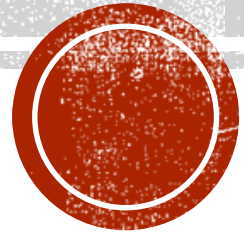
LECTURE 7

Advanced C/ C++ Programming

Structures

Chapter 6



LEARNING OBJECTIVES

- Structures
  - Structure types
  - Structures as function arguments
  - Initializing structures

STRUCTURES

- 2nd aggregate data type: struct
- Recall: aggregate meaning "grouping"
  - Recall array: collection of values of same type
  - Structure: collection of values of different types
- Treated as a single item, like arrays
- Major difference: Must first "define" struct
  - Prior to declaring any variables

STRUCTURE TYPES

- Define struct globally (typically)
- No memory is allocated
  - Just a "placeholder" for what our struct will "look like"
- Definition:

```
struct CDAccountV1   ←Name of new struct "type"
{
        double balance; ← member names
        double interestRate;
        int term;
};
```

DECLARE STRUCTURE VARIABLE

- With structure type defined, now declare variables of this new type:

CDAccountV1 **account**;

- Just like declaring simple types
- Variable *account* now of type CDAccountV1
- It contains "member values"
  - Each of the struct "parts"

ACCESSING STRUCTURE MEMBERS

- Dot Operator to access members
  - account.balance
  - account.interestRate
  - account.term
- Called "member variables"
  - The "parts" of the structure variable
  - Different structs can have same name member variables
    - No conflicts

STRUCTURE EXAMPLE: A STRUCTURE DEFINITION (1 OF 3)

Display 6.1 A Structure Definition

```
1  //Program to demonstrate the CDAccountV1 structure type.
2  #include <iostream>
3  using namespace std;

4  //Structure for a bank certificate of deposit:
5  struct CDAccountV1
6  {
7      double balance;
8      double interestRate;
9      int term;//months until maturity
10 };

11 void getData(CDAccountV1& theAccount);
12 //Postcondition: theAccount.balance, theAccount.interestRate, and
13 //theAccount.term have been given values that the user entered at the keyboar
```

An improved version of this structure will be given later in this chapter.

STRUCTURE EXAMPLE: A STRUCTURE DEFINITION (2 OF 3)

```
14  int main( )
15  {
16      CDAccountV1 account;
17      getData(account);

18      double rateFraction, interest;
19      rateFraction = account.interestRate/100.0;
20      interest = account.balance*(rateFraction*(account.term/12.0));
21      account.balance = account.balance + interest;

22      cout.setf(ios::fixed);
23      cout.setf(ios::showpoint);
24      cout.precision(2);
25      cout << "When your CD matures in "
26           << account.term << " months,\n"
27           << "it will have a balance of $"
28           << account.balance << endl;

29      return 0;
30  }
```

(continued)

STRUCTURE EXAMPLE: A STRUCTURE DEFINITION (3 OF 3)

Display 6.1 A Structure Definition

```
31  //Uses iostream:
32  void getData(CDAccountV1& theAccount)
33  {
34      cout << "Enter account balance: $";
35      cin >> theAccount.balance;
36      cout << "Enter account interest rate: ";
37      cin >> theAccount.interestRate;
38      cout << "Enter the number of months until maturity: ";
39      cin >> theAccount.term;
40  }
```

SAMPLE DIALOGUE

Enter account balance: **$100.00**
Enter account interest rate: **10.0**
Enter the number of months until maturity: **6**
When your CD matures in 6 months,
it will have a balance of $105.00

STRUCTURE PITFALL

- Semicolon after structure definition
  - ; MUST exist:

```
struct WeatherData
{
        double temperature;
        double windVelocity;
};
```

← **REQUIRED semicolon!**

  - Required since you "can" declare structure variables in this location

STRUCTURE ASSIGNMENTS

- Given structure named Fruits
- Declare two structure variables:
  Fruits apples, oranges;
  - Both are variables of "struct type Fruits "
  - Simple assignments are legal:
    apples = oranges;
    - Simply copies each member variable from apples into member variables from oranges

STRUCTURES AS FUNCTION ARGUMENTS

- Passed like any simple data type
  - Pass-by-value
  - Pass-by-reference
  - Or combination
- Can also be returned by function
  - Return-type is structure type
  - Return statement in function definition sends structure variable back to caller

INITIALIZING STRUCTURES

- Can initialize at declaration
  - Example:

```
struct Date
{
        int month;
        int day;
        int year;
};
Date dueDate = {12, 31, 2003};
```

  - Declaration provides initial data to all three member variables

- A member of a structure or of a class is accessed using the
  a) Comma operator
  b) The dot operator
  c) The indexing operator
  d) The ampersand operator
- Answer: b) the dot operator
- What is the error in the following structure definition?

```
struct A
{  int b;
  int c; }
int main(){
  A x;
  // other code
}
```

- Answer:
- The terminating semicolon is missing from the definition of struct A. Error messages for this may be clear "Semicolon missing from structure or class definition." However, one compiler says cryptically, "Too many types in declaration."

ARRAY OF STRUCTURES

```
#include <iostream>
#include <string>
using namespace std;
//------------------Bank Customer Structure-------------------------------
struct bankCustomer{
      char customerName[20];
      int accountNo;
      double balance;
};
```

```
int main(int argc, char** argv) {
int noOfCustomers = 0;
cout << "Please enter number of customers";
cin >> noOfCustomers;

bankCustomer customer[noOfCustomers];

for(int i = 0; i < noOfCustomers; i++){

        cout << "Customer Name: "<<endl;
        cin >> customer[i].customerName;

        cout << "Account number: "<<endl;
        cin >> customer[i].accountNo;

        cout << "Balance: "<<endl;
        cin >> customer[i].balance;

        }
```

```
cout << "/**************************/"<<endl;
cout << "////Details of existing cutomers////"<< endl;

for(int i = 0; i < noOfCustomers; i++){
        cout << "Customer Name: ";
        cout << customer[i].customerName<<endl;

        cout << "Account number: ";
        cout << customer[i].accountNo<<endl;

        cout << "Balance: ";
        cout << customer[i].balance<<endl;

        }
        return 0;
}
```