C Programming

Class- BCA IInd Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)

Department of Computer Application

Jagatpur P. G. College, Varanasi

Affiliated to Mahatma Gandhi Kashi Vidhyapith Varanasi

Email- dharmrajsingh67@yahoo.com

Outline

unit 4: Structure

- Array of Structures
- Additional Features of Structures
- Structures and functions
- Use of Typedef with Structures

Array of Structures

A better approach would be to use an array of structures. Following program shows how to use an array of structures.

```
 int main( )
{
struct book
{
char name[20] ;
float price ;
int pages ;
} ;
struct book b[3] ;
int i ;
for ( i = 0 ; i <= 2 ; i++ )
{
printf ( "\nEnter name, price and pages " ) ;
scanf ( "%s %f %d", b[i].name, &b[i].price, &b[i].pages ) ;
}
for ( i = 0 ; i <= 2 ; i++ )
printf ( "\n%s%f %d", b[i].name, b[i].price, b[i].pages ) ;
return 0;
}
```

Note:-The above C program is written and compiled in Turbo C. When this program is executed, the compiler displays the following error, *Scanf(): floating point formats not linked* and the program gets terminated abnormally. The solution is to call a floating-point (f-p)function or just add link of a file, which contains at least one floating-point (f-p) function.

➢It doesn't have to be in the module with the main program, as long as it's in a module that will be included in the link. Therefore, the above program should be written as follows:

```
void dummy(float *a)
{
 float b=*a;
 dummy(&b);
 }
struct book
{
char name[20] ;
float price ;
int pages ;
} ;
void main( )
{    struct book   b[3];
int i ;
for ( i = 0 ; i <= 2 ; i++ )
{
printf ( "\nEnter name, price and pages " ) ;
scanf ( "%s %f %d", b[i].name, &b[i].price,
&b[i].pages ) ;
}
for ( i = 0 ; i <= 2 ; i++ )
printf ( "\n%s%f %d", b[i].name, b[i].price,
b[i].pages ) ; getch();
}
```

Additional Features of Structures

The values of a structure variable can be assigned to another structure variable of the same type using the assignment operator.

```
 main( )
{
struct employee
{
char name[10] ;
int age ;
float salary ;
} ;
struct employee e1 = { "Sanjay", 30, 5500.50 } ;
struct employee e2, e3 ;
e2=e1;
e3 = e1 ; /* copying all elements at one go */
printf ( "\n%s %d %f", e1.name, e1.age, e1.salary ) ;
printf ( "\n%s %d %f", e2.name, e2.age, e2.salary ) ;
printf ( "\n%s %d %f", e3.name, e3.age, e3.salary ) ;
}
```

Additional Features of Structures

One structure can be nested within another structure. Using this facility complex data types can be created. The following program shows nested structures.

```
main( )
{
struct student
{
char phone[15] ;
char city[25] ;
int pin ;
} ;
struct emp
{
char name[25] ;
struct student s1;
} ;
struct emp s2 = { "raj", "531046", "nagpur", 10 };
printf ( "\nname = %s phone = %s", s2.name, s2.s1.phone ) ;
printf ( "\ncity = %s pin = %d", s2.s1.city, s2.s1.pin ) ;
}
```

```
struct address
{
char phone[15] ;
char city[25] ;
int pin ;
struct emp
{
char name[25] ;
char address1  [40];
} s1;

}s2;
```

WAP to read and display the information of a student using a nested structure.

```
void main()  {
struct DOB
{   int day;
    int month;
    int year;
};
struct student
{         char name[100];
           int roll_no;
          struct DOB d1;
};
struct student s1;
clrscr();
printf("\n Enter the name  and roll number : ");
scanf("%s  %d", s1.name, &s1.roll_no);
printf("\n Enter the DOB : ");
scanf("%d %d %d", &s1.d1.day, &s1.d1.month, &s1.d1.year);
printf("\n NAME = %s  ROLL No. = %d", s1.name, s1.roll_no);
printf("\n DOB = %d – %d – %d", s1.d1.day, s1.d1.month, s1.d1.year);
getch();
}
```

WAP to read and display the information of a student using a nested structure.

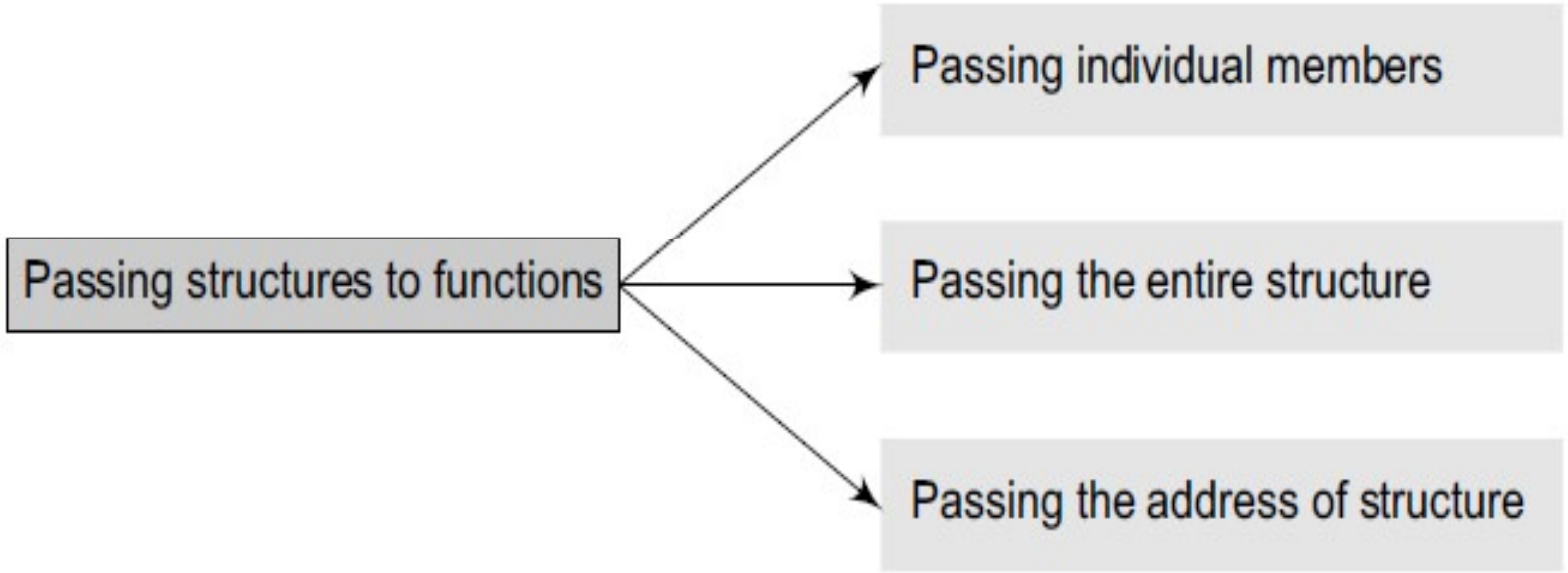
```
#include <stdio.h>
#include <string.h>
 struct student_college_detail
{    int college_id;
    char college_name[50];
};
 struct student_detail
{    int id;
    char name[20];
    float percentage;
    struct student_college_detail clg_data;         // structure within structure
}stu_data, *stu_data_ptr;
 int main()
{  struct student_detail stu_data = {1, "Raju", 90.5, 71145,  "Anna University"};
    stu_data_ptr = &stu_data;
     printf(" Id is: %d \n", stu_data_ptr->id);
    printf(" Name is: %s \n", stu_data_ptr->name);
    printf(" Percentage is: %f \n\n",      stu_data_ptr->percentage);
    printf(" College Id is: %d \n",    stu_data_ptr->clg_data.college_id);
    printf(" College Name is: %s \n",     stu_data_ptr->clg_data.college_name);
     return 0;
}
```

WAP to read and display the information of a student using a nested structure.

```
#include <stdio.h>
#include <string.h>
struct Employee
{    int id;
   char name[20];
   struct Date
    {  int dd;
      int mm;
      int yyyy;
    }doj;
}e1;
int main( )
{   e1.id=101;     //storing employee information
   strcpy(e1.name, "Sonoo Jaiswal");  //copying string into char array
   e1.doj.dd=10;
   e1.doj.mm=11;
   e1.doj.yyyy=2014;
    printf( "employee id : %d\n", e1.id);
   printf( "employee name : %s\n", e1.name);
   printf( ”date of joining (dd/mm/yyyy) : %d/%d/%d\n", e1.doj.dd,e1.doj.mm,e1.
doj.yyyy);
   return 0;
}
```

STRUCTURES AND FUNCTIONS

- we must have a mechanism to pass them to functions and return them. A function may access the members of a structure in three ways as shown in Fig.



Additional Features of Structures (Passing individual elements)

Like an ordinary variable, a structure variable can also be passed to a function. We may either pass individual structure elements or the entire structure variable at one go.

```
struct POINT
{
int x;
int y;
};
void display(int, int);
int main()
{
struct POINT p1 ; //= {2, 3};
printf("input the value of x and y");
scanf("%d%d", &p1.x, &p1.y);
display(p1.x, p1.y);
return 0;
}
void display(int a, int b)
{
printf(" The coordinates of the point are:
%d %d", a, b);
}
```

```
 void display ( char *, char *, int  );
int main( )
{
struct book
{
char name[25] ;
char author[25] ;
int callno ;
} ;
struct book  b1 = { "Let us C",
"YPK", 101 } ;
display ( b1.name, b1.author,
b1.callno ) ;
}
void display ( char *s, char *t, int
n )
{
printf ( "\n%s %s %d", s, t, n ) ;
}
```

Additional Features of Structures (Passing the Entire Structure)

A better way would be to pass the entire structure variable at a time. This method is shown in the following program.

```
struct book
{
char name[25] ;
char author[25] ;
int callno ;
} ;
void display( struct book  );
main( )
{
struct book  b1 = { "Let us C", "YPK", 101 } ;
display ( b1 ) ;
}
void display ( struct book  b )
{
printf ( "\n%s %s %d", b.name, b.author, b.callno ) ;
}
```

Additional Features of Structures

we can pass the address of a structure variable to a function. note that to access the structure elements using pointer to a structure we have to use the '**->**' **operator.**

```
/* Passing address of a structure variable */
struct book
{
char name[25] ;
char author[25] ;
int callno ;
} ;
void display ( struct book *b );
main( )
{
struct book b1 = { "Let us C", "YPK", 101 } ;
display ( &b1 ) ;
}
void display ( struct book *b )
{
printf ( "\n%s %s %d", b->name, b->author, b->callno ) ;
}
```

Additional Features of Structures

The way we can have a pointer pointing to an **int, or a pointer** pointing to a **char, similarly we can have a pointer pointing to** a **struct. Such pointers are known as 'structure pointers'.**

```
main( )
{
struct book
{
char name[25] ;
char author[25] ;
int callno ;
} ;
struct book  *b1 = { "Let us C", "YPK",
101 } ;
printf ( "\n%s %s %d", b1->name, b1-
>author, b1->callno ) ;
}
```

```
main( )
{
struct book
{
char name[25] ;
char author[25] ;
int callno ;
}*ptr ;
struct book  b1 = { "Let us C",
"YPK", 101 } ;
//struct book  *ptr ;
ptr = &b1 ;
printf ( "\n%s %s %d", b1.name,
b1.author, b1.callno ) ;
printf ( "\n%s %s %d", ptr->name,
ptr->author, ptr->callno ) ;
}
```

Use of Typedef with Structures

- The C language provides a keyword called **typedef** which we can use to synonyms for data types. Following is an example typedef

```
typedef struct Student
{
char[15] name;
int rollno;
int age;
} Stu;
```

Now that struct Student has been change to "Stu".

Exercise

1. Program to illustrate the relationship b/w union and structure
2. Write a C Program to Add Two Distances (in inch-feet) System Using Structures
3. Write C Program to Add Two Complex Numbers by Passing Structure to a Function
4. Write a C Program to Calculate Difference Between Two Time Periods.
5. Write a C Program to Store Information Using Structures with Dynamically Memory Allocation.
6. Write a C Program to implement employee record using pointers to structures.

References

- Kanetkar, Yashavant P. "Let Us C Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy
- Structured programming approach using C-Forouzah & Ceilberg Thomson learning Publication

Declaration

“The content is exclusively meant for academic purpose and for enhancing teaching and learning. Any other use for economic/commercial purpose is strictly prohibited. The users of the content shall not distribute, disseminate or share it with anyone else and its use is restricted to advancement of individual knowledge. The information provided in this e-content is authentic and best as per knowledge”.

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks