STRUCTURES

Definition

- A Structure is a collection of variables of different data types under a single name and provides a convenient way of grouping several of related information together.
- Unlike arrays, it can be used for the storage of heterogeneous data (data of different data types).

Three main aspects of working with structure

1. Defining a structure type (Creating a new type)
2. Initializing structure elements
3. Declaring variables and constants (objects) of the newly created type.

Defining Structure

Syntax

struct structure_name

{

element-1;

element-2;

element-3; //Variable declarations

...

...

element-n;

} v1,v2..... vn;

Where element1, element2, element3 are variables of any primitive or derived data types and v1,v2,. .vn are structure variable.

Example

struct book

{

char author[40];

float price;

int page;

}b1,b2;

Rules for defining structure

- Structure definition consists of the keyword struct followed by a structure tag name and a structure declaration list enclosed within braces.
- The structure declaration list consists of one or more variables declaration, possibly of different data types. The variable names declared in the structure declaration list are known as structure members.
- Structure members can be variables of the basic types(eg: char, float, int) or pointer type(eg: char *, int *) or aggregate type(eg: array).
- A structure declaration list cannot contain a member of void type or incomplete type or function type.
- Self referential structure: a structure may contain a pointer to an instance of itself is known as self referential structure.

Initializing Structure Elements

Syntax

Struct book

{

int page;

char author[10];

float price;

}b1;

Example

void main()

{

b1.author=”Kalam”;

printf(“Enter price:”);

scanf(“%f”,&b1.price);

b1.page=178;

}

Declaring Structure Objects

- Variables (or) constants of the created structure type can be created either at the time of structure definition (or) after the time of structure definition.

Syntax

[type qualifier] structure type identifier name [= initialization list]; (or) variables;

Example

struct book b1={3,2,1}; // it contains the initialization list

struct book b1,b2,b3; // it contains the variable

Rules for declaring structure objects:

- It is important to note that the structure members cannot be initialized during the structure definition; however the members of a structure object can be initialized by providing an initialization list.

Operations on Structures

- Aggregate operations
- Segregate operations

Aggregate Operations

- An aggregate operation treats an operand as an entity and operates on the entire operand as whole instead of operating on its constituent members.

Types

a) Accessing members of an object of a structure
b) Assigning a structure object to a structure variables
c) Address of a structure object
d) Size of a structure.

a) Accessing members of an object of a structure:

The members of a structure object can be accessed by using:

(i) Direct member access operator (dot operator)
(ii) Indirect member access operator(arrow operator)

(i) Direct member access operator (dot operator):

Syntax:

struct variable-name.struct-element-name

Example Program 2.1

//C program to print student details using structure

```
#include<stdio.h>
#include<conio.h>
struct student
{
        char name;
        int rno;
        float mark;
};
struct student s;
void main()
{


        printf("enter the name");
        scanf("%c",&s.name);
        printf("enter the rno");
        scanf("%d",&s.rno);
        printf("enter the mark");
        scanf("%f",&s.mark);
        printf("NAME=%c",s.name);
        printf("RNO=%d",s.rno);
        printf("MARK=%f",s.mark);
        getch();

        }
```

OUTPUT

enter the name

xyz

enter the rno

20

enter the mark

80

NAME=xyz

RNO=20

MARK=80

Example Program 2.2

/*C program to calculate the student's average marks and student details using structure*/

```
#include<stdio.h>
#include<conio.h>
struct student
{
        char name;
        int rno;
        int m1,m2,m3;
};
struct student s;
void main()
{
        float total,average;
        printf("enter the name");
        scanf("%c",&s.name);
```

```
        printf("enter the rno");
        scanf("%d",&s.rno);
        printf("enter the marks");
        scanf("%d %d %d",&s.m1,&s.m2,&s.m3);
        total=s.m1+s.m2+s.m3;
        average=total/3;
        printf("NAME=%c",s.name);
        printf("RNO=%d",s.rno);
        printf("AVERAGE MARK=%f",average);
        getch();
}
```

Output:

```
enter the name
xyz
enter the rno
20
enter the marks
80
90
95
NAME=xyz
RNO=20
AVERAGE MARK=88.33
```

Example Program 2.3

```
#include<stdio.h>
struct book //Struct datatype declaration
{
        int x,y;
```

```
};
void main()
{
    struct book s1={4,5}; //s1-> variable of structure and values are
    initialized
    int a=10 , b=20 ;
    printf("\na=%d",s1.x+a); // elements are accessed using dot operator(.)
    printf("\nb=%d", s1.y+b);
}
```

Output:

a=4

b=5

(ii) Indirect member access operator (arrow operator)

Syntax :

struct variable name -> struct element name

(or)

**struct variable name . struct element name*

Example Program 2.4

```
#include<stdio.h>
struct book // structure data type declaration
{
        int x,y;
};
struct book *b1; //pointer to structure
void main()
{
        printf("enter the values");
        scanf("%d", &b1->x);
        scanf("%d",&b1->y); //-> operator used
        printf("\nx=%d", b1->x);
        printf("\ny=%d", *b1.y);
}
```

Output:

Enter the values 10 20

x=10

y=20

b) Assigning a structure object to a structure variables

- Assignment operator (=) is used to assign the values of one variable to another variable. When assignment operator (=) is applied on structure variables, it performs member by member copy.

Example Program 2.5

```
#include<stdio.h>
struct book // struct datatype is declared
{
        char title[25], author[20];
        int price;
};
void main()
{
        struct book b1,b2,b3; //structure variables are declared
        b1={" cutting stone", "Abraham",400};
        b2.author=b1.author;
        b3=b1; // b1 variable values are assigned to b3
        printf("%s by %s is of Rs. %d \n", b1.title,b1.author,b1.price);
        printf("%s is the author of second book",b2.author);
        printf("%s by %s is of Rs. %d \n", b3.title,b3.author,b3.price);
}
```

Output:

cutting stone by Abraham is of Rs.400

Abraham is the author of second book

cutting stone by Abraham is of Rs. 400

c) Address of a structure object

- The address of operator (&) when applied to a structure object gives its base address. It can also be used to find the address of the constituting members of a structure object.

Example Program 2.6

```
#include<stdio.h>
struct book // struct datatype is declared
{
```

```
        char title[25], author[20];
        int price;
};
void main()
{
        struct book b1,b2,b3; //structure variables are declared
        b1={" cutting stone", "Abraham",400};
        b2.author=b1.author;
        b3=b1; // b1 variable values are assigned to b3
        printf("%s by %s is of Rs. %d \n", b1.title,b1.author,b1.price);
        printf("\n address of structure's element title %u ",&b1.title);
}
```

Output

cutting stone by Abraham is of Rs.400

address of structure's element title 1700printf("\n address of whole variable %u",&b1);

address of structure's element author 1725

address of whole variable 4000

d) Size of a structure.

- When the sizeof operator is applied to an operand of a structure type it will produce the result as how much memory space is occupied by that particular object.

Syntax:

sizeof (expression);

sizeof type

Example:

sizeof (struct book); // use structure's name

sizeof b1 // use variable

Program 2.7

```
#include<stdio.h>
struct book //structure type declaration
{
        char a; // elements are declared
        int b;
        char c;
        float d;
}; //structure type declarations are terminated
void main()
{
        struct book var; //variable declaratio
        printf("obj of struct book will take %d bytes\n",sizeof(struct book));
        printf("structure variable var takes %d bytes\n", sizeof var);
}
```

Output:

obj of struct pad will take 8 bytes

structure variable var takes 8 bytes

Segregate Operations

- A segregate operation operates on the individual members of a structure object.

Program 2.8

```
#include<stdio.h>
struct book
{
        char title[25], author[20];
        int page; float price;
};
void main()
```

```
{
    struct book b1;
    printf("enter title, author, page, price");
    scanf("%s, %s, %d, %f",& b1.title,&b1.author,&b1.page, &b1.price);
    printf("title is %s, author is %s, page no %d, price %d",b1.title,
    b1.author, b1.page, b1.price);
    // operations on individual element
    b1.page+=100;
    b1.price+=10;
    printf("title is %s, author is %s, page no %d",b1.title, b1.author,
    b1.page);
    printf("price %d",b1.price);
}
```

Output

Enter title, author, page, price

Principles of life, prabhu, 145, 200.00

Title is principles of life, author is prabhu, page no 145, price 200.00

Title is principles of life, author is prabhu, page no 245, price 210.00