Part 9: Pointers

Dr. Andi Toce
Lecture Notes for MAC 101 (Introduction to Computer Science)

Last updated / viewed: April 29, 15

Table of Contents

1. WHAT ARE POINTERS? 2

2. ARRAYS AND POINTER ARITHMETIC 4

1. What are Pointers?

Pointers are a powerful feature of C++. They allow the user to pass-by-reference which in turn allows manipulation of more complex, dynamic data structures. Pointers will be extensively used in later more advanced topics.

Pointer declaration: ***data_type * variable_identifier;***

Below is the first simple pointer example:

PointerExample1.cpp

```
#include <iostream>
using namespace std;

int main(){

        int x;
        int *xPointer; // Pointer to integer x

        x = 5;
        xPointer = &x; // Give xPointer the address of x

        cout << "The address of x is: " << &x << endl;
        cout << "The value of xPointer is: " << xPointer << endl << endl;
        cout << "The address of xPointer is: " << &xPointer << endl << endl;
        cout << "The value of x is: " << x << endl;
        cout << "The value of *xPointer is: " << *xPointer << endl;

        return 0;
}
```

Output

```
The address of x is: 0x22fe0c
The value of xPointer is: 0x22fe0c

The address of xPointer is:
0x22fe00

The value of x is: 5
The value of *xPointer is: 5
```

Variable name	Value	Address
x	5	0x22fe0c
.	.	.
.	.	.
.	.	.
.	.	.
xPointer	0x22fe0c	0x22fe00

Note: ***xPointer** has the value **5.**

When xPointer points to x, this statement....	....has the same effect as this statement
*xPointer = 10;	x = 10;
*xPointer = xPointer +5;	x = x + 5;
cout << *xPointer;	cout << x;
cin << *xPointer;	cin << x;

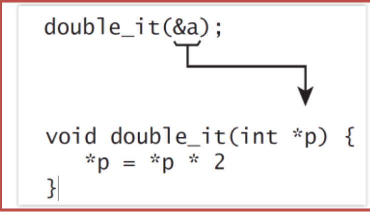
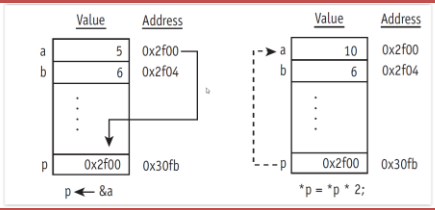
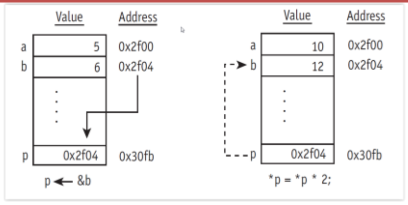
Another example using pointers:

PointerExample2.cpp	Output

```
#include <iostream>
using namespace std;

void double_it(int *p);

int main() {

        int a = 5, b = 6;

        cout << "Val. of a before doubling: " << a << endl;
        cout << "Val. of b before doubling: " << b << endl;

        double_it(&a); // Pass address of a.
        double_it(&b); // Pass address of b.

        cout << "Val. of a after doubling: " << a << endl;
        cout << "Val. of b after doubling: " << b << endl;

        return 0;
}

void double_it(int *p) {
        *p = *p * 2;
}
```

```
Val. of a before doubling: 5
Val. of b before doubling: 6
Val. of a after doubling: 10
Val. of b after doubling: 12
```

Function call	Doubling a	Doubling b

 `double_it(&a);`
 `void double_it(int *p) {`
 `*p = *p * 2`
 `}`
	 Value
	Address
	a
	b
	5
	6
	0x2f00
	0x2f04
	p
	0x2f00
	0x30fb
	p ← &a
	10
	6
	*p = *p * 2;
		 Value
		Address
		a
		b
		5
		6
		0x2f00
		0x2f04
		p
		0x2f04
		0x30fb
		p ← &b
		10
		12
		*p = *p * 2;

Try now: Write a C++ program that swaps the values of two integer variables a and b. Use a function *swap1(*p1, *p2).* Print a and b before and after swapping. Hint: Use a temp variable.

2. Arrays and Pointer Arithmetic

Arrays store values in consecutive memory locations. Pointer can be very useful when processing arrays. The example below illustrates some of the array properties and the use of pointers.

InitializingArrays.cpp

```cpp
#include <iostream>
using namespace std;

void double_it(int *p);

int main() {

        const int arraySize = 5;
        int myArray[arraySize] = {2, 4, 7, 9, 1}; // Initialize array
        int *p = myArray; // Pointer p holds the address in myArray

        // Note that int *p = myArray;
        // is equivalent to: int *p = &myArray[0];

        // First we print values and addresses of myArray

        cout << "Pos \t Value \t Address" << endl << endl;

        for (int i=0; i< arraySize; i++){
        cout << i << "\t";
        cout << myArray[i] << "\t";
        cout << &myArray[i] << "\t" << endl;
        }
        cout << endl;

        // Printing the value of p

        cout << "p is pointing at address: " << p << endl;
        cout << "The value of *p is: " << *p << endl << endl;

        cout << "Now moving pointer p to the next memory location." << endl;
        p = p+1;

        cout << "p is pointing at address: " << p << endl;
        cout << "The value of *p is: " << *p << endl << endl;

        return 0;
}
```

Output

```
Pos      Value     Address

0        2        0x22fde0
1        4        0x22fde4
2        7        0x22fde8
3        9        0x22fdec
4        1        0x22fdf0

p is pointing at address: 0x22fde0
The value of *p is: 2

Now moving pointer p to the next
memory location.
p is pointing at address: 0x22fde4
The value of *p is: 4
```

Try now: Change myArray from an array of integers to a char array. Run the program, compare any two consecutive addresses. What do you notice?

One more example: Arrays of strings. Recall the card shuffling simulator. Here is an analysis of one of the string arrays from this simulator.

FunWithPointers.cpp

```cpp
#include <iostream>
using namespace std;


char *suits[4] = {"hearts", "diamonds", "spades", "clubs"};

int main() {

        cout << "Printing the array content." << endl;
        for (int i=0; i<4; i++){
                cout << suits[i] << endl;
        }
        cout << endl;

        cout << "Printing the beginning address of each string." << endl;
        for (int i=0; i<4; i++){
                cout << &suits[i] << endl;
        }
        cout << endl;

        cout << "Printing substrings of the first string." << endl;
        for (int i=0; i<6; i++){
                cout << &suits[0][i] << endl;
        }
        cout << endl;

        cout << "Printing the value where pointers are." << endl;
        for (int i=0; i<4; i++){
                cout << *suits[i] << endl;
        }
        cout << endl;

    return 0;
}
```

Output

```
Printing the array content.
hearts
diamonds
spades
clubs

Printing the beginning address of
each string.
0x46f020
0x46f028
0x46f030
0x46f038

Printing substrings of the first
string.
hearts
earts
arts
rts
ts
s

Printing the value where pointers
are.
h
d
s
c
```