Lecture 2

ABSOLUTE C++
FIFTH EDITION

Structures and Classes

Walter Savitch

PEARSON
Learning Objectives

• Classes
  – Defining, member functions
  – Public and private members
  – Accessor and mutator functions
  – Structures vs. classes
Structure Definition

```c
struct type_name {
    type1 member_name1;
    type2 member_name2;
    type3 member_name3;
    ...
} object_names;

struct StudentInfo{
    int Id;
    int age;
    char Gender;
    double CGA;
} studentRecord;
```
Accessing Members

• use the *member access operator* `'.'`

```c++
struct Rectangle {
    int width;
    int length;
};

main() {
    Rectangle r1;
    r1.width = 5;
    r1.length = 8;
    cout << r1.width << endl;
    cout << r1.length << endl;
}
```
class class_name
{
    access specifier:
        member;
    access_specifier
        member;
    ...
};
class Rectangle
{
    private:
    int width;
    int length;
    public:
    void set(int w, int l);
    int area();
};
Define a Class Type

class class_name
{
    access_specifier: member;
    access_specifier: member;
    ...
};

class Rectangle
{
    private:
        int width;
        int length;
    public:
        void set(int w, int l);
        int area();
};
Class Member Access

• Members accessed same as structures
• Dot member selection operator ( . )
class Rectangle
{
    private:
        int width;
        int length;
    public:
        void set(int w, int l);
        int area();
};

main()
{
    Rectangle r1;
    Rectangle r2;

    r1.set(5, 8);
    cout<<r1.area()<<endl;

    r2.set(8,10);
    cout<<r2.area()<<endl;
}
• Member functions defined inside class

```cpp
class Circle {
    private:
        double radius;
    public:
        void setRadius(double r) { radius = r; }
        double getDiameter() { return radius * 2; }
        double getArea();
        double getCircumference();
};
```
Class Member Functions

• Member functions defined outside class
  • Using Binary scope resolution operator (::)
  • “Ties” member name to class name
  • Uniquely identify functions of particular class
  • Different classes can have member functions with same name
#include <iostream.h>

class circle
{
    private:
        double radius;
    
    public:
        void store(double);
        double area(void);
        void display(void);
};

// member function definitions

void circle::store(double r)
{
    radius = r;
}

double circle::area(void)
{
    return 3.14*radius*radius;
}

void circle::display(void)
{
    cout << "r = " << radius << endl;
}

int main(void) {
    circle c;   // an object of circle class
    c.store(5.0);
    cout << "The area of circle c is " << c.area() << endl;
    c.display();
}
Encapsulation

- Encapsulation
  - Means "bringing together as one"

- Declare a class ➔ get an object
What is an object?

OBJECT

Operations

- set of methods
  (public member functions)

Data

- internal state
  (values of private data members)
## Public and Private Members

<table>
<thead>
<tr>
<th>Access</th>
<th>Public</th>
<th>Protected</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Members of the same class</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Members of derived class</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Not members</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

```cpp
class Rectangle {
    private:
        int width;
        int length;
    public:
        void set(int w, int l);
        int area();
};
```
class Rectangle {

    int width;
    int length;

    public:
    void set(int w, int l);
    int area();

};
Abstract Data Type (ADT)

• data abstraction
• providing only essential information to the outside world
• hiding their background details
Accessor and Mutator Functions

• Accessor member functions
  – Also called "get member functions"
  – Allow object to read data

• Mutator member functions
  – Also called “set member functions”
  – Allow object to change data
// member function definitions

void circle::store(double r)
{
    radius = r;
}

double circle::area(void)
{
    return 3.14*radius*radius;
}

void circle::display(void)
{
    cout << "r = " << radius << endl;
}
Question

• Structures Vs Classes

<table>
<thead>
<tr>
<th>Structure</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default access specifier</td>
<td>Public</td>
</tr>
<tr>
<td>Member</td>
<td>Data</td>
</tr>
</tbody>
</table>

```c
struct Rectangle
{
    int width;
    int length;
};

class Rectangle
{
    private:
        int width;
        int length;
    public:
        void set(int w, int l);
        int area();
};
```
Complete Class Example:

Display 6.3  Class With a Member Function (1 of 4)

```cpp
// Program to demonstrate a very simple example of a class.
// A better version of the class DayOfYear will be given in Display 6.4.
#include <iostream>
using namespace std;

class DayOfYear
{
    public:
    void output();
    int month;
    int day;
};

int main()
{
    DayOfYear today, birthday;
    cout << "Enter today's date:\n";
    cout << "Enter month as a number: ";
    cin >> today.month;
    cout << "Enter the day of the month: ";
    cin >> today.day;
    cout << "Enter your birthday:\n";
    cout << "Enter month as a number: ";
    cin >> birthday.month;
    cout << "Enter the day of the month: ";
    cin >> birthday.day;
    (continued)
```
Complete Class Example:

**Display 6.3**  Class With a Member Function (2 of 4)

```cpp
Display 6.3  Class with a Member Function

25    cout << "Today's date is ";
26    today.output( );
27    cout << endl;
28    cout << "Your birthday is ";
29    birthday.output( );
30    cout << endl;
31    if (today.month == birthday.month && today.day == birthday.day)
32       cout << "Happy Birthday!\n";
33    else
34       cout << "Happy Unbirthday!\n";
35    return 0;
36 }
37  //Uses iostream:
38  void DayOfYear::output( )
39  {
40      switch (month)
41      {
42      case 1:
43         cout << "January "; break;
44      case 2:
45         cout << "February "; break;
46      case 3:
47         cout << "March "; break;
48      case 4:
49         cout << "April "; break;
Calls to the member function output
```

```cpp
Member function definition
```

Copyright © 2012 Pearson Addison-Wesley. All rights reserved.
Complete Class Example:

Display 6.3 Class With a Member Function (3 of 4)

```cpp
50    case 5:
51        cout << "May "; break;
52    case 6:
53        cout << "June "; break;
54    case 7:
55        cout << "July "; break;
56    case 8:
57        cout << "August "; break;
58    case 9:
59        cout << "September "; break;
60    case 10:
61        cout << "October "; break;
62    case 11:
63        cout << "November "; break;
64    case 12:
65        cout << "December "; break;
66    default:
67        cout << "Error in DayOfYear::output. Contact software vendor.";
68    }    cout << day;
69  }
70  }
```
Complete Class Example:

Display 6.3 Class With a Member Function (4 of 4)

Display 6.3 Class with a Member Function

Sample Dialogue

Enter today’s date:
Enter month as a number: 10
Enter the day of the month: 15
Enter your birthday:
Enter month as a number: 2
Enter the day of the month: 21
Today’s date is October 15
Your birthday is February 21
Happy Unbirthday!
Define a class student with the following specification

<table>
<thead>
<tr>
<th>Private members of class student</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admno</td>
<td>integer</td>
</tr>
<tr>
<td>sname</td>
<td>20 character</td>
</tr>
<tr>
<td>eng, math, science</td>
<td>float</td>
</tr>
<tr>
<td>total</td>
<td>float</td>
</tr>
<tr>
<td>ctotal()</td>
<td>a function to calculate eng+math+science with float return type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public members of class student</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Takedata()</td>
<td>Function to accept values for admno, sname, eng, science, and invoke ctotal() to calculate total</td>
</tr>
<tr>
<td>Showdata()</td>
<td>Function to display all the data members on the screen</td>
</tr>
</tbody>
</table>
#include <iostream>

using namespace std;

class student {
    private:
        int admno;
        char sname[20];
        float eng, math, science, total;
        float ctotal();
    public:
        void Takedata();
        void Showdata();
};
float student::ctotal(){
    return eng + math + science;
}

void student::Takedata(){
cout<<"Please key in admin number:"<<endl;
cin>>admno;
cout<<"Please key in student name:"<<endl;
cin>>sname;
cout<<"Please key in eng, math, science:"<<endl;
cin>>eng>>math>science;
total=ctotal();
}

void student::Showdata(){
cout<<"Admin number is:"<<admno<<endl;
cout<<"Student name is:"<<sname<<endl;
cout<<"English is:"<<eng<<endl;
cout<<"Math is:"<<math<<endl;
cout<<"Science is:"<<science<<endl;
cout<<"Total is:"<<total<<endl;
}
int main() {
    student st;
    st.Takedata();
    st.Showdata();
    return 0;
}

Please key in admin number: 12345
Please key in student name: kjh
Please key in eng, math, science: 30 49 51

Admin number is:12345
Student name is: kjh
English is: 30
Math is: 49
Science is: 51
Total is: 130
Thinking Objects

• Focus for programming changes
  – Before → algorithms center stage
  – OOP → data is focus

• Algorithms still exist
  – They simply focus on their data
  – Are "made" to "fit" the data

• Designing software solution
  – Define variety of objects and how they interact
Summary 1

• Structure is collection of different types

• Class used to combine data and functions into single unit -> object

• Member variables and member functions
  – Can be public → accessed outside class
  – Can be private → accessed only in a member function’s definition

• Class and structure types can be formal parameters to functions
Summary 2

• C++ class definition
  – Should separate two key parts
    • Interface: what user needs
    • Implementation: details of how class works
Example 1

class CellPhone {
    public: void setPrice (double newPrice);
    void setProfit (double newProfit);
    double getPrice();

    private:
    double price;
    double profit;
    double getProfit();

}

int main()
{
    CellPhone apple, samsung;

    apple.price = 1299.99;
    samsung.setPrice (2000.97);
    double aPrice, aProfit;
    aPrice = samsung.getPrice();
    aProfit = samsung.getProfit();
    aProfit = apple.getProfit();
    apple = samsung;

}