Structure

Collection of Data Members of Different type

Collection of logically related different data types (Primitive and Derived) referenced under one name.

Important Points(topics)

1. Declare Structure
2. Declare Data Members
3. Declare Structure object and its size
4. Object as local
5. Object as global
6. Call data members
7. Assign value to data members
8. **Assign value to DM inside structure not allow**
9. Input Data members
10. Static value(Default value) and dynamic value to data members
11. Global Structure
12. Locale Structure
13. Call Object as local
14. Call Object as global
15. **Data Members as Array**
16. **Object Members as Array**
17. **DM/ Object Members as Array**
18. Object as pointer
19. Structure Object as Function Argument
20. Nested Structure
21. Type Def

C/C++ arrays allow you to define variables that combine several data items of the same kind but structure is another user defined data type which allows you to combine data items of differentkinds.
Structures are used to represent a record, suppose you want to keep track of your books in alibrary. You might want to track the following attributes about each book:

Title
Author
Subject
Book ID

Defining a Structure:
To define a structure, you must use the struct statement. The struct statement defines a new datatype, with more than one member, for your program. The format of the struct statement is this:

```
struct [structure tag]
{
member definition;
member definition;
...
member definition;
} [one or m ore structure variables];
```

The **structure tag** is optional and each member definition is a normal variable definition, such asint i; or float f; or any other valid variable definition. At the end of the structure's definition, beforethe final semicolon, you can specify one or more structure variables but it is optional. Here is theway you would declare the Book structure:

```
struct Books
{
char title[50];
char author[50];
char subject[100];
int book_id;
}book;
```

```
void main()
{
cout<< "Size of Structure of Object in bytes :"<<sizeof(book);
}
```

Output: 202

A structure is a collection of variable which can be same or different types. You can refer to a structure as a single variable, and to its parts as **members** of that variable by using the dot (.) operator. The power of structures lies in the fact that once defined, the structure name becomes a **user-defined data type** and may be used the same way as other built-in data types, such as int, double, char.

```
struct STUDENT
{
    int rollno, age;
    char name[80];
    float marks;
} ;
int main()
{
   // declare two variables of the new type
    STUDENT s1, s3;
    //accessing of data members
    cin>>s1.rollno>>s1.age>>s1.name>>s1.marks;
    cout<<s1.rollno<<s1.age<<s1.name<<s1.marks;
    //initialization of structure variable
    STUDENT s2 = {100,17,"Aniket",92};
    cout<<s2.rollno<<s2.age<<s2.name<<s2.marks;
    //structure variable in assignment statement
    s3=s2;
    cout<<s3.rollno<<s3.age<<s3.name<<s3.marks;
    return 0; }
```

Defining a structure
When dealing with the students in a school, many variables of different types are needed. It may be necessary to keep track of name, age, Rollno, and marks point for example.

```
struct STUDENT
{
    int rollno, age;
    char name[80];
    float marks;
};
```

STUDENT is called the **structure tag,** and is your brand new data type, like int, double or char.
rollno, name, age, and **marks** are **structure members.**

Declaring Variables of Type struct

The most efficient method of dealing with structure variables is to define the structure **globally**. This tells "the whole world", namely main and any functions in the program, that a new data type exists. To declare a structure globally, place it **BEFORE** void main(). The structure variables can then be defined locally in main, for example…

```
struct STUDENT
{
    int rollno, age;
    char name[80];
    float marks;
};
int main()
{
    // declare two variables of the
new type
    STUDENT s1, s3;
    ……
    ……
    return 0;
}
```

Alternate method of declaring variables of type struct:

```
struct STUDENT
{
    int rollno, age;
    char name[80];
    float marks;
} s1, s3;
```

Accessing of data members

The accessing of data members is done by using the following format: structure variable. member name for example
cin>>s1.rollno>>s1.age>>s1.name>>
s1.marks;

Initialization of structure variable Initialization is done at the time of declaration of a variable. For example
STUDENT s2 = {100,17,"Aniket",92};

Structure variable in assignment statement s3=s2;

The statement assigns the value of each member of s2 to the corresponding member of s3. Note that one structure variable can be assigned to another only when they are of the same structure type, otherwise complier will give an error.

Nested structure (Structure within structure)

It is possible to use a structure to define another structure. This is called nesting of structure. Consider the following program

```
struct DAY
{
  int month, date, year;};
struct STUDENT
{    int rollno, age;
    char name[80];
    DAY date_of_birth;
    float marks;};
```

Typedef It is used to define new data type for an existing data type. It provides and alternative name for standard data type. It is used for self documenting the code by allowing descriptive name for the standard data type.
The general format is:**typedef existing datatype new datatype**for example: **typedef float real;**
Now, in a program one can use datatype real instead of float.Therefore, the following statement is valid:**real amount;**

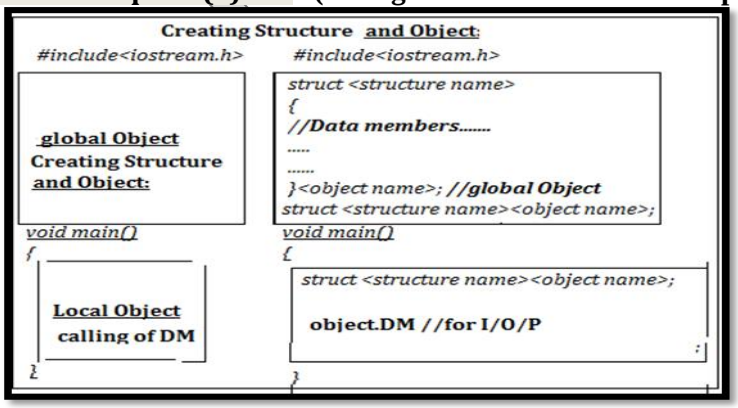
Enumerated data type The enum specifier defines the set of names which are stored internally as integer constant. The first name was given the integer value 0, the second value 1 and so on.

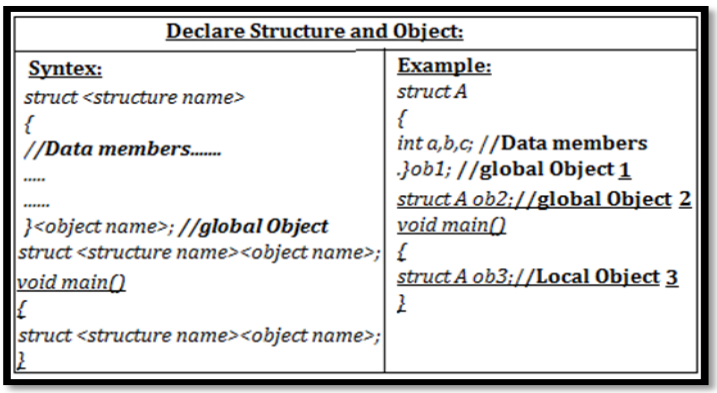
for example: enum months{jan, feb, mar, apr, may} ;
It has the following features:It is user defined., It works if you know in advance a finite list of values that a data type can take. The list cannot be input by the user or output on the screen.

#define preprocessor directive

The #define preprocessor allows to define symbolic names and constants e.g.#define pi 3.14159
This statement will translate every occurrence of PI in the program to 3.14159

Macros Macros are built on the #define preprocessor. Normally a macro would look lik7jme:
#define square(x) x*x (Its arguments substituted for replacement text, when the macro is expanded.)





Calling of Data Members by object

```
#include<iostream.h>
struct A
{
int a,b,c; //Data members
}ob1; //global Object 1
struct A ob2;//global Object 2
void main()
{
struct A ob3;//Local Object 3
 o1.a=5;//cin>>o1.a;(input)
 o1.b=6;//cin>>o1.b;(input)
 o1.c=o1.a+o1.b;//process
 cout<<o1.c; //display/output
 }
 output:
 11
```

Assign value to DM with object.

```
struct A
{
int a,b,c; //Data members
}ob1={3,5,8};
```

Object as Pointer

```
#include<iostream.h>
struct A
{
int a,b,c;
}* ob1;
```

o1 -> a=5;
o1 -> b=6;

Global Structure /Locale Structure

```
#include<iostream.h>
struct A
{
int a,b,c;
}ob1;
void main()
{
 struct B
 {
 int a,b,c;
 }obj2;
}
```

Global Structure

Locale Structure

Data Members as Array

```
#include<iostream.h>
struct A
{
int m[5];
}x;
void main()
{
int S=0;
for (int i=0;i<5;i++)
   {
        cin>>x.m[i];
        S=S+x.m[i];
   }
cout<<S;
}
```

Object as Array

```
#include<iostream.h>
struct A
{
int m;
}x[5];
void main()
{
int S=0;
for (int i=0;i<5;i++)
   {
        cin>>x[i].m;
        S=S+x[i].m;
   }
cout<<S;
}
```

DM and Object as Array

```
#include<iostream.h>
struct A
{
int m[5];
}x[5];
void main()
{
int S=0;
for (int i=0;i<5;i++)
   {
        for(j=0;j<5;j++)
        {
        cin>>x[i].m[j];
        S=S+x[i].m[j];
        }
   }
cout<<S;
}
```

Structure Object as Function Argument

```
#include<iostream.h>
struct A
int a,b;
}x;
void Add(A y);
void main()
{
x.a=6;
x.b=7;
Add(x);
}
void Add(A y)
{
int c=y.a+y.b;
cout<<c;
}
```

Structure as return type

```
#include<iostream.h>
struct A
int a,b;
}x,z;
A Add(A y);
void main()
{
x.a=6;
x.b=7;
z=Add(x);
cout<<z.a<<z.b;
}
A Add(A y)
{
y.a+=5;
y.b+=3;
return y;
}
```

Nested Structure

```
struct A
{ int a;
    struct B
    {
    int b;
    }obj2;
}ob1;
 void main()
 {
 obj1.a=5;//right
 obj2.b=5;//wrong
 obj1.obj2.b=5;//right
 }
```

Set 1.

1. WAP to collect information of a student(Roll No ,Name,class1,age, total_marks, fee).
2. WAP to collect information of a Employ (Emp No ,Name, city, age, Dept, sal).
3. WAP to collect information of 10 students (Roll No ,Name, class, age, total_marks, fee).
4. WAP to collect information of a students with 5 subjects. (Roll No ,Name, class, age, 5 subjects total_marks, fee,per).
5. WAP to collect information of 10 students with 5 subjects. (Roll No ,Name, class, age, 5 subjects total_marks, fee,per).
6. WAP to collect information of 100 students with 5 subjects. (Roll No ,Name, class, age, 5 subjects total_marks, fee,per). And s[illegible]h a record by: Roll no., Name,Class

Set 1 Find Output

Q1.
```
#include<iostream.h>
#include<conio.h>
struct three_d
{
int x,y,z; };
void movein(three_d &t, int step=1)
{
 t.x+=step;
t.y+=step;
t.z+=step;
}
void moveout(three_d &t, int step=1)
{
t.x-=step;
t.y+=step;
t.z-=step;
}
void main()
{
three_d t1={10,20,5},t2={30,10,40};
movein(t1);
moveout(t2,5);bcout<<t1.x<<","<<t1.y<<","<<t1.z<<endl;
cout<<t2.x<<","<<t2.y<<","<<t2.z<<endl;
movein(t2,10);bcout<<t2.x<<","<<t2.y<<","<<t2.z<<endl;
}
```

Q2.
```
#include<iostream.h>
#include<conio.h>
struct Box
{
int Len, Bre, Hei;};
void Dimension(Box B)
{
cout << B.Len << " X " << B.Bre << " X " << B.Hei << endl;
}
void main ( )
{
Box B1 = {10, 20, 8}, B2, B3;
++B1.Hei;
Dimension (B1); //first calling
B3= B1;
++B3.Len;
B3.Bre++;
Dimension (B3); // second function calling
B2= B3;
B2.Hei += 5;
B2.Len -= 2;
Dimension (B2); // third function calling
}
```

Q3.
```
#include<iostream.h>
#include<conio.h>
struct PLAY
{
 int Score, Bonus;};
void Calculate(PLAY &P, int N=10)
{
P.Score++;
P.Bonus+=N;
}
void main()
{
PLAY PL={10,15};
Calculate(PL,5);
cout<<PL.Score<<":"<<PL.Bonus<<endl;
Calculate(PL);
cout<<PL.Score<<":"<<PL.Bonus<<endl;
Calculate(PL,15);
cout<<PL.Score<<":"<<PL.Bonus<<endl;
}
```

Q4.
```
#include<iostream.h>
#include<conio.h>
#include<string.h>
struct student
{
int rno;
char name[20];
};
void main()
{
student a[2]={{1,"Amit"},{2,"Sumit"}};
for(int i=0;i<2;i++)
{
cout<<"\n Rno"<<a[i].rno;
cout<<"\n Name ";
for(int j=0;j<strlen(a[i].name);j++)
cout<<a[i].name[i]<<" ";
}
}
```

Q5.
```
#include<iostream.h>
#include<conio.h>
#include<string.h>
struct Game
{
char Magic[20];
int Score;
};
void main()
{
Game M={"Tiger",500};
char *Choice;
Choice=M.Magic;
Choice[4]='P';
Choice[2]='L';
M.Score+=50;
cout<<M.Magic<<":"<<M.Score<<endl;
Game N=M;
N.Magic[0]='A';N.Magic[3]='J';
N.Score-=120;
cout<<N.Magic<<":"<<N.Score<<endl;
}
```

Q6.
```
#include<iostream.h>
#include<conio.h>
#include<string.h>
struct GAME
{
int Score, Bonus;};
void Play(GAME &g, int N=10)
{
g.Score++;g.Bonus+=N;}
void main()
{
GAME G={110,50};
Play(G,10);cout<<G.Score<<":"<<G.Bonus<<endl;
```

```
Play(G);cout<<G.Score<<":"<<G.Bonus<<endl;
Play(G,15);cout<<G.Score<<":"<<G.Bonus<<endl;
}
```

Q7.
```
#include<iostream.h>
#include<conio.h>
#include<string.h>
struct KEY
{
char word[10];
int count;};
 void changekeyword(KEY somekey);
 void main()
{
KEY aKEY;
strcpy(aKEY.word, "#define");
aKEY.count=0;
changekeyword(aKEY);
cout<<aKEY.word<< "\t"<<aKEY.count<< "\n";
getch();
}
void changekeyword(KEY somekey)
{
strcpy(somekey.word, "const");
somekey.count=1;
cout<<somekey.word<< "\t" <<somekey.count<< "\n";
}
```

Q8.
```
#include<iostream.h>
#include<conio.h>
#include<string.h>
struct area
{
int leangth;
int breadth;
int areas;};
void calarea(area &P1,int y=10)
{
P1.areas=P1.leangth*P1.breadth; P1.areas/=y;
P1.leangth++;
P1.breadth++;}
void main( )
{area first={20,50,0},second={10,30,0};
calarea(first);
cout<<first.areas<<"#"<<first.leangth<<"#";
cout<<first.breadth;
cout<<endl;
calarea(second,5);
cout<<second.areas<<"#"<< second.leangth<<"#"<<
second.breadth;}
```

Q9. Find the output of the following program:
```
# include <iostream.h>
# include <conio.h>
struct triangle
 {  float angle;
  float base;
  float height; };
void display (triangle question)
 {
   cout << endl << " Base   : " << question.base;
   cout << endl << " Height : " << question.height;
   cout << endl << " Angle  : " << question.angle;
 }
void triangle difference (triangle one, triangle &two)
 {
  triangle three;
  if (one.base < two.base )
    {
 two.height = 100;
 three = two;
}
  else
{
    two.angle = 22;
     three = one;
  }
  return three;
}
void main ()
{
 clrscr();
 triangle building = {20,10,30}, flagstaff, differ;
 flagstaff = building;
 flagstaff.height =+5;
 flagstaff.base +=45;
 display (building);
 differ = difference (building, flagstaff);
 display (differ);
 display (flagstaff);
 getch();
}
```

Q10
```
struct Pixel
{
int C, R;};
void Display (Pixel P)
{
cout << "Col : " <<P.C << "Row : " <<P.R << endl ;}
void main()
{
Pixel X ={40, 50}, Y, Z;
 Z=X;
X.C+=10;
Y=Z;
 Y.C+=15 ;
 Y.R+=25;
 Z.C -= 30;
 Display (X)
 Display (Y);
 Display (Z);}
```

```
Col : 50 Row : 50
Col : 55 Row : 75
Col : 10 Row : 50
```

Q1

```
11,21,6
25,15,35
35,25,45
```

Q2.

```
10 X 20 X 9
11 X 21 X 9
9 X 21 X 14
```

Q3.

```
11:20
12:30
13:45
```

Q4.

```
Rno1
Name A A A A
Rno2
Name u u u u u
```

Q5.

```
TiLeP:550
AiLJP:430
```

Q6

```
111:60
112:70
113:85
```

Q7.

```
const  1
#define 0
```

Q8

```
100#21#51
60#11#31
```

Q9

```
Base   : 10
Height : 30
Angle  : 20
Base   : 55
Height : 100
Angle  : 20
Base   : 55
Height : 100
Angle  : 20_
```

Rewrite After Removing Errors

Q1. #include<iostream.h>
void main( )
{ struct STUDENT
{ char stu_name[20];
char stu_sex;
int **stu_age=17;**
} student;
gets(stu_name); ***student.DM***
gets(stu_sex);} ***student.DM***

Q 1. #include <idstream.h>
void main()
{ struct movie
{ char movie_name[20];
char movie_type;
int **ticket_cost = 100**
}MOVIE;
gets(movie_name); ***MOVIE.DM*** **gets(movie_type);}** ***MOVIE.DM***

Q 2. #include <iostream.h>
struct Pixels
{ int Color,Style;}
void ShowPoint(Pixels P)
{ cout<<P.Color,P.Style<<endl;}
void main()
{
Pixels Point1**=[5,3];**//**{5,3}**
ShowPoint(Point1);
Pixels Point2=Point1;
Color.Point1+=2;*//Point1.color*
ShowPoint(Point2);

Q 3. #include <iostream.h>
struct Pixels
{ int **Color=10,Style=12**;
}
void ShowPoint(Pixels P)
{ cout<<P.Color:P.Style<<**end**;
}
void main()
{
Pixels Point**=(5,3);**
ShowPoint(Point1);
Pixels Point2=Point1;
Color.Point1+=2;
ShowPoint(Point2);
}

Q 4. #include <iostream.h>
void main( )
{
struct Book
{
char Book_name[20];
char Publisher_name[20];
int **Price = 170;**
} New Book;
gets(Book_name);
gets(Publisher_name);}

Q 5. #include<iostream.h>
int main()
{
struct student
{int. rno, mark;
}stu;
student **stuA= (1001,49);**
student stuB= stuA;
if (stuA!= stuB)
stuA.mark= 55;
else
stuB.mark= 25;
cout<<stuA.mark<<**stub**.mark;}

Q6. structure swimmingclub
{
int **mem number;**
char mamname[20]
char **memtype[]="LIG";**
};
void main()
{
swimmingclub per1,per2;
cin<<"Member Number";
cin>>**memnumber.per1**;
cout<<"\n Member name";
cin>>per1.membername;
per1.memtype="HIG";
per2=per1;
cout<<"\n Member number "<<per2.memnumber;
cout<<"\n Member name "<<per2.memname;
cout<<"\n Member number "<<per2.memtype;
}