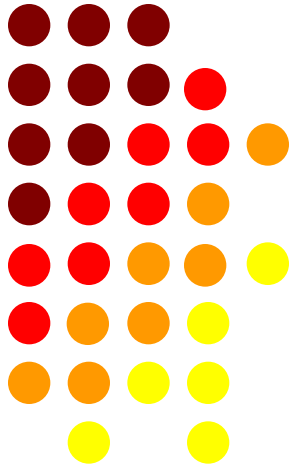
Programming For Problem Solving

Lecture 31



Function

- The self contained block of code used to perform some specific task is known as function.

Why Functions?

- Function breaks the longer programs into conceptually smaller programs which are precisely defined.
- Function is a piece of a code which repeats at many places and can be written only once and used again and again.
- Using function makes the debugging and maintenance very easy.

Type of Functions

- There are two types of functions:
- **Library Function:** The functions which are already defined in C library are known as library functions. Like scanf(), printf(), clrscr(), getch() etc.
- **User Defined Function:** The functions which are defined by programmer to break the solution of problem in to multiple modules are known as user defined functions. Like sum(), fact(), fib(), swap() etc.

Elements of user define function

- Function Declaration/Prototype
- Function calling
- Function Definition

Function Prototype:- The function prototype is also known as function declaration. It specifies the name of the function, type of parameter pass to the function & type of value returned from the function.

- Syntax:- return-type function-name(argument-list);
- **Example:-** void swap (int, int) ;

  In above declaration swap is the name of a function, 2 int are function passing parameters & void is the return type of a function.

Elements…. continue

- **Function calling**:- While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.

- **Syntax**:- function-name(actual arguments);

- **Example**:- swap(int, int);

- **Function Definition**:- It defines the actual body of a function inside a program for executing their tasks in C.

Syntax:- return-type function name(argument-list)

```
{
  Declarations and statements
}
```

Example:-

```
void swap(int x, int y)
 {
   int t;
   t=x;
   x=y;
   y=t;
 }
```

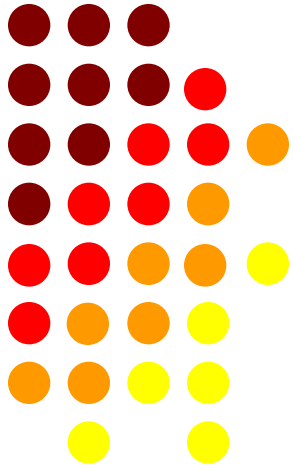

Example(Function):-

/*WAP to add two number using function:-

```
#include<stdio.h>
int add(int,int);
Int main()
{
   int a, b, sum;
   printf(“Enter two numbers:- “);
   scanf(“%d%d”,&a,&b);
    sum=add(a,b);
   printf(“Addition= %d”,sum);
 return 0;
}
Int add(int x, int y)
{
int z;
z=x+y;
return(z);
}
```

Programming For Problem Solving

Lecture 32



Function Parameter Passing Methods

- The parameter passing methods are the statements used to pass the value of variable from calling function to called function. There are two methods to pass the parameters

Call by Value	Call by Reference
In call by value the value of actual argument is pass to the function	In call by reference the address of actual argument is pass to function
In call by value the use of pointer is not necessary	In call by reference the use of pointer is necessary
In call by value, if the value of formal argument changes, there is no change in value of actual argument	In call by reference, if the value of formal argument changes, there is a change in value of actual argument

- **Actual Argument:** Variable or values which are used during function call are known as actual argument.
- **Formal Argument:** Variable or values which are used during function definition are known as formal argument.

Example-1 (Function)

```
/*WAP to swap 2 numbers using call by value*/
#include<stdio.h>
#include<conio.h>
void swap(int, int);
void main()
{
 int a, b;
 clrscr();
 printf("\n Enter value of a, b: \n");
 scanf("%d %d", &a, &b);
 swap(a, b);
 getch();
}
void swap(int a, int b)
{
 printf("\n Before Swapping \n");
 printf("a = %d \t b = %d", a, b);
 a = a + b;
 b = a - b;
 a = a - b;
 printf("\n After Swapping \n");
 printf("a = %d \t b = %d", a, b);
}
```

Example-2 (Function)

/*WAP to swap 2 numbers using call by reference*/

```
#include<stdio.h>
#include<conio.h>
void swap(int*, int*);
void main()
{
 int a, b;
 clrscr();
 printf("\n Enter value of a, b: \n");
 scanf("%d %d", &a, &b);
 swap(&a, &b);
 getch();
}
```

```
void swap(int *a, int *b)
{
 printf("\n Before Swapping \n");
 printf("a = %d \t b = %d", *a, *b);
 *a = *a + *b;
 *b = *a - *b;
 *a = *a - *b;
 printf("\n After Swapping \n");
 printf("a = %d \t b = %d", *a, *b);
}
```

Example-3 (Function)

/*WAP to find factorial of numbers using call by value*/

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main()
{
  int n, f;
 clrscr();
 printf("\n Enter Number: \n");
 scanf("%d", &n);
 f=fact(n);
 printf("factorial=%d", f);
 getch();
}
```

```
int fact(int n)
{
 int i, f=1;
 for(i=n ; i>1 ; i--)
 {
 f = f * i;
 }
 return(f);
}
```

Example-4 (Function)

/*WAP to find factorial of numbers using call by reference*/

```
#include<stdio.h>
#include<conio.h>
int fact(int*);
void main()
{
  int n, f;
 clrscr();
 printf("\n Enter Number: \n");
 scanf("%d", &n);
 f=fact(&n);
 printf("factorial=%d", f);
 getch();
}
```

```
int fact(int *n)
{
 int i, f=1;
 for(i=*n ; i>1 ; i--)
 {
 f = f * i;
 }
 return(f);
}
```

Example-5 (Function)

/*WAP to find sum of digits of numbers using call by value*/

```
#include<stdio.h>
#include<conio.h>
int digit(int);
void main()
{
 int n, s;
 clrscr();
 printf("\n Enter Number: \n");
 scanf("%d", &n);
 s=digit(n);
 printf("sum of digit=%d", s);
 getch();
}
```

```
int digit(int n)
{
 int d, s=0,m;
 m = n;
 while(m>0)
 {
 d = m % 10;
 s = s + d;
 m = m / 10;
 }
 return(s);
 }
```

Example-6 (Function)

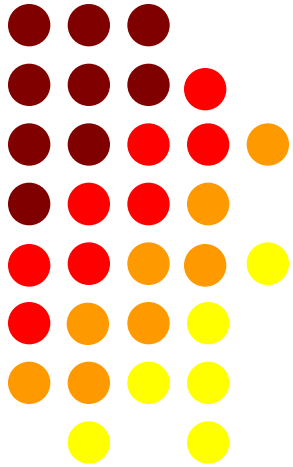
/*WAP to find sum of digits of numbers using call by reference*/

```
#include<stdio.h>
#include<conio.h>
int digit(int*);
void main()
{
 int n, s;
 clrscr();
 printf("\n Enter Number: \n");
 scanf("%d", &n);
 s=digit(&n);
 printf("sum of digit=%d", s);
 getch();
}
```

```
int digit(int *n)
{
 int d, s=0,m;
 m = *n;
 while(m>0)
 {
 d = m % 10;
 s = s + d;
 m = m / 10;
 }
 return(s);
 }
```

Programming For Problem Solving

Lecture 33

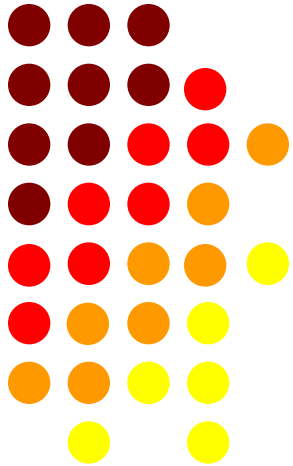


Storage Classes

Keyword	Storage	Initial Value	Scope	Life Time
auto	Memory	Garbage	Local	Till the execution of block
register	Register	Garbage	Local	Till the execution of block
static	Memory	Zero	Local	Value persist between different function calls
extern	Memory	Zero	Global	Till the execution of program

Programming For Problem Solving

Lecture 34



Recursion

- When a function call itself within its block it is known as recursion.
- **Principle of Recursion:** Their must always be a base condition to stop the infinite recursive calls.
- **Advantage:**
- Avoid unnecessary function calls
- Replacement of loop in some case
- **Disadvantage:**
- High memory utilization
- Time consuming

Example-7 (Recursion)

/*WAP to find factorial of numbers using recursion*/

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main()
{
 int n, f;
clrscr();
printf("\n Enter Number: \n");
scanf("%d", &n);
f=fact(n);
printf("factorial=%d", f);
getch();
}
```

```
int fact(int n)
{
 static int f=1;
 if(n==1)
 {
 return 1;
 }
 else
 {
 f = n * fact(n-1);
 }
 return(f);
}
```

Example-8 (Recursion)

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

Example-9 (Recursion)

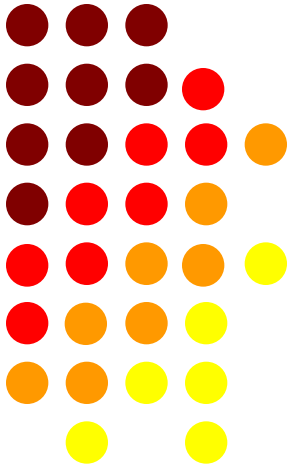
/*WAP to construct Fibonacci series using recursion*/

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

```
int fib(int y)
{
if(y==1)
return(0);
else if(y==2)
return(1);
else
return(fib(y-1)+fib(y-2));
}
```

Programming For Problem Solving

LECTURE-35



SEARCHING

This is the process by which one searches the group of elements for the desired element. There are different methods of searching but let us deal two popular methods of searching and they are linear search and binary search.

Linear Search

This is one of the simplest techniques for searching an unordered table for a particular element. In this each and every entry in the table is checked in a sequential manner until the desired element is found.

Binary Search

In binary search the basic requirement is the elements of the array should have been sorted alphabetically or numerically in the ascending order. In this technique the approximate middle entry of the array is located, and its key value is examined. If its value is too high, then the key value of the middle entry of the first half of the table is examined and the procedure is repeated on the first half until the required element is found or the search interval becomes empty. If the value is too low then the key of the middle entry of the second half of the array is tried and the procedure is repeated on the second half. The procedure continues until the desired key is found or the search interval becomes empty.

LINEAR SEARCH

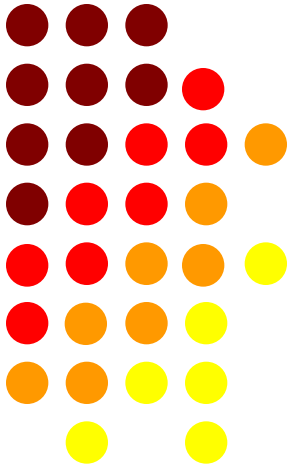
```
/*  PROGRAM FOR LINEAR SEARCH  */
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100];
int i,num,pos,c=0,n;
clrscr();
printf(“enter size of array”);
scanf(“%d”,&n);
for(i=0;i<n;i++)
{
printf(“enter value”);
scanf(“%d”,&a[i]);
}
printf(“enter number which you want to search”);
scanf(“%d”,&num);
```

LINEAR SEARCH (CONT…)

```
for(i=0;i<n;i++)
{
      if(num==a[i])
         {
          printf("%d is found at a[%d]\n",num,i);
                c++;
           }
 }
if(c==0)
printf("%d in not present in array");
else
printf("%d is fount %d times in array",num,c);
getch();
}
```

Programming For Problem Solving

LECTURE-36



BINARY SEARCH LOGIC

```
mid=(beg+end)/2;
        if(a[mid]==num)
          {       pos=mid;
                  break;
             }
       else if(a[mid]<num)
              beg=mid+1;
              else
                  end=mid-1;
```

10 20 30 40 50 60
Beg =0 end=5
Find = 5
mid=(beg+end)/2=2

Find < a [mid] so
end=mid-1=1
mid=(0+1)/2=0
Find < a[mid] so
end=0-1=-1
Beg > end so find is not in list

BINARY SEARCH (CONT…)

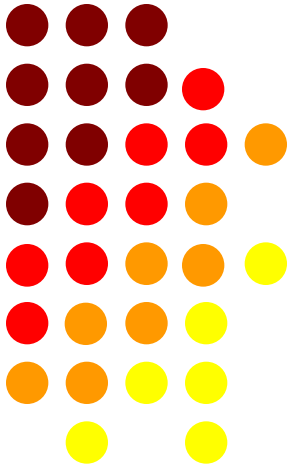
```
/*  PROGRAM FOR BINARY SEARCH  */
#include<stdio.h>
#include<conio.h>
void main()
{
int a[100];
int i,key,pos = -1,mid,beg,end,n;
clrscr();
printf("enter size of array");
scanf("%d",&n);
beg=0;
end=n-1;
for(i=0;i<n;i++)
{
printf("enter value");
scanf("%d",&a[i]);
}
printf("enter number which you want to search");
scanf("%d",&key);
```

BINARY SEARCH (CONT…)

```
while(beg<=end)
 {
        mid=(beg+end)/2;
        if(a[mid]==num)
        {
            printf("%d is found at %d position",num,pos);
             break;
          }
       else if(a[mid]<num)
                    beg=mid+1;
            else
                  end=mid-1;
   }
   if(beg>end)
      printf("%d is not present in array",num);
getch();
   }
```

Programming For Problem Solving

LECTURE-37



SORTING

Sorting is the basic operation in computer science. Sorting is the process of arranging data in some given sequence or order (in increasing or decreasing order).

For example you have an array which contain 10 elements as follow;

10, 3 ,6 12, 4, 17, 5, 9

After shorting value must be;

3, 4, 5, 6, 9, 10, 12, 17

Above value sort by apply any sorting technique.

We have following technique to sort values:

- Bubble Sort
- Selection Sort
- Insertion Sort

BUBBLE SORT

n=4 and numbers are 4 3 2 1

i=0	4	3	2	1	Compare a[0],a[1]
	3	4	2	1	Compare a[1],a[2]
	3	2	**4**	**1**	Compare a[2],a[3]
	3	2	1	4	3 Compare(n-1-i)
i=1	**3**	**2**	1	4	Compare a[0],a[1]
	2	**3**	**1**	4	Compare a[1],a[2]
	2	1	3	4	2 Compare(n-1-i)
i=2	**2**	**1**	3	4	Compare a[0],a[1]
	1	2	3	4	1 Compare (n-1-i)

Total pass= n-1=3

For each pass we need n-1-i Compare

BUBBLE SORT (CONT…)

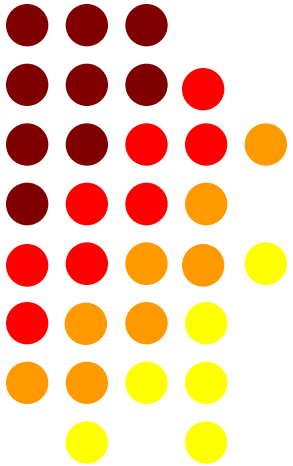
```
/*  Bubble Sort Program */
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,t,a[25],n;
clrscr();
printf("how many elements");
scanf("%d",&n);
for(i=0;i<n;i++)
  {
    printf("enter value");
     scanf("%d",&a[i]);
   }
```

BUBBLE SORT (CONT…)

```
for(i=0;i<n-1;i++)
   {
          for(j=0;j<n-1-i;j++)
             {
                    If(a[j]>a[j+1])
                     {
                       t=a[j];
                       a[j]=a[j+1];
                        a[j+1]=t;
                     }
               }
    }
  printf("\nsorted array is \n");
  for(i=0;i<n;i++)
       printf("%d\t",a[i]);
  getch();
  }
```

Programming For Problem Solving

LECTURE-38



SELECTION SORT

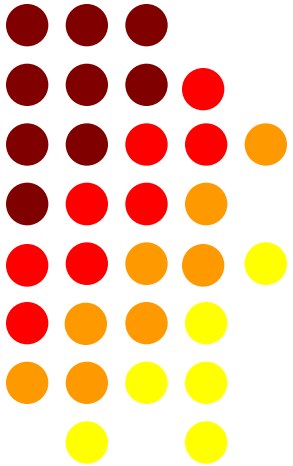
```
/* Selection sort program */
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,min,a[25],n,pos,t;
clrscr();
printf("how many elements");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter value");
scanf("%d",&a[i]);
}
```

SELECTION SORT (CONT…)

```
for(i=0;i<n-1;i++)
  {
        min=a[i];
         pos=i;
        for(j=i+1;j<n;j++)
          {
              if(a[j]<min)
                {
                      min=a[j];
                       pos=j;
                  }
               }
       t=a[i];
       a[i]=a[pos];
       a[pos]=t;
    }
  printf("\nsorted array is \n");
  for(i=0;i<n;i++)
       printf("%d\t",a[i]);
    getch();
   }
```

Programming For Problem Solving

LECTURE-39



INSERTION SORT

```
/*  Insertion sort program */
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,a[25],n,t;
clrscr();
printf("how many elements");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter value");
scanf("%d",&a[i]);
}
```


INSERTION SORT (CONT…)

```
for(i=1;i<n-1;i++)
{
        t=a[i];
        for(j=i-1;j>=0n;j--)
         {
                If(a[j]>t)
                {
                    a[j+1]=a[j];
                 }
                 else
                 break;
             }
        a[j+1]=t;
   }
printf("\nsorted array is \n");
  for(i=0;i<n;i++)
       printf("%d\t",a[i]);
getch();
}
```


THANK YOU :)