Structure in C

Structure

- Aggregate data type
- Composed of two or more related variables
  - Called member
  - Each member can be of different types

Structure (General Form)

```
struct tag-name {
    type member1;
    type member2;
    type member3;
    .
    .
    .
    type memberN;
    } variable-list;
```

- The keyword **struct** means that a structure *type* is defined
- *tag-name* is the name of the *type*
- Either *tag-name* or *variable-list* is optional

Structure Example

```
struct point {
    int x;
    int y;
} p1, p2;

struct {
    int x;
    int y;
} p1, p2;
```

Structure Example

```
struct point {
   int x;
   int y;
};
struct point p1, p2;
```

- Keyword **struct** before variable declaration is necessary
- Each instance of a structure contains its own copy of the members
- Structure declaration without any variable name does not reserve any storage
- Describes template or shape of a structure

Structure Initialization

```
p1.x=10;
p1.y=5;

struct point p3={5, 2};

p2={10, 5};//error, not possible
```

Structure Assignment

- Possible when type of the both objects are same

p2=p3;

Structure Member

```
printf("%d, %d\n", p3.x, p3.y);
scanf("%d %d\n", &p3.x, &p3.y);
```

Structure Array

```
#include <stdio.h>
struct point {
    int x;
    int y;
} ap[10] ;
int main(void)
{
    struct point p[10];
    int x;
    for(x=0; x<10; x++)
    {
            scanf("%d %d", &p[x].x, &p[x].y);
    }
    return 0;
}
```

Member x and int x are different

Nested Structure

```
#include <stdio.h>
struct point {
    int x;
    int y;
} p1, p2;
struct rect {
    struct point p1;
    struct point p2;
};
int main(void)
{
    struct rect r1;
    r1.p1.x=10;
    r1.p1.y=5;
    printf("%d, %d\n", r1.p1.x, r1.p1.y);
   return 0;
}
```

Function Returning Structure

```
struct point makepoint (int x, int y)
{
   struct point temp;
   temp.x=x;
   temp.y=y;
   return temp;
}
```