

stuct

Structure is a collection of variables of different types under a single name.

For example: You want to store some information about a person: his/her name, citizenship number and salary. You can easily create different variables `name`, `citNo`, `salary` to store these information separately.

However, in the future, you would want to store information about multiple persons. Now, you'd need to create different variables for each information per person: `name1`, `citNo1`, `salary1`, `name2`, `citNo2`, `salary2`

You can easily visualize how big and messy the code would look. Also, since no relation between the variables (information) would exist, it's going to be a daunting task.

A better approach will be to have a collection of all related information under a single name `Person`, and use it for every person. Now, the code looks much cleaner, readable and efficient as well.

This collection of all related information under a single name `Person` is a structure.

Syntax of structure

```
struct structure_name  
{  
    data_type member1;  
    data_type member2;  
    .  
    .  
    data_type member;  
};
```

NOTE THE SEMICOLON IN THE VERY LAST LINE

We can create the structure for a person as mentioned above as:

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

Structure variable declaration

When a structure is defined, it creates a user-defined type but, no storage or memory is allocated. For the above structure of a person, variable can be declared as:

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

int main()
{
    struct person person1, person2, person3[20];
    return 0;
}
```

Accessing members of a structure

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

1. Member operator(.)
2. Structure pointer operator(->) (is discussed in [structure and pointers tutorial](#))

Any member of a structure can be accessed as:

```
structure_variable_name.member_name
```

Suppose, we want to access salary for variable `person2`. Then, it can be accessed as:

```
person2.salary
```

EXAMPLE

```
#include <stdio.h>
struct Distance
{
    int feet;
    float inch;
} dist1, dist2, sum;

int main()
{
    printf("1st distance\n");

    // Input of feet for structure variable dist1
    printf("Enter feet: ");
    scanf("%d", &dist1.feet);

    // Input of inch for structure variable dist1
    printf("Enter inch: ");
    scanf("%f", &dist1.inch);

    printf("2nd distance\n");

    // Input of feet for structure variable dist2
    printf("Enter feet: ");
    scanf("%d", &dist2.feet);

    // Input of inch for structure variable dist2
    printf("Enter inch: ");
    scanf("%f", &dist2.inch);

    sum.feet = dist1.feet + dist2.feet;
    sum.inch = dist1.inch + dist2.inch;
    if (sum.inch > 12)
    {
        //If inch is greater than 12, changing it to feet.
        ++sum.feet;
        sum.inch = sum.inch - 12;
    }

    // printing sum of distance dist1 and dist2
    printf("Sum of distances = %d\'-%.1f\\\"", sum.feet, sum.inch);
    return 0;
}
```

Write a program that stores the information (name, roll and marks) of 10 students using structures.

This structure has three members: name (string), roll (integer) and marks (float). Then, we created a structure array of size 10 to store information of 10 students. Don't forget about loops.

Hints:

```
...  
struct student  
{  
    ??? name[50];  
    ??? roll;  
    ...  
} s[???];  
  
int main()  
{  
    int i;  
  
    printf("Enter information of  
students:\n");  
  
    ...
```

Go to the link below and solve all 10 struct problems.

DO NOT CHEAT!

Only look at the answer once you either have solved it or have been working on it for a long time. (If you are found to cheat, you will then have to bring 10 mars bars for everyone.... in the building)

Keep up the hard work.

<https://chortle.ccsu.edu/CPuzzles/PartG/CpuzzlesGsection01.html>