array");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter value");
scanf("%d",&a[i]);
s=s+a[i];
}
avg=s/n;
printf(“average = %f “,avg);
return 0;
}
```

Programming For Problem Solving

LECTURE-23



2-D ARRAY

An array having two subscripts or dimensions, one for row and other for column, is termed as two dimensional array or matrix.

Declaration of Array

int a[3][3];

where first size shows rows and second shows cloumns.

Initialization of Array

int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};//or int a[3][3]={1,2,3,4,5,6,7,8,9};

The elements of each row is surrounded by curly braces.

2-D ARRAY (CONT…)

Recieving array elements

```
int a[3][3],i,j;
for(i=0;i<3;i++)
{
     for(j=0;j<3;j++)
    {
        printf(“enter value”);
        scanf(“%d”,&a[i][j]);
        }
}
```


2-D ARRAY (CONT…)

Printing array elements

```
int a[3][3],i,j;
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
     {
          printf("%d",a[i][j]);
       }
}
```

2-D ARRAY (CONT…)

/* Write a program to find transpose of a matrix. */

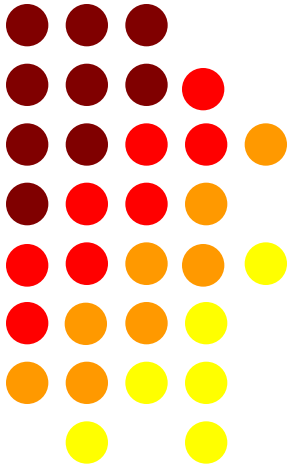
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5],b[5][5],i,j,m,n;
clrscr();
printf("enter the order of First matrix");
scanf("%d%d",&m,&n);
for(i=0;i<m;i++)
  {
      for(j=0;j<n;j++)
      {
         printf("enter First matrix");
         scanf("%d",&a[i][j]);
         b[j][i]=a[i][j];
        }
}
```

2-D ARRAY (CONT…)

```
 printf("\ntranspose of matrix is:\n");
for(i=0;i<n;i++)
  {
       for(j=0;j<m;j++)
       {
          printf("%d\t",b[i][j]);
          }
printf("\n");
}
getch();
}
```


Programming For Problem Solving

LECTURE-24



2-D ARRAY (CONT…)

GROUP OF INSTITUTIONS

/* PROGRAM TO PERFORM ADDITION OF TWO MATRIX. */

```
#include<stdio.h>
#include<conio.h>
void main()
{      int a[3][3],b[3][3],c[3][3],m,n,i,j;
       clrscr();
printf("\n enter order of matrix \n");
scanf("%d %d",&m,&n);
for(i=0;i<m;i++)
{
  for(j=0;j<n;j++)
     {    printf("enter value in first ");
          scanf("%d",&a[i][j]);
      }
}
```

2-D ARRAY (CONT…)

```
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
        {    printf(“enter value in second ”);
             scanf(“%d”,&b[i][j]);
              c[i][j]= a[i][j]+ b[i][j];
         }
   }

for(i=0;i<m;i++)
{
              for(j=0;j<n;j++)
              {
                  printf(“%d\t”,c[i][j]);
              }
     printf(“\n”);
}
getch();
}
```

2-D ARRAY (CONT…)

/* Write a program to find sum of all elements of 2D array */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5],i,j,m,n,s=0;
clrscr();
printf("enter the order of First matrix");
scanf("%d%d",&m,&n);
for(i=0;i<m;i++)
  {
      for(j=0;j<n;j++)
      {
         printf("enter First matrix");
         scanf("%d",&a[i][j]);
         s=s+a[i][j];
        }
}
   printf("\nsum=%d",s);
getch();
}
```

2-D ARRAY (CONT…)

/* Write a program to find sum of diagonal elements of 2D array */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5],i,j,m,n,s=0;
clrscr();
printf("enter the order of matrix");
scanf("%d%d",&m,&n);
for(i=0;i<m;i++)
  {
      for(j=0;j<n;j++)
      {
         printf("enter value in  matrix");
         scanf("%d",&a[i][j]);
        if(i==j)
         s=s+a[i][j];
        }
}
    printf("\nsum=%d",s);
getch();
}
```

2-D ARRAY (CONT…)

/* Write a program to multiply two matrices */

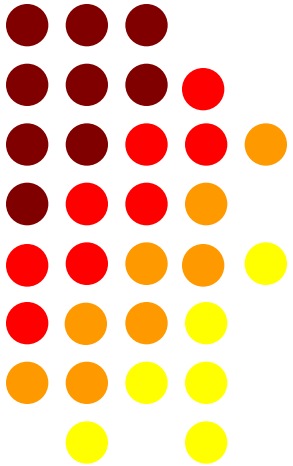
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5],b[5][5],c[5][5],i,j,k,m,n,p,q;
clrscr();
printf("enter the order of First matrix");
scanf("%d%d",&m,&n);
printf("enter the order of Second matrix");
scanf("%d%d",&p,&q);
if(n!=p)
  {   printf("Multiplication Not possible");
   }
else
{
for(i=0;i<m;i++)
  {     for(j=0;j<n;j++)
      {
              printf("enter First matrix");
              scanf("%d",&a[i][j]);
        }
    }
```


2-D ARRAY (CONT…)

```
for(i=0;i<p;i++)
{
      for(j=0;j<q;j++)
       {
            printf("\nenter Second matrix");
             scanf("%d",&b[i][j]);
         }
 }
 for(i=0;i<m;i++)
{
        for(j=0;j<n;j++)
           {     c[i][j]=0;
               for(k=0;k<n;k++)
                  c[i][j]=c[i][j]+a[i][k]*b[k][j];
            }
}
    for(i=0;i<m;i++)
{
       for(j=0;j<q;j++)
        {
            printf("%d\t",c[i][j]);
          }
    printf("\n");
    }
 }
getch();
  }
```

Programming For Problem Solving

LECTURE-25



PASSING ARRAY TO FUNCTION

There are three methods to pass an array as argument to function:

Method 1 :

```
#include <stdio.h>
void print(int []);
int main()
{
    int a[5],i;
   for(i=0;i<5;i++)
    {
        printf("enter value");
        scanf("%d",&a[i]);
    }
    print(a);
   return 0;
}
void print(int b[5])
{
    int i;
    for(i=0;i<5;i++)
    {
        printf("%d\t",b[i]);
    }
}
```

PASSING ARRAY TO FUNCTION (CONT…)

Method 2:

```
#include <stdio.h>
void show(int [],int);
int main()
{
    int a[20],n,i;
    printf("enter size of array");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter value");
        scanf("%d",&a[i]);
    }
    show(a,n);
     return 0;
}
void show(int b[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d\t",b[i]);
    }
}
```

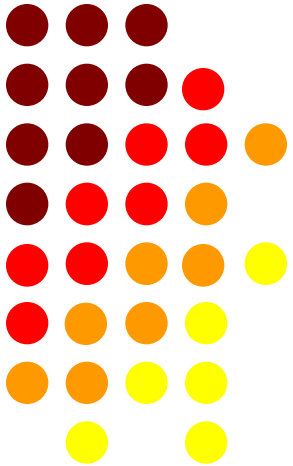
PASSING ARRAY TO FUNCTION (CONT…)

Method 3:

```
#include <stdio.h>
void display(int *,int);
int main()
{
    int a[20],n,i;
    printf("enter size of array");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter value");
        scanf("%d",&a[i]);
    }
display(a,n);
return 0;
}
void display(int *p,int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d\t",*(p+i));
    }
}
```

Programming For Problem Solving

LECTURE-26



STRING

A character type array is termed as string which is terminated by null character. Also the collection of characters (alphanumeric) enclosed in double quote and terminate by null character is known as string.

Each character occupy 1 byte and stored at contiguous memory location, start from 0^{th} location.

Size of string should always exceed by one while declaration of string.

String Declaration

char s[10];

where s is the name of string which can store 10 values of character

type (9 character + 1 null character)

Initialization of string

char s[10]= {‘h’,’e’,’l’,’l’,’o’,’\o’};

char s[10]= “hello”;

STRING (CONT…)

Receiving string elements

scanf(“%s”,s) --> for single word

gets(s); --> for multiple word

Printing string elements

printf(“%s”,s) .

puts(s).

String library function header file

#include<string.h>

gets V/S scanf AND puts V/S printf

gets()	scanf()
Only for string.	For all data type(int ,float ,char)
For single or multi word string	For single word string
Input only one string at a time.	Input one or more strings at a time
Unformatted function	Formatted function
puts()	**printf()**
Only for string.	For all data type(int ,float ,char)
print only one string at a time.	print one or more strings at a time
After print cursor goes to next line.	After print cursorremains same line.
Unformatted function	Formatted function

STRING LIBRARY FUNCTIONS

strlen(s)- it counts number of character present in a string s.

```
 void main()
{
char s[10];
int l;
clrscr();
printf(“enter string”);
gets(s);
l=strlen(s);
printf(“string length=%d”,l);
getch();
}
```

STRING LIBRARY FUNCTIONS (CONT…)

strcpy(s2,s1)- it copies the content of string s1 into string s2.

```
 void main()
{
char s1[10],s2[10];
clrscr();
printf("enter string");
gets(s1);
strcpy(s2,s1);
printf("\n string copied is \n");
puts(s2);
getch();
}
```


STRING LIBRARY FUNCTIONS (CONT…)

strcat(s2,s1)- it joins the content of string s1 into string s2.

```
 void main()
{
char s1[10],s2[10];
clrscr();
printf("enter string 1 \n");
gets(s1);
printf("\n enter string 2 \n");
gets(s2);
strcat(s2,s1);
printf("\n string after join \n");
puts(s2);
getch();
}
```

STRING LIBRARY FUNCTIONS (CONT…)

strcmp(s2,s1)- this function is used to compare two strings to find out whether they are same or different. The strings are compared character by character until there is a mismatch or null is found.

If the strings are same it will return zero otherwise it will return the difference of ASCII value of characters.

```
void main()
{
char s1[10]=”pull”, s2[10]=”push”;
int c;
clrscr();
c=strcmp(s2,s1);
if(c==0)
printf(“same strings”);
else
printf(“different strings”);
getch();
}
```


STRING LIBRARY FUNCTIONS (CONT…)

strrev(s)- this function is used to reverse the string.

```
 void main()
{
char s[10];
clrscr();
printf(“enter string”);
gets(s);
strrev(s);
printf(“string reverse is \n”);
puts(s);
getch();
}
```

STRING LIBRARY FUNCTIONS (CONT…)

/* WAP to check whether a given string is palindrome or not */

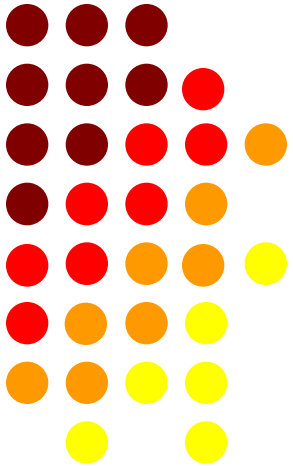
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char s1[10],s2[10];
clrscr();
printf(“enter string \n”);
gets(s1);
strcpy(s2,s1);
strrev(s2);
if(strcmp(s2,s1)==0)
{
```

STRING LIBRARY FUNCTIONS (CONT…)

```
printf("\n string palindrome \n");
}
else
printf("\n not palindrome \n");
getch();
}
```

Programming For Problem Solving

LECTURE-27



STRING USER FUNCTIONS

```
/* program to find length of  the string using user defined function */
#include<stdio.h>
#include<conio.h>
int strlen1(char *);
void main()
{
char name[100];
int l;
clrscr();
printf("enter a string \n");
gets(name);
l=strlen1(name);
printf("length of string is : %d",l);
getch();
}
```


STRING USER FUNCTIONS (CONT…)

```
int strlen1(char *p)
{
     int i=0;
     while(*p!='\0')
    {
        i++;
        p++;
    }
   return(i);
}
```

STRING USER FUNCTIONS (CONT…)

/* Program to copy string without using library function */

```
#include<conio.h>
#include<stdio.h>
void strcpy1(char *,char *);
void main()
{
char str1[50],str2[20];
clrscr();
puts("enter string");
gets(str1);
strcpy1(str2,str1);
printf("after copy string2 is\n");
puts(str2);
getch();
}
```


STRING USER FUNCTIONS (CONT…)

```
  void strcpy1(char *p,char *q)
{
   while(*q!='\0')
  {
  *p=*q;
  p++;
  q++;
  }
   *p='\0';
 }
```

STRING USER FUNCTIONS (CONT…)

/* Program for concatenate two strings using pointers. */

```
#include<conio.h>
#include<stdio.h>
void strcon(char *,char *);
void main()
{
char str1[50],str2[20];
clrscr();
puts("enter string");
gets(str1);
printf("enter second string");
gets(str2);
strcon(str1,str2);
printf("after concatenate string is\n");
puts(str1);
getch();
}
```


STRING USER FUNCTIONS (CONT…)

```
 void strcon(char *p,char *q)
{
   while(*p!='\0')
   p++;

  while(*q!='\0')
  {
  *p=*q;
  p++;
  q++;
  }
   *p='\0';
 }
```

STRING USER FUNCTIONS (CONT…)

/* Program to compare two strings without using library function. */

```
#include <stdio.h>
int strcmp1(char *,char *);
void main()
{
char str1[50],str2[20];
int c;
clrscr( );
puts("enter string1");
scanf("%s",&str1);
puts("enter string12");
scanf("%s",&str2);
c=strcmp1(str1,str2);
if(c==0)
printf("both are same");
else
printf("both are not same");
}
```

STRING USER FUNCTIONS (CONT…)

```
int strcmp1(char *p,char *q)
{
    while((*p!='\0')||(*q!='\0'))
  {
  if(*p!=*q)
  return(*p-*q);
  p++;
  q++;
  }
   return(0);
 }
```

STRING USER FUNCTIONS (CONT…)

/* Program to reverse a string using pointer. */

```
#include<stdio.h>
#include<conio.h>
void strrev1(char *);
void main()
{
char name[100];
int l;
clrscr();
printf("enter a string \n");
gets(name);
strrev1(name);
printf("After reverse string is :\n");
puts(name);
getch();
}
```

STRING USER FUNCTIONS (CONT…)

```
void strrev1(char *s)
{
     int i=0;
    char *l,t;
     l=s;
    while(*l!='\0')
    {
        i++;
        l++;
    }
     l--;
     i=i/2;
while(i>0)
   {
        t=*s;
        *s=*l;
        *l=t;
        s++;
        l--;
       i--;
    }
}
```

STRING USER FUNCTIONS (CONT…)

/* Program to check whether **a string is palindrome or not using pointer. */**

```
#include<stdio.h>
#include<conio.h>
int strpal(char *);
void main()
{
char name[100];
int c;
int l;
clrscr( );
printf("enter a string \n");
scanf("%s",&name);
c=strpal(name);
if(c==0)
printf(" %s is palindrome",name);
else
printf(" %s is not palindrome",name);
getch( );
}
```

STRING USER FUNCTIONS (CONT…)

```
int strpal(char *s)
{
     int i=0;
    char *l;
     l=s;
    while(*l!='\0')
    {
        i++;
        l++;
    }
     l--;
     i=i/2;
while(i>0)
   {
        if(*s!=*l)
        return(1);
        s++;
        l--;
       i--;
    }
    return(0);
}
```

2D STRING

A character type array having two subscripts or dimensions, one specifying number of strings that can be stored and other specifying length of string.

Declaration of String

char s[20][10];

here, 20 represnts number of strings/names

and 10 represents length of string.

Initialization of String

char s[5][10]={“abc”,”pqr”,”xyz”};

2D STRING (CONT...)

Recieving input

```
char s[5][10];
int i;
for(i=0;i<5;i++)
{
scanf(“%s”,&s[i]);
}
```

Printing output

```
char s[5][10];
int i;
for(i=0;i<5;i++)
{
printf(“%s”,s[i]);
}
```

2D STRING (CONT...)

/* PROGRAM TO SORT THE 2D ARRAY OF CHARACTER. */

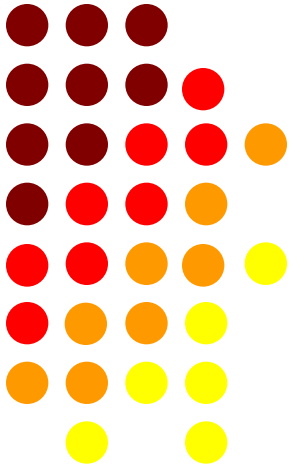
```
#include<stdio.h>
#include<conio.h>
void main()
{
char s[20][10],temp[10];
int n,i,j;
clrscr();
printf(“enter how many names”);
scanf(“%d”,&n);
printf(“enter %d names \n”,n);
for(i=0;i<n;i++)
{
scanf(“%s”,s[i]);
}
```


2D STRING (CONT...)

```
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(strcmp(s[i],s[j])>0)
{
strcpy(temp,s[i]);
strcpy(s[i],s[j]);
strcpy(s[j],temp);
}
}
}
printf("\n names alphabatically \n");
for(i=0;i<n;i++)
{
printf("\n %s",s[i]);
getch();
}
}
```

Programming For Problem Solving

LECTURE-28



STRUCTURE

A structure is a user defined data type which contains number of similar and dissimilar data type group together.

The structure type declaration does not reserve any memory until the variable is declared of that structure.

The structure declaration is terminated by semicolon.

The elements of structure are accessed using dot(.) operator for ordinary variable and arrow(->) operator for pointer variable.

Value of one structure variable can be assigned to another structure variable using = operator.

STRUCTURE (CONT…)

Declaration of Structure

```
struct book
{
        char title[10];
        char author[10];
        int page;
        float price;
};
```

Declaration of Structure variable

```
struct book b1,b2,b3;
```

Array of Structure

```
struct book b[10];
```

STRUCTURE (CONT…)

Initialization of Structure

struct book b={“cp”,”bs”, 500,320}

Here, cp is assign to title of book b and so on.

Recieving structure elements

struct book b; scanf(“%s %s %d %f”, &b.title, &b.author, &b.page, &b.price);

struct book b[10];

```
for(i=0;i<n;i++)
{
scanf(“%s%s%d%f”,b[i].title,b[i].author,&b[i].page,&b[i].price);
}
```

Memory allocation:

title	author	page	price
cp	**bs**	**500**	**320**

b

STRUCTURE (CONT…)

Printing structure elements

```
struct book b;

        printf("%s %s %d %f", b.title, b.author, b.page, b.price);

struct book b[10];
        for(i=0;i<n;i++)
        {
                printf("%s %s %d %f",b[i].title,b[i].author,b[i].page,b[i].price);
        }
```

STRUCTURE (CONT...)

NestedStructure

When a structure contains another structure, it is called nested structure. For example,we have two structures named Address and Employee. To make Address nested to Employee, we have to define Address structure before and outside Employee structure and create an object of Address structure inside Employee structure.

Syntax for structure within structure or nested structure

```
struct structure1
{
     - - - - - - - - - -
     - - - - - - - - - -
};

struct structure2
{
     - - - - - - - - - -
     - - - - - - - - - -
     struct structure1 obj;
};
```

STRUCTUE (CONT…)

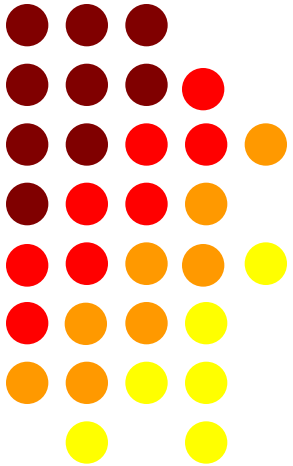
```
/* Example for structure within structure or nested structure */
#include<stdio.h>

    struct Address
    {
        char HouseNo[25];
        char City[25];
        char PinCode[25];
    };

    struct Employee
    {
       int Id;
       char Name[25];
       float Salary;
       struct Address Add;
    };
```


STRUCTUE (CONT…)

```
void main()
    {     int i;
          struct Employee E;
          printf("\n\tEnter Employee Id : ");
          scanf("%d",&E.Id);
          printf("\n\tEnter Employee Name : ");
          scanf("%s",&E.Name);
          printf("\n\tEnter Employee Salary : ");
          scanf("%f",&E.Salary);
          printf("\n\tEnter Employee House No : ");
          scanf("%s",&E.Add.HouseNo);
          printf("\n\tEnter Employee City : ");
          scanf("%s",&E.Add.City);
          printf("\n\tEnter Employee House No : ");
          scanf("%s",&E.Add.PinCode);
          printf("\nDetails of Employees");
          printf("\n\tEmployee Id : %d",E.Id);
          printf("\n\tEmployee Name : %s",E.Name);
          printf("\n\tEmployee Salary : %f",E.Salary);
          printf("\n\tEmployee House No : %s",E.Add.HouseNo);
          printf("\n\tEmployee City : %s",E.Add.City);
          printf("\n\tEmployee House No : %s",E.Add.PinCode);
          getch();
    }
```

Programming For Problem Solving

LECTURE-29



STRUCTURE (CONT…)

/* DECLARE STRUCTURE STUDENT WITH FOLLOWING MEMBER ROLL NO, NAME, FATHER NAME, AGE, CITY, MARKS, AND LIST STUDENTS WHOSE MARKS ARE MORE THAN 75. */

```
#include<stdio.h>
#include<conio.h>
struct student
{
        int rollno;
        char name[10];
        char fname[10];
        int age;
        char city[10];
        int marks;
};
void main()
{
int i, j=0;
struct student s[100];
```


STRUCTURE (CONT…)

```
printf(“enter details of 100 students”);
for(i=0;i<100;i++)
{
scanf(“%d %s %s %d %s %d”, &s[i].rollno, s[i].name, s[i].fname, &s[i].age, s[i].city, &s[i].marks);
}
for(i=0;i<100;i++)
{
If(s[i].marks>75)
{
J++;
printf(“%d %s %s %d %s %d”, s[i].rollno, s[i].name, s[i].fname, s[i].age, s[i].city, s[i].marks);
}
}
if(j==0)
{
printf(“no students marks are greater than 75”);
}
getch();
}
```

UNION

Union is a concept borrowed from structure, but major distinction between them in terms of storage.

In structure each member has its own storage location whereas all the members of union use the same location, union handle only one member at a time.

A union can be declared using keyword union.

Example-

```
union sample
{
int a;
int b;
int c;
};
union sample item;
```

Structure	Union
The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.

When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members. The smaller members may end with unused slack bytes.
Struct student
{
Int rollno; float marks; char name[10];
};
Struct student s;
Size of s=2+4+10=16
	When a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member.
	union student
	{
	Int rollno; float marks; char name[10];
	};
	union student s;
	Size of s=10(

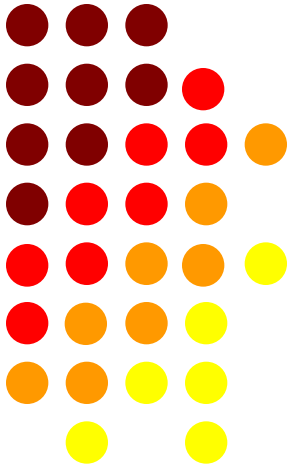
Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
The address of each member will be in ascending order This indicates that memory for each member will start at different offset values.	The address is same for all the members of a union. This indicates that every member begins at the same offset value.
Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Individual member can be accessed at a time	Only one member can be accessed at a time.
Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

Array	Structure
Array is a collection of homogeneous data.	Structure is a collection of heterogeneous data.
Array elements are referred by subscript.	Structure elements are referred by its unique name.
Array elements are accessed by it's position or subscript.	Structure elements are accessed by its object as '.' operator.
Array is a derived data type.	Structure is user defined data type.

<data_type> array_name[size];	struct struct_name
	{
	structure element 1;
	structure element 2;
	ucture element n;
	};
	struct struct_name variable name;

Programming For Problem Solving

LECTURE-30



ENUMERATED DATA TYPE

An enumeration is a user-defined data type

consists of integral constants and each integral constant is given a name.

Keyword enum is used to define enumerated data type.

enum name{ value1, value2,..., valueN };

Here, *name* is the name of enumerated data type or tag.

And *value1*, *value2*,....,*valueN* are values of *name*.

By default, *value1* will be equal to 0, *value2* will be 1 and so on but, the programmer can change the default value as below:

enum dept{ cse=10, me=40, it=13, ee=31};

Example-1

```
#include <stdio.h>
enum week{ sunday, monday, tuesday, wednesday, thursday, friday, Saturday};
nt main()
{   enum week today;
  today=wednesday;
 printf("%d day",today+1);    return 0;
}
```

Output

4 day

ENUMERATED DATA TYPE (CONT...)

Example-2

```
#include <stdio.h>
enum  dept{ cse=10,it=13,me=40,ece=23,ee=13};
nt main()
{    enum dept d1;
  d1=me;
 printf(“ Me branch code = %d ",d1);    return 0;
}
```

Output

Me branch code = 40