**How to change el\_t to be an object**

Previously with typedef we were letting el\_t stand for a built-in type such as char and int.

But for graph algorithms, we want to be able to store more than a simple type in each node:

* MST/Shortest Path – each node element consists of the **vertex name and the weight** on the incoming edge.
* Hash table – each node element consists of the **key and data items**.

How do we allow this without making many changes?

One way is to create a **class called el\_t**. Define it in **elem.h** and **elem.cpp.**

Then, **each node will contain an el\_t object**.

The list class will **include “elem.h”.**

**For the client of the list class: (this could be the graph class/hash class)**

L.addRear(X); // X has to be an el\_t object that has been created by the client

L.search(X); // X has to be an el\_t object

An el\_t object can be created using an **el\_t constructor** defined in the elem class.

**\*Inter \* If your graph class or the hash table class needs to access the parts of an el\_t object, what do you need to do? Note that the data members of el\_t are in private.**

**Within the list class:**

A node would look like:

Elem Next

pointer

El\_t object

So, everytime you say P->Elem, you are referring to an el\_t object that may have multiple parts in it.

e.g. P->Elem = E; // E should be an el\_t object

e.g. if (P->Elem == Key) // two el\_t objects are being compared

e.g. cout << P->Elem; // el\_t object is being displayed

Thus, we have to decide if we need to overload operators in elem.cpp to make the operations work with your el\_t objects.

* llist.cpp does **cout** and **=** with this el\_t object.
  + No need to define = for el\_t object. It should work if el\_t has no pointers.
  + Cout needs to be defined for el\_t object.
* slist.cpp does a search that uses **==.**
  + What should it compare to do the search? The whole object? A part of it?
* Slist.cpp uses != for el\_t objects (e.g. operator==)

**So, in the elem class, overload/define cout and == and != to work with your el\_t object.**