#16 Structure

By Saurabh Shukla | 2017©mysirg.com

Structure

Structure is a collection of heterogeneous data. Just like an array it is also a group of variables but can be of dissimilar types. Using structure we can create data type known as user defined data type. Such data types are also known as non-primitive data types.

Comparing structures with arrays, both are collection of memory blocks used to store multiple data. Array variables do not have names but have indexes, here in structure member variables do have names and can be of different data types.

Defining a structure

Defining a structure means telling a compiler about this new data type.

Syntax

```
struct <tag>
{
 Type variableName;
 Type variableName;
...
};
```

- struct is a keyword.
- <tag> is any name of your choice. It becomes the name of user defined data type.
- Type is any data type
- Variables declaring inside structure body are called member variables.
- Remember to put semicolon at the end of structure body.
- You can define a structure either inside a function block or outside a function. Former is known as local definition and later is known as global definition of structure. Locally defined structures can only be used in the body of block. Globally defined structures can be used anywhere in the program.

Using structure

Using structure means using new data type defined with the help of struct keyword.

Just like we declare variables of primitive types(int, char, float, etc), we can declare variables of user defined data type.

Example

```
#include<stdio.h>
struct student
{
    int rollno;
    char name[20];
};
int main()
{
    struct student s1;
    ...
}
```

- Once you have defined the structure you can use it as a data type in your program.
- Every time you use data type which is defined with the help of struct keyword, must be preceded by struct keyword. Observe the statement in main(), student is prefixed with struct.
- Student is a non-primitive or user defined data type
- s1 is a variable of type student.
- s1 occupies 24 bytes in the memory; first 4 bytes for 'rollno' and next 20 bytes are for char array 'name'.
- You can define a structure either inside a function or outside a function. When defined inside a function, it is known as local definition and can only be used by that function only. If the definition resides outside the function, it is known as global definition and can be used anywhere in the program. In the above example we have use the later one.
- You can also declare variables during defining a structure as follows

```
struct student
{
    int rollno;
    char name[20];
}s1,s2;
```

Accessing members of the structure

Accessing structure members is possible via membership operator (.).

Defining a structure doesn't mean creating a variable. No memory will be occupied by the structure until it is used to create variables.

Example

```
#include<stdio.h>
struct student
{
    int rollno;
    char name[20];
};
int main()
{
    struct student s1;
    printf("Enter rollno of student\n");
    scanf("%d",&s1.rollno);
    printf("Enter name of student\n");
    gets(s1.name);
    printf("You have entered:");
    printf("Roll no: %d",s1.rollno);
    printf("Name : %s",s1.name);
    return(0);
}
```

Structure and function

We can pass a structure variable in a function just like we pass values of primitive types.

Similarly we can return a structure.

Following is a sample program to handle student data using functions

Example

```
#include<stdio.h>
struct student
{
    int rollno;
    char name[20];
};
struct student input(void);
void display(struct student);
int main()
{
    struct student s1;
    s1=input();
    printf("You have entered:");
    display(s1);
    return(0);
}

struct student input()
```

```
{

    struct student s;
    printf("Enter rollno of student\n");
    scanf("%d",&s.rollno);
    printf("Enter name of student\n");
    gets(s.name);
    return(s);
}
void display(struct student s)
{
    printf("Roll no: %d",s.rollno);
    printf("Name : %s",s.name);
}
```

Structure and pointer

As we know from previous chapters, the type of pointer should be the same as the type of block whose address is contained by that pointer. Thus to contain address of structure variable, the type of the pointer should also be of same type.

```
struct student *p;
struct student s1;
p=&s1;
```

In the above code, p is a pointer of type student, so it can point to any student block.s1 is a variable of type student, hence we can assign address of s1 in p.
Accessing members of structure s1 through p need a different operator called member access operator (->), popularly known as arrow operator.

Following is the changed version of input function that takes an address of s1 in structure pointer. It inputs data and store them directly to the variable s1, so no need to return any value.

```
void input(struct student* p)
{
    printf("Enter rollno of student\n");
    scanf("%d",&p->rollno);
    printf("Enter name of student\n");
    gets(p->name);
}
```

Complete example

```
#include<stdio.h>
struct student
{
    int rollno;
    char name[20];
};
void input(struct student);
void display(struct student);
int main()
{
    struct student s1;
    input(&s1);
    printf("You have entered:");
    display(s1);
    return(0);
}

void input(struct student* p)
{
    printf("Enter rollno of student\n");
    scanf("%d",&p->rollno);
    printf("Enter name of student\n");
    gets(p->name);
}

void display(struct student s)
{
    printf("Roll no: %d",s.rollno);
    printf("Name : %s",s.name);
}
```

Array of Structure

You can create array of structure just like array of any type. In the following example we have created an array of size 4 of type student in main function. We take input from user by calling input function four times for each variable of array. Similarly we display their values.

```
#include<stdio.h>
struct student
{
    int rollno;
    char name[20];
};
void input(struct student);
void display(struct student);
```

```
int main()
{
    struct student s[4];
    int i;
    for(i=0;i<=3;i++)
         input(&s[i]);
    for(i=0;i<=3;i++)
    {
         printf("\nStudent - %d\n",i+1);
         display(s[i]);
    }
    return(0);
}
void input(struct student* p)
{
    printf("Enter rollno of student\n");
    scanf("%d",&p->rollno);
    printf("Enter name of student\n");
    gets(p->name);
}

void display(struct student s)
{
    printf("Roll no: %d",s.rollno);
    printf("Name : %s",s.name);
}
```

References:

YouTube video links

- Lecture 16 Structure in C part-1
  - https://youtu.be/zdUhS4YSWHg?list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW
- Lecture 16 Structure in C part-2
  - https://youtu.be/7-HetiTtbSM?list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW
- Example program of structure
  - https://www.youtube.com/watch?v=R-meb8n8dxI&feature=youtu.be&list=PL7ersPsTyYt1d8g5qaxbE6sjWDzs4D_1v

Exercise

1) Write a program which defines structure date and demonstrate the use of it.
2) Write a program which defines structure time and demonstrate the use of it.
3) Write a program to manage employee data (empid, ename, salary, department). Define structure for employee, define functions for input and output employee data.
4) Define a structure to store one string and one int value. First is name of the movie and second is year of release. Create an array of structure to store at most 10 movie records. Write a program with following functions:
   a. void setRecord(struct movieRecord *p, char *movie, int year);
   b. void showRecord(struct movieRecord *p);