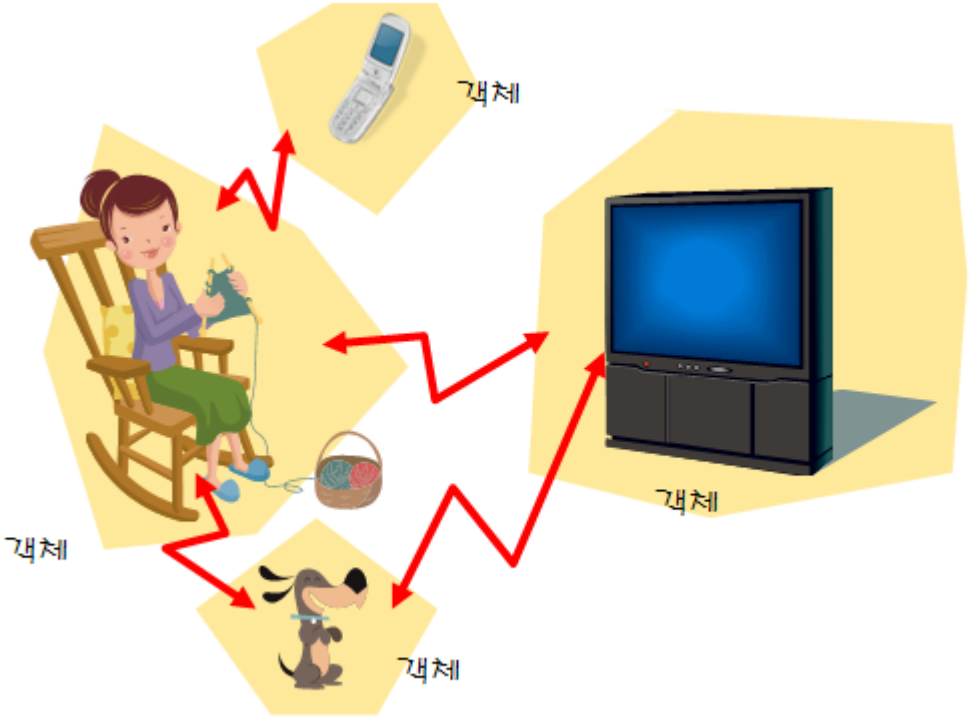
C++ Exspresso

클래스의 기초 - 보충 자료



기호 상수를 만드는 방법

const 키워드 이용

```
#include <iostream>
using namespace std;        // 이름공간설정
int main()
{
    const int MONTHS = 12;// 기호상수선언
    double m_salary, y_salary;        // 변수선언

    cout << "월급을입력하시요: ";    // 입력안내문
    cin >> m_salary;

    y_salary = 12 * m_salary;         // 순수입계산
    cout << "연봉은" << y_salary << "입니다" << endl;
    return 0;
}
```

기호 상수 정의

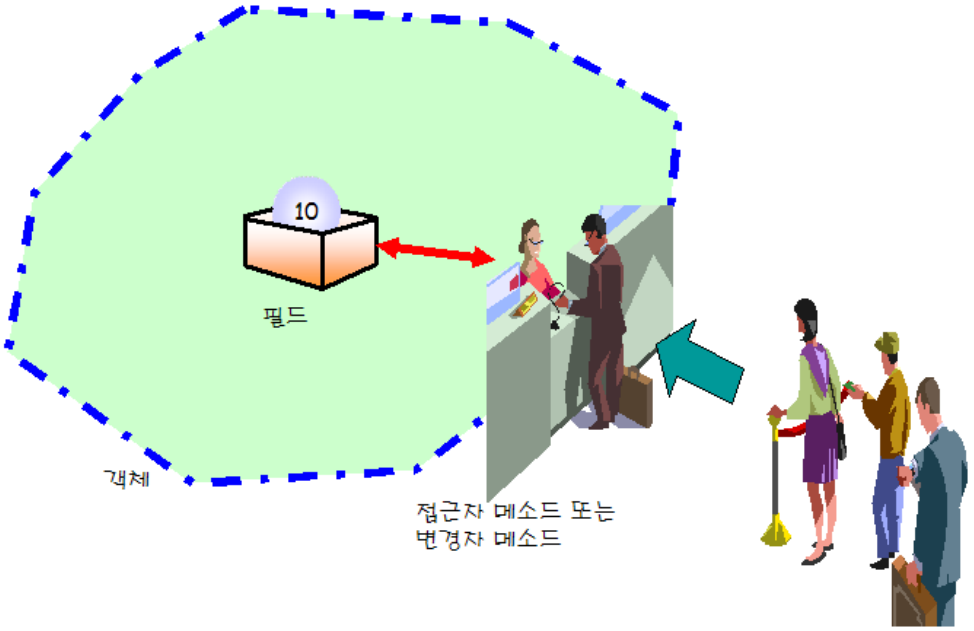
멤버함수 접근자와 설정자

- 접근자(accessor): 멤버 변수의 값을 반환

(예) getBalance()

- 설정자(mutator): 멤버 변수의 값을 설정

(예) setBalance();



예제 -1

```
class Car {
private:
    // 멤버 변수 선언
    int speed;          //속도
    int gear;           //기어
    string color;       //색상
    ...
public:
    // 접근자 선언
    int getSpeed() {
        return speed;
    }
    // 설정자 선언
    void setSpeed(int s) {
        speed = s;
    }
```

예 제

```
// 접근자 선언
    int getGear() {
        return gear;
    }
    // 변경자 선언
    void setGear(int g) {
        gear = g;
    }
    // 접근자 선언
    string getColor() {
        return color;
    }
    // 변경자 선언
    void setColor(string c) {
        color = c;
    }
};
```

예제 -2

```
class Car {
private:
    // 멤버 변수 선언
    int speed;          //속도
    int gear;           //기어
    string color;       //색상
    ...
public:
    // 접근자 선언
    int getSpeed() const {
        return speed;
    }
    // 설정자 선언
    void setSpeed(int s) {
        speed = s;
    }
```

예제

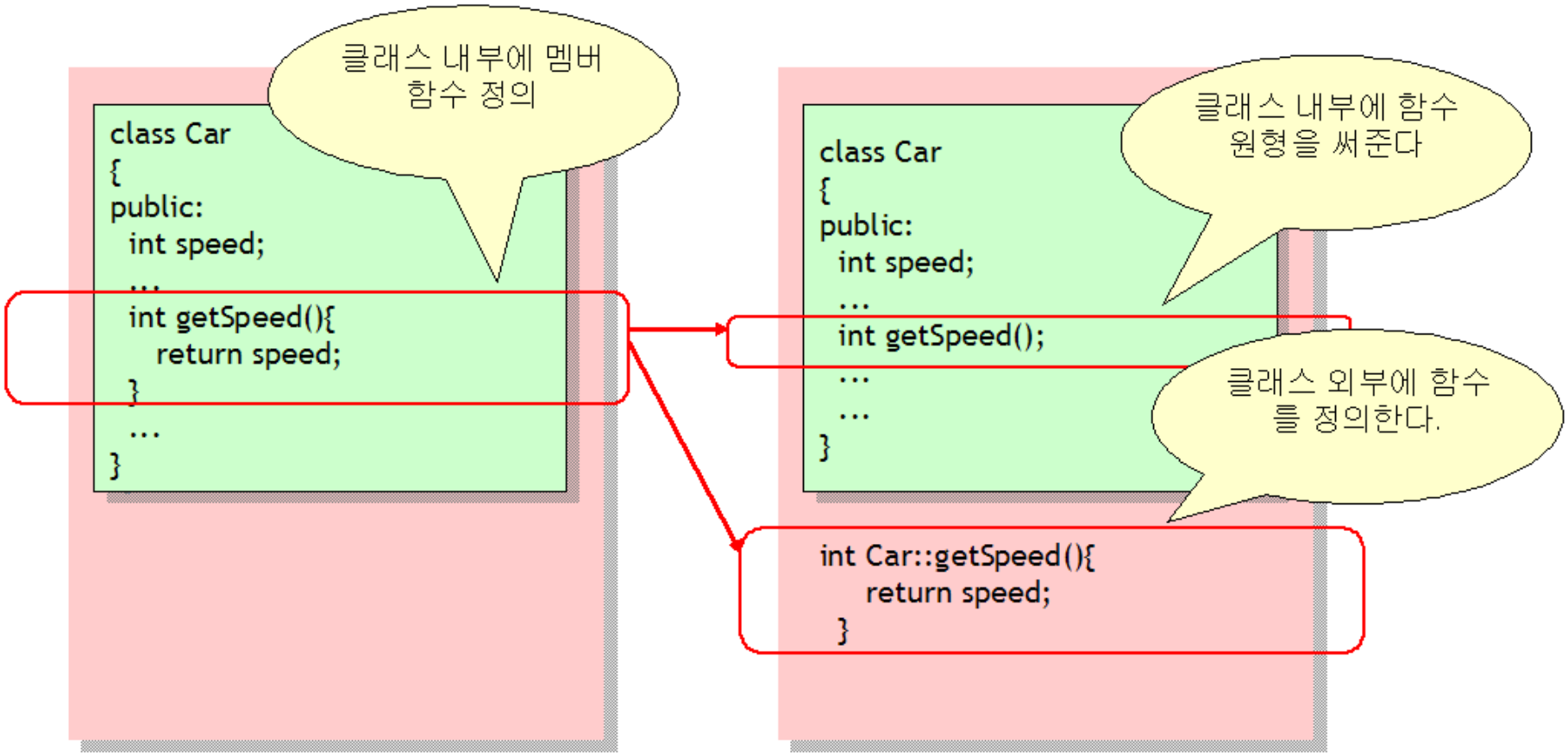
```
// 접근자 선언
    int getGear() const {
        return gear;
    }
    // 변경자 선언
    void setGear(int g) {
        gear = g;
    }
    // 접근자 선언
    string getColor() const {
        return color;
    }
    // 변경자 선언
    void setColor(string c) {
        color = c;
    }
};
```

접근자와 설정자의 장점

- 설정자의 매개 변수를 통하여 잘못된 값이 넘어오는 경우, 이를 사전에 차단할 수 있다.
- 멤버 변수값을 필요할 때마다 계산하여 반환할 수 있다.
- 접근자만을 제공하면 자동적으로 읽기만 가능한 멤버 변수를 만들 수 있다.

```
void setSpeed(int s)
{
        if( s < 0 )
                speed = 0;
        else
                speed = s;
}
```

멤버 함수의 외부 정의



내부 정의와 외부 정의의 차이

- 멤버 함수가 클래스 내부에 정의되면 자동적으로 인라인(inline) 함수가 된다.
- 멤버 함수가 클래스 외부에 정의되면 일반적인 함수와 동일하게 호출한다.

예 제

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

class Car {
public:
        int getSpeed();
        void setSpeed(int s);
        void honk();
private:
        int speed;                      //속도
};

int Car::getSpeed()
{
        return speed;
}
void Car::setSpeed(int s)
{
        speed = s;
}
```

예제

```
void Car::honk()
{
        cout << "빵빵!" << endl;
}

int main()
{
        Car myCar;
        myCar.setSpeed(80);
        myCar.honk();
        cout << "현재 속도는" << myCar.getSpeed() << endl;
        return 0;
}
```

```
빵빵!
현재 속도는 80
계속하려면 아무 키나 누르십시오 . . .
```

인라인 함수

- 함수 이름 앞에 **inline**이 붙으면 컴파일러는 함수를 생성하지 않고 함수의 코드를 호출한 곳에 직접 집어넣는다.

```
// 실수값을 제곱하는 함수
inline double square(double i)
{
        return i*i;
}
```

예제

main.cpp

```
#include <iostream>
#include "car.h"                    //
현재위치에car.h를읽어서넣으라는것을의미한다.
using namespace std;

int main()
{
        Car myCar;
        myCar.setSpeed(80);
        myCar.honk();
        cout << "현재속도는" << myCar.getSpeed() << endl;
        return 0;
}
```

클래스를 사용한다.

멤버 함수의 중복 정의

- 멤버 함수도 중복 정의(오버로딩)가 가능함

```
class Car {
private:
    int speed;                  //속도
    int gear;                   //기어
    string color;       //색상
public:
    int getSpeed();
    void setSpeed(int s);
    void setSpeed(double s);
};
```

멤버 함수의 중복 정의

```
int Car::getSpeed() {
    return speed;
}
void Car::setSpeed(int s) {
    speed = s;
}
void Car::setSpeed(double s) {
    speed = (int)s;
}
int main()
{
        Car myCar;
        myCar.setSpeed(80);
        myCar.setSpeed(100.0);
        cout << "차의 속도: " << myCar.getSpeed() << endl;
        return 0;
}
```

멤버 함수 중복 정의