Forward declaration

Samuel Li

Idea

- Imagine you're a compiler looking at a program.
- You need to see a **variable declaration** before you operate on it
  - int a = 0;
  - a++;
- You need to see a **struct declaration** before you initiate variables of that type
  - struct student { char* name; int id; };
  - struct student Tom, Jerry;
- You need to see a function declaration before you call that function
  - void foo(int a) {a++;}
  - int i = 0; foo(i);

Logistics

- This slides: https://goo.gl/G50tNW
- Test program: http://ix.cs.uoregon.edu/~samuelli/files/lab5.tar.gz
  - wget http://ix.cs.uoregon.edu/~samuelli/files/lab5.tar.gz
  - tar zxvf lab5.tar.gz

Look at a program with graph information

```
struct Edge
{
    struct Vertex vertices[2];  /* stores 2 vertices */
};

struct Vertex
{
    float  location[2];    /* x, y coordinates of this vertex */
    int    nEdges;         /* number of edges connect to this vertex */
    struct Edge *edges;    /* an array of edges that connect to this vertex */
};
```

Compiler may give you some hints

```
California:lab5 samuel$ gcc graph.c
In file included from graph.c:3:
./graph.h:4:27: error: array has incomplete element type 'struct Vertex'
    struct Vertex vertices[2];  /* pointing to the 2 vertices */
                          ^
./graph.h:4:12: note: forward declaration of 'struct Vertex'
    struct Vertex vertices[2];  /* pointing to the 2 vertices */
           ^
```

- ~~Solution: forward declaration of ‘struct Vertex’ !~~
- ~~How? Simply add a line before line 4~~
  - ~~`struct Vertex;`~~
  - ~~leave the implementation the same place.~~

Look at next slide!

Compiler may give you some hints

```
California:lab5 samuel$ gcc graph.c
In file included from graph.c:3:
./graph.h:4:27: error: array has incomplete element type 'struct Vertex'
    struct Vertex vertices[2];  /* pointing to the 2 vertices */
                          ^
./graph.h:4:12: note: forward declaration of 'struct Vertex'
    struct Vertex vertices[2];  /* pointing to the 2 vertices */
           ^
```

- Solution: put `struct Vertex` definition before `struct Edge`.
- New question: `struct Vertex` also has as a member pointer pointing to `struct Edge`?

Order matters

```
1  struct Edge;
2
3  struct Vertex
4  {
5      float  location[2];     /* x, y coordinates of this vertex */
6      int    nEdges;          /* number of edges connect to this vertex */
7      struct Edge *edges;     /* an array of edges that connect to this vertex */
8                              /* type pointer always have a fixed size: 8 bytes */
9  };
10
11 struct Edge
12 {
13     struct Vertex vertices[2];  /* 2 ends of an edge. */
14 };
```

- Pointers always have 8 bytes, so compilers can still allocate memory, without knowing details about `struct Edge`.
- Older compilers may require a forward declaration of the struct type. (line 1)

Compiler may give you some hints

```
graph.c:27:2: warning: implicit declaration of function 'PrintEdgeInfo' is invalid in
      C99 [-Wimplicit-function-declaration]
        PrintEdgeInfo( &e1 );
        ^
graph.c:64:6: error: conflicting types for 'PrintEdgeInfo'
void PrintEdgeInfo( struct Edge *e )
     ^
graph.c:27:2: note: previous implicit declaration is here
        PrintEdgeInfo( &e1 );
        ^
```

- Warning (line 27): compiler thinks you're doing an "implicit declaration" of function 'PrintEdgeInfo.'
- Error (line 64): later, compiler sees declaration of 'PrintEdgeInfo' again, and throws an error.
- At the same time, compiler thinks you may intend the same function in line 64 and line 27, but not confident. It prints you the hint message at last.

Your task

- Fix the program by adding forward declarations.

- Program to download:
  - http://ix.cs.uoregon.edu/~samuelli/files/lab5.tar.gz
- This slides: https://goo.gl/G50tNW