Programming and Problem Solving
Structures in C

R.M.U.A. Rathnayake

What is a struct?

- In Object Oriented Languages, the concepts like “Class” and “Object” is present
- C is not an object-oriented language and it does not support for classes (No complex data structures like objects can be used)
- But C supports for defining structured types (like the data part of classes)
- A struct is a type used to represent a heterogeneous collection of data

What is a struct?

- In another way, a set of different types as a single, coherent unit
- In C programming, a struct (or structure) is a collection of variables (can be of different types) under a single name
- Example, a student may have a name, age, gpa, and graduation year
- A struct type can be defined to store these four different types of data associated with a student

Structure vs array

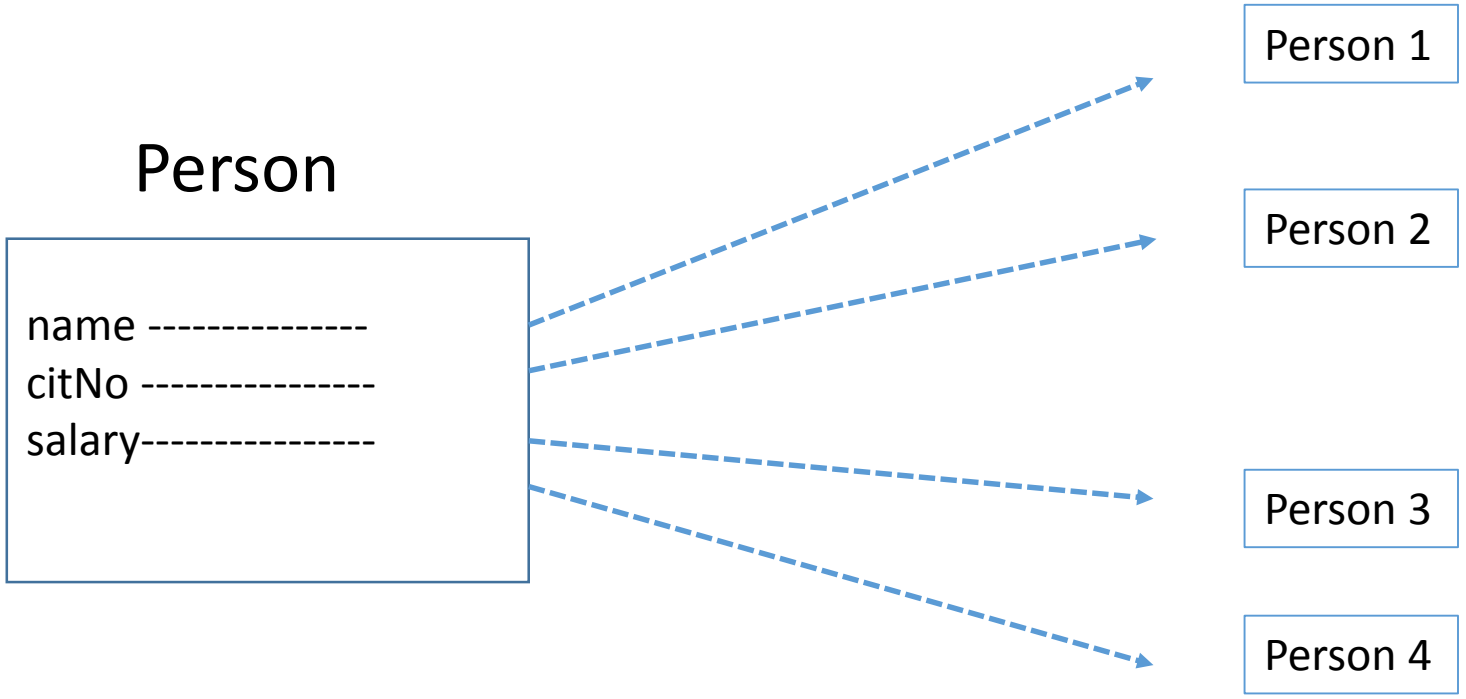
- Structure helps to construct a complex data type which is more meaningful
- An array can be seen as a similar structure which holds multiple items
- How it differs from array ??

Why structs in C

- Structs allow to store multiple data related to a particular concept
  - Eg: store information about a person: name, citizenship number, and salary
- You can create different variables name, citNo and salary to store this information
- But if you need to store information of more than one person?
  - Need to create different variables for each information per person
  - Eg: name1, citNo1, salary1, name2, citNo2, salary2, etc.
  - What if the number of persons increase? (imagine 1000 for example)

Why structs in C

- A better approach would be to have a collection of all related information under a single name Person structure and use it for every person



Defining a struct type

- Struct type definitions should appear near the top of a program file
- Should be outside of any function definition
- “struct” keyword is used to define the structure in C
- Boundary of the struct definition can be identified by “{“ and “}” symbols

Syntax to define a structure in C

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

```
struct <struct name> {
        <field 1 type> <field 1 name>;
        <field 2 type> <field 2 name>;
        <field 3 type> <field 3 name>;
        ...
};
```

Example

- Here is an example of defining a new type 'struct student' for storing student data

```
struct student {
        char name[64];
        int  age;
        int  gradYear;
        float gpa;
};
```

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

When you write the program

```c
#include <stdio.h>
struct student {
    char name[50];
    int age;
    int gradYear;
    float gpa;
};
int main()
{

    return 0;
};
```

Structure Variables

- Variables used to hold individual data
- When a struct type is declared, no storage or memory is allocated
- To allocate memory of a given structure type and work with it, we need to create variables

Creating struct variables

```
struct student  student1;
```

student1 is a struct type of **student**

```
student  student2;
```

student2 is also a struct **student** (we are just using the typedef alias name)

```
student csStd[10];
```

an array of **student** structs: each bucket stores a studentT struct

Example 1

```
struct person {
        char name[50];
        int citNo;
        float salary;
};
int main() {
        struct person person1, person2, p[20];
        return 0;
}
```

Structure definition

Creating variables of structure for using the structure

Example 2

```
struct person {
        char name[50];
        int citNo;
        float salary;
} person1, person2, p[20];
```

In both cases, two variables person1, person2, and an array variable p having 20 elements of type struct "person" will be created

How to access members of a structure

- There are two types of operators used for accessing members of a structure.

1. . - Member operator
2. -> - Structure pointer operator

- If you want to access the salary of person2. Here's how you can do it.

```
person2.salary
```

Passing structs to functions

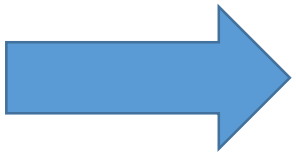
- A structure can be passed to any function from main function or from any sub function
- Structure definition is available within the function it is defined
- Not visible to other functions unless it is passed
  - by value
  - by address(reference)
- Declaring structure variable as global variable fix the scope issue
  - structure variable declare outside the main function

Three main ways to work

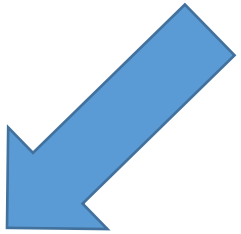
- Passing structure to a function by value
- Passing structure to a function by address(reference)
- No need to pass a structure
  - Declare structure variable as global

Passing structure to function in C by value

```c
struct Product{
  int pid;
  char pname[20];
  float price;
};
```



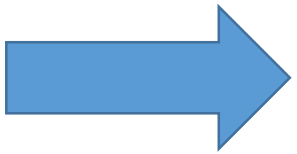
```c
printProduct(item1);
```



```c
void printProduct(struct Product item){

    printf("Product name : %s\n", item.pname);
    printf("Product id : %d\n", item.pid);
    printf("Product price : %f", item.price);
}
```

Passing structure to function in C by address

```c
struct Product{
  int pid;
  char pname[20];
  float price;
};
```

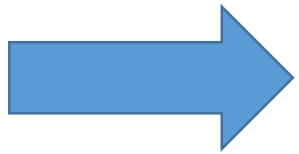


```c
printProduct(&item1);
```

```c
void printProduct(struct Product *item){

    printf("Product name : %s\n", item->pname);
    printf("Product id : %d\n", item->pid);
    printf("Product price : %f", item->price);
}
```

Define structure as global in C

```c
struct Product{
  int pid;
  char pname[20];
  float price;
};
struct Product item1;

void printProduct();
```

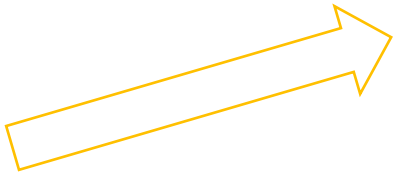


```c
printProduct();
```

```c
void printProduct(){

    printf("Product name : %s\n", item1.pname);
    printf("Product id : %d\n", item1.pid);
    printf("Product price : %f", item1.price);
}
```

Return struct from function

```c
struct Person p1;

p1 = getInfo();

printf("Person name is: ");
printf("%s\n",p1.name);
printf("The age is: ");
printf("%d",p1.age);
```



```c
struct Person getInfo(){

    struct Person p;

    printf("Enter your name here: ");
    scanf("%[^\n]%*c",p.name);
    printf("Enter your age: ");
    scanf("%d",&p.age);

    return p;
}
```

Questions ??