*Notes 11/1*

C strings

S: 0 1 2 3 4 5 6 7 8 9 10 … 99  
 J e l l o \0 . . . . . .

T: 0 1 2 3 4 5 6 7 8  
 J e l l o \0 . . .

#include <cstring>  
using namespace std;  
  
char s[100] = “”;  
char t[9] = “Hello”;  
t[0] = ‘J’;  
cout << strlen(t); //writes 5  
  
cout << t;  
cin.getline(s, 100);  
  
s = t; //Error! Cannot assign arrays!  
strcpy(s,t); // *strcpy(destination, source)*;

s =+ “!!!”; //Error! Can’t do this with c-strings.

Strcat(s, “!!!”); // *strcat(string to add on to, string being added)* adds the second string onto the first  
  
if (t < s) // compiles, but doesn’t compare the text of the strings.

Comparison: strcmp(s, t)  
 negative: if s comes before t  
 0: if s == t  
 positive: if t comes before s

Pointers

A pointer is basically just an indication of where something is; it’s another way to implement passing by reference (mostly a C thing). You can also use them to traverse arrays, manipulate dynamic storage, and represent relationships in data structures.

*Example:*

Void polarToCartesian(double rho, double theta, double& xx, double& yy);  
   
int main()  
{  
 double r;  
 double angle;  
 … //get values for r and angle  
 double x;  
 double y;  
 polarToCartesian(r, angle, x, y);  
}  
  
void polarToCartesian(double rho, double theta, double& xx, double& yy)  
{  
 xx = rho \* cos(theta);  
 yy = rho \* sin(theta);  
}  
  
The double& declaration means it is a *reference-to-double*. It’s another name for an already existing double – if it gets changed in the function it gets changed in the main routine.

To declare a pointer-to-double you use the syntax *double\* p.* Using this, instead of passing actual values to functions you can just pass an “arrow” to a value from another function. Put an “&” in front of the value when calling the function to use a pointer to the object rather than simply copying the value of it, and put a “\*” in front of the value whenever you call in in the new function. This means any operations done by the function to that variable will change the value of the original.

*Example of the same program, using pointers:*

Void polarToCartesian(double rho, double theta, double& xx, double& yy);  
   
int main()  
{  
 double r;  
 double angle;  
 … //get values for r and angle  
 double x;  
 double y;  
 polarToCartesian(r, angle, &x, &y);  
}  
  
void polarToCartesian(double rho, double theta, double\* xx, double\* yy)  
{  
 &xx = rho \* cos(theta);  
 &yy = rho \* sin(theta);  
}