Exp No: 1 Date:27/4/21

Familiarization of Hardware Components of a Computer

Computer hardware refers to the physical parts or components of a computer such as the monitor, mouse, keyboard, computer data storage, hard drive disk (HDD), system unit (graphic cards, sound cards, memory, motherboard and chips), etc. all of which are physical objects that can be touched.
The main hardware components are listed below:-

- Microprocessor
- Motherboard
- RAM
- Hard Disk Drive
- Optical disc drive [CD / DVD Drive]
- Keyboard
- Mouse
- Monitor
- Computer case and SMPS
- Computer Speaker
- Uninterrupted power supply (UPS)

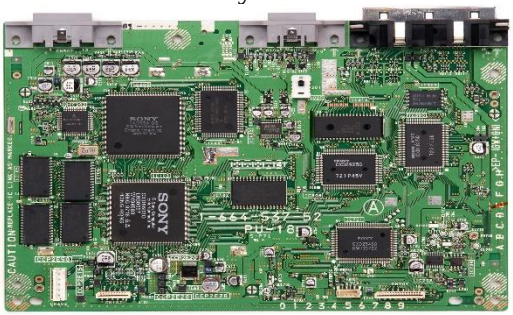
Microprocessor

A microprocessor is an electronic component that is used by a computer to do its work. It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistors, resistors, and diode

that work together. A microprocessor incorporates most or all of the functions of a central processing unit (CPU) on a single integrated circuit (IC).

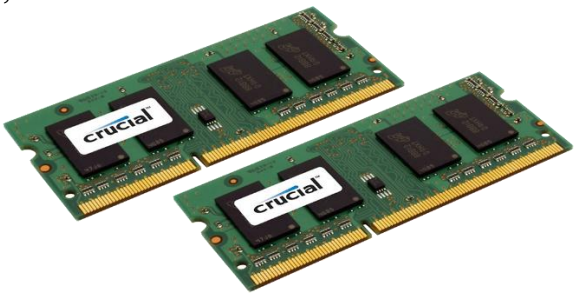
Motherboard

A motherboard is the central or primary printed circuit board (PCB) making up a complex electronic system, such as a modern computer. It is also known as a main board, baseboard, system board, planar board, or, on Apple computers, logic board, and is sometimes abbreviated casually as mobo.



RAM

Random access memory (usually known by its acronym, RAM) is a type of computer data storage. Today it takes the form of integrated circuits that allow the stored data to be accessed in any order, i.e. at random.



Hard Disk Drive

A hard disk drive (HDD), commonly referred to as a hard drive, hard disk, or fixed disk drive, is a non-volatile storage device which stores digitally encoded data on rapidly rotating platters with magnetic surfaces. Strictly speaking, "drive" refers to a device distinct from its medium, such as a tape drive and its tape, or a floppy disk drive and its floppy disk. Early HDDs had removable media; however, an HDD today is typically a sealed unit

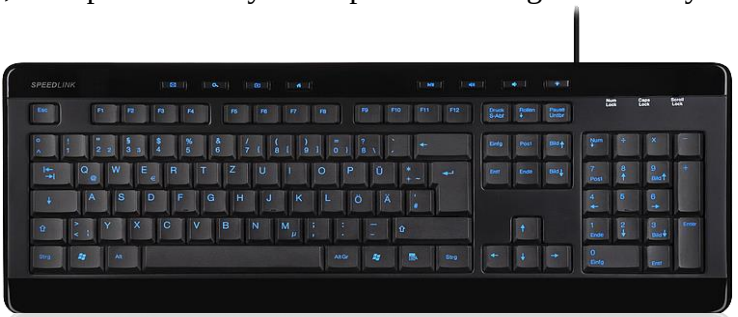
Optical disc drive [CD / DVD Drive]

An optical disc drive (ODD) is a disk drive that uses laser light or electromagnetic waves near the light spectrum as part of the process of reading and writing data.It is a computer's peripheral

device that stores data on optical discs.Some drives can only read from discs, but commonly drives are both readers and recorders. Recorders are sometimes called burners or writers.

Keyboard

A keyboard is an arrangement of buttons, or keys. A keyboard typically has characters engraved or printed on the keys; in most cases, each press of a key corresponds to a single written symbol.



Mouse

A mouse (plural mice, mouse devices, or mouses) is a pointing device that functions by
detecting two-dimensional motion relative to its supporting surface. Physically, a mouse
consists of a small case, held under one of the user's hands, with one or more buttons.

Monitor

A monitor is a piece of computer hardware that displays the video and graphics information generated by a connected computer through the computer's video card.

Computer case and SMPS

A computer case is the enclosure that contains the main components of a computer. Cases are usually constructed from steel, aluminium, or plastic, although other materials such as wood, plexiglas or fans have also been used in case designs. Cases can come in many different sizes, or form factors.

A switched-mode power supply, switching-mode power supply or SMPS, is an electronic power supply unit (PSU) that incorporates a switching regulator. While a linear regulator maintains the desired output voltage by dissipating excess power in a "pass" power transistor, the SMPS rapidly switches a power transistor between saturation (full on) and cutoff (completely off) with a variable duty

cycle whose average is the desired output voltage.

Computer Speaker

Computer speakers, or multimedia speakers, are external speakers, commonly equipped with a low-power internal amplifier. The standard audio connection is 3.5mm (1/8 inch) stereo jacks plug often colour-coded lime green (following the PC 99 standard) for computer sound cards.

Uninterrupted power supply (UPS)

An uninterruptible power supply (UPS), also known as a continuous power supply (CPS) or a battery backup is a device which maintains a continuous supply of electric power to connected equipment by supplying power from a separate source when utility power is not
available.

Small UPS systems provide power for a few minutes; enough to power down the computer in an orderly manner, while larger systems have enough battery for several hours.

Exp No:2 Date:4/5/21

Familiarization of Linux environment –
How to do Programming in C with Linux

Student can login into the system using the username and password then take the terminal, for getting command terminal, search for terminal or use shortcut key *Ctrl+Alt+t.* In terminal each student can login into the lab server using ssh command.
ssh login name@server IP address then press the enter key

Type password then press the enter key

After login 'vi' editor is used for doing programs.

vi program name.c

Press insert key to type the program

For saving the program, press Esc key then type *:wq*

For compile the program using cc command: cc program name.c

After successful compilation take the output of the program using *./a.out* command.

LINUX COMMANDS

1**. ls** :- used to list the files and directories under the current directory.
Syntax:- ls press enter key

2. **rm**:- used to remove a particular file.
Syntax:- rm < file name>
3. **mkdir**:- used to create a directory.
Syntax:- mkdir <directory name>
4. **cd**:- used to change the directory.
Syntax:- cd <directory name>
5. **cd** .. :-used to leave from a particular directory.
Syntax:- cd .. press enter key
6. **rmdir**:- used to remove an empty directory.
Syntax:- rmdir <empty directory name>
7. **cp** :- used copy a file to another file.
Syntax:- cp <source file name> <destination file name>
8. **mv**:- used to moves files or directories from one place to another.
Syntax:- mv <source file name> <destination file name>
9. **man**:- used to displays the whole manual of the command.
Syntax: man <command name>

Algorithm:

Step 1: Start
Step 2: Print “Hello World”
Step 3:Stop

Output:

Hello World

Exp No:3 Date:28/5/21

Display "Hello World"

Aim: Write a program to display "Hello World" on the screen

```
#include <stdio.h>
int main()
{
printf("Hello World\n");
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read two numbera a and b
Step 3: Claculate sum=a+b
Step 4: Print sum
Step 5: Stop

Output:

Enter the two numbers
2
3
Sum=5

Exp No:4 Date:21

Sum of two numbers

Aim: Write a program to read two numbers add them and find their sum

```
#include <stdio.h>
int main()
{
int a,b,sum;
printf("Enter the two numbers\n");
scanf("%d%d",&a,&b);
sum=a+b;
printf("Sum=%d\n",sum);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Declare variables radius and area
Step 3: Read radius of circle and store in radius
Step 4: Calculate area, area=3.14*radius*radius
Step 5: Print Area
Step 6: Stop

Output:

Enter radius of circle: 5
Area of circle is : 78.5

Exp No:5 Date:21

Finding the area of circle

Aim: Write a program to read the radius of a circle, calculate its area and display it

```
#include <stdio.h>
#include<conio.h>
void main()
{
   int radius;
   float  area ;
   printf("\nEnter radius of circle: ");
   scanf("%d",&radius);
   area = 3.14 * radius * radius;
   printf("\nArea of circle is: %f",area);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Declare variables a,b,c,d,e,f,g and exp
Step 3: Read a,b,c,d,e,f and g
Step 4: Calculate ((a -b / c * d + e) * (f +g)) and store value in exp
Step 5: Display exp
Step 6: Stop

Output:

```
Enter the values of a b c d e f in order
2
3
4
5
6
7
Expression value=29.7500
```

Exp No:6 Date:21

Evaluation of Arithmetic expression ((a -b / c * d + e) * (f +g))

Aim: Write a program to evaluate the arithmetic expression ((a -b / c * d + e) * (f +g)) and display its solution.

```
#include <stdio.h>
int main()
{
float a,b,c,d,e,f,g,exp;
printf("Enter the values a b c d e f in order\n");
scanf("%f%f%f%f%f%f",&a,&b,&c,&d,&e,&f);
exp=((a -b / c * d + e) * (f +g));
printf("Expression value=%f\n",exp);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read three numbers a, b and c
Step 3: if(a>=b && a>=c) print “a is largest number”
Step 4: if(b>=a && b>=c) print “b is largest number”
Step 5: if(c>=a && c>=b) print “c is largest number”
Step 6: Stop

Output:

Enter three numbers: 12
15
10
15 is the largest number.

Exp No:7 Date:21

Finding the largest among three numbers

Aim: Write a program to read 3 integer values, find the largest among them.

```
#include <stdio.h>
int main()
{
   int a, b, c;

   printf("Enter three numbers: ");
   scanf("%d %d %d", &a, &b, &c);

   if (a >= b && a >= c)
      printf("%d is the largest number.\n", a);

   if (b >= a && b >= c)
      printf("%d is the largest number.\n", b);

    if (c >= a && c >= b)
      printf("%d is the largest number.\n", c);
} }
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read a number n
Step 3: flag=1
Step 4: i=2
Step 5: Repeat step (a)-(d) until i<n/2
(a): if(n%i==0) goto (b)
(b): flag=0 goto step 6
(c): i=i+1
END WHILE
Step 6: if flag=1 goto step 7 else goto step 8
Step 7: Print “n is a prime number”
Step 8: Print “n is not a prime number”
Step 9: Stop

Output:

Enter a numbers:
23
23 is a prime number

Exp No:8 Date:

Check whether the given number is prime or not

Aim: Write a program to read a Natural Number and check whether the number is prime or not.

```
#include <stdio.h>
int main()
{
   int n, i, flag = 1;
   printf("Enter a number: \n");
   scanf("%d", &n);
   for (i = 2; i <= n / 2; i++)
      {
      // If n is divisible by any number between
      // 2 and n/2, it is not prime
      if (n % i == 0) {
         flag = 0;
         break;
      }
   }
   if (flag == 1)
      printf("%d is a prime number\n", n);
   else
      printf("%d is not a prime number\n", n);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read a number n
Step 3: temp=n, sum=0
Step 4: Repeat step (a)-(c) until n<=0
(a): r=n%10
(b): sum=sum=(r*r*r)
(c): n=n/10
END WHILE
Step 5: if temp=sum THEN
Print “amstrong number”
Step 7: Else Print “not amstrong number”
Step 8: Stop

Output:

Enter a numbers:
153
amstrong number

Exp No:9 Date:

Check whether the given number is Amstrong or not

Aim: Write a program to read a Natural Number and check whether the number is Armstrong or not

```
#include<stdio.h>
 int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong  number ");
else
printf("not armstrong number");
 return 0;
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read the number of elements in array n
Step 3: sum=0
Step 4: read n elements of array a[]
Step 5: i=0
Step 6: Repeat step (a)-(b) until i<n
(a): sum=sum+a[i]
(b): i=i+1
END WHILE
Step 7: average=sum/n
Step 8: Print sum and average
Step 9: Stop

Output:

```
Enter number of elements:
5
Enter the elements:
2
4
6
8
10
Sum= 30.000 Average=6.000
```

Exp No:10 Date:

Find sum and average of n elements in an array

Aim: Write a program to read n integers, store them in an array and find their sum and average

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    int n, i,a[100];
    float sum=0,avg;
    printf("Enter  number of elements: \n");
    scanf("%d", &n);
    printf("Enter the elements..\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        sum=sum+a[i];
   }
    avg=sum/n;
   printf("Sum=%f Average=%f\n",sum,avg);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read the number of elements in array n
Step 3: flag=0
Step 4: read n elements of array a[]
Step 5: read the number to be searched and store it to key
Step 6: Repeat step (a)-(b) until i<n
(a): if a[i]==key THEN
flag=1
goto step7
(b): i=i+1
END WHILE
Step 7: Print("Element is found at index i")
Step 8: if flag!=0 print("element is not found")
Step 9: Stop

Output:

```
Enter number of elements:
5
Enter the elements:
2
4
6
8
10
Enter the element to search
6
Element is found at index 2
```

Exp No:11 Date:

Perform Linear Search in an array

Aim: Write a program to read n integers, store them in an array and search for an element in the array using an algorithm for Linear Search

```
#include <stdio.h>
int main()
{
   int n, i,a[100],key,flag=-1;
   printf("Enter number of elements: \n");
   scanf("%d",&n);
   printf("Enter the list of elements..\n");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   printf("Enter the element to search..\n");
      scanf("%d",&key);
   for (i=0;i<n;i++)
      if (a[i]==key)
       { flag=1;break;}
  if(flag==1)
       printf("Element is found at index %d\n",i);
  else
       printf("Element is not found\n");
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read the number of elements in array n
Step 3: flag=0
Step 4: read n elements of array a[]
Step 5:
Step 6: Repeat step (a)-(b) until i<n
(a): repeat steps (c)-() until j<n-i- 1
(b): flag=0
(c):if a[j]>a[j+1] THEN
temp=a[j]
a[j]=a[j+1]
a[j+1]=temp
flag=1
(d)j=j+1
(e)i=i+1
END WHILE
Step 7:Print the sorted list
Step 8: Stop

Output:
Enter number of elements:
5
Enter the elements:
78
10
32
12
56
The sorted list is …..
10
12
32
56

Exp No:12 Date:

Perform Bubble sort in an Array

Aim: Write a program to read n integers, store them in an array and sort the elements in the array using Bubble Sort algorithm

```
#include <stdio.h>
int main()
{
   int n, i,j,a[100],temp,flag;
   printf("Enter number of elements: \n");
   scanf("%d",&n);
   printf("Enter the list of elements..\n");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   for(i = 0; i < n; i++)
   {
     for(j = 0; j < n-i-1; j++)
      {
          // introducing a flag to monitor swapping
       flag = 0;
       if( a[j] > a[j+1])
         {
              // swap the elements
              temp = a[j];
              a[j] = a[j+1];
              a[j+1] = temp;
              // if swapping happens update flag to 1
```

56
78

```
flag = 1;
        }     }
      if ( flag==0) break;// if no swapping the list is sorted
     }
     printf("The sorted list is ...\n");
     for(i=0;i<n;i++)
      printf("%d\n",a[i]);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read the string str
Step 3: find the length of str using library function and store it in ln
Step 4: i=ln and j=0
Step 5: Repeat step (a)-(d) until i>=0
(a): rstr[j]=str[i]
(b): rstr[j]='/0'
(c): j=j+1
(d): i=i-1
END WHILE
Step 7: if strcmp(rstr,str)=0
Print("Palindrome") else goto step 8
Step 8: Print("Not Palindrome")
Step 9: Stop

Output:

Enter the string(word):
malayalam
Palindrome

Exp No:13 Date:

Check whether the given string is palindrome or not

Aim: Write a program to read a string (word), store it in an array and check whether it is a palindrome word or not.

```
#include <string.h>
int main()
{
   char str[100],rstr[100];
   int i,j,ln;
   printf("Enter the string(word): ");
   scanf("%s",str);
   ln=strlen(str);// reversing the string
   for(i=ln-1,j=0;i>=0;i--,j++)
        rstr[j]=str[i];
   rstr[j]='\0';
   if(strcmp(rstr,str)==0)
     printf("Palindrome\n");
   else
     printf("Not Palindrome\n");
 }
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read the string str1 and str2
Step 3: i=0 and j=0
Step 4: Repeat step (a)-(e) until str1[i]!=$
(a): repeat steps (b)-(d) until str2[j]!=$
(b): str1[i]=str2[j]
(c): str1[i]='/0'
(d): j=j+1
(e):i=i+1
END WHILE
Step 5: Print the concatenated string str1
Step 6: Stop

Output:

Enter the string1: Computer$
Enter the string2: Programming$
Concatenated string is
ComputerProgramming

Exp No:14 Date:21

Concatenate two strings without using library functions

Aim: Write a program to read two strings (each one ending with a $ symbol), store them in arrays and concatenate them without using library functions

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str1[100],str2[100];
   int i,j;
   printf("Enter the string1: ");
   fgets(str1,100,stdin);
   printf("Enter the string2: ");
   fgets(str2,100,stdin);
      for(i=0;str1[i]!='$';i++);
   for(j=0;str2[j]!='$';j++,i++)
     str1[i]=str2[j];
   str1[i]='\0';
   printf("conatenated string is \n");
   printf("%s\n",str1);
 }
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Read the string str
Step 3: initialize vowels=consonants=digits=spaces=0
Step 4: i=0 and j=0
Step 5: Repeat step (a)-(e) until str[i]!=0
(a): check for vowel. If vowel is present at str[i]
vowels=vowels+1
(b): check for consonants if present at str[i]
consonanats=consonants+1
(c): if digit is present at str[i]
Digit=digit+1
(d): if space is present at str[i]
spaces=spaces+1
(e): i=i+1
END WHILE
Step 7: Print vowels
Step 8: Print consonants
Step 9: Print digits
Step 9: Print spaces
Step 10: Stop

Exp No:15 Date:

Count the number of Vowels, Consonants and spaces in a string

Aim: Write a program to read a string (ending with a $ symbol), store it in an array and count the number of vowels, consonants and spaces in it

```
#include <stdio.h>
int main()
{
   char str[150];
   int i, vowels, consonants, digits, spaces;
   vowels =  consonants = digits = spaces = 0;
   printf("Enter a  string: ");
   scanf("%[^\n]", str);
   for(i=0; str[i]!='\0'; ++i)
   {
      if(str[i]=='a' || str[i]=='e' || str[i]=='i' ||
         str[i]=='o' || str[i]=='u' || str[i]=='A' ||
         str[i]=='E' || str[i]=='I' || str[i]=='O' ||
         str[i]=='U')
      {
         ++vowels;
      }
      else if((str[i]>='a'&& str[i]<='z') || (str[i]>='A'&& str[i]<='Z'))
      {
         ++consonants;
      }
      else if(str[i]>='0' && str[i]<='9')
      {
```

```
        ++digits;
```

Output:

```
Enter a string: computer programming
Vowels: 6
Consonants: 13
Digits: 0
Whit spaces: 1
```

```
 }

else if (str[i]==' ')
        {
           ++spaces;
        }
    }
    printf("Vowels: %d",vowels);
    printf("\nConsonants: %d",consonants);
    printf("\nDigits: %d",digits);
    printf("\nWhite spaces: %d\n", spaces);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Create a structure Point
Declare two structure members x and y
Step 3: Read the first point p1.x and p1.y
Step 4: Read the second point p2.x and p2.y
Step 5: calculate p3.x=p1.x+p2.x
Step 7: calculate p3.y=p1.y+p2.y
Step 8: Print p3.x and p3.y
Step 9: Stop

Output:

```
Enter the first point(x1,y1)
 10
Enter the second point(x2,y2)
20
new point after addition
(30,0)
```

Exp No:16 Date:21

Find the sum of two distance values using structure

Aim: Write a program to read two input each representing the distances between two points in the Euclidean space, store these in structure variables and add the two distance values.

```
#include <stdio.h>
struct Point
{
int x;
int y;
}p1,p2,p3;
int main()
{
  printf("Enter the first point(x1,y1)\n");
  scanf("%d,%d",&p1.x,&p1.y);
  printf("Enter the second point(x2,y2)\n");
  scanf("%d,%d",&p2.x,&p2.y);
  p3.x=p1.x+p2.x;
  p3.y=p1.y+p2.y;
  printf("new point after addition\n");
  printf("(%d,%d)\n",p3.x,p3.y);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Create a structure employee
Declare three members int empid
char name[50]
int salary
Step 3: Read number of employees
Step 4: i=0
Step 5: Repeat step (a)-(d) until i<n
(a): Read employee id emp[i].id
(b): Read employee name emp[i].name
(c): Read employye salary emp[i].salary
(d): i=i+1
END WHILE
Step 7: set i=0
Step 8: Repeat step (a)-(d) until i<n
(a): Print employee id emp[i].id
(b): Print employee name emp[i].name
(c): Print employye salary emp[i].salary
(d): i=i+1
END WHILE
Step 9: Stop

Exp No:17 Date:

Read and print the data of n employees using structure

Aim: Write a program to read and print data of n employees using structures. (Name, Employee Id and Salary)

```
#include <stdio.h>
struct Employee
{
int empid;
char name[50];
int salary;
}emp[50];
int main()
{
  int n,i:
  printf("Enter the number of employees\n");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    printf("Enter the employee details-%d\n",i+1);
    printf("Employee id:");
    scanf("%d",&emp[i].empid);
    getchar();
    printf("Employee name:");
    scanf("%[^\n]",emp[i].name);
    printf("Employee salary:");
     scanf("%d",&emp[i].salary);
  }
```

Output:

```
Enter the number of employees
2
Enter the employee details-1
Employee id: 301
Employee name: Rahul
Employee salary: 30000
Enter the employee details-2
Employee id: 303
Employee name: Jackson
Employee salary: 40000

Employee Details
Employee id ---Employee name---Employee salary
301             Rahul           30000
303             Jackson          40000
```

```
 //printing the details
 printf("Employee Details\n");
 printf("Employee id---Employee name---Employee salary\n");
 for(i=0;i<n;i++)
 printf("%-15d %-15s%10d\n",
emp[i].empid,emp[i].name,emp[i].salary);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: define C_size as 50
Step 3: declare a union named Address
    3.1: declare union members name, hname, cityname, state and pin each with C_size
Step 5: Declare a variable record1 of type union Address
Step 4: Read name, housename, city name, state and pin by accessing each union members
Step 5: Display name name, housename, city name, state and pin by accessing each union members
Step 6: Stop

Exp No:18 Date:

Read and display address of a person using union

Aim: Write a program to read and display the address of a person using a variable of the union. Declare a union containing 5 string variables (Name, House Name, City Name, State and Pin code) each with a length of C_SIZE (user defined constant).

```
#include <stdio.h>
#include <string.h>
#define C_SIZE 50
union Address
{
      char name[C_SIZE];
      char hname[C_SIZE];
      char cityname[C_SIZE];
      char state[C_SIZE];
      char pin[C_SIZE];
};

int main()
{
   union Address record1;

   printf("Enter name:");
   scanf("%[^\n]",record1.name);
   getchar();
```

Output:

```
Enter name:Jimmy
Enter house name:xyz
Enter city name:Kochi
Enter state name:Kerala
Enter pin:680245

Union record1 values ....
 Name:  680245
 House Name:   680245
 City Name:  680245
 State name:  680245
 Pin:  680245
```

```
    printf("Enter house name:");
    scanf("%[^\n]",record1.hname);
    getchar();
    printf("Enter city name:");
    scanf("%[^\n]",record1.cityname);
    getchar();
    printf("Enter state name:");
    scanf("%[^\n]",record1.state);
    getchar();
    printf("Enter pin:");
    scanf("%[^\n]",record1.pin);

    printf("Union record1 values ....\n");
    printf(" Name:  %s \n", record1.name);
    printf(" House Name:  %s \n", record1.hname);
    printf(" City Name:  %s \n\n", record1.cityname);
    printf(" State name:  %s \n", record1.state);
    printf(" Pin:  %s \n\n", record1.pin);
 }
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: declare a non recursive function factnr(int n) to find factorial
    2.1: f=1 and i=0
    2.2: repeat step (a)-(c) until i<=n
        (a): f=f*i
        (b): return f
        (c): i=i+1
Step 3: declare a recursive function factr(int n) to find factorial
    3.1: if(n==0) return 1
        else return(n*factr(n-1))
Step 4: Read a number n to find factorial
Step 5: To find factorial using non recursive function goto step 2
    Print factorial
Step 6: To find factorial using non recursive function goto step 3
    Print factorial
Step 9: Stop

Output:

Enter the number
5
Factorial using non recursive function 5!=120
Factorial using recursive function 5!=120

Exp No:19 Date:

Factorial of a number using recursive and non recursive functions

Aim: Write a program to find the factorial of a given Natural Number n using:

i) a non recursive function

ii) a recursive function

```
#include <stdio.h>
long int factnr(int n)
{ int i;
  long int f=1;
  for(i=1;i<=n;i++)
      f=f*i;
  return f;
}
long int factr(int n)
{
  if(n==0) return 1;
  else
  return (n*factr(n-1));
}
int main()
{
int n;
printf("Enter the number \n");
scanf("%d",&n);
printf("Factorial using non recursive function  %d!=%ld\n",
n,factnr(n));
printf("Factorial using recursive function  %d!=%ld\n", n,factr(n));
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Define a function reversestr(char str[]) with return type void
    2.1: declare two integer variables i and n
    2.2: declare a character c
    2.3: find length of str and store it into n
    2.4: i=0
    2.5 Repeat step (a)-() until i<n/2
        (a): c=str[i]
        (b): str[i]=str[n-1-i]
        (c): str[n-1-i]=c
        (d): i=i+1
    END WHILE
Step 3: Read a string str
Step 4: call the function reversestr(str)
Step 5: Print reversed string
Step 6: Stop

Exp No:20 Date:

Find the reverse of a string using user defined functions

Aim: Write a program to read a string (word), store it in an array and obtain its reverse by using a user defined function.

```
#include <stdio.h>
#include <string.h>
void reversestr(char str[])
{
 int i,n;
  char c;
  n=strlen(str);
  for(i=0;i<n/2;i++)
    {
       c=str[i];
        str[i]=str[n-1-i];
        str[n-1-i]=c;
     }
}
int main()
{
  char str[100];
  system("clear");
  printf("Enter the string \n");
  scanf("%[^\n]",str);
  reversestr(str);
  printf("Reversed string is %s\n",str);
}
```

Output:

```
Enter the string
Computer
Reversed string is retupmoc
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Declare a function readmatrix(int a[][100],int m,intn) with void return type
2.1: i=0, j=0
2.2: Repeat step(a)-(d) until i<m
(a): repeat step (b)-(c) until j<n
(b): read element a[i][j]
(c): j=j+1
(d): i=i+1
Step 3: declare a function displaymatrix(int a[][100],int m, int n)
3.1: i=0, j=0
3.2: Repeat step(a)-(d) until i<m
(a): repeat step (b)-(c) until j<n
(b): print element a[i][j]
(c): j=j+1
(d): i=i+1
END WHILE
Step 4: Declare function addmatrix
4.1: i=0, j=0
4.2: declare another array c[][100]
4.3: Repeat step(a)-(d) until i<m
(a): repeat step (b)-(c) until j<n
(b):print c[i][j]=a[i][j]+b[i][j]
(c): j=j+1
(d): i=i+1
END WHILE
Step 5: Step 4: Declare function transpose
4.1: i=0, j=0
4.2: declare another array c[][100]
4.3: Repeat step(a)-(d) until i<m
(a): repeat step (b)-(c) until j<n
(b):print c[i][j]=a[i][j]+b[i][j]

Exp No:21 Date:

Menu driven program to perform matrix operations

Aim: Write a menu driven program for performing matrix addition, multiplication and finding the transpose. Use functions to
(i) read a matrix, (ii) find the sum of two matrices,
(iii) find the product of two matrices,(iv) find the transpose of a matrix and (v) display a matrix.

```
#include <stdio.h>
#include <stdlib.h>
void readmatrix(int a[][100],int m,int n)
{
 int i,j;
 printf("enter the elements row by row\n");
 for(i=0;i<m;i++)
   for(j=0;j<n;j++)
    scanf("%d",&a[i][j]);
}
void displaymatrix(int a[][100],int m,int n)
{
 int i,j;
 for(i=0;i<m;i++)
 {
  for(j=0;j<n;j++)
    printf("%5d",a[i][j]);
  printf("\n");
  }
```

(c): call function displaymatrix and pass values of c n and m
(d): j=j+1
(e): i=i+1
END WHILE

Step 6: Declare function multmatrix
4.1: i=0, j=0
4.2: declare another array c[100][100]
4.3: Repeat step(a)-(d) until i<m1
(a): repeat step (b)-(c) until j<n2
(b): set c[i][j]=0
(c): repeat steps (d)-()until k<n1
(d): c[i][j] = c[i][j]+a[i][k] * b[k][j]
(e): k=k+1
(e): j=j+1
(d): i=i+1
END WHILE
4.4: call the function displaymatrix pass the values c,m1,n2
Step 7: Read the size of matrix A row size m1 and column size n1
Step 8: Read matrix A by calling function readmatrix and pass values m1 and n1 goto step 2
Step 8: Read the size of matrix B row size m2 and column size n2
Step 9: Read matrix B by calling function readmatrix and pass values m2 and n2 goto step 2
Step 10: Display matrix A, call function displaymatrix and pass values a,m1,n1 goto step 3
Step 11: Display matrix B, call function displaymatrix and pass values b,m2,n2 goto step 3
Step 12: Display a menu with choices 1.add 2.multiply 3.transpose 4.exit
Step 13: Read the choice
Step 14: if choice =1
if(m1==m2 && n1==n2) THEN
goto step 2

```
}
void addmatrix(int a[][100],int b[][100],int m,int n)
{
 int i,j,c[100][100];
 for(i=0;i<m;i++)
   for(j=0;j<n;j++)
    c[i][j]=a[i][j]+b[i][j];
 printf("Sum of matrix...\n");
 displaymatrix(c,m,n);
}
void transpose(int a[][100],int m,int n)
{
 int i,j,c[100][100];
 for(i=0;i<m;i++)
   for(j=0;j<n;j++)
    c[j][i]=a[i][j];

 displaymatrix(c,n,m);
}
void multmatrix(int a[][100],int b[][100],int m1,int n1,int n2)
{
 int c[100][100],i,j,k;
// Multiply the two
    for (i = 0; i < m1; i++)
    {
       for (j = 0; j < n2; j++)
        {
           c[i][j] = 0;
           for (k = 0; k < n1; k++)
            c[i][j] += a[i][k] * b[k][j];
        }
     }
```

else print (“incompatible matrix cannot add”)
Step 15: if choice=2
If (n1=m2) THEN goto step 6
else print(“Incompatible matrix cannot multiply”)
Step 16: if choice= 3 goto step 5
Step 17: if choice is 4 goto step 18
Step 18: Stop

Output:

```
Enter the size of the matrix A row,column
2
2
Enter Matrix A
enter the elements row by row
1
2
3
4
Enter the size of the matrix B row column
2
2
Enter Matrix B
enter the elements row by row
1
2
3
4
Matrix A..
   1   2
   3   4
Matrix B..
   1   2
   3   4
```

```
printf("Product of matrix...\n");
 displaymatrix(c,m1,n2);
}
int main()
{ int a[100][100],b[100][100],m1,n1,m2,n2,op;
  printf("Enter the size of the matrix A row,column\n");
  scanf("%d%d",&m1,&n1);
  printf("Enter Matrix A\n");
  readmatrix(a,m1,n1);
  printf("Enter the size of the matrix B row column\n");
  scanf("%d%d",&m2,&n2);
  printf("Enter Matrix B\n");
  readmatrix(b,m2,n2);
  printf("Matrix A..\n");
  displaymatrix(a,m1,n1);
  printf("Matrix B..\n");
  displaymatrix(b,m2,n2);
  while(1)
  {
   printf("\n************************************\n");
   printf("1.add  2.multiply  3.transpose 4.exit \n");
   printf("Enter the option.....:");
   scanf("%d",&op);
   switch(op)
   {
    case 1: if(m1==m2 && n1==n2)
              addmatrix(a,b,m1,n1);
            else
              printf("Incompatable matrix...cannot add..\n");
             break;
```

```
1.add  2.multiply  3.transpose 4.exit
Enter the option.....:2
Product of matrix...
   7   10
  15   22

***********************************
1.add  2.multiply  3.transpose 4.exit
Enter the option.....:3
Transpose of A..
   1   3
   2   4
Transpose of B..
   1   3
   2   4

***********************************
1.add  2.multiply  3.transpose 4.exit
Enter the option.....:4
```

```
 case 2: if(n1==m2)
                multmatrix(a,b,m1,n1,n2);
              else
                printf("Incompatable matrix...cannot mutliply..\n");
               break;
    case 3: printf("Transpose of A..\n");
               transpose(a,m1,n1);
               printf("Transpose of B..\n");
               transpose(b,m2,n2);
               break;
    case 4: exit(0);
   }
   }
}
```

Result: The program has been executed and output has been obtained

Algorithm:

1)
Step 1: Start
Step 2: declare integer variables first, second, pointer p, pointer q and sum
Step 3: Read integers first and second
Step 4: pass address of first to p
Step 5: pass address of second to q
Step 6: add p abd q and store it in sum
Step 7: Display sum
Step 8: Stop

Output:

Enter two integers to add
12
13
Sum of the numbers=25

2)
Algorithm:

Step 1: Start
Step 2: define a function swap(int *xp, int *yp)
    2.1: set temp=*xp
    2.2: *xp=*yp, swapping the values
    2.3: *yp=temp
Step 3: Declare two integer variables x and y
Step 4: Read values of x and y

Exp No:22 Date:

Program using pointers

Aim: Write a program using pointers to

1) add two numbers
2) swap two numbers using user defined functions

1) add two numbers

```
#include <stdio.h>
int main()
   {
     int first, second, *p, *q, sum;
     printf("Enter two integers to add\n");
     scanf("%d%d", &first, &second);
     p = &first;
     q = &second;
     sum = *p + *q;
     printf("Sum of the numbers = %d\n", sum);
    }
```

2) swap two numbers using user defined function

```
#include <stdio.h>
// This function swaps values pointed by xp and yp
void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
```

Step 5: call the function swap goto step 2 and pass the address of x and y
Step 6: Display the values of x and y after swapping
Step 7: Stop

Output:

Enter value of x 23
Enter value of y 34

After swapping:x=34, y=23

```
*yp = temp;
}
int main()
{
   int x, y;
   printf("Enter Value of x ");
   scanf("%d", &x);
   printf("\nEnter Value of y ");
   scanf("%d", &y);
   swap(&x, &y);
   printf("\nAfter Swapping: x = %d, y = %d", x, y);
   return 0;
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Declare array arr[100], pointer ptr and i
Step 3: Pass pointer to arr[0]
Step 4: Read the number of elements n in array
Step 5: i=0
Step 6: repeat steps (a)-(b) until i<n
(a): read i^{th} element using pointer (ptr+i)
(b): i=i+1
END WHILE
Step 7: Repeat steps (a)-(b) until i<n
(a): print i^{th} element using pointer *(ptr+i)
(b): i=i+1
END WHILE
Step 8: Stop

Output:

```
Enter size of array: 5
Enter elements in array:
1
2
3
4
5
Array elements:
1
2
3
4
5
```

Exp No:23 Date:

Input and print elements of an array using pointers

Aim: Write a program to input and print the elements of an array using pointers

```
#include <stdio.h>
int main()
{
   int arr[100];
   int n, i;
   int * ptr = arr;    // Pointer to arr[0]
   printf("Enter size of array: ");
   scanf("%d", &n);
    printf("Enter elements in array:\n");
   for (i = 0; i < n; i++)
   {
      scanf("%d", (ptr + i));
   }

   printf("Array elements: \n");
   for (i = 0; i < n; i++)
   {
      printf("%d\n", *(ptr + i));
   }
 }
```

Result: The program has been executed and output has been obtained.

Algorithm:

Step 1: Start
Step 2: define a function arraysum(int *ptr, int n)
    2.1: declare sum=0 and i=0
    2.2: Repeat steps(a)-(b) until i<n
        (a): sum=sum+ *(ptr+i)
        (b): i=i+1
        END WHILE
Step 3: declare array arr[]={4,5,6,7,8,9,10,1,2,3}
Step 4: declare integer variable sum
Step 5: Call the function arraysum and pass array arr and 10
Step 6: Display array sum, sum
Step 7: Stop

Output:

Array element sum=55

Exp No:24 Date:

Compute sum of the elements stored in an array using pointers and user defined function

Aim: Write a program to compute sum of the elements stored in an array using pointers and user defined functions.

```
#include <stdio.h>
#include <stdlib.h>
int arraysum(int *ptr,int n)
{
 int sum=0,i;
for (i = 0; i < n; i++)
    {
        sum=sum+ *(ptr + i);
    }
 return sum;
}
int main()
{
    int arr[]={4,5,6,7,8,9,10,1,2,3};
    int  sum;
    sum=arraysum(arr,10);
    printf("Array elements sum=%d \n",sum);
}
```

Result: The program has been executed and output has been obtained

Algorithm:

i)
Step 1: Declare variable fp as a pointer to data type FILE
Step 2: Declare a character array t[100]
Step 3: open the file a.txt to read
Step 4: If fp=NULL, THEN
Print (“error opening file “)
Exit the program
Step 5: add text to file “Welcome to file handling in C”
Step 6: close the file
Step 7: Stop

Output:

File Created...named a.txt

Exp No:25 Date:

Create a file and write data, read data and append new data

Aim: write a program to create a file and perform the following

i) Write data to the file

ii) Read the data in a given file & display the file content on console

iii) append new data and display on console

i) Write data to the file

```
#include <stdlib.h>
#include<stdio.h>
int main()
{
FILE *fp;
fp=fopen("b.txt","w");
if (fp==NULL)
{
printf("error opening file..\n");
exit(1);
}
else
{
fprintf(fp,"%s","Welcome\n");
fprintf(fp,"%s","to file handling in C\n");
}
printf("File Created...named a.txt");
fclose(fp);
}
```

ii)

Step 1: Start
Step 2: : Declare variable fp as a pointer to data type FILE
Step 3: open the file a.txt to read
Step 4: If fp=NULL, THEN
Print (“error opening file “)
Exit the program
Step 5: read the data in file line by line using fgets()
Step 6: Print the data
Step 7: Close the file
Step 8: Stop

Output:

Content of File a.txt
.....
Welcome to file handling in C

iii)
Step 1: Start
Step 2: Declare variable fp as a pointer to data type FILE
Step 3: open the file a.txt to read
Step 4: If fp=NULL, THEN
Print (“error opening file “)
Exit the program

ii) Read the data in a given file & display the file content on console

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   FILE *fp ;
   char t[100];
   fp=fopen("a.txt","r");
   if (fp == NULL)
   {
      printf("Error opening source file..");
      exit(1);
   }
   printf("Content of File a.txt\n.....\n");
   while((fgets(t,sizeof(t),fp)!=NULL));
   {
      printf("%s\n", t);
   }
   fclose(fp);
}
```

iii Append data to a file and display the contents to console

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *fp;
```

Step 5: Read the contents to append using gets()
Step 6: close the file
Step 7: Open the file a.txt to read
Step 8: Print the content in the file
Step 9: Close the file
Step 10: Stop

Output:

Enter the contents to append.....
Welcome to C Programming

```
char t[100];
fp=fopen("a.txt","a");
if(fp==NULL)
  {
 printf("Error opening source file..");
  exit(1);
   }
printf("Enter the contents to append.....\n");
while(1){
 fgets(t,sizeof(t),stdin);
if(strcmp(t,"end\n")==0) break;
 fputs(t,fp);
}
fclose(fp);
fp=fopen("a.txt","r");
printf("File contents after appending...\n");
printf("**********\n");
while(fgets(t,sizeof(t),fp)!=NULL)
{
printf("%s",t);
}
fclose(fp)
}
}
```

Result: The program has been executed and output has been obtained

Algorithm:

Step 1: Start
Step 2: Declare variable fp as a pointer to data type FILE
Step 3: Declare a character fname[50]
Step 4: Declare integer variables ch,nl,nc,nw
Step 5: set nl=0,nc=0, nw=0
Step 6: Read the file name
Step 7: open the file to read
Step 8: if (fp=NULL) THEN
Print("error opening file")
Exit the code
Step 9: Get the character from file and store it in ch
Step 10: While (ch!=EOF)
If (ch=='\n') nl=nl+1
If(ch==' ') nw=nw+1
nc=nc+1
get character from file
Step 11: close the file
Step 12: print number of lines(nl), number of words(nc), number of characters(nw+nl)
Step 13: open the file and write the result into the file
Step 14: Stop

Exp No:26 Date:

Open a text input file and count number of characters, words and lines in it; and store the results in an output file

Aim: Write a program to open a text input file and count number of characters, words and lines in it; and store the results in an output file.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *fp;
char fname[50];
int ch;
int nl=0,nc=0,nw=0;
printf("Enter the file name....\n");
scanf("%[^\n]",fname);
fp=fopen(fname,"r");
if(fp==NULL)
  {
  printf("Error opening file..");
  exit(1);
   }
ch=getc(fp);
while(ch!=EOF)
{
if (ch=='\n') nl++;
```

```
if(ch==' ') nw++;
nc++;
ch=getc(fp);
}
fclose(fp);
printf("Number of lines=%d Number of words=%d ,Number of
characters = %d,\n",nl,nc,nw+nl);
printf("results are written into result.dat file..\n");
fp=fopen("result.dat","w");
fprintf(fp,"Number of lines=%d Number of words=%d ,Number of
characters = %d,\n",nl,nc,nw+nl);
fclose(fp);
}
```

Result: The program has been executed and output has been obtained