Passing by reference.

* Template <class T>
* Void swap(T x, Ty)
  + T temp = x;
  + X = y;
  + Y = temp;
* Main()
  + Card c1(2, diamonds), c2;
  + Swap(c1, c2);
  + Cout << "card 1: " << c1 << "card 2" << c2;
* This Program will not run correctly. The variables need to be passed by pointerin order to make the variables actually change.
* Swap(&c1, &c2) The & needs to be in the parameters. Void swap(T\* x, T\* y) in order to pass the pointers. Then the variables need to be dereferenced. T temp = \*x; \*x = \*y; \*y = temp;.

Ch 9 List

* #include <list>
* Main()
  + List<int> l;
  + l.push\_back(44);
  + l.push\_back(33);
  + l.push\_back(11);
  + l.push\_back(22);
    - 44 -> 33 -> 11 -> 22
    - List may not always be next to each other in memory. They are linked through pointers. One variable points to the next.
  + For(list<int>::iterator I = l.begin(); I != l.end(); I++)
    - Cout << \*I;
  + l.popback(); // removes 22
  + l.pop\_front(); // removes 44
  + List <int> m = l; //default constructor for m, overload = to copy l to m
  + List<int> m(l); //copy constructor
    - Note: elements of list are dynamically allocated from heap.
    - List does not have operator[];

Lab 5 – List

* Elements are linked together using pointers/addresses
* |'0/' |44| |-> <-| |33| |-><-| |11| |-><- | |22| '0/'|
* Doubly linked\_list
* Template <class T>
* Class Link
  + Link(const T & x): value(x), next\_link(0), prev\_link(0) {}
  + T value;
  + Link<T> \* next\_link;
  + Link<T> \* prev\_link;
  + Friend class List<T>;
* Template<class T>
* Class List
  + Link<T> \* first\_link;
  + Link<T> \* last\_link;
  + Int size;
  + Public:
    - List();
    - List(const List<T> &l); copy constructo
    - Push\_back(T);
    - Pop\_back(); // have a check for size == 0;
      * Last\_link = last\_link -> prev\_link;
    - Push\_front(T;
    - Pop\_front();
    - Insert(iterator,T);
      * //l.insert(I, 5);
* Template<class T>
* Class List\_Iterator
  + Link<T> \* current\_link;
  + Public:
    - T & operator\*(); //dereferencing operator
      * Return current\_link->value;
      * List\_iterator & operator++(); //pre increment
      * List\_iterator operator++(int); //post increment