STRUCTURES & FILE MANAGEMENT

Definition:

Structures are the Collection of data of same type or different type. The Structures Provides a mechanism for the Programmer to Create his/her own data type Called as User defined data Type.

Syntax:

Declaration of Structures:

```
Struct tag
{
    member-declarations;
};
```

Here, The Struct is a Keyword in C for Structure declaration.

The tag is the name of the structure declared.

The Variables declared within the flower brackets in the above Structure are Called members, Components.

Example: Keyword

```
Struct tree
{
    Char name[6];
    float height;
    int nodes;
};
```

Name of Structure.

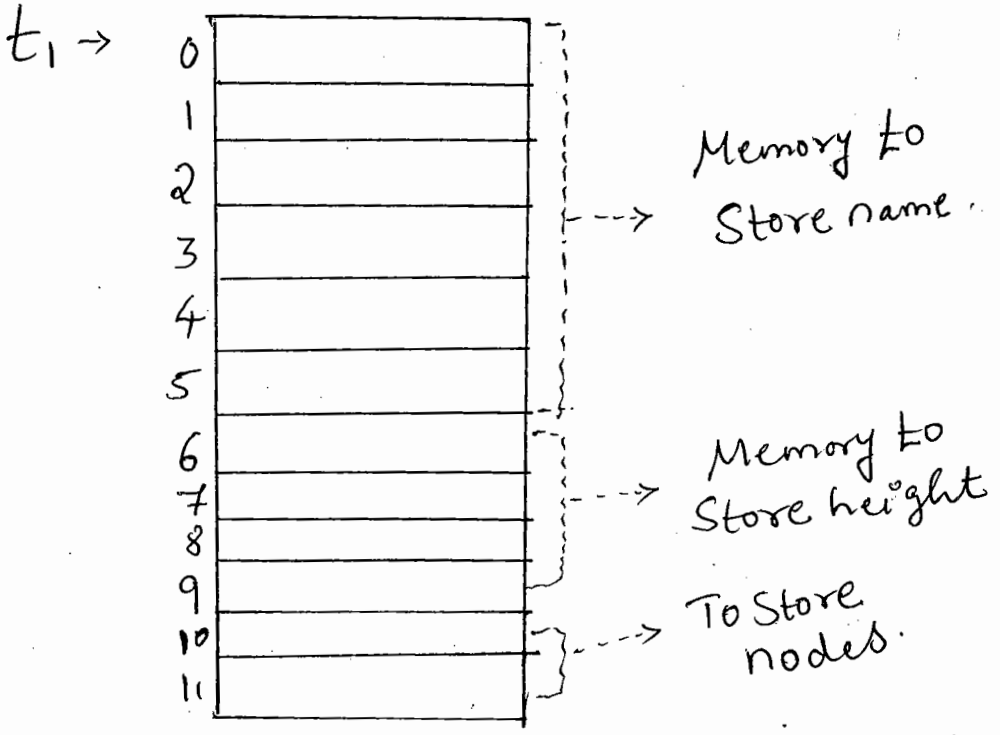
Structure Members.

Accessing Members of Structure:-

To access the members of the Structure, a Variable for the Structure has to be Created.

```
Struct tree t1;
```

Variable.



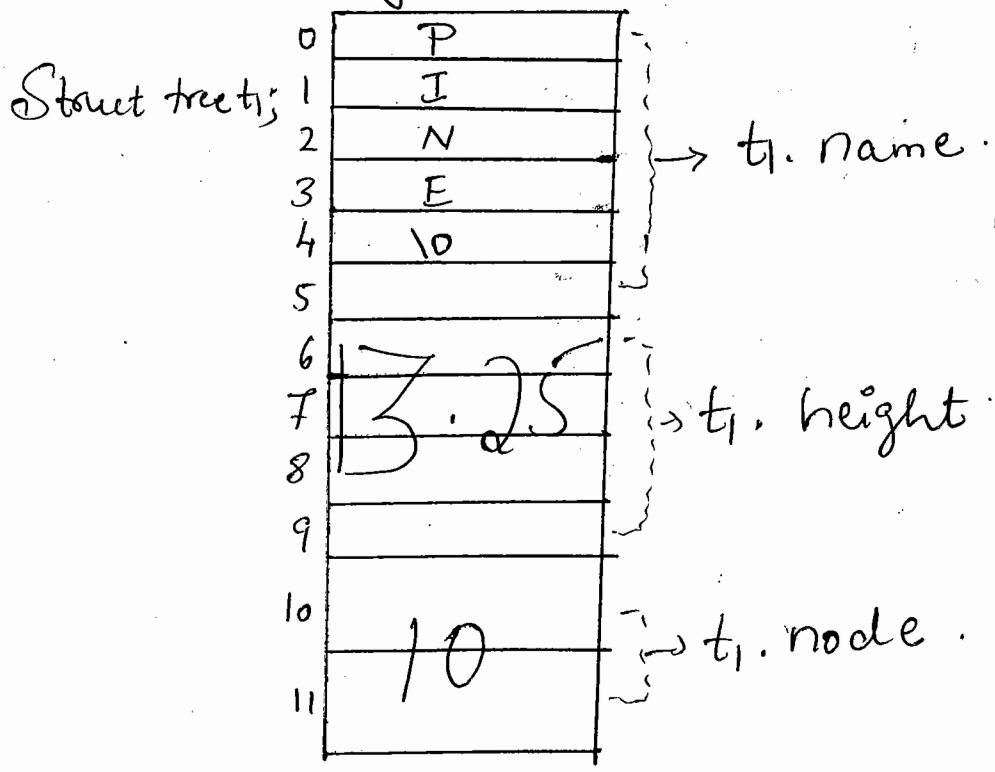
Initialize the Values for Members:-

The . (Dot) Operator is called the member access operator which is used to access the members of the Structure.

The assigning the Values to the members of Structures called as initialization.

```
Struct tree t1;
t1.name = "PINE";
t1.height = 13.25;
t1.node = 10;
```

Here, t_1 is the Variable Created for the Structure to initialize the Values by accessing the members of Structure.



Example Program:-

```
#include <Stdio.h>
#include <conio.h>
Struct Person
{
    Char name[20];
    int age;
    Char gender;
};
Void main()
{
    Struct Person P;
    Printf(" Enter Person details ");
    Scanf("%.s %d %.s", &P.name, &P.age,
                        P.gender);
    Printf(" The details are ");
    Printf(" Name=%.s Age=%d Gender=%.s",
            P.name, P.age, P.gender);
}
```


OUTPUT: Enter details
ABC 18 M.
The details are
Name = ABC Age = 18 Gender = M.

Structures and Functions:

As the normal Variables Passed to the functions, it is Possible to Pass even the Structure Variables to functions. Passing a Structure to a function Syntax is Shown below.

Syntax:

```
type functionname (Struct tag Param)
{
    ------
    ------ function body.
}
```

Where, tag is the Structure name
Param is function Parameter name,
ie Structure Variable.

The Structures Can be Passed to functions in two ways as.

(i) Pass by Value
(ii) Pass by Refevence.

The members of Parameter Param is accessed in the function through (.) operator as Param.member1, Param.member2, etc..

Passing Structure by Value:-

A Structure Variable Can be Passed to the function as an argument, similar to the normal Variables Passed to functions.

Example:-

```
#include <Stdio.h>
Void display(Struct book b1);
 Struct book
 {
    Char bname[15];
    int  bno;
 };
Void main()
{
    Struct book b;
    Printf(" Enter book name and number");
    Scanf(" %.s%.d", &b.bname, &b.bno);
    display(b);
}
Void display( Struct book b1)
{
   Printf(" Name = %.s", b1.bname);
   Printf(" Number = %.d", b1.bno);
}
```

Passing Structures By Reference :-

Similar to the normal Variables Passed to the function, It is Possible to Pass Structures by reference. Here in Pass by reference the address of the Structure is Passed, to Collect the address the Pointer is Used.

Example :-

```
#include <Stdio.h>
Void display (Struct book *b1);
Struct book
{
    Char bname[20];
    int bno;
};
Void main()
{
    Struct book b;
    Printf("Enter book name & Number");
    Scanf("%s%d", &b.bname, &b.bno);
    display (&b);
}
```

```
Void display(Struct book *b1)
{
   Printf(" Book name = %.S", b1 → bname);
   Printf(" Book num = %.d", b1 → bno);
}
```

Output:

Enter book name & number.

PCD 01

Book name = PCD

Book num = 01

Note: To access the members of Structure arrow operator (→) is used when the address is passed to function & accessed through pointers. ie for Reference.

If the Structure is Passed by Reference, Changes made in Structure Variable in Called function definition will be reflected & Changed in the Original Structure Variable in Calling function also.

Array Of Structures:

The array of Structures, which is similar to creation of array of integers, strings, etc. we can declare array of Structures in one-dimensional or multidimensional array of Structures.

Syntax:

(i) Struct tag arr[exp];

where, tag - array of Structures Structure name.

arr - array of Structures.

exp -> Size of an array.

(ii) Struct tag arr[exp1][exp2]....[expn];

Here we can declare multiple array of Structures of any dimension in a single declaration.

Accessing array of Structure elements:

An element of one dimensional array arr of Structures can be accessed as arr[i].

Ex:- arr[i].member;

Example Program:-

```
#include<stdio.h>
#include<conio.h>
Struct Person
{
   Char name[15];
    int age;
};
Void main()
{
    Struct Person P[2];
    int i;
    for(i=0 ; i<=1 ; i++)
    {
      Printf("Enter name and age");
      Scanf("%s %d", &P[i].name, &P[i].age);
    }
    for(i=0 ; i<=1 ; i++)
    {
      Printf("Name = %s", P[i].name);
      Printf("Age = %d", P[i].age);
    }
}
```

Output :-

Enter ~~Person~~ Name and age

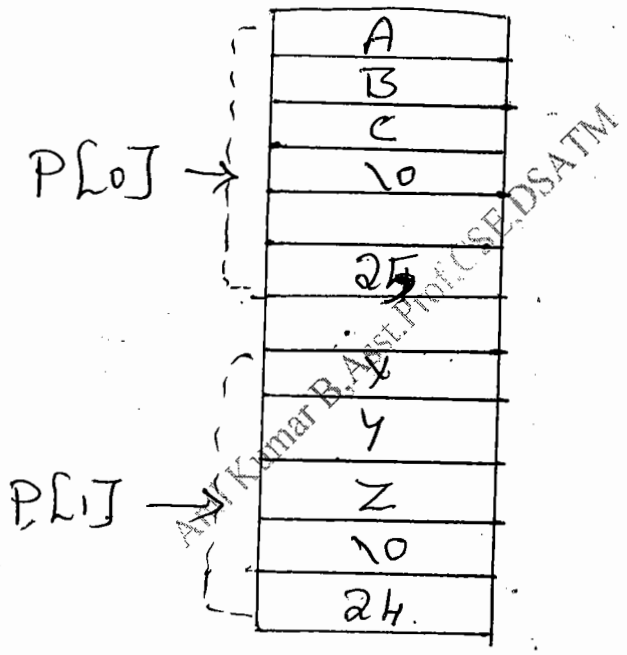
ABC 25

Enter Person name and age

XYZ 24

Name = ABC Age = 25

Name = XYZ Age = 24.



Type definition of Structure:

The typedef feature Provides an alternative name to an existing datatype. and also enables to declare new types for the Good readability of Program.

The typedef is the Keyword used to declare the Variables.

Syntax:

```
typedef int marks;
```

Here, now marks is the new data type declared we can declare variables using this marks as shown below.

```
marks Phy, Chem, CPD;
```

For Structure typedef is created as:-

(i)
```
typedef Struct
{
    Char name[15];
    int rollno;
} Student;
```


or

(ii)
```
typedef Struct stinfo
{
    Char name[15];
    int rollno;
} Student;
```

Here, stinfo is variable for the Structure Created .ie the type defname. of the Structure.

Example :-

```
#include <stdio.h>
typedef Struct
{
    Char name[15];
    int age;
} Person;
Void main()
{
    Person P = {"ABC", 21};
    Printf("Name = %s", P.name);
    Printf("Age = %d", P.age);
}
```


OUTPUT: Name = ABC
Age = 21.

Note:-

```
typedef Struct Person
{
    char name[15]; int age;
} Person;
```

If typedef name and Structure name is same we can omit the Structure name.

Nested Structures : (Structure within a Structure).

→ Having a Structure Variable as data member inside another Structure.

```
Struct date
{
    int dd, mm, YYYY;
};
Struct Employee.
{
    Char ename[50];
    int eid;
    Struct date doJ;
};
```

doJ is Variable of Structure date.

→ Declaring Structure inside another Structure.

```
Struct Employee
{
    Char ename[25];
    int eid;
    Struct date
    {
        int dd, mm, YYYY;
    } doJ;
};
```

doJ is Variable of date Structure.

FILE MANAGEMENT.

Definition of File:

A File is a logical unit of data Used to Store related information.

* The information on Secondary Storage as Harddisk, magnetic tape, etc is arranged inthe form of files.
* The OS and Various applications on the Computer are all stored as files.

Types of Files:

A file is a sequence of characters. Depending on contents of files there are two types as binary and text files.

Text files:-

AS name indicates text file contains textual information such as Printable characters letters, digits, Special Symbols etc.

Binary Files:-

A Binary file is a Sequence of ASCII Values.

A binary file Stores the internal (Binary) representation of data to a disk file.

-> Numeric data requires less Space in a binary file than in text file.

-> The fread & fwrite functions Used for binary file I/O. Can read/write

Opening and Closing of Files:-

(i) fOpen():

The function fopen() Creates a new file or opens an existing file.

Syntax:-

```
FILE *fopen(Const Char *filePath, Const
                                Char *mode);
```

filePath → location of file.

mode → access mode of file (Refer Table)

Example:-

```
#include <Stdio.h>
Void main()
{
  FILE *fp;
  fp = fopen("abc.txt","r");
}
```

fopen returns NULL on Failure, Pointer on Success.

The modes of File:-

MODE	DESCRIPTION.
r	OPens an existing File.
w	OPens a text file for writing, if file does not exist, new file is Created.
a	OPens a text file for appending (writing at end of existing file) & Create file if does not exist.
r+	OPens a text file for reading & writing.
w+	OPens a file for reading & writing and Creates file if does not exist.
a+	OPen for reading & writing/appending, reading Start from begining but writing Can only be appended at the end only.

(ii) f Close():

The function Closes the file which has been opened using fopen.

Syntax:

```
int fclose (FILE *Stream);
```

The argument for fclose is FILE Pointer given by fopen() function.

Example:-

```
#include<stdio.h>
Void main()
{
    FILE *f;
    f = fopen ("xyz.txt", "r");
    fclose (f);
}
```

↓ File Pointer.

Input and Output Operations on Files:

(i) fScanf():

The fscanf() function can be used to read data from the file.

Syntax:-

```
int fscanf(FILE *stream, Const Char *format,
                                    ...);
```

Where,

First argument → File Pointer given by fopen.

Second argument → Format Specifiers

Third argument → list of Variables.

Example:-

```
#include<stdio.h>
Void main()
{
    float a;
    Char s[5];
    FILE *fp;
    fp= fopen("abc.txt","r");
    fScanf(fp, "%f", &a);
    fscanf(fp, "%s", s);
    fclose(fp);
    Printf("Contents are %f %s", a, s);
}
```


OUTPUT:

Contents are 10.000000 VTU

The file abc.txt Contains.

10.0 VTU

Here, the Contents are read from the file abc.txt and displayed on the output screen using Printf.

(ii) fPrintf():

The function fPrintf() can be used to write contents into the file.

Syntax:

```
int fPrintf(FILE *Stream, Const Char *format,
                                        ....);
```

First argument → File Pointer by fOpen()

Second argument → format Specifier

Example:

```
#include<Stdio.h>
Void main()
{
   Char St[50];
   FILE *fp;
  fp = fOpen("abc.txt","W");
 fPrintf(fp, "%S", "HELLO WELCOME");
  fClose(fp);
}
```


The Contents of abc.txt is

abc.txt

```
HELLO WELCOME.
```

(iii) fgetc ():

The fgetc() is an unformatted input function used to read a character from the file.

Syntax:

```
int fgetc (FILE *Stream);
```

The argument is the File Pointer given by fopen() function. The function returns the character read in integer form.

Example:

```
#include <stdio.h>
Void main()
{
  FILE *fp = fopen ("a.txt", "r");
  int Ch=fgetc (fp);
  While ( Ch != EOF)
  {
     Printf("%c", Ch);
      Ch=fgetc (fp);
  }
  fclose (fp);
}
```


OUTPUT:

```
HELLO VTU
```

(iv) fPutC():

The fPutC() function is used to write a character into the file.

Syntax:

```
int fPutC(int c, FILE *Stream);
```

First argument → Char to be inserted to file

Second argument → File Pointer returned by fOpen()

Example:

```
#include <Stdio.h>
Void main()
{
    FILE *fp;
    int Ch;
    fp=fOpen("a.txt","W");
    for(Ch=65; Ch<=90; Ch++)
    {   fPutC(Ch, fp);
    }
    fclose(fp);
}
```


The Contents of file a.txt

```
ABCDEFGHIJKLMNOP..... Z.
```

(v) fgets():

The function fgets() reads sequence of characters as string from file.

Syntax:

```
Char *fgets(Char *str, int n, FILE *Stream);
```

First argument → Where the read arguments to be stored

Second argument → Number of bytes to be read

Third argument → File Pointer given by fopen()

Example:

```
#include<Stdio.h>
Void main()
{
    FILE *fp;
    Char s[100];
    fp= fopen("a.txt","r");
    fgets(s, 100, fp);
    Printf("%s\n", s);
    fclose(fp);
}
```


OUTPUT:

```
HELLO WELCOME.
```

a.txt

HELLO WELCOME.

(vi) fputs():

The fputs() function writes String of Characters into the file.

Syntax:

```
int fputs(const char *s, FILE *stream);
```

First argument → Data that needs to be written.

Second argument → File Pointer given by fopen().

Example:-

```
#include <stdio.h>
Void main()
{
   FILE *fp;
   fp = fopen("a.txt", "w+");
   fputs("WELCOME TO VTU", fp);
   fputs("BELGAUM", fp);
   fclose(fp);
}
```


The Content of a.txt is

```
WELCOME TO VTU
BELGAUM.
```

Programming Examples:-

Program-1 : [STRUCTURES]

C-Program to maintain a record of n Student details using array of structures with four fields (rollnumber, name, marks & grade). Print marks of Student, given Student name as input..

```
#include <Stdio.h>
#include <String.h>
Struct Student
{
    int rollno, marks;
    Char name[25], grade[1];
};
Void main()
{
    int i, n, found=0;
    Struct Student S[20];
    Char Sname[25];
    Printf("enter number of Students");
    Scanf("%d", &n);
    for(i=0; i<n; i++)
    {
```

```
Printf(" Enter %d Student details ", i+1);
Printf(" Enter rollno and name ");
Scanf(" %d %s", &SLi].rollno, &SLi].name);
Printf(" Enter marks and grade ");
Scanf(" %d %s", &SLi].marks, SLi].grade);
}
Printf(" Student Details are ");
Printf("\n Rollno \t Name \t marks \t Grade ");
Printf(" %d \t %s \t %d \t %s \t ", SLi].rollno,
                SLi].name, SLi].marks, SLi].grade);
Printf(" Enter Student name to Print marks ");
Scanf(" %s", Sname);
for(i=0 ; i<n ; i++)
{
    if ( Strcmp (SLi].name, Sname)==0)
    {
        Printf(" marks of Student is ", SLi].marks);
            found = 1;
    }
}
```

```
    if (found == 0)
    {
        Printf(" Student not found");
    }
}
```

OUTPUT:

```
Enter number of Students
2
Enter 1 Student details
 Enter rollno & name
  01   ABC
 Enter marks & grade
  50   A
Enter 2 Student details
 Enter rollno & name
  02   XYZ
 Enter marks & grade
  80   B
 Student Details are
   Rollno    Name    marks   Grade
     01      ABC      50       A
     02      XYZ      85       B
 Enter Student name to Print marks
 XYZ
 Marks of Student is: 85
```

Program 2:-

C- Program to illustrate Structure within Structure.

```
#include<Stdio.h>
Struct Employee
{
    Char ename[20];
    int eid;
     Struct date
     {
       int dd, mm, YYYY;
     }doj;
};
Void main()
{
    Struct Employee e1;
    e1.doj.dd = 05;
    e1.doj.mm = 01;
    e1.doj.yyyy = 2013;

   Printf("Date of Joining %d", e1.doj.dd);
   Printf("month of Join %d", e1.doj.mm);
   Printf("Year of Join %d", e1.doj.yyyy);
}
```

OUTPUT:

Date of Joining 05

Month of Join 01

Year of Join 2013.

Program-3 [FILES]

C- Program to Copy Content of one File to another File.

```
#include <stdio.h>
#include <conio.h>
Void main()
{
   FILE *f1, *f2;
   Char ch;
   f1 = fopen ("abc.txt","r");
   f2 = fopen ("xyz.txt","w");
   While ((ch = fgetc(fp)) != EOF);
   fputc (ch, f2);
   fclose (f1); fclose(f2);
}
```


abc.txt

```
WELCOME TO VTU
BELGAUM.
```

xyz.txt

```
```

← Before execution.

xyz.txt

```
WELCOME TO VTU
BELGAUM.
```

← After execution.

Program-4

C- Program to Count Numbers of Characters, Lines and White Spaces from a File.

```
#include <stdio.h>
Void main()
{
    FILE *f;
    Char c;
    int lines=0, ch=0, spaces=0;
    f= fopen("abc.txt", "r");
    while ((c=fgetc(f))!= EOF)
    {
        Switch (c)
        {
            Case '\t':
            Case ' ': spaces++;
                      break;
            Case '\n': lines++;
                       break;
            default: ch++;
                     break;
        }
    }
    fclose(f);
```

```
Printf(" No. of Characters = %d ", ch);
Printf(" No. of lines = %d \n", lines);
Printf(" No. of Spaces = %d", Spaces);
}
```

OUTPUT:

abc.txt

```
WELCOME TO VTU
BELGAUM BANGALORE.
```

No. of Characters = 28

No. of lines = 2

No. of Spaces = 3