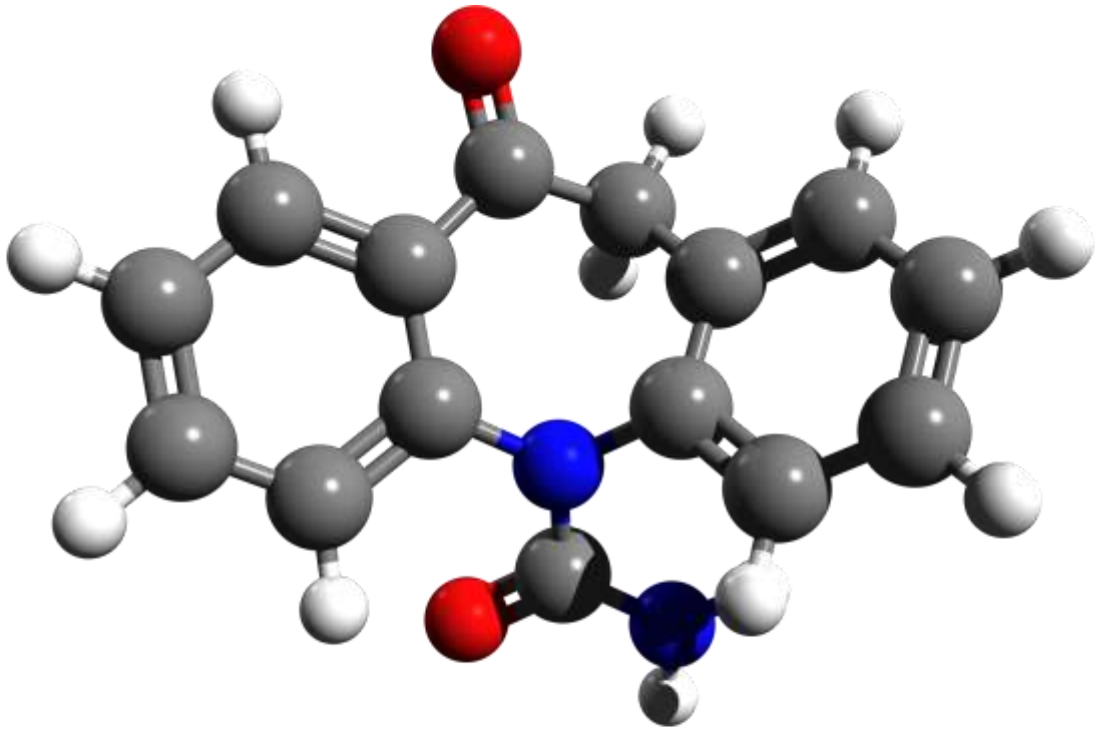
Structure & Union



Content

Introduction

- User defined data type
- Collection of heterogeneous data
- Referred by a common name
- A function can return a structure

Definition

- Collection of logically related data with same or different data-type, the logically related data are grouped together in a common name.
- Structure Tag
- Data Member or Fields

Syntax

- **struct** tag_name
  ```
  {
        data-type  Field1;
         data-type  Field2;
        ………
  };
  ```

Example

-
  ```
  struct    book
  {
        char name[20];
        char author[10];
        int pages;
        float price;
  } ;
  ```

Declaration of Variable

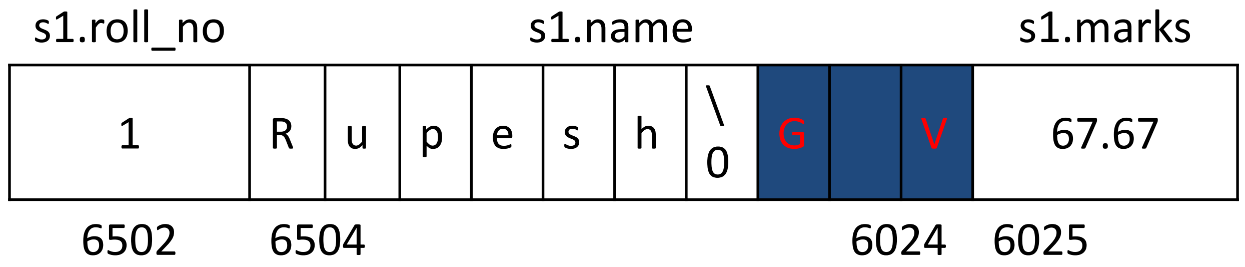
-
  ```
  struct   student
  {
        int roll_no;
        char name[20];
        float marks;
  } s1,*s2;
  ```

Using Structure Tag

- struct struct_tag variable-list;
- struct student s1,*s2;
- Variable can be declared inside(local) or outside(global) of main function

Memory Allocation

- struct student s1 = {1,"Rupesh",67.67};



- N=sizeof(s1);
- N ????

Program - Structure

```
#include <stdio.h>
struct student
{
    char name[50];
    int roll;
    float marks;
};
```

```
int main()
{
    struct student s;          // Structure Variable


    printf("Enter information of students:\n\n");
    printf("Enter name: ");  scanf("%s",s.name);
    printf("Enter roll number: ");  scanf("%d",&s.roll);
    printf("Enter marks: ");  scanf("%f",&s.marks);
printf("\nDisplaying Information\n");
    printf("Name: %s\n",s.name);
    printf("Roll: %d\n",s.roll);
    printf("Marks: %.2f\n",s.marks);
    return 0;
}
```

typedef

- It is used to give a new symbolic name (alias) to the existing entity of program.

- typedef existing name alias_name;

- typedef int unit;
- unit a,b;

typedef with structure

- **typedef struct** tag_name

```
{
      data-type  Field1;
       data-type  Field2;
      ………
} alias_name;
```

- alias_name variable list;

typedef with structure

-
  ```
  typedef struct    book
  {
        char name[20];
        char author[10];
        int pages;
        float price;
  } book_bank;
  ```

- `book_bank b1,b2;`

Accessing Structure Member

- Member Access Operator
  - Dot (.) operator
  - s1.name
- Structure Pointer Operator
  - Arrow (->) Operator
  - s2->name

Nested Structure

- struct date
  ```
  {
          int  day, mon, year;
  };
  ```

- Struct emp
  ```
  {
          char name[20];
          struct date birthday;
          float salary;
  };
  ```

Nested Structure

```
Struct emp
   {
         char name[20];
         struct date
         {
         int  day, mon, year;
         } birthday;

         float salary;
   };
```

Nested Structure

- struct emp e1 = {“Ratan”,{28,12,1937},68.4};
- printf(“name =%s”,e1.name);
  - Output : Ratan
- Printf(“Month of Birth=%d”,e1.birthday.mon);
  - Output ??

Nested Structure

- struct emp e1 = {“Ratan”,{28,12,1937},68.4};
- printf(“name =%s”,e1.name);
  - Output : Ratan
- Printf(“Month of Birth=%d”,e1.birthday.mon);
  - Output : 12

Array of Structure

-
  ```
  struct   student
  {
      int roll_no;
      char name[20];
      float marks;
  } s[100];
  ```

- **Program - Array of Structure**

```
#include <stdio.h>
struct employee
{
  int  emp_id;
  char name[20];
  float salary;
  int dept_no;
  int age;
};
```

```
Int main ( )
  {
    struct employee e[50];
    int I,n;

    printf(Enter No of Employee );
    Scanf("%d",&n);
```

```
for (i=0; i<n; i++)
    {
        printf (“Enter employee id, name salary,
        departement id and age  of employee”);
        scanf (“%d”,&e[i].emp_id);
        scanf (“%s”,e[i].name);
        scanf (“%f”,&e[i].salary);
        scanf (“%d”,&e[i].dept_no);
        scanf (“%d”,&e[i].age);
    }
```

```
for (i=0; i<n; i++)
   {
        printf (“Employee id : %d”, e[i].emp_id);
        printf (“Name : %s”,e[i].name);
        printf (“Salary : %f”, e[i].salary);
        printf (“Department ID: %d”, e[i].dept_no);
        printf (“Age : %d”, e[i].age);
  }
  return 0;
  }
```

Structure in Function

- Parameter Passing
- Return Value

Passing Parameter

```
#include <stdio.h>
struct employee
{
   int  emp_id;
   char name[20];
   float salary;
};
```

```
Int main ( )
  {
    struct employee e;

    printf (“Enter the employee id of employee”);
    scanf(“%d”,&e.emp_id);
    printf (“Enter the name of employee”);
    scanf(“%s”,e.name);
    printf (“Enter the salary of employee”);
    scanf(“%f”,&e.salary);

    printdata (struct employee e);                 // fn Call

   return 0;
  }
```

```
void printdata( struct employee emp)
 {
    printf (“\nThe employee id of employee is : %d”,emp.emp_id);
    printf (“\nThe name of employee is : %s”, emp.name);
    printf (“\nThe salary of employee is : %f”, emp.salary);
  }
```

Return Value

```
#include <stdio.h>

struct employee
{
   int  emp_id;
   char name[20];
   float salary;
};
```

```
void main ( )
  {
    struct employee emp;

    emp=getdata();                         // fn Call

    printf (“\nThe employee id is:%d”,emp.emp_id);
    printf (“\nThe name is : %s”, emp.name);
    printf (“\nThe salary is : %f”, emp.salary);
   return 0;
  }
```

```
struct employee getdata( )
 {
   struct employee e;
   printf (“Enter the employee id of employee”);
   scanf(“%d”,&e.emp_id);
   printf (“Enter the name of employee”);
   scanf(“%s”,e.name);
   printf (“Enter the salary of employee”);
   scanf(“%f”,&e.salary);
   return(e);
  }
```

Union

- Definition – Same as structure
- Syntax – Same as structure
- Difference in structure and union
  - Keyword – union
  - Memory allocation
  - Variable Storage

Syntax

- **union** tag_name

  ```
  {
        data-type  Field1;
         data-type  Field2;
        ………
  };
  ```

Memory Allocation

- union student

  {

        int roll_no;

        char name[20];

        float marks;

  } **s1,*s2**;

sizeof(s1)= ????

Memory Allocation

- union student

  {

      int roll_no;

      char name[20];

      float marks;

  } **s1,*s2**;

sizeof(s1)= 20 byte (Field with Max sixe: Name)