# C Structures

- The *structure* in *C* is a *user-defined data type* that ca
  types into a single type.
- *struct* keyword is used to define the structure.
- Items in the structure are called its *member* and the
- **Syntax:**

```
struct structureName {
        dataType member1;
        dataType member2;
        ...
};
```

- **Example:**

```
struct Person {
        char name[50];
        int citNo;
        float salary;
};
```

# C Structures | Create struct va

```
struct Person {
    // code
};
```

```
int main() {
    struct Person person1, person2, p[20];
    return 0;
}
```

OR

```
struct Person {
  // code
} person1, person2, p[20];
```

- *person1* and *p*
- *p[]* is a struct

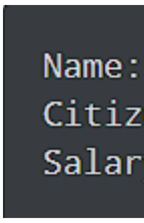
# C Structures | Access Membe

- There are two types of operators used for acces
  - .          Member operator
  - ->        Structure pointer operator

- Suppose, you want to access the *salary* of *perso*
  - person2.salary

# C Structures | Example

```
#include <stdio.h>
#include <string.h>

// create struct with person1 variable
struct Person {
  char name[50];
  int citNo;
  float salary;
} person1;

int main() {

  // assign value to name of person1
  strcpy(person1.name, "George Orwell");

  // assign values to other person1 variables
  person1.citNo = 1984;
  person1. salary = 2500;

  // print struct variables
  printf("Name: %s\n", person1.name);
  printf("Citizenship No.: %d\n", person1.citNo);
  printf("Salary: %.2f", person1.salary);

  return 0;
}
```



```
Name:
Citiz
Salar
```

Notice that we have used *strcpy(*

This is because *name* is a *char ar*
it after we have declared the stri

# C Structures |Nested Structur

-
  ```
  struct complex {
      int imag;
      float real;
  };
  ```

-
  ```
  struct number {
       struct complex comp;
      int integers;
  } num1, num2;
  ```

- Suppose, you want to set *imag* of *num2* variable
  - *num2.comp.imag = 11;*

# C structs and Pointers

```
struct name {
   member1;
   member2;
   .
   .
};
```

```
int main()
{
   struct name *ptr, Harry;
}
```

Here, *ptr* is a pointer to struct

# C structs | Access members us

```
#include <stdio.h>
struct person
{
  int age;
  float weight;
};
int main()
{
   struct person *personPtr, person1;
   personPtr = &person1;

   printf("Enter age: ");
   scanf("%d", &personPtr->age);

   printf("Enter weight: ");
   scanf("%f", &personPtr->weight);

   printf("Displaying:\n");
   printf("Age: %d\n", personPtr->age);
   printf("weight: %f", personPtr->weight);

   return 0;}
```

- In this example, the address of *per*
- Now, we can access the members

- *personPtr->age* is equivalent to *(*p*
- *personPtr->weight* is equivalent to

# C structs | Access members us

```
struct Point {
   int x, y;
};

int main()
{
   struct Point str = { 1, 2 };

   // p2 is a pointer to structure p1
   struct Point* ptr = &str;

   // Accessing structure members using structure pointer
   printf("%d %d", ptr->x, ptr->y);

   return 0;
}
```