Arrays/strings: pointers and memory allocation

- Why not rely solely on string and vector classes?
  - How are string and vector implemented?
  - Lower level access can be more efficient (but be leery of claims that C-style arrays/strings required for efficiency)
  - Real understanding comes when more levels of abstraction are understood
- String and vector classes insulate programmers from inadvertent attempts to access memory that’s not accessible
  - What is the value of a pointer?
  - What is a segmentation violation?

Contiguous chunks of memory

- In C++, allocate using array form of new
  - New [] returns a pointer to a block of memory
  - How big? Where?
- Size of chunk can be set at runtime, not the case with int a[100];
  - Cin >> howBig;
  - Int a[howBig];
- Delete [] a; // Storage returned

C-style contiguous chunks of memory

- In C, malloc is used to allocate memory
  - Int * a = (int *) malloc(100 * sizeof(int));
  - Double * d = (double *) malloc(200 * sizeof(double));
- Malloc must be cast, is NOT type-safe (returns void *)
  - Void * is ‘generic’ type, can be cast to any pointer type
- Free(); // return storage

Address calculations: sizeof(...)

- X is a pointer, what is x+33?
  - A pointer, but where?
  - What does calculation depend on?
- Result of adding an int to a pointer depends on size of object pointed to
- Result of subtracting two pointers is an int:
  - (d - 3) - d == _______
More pointer arithmetic

- Address one past the end of an array is ok for pointer comparison only
- What about *(begin+44)?
- What does begin++ mean?
- How are pointers compared using < and using ==?
- What is value of end - begin?

```cpp
char * a = new int[44];
char * begin = a;
char * end = a + 44;
while (begin < end)
{
    *begin = 'z';
    begin++; // *begin++ = 'z'
}
```

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 <cstring> (or <string.h>)

C style strings/string functions

- Strlen is the # of characters in a string
  - Same as # elements in char array?
    ```cpp
    int strlen(char * s)
    { // pre: '\0' terminated
        int count=0;
        while (*s++) count++;
        return count;
    }
    ```
    - Are these less cryptic?
      ```cpp
      while [*[count]] count++; // Ok, is this right?
      char * t = s;
      while [*[t]]
      return t-s;
      ```

More string functions (from <cstring>)

- Strcpy copies strings
  - Who supplies storage?
    ```cpp
    int countQs(char * s)
    { // pre: '\0' terminated
        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?
- What’s “wrong” with this code?
- What’s wrong with s = t?
  ```cpp
  char s[5];
  char t[6];
  char * h = "hello";
  strcpy(s,h); // trouble!
  strcpy(t,h); // ok
  char * strcpy(char* t,char* s)
  { // pre: target has space
      int k=0;
      while (*t = *s) k++;
      return t;
  }
  ```
- What about relational operators <, ==, etc.?
- Can’t overload operators for pointers, no overloaded operators in C
- Strcmp (also strncmp)
  ```cpp
  if (strcmp(s,t)==0) // equal
  if (strcmp(s,t) < 0) // less
  if (strcmp(s,t) > 0) // ????
  ```
- Strncpy copies n chars (safer?)
Arrays and pointers

- These definitions are related, but not the same
  int a[100];
  int * ap = new int[10];
- both a and ap represent ‘arrays’, but ap is an lvalue

- arrays converted to pointers for function calls:
  char s[] = “hello”;
  // prototype: int strlen(char * sp);
  cout << strlen(s) << endl;
- multidimensional arrays and arrays of arrays
  int a[20][5];
  int * b[10]; for(k=0; k < 10; k++) b[k] = new int[30];

Microsoft question

- Write atoi, write itoa, which is harder?
- Questions? Issues? Problems?
  int atoi(const char * sp);
  char * itoa(int num);
- Difference between const char * p and char * const p
  ➤ one is a pointer to a constant character
  ➤ one is a constant pointer to a character