CS 101: Computer Programming and Utilization

Shivaram Kalyanakrishnan
(Abhiram Ranade's slides, borrowed and edited)
Lecture 20

Today's Lecture

- Arrays of structures
- Structures with member functions

Structures with member functions

```
struct V3{
 double x, y, z;
 double length(){
   return sqrt(x*x +
               y*y + z*z);
 }
};
int main(){
 V3 v={1,2,2};
 cout << v.length()
      << endl;
}
```

- `length` is a member function.
- Member function `f` of a structure `X` must be invoked "on" a structure `s` of type `X` by writing `s.f(arguments)`.
- `s` is called receiver of the call.
- Example: `v.length()`. In this `v` is the receiver.
- The function executes by creating an activation frame as usual.
- References to members in the body of the definition of the function refer to the corresponding members of the receiver.
- Thus when `v.length()` executes, `x, y, z` refer to `v.x, v.y, v.z`.
- Thus the `v.length()` will return $\text{sqrt}(1^2+2^2+2^2) = 3$
- Member functions can modify receiver members. receiver is passed by reference

The complete definition of V3

```
struct V3{
  double x, y, z;
  double length(){
    return sqrt(x*x + y*y +
z*z);
  }
  V3 sum(V3 b){
    V3 v;
    v.x = x+b.x; v.y=y+b.y;
v.z=z+b.z;
    return v;
  }
  V3 scale(double f){
    V3 v;
    v.x = x*f; v.y = y*f; v.z
= z*f;
    return v;
  }
}
```

```
int main(){
 V3 u, a, s;
 double t;
 cin >> u.x >> u.y >> u.z >>
        a.x >> a.y >> a.z >>
t;
 V3 ut = u.scale(t);
 V3 at2by2 = a.scale(t*t/2);
 s = ut.sum(at2by2);
 cout << s.length() << endl;
// green statements equivalent
to red:
 cout <<  u.scale(t).
          sum(a.scale(t*t/2)).
          length() << endl;
}
```