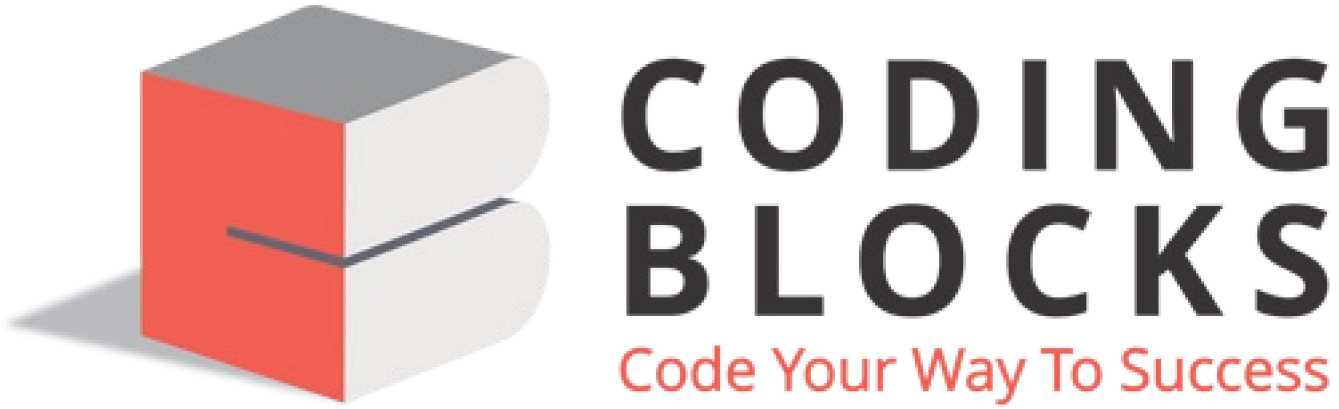

C++ Launchpad

Pointers

What are pointers?

- Pointers are one of the most powerful and confusing aspects of the C/C++ language.
- A pointer is a variable that holds the address of another variable.
- To declare a pointer, we use an asterisk between the data type and the variable name.

```
int *pnPtr; // a pointer to an integer value d
double *pdPtr; // a pointer to a double value
int* pnPtr2; // also valid syntax
int * pnPtr3; // also valid syntax
```

Address of Operator (&)

Since pointers only hold addresses, when we assign a value to a pointer, the value has to be an address. To get the address of a variable, we can use **the address-of operator (&)**

```
int p = 5; int * q = &p; // assign address of p in q
```

Dereference Operator [*]

An interesting property of pointers is that they can be used to access the variable they point to directly. This is done by preceding the pointer name with the dereference operator. The operator itself can be read as **"value pointed to by”**

Therefore the value pointed by q in previous example can be accessed as `int r = *q;`

Null Pointer

Sometimes it is useful to make our pointers point to nothing. This is called a null pointer. We assign a pointer a null value by setting it to address 0.

`double *p = 0;`

Arithmetic Operators & Pointers

- Addition, Multiplication, Division of two addresses doesn’t make

- Addition of an address by a constant integer value i.e. ptr +5 means address of cell which is 5 * sizeof(*ptr) away from ptr.
- Similar for subtraction
- Again Multiplying/Dividing an address with some constant value doesn’t make any sense
- Subtracting two address of same type would give you number of elements between them

Arrays and Pointers

- Pointers and arrays are intricately linked in the C language.

- An Array is actually a pointer that points to the first element of the array! Because the array variable is a pointer, you can dereference it, which returns array element 0.

- a[i] is same as `*(a + i)`

- Its possible to pass part of an array to function.
  Monday

Difference – Arrays & Pointers

The sizeof operator

- `sizeof(array)` returns the amount of memory used by all elements in array

- `sizeof(pointer)` only returns the amount of memory used by the pointer variable itself

The & operator

- `&array` is an alias for &array[0] and returns the address of the first element in array
- `&pointer` returns the address of pointer

String literal initialization of a character array

- char `array[] = “abc”` sets the first four elements in array to ‘a’, ‘b’, ‘c’, and ‘\0′
- char `*pointer = “abc”` sets pointer to the address of the “abc” string (which may be stored in read-only memory and thus unchangeable)

Assignment/Re-assigment

- Pointer variable can be assigned a value whereas array variable cannot be.

```
int a[10];
int *p;
p=a; /*legal*/
a=p; /*illegal*/
```

Arithmetic

- Arithmetic on pointer variable is allowed but array can’t be incremented/decremented.

```
p++; /*Legal*/
a++; /*illegal*/
```