Chapter 6. 구조체와 클래스

박 종 혁 교수
UCS Lab
(http://www.parkjonghyuk.net)

Tel: 970-6702
Email: jhpark1@snut.ac.kr

Learning Objectives

- 구조체
  - 구조체 형
  - 함수 매개변수로서의 구조체
  - 구조체 초기화
- 클래스
  - 정의, 멤버 함수
  - public 과 private 멤버
  - accessor 와 mutator 함수
  - 구조체 vs. 클래스

구조체

- 두 번째 집합적 자료형: 구조체
- 집합의 의미는 "그룹화"
  - 배열: 같은 형의 값의 집합
  - 구조체: 다른 형의 값의 집합
- 배열과 같이 하나의 아이템으로 취급한다
- 주요한 차이점: 구조체는 먼저 "정의"해야 한다
  - 다른 변수들 보다 앞서서 선언

구조체 형

- 구조체는 전역으로 정의 (일반적)
- 메모리에 할당되는 것이 아니다
  - 공간 확보자(placeholder)로서의 역할
  - 새로운 형(사용자 정의형)을 생성하는 것
- 정의:

```
struct CDAccountV1   ← 새로운 구조체 형의 이름
{
        double balance;          ← 멤버들의 이름
        double interestRate;
        int term;
};
```

구조체 변수 선언

- 구조체 형을 정의하고, 새로운 형의 변수를 선언하는 것:

  CDAccountV1 account;

  - 기본형의 선언과 동일하다
  - 변수 *account* 는 CDAccountV1형
  - "멤버 값"을 가진다
    - 구조체의 각 부분

구조체 멤버 접근

- 도트(.) 연산자를 이용하여 멤버에 접근
  - account.balance
  - account.interestRate
  - account.term
- "멤버 변수"라고 부름
  - 구조체 변수의 부분
  - 다른 구조체는 같은 이름의 멤버 변수를 가질 수 있다
    - 충돌 없음

디스플레이 6.1 구조체 정의 (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.

디스플레이 6.1 구조체 정의 (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)

디스플레이 6.1 구조체 정의 (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

구조체 함정

- 구조체 정의 이후의 세미콜론(;)
  - 세미콜론이 반드시 존재해야 한다:

```
struct WeatherData
{
  double temperature;
  double windVelocity;
}; ← 세미콜론 필요!
```

  - 세미콜론 전에 구조체 변수를 선언할 수 있기 때문에 필요하다

구조체 할당

- CropYield 구조체
- 두 개의 구조체 변수 선언:
  CropYield apples, oranges;
  - 두 변수 모두 CropYield 구조체 형
  - 단순하게 할당이 가능하다:
    apples = oranges;
    - oranges의 멤버 변수가 apples의 각각의 멤버 변수로 단순하게 복사된다

함수 인자로서의 구조체

- 기본형 자료와 비슷하게 전달
  - Pass-by-value (Call-by-balue)
  - Pass-by-reference (Call-by-reference)
  - 또는 복합적으로 사용
- 함수에 의해 리턴될 수도 있다
  - 리턴 형은 구조체 형이 된다
  - 함수 정의부 내의 리턴문은 호출자에게 구조체 변수를 전달한다

계층적 구조체
(구조체 멤버를 가지는 구조체)

- (1)번 구조체의 태그가 (2)번 구조체의 멤버로 들어있다.
- 사용방법
  - PersonInfo person;
    person.birthday;
    person.birthday.year;

```
struct Date          ◄ (1)
{
Int month;
int day;
int year;
};

struct PersonInfo ◄ (2)
{
double height;
int weight;
Date birthday;
};
```

구조체 초기화

- 선언과 동시에 초기화 가능
  - 예:

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

  - 선언에서 세 개의 멤버 변수 모두에게 초기화 데이터를 제공

클래스

- 구조체와 유사
  - 멤버 변수 뿐만 아니라 "멤버 함수"가 추가된다
- OOP(object-oriented programming)의 핵심
  - 객체에 초점을 맞춘다
    - 객체: 데이터와 연산을 모두 포함한다
    - C++에서, 클래스 형의 변수가 객체가 된다

클래스 정의

- 구조체의 정의와 비슷하다
- 예:

```
class DayOfYear   ← 새로운 클래스 형의 이름
{
public:
        void output();   ← 멤버 함수!
        int month;
        int day;
};
```

- 단지 멤버 함수의 원형인 것에 주목!
  - 함수의 구현부는 다른 부분에 위치한다

객체의 선언

- 다른 변수의 선언과 동일
  - 사전 정의 형, 구조체 형
- 예:

  DayOfYear today, birthday;
  - DayOfYear 형의 두 개의 객체를 선언
- 객체가 포함하는 것:
  - 데이터(멤버 변수)
    - month, day
  - 연산 (멤버 함수)
    - output()

클래스 멤버 접근

- 멤버들에는 구조체와 같은 방법으로 접근
- 예:

  today.month

  today.day

  – 멤버 함수 접근:

  today.output(); ← 멤버 함수의 호출

클래스 멤버 함수

- 클래스 멤버 함수는 반드시 정의 또는 구현되어야 한다
- 다른 함수의 정의와 같다
  - main() 함수 이후에 정의 가능
  - 소속된 클래스를 반드시 명시해야 한다:
    void DayOfYear::output()
    {…}
    - :: 는 영역지정 연산자 (scope resolution operator) 또는 범위 결정 연산자
    - 컴파일러에게 함수가 어떤 클래스의 멤버인지를 알려준다
    - ::이전의 아이템을 제한자라고 한다

클래스 멤버 함수의 정의

- output() 멤버 함수의 정의에 주목 (다음 예제)
- 클래스 멤버 변수를 사용
  - 제한이 없다
- 함수는 클래스의 모든 객체에 대하여 사용된다
  - 현재 사용되는 객체의 데이터를 참조
  - 예:
    today.output();
    - today 객체의 데이터를 출력한다

디스플레이 6.3 멤버 함수를 가지는 클래스 (1 of 4)

Display 6.3 Class with a Member Function

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

5  class DayOfYear
6  {
7  public:
8      void output( );
9      int month;
10     int day;
11 };

12 int main( )
13 {
14     DayOfYear today, birthday;
15     cout << "Enter today's date:\n";
16     cout << "Enter month as a number: ";
17     cin >> today.month;
18     cout << "Enter the day of the month: ";
19     cin >> today.day;
20     cout << "Enter your birthday:\n";
21     cout << "Enter month as a number: ";
22     cin >> birthday.month;
23     cout << "Enter the day of the month: ";
24     cin >> birthday.day;
```

Normally, member variables are `private` *and not* `public`*, as in this example. This is discussed a bit later in this chapter.*

Member function declaration

(continued)

디스플레이 6.3 멤버 함수를 가지는 클래스 (2 of 4)

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**

Member function definition

디스플레이 6.3 멤버 함수를 가지는 클래스 (3 of 4)

```
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      }
69
70      cout << day;
71  }
```

디스플레이 6.3 멤버 함수를 가지는 클래스 (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!
```

도트(.) 와 영역 지정 연산자

- 멤버가 어디에 속하는지 지정하는 데 사용
- 도트(.) 연산자:
  - 특정 개체의 멤버 변수를 지정
- 영역 지정 연산자:
  - 어떤 클래스의 멤버 함수인지를 지정

클래스의 지위

- 클래스는 모든 것을 갖춘 형!
  - int, double 등과 같은 하나의 형
- 클래스 형의 변수를 가질 수 있다
  - 객체라고 부른다
- 클래스 형의 매개변수를 가질 수 있다
  - Pass-by-value (Call-by-value)
  - Pass-by-reference (Call-by-reference)
- 다른 형과 같이 클래스 형을 사용할 수 있다!

캡슐화(Encapsulation)

- 다음을 포함하는 어떠한 데이터 형
  - 데이터 (자료의 범위)
  - 연산 (데이터를 조작할 수 있는)
- 예:

  int 자료형은 :
  데이터 범위: +-32,767
  연산: +,-,*,/,%,logical,etc.
- 클래스도 동일하다
  - 우리는 데이터를 명시하고, 연산은 데이터의 조작을 허용한다!

추상 데이터 형
(Abstract Data Types)

- "추상(Abstract)"
  - 프로그래머는 세부 사항을 모른다
- 약어로 "ADT"
  - 데이터와 데이터의 값을 세팅하는 연산의 집합
- ADT는 "언어에 독립적이다"
  - C++에서 ADT는 클래스를 이용하여 구현
    - C++ 클래스는 ADT를 정의
  - 다른 언어들 또한 ADT구현

캡슐화의 확장

- 캡슐화
  - 하나의 단위로 모든 것을 전달
- 클래스의 선언 → 객체를 가진다
- 객체는 다음을 캡슐화한다
  - 데이터 (멤버 변수)
  - 자료의 연산 (멤버 함수)

OOP의 규칙

- 정보의 은닉
  - 클래스의 사용자는 연산이 어떻게 동작하는지에 대한 세부적인 사항을 알 수 없다
- 데이터의 추상화
  - 사용자는 ADT/클래스 내에서 데이터가 어떻게 취급되는지에 대한 세부 사항을 알지 못한다
- 캡슐화
  - 데이터와 연산을 함께 제공하지만, 세부 사항은 숨겨짐

public 과 private 멤버 (접근 제어자)

- 클래스 내의 데이터는 대부분 정의 안에서 private으로 설계!
  - OOP 규칙의 유지
  - 사용자로부터 데이터(멤버 변수)를 숨긴다
  - 연산을 통해서만 데이터의 조작을 허용
    - 멤버 함수를 사용
- public 아이템(대체로 멤버 함수)은 사용자의 접근이 가능하다

public 과 private 예 (1/2)

- 이전의 예를 수정:

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

- 데이터(멤버 변수)는 private
- 객체는 데이터를 직접 접근하지 못한다

public 과 private 예 (2/2)

- 이전의 예에서...
- 객체의 선언:
  DayOfYear today;
- 객체 *today*는 public 멤버에만 접근이 가능하다
  - cin >> today.month; // 허용되지 않음!
  - cout << today.day; // 허용되지 않음!
  - 대신 public 연산을 호출해야 한다:
    - today.input();
    - today.output();

public 과 private 스타일

- public & private은 혼용이 가능
- 더욱 일반적으로 public이 먼저 위치한다
  - 클래스를 사용하는 프로그래머에게 편의를 제공하기 위함
  - private 데이터는 사용자와 무관하므로 숨겨진다
- 클래스 정의 외부에서는 private 데이터의 접근과 수정이 불가능하다

accessor 와 mutator 함수

- 객체는 데이터를 조작할 수 있는 무엇인가가 필요하다
- accessor 멤버 함수 호출
  - 객체가 데이터를 읽을 수 있도록 허용
  - “get 멤버 함수”라고도 한다
  - 멤버 변수의 단순 검색
- mutator 멤버 함수
  - 객체가 데이터를 변경할 수 있도록 허용
  - “set 멤버 함수”라고도 한다
  - 응용에 기초한 조작

인터페이스(Interface)와 구현(Implementation)의 분리

- 클래스의 사용자는 클래스의 구현이 어떻게 되어있는지 세부적으로 알 필요는 없다
  - OOP의 규칙 → 캡슐화(encapsulation)
- 사용자는 단지 "사용방법"이 필요
  - 클래스의 인터페이스(interface)라고 부른다
    - C++ → public 멤버 함수와 그에 연관된 주석을 지칭한다
- 클래스의 구현은 숨겨진다
  - 멤버 함수는 다른 부분에 정의
  - 사용자는 볼 수 없다

구조체 vs. 클래스

- 구조체
  - 일반적으로 모든 멤버는 public
  - 멤버 함수가 없다
- 클래스
  - 일반적으로 모든 데이터 멤버는 private
  - 인터페이스 멤버 함수는 public
- 기술적으로는 같다
  - 개념적으로는 매우 다른 매커니즘

객체 중심의 사고

- 프로그래밍에 대한 초점(패러다임)의 변화
  - 이전 → 알고리즘이 중심
  - OOP → 객체(데이터) 중심
- 알고리즘은 여전히 존재한다
  - 단지 자신의 데이터에 초점을 준다
  - 데이터를 만들거나 수정
- 소프트웨어 솔루션의 설계
  - 다양한 객체의 정의와 그것들이 상호작용하는 방법

요약 (1/2)

- 구조체는 다른 데이터 형의 집합
- 클래스는 하나의 단위(객체)에서 데이터와 함수를 복합적으로 사용한다
- 멤버 변수와 멤버 함수
  - public → 클래스 외부에서 접근 가능
  - private → 멤버 함수의 정의를 통해서만 접근 가능
- 클래스와 구조체 형은 함수의 형식 매개변수가 될 수 있다

요약 (2/2)

- C++ 클래스의 정의
  - 두 개의 핵심 부분으로 구분한다
    - 인터페이스(Interface): 사용자가 필요로 하는 부분
    - 구현(Implementation): 클래스의 동작 방법에 대한 세부적인 부분

Q&A