Structure

Group of data items of different data types held together in a single unit.a

Declaring a Structure

```
struct  sname
{
    type var1;
    type var2;
    type var3;
        .
        .
    type varN;
};
```

struct is a keyword to define a structure.
sname is the name given to the structure data type.
type is a built-in data type.
var1,var2,var3,.....,varN are elements of structure being defined.

Structure template

sname is called **tag** makes it possible to declare other variable of the same structure type without having to rewrite the template itself. It's a type name.tag is optional.

Eg to store information of an employee, our structure declaration may look like as shown overleaf

```
struct  employee_type
{
    int code;
    char name[20];
    int dept_code;
    float salary;
};
```

- No variable has been associated with this structure
- No memory is set aside for this structure.

Declaring Variables of the Structure Type

Declares a variable employee of type employee_type

```
struct  employee_type  employee;
```

At this point, the memory is set aside for the structure variable employee.

- We can also declare variable(s) of structure type along with the structure declaration.

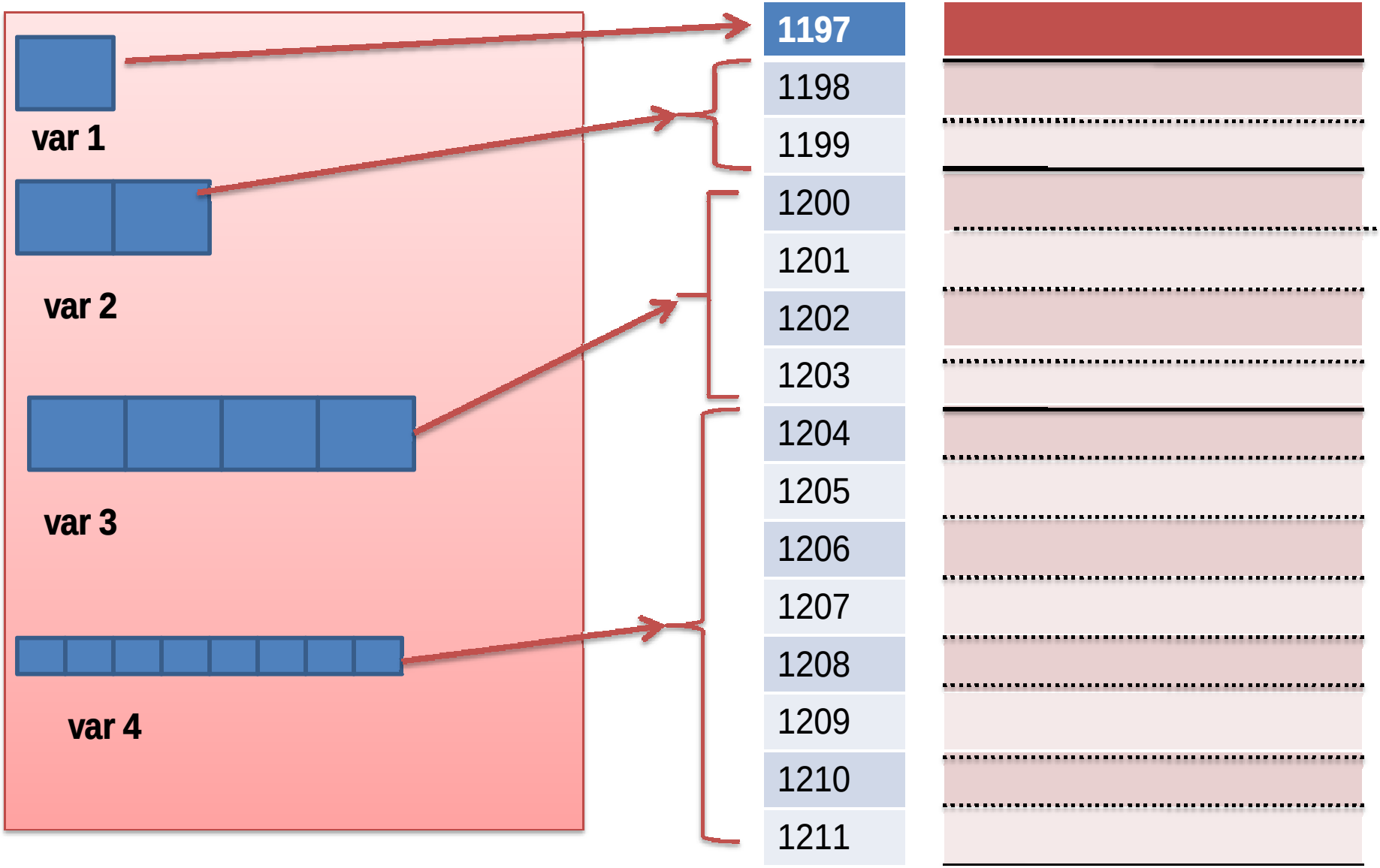
```
struct  employee_type
{
int code;
char name[20];
int dept_code;
float salary;
}employee
```

Consider the declarations to understand how the elements of the structure variables are stored in memory

```
struct  example_type
{
   char var1;
   int var2;
   float var3;
   double var4;
};
struct example_type  sample1;
```

Note: **all members are stored in contiguous memory location in order in which they are declared.**

How the elements of the structure variables are stored in memory



Intializing Structures

```
struct  student_type
{
   int rollno;
   char name[25];
   int age;
   float height;
};
struct  student_type student={1000,”Surbhi salaria”,18,5.6};
```

Accessing Structure Elements

- Elements are accessed using dot operator.
- It provides a powerful and clear way to refer to an individual element.

Syntax: **sname.vname**

sname is structure variable name.

vname is name of the element of the structure.

Eg: the element of the structure variable ***student*** can be accessed as

student.rollno,student.name,student.age,student.height

Entering Data into Structures

```
struct  employee_type
{
      int code;
      char name[25];
      char dept[15];
      float salary;
};
main()
{
      struct employee_type employee;
      printf("\nEnter employee code:\n");
      scanf("%d",&code);
      printf("\nEnter name:\n");
      gets(employee.name);
      printf("\nEnter employee's dept:\n");
      gets(employee.dept);
      printf("\nEnter employee's salary:\n");
      scanf("%f",&employee.salary);
```

continue

continue

```
printf("\n\nParticulars of emp as entered by user\n");

printf("\nEmployees code:%d",employee.code);

printf("\nEmployee's name:%s", employee.name);

printf("\nEmployee's dept:%s",employee.dept);

Printf("\nEmployee's sal:%f",employee.salary);
```

Use of Assignment Statement for Structures

- Value of one structure variable can be assigned to another variable of the same type using simple assignment statement.if student1 and student2 are structure variable of type student_type,then

student2=student1;

Assigns value of structure variable ***student1*** to ***student2***

Simple assignment cannot be used this way for arrays.

Pointers and Structures

struct student_type student,*ptr

It declares a structures variable *student* and a pointer variable *ptr* to structure of type *student_type*.

ptr can be initialized with the following assignment statement

ptr=&student;

HOW WE CAN ACCESS THE ELEMENTS OF STRUCTURE?

**ptr.rollno,*ptr.name,*ptr.age,*ptr.height*

*But this approach **will not work** because dot has higher priority*

Correctly way to write is:

*(*ptr).rollno,(*ptr).name,(*ptr).age,(*ptr).height*

or

ptr->rollno,ptr->name,ptr->age,ptr->height

Structure and Function

- The relationship of structure with the function can be viewed from three angles:-
    1. Passing Structures to a function.
    2. Function Returning Structure.
    3. Passing array of Structures to Function.

Passing Structure to a Function

Similar to passing array of variable,structure can be passed to a function as argument

Syntax:

type-specifier func-name(struct-variable); /*actual argument*/

Read and display student grade by using structure with function

```
struct  student
{
    int rn;
    char name[20];
    char grade;
}s;
main()
{
    printf("\nEnter rollno,name and grade of student:\n");
    scanf("%d %s %c",&s.rn,s.name,&s.grade);
    display(s);
    getche();
}
```

```
display(m)
struct  student m;
{
        printf("\nRollno is %d",m.rn);
        printf("\n Name is %s",m.name);
        printf("\n Grade is: %c",m.grade);
        grade;
}
```

Passing of structure variables by value to a function

```
struct date
{
    int day;
    int month;
    int year;
};
void print_date(struct date);
void main()
{
    struct date d={10,12,1997};

    print_date(d);
}
void  print_date(struct date a)
{
    printf("\nDate in format %d %d %d",a.day,a.month,a.year);
```

Passing of structure variables by reference to a function

```
struct  date
{
    int day;
    int month;
    int year;
};
main()
{
    struct date d;
    void get_date(struct date *);
    printf("Enter date in the format:\n");
    get_date(&d);
    printf("\nDate entered by you %d %d %d",d.day,d.month,d.year);
}
void get_date(struct date *a)
{   scanf("%d %d %d",&a->day,&a->month,a->year);
}
```

Function Returning Structure

```
struct date
{
    int day;
    int month;
    int year;
};
struct date get_date(void);

main()
{
    struct date d;
        printf("\nEnter date in format day/month/year");
    d=get_date();
    printf("\nDAte entered by you is %d %d %d\n",d.day,d.month,d.year);
}
```

```
struct  date  get_date(void)
{
        struct date a;
        scanf("%d %d %d",&a.day,&a.month,&a.year);
        return a;
}
```

Array of Structures

If we wish to process a list of values of structure type,then we need an array of such structures.

Declaring an Array of Structures

```
struct employee_type
{
   int code;
   char name[25];
   char dept[15];
   float salary;
};
struct  employee_type employee[50];
```

Accessing elements an array of structures

Individual elements of a structure in an array of structure are accessed by referring to

structure variable name.

1. Followed by subscript.
2. Followed by dot operator.
3. Ending with structure element desired.

Suppose we want to access salary of 7th employee,we can do so by writing

employee[6].salary

Union

Union:

Union is similar as structure. The major distinction between them in terms of storage.

In structure each member has its own storage location whereas all the members of union uses the same location.

The union may contain many members of different data type it can handle only one member at a time union can be declared using the keyword union.

Union item

```
{
int m;
float x;
char c;
} code;
```

This declare a variable code of type union item.