윈도우 프로그래밍

4. 클래스(1)

2018. 3.30.

심미나 교수

목 차

I. C++의 구조체

C++의 구조체

구조체의 활용

- 서로 관련된 데이터를 하나로 묶어 프로그램 구현/관리의 편의성 증가
  - 관련된 데이터들은 생성 및 소멸 시점, 이동 및 전달 시점 또는 방법이 일치하므로
- 즉, 구조체란, 연관된 데이터를 하나로 그룹화하는 문법적 장치

```
(형식) struct 구조체명 {
            자료형   멤버변수;
            자료형   멤버변수;
            .....
            .....
        };
```

```
struct Employee
{
    char name[20];
    char dept[30];
    char job[30];
    int  year;
    char tel[20];
    char email[40];
};
```

- 이름
- 소속
- 담당업무
- 재직기간
- 전화번호
- 메일주소

C++의 구조체

구조체 변수의 선언

- **구조체 변수 선언**

(형식) 구조체명 구조체 변수명;

  - C와 달리 struct 키워드 생략 위한 typedef선언 불필요

```
struct Car basicCar;
struct Car simpleCar;
```

⇒

```
Car basicCar;
Car simpleCar;
```

```
struct Car
{
    char gamerID[ID_LEN];   // 소유자ID
    int fuelGauge;          // 연료량
    int curSpeed;           // 현재속도
};
```

- 소유주
- 연료량
- 현재속도
- 취득점수
- 취득아이템

C++의 구조체

구조체 변수의 선언

- **구조체 변수 선언**

(형식) 구조체명 구조체 변수명;

– 데이터 뿐만 아니라, 해당 데이터와 연관된 함수도 하나로 그룹화하여 활용 가치를 부여

```
struct Car
{
    char gamerID[ID_LEN];   // 소유자ID
    int fuelGauge;          // 연료량
    int curSpeed;           // 현재속도
};
```

```
void ShowCarState(const Car &car)
{
    . . . .
}
void Accel(Car &car)
{
    . . . .
}
void Break(Car &car)
{
    . . . .
}
```

C++의 구조체

구조체 변수의 선언

- **구조체 내 함수 삽입**
  - C++에서는 구조체 내에 함수 삽입이 가능 → '클래스'
  - 구조체 내에 함께 선언된 변수에는 직접 접근이 가능

```
void ShowCarState()
{
    cout<<"소유자ID: "<<gamerID<<endl;    // 위에 선언된 gamerID에 접근
    cout<<"연료량: "<<fuelGauge<<"%"<<endl;
    cout<<"현재속도: "<<curSpeed<<"km/s"<<endl<<endl;
}
```

```
void Break()
{
    if(curSpeed<BRK_STEP)
    {
        curSpeed=0;    // 위에 선언된 curSpeed에 접근
        return;
    }

    curSpeed-=BRK_STEP;
}
```

```
struct Car
{
    char gamerID[ID_LEN];
    int fuelGauge;
    int curSpeed;

    void ShowCarState()
    {
        . . . .
    }
    void Accel()
    {
        . . . .
    }
    void Break()
    {
        . . . .
    }
};
```

ⓘ 같은 구조체 안에 선언된 변수 접근 시 변수명 그대로 사용

ⓘ 구조체 안의 함수에서는 동일수준의 함수를 그대로 호출 가능

C++의 구조체

구조체 변수의 선언

- **구조체 변수의 생성**
  - 생성된 구조체 변수마다 함수가 독립적으로 존재하는 구조로 보임(논리적 의미)
    → '변수(객체)'로 이해하기!
  - 구조체 함수는 초기화 대상이 아니며, 멤버만 초기화!

(예시)
Car run99 = {"run99", 100, 0}
Car sped77 = {"sped77", 100, 0}

ⓘ 멤버변수접근 (객체명.멤버변수)
- run99.gamerID

ⓘ 멤버함수호출 (객체명.멤버함수())
- run99.ShowCarState

C++의 구조체

구조체 정의

- **구조체 내에 enum 상수의 선언**
  - 구조체 내에 enum선언을 함으로써 잘못된 외부 접근을 제한 가능
  - (예시) Car 클래스를 위해 정의된 상수

```
#define ID_LEN      20
#define MAX_SPD     200
#define FUEL_STEP   2
#define ACC_STEP    10
#define BRK_STEP    10
```

```
namespace CAR_CONST
{
    enum
    {
        ID_LEN      =20,
        MAX_SPD     =200,
        FUEL_STEP   =2,
        ACC_STEP    =10,
        BRK_STEP    =10
    };
}
```

ⓘ CAR_CONST.ID_LEN 으로 접근

* 연관된 상수들을 하나의 이름공간에 별도로 그룹화 가능

```
struct Car
{
    enum
    {
        ID_LEN      =20,
        MAX_SPD     =200,
        FUEL_STEP   =2,
        ACC_STEP    =10,
        BRK_STEP    =10
    };

    char gamerID[ID_LEN];
    int fuelGauge;
    int curSpeed;

    void ShowCarState() { . . . . }
    void Accel() { . . . . }
    void Break() { . . . . }
};
```

ⓘ 직접접근

C++의 구조체

구조체 정의

- **구조체 내에 선언된 함수의 정의**
  - 구조체 안에 정의된 함수는 inline 선언된 것으로 간주

```
struct Car
{
    . . . . .
    void ShowCarState();
    void Accel();
    . . . . .
};
```

ⓘ 구조체 내 삽입된 함수 선언

```
void Car::ShowCarState()
{
    . . . . .
}
void Car::Accel()
{
    . . . . .
}
```

ⓘ 구조체 내 선언된 함수 정의

  - 외부에 함수 정의시 명시적 inline 선언

```
inline void Car::ShowCarState() { . . . . . }
inline void Car::Accel() { . . . . }
inline void Car::Break() { . . . . }
```

II. 클래스의 이해

클래스의 이해

클래스(Class)와 객체(Object)

- **접근지정자**, private와 public
  - Public : 어디서나 접근이 허용되는 상태
  - Private : 클래스 내(에 정의된 함수)에서만 접근이 허용되는 상태
  - Protected : (상속관계에 있을 경우), 유도클래스에서만 접근이 허용되는 상태

ⓘ private 멤버선언

ⓘ public 멤버선언

```
class Car
{
private:
    char gamerID[CAR_CONST::ID_LEN];
    int fuelGauge;
    int curSpeed;
public:
    void InitMembers(char * ID, int fuel);
    void ShowCarState();
    void Accel();
    void Break();
};
```

```
int main(void)
{
    Car run99;
    run99.InitMembers("run99", 100);
    run99.Accel();
    run99.Accel();
    run99.Accel();
    run99.ShowCarState();
    run99.Break();
    run99.ShowCarState();
    return 0;
}
```

ⓘ C++ 객체

ⓘCar의 멤버함수는 모두 public이므로 클래스외부의 main함수에서 접근 가능함

클래스의 이해

클래스(Class)와 객체(Object)

- 용어정의 - 객체(object), 멤버변수, 멤버함수
  - 객체 : 설계된 특정 클래스를 대상으로 생성된 변수
    즉, 추상화된 자료를 클래스로 설계한 후, 이를 실체화(instance)한 것
  - 멤버변수 : 특정 클래스 내에 선언된 변수 즉, 클래스를 구성하는 변수
  - 멤버함수(메소드) : 특정 클래스 내에 정의된 함수
    즉, 멤버함수는 클래스 내에 정의된 private멤버 접근을 위한 것

ⓘ 멤버변수 선언

ⓘ 멤버함수 선언

```
class Car
{
private:
    char gamerID[CAR_CONST::ID_LEN];
    int fuelGauge;
    int curSpeed;
public:
    void InitMembers(char * ID, int fuel);
    void ShowCarState();
    void Accel();
    void Break();
};
```

ⓘ C++ 객체

```
int main(void)
{
    Car run99;
    run99.InitMembers("run99", 100);
    run99.Accel();
    run99.Accel();
    run99.Accel();
    run99.ShowCarState();
    run99.Break();
    run99.ShowCarState();
    return 0;
}
```

© 2018 by Mina Shim

클래스의 이해

구조체와의 비교

- 외형적 차이 - 키워드 struct 대신 class 사용한다는 것
  - C++에서는 struct로 선언되어도 모두 클래스로 취급
- 의미상 차이 - 접근지정자 없을 때 디폴트 접근허용 여부의 차이

"struct"로 선언되고 접근명시 없을 때, 디폴트는 public
"class"로 선언되고 접근명시 없을 때, 디폴트는 private

```
class Car
{
    Private:
    char gamerID[CAR_CONST::ID_LEN];
    int fuelGauge;
    int curSpeed;

    void ShowCarState() { . . . . }
    void Accel() { . . . . }
    void Break() { . . . . }
};
```

```
int main(void)
{
    Car run99;
    strcpy(run99.gamerID, "run99");      (×)
    run99.fuelGauge=100;                 (×)
    run99.curSpeed=0;                    (×)
    . . . .
```

ⓘ Car가 class가 아닌 struct로 선언되었다면 모두 접근 가능함

클래스의 이해

클래스 선언

- **클래스 선언 - 일반적으로 헤더파일에 추가**
  - 객체 생성문과 멤버의 접근 등 호출관계를 정립하기 위해 컴파일 과정에 필요
  - 클래스 이름을 따라 "클래스명.h"로 명명 (class car -> car.h)
  - 멤버변수, 멤버함수의 원형은 헤더파일에 선언
  - 인라인 함수는 컴파일 과정에서 함수 호출문을 대체하므로 헤더파일에 정의
- **멤버함수의 몸체 정의**
  - 멤버함수의 몸체는 링크 과정에 필요하므로 cpp파일에 정의 (car.cpp)

car.h

```
class Car
{
private:
    char gamerID[CAR_CONST::ID_LEN];
    int fuelGauge;
    int curSpeed;
public:
    void InitMembers(char * ID, int fuel);
    void ShowCarState();
    void Accel();
    void Break();
};
```

ⓘ 멤버함수 원형선언

```
void Car::InitMembers(char * ID, int fuel) { . . . . }
void Car::ShowCarState() { . . . . }
void Car::Accel() { . . . . }
void Car::Break() { . . . . }
```

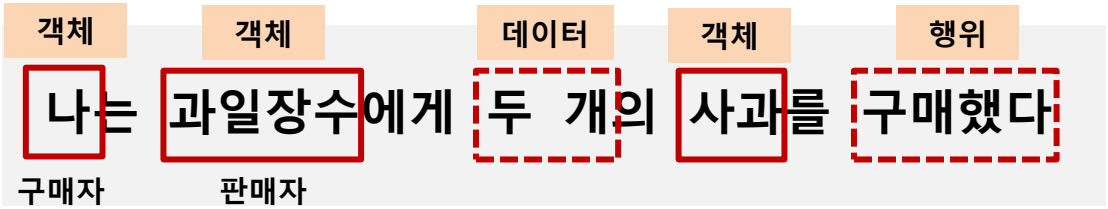
ⓘ 멤버함수의 몸체 정의

car.cpp

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- 객체(Object)
  - (사전적 의미) 물건 또는 대상
  - (객체지향프로그래밍 의미) '객체 중심의' 프로그래밍
- 객체지향 프로그래밍이란, 현실에 존재하는 사물과 대상, 그에 따른 행위들을 있는 그대로 실체화시키는 형태의 프로그래밍



클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- 객체(Object) - 데이터(멤버변수), 기능(멤버함수)
  - 객체의 행위는 기능으로 '함수'로 표현하고, 상태는 변화하는 '데이터'로 표현됨
- 과일장수 객체의 표현

- 과일장수는 과일을 판다. (행위)
- 과일장수는 사과 20개와 오렌지 10개를 갖고 있다. (상태)
- 과일장수의 과일판매수익은 현재까지 50,000원이다. (상태)

ⓘ 과일장수의 행위(기능) 표현

```
int SaleApples(int money) //사과구매액이 인자로 전달
{
    int num = money/1000;   // 사과 단가가 1000원
    numOfApples -= num;     // (판매로) 보유한 사과수 감소
    myMoney += money;       // (판매로) 판매수익 발생
    return num;             // 실제 구매가 발생한(판매한)
                               사과 수 반환
}
```

```
int numOfApples; //보유한 사과 수
int myMoney;     // 판매수익
```

ⓘ 과일장수의 데이터 표현

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- '과일장수(판매자)' 클래스 정의와 멤버변수의 상수화
  - 판매에 따라 상태가 변화하는 데이터(사과 수, 판매수익)를 멤버변수로 선언
  - 판매행위를 구현하는 함수(사과판매함수)를 멤버함수로 정의
  - 판매현황을 보여주는 함수(남은 상태, 판매수익 보기함수)를 멤버함수로 정의
  - 멤버(사과 단가, 사과 수, 판매수익)를 초기화하는 함수를 멤버함수로 정의

```
class FruitSeller
{
private:
   int APPLE_PPRICE;
   int numOfApples;
   int myMoney;

public:
   int SaleApples(int money)
   {
      int num = money/1000;
      numOfApples -= num;
      myMoney += money;
      return num;
   }
};
```

ⓘ 단가는 변하는 값이 아님
const int APPLE_PRICE!?

ⓘ 변수 선언

ⓘ 함수 정의

1) 판매 함수

2) 초기화 함수

```
void InitMembers(int price, int num, int money)
{
   APPLE_PRICE = price;
   numOfApples = num;
   myMoney = money;
}
```

3) 판매현황보기 함수

```
void ShowSalesResult()
{
   cout<<"남은 사과:  "<<numOfApples<<endl;
   cout<<"판매 수익:  "<<myMoney<<endl;
}
```

© 2018 by Mina Shim

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- '나(구매자)' 클래스 정의
  - 구매에 따라 상태가 변화하는 데이터(사과 수, 보유금액)를 멤버변수로 선언
  - 구매행위를 구현하는 함수(사과구매함수)를 멤버함수로 정의
  - 구매현황을 보여주는 함수(구매사과 수, 보유잔액 보기함수)를 멤버함수로 정의
  - 멤버((구매전) 보유금액, 사과 수)를 초기화하는 함수를 멤버함수로 정의

ⓘ 변수 선언

```
class FruitBuyer
{
    int myMoney;       //private
    int numOfApples;   //private
public:
    void InitMembers(int money)
    {
        myMoney = money;
        numOfApples=0;  //구매 전
    }
    void BuyApples(FruitSeller &seller, int money)
    {
        numOfApples += seller.SaleApples(money);
        myMoney -= money;
    }
    void ShowBuyResult()
    {
        cout<<"현재 잔액:  "<<myMoney<<endl;
        cout<<"사과 개수:  "<<numOfApples<<endl;
    }
};
```

ⓘ 함수 정의

1) 초기화 함수
2) 구매 함수
3) 구매현황보기 함수

윈도우프로그

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- **객체 생성**
  - '과일장수(판매자)'와 '나(구매자)'의 객체 생성

1) 일반적인 변수 선언 방식의 객체생성

```
FruitSeller seller;
FruitBuyer buyer;
```

2) 동적할당 방식의 객체 생성 → 클래스 적용 가능!

- 포인터 구조체 형태로 메모리 동적할당 받아 생성

```
FruitSeller * objPtr1 = new FruitSeller;
FruitBuyer * objPtr2 = new FruitBuyer;
```

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- **현실구현을 위한 메시지 전달 과정**
  - 메시지 전달 준비 → 실직적인 메시지 전달 → 요구에 대한 반환

- **과일장수 시뮬레이션**
  - '과일장수(판매자)', '나(구매자)' 객체 생성: 과일장수 아저씨와 나 등장
  - 구매자가 사과구매: "아저씨, 사과 2000원어치 주세요!"
  - 구매자가 사과 판매현황 질의: "아저씨, 오늘 얼마나 파셨어요?"
  - 구매자가 사과 구매현황 확인: "오늘 사과사고 돈이 얼마나 남았지?"

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- **두 객체간 대화방법**
  - 함수호출: 메시지 전달(메시지패싱)

```
int main(void)
{
    FruitSeller seller;
    seller.InitMembers(1000, 20, 0);
    FruitBuyer buyer;
    buyer.InitMembers(5000);
    buyer.BuyApples(seller, 2000);

    cout<<"과일 판매자의 현황"<<endl;
    seller.ShowSalesResult();
    cout<<"과일 구매자의 현황"<<endl;
    buyer.ShowBuyResult();
    return 0;
}
```

1) 과일장수 seller 아저씨 등장
   (단가 1000원, 사과 20개, 0원 보유)

2) 나 buyer 등장 (5000원 보유)

3) "seller 아저씨, 사과 2000원어치 주세요!"

4) "seller 아저씨, 오늘 얼마나 파셨어요?"

5) "오늘 사과사고 돈이 (나 buyer에게) 얼마나 남았지?"

클래스의 이해 - 사례

객체지향 프로그래밍의 이해 - 과일장수 사례

- **두 객체간 대화방법**
  - **함수호출: 메시지 전달(메시지패싱)**

```
int main(void)
{
    FruitSeller seller;
    seller.InitMembers(1000, 20, 0);
    FruitBuyer buyer;
    buyer.InitMembers(5000);
    buyer.BuyApples(seller, 2000);

    cout<<"과일 판매자의 현황"<<endl;
    seller.ShowSalesResult();
    cout<<"과일 구매자의 현황"<<endl;
    buyer.ShowBuyResult();
    return 0;
}
```

(준비) Seller를 대상으로, 2000원어치, 구매하겠다

(전달) FruitBuyer 객체가 FruitSeller 객체의 SaleApples 함수 호출 (메시지 전달!!!)

```
void BuyApples(FruitSeller &seller, int money)
{
    numOfApples+=seller.SaleApples(money);
    myMoney-=money;
}
```

(반환)

IV. 실습

과일장수 시뮬레이션하기

- 클래스 FruitSeller 정의

```
#include <iostream>
using namespace std;

class FruitSeller
{
private:
		int APPLE_PRICE;
		int numOfApples;
		int myMoney;

public:
		void InitMembers(int price, int num, int money)
		{
			APPLE_PRICE=price;
			numOfApples=num;
			myMoney=money;
		}
		int SaleApples(int money)
		{
			int num=money/APPLE_PRICE;
			numOfApples-=num;
			myMoney+=money;
			return num;
		}
		void ShowSalesResult()
		{
			cout<<"남은 사과: "<<numOfApples<<endl;
			cout<<"판매 수익: "<<myMoney<<endl<<endl;
		}
};
```

과일장수 시뮬레이션하기

- 클래스 FruitBuyer 정의

```
class FruitBuyer
{
		int myMoney;			// private:
		int numOfApples;		// private:

public:
		void InitMembers(int money)
		{
				myMoney=money;
				numOfApples=0;
		}
		void BuyApples(FruitSeller &seller, int money)
		{
				numOfApples+=seller.SaleApples(money);
				myMoney-=money;
		}
		void ShowBuyResult()
		{
				cout<<"현재 잔액: "<<myMoney<<endl;
				cout<<"사과 개수: "<<numOfApples<<endl<<endl;
		}
};
```

과일장수 시뮬레이션하기

- Main함수

```
int main(void)
{
		FruitSeller seller;
		seller.InitMembers(1000, 20, 0);
		FruitBuyer buyer;
		buyer.InitMembers(5000);
		buyer.BuyApples(seller, 2000);

		cout<<"과일 판매자의 현황"<<endl;
		seller.ShowSalesResult();
		cout<<"과일 구매자의 현황"<<endl;
		buyer.ShowBuyResult();
		return 0;
}
```

실습4-02

교재 - 예제 10-1.(p.359)

복소수를 클래스로 설계하기

```
01 #include <iostream>
02 using namespace std;
03 class Complex
04 {
05 private :
06     int real;
07     int image;
08 public :
09     void SetComplex();
10     void ShowComplex();
11 };
12
13 void Complex::SetComplex()
14 {
15     real=2;
16     image=5;
17 }
```

```
18 void Complex::ShowComplex()
19 {
20     cout<<"( " <<real <<" + " <<image
        << "i )" <<endl ;
21 }
22 void main()
23 {
24     Complex x, y;
25
26     x.SetComplex();
27     x.ShowComplex();
28     y.SetComplex();
29     y.ShowComplex();
30 }
```

실행 결과

```
( 2 + 5i )
( 2 + 5i )
```

실습4-03

교재 - 예제 10-2.(p.361)

private 멤버 성격 파악하기

```
01 #include <iostream>
02 using namespace std;
03 class Complex
04 {
05 private :
06     int real;
07     int image;
08 public :
09     void SetComplex();
10     void ShowComplex();
11 };
12
13 void Complex::SetComplex()
14 {
15     real=2;
16     image=5;
17 }
```

```
18 void Complex::ShowComplex()
19 {
20     cout<<"( " <<real <<" + " <<image
        << "i )" <<endl ;
21 }
22 void main()
23 {
24     Complex x, y;
25
26     x.real = 5;  //컴파일 에러
27     x.image = 10;  //컴파일 에러
28     y.SetComplex();
29     y.ShowComplex();
30 }
```

```
error C2248: 'Complex::real' : private 멤버('Complex' 클래스에서 선언)에 액세스할 수 없습니다.
error C2248: 'Complex::image' : private 멤버('Complex' 클래스에서 선언)에 액세스할 수 없습니다.
IntelliSense: 멤버 "Complex::real" (선언된 줄 6)에 액세스할 수 없습니다.
IntelliSense: 멤버 "Complex::image" (선언된 줄 7)에 액세스할 수 없습니다.
```

과제4

파일명 “과제4_분반_학번_이름”으로 제출

- **과제4**
  - 과제4-1 : 소스코드(ex401_학번.cpp), 실행결과화면(ex402_학번.jpg)
  - 과제4-2 : 소스코드(ex402_학번.cpp), 실행결과화면(ex402_학번.jpg)
- **제출 시 주의사항**
  - 실행결과 마지막에는 “학과, 학년, 분반, 학번, 이름” 출력할 것
    - (형식: 컴퓨터공학부 2학년 1반, 2017000번, 홍길동입니다.)
  - 각 예제의 소스코드(*.cpp)와 실행결과화면(*.jpg 등를 한 개의 zip파일로 만들어 제출할 것

감사합니다

mnshim@sungkyul.ac.kr