TEK97

FE SEM II CBCGS - 2018 EDITION

STRUCTURED PROGRAMMING APPROACH

RANJI RAJ NAIR

TEK97

STRUCTURED PROGRAMMING APPROACH

FE SEM II (Common For All Branches)

(New CBCGS Syllabus – Mumbai University)

RANJI RAJ NAIR

2018 Edition

Syllabus

Module	Topic	Detailed Contents
01	**Introduction to Computer, Algorithm And Flowchart**	**1.1 Basics of Computer:** Turing Model, Von Neumann Model, Basics of Positional Number System, Introduction to Operating System and component of an Operating System. **1.2 Algorithm & Flowchart :** Three construct of Algorithm and flowchart: Sequence, Decision (Selection) and Repetition
02	**Fundamentals of C-Programming**	**2.1** Character Set, Identifiers and keywords, Data types, Constants, Variables. **2.2 Operators**-Arithmetic, Relational and logical, Assignment, Unary, Conditional, Bitwise, Comma, other operators. Expression, statements, Library Functions, Preprocessor. **2.3 Data Input and Output** – getchar( ), putchar( ), scanf( ), printf( ), gets( ), puts( ), Structure of C program .
03	**Control Structures**	**3.1 Branching** - If statement, If-else Statement, Multiway decision. **3.2 Looping** – while , do-while, for **3.3 Nested control structure**- Switch statement, Continue statement Break statement, Goto statement.
04	**Functions and Parameter**	**4.1Function** -Introduction of Function, Function Main, Defining a Function, Accessing a Function, Function Prototype, Passing Arguments to a Function, Recursion. **4.2 Storage Classes** –Auto , Extern , Static, Register
05	**Arrays , String Structure and Union**	**5.1 Array**-Concepts, Declaration, Definition, Accessing array element, One-dimensional and Multidimensional array. **5.2 String**- Basic of String, Array of String , Functions in

		String.h **5.3 Structure**- Declaration, Initialization, structure within structure, Operation on structures, Array of Structure. **5.4 Union** - Definition , Difference between structure and union , Operations on a union
06	**Pointer and Files**	**6.1 Pointer** : Introduction, Definition and uses of Pointers, Address Operator, Pointer Variables, Dereferencing Pointer, Void Pointer, Pointer Arithmetic, Pointers to Pointers, Pointers and Array, Passing Arrays to Function, Pointers and Function, Pointers and two dimensional Array, Array of Pointers, Dynamic Memory Allocation. **6.2 Files:** Types of File, File operation- Opening, Closing, Creating, Reading, Processing File.

Index

3. Control Structures

4. Functions and Parameter

5. Arrays, String Structure and Union

7. Programs

Module 01 - Introduction to Computer, Algorithm & Flowchart

1.1 Basics of Computers:

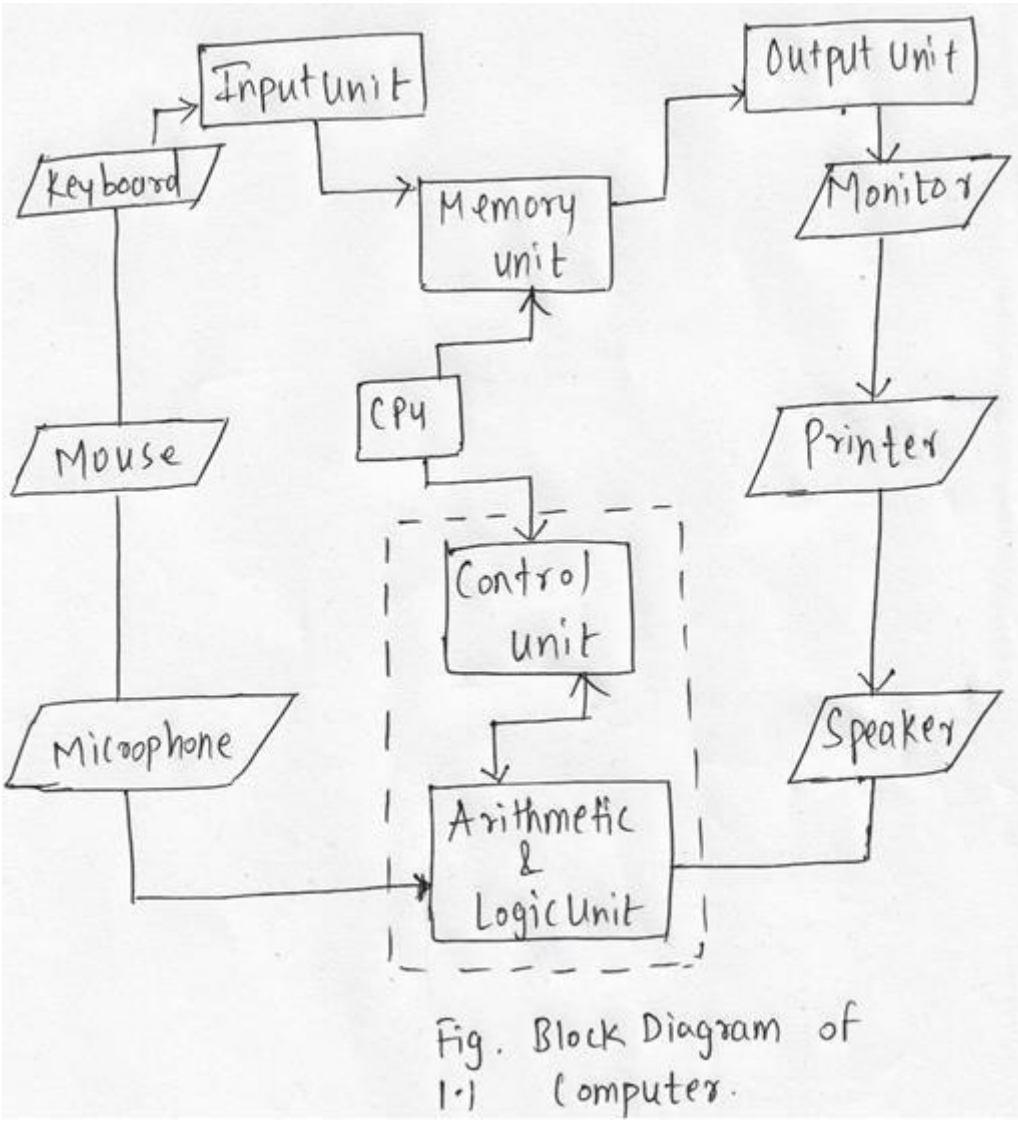


Fig. Block Diagram of Computer.
1-1

- A computer is an electronic device or a digital machine that **reads** data, **stores** data & **processes** the data and produces one or more results.
- Thus a basic computer can perform 3 basic tasks : Input, Processing & Output
- The various units of a computer system are as follows:

a) Input Unit:

- Input is the process of giving data to the computer which will later be processed by the computer. An input unit will thus do the task of giving data to the computer.
- Examples of input devices are keyboards, mouse, scanners etc.

b) Memory Unit:

- Memory unit is the place where **data** & **instructions** are stored.
- It can be considered as a sequence of locations where each location can hold a byte or a word.
- A byte is comprised of **8 bits (1 byte = 8 bits)** which can be either 0 or 1.
- And so data and instructions are stored as a sequence of 1's and 0's in memory.
- The capacity of a memory chip is in the multiples of bytes which the chip can hold. (A typical 1 KB has 1024 bytes or 1024 memory addressable locations).

c) Arithmetic & Logic Unit (ALU):

- This sections deals with all the **Logical** & **Arithmetical** operations on data.
- Operations can be addition, subtraction, division, multiplication, modulus or a combination of all of these.

d) Control Unit:

- It is the core (heart) of any computer system, since all the operations like arithmetic and logic are controlled by the control unit.
- The Control Unit with the ALU forms the Central Processing Unit (CPU) (CU + ALU = CPU)

e) Output Unit:

- The processed data in the ALU may be stored in the memory or may be given to the output device like monitor, printer or speaker etc.
- Data stored in the memory may also be directed to an output device for communication with the user or some other device.

Q) Explain what is a computer with the help of a block diagram?

1.1.1 Turing Model

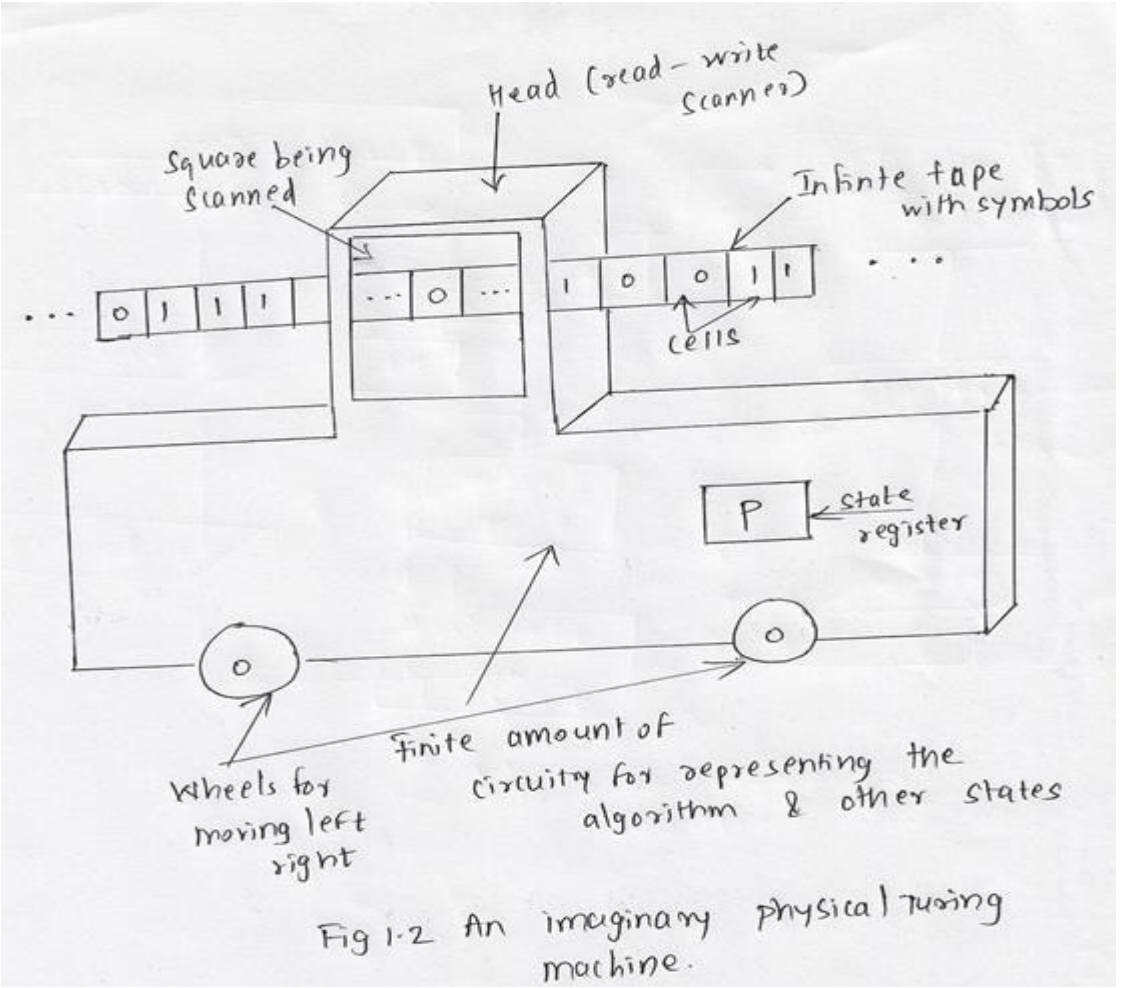


Fig 1.2 An imaginary physical Turing machine.

- A Turing Model or Turing Machine (TM) is a mathematical computational model consisting of an infinite length tape divided into multiple cells on which input is given.

- It has a **head** which reads the input tape and a **state register** which stores the state of the TM.
- After a cell consisting of an input symbol is read by the TM changes its internal state with the help of **state register** & it moves from one cell to the left or right.
- If TM reaches the final state, input string is accepted; otherwise rejected.
- In the above block diagram, the head (read-write scanner) reads the input tape of infinite length.
- The square or a bracket (cell) is being scanned. The TM takes only one symbol at a time.
- The machine changes its state internally with the help of a **P** register.
- The machine moves by having the wheels left or right depending upon the symbol which is read or a sequence of symbols.

Q) Explain Turing Model (TM) with the help of an Example

1.1.2 Von Neumann Model

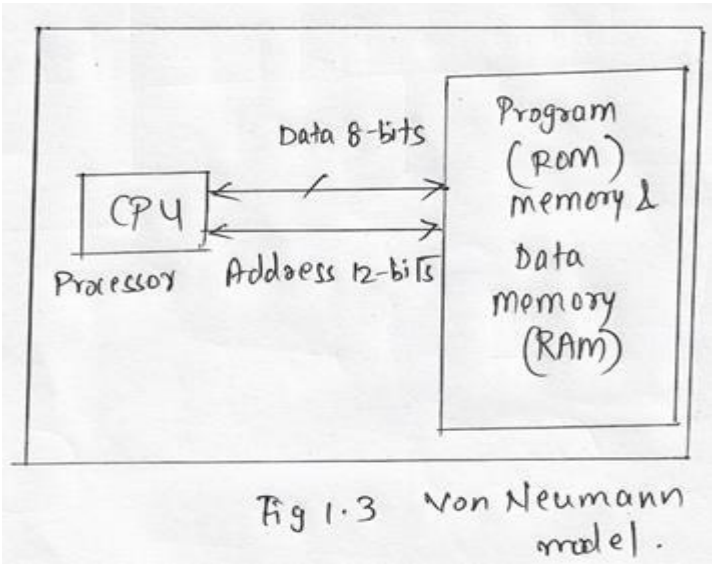


Fig 1.3 Von Neumann model.

- A Von Neumann model consists of a **memory**, **CPU** & **I/O devices**.
- In this model the instruction used to be stored in the memory and then CPU fetches it and executes it one at a time.
- The rate of process execution in this model was one at a time and so the overall memory execution cycle time was slow.
- These machines are called as control flow since they obeys the way a computer system performs; serves one request at a time.
- To overcome this problem on Von Neumann model a parallel processing of computers was developed which were clubbed with serial CPU's.
- This model serves as a base for stored computers which accompanies both data as well as instructions.
- This model assumes two kinds of buses: **Data** bus & **Address** bus between the CPU (Processor) and storage (memory).
- Since, now there are two different buses the data operation and instruction processing has to be done in a sequential way reducing the memory execution cycle.
- This model is more used when there is a need to interface an external memory. (I.e. connecting a secondary storage).

Q) Explain Von Neumann Model with a block diagram.

1.1.3 Basics of Positional Number System

Number systems in computer language are classified into three main types:

Sr. No.	Number System	Description
1	Binary	Base 2. Digits used : 0 and 1
2	Octal	Base 8. Digits used : 0 to 7
3	Hexadecimal	Base 16. Digits used : 0 to 9 and Letters used: A – F

1. Binary Number System:

- Uses **two** digits, **0** and **1**
- Also called as **base 2 number system.**
- Each position in a binary number represents a '**0**' power of the base (2). Example 2^0
- Last position in a binary number represents a '**x**' power of the base (2). Example 2^x where x represents the **last position - 1**.

Example: **Binary Number 10101_2**

Calculating decimal equivalent –

Step	Binary Number	Decimal Number
1	10101_2	$((1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
2	10101_2	$(16 + 0 + 4 + 0 + 1)_{10}$
3	10101_2	21_{10}

1.1.4 Introduction to Operating System & Components

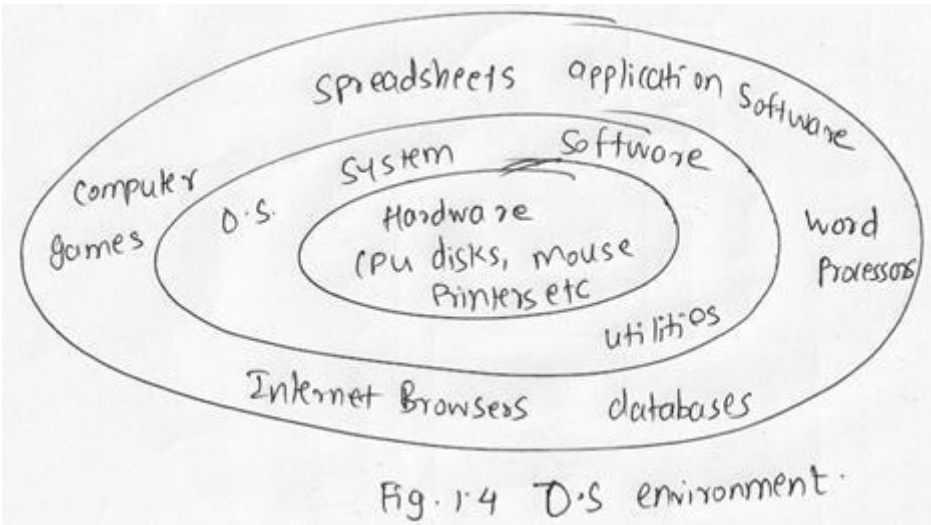


Fig. 1.4 D.S environment.

- An operating system is **system program / software** which controls the execution of application programs& acts as an interface between application and the hardware.
- OS can serve as an extended machine as in where the real processors, hardware & other devices are very complicated and difficult to work with.
- OS hides the hardware & presents programs with nice clean, elegant and simple interfaces to work with.
- OS are **convenient** as they should allow the system to be utilized in a convenient manner.
- OS should use the system resources in an **efficient** manner.
- OS should allow the introduction of new features without any modifications to the underlying OS.
- Examples are Microsoft Windows, Linux, Android, Tizen, Macintosh, iOS etc.

Components of OS

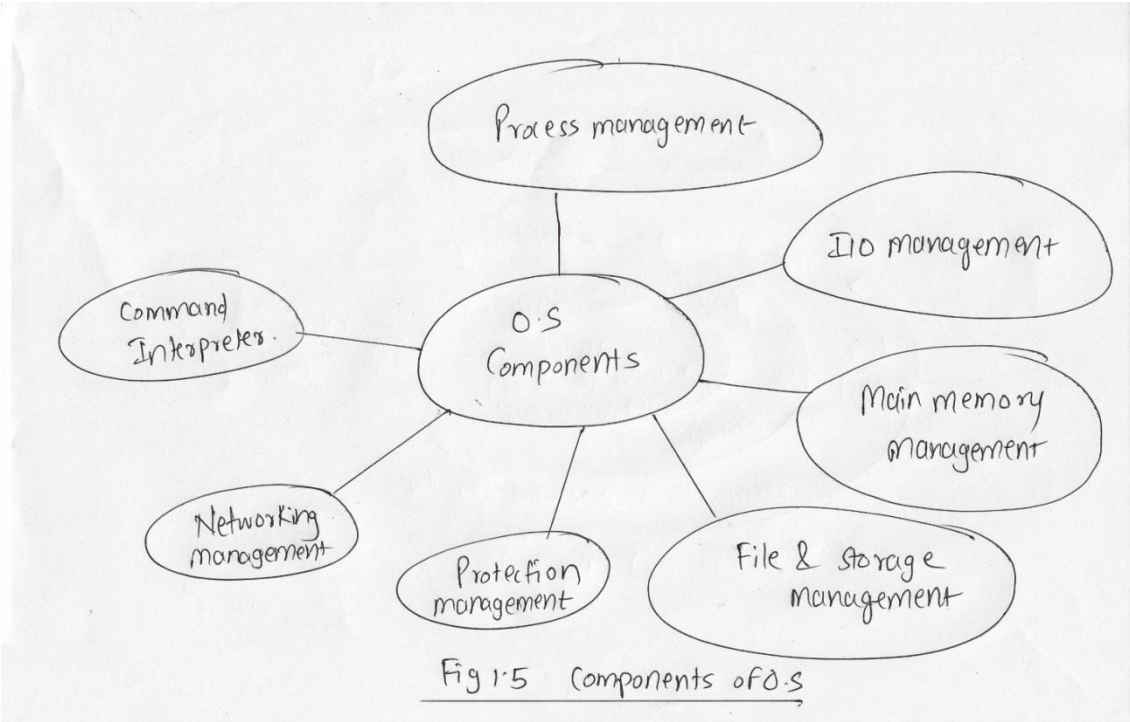


Fig 1.5 Components of O.S

Process Management:

- Process is a program or a part of a program which is loaded in the main memory.
- OS manages its pool of process execution in a time shared manner.

IO Management:

- OS hides the hardware details of the devices and maintains device driver details.
- Manages the main memory by using caching, spooling & buffering techniques.

Main Memory Management:

- Keeps track of which memory is used and to whom it is assigned (ex: windows Task Manager)
- Allocation or De-allocation of memory as needed.

File & Storage Management:

- Allocation, de-allocation and defragmentation of blocks.
- Bad block marking
- Scheduling of multiple IO request to optimize the performance.

Protection Management:

- Protection of hardware resources, Kernel codes, processes, files & data from erroneous and malicious programs.

Network Management:

- Permits communication between computers more importantly between client server OS and Distributed OS.

Command Interpreter:

- OS allows users to interact with the system level (ex. Windows cmd, Linux terminal)
- Provides convenient environment and user friendly commands to users to interact.

Q) What is OS and discuss its components.

1.2 Algorithm

- Algorithm is a step-by-step procedure for solving a particular problem to find a solution.
- It is an English like representation of a program with set of variables and constants (taken for assumptions).
- It does not contain any symbolic notations to express the step.
- Example below is an algorithm to read two **unequal numbers** and print **maximum**

1. Start
2. Read two unequal numbers as **a** and **b**
3. If **a** is greater than **b** go to step 4 else go to step 5
4. Print a. Go to step 6
5. Print b. Go to step 6
6. Stop

Q) What is an Algorithm / Define Algorithm?

1.2 Flowchart

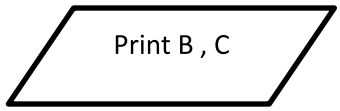
- Flowchart is a pictorial representation of a program.
- It is the graphical representation of steps written in an **algorithm.**
- Flowcharts tend to be more expressive & precise than algorithms which are sometimes ambiguous since natural languages are ambiguous.
- Certain standard symbols used in flowcharts are as follows:

a) Input / Output

- All the input output operations are represented with a parallelogram.

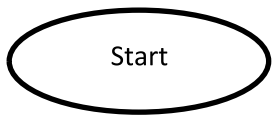
To read a variable X:

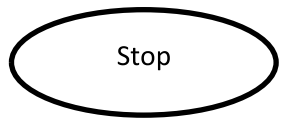
To display variables B and C



b) Start / Stop

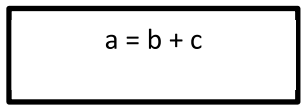
- Flat **ovals** or **rectangular** boxes rounded at edges depict the start / end of programs or procedures.





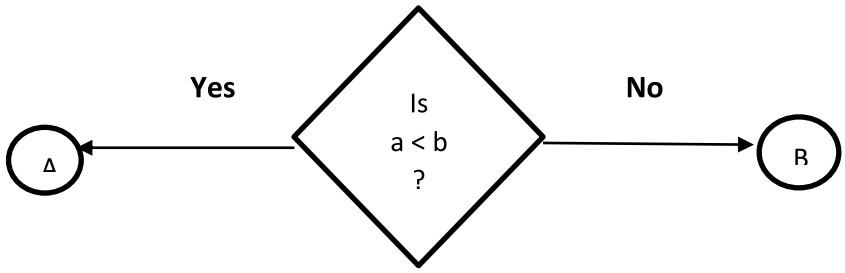
Processes

- Rectangles are used to show processing operations like addition, subtraction, assignment etc.



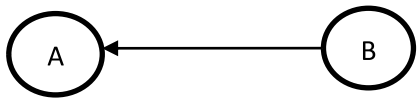
c) Decisions

- Diamond shaped boxes are used to depict decision making
- This test box has **one** or **more** entry points but, it must have **two** or **more** exit points.



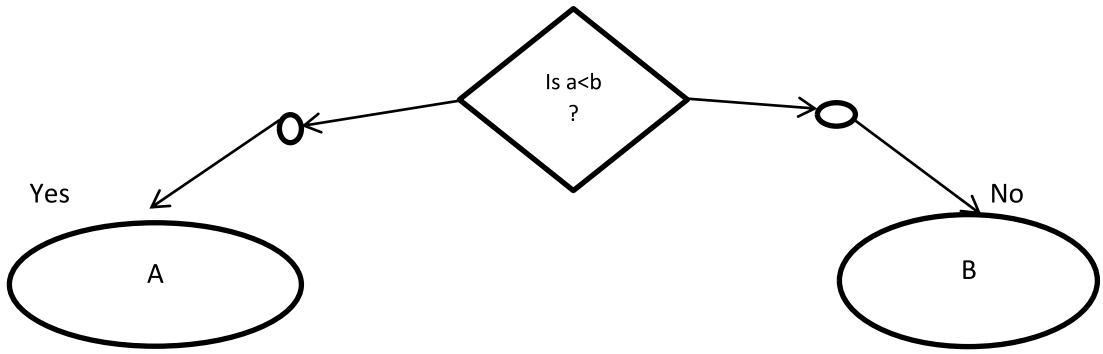
d) Paths

- A path is depicted by an arrow.
- Arrows shows the path to be followed in a flowchart & connect the different flowcharting symbols.

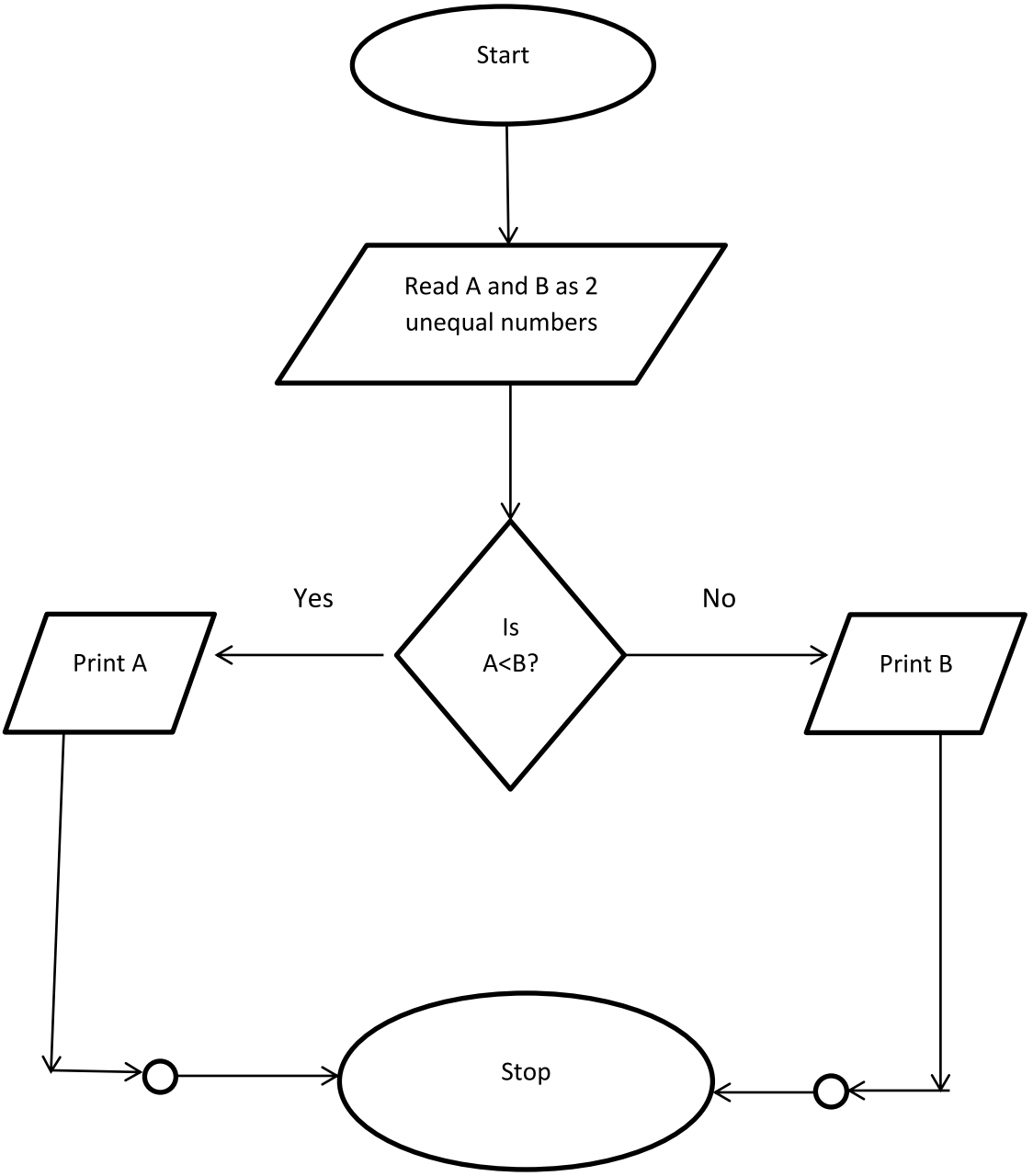


e) Connectors

- Sometimes it might not be possible to connect between two different flowcharts.
- Suppose, two parts of a flowchart may be lying on two different pages. At times, parts of the flowchart to be connected may be lying on the same page but it may not be easy to connect them neatly.
- In such cases, a circle is used as a connector to show how the connection between two different parts of a flowchart



Flowchart to read the two unequal numbers and print the maximum



Important Programs For Practice

A bank of 100+ programs to help students understand and learn the core fundamentals of the C language. Students are expected to thoroughly practice these which will eventually help them in solving problems in theory as well as practical examinations.

Data Input and Output

1. **pow() - program to print square, cube and fourth power of a given number without doing any unnecessary calculations**

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

void main()
{
 float  x ;

 clrscr() ;

 printf("Enter a number: ") ;
 scanf("%f" , &x) ;

 printf("Square = %f \n" , pow(x,2) ) ;
 printf("Cube = %f \n" , pow(x,3) ) ;
 printf("Fourth power = %f \n" , pow(x,4) ) ;

 getch() ;
}
```

Output:

```
Enter a number: 2.5
Square = 6.250000
Cube = 15.625000
Fourth power = 39.062500
```

2. swap - program to swap values of two integers without using a temporary variable

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

void main()
{
  int  x , y ;

  clrscr() ;

  printf("Enter two integers: ") ;
  scanf("%d %d" , &x , &y) ;

  printf("x=%d y=%d \n" , x , y) ;

  x=x+y ;
  y=x-y ;
  x=x-y ;

  printf("After swapping: \n") ;
  printf("x=%d y=%d" , x , y) ;

  getch() ;
}
```

Output:

```
Enter two integers: 5  8
x=5 y=8
After swapping:
x=8 y=5
```

Arrays

1. ***Average - Calculate sum and average of a list of elements. Also count how many elements in the list are less than average and more than average***

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

void main()
{
 int  i , n , less=0 , more=0 ;
 float  x[50] , sum=0 , avg ;

 clrscr() ;

 printf("Enter the number of elements: ") ;
 scanf("%d" , &n) ;
 printf("Enter the elements: \n") ;
 for(i=0 ; i<n ; i++)
 {
  scanf("%f" , &x[i]) ;
  sum=sum+x[i] ;
 }
 avg=sum/n ;

 printf("Sum of elements in list is: %f \n" , sum) ;
 printf("Average of elements in list is: %f \n" , avg) ;

 for(i=0 ; i<n ; i++)
 {
   if(x[i]>avg)
     more++;
   if(x[i]<avg)    /* writing else here would be incorrect */
     less++;
 }

 printf("No. of elements more than average is: %d \n" , more) ;
```

```
  printf("No. of elements less than average is: %d \n" , less) ;

  getch();
}
```

Output:

```
Enter the number of elements: 4
Enter the elements:
2.1
5.6
3
2.3
Sum of elements in list is: 13.000000
Average of elements in list is: 3.250000
No. of elements more than average is: 1
No. of elements less than average is: 3
```

2. Matrix Column Sum - Program to find sum of column elements of all the columns of a given matrix

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

void main()
{
 int  x[10][10] , nr , nc , r , c , sum ;

 clrscr() ;

 printf("Enter the number of rows and columns: ") ;
 scanf("%d %d", &nr , &nc) ;

 printf("Enter elements of the matrix row-wise: \n") ;
 for(r=0 ; r<nr ; r++)
  for(c=0 ; c<nc ; c++)
     scanf("%d" , &x[r][c]) ;

 for(c=0 ; c<nc ; c++)
 {
          sum=0;
  for(r=0 ; r<nr ; r++)
    sum=sum+x[r][c];
  printf("Sum of elements of column no %d is %d \n" , c , sum) ;
 }

 getch();
}
```

Output:

```
Enter the number of rows and columns: 3  3
Enter elements of the matrix:
1  5  8
2  3  4
6  7  9
Sum of elements of column no 0 is 9
Sum of elements of column no 1 is 15
Sum of elements of column no 2 is 21
```

3. ***String remove comma - Program that reads a number greater than or equal to 1000 from the user. The user enters a comma in the input. Print the number without the comma***

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

void main()
{
 int  i ;
 char  n[20] ;

 clrscr() ;

 printf("Enter a number greater than or equal to 1000: ") ;
 gets(n) ;

 printf("Number without comma is:") ;
 for(i=0 ; n[i]!='\0' ; i++)
  if(n[i] != ',')
    putchar(n[i]) ;

 getch();
}
```

Output:

```
Enter a number greater than or equal to 1000: 2,46,798
Number without comma is:246798
```

4. ***Four Experiments are performed, each experiment consisting of six results. The result for each experiment follows. Write a program to compute and display the average of the test results for each experiment.***

1st experiment results	**23.2 31.5 16.9 28.0 26.3 28.2**
2nd experiment results	**34.8 45.2 20.8 39.4 33.4 36.8**
3rd experiment results	**19.4 50.6 45.1 20.8 50.6 13.4**
4th experiment results	**36.9 42.7 20.8 10.2 16.8 42.7**

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

void main()
{
 float  x[4][6] = {23.2 , 31.5 , 16.9 , 28.0 , 26.3 , 28.2 , 34.8 , 45.2 , 20.8 , 39.4 , 33.4 , 36.8 , 19.4
, 50.6 , 45.1 , 20.8 , 50.6 , 13.4 , 36.9 , 42.7 , 20.8 ,
                  10.2 , 16.8 , 42.7} ;
 float  sum ;
 int r , c ;

 clrscr() ;

 for(r=0 ; r<4 ; r++)
 {
  sum = 0 ;
  for(c=0 ; c<6 ; c++)
     sum = sum + x[r][c];
  printf("Test results average for experiment number %d is %f \n" , r+1 , sum/6 );
 }

 getch();
}
```

Output:

```
Test results average for experiment number 1 is 25.683332
Test results average for experiment number 2 is 35.066669
Test results average for experiment number 3 is 33.316666
Test results average for experiment number 4 is 28.350000
```

5. String concatenate and length - Program to concatenate two strings and find string length using functions

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

void  concatenate(char  str1[ ] , char  str2[ ]) ;
int  stringlength(char  str[ ]) ;

void main()
{
 char  str1[50] , str2[50] , str[50] ;

 clrscr();

 printf("Performing string concatenation \n") ;
 printf("Enter two strings:\n") ;
 gets(str1) ;
 gets(str2) ;
 concatenate(str1,str2) ;
 printf("Concatenated string is: ") ;
 puts(str1);

 printf("\nFinding string length \n") ;
 printf("Enter a string: ") ;
 gets(str) ;
 printf("String length is: %d" , stringlength(str) ) ;

 getch();
}

void  concatenate(char  str1[ ] , char  str2[ ])
{
 int  i , j ;

 for(i=0 ; str1[i]!='\0' ; i++) ;

 for(j=0 ; str2[j]!='\0' ; j++ , i++)
   str1[i]=str2[j] ;
```

```
  str1[i]='\0' ;
}

int  stringlength(char  str[ ])
{
  int  i ;

  for(i=0 ; str[i]!='\0' ; i++) ;

  return  i ;
}
```

Output:

```
Performing string concatenation
Enter two strings:
New
York
Concatenated string is: NewYork

Finding string length
Enter a string: Apple
String length is: 5
```

6. String Copying - Program to copy one string into another using a function

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

void  copy(char  str1[ ] , char  str2[ ]) ;
/* This function will copy contents of string 2 in string 1 */

void main()
{
  char  str1[50] , str2[50] ;

  clrscr();

  printf("Enter source string: ") ;
  gets(str2) ;

  copy(str1,str2) ;

  printf("Copied string: %s" , str1) ;

  getch();
}

void  copy(char  str1[ ] , char  str2[ ])
{
  int  i , j ;

  for(i=0 , j=0 ; str2[i]!='\0' ; j++ , i++)
    str1[j]=str2[i] ;

  str1[j]='\0' ;
}
```

Output:

```
Enter source string: SPA sem 2
Copied string: SPA sem 2
```

7. *Delete duplicate array elements - Program to delete all the duplicate values from a given array*

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

void  main()
{
 int  i , j , k , m , n , x[50] ;

 clrscr() ;

 printf("Enter the number of elements: ") ;
 scanf("%d" , &n) ;
 printf("Enter the elements: \n") ;
 for(i=0 ; i<n ; i++)
  scanf("%d" , &x[i]) ;

 printf("Original array is as shown: \n") ;
 for(i=0 ; i<n ; i++)
  printf("%d " , x[i]) ;
 printf("\n") ;

 /* Removing duplicates */
 m=n ;
 for(i=0 ; i<n-1 ; i++)
  for(j=i+1 ; j<m ; j++)
   if(x[i]==x[j]) /* true if duplicate found */
   {
          for(k=j+1 ; k<m ; k++)
      x[k-1]=x[k] ;
          m-- ;
   }

 /* At this stage, array length is m not n */
 printf("New array with no duplicates is as shown: \n") ;
 for(i=0 ; i<m ; i++)
  printf("%d " , x[i]) ;
```

```
  getch();
}
```

Output1:

```
Enter the number of elements: 8
Enter the elements:
2 1 4 2 9 6 4 5
Original array is as shown:
2 1 4 2 9 6 4 5
New array with no duplicates is as shown:
2 1 4 9 6 5
```

Output2:

```
Enter the number of elements: 4
Enter the elements:
2 8 4 5
Original array is as shown:
2 8 4 5
New array with no duplicates is as shown:
2 8 4 5
```

8. *Delete element from array - Program to delete all occurences of given number from an array*

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

void  main()
{
 int  i , key , n , position , flag , x[50] ;

 clrscr() ;

 printf("Enter the number of elements: ") ;
 scanf("%d" , &n) ;
 printf("Enter the elements: \n") ;
 for(i=0 ; i<n ; i++)
  scanf("%d" , &x[i]) ;

 printf("Enter the element to be deleted: ") ;
 scanf("%d" , &key) ;

 position=0;
 do
 {
  flag=0;

  for(i=position ; i<n ; i++)
   if(x[i]==key)   /* true if element found */
   {
            position=i ;
            flag=1 ;
            break ;
   }

  /* Element found at 'position' */
  if(flag==1)
  {
   for(i=position ; i<n-1 ; i++)
    x[i]=x[i+1];
```

```
      n--;
    }
  }
  while(flag==1);

  printf("New array is as shown: \n") ;
  for(i=0 ; i<n ; i++)
    printf("%d " , x[i]) ;

  getch();
}
```

Output:

```
Enter the number of elements: 5
Enter the elements:
2 4 4 6 4
Enter the element to be deleted: 4
New array is as shown:
2 6
```

9. *Delete element from array - Program to delete all occurences of given number from an array*

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

void  main()
{
 int  i , key , n , position , flag , x[50] ;

 clrscr() ;

 printf("Enter the number of elements: ") ;
 scanf("%d" , &n) ;
 printf("Enter the elements: \n") ;
 for(i=0 ; i<n ; i++)
  scanf("%d" , &x[i]) ;

 printf("Enter the element to be deleted: ") ;
 scanf("%d" , &key) ;

 position=0;
 do
 {
  flag=0;

  for(i=position ; i<n ; i++)
   if(x[i]==key)   /* true if element found */
   {
            position=i ;
            flag=1 ;
            break ;
   }

  /* Element found at 'position' */
  if(flag==1)
  {
   for(i=position ; i<n-1 ; i++)
    x[i]=x[i+1];
```

```
      n--;
    }
  }
  while(flag==1);

  printf("New array is as shown: \n") ;
  for(i=0 ; i<n ; i++)
    printf("%d " , x[i]) ;

  getch();
}
```

Output:

```
Enter the number of elements: 5
Enter the elements:
2 4 4 6 4
Enter the element to be deleted: 4
New array is as shown:
2 6
```

10 Matrix Diagonal - Program to check whether a given matrix is diagonal matrix

A diagonal matrix is that square matrix whose diagonal elements from upper left to lower right are non-zero and all other elements are zero. For example,
2 0 0
0 4 0
0 0 6

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

void main()
{
 int  x[10][10] , nr , nc , r , c , flag ;

 clrscr() ;

 printf("Enter the number of rows and columns: ") ;
 scanf("%d %d", &nr , &nc) ;

 if(nr==nc)                    /* checking for square matrix */
 {
  printf("Enter elements of the matrix: \n") ;
  for(r=0 ; r<nr ; r++)
   for(c=0 ; c<nc ; c++)
     scanf("%d" , &x[r][c]) ;
  flag=1 ;
  for(r=0 ; r<nr ; r++)
   for(c=0 ; c<nc ; c++)
     if(r==c) /* true for diagonal elements */
     {
              if(x[r][c]==0)
                flag=0;
     }
     else
     {
              if(x[r][c]!=0)
                flag=0;
```

```
      }
   if(flag==1)
     printf("The matrix is diagonal") ;
   else
     printf("The matrix is not diagonal") ;
 }
 else
   printf("The matrix is not a sqaure matrix") ;

 getch() ;
}
```

Output1:

```
Enter the number of rows and columns: 3 3
Enter elements of the matrix:
11 0 0
0 22 0
0 0 44
The matrix is diagonal
```

Output2:

```
Enter the number of rows and columns: 3 3
Enter elements of the matrix:
11 0 2
0 22 0
0 0 44
The matrix is not diagonal
```

Output3:

```
Enter the number of rows and columns: 3  3
Enter elements of the matrix:
0 0 0
0 9 0
0 0 8
The matrix is not diagonal
```

11. String - Program to read a positive integer and print its digits in words. For example, if the input is 126 the o/p should be One Two Six

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

void main()
{
 char  n[20] ;
 int i;

 clrscr();
 printf("Enter a positive integer: ");
 gets(n);
 printf("%s in words: " , n);
 for(i=0 ; n[i]!='\0' ; i++)
 {
  switch(n[i])
  {
    case '0': printf("Zero ");          break;
    case '1': printf("One ");           break;
    case '2': printf("Two ");           break;
    case '3': printf("Three ");         break;
    case '4': printf("Four ");          break;
    case '5': printf("Five ");          break;
    case '6': printf("Six ");           break;
    case '7': printf("Seven ");         break;
    case '8': printf("Eight ");         break;
    case '9': printf("Nine ");          break;
  }
 }
 getch();
}
```

Output:

```
Enter a positive integer: 1760
1760 in words: One Seven Six Zero
```