C++ Notes: Pointers

Pointers in C/C++

Pointers are variables that contain memory addresses (see Addresses, Pointers, and References). They are an essential data type in C and C++, and are used for the following:

- Array variables are pointers and pointers can be used as alternative to subscripts.
- Dynamic memory allocation/deallocation returns a pointer to the allocated memory.
- For efficiency by passing the address of an object instead of copying the object to a function. Reference parameters typically used.
- For function parameters that will be changed by the function (*out* or *inout* parameters). Reference parameters typically used.

Declaring a pointer

Pointers are declared to point to a particular datatype. A "*" is written after the type to indicate that this is a pointer to that type. For example, a pointer to an int would be declared like this.

```
int* ip;  // declares ip to be a pointer to an int.
```

You will also see variations in the spacing such as

```
int *ip;

int * ip;
```

NULL

NULL is the pointer to nothing, and should be used as the initial value for pointers because using NULL will cause an error in most systems.

Pointer operators: * (dereference), & (address of), and arithmetic

Two operators are used when dealing with memory addresses:

- Unary & gives the *address of* (pointer to) something.
- Unary * takes the contents of a pointer (*dereferences* a pointer).
- Integers can be added to pointers, in which case they are first multiplied by the element size
- Two pointers may be subtracted to yield an integer.

Example

```
char* cp;     // declare pointer to char

char  c, d;   // declare char variables


cp  = &c;     // puts address of c into cp
```

```
c   = 'x';     // assigns 'x' to c
*cp = 'x';     // also assigns 'x' to c
d   = *cp;     // copies c's value to d
cp  = &d;      // puts address of d into cp
*cp = c;       // puts value of c into d
```