Arrays, pointers, iterators

- In C an array is a contiguous memory, a pointer can be treated as an array (contiguous) and can be one.
  - `int a[100]; // is created at compile time`
  - `int * a = new int[5000]; // at run time`
- Calling new means
  - Created on heap, can last past method/function
  - Can allocate at run time
  - Eventually you'll run out of memory
    - No garbage collection in C++
    - Should eventually call delete, YAHOO! Rolling reboot?

What is a hashtable?

- An array of pointers to nodes in linked lists
  - What's the same? Different? In code below

```
Node * table[100];
Node ** table = new Node * table[1000];
table[0] = new Node(…);
table[1] = NULL;
table[2] = 0;
```

- Create an array of C-style strings

```
char ** list = new char *[100];
// what is list? What is list[0]?
```


What is a C-style string?

- array of char terminated by sentinel '\0' char
  - sentinel char facilitates string functions
  - '\0' is nul char, unfortunate terminology
  - how big an array is needed for string "hello"?
- a string is a pointer to the first character just as an array is a pointer to the first element
  - `char * s = new char[6];`
  - what is the value of s? of s[0]?
- char * string functions in <string.h>, <cstring>

C style strings/string functions

- strlen is the # of characters in a string
  - same as # elements in char array?

```
int strlen(char * s)
// pre: '\0' terminated
// post: returns # chars
{
    int count=0;
    while (*s++) count++;
    return count;
}
```

- Are these less cryptic?

```
while (s[count]) count++;
// OR, is this right?
char * t = s;
while (*t++);
return t-s;
```

- what's "wrong" with this code?

```
int countQs(char * s)
// pre: '\0' terminated
// post: returns # q's
{
  int count=0;
  for(k=0;k < strlen(s);k++)
    if (s[k]=='q') count++;
  return count;
}
```

- how many chars examined for 10 character string?
- solution?