Learning C/C++ Step-By-Step - Page 10

Submitted by ganesh35 (Contact Author) (Forums) on Wed, 2009-01-07 18:34. ::

0 Tweet

10. Step-by-Step C/C++ --- C Programming - Structure

Structures

1. Introduction

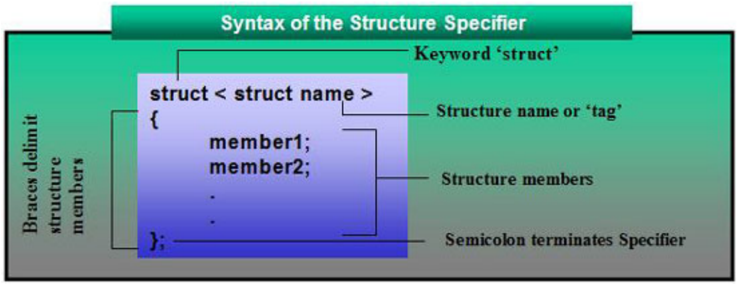
```
int a[4] = { 3, 4, 5, 6 };                    /* Valid expression */
int a[4] = { 3, 4.23, 5, 6 };                    /* Invalid expression */
int a[4] = { 3, "Siglov", 5,3}                    /* Invalid expression */
```

Why the last two expressions are invalid? An array can store values of same type. Must be the same type. Where as a structure can hold more than one type of data according to its definition.

• A group of one or more variables of different data types organized together under a single name is called a structure or
• A collection of heterogeneous (dissimilar) types of data grouped together under a single name is called a structure or
• A structure is a collection of simple variables. The variable in a structure can be of different types. The data items in a structure are called the members of the structures.

2. Declaration of a structure

When a structure is defined the entire group is referenced through the structure name. The individual components present in the structure are called as the structure members and these can be accessed and processed separately.



Eg:

```
struct date
{
      int day;
      int month;
      int year;
};
```

```
struct student
{
      int sno;
      char name[20];
      int marks;
      float avg;
};
```

3. Defining a Structure Variable

Defining a structure variable is the same as that for defining a built-in data type such as **int**.

```
int a;              /*  valid */
date d;             /* valid (But in C++ only ) */
struct date d;      /* valid in both C and C++  */
```

4. Initializing a Structure variable

The members of the structure can be initialized like other variables. This can be done at the time of declaration or at the design time.

1. Initialization at Declaration:

```
struct ddate
{
   int day;
   int month;
   int year;
} d = { 27, 10, 2000 };
```

2. Initialization at Definition:

struct ddate d = { 27, 10, 2000 };

1. **Initialization at design time:**

```
ddate d;
d.day = 27;
d.month = 10;
d.year = 2000;
```

4. Initialization at run time:

scanf("%d%d%d", &d.day, &d.month, &d.year);

Eg:

```
/* Write a program to accept and print the details of an employee */
/* 73_struct.c */
#include <stdio.h>
struct emp
{
           int eno;
           char name[20];
           float sal;
};
int main()
{
           struct emp e;

           printf("Enter Employee number    :"); scanf("%d", &e.eno);
           printf("Enter Employee name       :"); scanf("%s", e.name);
           printf("Enter Employee salary      :"); scanf("%d", &e.sal);
           printf("\n\nEmployee Details are as follows….\n");
           printf("%d    %s       %d", e.eno, e.name, e.sal);
           return 0;
}
```

5. Direct assignment of structures

Direct assignment of more than one variable is made possible using structures.

```
struct emp a, b = {1001, "Vimal", 6700.00 };
a = b; /* Valid */
printf("%d %s %d" , a.eno, a.name, a.sal );
```

Output:

```
1001 Vimal 6700.00
```

6. Calculation of structure size

Every data type in C/C++ has a specified size, i.e int has 2 bytes of size, float has 4 bytes of size and so on. Here is the way to find the size of a structure variable.

sizeof :- This function is used to find the size of a given variable.

```
printf("%d", sizeof(int));                          /* 2  */
printf("%d", sizeof(float));                       /* 4  */
printf("%d", sizeof(struct emp));              /* Displays the size of the emp structure */
```

7. Nested Structures

Structure with in structures in known as nested structures. For accessing nested structure members we must apply the dot operator twice in calling structure members.

Eg:

```
/* program to demonstrate nested structure with employee structure */
/* 74_nested.c */
#include <stdio.h>
struct emp
{
            int eno;
            char name[10];
            float sal;
           struct                                                  /* Nested Structure  */
            {
                        street char[10];
                        city char[10];
            } addr;
};
int main()
{
            struct emp e;
            printf("Enter emp_no, emp_name, emp_sal, street, city ");
            scanf("%d%s%d%s%s", &e.eno, e.name, &e.sal, e.addr.street, e.addr.city );
            printf("\n\nEmployee Details are as follows   ….\n");
            printf("%d%s%d%s%s", e.eno, e.name, e.sal, e.addr.street, e.addr.city );
            return 0;
}
```

8. Array of Structures

We can create an array of structures. The array will have individual structures as its elements.

```
/* Write a program to accept and print the details of an employee */
/* 75_array.c  */
#include <stdio.h>
struct emp
{
            int eno;
            char name[20];
            float sal;
};
int main()
{
            struct emp e[10];
            int i;
            for(i = 0; i<10; i++)
            {
                        printf("Enter Employee number    :"); scanf("%d", &e[i].eno);
                        printf("Enter Employee name       :"); scanf("%s", e[i].name);
                        printf("Enter Employee salary     :"); scanf("%d", &e[i].sal);
            }
            printf("\n\nEmployee Details are as follows….\n");
            for(i = 0; i<10; i++)
                         printf("%d    %s      %d", e[i].eno, e[i].name, e[i].sal);
            return 0;
}
```

Nothing is new in the above program. Entire program is same as simple structured program except the marked data.

9. Arrays with in Structures

There may be a situation to utilize arrays with in structures. How to achieve arrays with in structures. Here is the approach with simple program.

```
/* Program to accept and print a student information  */
/* 76_array.c */
#include <stdio.h>
struct stud
{
            int sno;
            char name[10];
            int marks[5];                              /* Array with in structure */
};
int main()
```

```
{
            struct stud s;
            int i;
            printf("Enter Student number  "); scanf("%d", &s.sno);
            printf("Enter Student name          "); scanf("%d", s.name);
            for( i = 0; i<3; i++)
            {
                        printf("Enter student marks  "); scanf("%d", &s.marks[i]);
            }
            printf("\n\nStudent Records is as follows….\n");
            printf("%d  %s  %d  %d  %d", s.sno, s.name, s.marks[0], s.marks[1], s.marks[2] );
            return 0;
}
```

10. Passing Structures to Functions

It is possible to send entire structures to functions as arguments in the function call. The structure variable is treated as any ordinary variable.

```
/* Program to pass a structure variable to function */
/* 77_funct.c */
#include <stdio.h>
struct emp
{
            int eno;
            char name[10];
            float sal;
};
void display(struct emp temp);
int main()
{
            struct emp e;
            display(e);
            return 0;
}
void display(struct emp temp)
{
            printf("%d  %s  %d", temp.eno, temp.name, temp.sal );
}
```

11. Returning Structures from functions

We can return structures from functions. Yes structures can be returned from functions just as variables of any other type.

```
/* Returning structure object from a function */
/* 78_funct.c */
struct emprec
{
            int eno;
            char name[10];
};
struct emprec read();
void write(struct emprec t);
int main()
{
            struct emprec e;
            e = read();
            write(e);
            return 0;
}
void write(struct emprec t)
{
            printf("\n\n%d  %s", t.eno, t.name);
}
struct emprec read()
{
            struct emprec t;
            printf("Enter Employee number   :"); scanf("%d",
&t.eno);
            printf("Enter Employee name         :"); scanf("%s",
t.name);
            return t;
}
```

12. Pointer to Structure

Till now we have seen that the members of a structure can be of data types like int, char, float or even structure. C/C++ language also permits to declare a pointer variable as a member to a structure. Pointer variables can be used to store the address of a structure variable also. A pointer can be declared as if it points to a structure data type.

```
/* Program to demonstrate the process of Pointer to structure */
/* 79_pointer.c */
#include <stdio.h>
struct employee
{
            int eno;
            char name[10];
};
struct employee *emp;
int main()
{
            emp = (struct employee * )malloc(sizeof(emp));
            printf("Enter Employee Details ..");
            scanf("%d%s%", &emp->eno, emp->name);
            printf("\n\n%d   %s", emp->eno, emp->name);
            return 0;
}
```

The marked data is essential to implement pointer to structure.
The following statement is optional, but better to utilize to organize better memory management.
emp = (struct employee *)malloc(sizeof(emp));

13. Structures Containing Pointers

A pointer variable can also be used as a member in the structure.
The following program contains pointer members contained by a pointer variable of structure.

```
/* program to demonstrate the use of structures containing Pointers */
/* 80_pointers.c  */
#include <stdio.h>
struct
{
            int *a;
            int *b;
} *temp;
int main()
{
            int x, y;
            x = 20; y = 50;
            rk -> a = &x;
            rk -> b = &y;

            printf("%d %d ", *temp->a, *temp->b );
            return 0;
}
```

output:

```
20   50
```

14. Self Referential Structures

Structures can have members, which are of the type the same structure itself in which they are included. This is possible with pointers and the phenomenon is called as self-referential structures.

```
struct emp
{
            int eno;
            char name[10];
            struct emp *e;
};
```

Self-referential structures can be used mainly in arranging data, sorting, searching elements, insertion, deletion of elements and so on.

This way of approach leads to Data structures (i.e., Linked Lists, Stacks, Queues, Trees and Graphs).

previous up next
Learning C/C++ Step-By-Step - Page 09 Learning C/C++ Step-By-Step - Page 11

0 Tweet

add comment | view as pdf | print: this | all page(s)

Related Tutorials

- Beginner's Guide To c++
- An Explanation of Pointers (C++)

Howtos | Mini-Howtos | Forums | News | Search | Contribute | Subscription
Site Map/RSS Feeds | Advertise | Contact | Disclaimer | Imprint