C Structures (structs)

❮ Previous Next ❯

C Structures (structs)

Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a **member** of the structure.

Unlike an array, a structure can contain many different data types (int, float, char, etc.).

Create a Structure

You can create a structure by using the `struct` keyword and declare each of its members inside curly braces:

```
struct MyStructure {   // Structure declaration
  int myNum;           // Member (int variable)
  char myLetter;       // Member (char variable)
}; // End the structure with a semicolon
```

To access the structure, you must create a variable of it.

Use the `struct` keyword inside the `main()` method, followed by the name of the structure and then the name of the structure variable:

Create a struct variable with the name "s1":

```
struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  struct myStructure s1;
  return 0;
}
```

Access Structure Members

To access members of a structure, use the dot syntax (`.`):

Example

```
// Create a structure called myStructure
struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  // Create a structure variable of myStructure called s1
  struct myStructure s1;

  // Assign values to members of s1
  s1.myNum = 13;
  s1.myLetter = 'B';

  // Print values
  printf("My number: %d\n", s1.myNum);
  printf("My letter: %c\n", s1.myLetter);

  return 0;
}
```

Try it Yourself »

Now you can easily create multiple structure variables with different values, using just one structure:

Example

```
// Create different struct variables
struct myStructure s1;
struct myStructure s2;

// Assign values to different struct variables
s1.myNum = 13;
s1.myLetter = 'B';

s2.myNum = 20;
s2.myLetter = 'C';
```

Try it Yourself »

What About Strings in Structures?

Remember that strings in C are actually an array of characters, and unfortunately, you can't assign a value to an array like this:

Example

```
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];  // String
};

int main() {
  struct myStructure s1;

  // Trying to assign a value to the string
  s1.myString = "Some text";

  // Trying to print the value
  printf("My string: %s", s1.myString);

  return 0;
}
```

An error will occur:

```
prog.c:12:15: error: assignment to expression with array type
```

Try it Yourself »

However, there is a solution for this! You can use the `strcpy()` function and assign the value to `s1.myString`, like this:

Example

```
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30]; // String
};

int main() {
  struct myStructure s1;

  // Assign a value to the string using the strcpy function
  strcpy(s1.myString, "Some text");
  // Print the value
  printf("My string: %s", s1.myString);

  return 0;
}
```

Result:

```
My string: Some text
```

Try it Yourself »

Simpler Syntax

You can also assign values to members of a structure variable at declaration time, in a single line.

Just insert the values in a comma-separated list inside curly braces `{}`. Note that you don't have to use the `strcpy()` function for string values with this technique:

Example

```
// Create a structure
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  // Create a structure variable and assign values to it
  struct myStructure s1 = {13, 'B', "Some text"};

  // Print values
  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);

  return 0;
}
```

Try it Yourself »

Note: The order of the inserted values must match the order of the variable types declared in the structure (13 for int, 'B' for char, etc).

Copy Structures

You can also assign one structure to another.

In the following example, the values of s1 are copied to s2:

Example

```
struct myStructure s1 = {13, 'B', "Some text"};
struct myStructure s2;

s2 = s1;
```

Try it Yourself »

Modify Values

If you want to change/modify a value, you can use the dot syntax (`.`).

And to modify a string value, the `strcpy()` function is useful again:

Example

```
struct myStructure {
  int myNum;
  char myLetter;
  char myString[30];
};

int main() {
  // Create a structure variable and assign values to it
  struct myStructure s1 = {13, 'B', "Some text"};

  // Modify values
  s1.myNum = 30;
  s1.myLetter = 'C';
  strcpy(s1.myString, "Something else");

  // Print values
  printf("%d %c %s", s1.myNum, s1.myLetter, s1.myString);

  return 0;
}
```

Try it Yourself »

Modifying values are especially useful when you copy structure values:

Example

```
// Create a structure variable and assign values to it
struct myStructure s1 = {13, 'B', "Some text"};

// Create another structure variable
struct myStructure s2;

// Copy s1 values to s2
s2 = s1;

// Change s2 values
s2.myNum = 30;
s2.myLetter = 'C';
```

```
strcpy(s2.myString, "Something else");

// Print values
printf("%d %c %s\n", s1.myNum, s1.myLetter, s1.myString);
printf("%d %c %s\n", s2.myNum, s2.myLetter, s2.myString);
```

Try it Yourself »

Ok, so, how are structures useful?

Imagine you have to write a program to store different information about Cars, such as brand, model, and year. What's great about structures is that you can create a single "Car template" and use it for every cars you make. See below for a real life example.

Real Life Example

Use a structure to store different information about Cars:

Example

```
struct Car {
  char brand[50];
  char model[50];
  int year;
};

int main() {
  struct Car car1 = {"BMW", "X5", 1999};
  struct Car car2 = {"Ford", "Mustang", 1969};
  struct Car car3 = {"Toyota", "Corolla", 2011};

  printf("%s %s %d\n", car1.brand, car1.model, car1.year);
  printf("%s %s %d\n", car2.brand, car2.model, car2.year);
  printf("%s %s %d\n", car3.brand, car3.model, car3.year);
  return 0;
}
```

Try it Yourself »

C Exercises

Test Yourself With Exercises

Exercise:

Fill in the missing part to create a Car structure:

```
       Car {
  char brand[50];
  char model[50];
  int year;
};
```

Submit Answer »

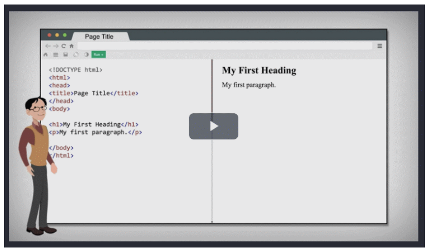
Start the Exercise

❮ Previous Next ❯

ADVERTISEMENT

NEW

We just launched
W3Schools videos



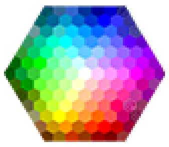
```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

My First Heading

My first paragraph.

Explore now

COLOR PICKER



Get certified
by completing
a course today!

ADVERTISEMENT

ADVERTISEMENT

Report Error

Spaces

Pro

Get Certified

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples

XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate

FORUM | ABOUT