Structures

- To bring logically related elements under one unit, we have structures.
- For example details of student like name, roll number, marks, address etc.

Name : char name[10];

roll : int rno;

Marks: float marks;

address: char add[20];

```
struct   student
 {
        char name[10];
       int rno;
       float marks;
      char add[20];
   } ;
```

The following code shows how to define a structure, the keyword struct tells the compiler that a structure is being defined.

Syntax for declaring a structure is

```
struct <tag-name>
   { type var_name;
      type var_name;
      type var_name;
   } structure_variable;
```

How to declare a variable for structure?

To declare a variable for the structure

structure_name variable name;

eg student st;

or

struct student

{ .....

....

} st;

Number of bytes for the structure variable?

```
struct student
 {     char name[10];
        int rno;
        float marks;
       char add[20];   } st;
```

Number of bytes for variable

st : 10 + 2 + 4 +20 = 36

Write a structure definition stmt to hold employee details like empId, sal, grade, post

```
struct Employee
{  int ID;
    float sal;
   char g,post[20];
 } emp;
```

Referencing structure elements

Once the structure variable has been defined, its members can be accessed through the use of . (dot operator).

The syntax for accessing a structure element is

structure_name.element_name

- For eg: cin>>emp.ID;
- emp.ID=emp.ID*10;
- cout<<emp.ID;
- gets(emp.post);
- cout<<emp.post;

- Write a program to accept the name , roll number and marks of a student using class.

```
struct student
{  char name[10];
   int rn;
   float marks; } ;
void main()
{   student st;
    gets(st.name);
    cin>>st.rn;
    cin>>st.marks ;
```

```
cout<< “The details of the
students are:” ;
cout<<“\nName:”<<st.name;
cout<<“\nName:”<<st.rn;
cout<<“\nMarks:”<<st.marks;


}
```

- Write the pgm to accept the name , roll number and marks of 5 subjects of a student using class.

```
struct student
{  char name[10];
   int rn;
   float marks [5] ;
 } st;
```

```
// for one student
 gets(name);
 cin>>st.rn;
 for(i=0;i<5;i++)
  { cin>>st.marks[i]; }
```

Initializing structure elements

- The structure elements of a structure can be intialized separately or jointly
- Eg : st.rn=10;
- strcpy(st.name,”james”);
- st.marks=78.5

or

student obj={14, “Seema”,67.5};

- Write a pgm to accept the details of an employee and print it. (using structures)

- Write a program to accept the name , roll number and marks of 5 subjects and finding the average of a student using structure (having 4 elements).

- Program to read the day, month and year using a structure and add number of days from the user of the date and display the correct date.

Array of structure variable

```
struct Employee
{  int ID;
    float sal;
} emp[10];

for(int i=0;i<3;i++)
   { cin>>emp[i].ID >>emp[i].sal; }
```

- Program to store information of 5 employees details in a structure and to display the information of an employee depending upon the employee no given.

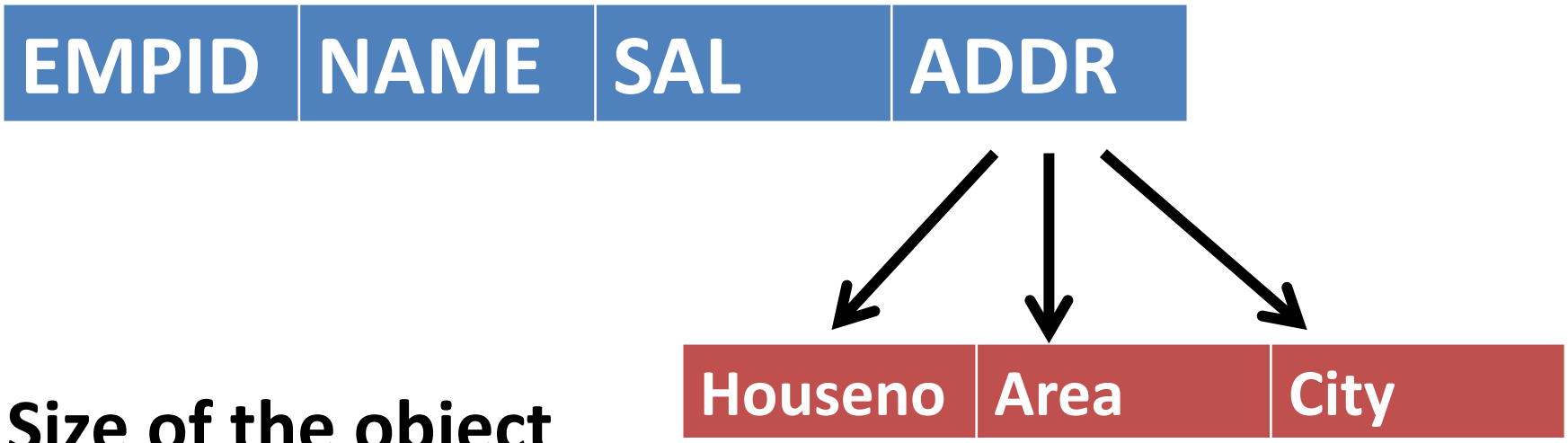
Nested Structure

- A structure inside another structure is called a nested structure.

```
struct Address
{ int houseno;
  char area[20];
  char city[20];
};
```

```
struct Emp
{ int empID;
  char name[20];
  float sal;
   Address addr;
};
```

So an object of Employee will have



Size of the object of structure Employee ?

68

Accessing Nested Structure Members

- It is accessed using dot operator.
- To access the city member

  Employee obj;

  obj.addr.city;

How to initialize nested structures

```
Emp emp1=
 { 101,
   ”JOY”,
    45000,
    {1234,”BG Road”, “Bangalore”}      };

Emp emp2={ 101, ”JOY”, 45000, {1234,”BG Road”, “Bangalore”}
  };
```

- Write the above program and print the details of the person having the highest salary.

SA -2

Structures

Passing structures to functions

- You can pass the structure element separately or the structure as a whole.

eg struct date { int day;

int month;

int year; }bdate;

// Assume you are calling a function which takes only the day and month

- Declaration

```
void date_fun (int, int) ;
```

- Function call

```
date_fun(bdate.day, bdate.month);
```

- Definition

```
void date_fun(int d, int m)
{ ……..  }
```

- You can make the function either call by value or call by reference depending on how you want to program
- If it is call by reference, the changes made in the function will be reflected in the structure value.

Entire structure can be passed as a parameter in the function

- Call by value

eg struct date { int day;
int month;
int year; };

- Declaration

```
void date_fun (date, date) ;
```

- Function call

```
date_fun(d1, d2);
```

- Definition

```
void date_fun(date d1, date d2)
{ ……..  }
```

Entire structure can be passed as a parameter in the function

- Call by reference

eg struct date { int day;
int month;
int year; };

- Declaration

```
void date_fun (date &, date &) ;
```

- Function call

```
date_fun(d1, d2);
```

- Definition

```
void date_fun(date &d1, date &d2)
{ ……..  }
```

Returning structure from a function

- Declaration

```
date date_fun (date &, date &) ;
```

- Function call

```
date d3=date_fun(d1, d2);
```

- Definition

```
date date_fun(date &d1, date &d2)
{…… return d1; }
```

- Declaration

```
void date_fun (date &, date &) ;
```

- Function call

```
date_fun(d1, d2);
```

- Definition

```
void date_fun(date &d1, date &d2)
{ ……..  }
```

typedef

- C++ allows you to define explicitly new data type names by using the keyword typedef.
- Syntax for typedef is
- typedef type name;
- eg
- typedef float amount;

So instead of declaring

float money;

we can declare

amount money;

here money is a variable of type amount

which inturn is a variable of type float.

This stmt : type amount money;

tells the compiler to recognize money as another name for amount, another name for float.

#define processor directive

- Processor commands are called directives and begins with a #(hash/pound symbol)
- No white space should appear before # and a semicolon is not required at the end.
- eg
- #define PI 3.14
- #define MAX 70
- #define NAME “Computer Science”

We can create a macro using #define

```
#define SQUARE(x) x*x
void main()
{ int value=3;
  cout<<SQUARE(value);
}
```

output:

9

Things to remember about Macro

- A macro with no argument is treated as a symbolic constant
- A macro with arguments has its arguments substituted for replacement value, when the macro is expanded.
- A macro substitute text only, it does not check for data type

```
#define PI 3.14
#define CIR_AREA(x)  PI*x*x

void main()
{ int c=2;
int area=CIR_AREA(c+2);
cout<<area;
}
```