Structures in C

By

Prof. Muhammad Iqbal Bhat

Department of Higher Education

Government Degree College Beerwah

Topics:

1. What are Structures
2. Advantages of Using Structures
3. Syntax for creating Structures
4. Accessing Structure Members

What are Structures?

Structures are derived data types—they're constructed using objects of other types.

Structures are a fundamental concept in C programming, and they provide a way to group related data items of different types into a single unit.

The structure allows you to define a new data type, where each element has its own name and data type, and access them in a unified way.

Structures are used to represent complex data structures such as a student record, a customer order, or a product information with multiple fields.

Advantages of using

Structures help to organize data in a structured and meaningful way.

They allow you to create complex data structures that can be accessed and manipulated easily.

Structures provide an efficient way to store and retrieve data from memory.

Structures also help to reduce program complexity and improve readability and maintenance.

Syntax for Creating Structures:

The syntax for creating a structure in C is

```
struct <structure_name>
{
    data_type member1;
    data_type member2;
    ...
    data_type memberN;
};
```



```
struct employee {
    char firstName[20];
    char lastName[20];
    int age;
    double hourlySalary;
};
```

```
struct card {
    const char *face;
    const char *suit;
};
```

Defining Variables of Structure Types

A structure definition does not reserve any space in memory. Rather, it creates a new data type you can use to define variables.

It's like a blueprint showing how to build instances of that struct.

The following statements reserve memory for variables using the type struct card

```
struct card myCard;
struct card deck[52];
struct card *cardPtr;
```

```
struct card {
   const char *face;
   const char *suit;
} myCard, deck[52], *cardPtr;
```

Initializing Structures:

Like arrays, you can initialize a struct variable via an initializer list.

```
struct card {
   const char *face;
   const char *suit;
};
```



```
struct card myCard = {"Three", "Hearts"};
```

If there are fewer initializers than members, the remaining members are automatically initialized to 0 or NULL (for pointer members).

Accessing Structure Members:

You can access structure members with:

- the **structure member operator** (**.**), or dot operator,

```
printf("%s", myCard.suit); // displays Hearts
```

- the **structure pointer operator** (**->**), or **arrow operator**.

```
printf("%s", cardPtr->suit); // displays Hearts
```

equivalent to `(*cardPtr).suit`

Examples of Structure:

```c
#include <stdio.h>

// card structure definition
struct card {
   const char *face; // define pointer face
   const char *suit; // define pointer suit
};

int main(void) {
   struct card myCard; // define one struct card variable

   // place strings into myCard
   myCard.face = "Ace";
   myCard.suit = "Spades";

   struct card *cardPtr = &myCard; // assign myCard's address to cardPtr

   printf("%s of %s\n", myCard.face, myCard.suit);
   printf("%s of %s\n", cardPtr->face, cardPtr->suit);
   printf("%s of %s\n", (*cardPtr).face, (*cardPtr).suit);
}
```

```c
#include <stdio.h>
#include <string.h>

struct Employee {
    char name[50];
    int id;
    char job_title[50];
    float salary;
};
int main() {
    struct Employee emp1;
    strcpy(emp1.name, “Abdullah");
    emp1.id = 12345;
    strcpy(emp1.job_title, "Software Engineer");
    emp1.salary = 75000.00;

    printf("Employee name: %s\n", emp1.name);
    printf("Employee ID: %d\n", emp1.id);
    printf("Employee job title: %s\n", emp1.job_title);
    printf("Employee salary: $%.2f\n", emp1.salary);
    return 0;
}
```

```c
#include <stdio.h>
#include <string.h>
struct Student {
    char name[50];
    int id;
    int grade_level;
    float math_grade;
    float science_grade;
    float english_grade;
};
int main() {
    struct Student student1;
    strcpy(student1.name, “Abdullah");
    student1.id = 98765;
    student1.grade_level = 10;
    student1.math_grade = 92.5;
    student1.science_grade = 87.0;
    student1.english_grade = 94.5;
    printf("Student name: %s\n", student1.name);
    printf("Student ID: %d\n", student1.id);
    printf("Grade level: %d\n", student1.grade_level);
    printf("Math grade: %.1f\n", student1.math_grade);
    printf("Science grade: %.1f\n", student1.science_grade);
    printf("English grade: %.1f\n", student1.english_grade);

    return 0;
}
```

```c
#include <stdio.h>

struct Point {
    float x;
    float y;
};

int main() {
    struct Point p1, p2;

    printf("Enter coordinates of point 1 (x y): ");
    scanf("%f %f", &p1.x, &p1.y);

    printf("Enter coordinates of point 2 (x y): ");
    scanf("%f %f", &p2.x, &p2.y);

    printf("Point 1 coordinates: (%.1f, %.1f)\n", p1.x, p1.y);
    printf("Point 2 coordinates: (%.1f, %.1f)\n", p2.x, p2.y);

    return 0;
}
```

```c
#include <stdio.h>

struct Point {
    float x;
    float y;
};

int main() {
    struct Point p1, p2;

    printf("Enter coordinates of point 1 (x y): ");
    scanf("%f %f", &p1.x, &p1.y);

    printf("Enter coordinates of point 2 (x y): ");
    scanf("%f %f", &p2.x, &p2.y);

    printf("Point 1 coordinates: (%.1f, %.1f)\n", p1.x, p1.y);
    printf("Point 2 coordinates: (%.1f, %.1f)\n", p2.x, p2.y);

    return 0;
}
```

```c
#include <stdio.h>

struct Complex {
    float real;
    float imag;
};

int main() {
    struct Complex c1, c2, result;

    printf("Enter real and imaginary parts of complex number 1: ");
    scanf("%f %f", &c1.real, &c1.imag);

    printf("Enter real and imaginary parts of complex number 2: ");
    scanf("%f %f", &c2.real, &c2.imag);

    result.real = c1.real + c2.real;
    result.imag = c1.imag + c2.imag;

    printf("Sum of complex numbers: %.1f + %.1fi\n", result.real, result.imag);

    return 0;
}
```

```c
#include <stdio.h>

struct Employee {
    int emp_id;
    char name[50];
    int age;
};

int main() {
    struct Employee e1 = {101, “Abdullah", 30};
    struct Employee *ptr;

    ptr = &e1;

    printf("Employee ID: %d\n", ptr->emp_id);
    printf("Employee Name: %s\n", ptr->name);
    printf("Employee Age: %d\n", ptr->age);

    return 0;
}
```

```c
#include <stdio.h>

struct Book {
    char title[50];
    char author[50];
    int pages;
};
void display(struct Book *b);
int main() {
    struct Book b1 = {"The Alchemist", "Paulo Coelho", 197};
    struct Book *ptr;

    ptr = &b1;

    display(ptr);

    return 0;
}
void display(struct Book *b) {
    printf("Title: %s\n", b->title);
    printf("Author: %s\n", b->author);
    printf("Pages: %d\n", b->pages);
}
```

Questions?