**Structures in C**

Structure is a collection of variables (can be of different types) under a single name.

**For example:**You want to store information about a person: his/her name, citizenship number and salary. You can create different variables name, citNo and salary to store these information separately.

What if you need to store information of more than one person? Now, you need to create different variables for each information per person: name1, citNo1, salary1, name2, citNo2, salary2 etc.

A better approach would be to have a collection of all related information under a single name Person structure, and use it for every person.

**How to define a structure?**

Keyword struct is used for creating a structure.

**Syntax of structure**

struct structure\_name

{

data\_type member1;

data\_type member2;

.

.

data\_type memeber;

};

Here is an exampl:

struct Person

{

char name[50];

int citNo;

float salary;

};

Here, a derived type struct Person is defined.

**Create structure variable**

When a structure is defined, it creates a user-defined type. However, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables.

Here's how we create structure variables:

struct Person

{

char name[50];

int citNo;

float salary;

};

int main()

{

struct Person person1, person2, p[20];

return 0;

}

**Another way of creating a structure variable is:**

struct Person

{

char name[50];

int citNo;

float salary;

} person1, person2, p[20];

In both cases, two variables person1, person2, and an array variable p having 20 elements of type **struct Person** are created.

**How to Access members of a structure?**

There are two types of operators used for accessing members of a structure.

Member operator(.)

Structure pointer operator(->)

Suppose, you want to access salary of person2. Here's how you can do it:

person2.salary

**Example: C structure**

// Program to add two distances which is in feet and inches

#include <stdio.h>

struct Distance

{

int feet;

float inch;

} dist1, dist2, sum;

int main()

{

printf("1st distance\n");

printf("Enter feet: ");

scanf("%d", &dist1.feet);

printf("Enter inch: ");

scanf("%f", &dist1.inch);

printf("2nd distance\n");

printf("Enter feet: ");

scanf("%d", &dist2.feet);

printf("Enter inch: ");

scanf("%f", &dist2.inch);

// adding feet

sum.feet = dist1.feet + dist2.feet;

// adding inches

sum.inch = dist1.inch + dist2.inch;

// changing feet if inch is greater than 12

while (sum.inch >= 12)

{

++sum.feet;

sum.inch = sum.inch - 12;

}

printf("Sum of distances = %d\'-%.1f\"", sum.feet, sum.inch);

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**

Keyword typedef can be used to simplify syntax of a structure.

struct Distance{

int feet;

float inch;

};

int main() {

structure Distance d1, d2;

}

**is equivalent to**

typedef struct Distance{

int feet;

float inch;

} distances;

int main() {

distances dist1, dist2, sum;

}

**Example: Store Information and Display it Using Structure**

#include <stdio.h>

struct student

{

char name[50];

int roll;

float marks;

} s;

int main()

{

printf("Enter information:\n");

printf("Enter name: ");

scanf("%s", s.name);

printf("Enter roll number: ");

scanf("%d", &s.roll);

printf("Enter marks: ");

scanf("%f", &s.marks);

printf("Displaying Information:\n");

printf("Name: ");

puts(s.name);

printf("Roll number: %d\n",s.roll);

printf("Marks: %.1f\n", s.marks);

return 0;

}

**Output**

Enter information:

Enter name: Jack

Enter roll number: 23

Enter marks: 34.5

Displaying Information:

Name: Jack

Roll number: 23

Marks: 34.5

**Passing structure to a function**

# **C – Array of Structures**

As you know, C Structure is collection of different datatypes ( variables ) which are grouped together. Whereas, array of structures is nothing but collection of structures. This is also called as structure array in C.

#### **EXAMPLE PROGRAM FOR ARRAY OF STRUCTURES IN C:**

This program is used to store and access “id, name and percentage” for 3 students. Structure array is used in this program to store and display records for many students. You can store “n” number of students record by declaring structure variable as ‘struct student record[n]“.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | #include <stdio.h>  #include <string.h>    struct student  {       int id;       char name[30];       float percentage;  };    int main()  {       int i;       struct student record[2];         // 1st student's record       record[0].id=1;       strcpy(record[0].name, "Raju");       record[0].percentage = 86.5;         // 2nd student's record       record[1].id=2;       strcpy(record[1].name, "Surendren");       record[1].percentage = 90.5;         // 3rd student's record       record[2].id=3;       strcpy(record[2].name, "Thiyagu");       record[2].percentage = 81.5;         for(i=0; i<3; i++)       {           printf("     Records of STUDENT : %d \n", i+1);           printf(" Id is: %d \n", record[i].id);           printf(" Name is: %s \n", record[i].name);           printf(" Percentage is: %f\n\n",record[i].percentage);       }       return 0;  } |

#### **OUTPUT:**

|  |
| --- |
| **Records of STUDENT : 1** Id is: 1 Name is: Raju Percentage is: 86.500000    **Records of STUDENT : 2** Id is: 2 Name is: Surendren Percentage is: 90.500000    **Records of STUDENT : 3** Id is: 3 Name is: Thiyagu Percentage is: 81.500000 |

#### **EXAMPLE PROGRAM FOR DECLARING MANY STRUCTURE VARIABLE IN C:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #include <stdio.h>  #include <string.h>    struct student  {       int id;       char name[30];       float percentage;  };    int main()  {       int i;       struct student record1 = {1, "Raju", 90.5};       struct student record2 = {2, "Mani", 93.5};         printf("Records of STUDENT1: \n");       printf("  Id is: %d \n", record1.id);       printf("  Name is: %s \n", record1.name);       printf("  Percentage is: %f \n\n", record1.percentage);         printf("Records of STUDENT2: \n");       printf("  Id is: %d \n", record2.id);       printf("  Name is: %s \n", record2.name);       printf("  Percentage is: %f \n\n", record2.percentage);         return 0;  } |

#### **OUTPUT:**

|  |
| --- |
| **Records of STUDENT1:** Id is: 1 Name is: Raju Percentage is: 90.500000 **Records of STUDENT2:** Id is: 2 Name is: Mani Percentage is: 93.500000 |

**Passing structure to a function**

#include <stdio.h>

struct student

{

char name[50];

int age;

};

// function prototype

void display(struct student s);

int main()

{

struct student s1;

printf("Enter name:");

scanf ("%[^\n]%\*c", s1.name);

printf("Enter age:");

scanf("%d", &s1.age);

display(s1); // passing structure as an argument

return 0;

}

void display(struct student s)

{

printf("\nDisplaying information\n");

printf("Name: %s", s.name);

printf("\nRoll: %d", s.age);

}

When you run the program, the output will be:

Enter name: Bond

Enter age: 13

Displaying information

Name: Bond

Roll: 13

Here, a structure variable s1 of type struct student is created. This structure variable is passed to the display() function using display(s1); statement.

**Returning structure from a function**

Here's how you can return structure from a function:

#include <stdio.h>

struct student

{

char name[50];

int age;

};

// function prototype

struct student getInformation();

int main()

{

struct student s;

s = getInformation();

printf("\nDisplaying information\n");

printf("Name: %s", s.name);

printf("\nRoll: %d", s.age);

return 0;

}

struct student getInformation()

{

struct student s1;

printf("Enter name:");

scanf ("%[^\n]%\*c", s1.name);

printf("Enter age:");

scanf("%d", &s1.age);

return s1;

}

Here, getInformation() function is called using s = getInformation(); statement. The function returns a structure of type struct student. The returned structure is displayed from the main() function.

**C Program to Add Two Complex Numbers by Passing Structure to a Function**

#include <stdio.h>

typedef struct complex

{

float real;

float imag;

} complex;

complex add(complex n1,complex n2);

int main()

{

complex n1, n2, temp;

printf("For 1st complex number \n");

printf("Enter real and imaginary part respectively:\n");

scanf("%f %f", &n1.real, &n1.imag);

printf("\nFor 2nd complex number \n");

printf("Enter real and imaginary part respectively:\n");

scanf("%f %f", &n2.real, &n2.imag);

temp = add(n1, n2);

printf("Sum = %.1f + %.1fi", temp.real, temp.imag);

return 0;

}

complex add(complex n1, complex n2)

{

complex temp;

temp.real = n1.real + n2.real;

temp.imag = n1.imag + n2.imag;

return(temp);

}

**Output**

For 1st complex number

Enter real and imaginary part respectively: 2.3

4.5

For 2nd complex number

Enter real and imaginary part respectively: 3.4

5

Sum = 5.7 + 9.5i

In this program, structures n1 and n2 are passed as an argument of function add().

**C Programming Structure and Pointer**

[Structures](https://www.programiz.com/c-programming/c-structures" \o "C structures) can be accessed using [pointers](https://www.programiz.com/c-programming/c-pointers). Here's how:

struct name {

member1;

member2;

.

.

};

int main()

{

struct name \*ptr, Harry;

}

Here, a pointer ptr of type **struct name** is created. The pointer can access members of Harry.

**Example: Access structure members using pointer**

#include <stdio.h>

struct person

{

int age;

float weight;

};

int main()

{

struct person \*personPtr, person1;

personPtr = &person1;

printf("Enter age:");

scanf("%d", &personPtr->age);

printf("Enter weight:");

scanf("%f", &personPtr->weight);

printf("Displaying:\n");

printf("Age: %d\n", personPtr->age);

printf("weight: %f", personPtr->weight);

return 0;

}

In this example, the address of person1 is stored in personPtr pointer variable using code personPtr = &person1;.

Now, you can access members of person1 using personPtr pointer. For that we use ->operator.

By the way,

personPtr->age is equivalent to (\*personPtr).age

personPtr->weight is equivalent to (\*personPtr).weight

**Nested Structures**

You can create structures within a structure in C programming. For example:

struct complex

{

int imag;

float real;

};

struct number

{

struct complex comp;

int integers;

} num1, num2;

Suppose, you want to set imag of num2 variable to 11. Here's how you can do it:

num2.comp.imag = 11;

**Example program**

#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;

// structure within structure

struct student\_college\_detail clg\_data;

}stu\_data;

int main()

{

struct student\_detail stu\_data = {1, "Raju", 90.5, 71145,

"Anna University"};

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);

return 0;

}

**OUTPUT:**

|  |
| --- |
| Id is: 1 Name is: Raju Percentage is: 90.500000  College Id is: 71145 College Name is: Anna University |