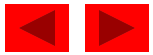
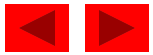
Chapter 10 - C Structures

Outline



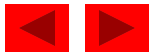
Objectives

- In this tutorial, you will learn:
  - To be able to create and use structures.
  - To be able to use pointers with structures.
  - To be able to use arrays with structures.
  - To be able to pass structures to functions call by value and call by reference.



10.1 Introduction

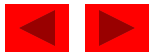
- Structures
  - Collections of related variables (aggregates) under one name
    - Can contain variables of different data types
  - Commonly used to define records to be stored in files
  - Combined with pointers, can create linked lists, stacks, queues, and trees
  - Structure is a user-defined data type in C language which allows us to combine data of different types together which is practical more useful.
  - Structure helps to construct a complex data type which is more meaningful.
  - It is somewhat similar to an Array, but an array holds data of similar type only.



10.1 Introduction

- **For example:**
  - If I have to write a program to store Books information, which will have Books's title, author, subject and bookID, which included string values and integer value, how can I use arrays for this problem, I will require something which can hold data of different types together.

- In structure, data is stored in form of **records**.

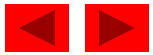


10.2 Structure Definitions

- struct keyword is used to define a structure. struct defines a new data type which is a collection of different datatypes.

- The format of the struct statement is as follows:

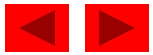
```
struct [structure_tag] {
        member_variable1 ;
        member_variable2 ;
        ….
        member_variableN;
}[structure_variables];
```

10.2 Structure Definitions

- The **structure tag** is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional.
- Here is the way you would declare the Books structure :

```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book;
```

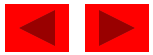



10.2 Structure Definitions

- Example

```
struct Books {
  char  title[50];
  char  author[50];
  char  subject[100];
  int   book_id;
}book;
```

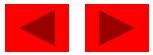
  - `struct` introduces the definition for structure `Books`
  - `Books` is the structure name and is used to declare variables of the structure type
  - `Books` contains three members of type `char` and one member of type **int**.
    - These members are `title,author,subject and book_id`
  - `book` is the structure variable



10.2 Structure Variable Definitions

- Definitions
  - Defined like other variables:
  - Can use a comma separated list:

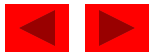
```
struct Books {
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
} book,b[3],*bPtr;
```

10.2 Structure Variable Definitions

- Declaring Structure variables separately

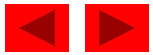
```
struct Student
{
    char name[25];
    int age;
    char gender; //F for female and M for male
};
struct Student s1;  //declaring variables of struct
Student
```

10.2 Structure Variable Definitions

- Declaring Structure variables with structure definition

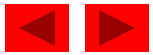
```
struct Student
{
    char name[25];
    int age;
    char gender; //F for female and M for male
}s1;
```

10.3 Accessing Members of Structures

- Structure members can be accessed and assigned values in a number of ways.
- Structure members have no meaning individually without the structure.
- In order to assign a value to any structure member, the member name must be linked with the **structure** variable using a dot.

structure_variable**.**member

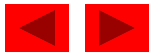


10.3 Accessing Members of Structures

- Example

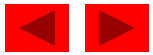
```
#include "stdafx.h"
#include<string.h>
struct Student
{
      char name[25];
      int age;
      char gender; //F for female and M for male
};
void main()
{
      struct Student s1;
      s1.age = 18;
      strcpy_s(s1.name, "Efe");
      s1.gender = 'M';
      printf("Name of Student: %s\n", s1.name);
      printf("Age  of Student: %d\n", s1.age);
      printf("Gender of Student: %c\n", s1.gender);
}
```

```
Name of Student: Efe
Age  of Student: 18
Gender of Student: M
```

it's optional to provide your structure a name, we suggest you to give it a name

```
#include "stdafx.h"
#include<string.h>
struct
{
    char name[25];
    int age;
    char gender;
}s1;
void main()
{
    s1.age = 18;
    strcpy_s(s1.name, "Efe");
    s1.gender = 'M';
    printf("Name of Student: %s\n", s1.name);
    printf("Age  of Student: %d\n", s1.age);
    printf("Gender of Student: %c\n", s1.gender);
}
```

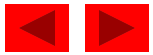
10.4 Initializing Structures

- Initializer lists
  - Example:

```
struct Student s1 = { "Efe",18 ,'M' };
```

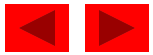
- Assignment statements
  - Example:

```
s1.age = 18;
```

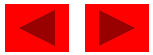



```c
#include "stdafx.h"
struct
{
	char name[25];
	int age;
	char gender;
}s1;
void main()
{
	s1 = { "Efe",18 ,'M' };
	printf("Name of Student: %s\n", s1.name);
	printf("Age  of Student: %d\n", s1.age);
	printf("Gender of Student: %c\n", s1.gender);

}
```

```c
#include "stdafx.h"
struct
{
	char name[25];
	int age;
	char gender;
}s1 = { "Efe",18 ,'M' };
void main()
{
	printf("Name of Student: %s\n", s1.name);
	printf("Age  of Student: %d\n", s1.age);
	printf("Gender of Student: %c\n", s1.gender);
}
```

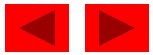
```c
#include "stdafx.h"
struct Student
{
    char name[25];
    int age;
    char gender;
};
void main()
{
    struct Student s1 = { "Efe",18 ,'M' };
    printf("Name of Student: %s\n", s1.name);
    printf("Age  of Student: %d\n", s1.age);
    printf("Gender of Student: %c\n", s1.gender);
}
```

10.5 Array of Structure

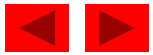
- We can also declare an array of **structure** variables. in which each element of the array will represent a **structure** variable.

Example : struct Student s1[3];



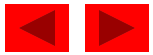
```c
#include "stdafx.h"
struct Student
{
	char name[25];
	int age;
	char gender;
}s1[2];
void main()
{
int i;
for (i = 0; i < 2; i++)
{
	printf("\nEnter %dst Student record:\n", i + 1);
	printf("Student name: ");
	gets(s1[i].name);
	printf("Enter age: ");
	scanf_s("%d", &s1[i].age);
	printf("Enter gender: ");
	getchar();
	s1[i].gender = getchar();
	getchar();
}
printf("\nDisplaying Student record:\n");
for (i = 0; i < 2; i++)
{
	printf("Name of %dst Student: %s\n", i+1,s1[i].name);
	printf("Age  of %dst Student: %d\n", i + 1, s1[i].age);
	printf("Gender  of %dst Student: %c\n\n", i + 1, s1[i].gender);
}
}
```

```
Enter 1st Student record:
Student name: Alpay
Enter age: 21
Enter gender: M

Enter 2st Student record:
Student name: Aya
Enter age: 20
Enter gender: F

Displaying Student record:
Name of 1st Student: Alpay
Age  of 1st Student: 21
Gender  of 1st Student: M

Name of 2st Student: Aya
Age  of 2st Student: 20
Gender  of 2st Student: F
```

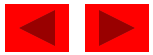
10.6 Using Structures With Functions and Pointers

- Passing structures to functions
  - Pass entire structure
    - Or, pass individual members
  - Both pass call by value
- To pass structures call-by-reference
  - Pass its address
  - Pass reference to it
- To pass arrays call-by-value
  - Create a structure with the array as a member
  - Pass the structure



```c
#include "stdafx.h"
struct Student
{
	char name[25];
	int age;
	char gender;
};
void Display(struct Student x);
void main()
{
	struct Student s1;
	printf("\nEnter Student record:\n");
	printf("Student name: ");
	gets(s1.name);
	printf("Enter age: ");
	scanf_s("%d", &s1.age);
	printf("Enter gender: ");
	getchar();
	s1.gender = getchar();
	getchar();
	Display(s1);
}
void Display(struct Student x)
{
	printf("\nDisplaying Student record:\n");
	printf("Name of Student: %s\n",  x.name);
	printf("Age  of Student: %d\n",  x.age);
	printf("Gender  of Student: %c\n\n",  x.gender);
}
```

Call by value



```
#include "stdafx.h"
struct Student
      {
      char name[25];
      int age;
      char gender;
}s1[2];
void Display(struct Student *);
void main()
{
int i;
for (i = 0; i < 2; i++){
      printf("\nEnter %dst Student record:\n", i + 1);
      printf("Student name: ");
      gets(s1[i].name);
      printf("Enter age: ");
      scanf_s("%d", &s1[i].age);
      printf("Enter gender: ");
      getchar();
      s1[i].gender = getchar();
      getchar();  }
      Display(&s1);
}
void Display(struct Student *x)
{
      printf("\nDisplaying Student record:\n");
      for (int i = 0; i < 2; i++) {
      printf("Name of %dst Student: %s\n", i + 1, (*(x + i)).name);
      printf("Age  of %dst Student: %d\n", i + 1, (*(x + i)).age);
      printf("Gender  of %dst Student: %c\n\n", i + 1, (*(x + i)).gender); }
}
```

Call by reference

10.6 Using Structures With Functions and Pointers

- You can define pointers to structures in the same way as you define pointer to any other variable

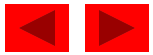
```
struct Student s1= { "Efe",18 ,'M' },*sp;
```

- Now, you can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the '&'; operator before the structure's name as follows

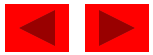
```
sp = &s1;
```

- To access the members of a structure using a pointer to that structure, you must use the → operator as follows

```
sp → age;    OR    (*sp).age
```

```c
#include "stdafx.h"
struct Student
{
    char name[25];
    int age;
    char gender;
}s1 = { "Efe",18 ,'M' }, *p;
void main()
{
    p=&s1;
    printf("Name of Student: %s\n", p->name);
    printf("Age  of Student: %d\n", (*p).age);
    printf("Gender of Student: %c\n", p->gender);
}
```

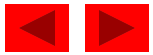



10.6 Nested Structure

- you can create structures within a structure

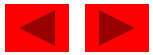
```
#include "stdafx.h“

struct student_college_detail
{
    int college_id;
    char college_name[50];
};

struct student_detail
{
    int id;
    char name[20];
    float percentage;
    // structure within structure
    struct student_college_detail clg_data;
}stu_data = { 1, "Raju", 90.5, 71145,"Anna University" };
```

```
void main()
{
    printf(" Id is: %d \n",
    stu_data.id);
    printf(" Name is: %s \n",
    stu_data.name);
    printf(" Percentage is: %f \n\n",
    stu_data.percentage);
    printf(" College Id is: %d \n",
    stu_data.clg_data.college_id);
    printf(" College Name is: %s \n",
    stu_data.clg_data.college_name);
}
```

- Nested structure with pointer

```
#include "stdafx.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;
};
void main()
{
     struct student_detail stu_data = { 1, "Raju", 90.5, 71145, "Anna University" },*stu_data_ptr;
     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);
}
```

- Nested Structure with Pointer and Function

```
#include "stdafx.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; }stu_data = { 1, "Raju", 90.5, 71145,"Anna
     University" };

void func(struct student_detail *);
void main()
{
func(&stu_data);
}
void func(struct student_detail *p)
{
     printf(" Id is: %d \n", (*p).id);
     printf(" Name is: %s \n", p->name);
     printf(" Percentage is: %f \n\n", (*p).percentage);
     printf(" College Id is: %d \n", (*p).clg_data.college_id);
     printf(" College Name is: %s \n", p->clg_data.college_name);
}
```