C Programming Structure

http://www.programiz.com/c-programming/c-structures

Structure is the collection of variables of different types under a single name for better handling. For example: You want to store the information about person about his/her name, citizenship number and salary. You can create these information separately but, better approach will be collection of these information under single name because all these information are related to person.

Structure Definition in C

Keyword struct is used for creating a structure.

Syntax of structure

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

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

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

This declaration above creates the derived data type **struct person**.

Structure variable declaration

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

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

Inside main function:
struct person p1, p2, p[20];
```

Another way of creating sturcture variable is:

```
struct person
{
    char name[50];
    int cit_no;
    float salary;
}p1 ,p2 ,p[20];
```

In both cases, 2 variables p1, p2 and array p having 20 elements of type **struct person** are created.

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(->) (will be discussed in structure and pointers chapter)

Any member of a structure can be accessed as:
structure_variable_name.member_name

Suppose, we want to access salary for variable p2. Then, it can be accessed as:

```
p2.salary
```

Example of structure

Write a C program to add two distances entered by user. Measurement of distance should be in inch and feet (Note: 12 inches = 1 foot)

```
#include <stdio.h>
struct Distance{
    int feet;
    float inch;
}d1,d2,sum;
int main(){
    printf("1st distance\n");
    printf("Enter feet: ");
    scanf("%d",&d1.feet);  /* input of feet for structure
variable d1 */
    printf("Enter inch: ");
    scanf("%f",&d1.inch);  /* input of inch for structure
variable d1 */
    printf("2nd distance\n");
    printf("Enter feet: ");
```

```
    scanf("%d",&d2.feet);  /* input of feet for structure
variable d2 */
    printf("Enter inch: ");
    scanf("%f",&d2.inch);  /* input of inch for structure
variable d2 */
    sum.feet=d1.feet+d2.feet;
    sum.inch=d1.inch+d2.inch;
    if (sum.inch>12){  //If inch is greater than 12, changing it
to feet.
        ++sum.feet;
        sum.inch=sum.inch-12;
    }
    printf("Sum of distances=%d\'-%.1f\"",sum.feet,sum.inch);
/* printing sum of distance d1 and d2 */
    return 0;
}
```

Output

```
1st distance
Enter feet: 12
Enter inch: 7.9
2nd distance
Enter feet: 2
Enter inch: 9.8
Sum of distances= 15'-5.7"
```

Keyword typedef while using structure

Programmer generally use typedef while using structure in C language. For example:

```
typedef struct complex{
  int imag;
  float real;
}comp;

Inside main:
comp c1,c2;
```

Here, typedef keyword is used in creating a type comp(which is of type as **struct complex**). Then, two structure variables c1 and c2 are created by this comp type.

Structures within structures

Structures can be nested within other structures in C programming.

```
struct complex
{
 int imag_value;
 float real_value;
};
struct number{
   struct complex c1;
   int real;
}n1,n2;
```

Suppose you want to access imag_value for n2 structure variable then, structure member n1.c1.imag_value is used.

C Programming Structure and Pointer

http://www.programiz.com/c-programming/c-structures-pointers

Pointers can be accessed along with structures. A pointer variable of structure can be created as below:

```
struct name {
    member1;
    member2;
    .
    .
};
-------- Inside function -------
struct name *ptr;
```

Here, the pointer variable of type **struct name** is created.

Structure's member through pointer can be used in two ways:

1. Referencing pointer to another address to access memory
2. Using dynamic memory allocation

/* Consider an example to access structure's member through pointer */

```
#include <stdio.h>
struct name{
   int a;
   float b;
};
int main(){
    struct name *ptr,p;
    ptr=&p;            /* Referencing pointer to memory address
of p */
    printf("Enter integer: ");
    scanf("%d",&(*ptr).a);
    printf("Enter number: ");
    scanf("%f",&(*ptr).b);
    printf("Displaying: ");
    printf("%d%f",(*ptr).a,(*ptr).b);
    return 0;
}
```

In this example, the pointer variable of type **struct name** is referenced to the address of p. Then, only the structure member through pointer can can accessed.

Structure pointer member can also be accessed using -> operator.

```
(*ptr).a is same as ptr->a
(*ptr).b is same as ptr->b
```

Accessing structure member through pointer using dynamic memory allocation

To access structure member using pointers, memory can be allocated dynamically using malloc() function defined under "stdlib.h" header file.

Syntax to use malloc()

```
ptr=(cast-type*)malloc(byte-size)
```

Example to use structure's member through pointer using malloc() function.

```
/* Example to use structure's member through pointer using malloc() function */
#include <stdio.h>
#include<stdlib.h>
struct name {
   int a;
   float b;
   char c[30];
};
int main(){
   struct name *ptr;
   int i,n;
   printf("Enter n: ");
   scanf("%d",&n);
   ptr=(struct name*)malloc(n*sizeof(struct name));
/* Above statement allocates the memory for n structures with pointer ptr
pointing to base address */
   for(i=0;i<n;++i){
       printf("Enter string, integer and floating number
respectively:\n");
       scanf("%s%d%f",&(ptr+i)->c,&(ptr+i)->a,&(ptr+i)->b);
   }
   printf("Displaying Infromation:\n");
   for(i=0;i<n;++i)
       printf("%s\t%d\t%.2f\n",(ptr+i)->c,(ptr+i)->a,(ptr+i)->b);
   return 0;
}
```

Output

```
Enter n: 2
Enter string, integer and floating number  respectively:
Programming
2
3.2
Enter string, integer and floating number  respectively:
Structure
6
2.3
Displaying Information
Programming    2      3.20
Structure      6      2.30
```

C Programming Structure and Function

Reference: http://www.programiz.com/c-programming/c-structure-function

In C, structure can be passed to functions by two methods:

1. Passing by value (passing actual value as argument)
2. Passing by reference (passing address of an argument)

Passing structure by value

A structure variable can be passed to the function as an argument as normal variable. If structure is passed by value, change made in structure variable in function definition does not reflect in original structure variable in calling function.

Write a C program to create a structure student, containing name and roll. Ask user the name and roll of a student in main function. Pass this structure to a function and display the information in that function.

```c
/* a C program to create a structure student, containing name and roll */
#include <stdio.h>
struct student{
    char name[50];
    int roll;
};
void Display(struct student stu);
/* function prototype should be below to the structure
declaration otherwise compiler shows error */
int main(){
    struct student s1;
    printf("Enter student's name: ");
    scanf("%s",&s1.name);
    printf("Enter roll number:");
    scanf("%d",&s1.roll);
    Display(s1);   // passing structure variable s1 as argument
    return 0;
}
void Display(struct student stu){
  printf("Output\nName: %s",stu.name);
  printf("\nRoll: %d",stu.roll);
}
```

Output

```
Enter student's name: Kevin Amla
Enter roll number: 149
Output
Name: Kevin Amla
Roll: 149
```

Passing structure by reference

The address location of structure variable is passed to function while passing it by reference. If structure is passed by reference, change made in structure variable in function definition reflects in original structure variable in the calling function.

Write a C program to add two distances(feet-inch system) entered by user. To solve this program, make a structure. Pass two structure variable (containing distance in feet and inch) to add function by reference and display the result in main function without returning it.

/* Write a C program to add two distances(feet-inch system) entered by user */

```
#include <stdio.h>
struct distance{
    int feet;
    float inch;
};
void Add(struct distance d1,struct distance d2, struct distance
*d3);
int main()
{
    struct distance dist1, dist2, dist3;
    printf("First distance\n");
    printf("Enter feet: ");
    scanf("%d",&dist1.feet);
    printf("Enter inch: ");
    scanf("%f",&dist1.inch);
    printf("Second distance\n");
    printf("Enter feet: ");
    scanf("%d",&dist2.feet);
    printf("Enter inch: ");
    scanf("%f",&dist2.inch);
    Add(dist1, dist2, &dist3);
```

```
/*passing structure variables dist1 and dist2 by value whereas
passing structure variable dist3 by reference */
    printf("\nSum of distances = %d\'-%.1f\"",dist3.feet,
dist3.inch);
    return 0;
}
void Add(struct distance d1,struct distance d2, struct distance
*d3)
{
/* Adding distances d1 and d2 and storing it in d3 */
     d3->feet=d1.feet+d2.feet;
     d3->inch=d1.inch+d2.inch;
     if (d3->inch>=12) {     /* if inch is greater or equal to
12, converting it to feet. */
          d3->inch-=12;
          ++d3->feet;
    }
}
```

Output

```
First distance
Enter feet: 12
Enter inch: 6.8
Second distance
Enter feet: 5
Enter inch: 7.5

Sum of distances = 18'-2.3"
```

Explaination

In this program, structure variables dist1 and dist2 are passed by value (because value of dist1 and dist2 does not need to be displayed in main function) and dist3 is passed by reference ,i.e, address of dist3 (&dist3) is passed as an argument. Thus, the structure pointer variable d3 points to the address of dist3. If any change is made in d3 variable, effect of it is seed in dist3 variable in main function.
- See more at: http://www.programiz.com/c-programming/c-structure-function#sthash.86tHcJkb.dpuf

C Program to Store Information(name, roll and marks) of a Student Using Structure

In this program, a structure(student) is created which contains name, roll and marks as its data member. Then, a structure variable(s) is created. Then, data (name, roll and marks) is taken from user and stored in data members of structure variable s. Finally, the data entered by user is displayed. –

See more at: http://www.programiz.com/c-programming/examples/structure-store-information#sthash.zQqD7I3T.dpuf

C Program to Store Information of Single Variable

```
#include <stdio.h>
struct student{
    char name[50];
    int roll;
    float marks;
};
int main(){
    struct student s;
    printf("Enter information of students:\n\n");
    printf("Enter name: ");
    scanf("%s",s.name);
    printf("Enter roll number: ");
    scanf("%d",&s.roll);
    printf("Enter marks: ");
    scanf("%f",&s.marks);
    printf("\nDisplaying Information\n");
    printf("Name: %s\n",s.name);
    printf("Roll: %d\n",s.roll);
    printf("Marks: %.2f\n",s.marks);
    return 0;
}
```

Output

- See more at: http://www.programiz.com/c-programming/examples/structure-store-information#sthash.D1b66Atk.dpuf

C Program to Store Information of 10 Students Using Structure

In this program, a structure(student) is created which contains name, roll and marks as its data member. Then, an array of structure of 10 elements is created. Then, data(name, roll and marks) for 10 elements is asked to user and stored in array of structure. Finally, the data entered by user is displayed.

Source Code to Store Information of 10 students Using Structure

```
#include <stdio.h>
struct student{
    char name[50];
    int roll;
    float marks;
};
int main(){
    struct student s[10];
    int i;
    printf("Enter information of students:\n");
    for(i=0;i<10;++i)
    {
        s[i].roll=i+1;
        printf("\nFor roll number %d\n",s[i].roll);
        printf("Enter name: ");
```

```
        scanf("%s",s[i].name);
        printf("Enter marks: ");
        scanf("%f",&s[i].marks);
        printf("\n");
    }
    printf("Displaying information of students:\n\n");
    for(i=0;i<10;++i)
    {
     printf("\nInformation for roll number %d:\n",i+1);
     printf("Name: ");
     puts(s[i].name);
     printf("Marks: %.1f",s[i].marks);
   }
   return 0;
}
```

Output

```
Enter information of students:

For roll number 1
Enter name: Tom
Enter marks: 98

For roll number 2
Enter name: Jerry
Enter marks: 89
.
.
.
Displaying information of students:

Information for roll number 1:
Name: Tom
Marks: 98
.
.
```

- See more at: http://www.programiz.com/c-programming/examples/information-structure-array#sthash.l6gpfl0X.dpuf

C Program to Add Two Distances (in inch-feet) System Using Structures

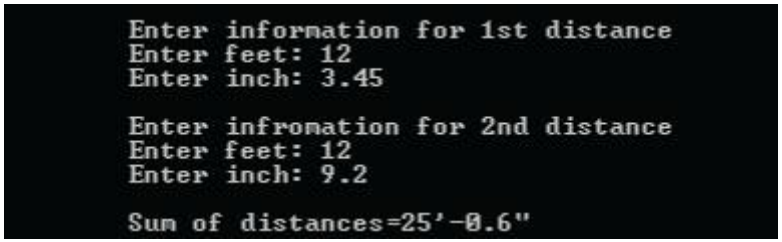
This program takes two distances in inch-feet system and stores in data members of two structure variables. Then, this program calculates the sum of two distances and displays it.

Source code to add two distance using structure

```
#include <stdio.h>
struct Distance{
    int feet;
    float inch;
}d1,d2,sum;
int main(){
    printf("Enter information for 1st distance\n");
    printf("Enter feet: ");
    scanf("%d",&d1.feet);
    printf("Enter inch: ");
    scanf("%f",&d1.inch);
    printf("\nEnter infromation for 2nd distance\n");
    printf("Enter feet: ");
    scanf("%d",&d2.feet);
    printf("Enter inch: ");
    scanf("%f",&d2.inch);
    sum.feet=d1.feet+d2.feet;
    sum.inch=d1.inch+d2.inch;

/* If inch is greater than 12, changing it to feet. */
    if (sum.inch>12.0)
    {
        sum.inch=sum.inch-12.0;
        ++sum.feet;
    }
    printf("\nSum of distances=%d\'-%.1f\"",sum.feet,sum.inch);
    return 0;
}
```

Output



In this program, a structure(Distance) is defined with inch and feet as its members. Then, three variables(d1, d2 and sum) of struct Distance type is created. Two variables(d1 and d2) are used for taking distance from user and the sum of two distance is stored in variable sum and then, displayed.

- See more at: http://www.programiz.com/c-programming/examples/inch-feet-structure#sthash.3tVPF2Up.dpuf