C Pointers

‹ Previous Next ›

Creating Pointers

You learned from the previous chapter, that we can get the **memory address** of a variable with the reference operator `&`:

Example

```
int myAge = 43; // an int variable

printf("%d", myAge);  // Outputs the value of myAge (43)
printf("%p", &myAge); // Outputs the memory address of myAge (0x7ffe5367e044)
```

Try it Yourself »

In the example above, `&myAge` is also known as a **pointer**.

A **pointer** is a variable that stores the memory address of another variable as its value.

A **pointer variable points to a data type** (like `int`) of the same type, and is created with the `*` operator. The address of the variable you are working with is assigned to the pointer:

Example

```
int myAge = 43;     // An int variable
int* ptr = &myAge;  // A pointer variable, with the name ptr, that stores the address of myAge

// Output the value of myAge (43)
printf("%d\n", myAge);

// Output the memory address of myAge (0x7ffe5367e044)
printf("%p\n", &myAge);

// Output the memory address of myAge with the pointer (0x7ffe5367e044)
printf("%p\n", ptr);
```

Try it Yourself »

Example explained

Create a pointer variable with the name `ptr`, that **points to** an `int` variable (`myAge`). Note that the type of the pointer has to match the type of the variable you're working with.

Use the `&` operator to store the memory address of the `myAge` variable, and assign it to the pointer.

Now, `ptr` holds the value of `myAge`'s memory address.

Dereference

In the example above, we used the pointer variable to get the memory address of a variable (used together with the `&` **reference** operator).

However, you can also get the value of the variable the pointer points to, by using the `*` operator (the **dereference** operator):

Example

```
int myAge = 43;     // Variable declaration
int* ptr = &myAge;  // Pointer declaration

// Reference: Output the memory address of myAge with the pointer
(0x7ffe5367e044)
printf("%p\n", ptr);

// Dereference: Output the value of myAge with the pointer (43)
printf("%d\n", *ptr);
```

Try it Yourself »

Note that the `*` sign can be confusing here, as it does two different things in our code:

- When used in declaration (`int* ptr`), it creates a **pointer variable**.
- When not used in declaration, it act as a **dereference operator**.

Why Should I Learn About Pointers?

Pointers are one of the things that make C stand out from other programming languages, like Python and Java.

This chapter was just a short introduction to Pointers. They are important in C, because they give you the ability to manipulate the data in the computer's memory - this can reduce the code and improve the performance. **However**, pointers must be handled with care, since it is possible to damage data stored in other memory addresses.

Good To Know: There are three ways to declare pointer variables, but the first way is mostly used:

```
int* myNum; // Most used
int *myNum;
int * myNum;
```

C Exercises

Test Yourself With Exercises

Exercise:

Create a pointer variable called **ptr**, that points to the `int` variable myAge:

```
int myAge = 43;
          = &myAge;
```

Submit Answer »

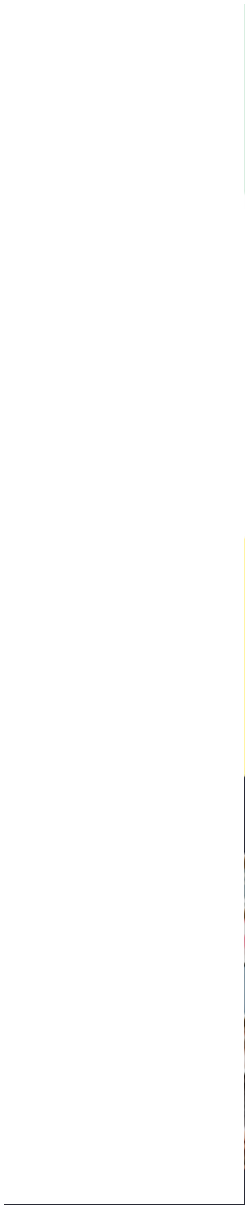
Start the Exercise

‹ Previous

Next ›

CODE GAME

Play Game



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