For everything about pointers in C / C++, please refer to this document

Albert Hsu

December 11, 2014

Abstract

Everything you need to know about pointers in C and C++ in one location.

1 Basic (Where do you live?)

To declare a pointer type use the * right before declaring a variable name. Think of a pointer as a address of a house

char *address

This would declare a char pointer named address. To see who live inside the address, put a * in front of the pointer.

*address

This will tell me what the content of the pointer. AKA who lives there.

Try not to make a habit of putting the * after type but instead before the variable name so that you don't make this mistake

char* address, NotaAddress

This would declare a char pointer named address and a char variable named NotaAddress. If you want to declare two multiple pointers, do this:

char *address1, *address2

This would correctly create two char pointer.

Using the & symbol to show someone address. We can use this to assign address to an already existing location. Or we can have two different address pointing to the same house

address = &AlbertsHouse or address2 = address

Both address will give me the location of AlbertsHouse.

2 Pointers and Arrays (Street Name

Arrays are basically const pointers. Think of arrays are street names. The street name will tell you where the first house will be and you can find other houses on the street by walking. Similarly the array points to the first element of the array and each subsequent element is just the next address.

MagicalLane[3] is basically the same as MagicalLane+3

This will access the third house on magical lane street.

The same properties that applies to pointers also applies to array element since they are basically the same except they're const. For example

int MagicalLane[3] = 0,1,2	
*MagicalLane = 5	This will set MagicalLane[0] = 5
*(MagicalLane+2)	This will return 2

Array pointers are const but the content is not. This is why you change the content but you can not make array point to somewhere else. This is why arrays cannot change size during runtime without using heap. To use heap, use "new" function for c++ and malloc for c. Just remember to free them up so you don't have memory leaks.

3 Math with pointers

The rule of thumb when it comes to doing math with pointers is to use parentheses around the pointer.
The ++ could be replaced with any other math operator.

Command	Memory Address	Memory Content
p	Yep	Nope
*p	Nope	yep
*p++	Incremented after value is read	Unchanged
*(p++)	Incremented after value is read	Unchanged
(*p)++	Unchanged	Incremented after value is read
*++p	Incremented after value is read	Unchanged
*(++p)	Incremented after value is read	Unchanged
++*p	Unchanged	Incremented after value is read
++(*p)	Unchanged	Incremented after value is read

4 Ugly ** double pointer

Double pointers are mainly used in C for array of cstring. It's easier to explain it with examples so:

On the screen	What It is	Seen by compiler
array +1	An Address	A pointer
*(array+1)	Content of address	A string
((array+1))	Content of Cstring	A character
**(array+1)	Same as above	Same as above
*(array+1)[0]	First character of the string	A character

Please note that it double pointer is not exclusively for cstrings. It could just be a pointer pointing to a pointer that points to anything.

5 Function pointers (Factory address)

A factory is a lot like functions. They take in materials and people (parameters) and produce things (function return) that people want. A function pointer is like an address pointing to that factory. To declare a function pointers do something like this:

product (*FactoryAddress)(workers, diamonds

where product can be any return types and (workers and diamonds can be any data types) To avoid confusion, you must put parenthesis around FactoryAddress to show that it's a function pointer.

Some real function pointer examples:

```
int (*adder) (int , int)
void (*sayhello) ()
char* (GiveMeAPointer) (char*)
```

All of the above are just pointers not pointing to anything function yet. An empty address. To point to a function use the = operator and assign it to a function name. No need for * or & symbol. Example:

```
void (*FactoryAddress) () = AlbertsFactory
FactoryAddress = AlbertsSecondFactory
```

Both of these are fine.

5.1 Using Function pointers as parameter

To declare a function that takes in function pointers:

int doMath(int (*mathfunction)(int, int), int a, int b)

Note that doMath is not a function pointer and its first parameter is a function pointer.

Let say I have two other function call add and divide. I can pass in those function as function pointer when calling doMath function by:

result = doMath(add, 10, 5)
or
result = doMath(divide, 10, 5)

You can use function name where it is asking for function pointers. You can also declare a function pointer, set it to the function you want, and then pass it in. Both works.