
Structs

Chapter 7
pp. 391 - 401

2/12/07

CS250 Introduction to Computer Science II

1

Arrays and Data Types

- Useful for storing a collection of data elements of the **same** data type (float, int, string).

```
char myName[5]; //All elements chars
float salaries[NUM_EMP]; //All elements floats
char vowels[]={ 'A', 'E', 'I', 'O', 'U' };
```

- What about storing a collection of data elements of **different** data types?

2/12/07

CS250 Introduction to Computer Science II

2

Data with Different Data Types (7.1)

- For example, what if we wanted to keep the following information on a particular employee:
 - employee id
 - SS#
 - number of children
 - salary
 - citizen
- The elements have different data types, so we can't conveniently use an array. Instead we will use a **struct** (short for structure)

2/12/07

CS250 Introduction to Computer Science II

3

Structure Declaration (7.1)

To store this information: We would begin by defining a structure :

```
struct Employ
{
• employee id → int id;
• SS# → int ssnun;
• number of children → int numchild;
• salary → float salary;
• citizen → bool bCitizen;
};
```

2/12/07

CS250 Introduction to Computer Science II

4

Struct Terminology (7.1)

For this struct:

```
struct Employ
{
  int id;
  int ssnun;
  int numchild;
  float salary;
  bool bCitizen;
};
```

- **Employ** is the **identifier name** and a new data type.
- The individual components **id**, **ssnum**, etc. are called **members**.

2/12/07

CS250 Introduction to Computer Science II

5

Notes on Structures (7.1)

- A semicolon is required after the closing brace of the structure declaration
- The structure declaration does not create a variable
- It just tells the compiler what that structure is made of
- The struct declaration is usually placed above the main

2/12/07

CS250 Introduction to Computer Science II

6

Variable Declaration (7.1)

- As with all data types, in order to use our new data type `Employ` we must **allocate** storage space by **declaring** variables of this data type:

```
Employ sEngineer, sTech;
```

- This will allocate space for two variables called `sEngineer` and `sTech`, each containing the previously described members `id`, `ssnum`, etc.
- Each of these variables is a separate instance of `Employ`

2/12/07

CS250 Introduction to Computer Science II

7

Dot Operator (7.2)

- To access a struct member, we use the **dot operator** (period between struct variable name and member name).
- In the variable `sEngineer` of data type `Employ` we can make the assignments:

```
sEngineer.id = 12345;  
sEngineer.ssnum = 534334343;  
sEngineer.numchild = 2;  
sEngineer.salary = 45443.34;  
sEngineer.bCitizen = true;
```

2/12/07

CS250 Introduction to Computer Science II

8

Notes on Structures (7.2)

- You cannot output the entire contents of a struct variable by simply using its name
 - `cout << sEngineer; // ERROR!`
- Similarly, you cannot compare two struct variables by using their name
 - `if(sEngineer == sTech) // ERROR!`

2/12/07

CS250 Introduction to Computer Science II

9

Example

- Write a C++ struct data type **RealNum** that will have members **number**, **realPart**, and **intPart**
- Declare a variable **sNumInfo** of that type
- Place the value **3.14159** in the field **number**

2/12/07

CS250 Introduction to Computer Science II

10

Structs as function arguments

- Structs can be passed to functions by reference or value in the same manner that other data types have been passed
- Generally, passing structs by reference is preferred since passing by value requires a local copy of the struct to be created within the function's variables

2/12/07

CS250 Introduction to Computer Science II

11

Example

- Write a C++ function **split** that accepts a variable of type **RealNum**
- Assign the integer part of the number to the member variable **intPart** and the real part of the number to the member variable **realPart**
- See the function prototype on the next slide

2/12/07

CS250 Introduction to Computer Science II

12

Example

- Function prototype:
`void split(RealNum &);`
- Function call:
`split (sNuminfo);`
- Function definition?

2/12/07

CS250 Introduction to Computer Science II

13

Initializing Structs (7.3)

- Use an initializer list
 - `Employ manager(12345, 534334356, 1, 76899, true);`
- You can initialize only some of the members in a struct, but members that follow a non initialized member must also be not initialized
 - `Employ manager(12345, 534334356, 1;`
 - `Employ manager(12345,,,, true);`

2/12/07

CS250 Introduction to Computer Science II

14

Initializing Structs (7.3)

- You cannot initialize structures in the declaration

```
struct Employ
{
    int id = 12345;
    int ssn = 534334356;
    int numchild = 1;
    float salary = 75000;
    bool bCitizen = true;
};
```

ERROR!

- Why?

2/12/07

CS250 Introduction to Computer Science II

15

Using a Constructor (7.3)

- It is possible to initialize a structure during declaration
- Use a constructor
- Constructor: A special function that can be used to construct, or set up and initialize a structure
- Looks like a regular function, but it's name is the same name as the name of the structure

2/12/07

CS250 Introduction to Computer Science II

16

Constructor Example (7.3)

```
struct Employ
{
    int id;
    int ssn;
    int numchild;
    float salary;
    bool bCitizen;
    Employ()
    {
        id = 0;
        ssn = 0;
        numchild = 0;
        salary = 0;
        bCitizen = true;
    }
};
```

2/12/07

CS250 Introduction to Computer Science II

17

Constructors (7.3)

- Constructors can accept arguments

```
struct PopInfo
{
    string name;
    long population;

    PopInfo(string n, long p)
    {
        name = n;
        population = p;
    }
};
```

2/12/07

CS250 Introduction to Computer Science II

18

Constructors (7.3)

- This allows us to initialize structure variables as they are defined

```
PopInfo forestGrove("Forest Grove", 19000);  
PopInfo portland("Portland", 556000);
```

2/12/07

CS250 Introduction to Computer Science II

19

Constructors (7.3)

- But, what if we didn't want to initialize the struct variable
 - `PopInfo city;`
- Adding empty parenthesis is incorrect:
 - `PopInfo city(); // ERROR!`

2/12/07

CS250 Introduction to Computer Science II

20
