Ch 11. Structured Data
(11.2 to 11.8)

CS 2308
Spring 2013

Jill Seaman

Composite Data Types (C/C++)

- Arrays: ordered sequence of values of the same type
- Structures: named components of various types
  - Used to represent a relationship between values of different types
  - Example: student
    - ID Number
    - Name
    - Age
    - Major
    - Address

the values are related because they belong to the same student

Structures

- Define the student as a struct in C++:

```
struct Student {
   int idNumber;
   string name;
   int age;
   string major;
};
```

  - Defines a new data type
  - The components are called "members" (or "fields").
- To define a variable of type Student:

```
Student csStudent;
```


Initializing, Accessing Structures

- Struct variable can be initialized when it is defined:
  - values must be in order of struct declaration

```
Student student1 = {123456,"John Smith",22, "Math"};
```

- Use dot notation to access members of a struct variable:

```
student1.age = 18;
student2.idNumber = 123456;
cin >> gradStudent.name;
gradStudent.major = "Rocket Science";
```

Structures: operations

- Valid operations over entire structs:
  - assignment: `student1 = student2;`
  - function call: `showStudent(gradStudent);`
- Invalid operations over structs:
  - comparison: `student1 == student2`
  - output: `cout << student1;`
  - input: `cin >> student2;`
  - Must do these member by member

Arrays of Structures

- You can store values of structure types in arrays.

```
Student roster[40];  //holds 40 Student structs
```

- Each student is accessible via the subscript notation.

```
roster[0] = student1;
```

- Members of structure accessible via dot notation

```
cout << roster[0].name << endl;
```


Nested Structures

- You can nest one structure inside another.

```
struct Address {
   string street;
   string city;
   string state;
   int zip;
};
```

```
struct Student {
   int idNumber;
   string name;
   Address homeAddress;
};
```

- Use dot operator multiple times to get into the nested structure:

```
Student student1;
student1.name = "Bob Lambert";
student1.homeAddress.city = "San Angelo";
student1.homeAddress.state = "TX";
```


Structures as function arguments

- Structure variables may be passed as arguments to functions.

```
void showStudent(Student x) {
   cout << x.idNumber << endl;
   cout << x.name << endl;
   cout << x.age << endl;
   cout << x.major << endl;
}
```

- Like regular variables:
  - structure variables are passed by value by default.
  - pass by reference can be used to change the value of a member in the function.