

C++ Structures

Starting to think about objects...

Structure

- A Structure is a container, it can hold a bunch of *things*.
 - These things can be of any type.
- Structures are used to organize related data (variables) in to a nice neat package.

Example - Student Record

- Student Record:
 - Name a string
 - HW Grades an array of 3 doubles
 - Test Grades an array of 2 doubles
 - Final Average a double

Structure Members

- Each *thing* in a structure is called *member*.
- Each *member* has a name, a type and a value.
- Names follow the rules for variable names.
- Types can be any defined type.

C++ Spring 2000 Structures

4

Example Structure Definition

```
struct StudentRecord {  
    char *name;    // student name  
    double hw[3]; // homework grades  
    double test[2]; // test grades  
    double ave;   // final average  
};
```

C++ Spring 2000 Structures

5

Using a struct

- By defining a structure you create a new data type.
- Once a struct is defined, you can create variables of the new type.

```
StudentRecord stu;
```

C++ Spring 2000 Structures

6

Accessing Members

- You can treat the members of a struct just like variables.
- You need to use the *member access operator* '.' (pronounced "dot"):

```
cout << stu.name << endl;
    stu.hw[2] = 82.3;
    stu.ave = total/100;
```

C++ Spring 2000 Structures

7

Structure Assignment

- You can use structures just like variables:

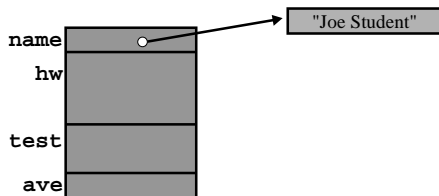
```
StudentRecord s1,s2;
s1.name = "Joe Student";
...
s2 = s1; ← Copies the entire structure
```

C++ Spring 2000 Structures

8

Be Careful

- If a member is a pointer, *copying* means copying the pointer (not what is pointed to).



C++ Spring 2000 Structures

9

Probably not what you want

```
StudentRecord s1,s2;
s1.name = "Joe Student";
...
s2 = s1;
s2.name = "Jane Doe";

// now s1.name and s2.name are both
// "Jane Doe"
```

C++ Spring 2000 Structures

10

Pointers to Structures

- Pointers to structures are used often.
- There is another *member access operator* used with pointers: `->`

```
StudentRecord *sptr;
...
cout << "Name is" << sptr->name;
cout << "Ave is " << sptr->ave;
```

it looks like a "pointer"!

C++ Spring 2000 Structures

11

Sample Function (won't work!)

```
void update_average( StudentRecord stu) {
    double tot=0;

    for (int i=0;i<3;i++)
        tot += stu.hw[i];
    for (int i=0;i<3;i++)
        tot += stu.test[i];
    stu.ave = tot/5;
}
```

C++ Spring 2000 Structures

12

This one works

```
void update_average( StudentRecord *stu)
{
    double tot=0;

    for (int i=0;i<3;i++)
        tot += stu->hw[i];
    for (int i=0;i<3;i++)
        tot += stu->test[i];
    stu->ave = tot/5;
}
```

C++ Spring 2000 Structures

13

Or use a reference parameter

```
void update_average( StudentRecord &stu) {
    double tot=0;

    for (int i=0;i<3;i++)
        tot += stu.hw[i];
    for (int i=0;i<3;i++)
        tot += stu.test[i];
    stu.ave = tot/5;
}
```

C++ Spring 2000 Structures

14

Other stuff you can do with a struct

- You can also associate special functions with a structure (called *member functions*).
- A C++ *class* is very similar to a structure, we will focus on classes.
 - Classes can have (data) members
 - Classes can have member functions.
 - Classes can also *hide* some of the members (functions and data).

C++ Spring 2000 Structures

15

Quick Example

```
struct StudentRecord {
    char *name;           // student name
    double hw[3];        // homework
    grades
    double test[2];      // test grades
    double ave;          // final average

    void print_ave() {
        cout << "Name: " << name << endl;
        cout << "Average: " << ave << endl;
    }
};
```

C++ Spring 2000 Structures

16

Using the member function

```
doubleStudentRecord stu;

... // set values in the structure

stu.print_ave();
```

C++ Spring 2000 Structures

17
