Unions and Enumerations

12-3-2003

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?

Enumerations

- C has a construct for representing small fixed sets of different types. It uses the keyword enum.
- Unlike using #defines, the enum actually declares a type that is associated with the names in the enumeration.
- Having a type makes the code safer in most situations because it moves logic errors into the realm of syntax errors.

Syntax of enum

- Like with a struct, an enum defines a type and we can name that type with a tag or a typedef.

```
enum Pstatus {ACTIVE,INJURED,SUSPENDED};
enum Pstatus stat1,stat2;
or
typedef enum {ACTIVE,INJURED,SUSPENDED}
     Pstatus;
Pstatus stat1,stat2;
stat1=active;
```

Motivation for Unions

- Sometimes you have a desire to make structures that can be one of several different types. For example, in the graphics code you have lights, triangles, and spheres. You might want to have a single array that stores everything. To do this you could put data in your struct for all of them and only use one, but that wastes space.

Unions

- A union is another user defined type that works like a structure, but it only gets enough memory for the largest field in it and the fields share space so you can only safely use one.
- Typically you put a union in a structure and have a variable in the structure that tells you what "flavor" the union is in that case.

Union Syntax

- Unions are typically only found in structs because we need a type as well. For this reason we rarely name them.

```
enum ObjType {LIGHT,TRIANGLE,SPHERE};
struct DrawObj {
     enum ObjType type;
     union {
          struct Light light;
          struct Triangle tri;
          struct Sphere sphere;
     } data;
};
```

Code

- Now lets write a little code that demonstrates this. A sports analogy works well here if we write code for stats is a sport where different players keep different stats.

Minute Essay

- Write a structure using a union that can represent different methods of getting places (car, bike, etc.).
- Quiz #6 will be given next class.