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)

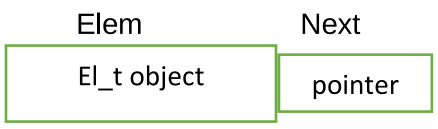
```
L.addRear(X);   // X has to be an el_t object that has been created by the client
L.search(X);    // X has to be an el_t object
```

An el_t object can be created using an **el_t constructor** defined in the elem class.

***Question* 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 in elem.h? Note that the data members of el_t are in private.**

Within the llist class:

A node would look like:



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 overloaded/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?

So, in the elem class, overload/define cout and == to work with your el_t object.