Module 5

Structure, Union and Enumerated data typeStructure:

- Structure is basically a user-defined data type that can store related information together.
- The major difference between structure and an array is that, an array contains related information of the same data type.

Structure declaration:

A structure is declared using the keyword struct followed by a structure name .all the variable of the structure is declared within the structure.

Syntax 1:

```
Struct structure name
{
Data type var-name;
Data type var-name;
……..
};
```

Example:

```
Struct student
{
    int r_no;
    char name[20];
    char course[20];
    float fees;
};
```

Now ,structure has become user defined data type .exach var-name declared within a structure is called a member of the structure.

Syntax 2:

```
Struct student
{
int r_no;
char name[20];
char course[20];
float fees;
}stud1,stud2;
```

In declaration ,we declare two variable stud1 and stud2 of the structure student.So,if you want Tto declare more than one variable of the structure,then separate the variable using comma.

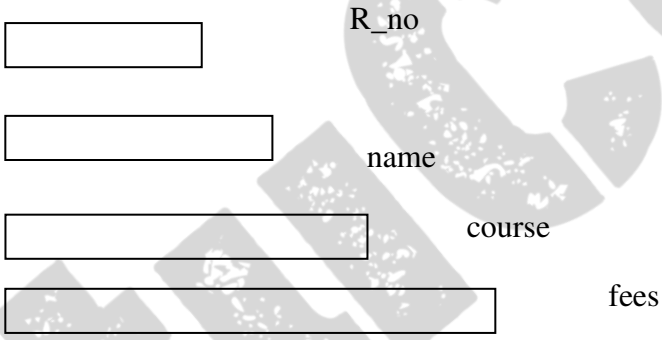


fig: memory allocation for a structure variable

Example1: Declare a structure to store information about the points in the coordinate system

Solution:

```
Struct point
{
    int x;
    int y;
}
```

Example2: Declare a structure to store customer information .

Solution:

```
Struct customer
{
    int cust_id;
    char name[20];
   char address[20];
   long int phone_num;
   int DOB;
}
```

Typedef Declaration:

The typedef (derived from type definition) keyword enables the programmer to create a new type name for an existing data type. By using typedef, no new data is created; rather an alternate name is given to a known data type.

Syntax:

Typedef existing data type new data type;

Example:

Typedef int INTEGER;

Then INTEGER is the new name of data type int. To declare variables using the new data type name, precede the variable name, precede the variable name with the data type name.Therefore,to define an integer variable, we may now write

INTEGER num=5;

Initialization of structure:

Initializing a structure means assigning some constants to the members of the structure. When the user does not explicitly initialize the structure, then C automatically does that.For int and float members, the values are initialized to zero and char and string members are initialized to the '\0' by default.

Syntax:

```
Struct  struct_name
{
Data_type member_name1;
Data_type member_name2;
Data_type member_name3;
…….
}struct_var={constant1,constant2,constant3,….};
```

OR

```
Struct struct_name
{
Data_type ember _name1;
Data_type ember _name2;
Data_type ember _nam3;
…….
};
Struct struct_name struct_var={constant1,constant2,constant3,…..};
```

Example:

```
Struct student
{
int rollno;
char name[20];
char course[20];
float fees;
}stud1={01,”Rahul”,”BCA”,45000};
```

OR

```
Struct student
{
int rollno;
char name[20];
char course[20];
float fees;
}
Struct student stud1={01,"Rahul","BCA",45000};
```

Accessing the members of a structure:

A structure member variable is generally accessed using a '.'(dot) operator.

Syntax:

Struct_var.member_name;

The dot operator is used to select a particular member of the structure.To assign value to the individual data members of the structure variable stud1.

```
Stud1.rollno=01;
Stud1.name="Rahul";
Stud1.course="BCA";
Stud1.fees=45000;
```

To input values for data members of the structure variable stud1 we can write,

```
Scanf("%d",&stud1.rollno);
Scanf("%s",&stud1.name);
```

Similarly,to print the values of structure variable stud1,we may write,

```
Printf("%s",stud1.course);
```

Printf("%f",stud1.fees);

Copying and comparing structure:

Assign a structure to another structure of the same type.

Example:

Two structure Variables stud1 and stud2 of type struct student is given as:

Struct student stud1={01,"Rahul"."BCA",45000};

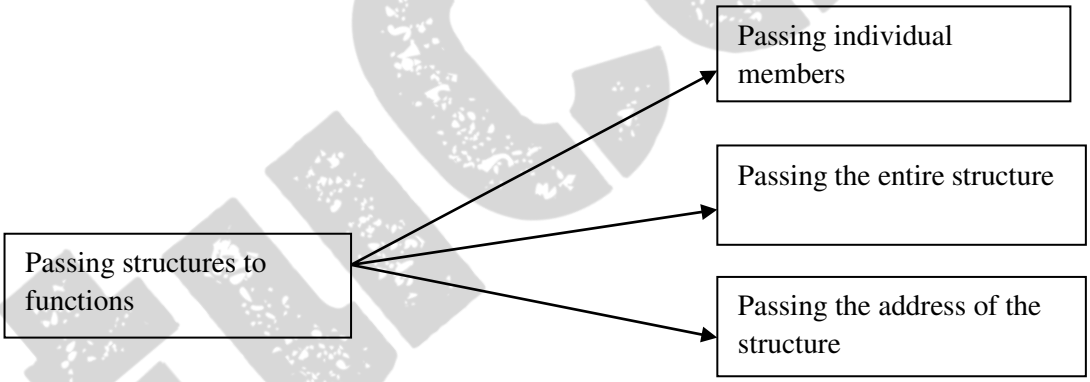
Then to assign one structure variable to another we will write stud2=stud1;

Nested structures:

Structure can be placed within another structure, i.e., a structure may contain another structure as its member. A structure that contains another structure as its member is called a nested structure.

Structures and functions:

A function may access the members of a structure in three ways as



Passing individual members:

- To pass any individual member of the structure to a function, we must use the direct selection operator to refer to the individual members for the actual parameters.
- The called program does not know if the two variables are ordinary variables or structure members.

Example:

```
#include<stdio.h>
Typedef struct
{
    int x;
```

```
    int y;
}POINT;

void display(int,int);
main()
{
     POINT p1={2,3};
    display (p1.x,p1.y);
    return 0;
}
void display(int a,int b)
{
     printf("the coordinates of the point are"%d%d",a,b);
}
```

Output:

The coordinates of the point are : 2 3

Passing the entire structure:

Just like any other variable, we can pass an entire structure as a function argument. when a structure is passed as an argument, it is passed using the call by value method, i.e., a copy of each member of the structure is made.

The general syntax for passing structure to a function and returning a structure can be given as,

Struct struct_name func_name(struct struct_name struct_var);

Example:

```
#include<stdio.h>

typedef struct

{

    int x;

    int y;

}POINT;
```

```
void display(POINT);
main()
{
    POINT p1={2,3};
   display (p1);
   return 0;
}
void display(POINT p)
{
    printf("the coordinates of the point are"%d%d",a,b);
}
```

Output:

The coordinates of the point are :2 3

Passing structures through pointers:

Passing large structures to function using the call by value method is very inefficient.Thereferore it is preferred to pass structures through pointers

Syntax:

```
Struct struct_name
{
Data type member_name1;
Data type member_name2;
Data type member_name3;
……
}*ptr;
```

OR

Struct struct_name *ptr;

For our student structure we can declare a pointer variable by writing.

Struct student *ptr_stud, stud;

To assign the address of stud to the pointer using the address operator(&) as we would do in case of any other pointer. So to assign the address, we will write.

ptr_stud=&stud;

to access the members of the structure one way is to write.

(*ptr_stud).roll_no;

Since parenthesis have a higher precedence than *, writing this statement would work well.

C introduces a new operator to do the same task. This operator is known as the pointing to operator (→).

ptr_stud→ roll_no=01;

Union:

- Similar to structures, a union is a collection of variables of different data types. The only difference between a structure and a union is that in case of unions, you can only store information in one field at any one time.
- To better understand a union, think of it as a chunk of memory that is used to store variables of different types. When a new value is assigned to afield, the existing data is replaced with the new data.
- Unions are used to save memory. They are useful for applications that involve multiple members, where values need not be assigned to all the members at any one time.

Declaring a Union:

the syntax for declaring a union is

```
union union name
{
    Data type var-name;
    Data type var-name;
    …..
```

```
};
```

Accessing a member of a union:

A member of a union can be accessed using the same syntax as that of a structure.To access the field of a union,use the dot operator(.).

Syntax:

Union variable name .member name;

Initializing unions:

```
#include<stdio.h>
typedef struct POINT1
{
    int x;
    int y;
}
typedef union POINT2
{
    int x;
    int y;
};
main()
{
     POINT p1={2,3};
    //point p2={4,5};illegal with union
      POINT2=p2;
      P2.x=4;
      P2.y=5;
```

```
Printf("\n the coordinates of p1 are %d and %d",p1.y);

Printf("\n thye coordinated of p2 are %d and %d",p2.y)'

return o;

}
```

Output:

The coordinates of p1 are 2 and 3

The coordinates of p2 are 5 and 5

- In this code POINT1 is a structure and POINt2 is a union. However both the declarations are almost same in main (), you see a point of difference while initializing values. The fields of a union cannot be initialized all at once.

Union inside structures:

```
#include<stdio.h>

Struct student

{

        Union

   {

         Char name[20];

         int roll_no;

    };

main()

{

     Struct  student stud;

    Char choice;

    printf("\n you can enter the name or roll number of the student");

   printf("\n do you want to enter the name?(Y or N):");

   gets(choice);
```

```
if(choice==’y’ || choice==’Y’)
{
   printf(“\n eneter the name:”);
   gets(stud.name);
}
else
{
     Printf(“\n enter the roll number:”);
     Scanf(“%d”,&stud.rollno);
}
    printf(“\n enter the marks☺;
  Scanf(“%d”,&stud.marks);
  if(choice==’Y’ || choice==’y’)
  printf(“\n Name:%s”,stud.name);
  else
   printf(“\n Roll number:%d”,stud.rollno);
   return 0;
}
```

Enumerated data types:

- The enumerated data type is a user defined type based on the standard integer type.
- An enumeration consist of a set of named integer constants.
- In an enumerated type each integer value is assigned an identifier.
- To define enumerated data type we use the keyword enum,which is the abbreviation for enumerate.
- Enumerations create new data types to contain values that are not limited to the values that the fundamental data types may take.

Syntax:

```
enum enumeration name{identifier1,identifier2,…..identifier 3};
```

The enum keyword is basically used to declare and initialize a sequence of integer constants.

Example: which creates a new type of variable called COLOURS to store colour constants

enum COLOURS {RED, BLUE, BLACK, GREEN, YELLOW, PURPPLE, WHITE};

COLOUR is the name given to the set of constants. In case you do not assign any value to a constant,the default value for the first one in the list –RED has the value of 0.the rest of the undefined constants have a value 1 more than its previous one.

RED=0, BLUE=1,BLACK=2,gREEN=3,YELLOW=4,PURPLE=5,WHITE=6

- If you want to explicitly assign values to these integer constants then you should specifically mention those values as:

enum COLOUR{RED=2,BLUE,BLACK=5,GREEN=7,YELLOW,PURPLE,WHITE=15};

- as a result of this statement
  RED=2,BLUE=3,BLACK=5,GREEN=7,YELLOW=8,PURPLE=9,WHITE=15.

Example:

```
#include<stdio.h>

main()

{

    enum{RED=2,BLUE,BLACK=5,GREEN=7,YELLOW,PURPLE,WHITE=15};

    Printf("\n RED=%d",RED);

    Printf("\n BLUE=%d",BLUE);

    Printf("\n BLACK=%d",BLACK);

     return0;

}
```

Output:

RED=2

BLUE=3

BLACK=5

enum variable:

syntax for declaring a variable of an enumerated data type can be given as

1 .enumeration name variable name;

Example:

enum COLOURS bg_colour;

2. enumeration name {identifier1,identifier2,identidier3…}variable name;

Example:

enum COLOURS{RED,BLUE,BLACK,GREEN,YELLOW,PURPLE,WHITE}bg_colour,fore_colour;

Using type def keyword:

typedef enum COLOUR colour;

Then declare variable by writing

Colour forecolour=RED;

Assigning values to enumerated variable:

Once the enumerated variable has been declared values can be stored in it.however an enumerated variable can hold only declared values for the type.

Example:

To assign the colour black to the background colour as

bg_colour=BLACK;

- Once an enumerated variable has been assigned a value,we can store its value in another variable of the same type.

enum COLOUR bg_colour,border_colour;

bg_colour=BLACK;

border_colour=bg_colour;

Differentiate between Structures and Unions.

Structures	Unions
struct keyword is used to define a structure.	union keyword is used to define a union.
Every member within structure is assigned a unique memory location.	In union, a memory location is shared by all the data members.
It enables you to initialize several members at once.	It enables you to initialize only the first member of union.
The total size of the structure is the sum of the size of every data member.	The total size of the union is the size of the largest data member.
We can retrieve any member at a time.	We can access one member at a time in the union.
It supports flexible array.	It does not support a flexible array.

Write a c program that accepts a structure variable as parameters to a function from a function call.

```
#include<stdio.h>
#include<string.h
>struct student
{
        int rollno;
char name[50]; float percentage;

};
void func(struct student
s1);int main( )
{
        struct student
        s1;s1.id = 1;
        strcpy(s1.name,
        “Suvika”);
        s1.percentage = 85.5;
        func(s1);
        return 0;
}
void func(struct student s1)
{
        printf(“ID = %d”, s1.id);
        printf(“Name = %s”, s1.name);
        printf(“Percentage = %f”, s1.percentage);
```

}

Write a program to read details of 10 students and to print the marks of the student if his name is givenas input.

```
#include<stdio.h>
#include<string.h
>struct student
{
        int rollno;
        float marks,
        char name[100], grade[10];
 };
void main()
 {
struct student s[20];int i;
        char checkname[100];
        for(i=0;i<10;i++)
        {
                printf("Enter the detail of %d
                students",i+1);printf("Enter rollno=");
                scanf("%d",&s[i].rollno);
                printf("Enter marks=");
                scanf("%f",&s[i].marks);
                printf("Enter Name=");
                scanf("%s",s[i].name);
                printf("Enter Grade=");
                scanf("%s",s[i].grade);
        }
        printf("Enter the student name to check the
        marks");scanf("%s", checkname);
        for(i=0;i<10;i++)
        {
```

```
        if((strcmp(checkname, s[i].name)) == 0)
            printf("The marks of the student is %f", s[i].marks);
    }
}
```

Files

A file is a collection of data stored on a secondary storage device like hard disk.

The three standard streams in c language are as follows:

- Standard input(stdin)
- Standard output(stdout)
- Standard error(stderr)

Standard input (stdin):

Standard input is the stream from which the program receives its data. The program requests transfer of data using the read operation.

Standard output (stdout):

Standard output is the stream where a program writes its output data. The program requests data transfer using the write operation.

Standard error (stderr):

Standard error is basically an output stream used by programs to report error messages or diagnostics.It is a stream independent of standard output and can be redirected separately..

Types of files:

Two types:

1. **ASCII text files:**
   - A text file is a stream of characters that can be sequentially processed by a computer in forward direction.
   - A text file is usually opened for only one kind off operation at any given time. Because text only process characters, they can only read or write data character at a time.
2. **Binary files:**
   - A binary file may contain any type of data, encoded in binary form for computer storage and processing purpose.

- Like a text file, a binary file is a collection of bytes. In c a byte and a character is equivalent.
- A binary file is also referred to as a character stream with the following two essential differences:
    - A binary file does not require any special processing of the data and each byte of data is transferred to or from the disk unprocessed.
    - C places no constructs on the file, and it may be read from or written to in any manner the programmer wants.

Using files in C:

To use files in C, we must follow the steps given below:

- Declare a file pointer variable.
- Open the file
- Process the file.
- Close the file

Declaring a file pointer variable:

There can be number of files on the disk .In order to access a particular file, you must specify the name of the file that has to be used. This is accomplished by using a file pointer variable that points to a structure FILE.

Syntax:

FILE*file_pointer_name;

Example:

FILE*fp;

Opening a file:

A file must first be opened before data can be read from or written to it. In order to open a file and associated it with a stream, the fopen () function is used.

Syntax:

FILE *fopen(const *file_name,const char *mode);

Table 9.1 File modes

Mode	Description
r	Open a text file for reading. If the stream (file) does not exist, then an error will be reported.
w	Open a text file for writing. If the stream does not exist, then it is created. If the file already exists, then its contents would be deleted.
a	Append to a text file. If the file does not exist, it is created.
rb	Open a binary file for reading. b indicates binary. By default this will be a sequential file in Media 4 format.
wb	Open a binary file for writing.
ab	Append to a binary file.
r+	Open a text file for both reading and writing. The stream will be positioned at the beginning of the file. When you specify 'r+', you indicate that you want to read the file before you write to it. Thus, the file must already exist.
w+	Open a text file for both reading and writing. The stream will be created if it does not exist, and will be truncated if it exists.
a+	Open a text file for both reading and writing. The stream will be positioned at the end of the file content.
r+b/rb+	Open a binary file for read/write.
w+b/wb+	Create a binary file for read/write.
a+b/ab+	Append a binary file for read/write.

Closing a file:

- To close an open file,the fclose()function is used which disconnects a file pointer from a file .After fclose() has disconnected the file pointer from a file ,the pointer can be used to access a different file or the same file but in a different mode.
- The fclose() function not only closes the file ,but aclso flushes all the buffers that are maintained for that file.

  Syntax:

Fclose(FILE*fp);

- In addition to fclose(),there is an fcloseall() function which closes all the streams that are currently open except the standard streams.
  Syntax:

  int fcloseall(void);

Reading data from files:

C provides the following set of functions to read data from a file.

- fscanf()
- fgets()
- fgetc()
- fread()

fscanf():

- The fscanf () function is used to read formatted data from the stream. Its syntax can be given as

Syntax:

fscanf(FILE *stream ,const char *format,…);

- The fscanf () function is used to read data from the stream and store them according to the parameter format into the locations pointed by the additional argument.

fgets():

- The function fgets() stands for file string. This function is used to get a string from a stream.

Syntax:

Char*fgets(char*str,int size,FILE*stream);

- The fgets() function reads atmost one less than the number of characters specified by size from the given stream and stores them in the string str.

fgetc():

- The fgetc() function returns the next character from stream and EOF if the end of file is reached or if there is an error.

Syntax:

fgets(FILE *stream);

- fgetc() returns the character read as an int or return EOF to indicate an error or end of file.
- fgets() reads a single character from the current position of a file .

fread():

- fread() function is used to read data from a file .

syntax:

fread(void*str,size_tsize,size_t num,FILE*stream);

- fread() fumction reads num number of objects and places them into the array pointed to by str.

Writing data to files:

C provides the following set of functions to read data from a file:

fprintf()

fputc()

fputs()

fwrite()

fprintf():

fprintf() is used to write formatted output to stream.

Syntax:

int fprintf(FILE*stream,const char*format,..);

The function writes data that is formatted as specified by the format argument to the specified stream.

fputc():

- The fputc() is just the opposite of fgets()and is used to write a character to the stream.

Syntax:

int fputc(int c,FILE *stream);

- The fputs() function will write the byte specified by c to the output stream pointed by stream.on successful completion ,fputc()will return the value it has written .otherwise in case of error he function will return EOF and the error indicator for the stream will be set.

fputs():

- The opposite of fgets()is fputs().The fputs() function is used to write a line to a file .

Syntax:

int fputs(const char *str,FILE*stream);

- The fputs()function writes the string pointed to by str to the stream pointed to by stream.On successful completion fputs() return 0.In case of any error fputs()return EOF.

fwrite():

- The fwrite() function is used to write data to a file.

Syntax:

int fwrite(const void*str,size_t size,size_t count,FILE*stream);

- The fwrite() function will write objects of size specified by size from the array pointed to by ptr to the stream pointed to by stream.

Detecting the END-OF_FILE:

When reading or writing data from files,we often do not know exactly how long the file is.for example,while reading the file,we usually start reading from the beginning and proceed towards the end of the file.

In C there are two ways to detect EOF:

- While reading the file in text mode, character by character the programmer can compare the character that has been read with the EOF which is a symbolic constant defined in stdio.h with a value -1 .

```
while(1)
{
   C=fgetc(fp);
```

```
if(c==EOF)

break;

printf("%c",c);

}
```

- The other way is to use the standard library function feof ()which is defined in stdio.h. It is used to distinguish between two cases:
  - When a stream operation has reached the end of a file .
  - When the EOF error code has returned an error indicator even when the end of the file has not been reached.

Syntax:

```
int feof(FILE*fp);
```

QUESTIONS

1. Discuss the general syntax of structure variable declaration of structure to store book information.
2. Differentiate between structure and union.
3. Write a program to write employees details in a file called employee.txt. Then read the record of the nth employee and calculate his salary.
4. Discuss the different modes of operation on files with suitable example.
5. Differentiate between gets() and fgets().
6. Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students
7. What are enumeration variable? Explain the declaration and access of enumerated data types with a code in C.
8. Write a short note on functions used to
   Read data from a file
   Write to the file
9. How to detect END-OF-FILE
10. Explain I/O stream and Data I/P stream used in files.
11. Write a program to Implement structures to read and write Book-Title, Book-Author, and Book-id of n books.