



Without practice, your knowledge is poison.

Wollo University, Kombolcha

Institute of Technology

College of Informatics

Department of Information System

By Daniel G.

Mar/2021 G.C

A cluster of hexagonal icons in various shades of blue and cyan. The icons include a lightbulb, a thumbs up, a network node, a smartphone, a magnifying glass, a gear, and a speech bubble.

INSY2031

Chapter Three

Pointer



Contents

- ◇ Basic concept of pointers
- ◇ Pointer variables and declaration
- ◇ Pointer expression, operation and arithmetic.
- ◇ Strings and pointers
- ◇ Relationship between pointers and arrays
- ◇ Revisiting function calling by reference (using pointers)





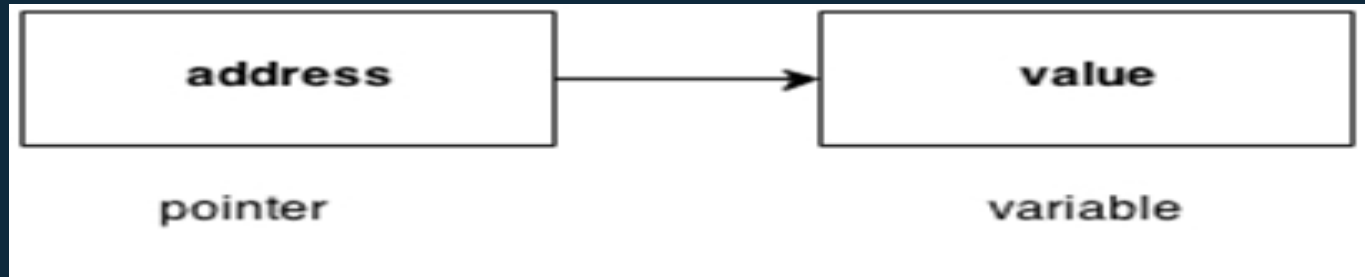
Basic concept of pointers

1

A pointer refers to a variable that holds the address of another variable. It have a data type. For example, a pointer of type integer can hold the address of a variable of type integer.

Basic concept of pointers

The pointer in C++ language is a variable, it is also known as locator or indicator that points to an address of a value.



Advantage of pointer

1

- ❑ Pointer reduces the code and improves the performance, and used with arrays, structures and functions.
- ❑ We can return multiple values from function using pointer.
- ❑ It makes you able to access any memory location in the computer's memory(dynamic memory allocation).

Pointer Variables declaration.

`int* x`
`char* y;`
example.

`int age = 32;`
`int* p = &age;`

& is called **address-of** operator.

Accessing an object's value from its address is called **dereferencing**.

now `*p` is another way of referring to `age`.


1



Example

1

```
char ch = 'Q';  
char* p = &ch; // p holds the address of ch  
cout << *p;     // outputs the character 'Q'  
ch = 'Z';       // ch now holds 'Z'  
cout << *p;     // outputs the character 'Z'  
*p = 'X';       // ch now holds 'X'  
cout << ch;    // outputs the character 'X'
```

A void pointer is a general-purpose pointer that can hold the address of any data type, but it is not associated with any data type.

1

```
int *ptr; // integer pointer declaration  
float a=10.2; // floating variable initialization  
ptr= &a; // This statement throws an error.
```



take care.

```
int* x, y, z;
```

1

```
// same as int*, int y, int z;
```



1

```
#include <iostream>
using namespace std;
int main()
{
    int a=20,b=10,* p1=&a,* p2=&b;
    cout<<"Before swap: * p1="<<* p1<<"
        * p2="<<* p2<<endl;
    * p1=* p1+* p2;
    * p2=* p1-* p2;
    * p1=* p1-* p2;
    cout<<"After swap: * p1="<<* p1<<" * p2="<<* p2<<endl;
    return 0;
}
```



Pointers and Arrays

1

There is an interesting connection between arrays and pointers, which C++ inherited from the C programming language the name of an array is equivalent to a pointer to the array's initial element and vice versa.

Example.

1

```
int c[ ] = {1, 2, 3};  
int* p = c;           // p points to c[0]  
int* q = &c[0];       // q also points to c[0]
```

```
cout << c[2] << p[2] << q[2];
```

```
// outputs 333
```



1

Pointer expression, operation and arithmetic.

as we know pointer is an address which is a numeric value. there for we can perform arithmetic operations.

there are four operators that can be used on pointers : $++$, $--$, $+$, $-$;



continued..

let us consider that `ptr` is an integer pointer which points to the address `1000`. lets perform this operations `ptr++`.

1

this operation will move the pointer to next memory location without impacting actual value at the memory location which is `1004`. because it is an integer.

if `ptr` points to a character whose address is `1000`, then the above operation will point to the location `1001` because next character is available at `1001`.



Pointer Comparison

Pointer may be compared by using relational operators such as `==`, `<` and `>`.

1

if `p1` and `p2` point to variables that are related to each other such as elements of the same array, then `p1` and `p2` can be meaningfully compared.

1

```
const int MAX = 3;
```

```
int main(){  
    int var[MAX] = {10, 100, 200};  
    int *ptr;
```

```
    // let us have array address in pointer.  
    ptr = var;
```

```
    for (int i = 0; i < MAX; i++) {  
        cout << "Address of var[" << i << "] = ";  
        cout << ptr << endl;
```

```
        cout << "Value of var[" << i << "] = ";  
        cout << *ptr << endl;
```

```
        // point to the next location  
        ptr++;  
    }  
}
```

Output

Address of var[0] = 0xbfa088b0

Value of var[0] = 10

Address of var[1] = 0xbfa088b4

Value of var[1] = 100

Address of var[2] = 0xbfa088b8

Value of var[2] = 200



Strings and pointers

1

When the pointers are used for character array or strings, then it is called as string pointers. It works similar to any other array pointers. When we increment or decrement string pointers, it increments or decrements the address by 1 byte.

Strings and pointers

```
int main(){
    string str = "helow";
    char x[5] = "helow";

    string*ptr;


    // let us have array address in pointer.
    ptr = str;
}
```

1

Strings and pointers

```
int main(){  
    string str = "helow";  
    string*ptr;  
  
    // let us have array address in pointer.  
    ptr = str;  
}
```

1



Revisiting function calling by reference (using pointers)

1

There is another way of passing arguments to a function where the actual values of arguments are not passed. Instead, the reference to values is passed.



END
OF
CHAPTER





Thanks!

Any questions?

Feel free!

