07

프렌드와 연산자 중복

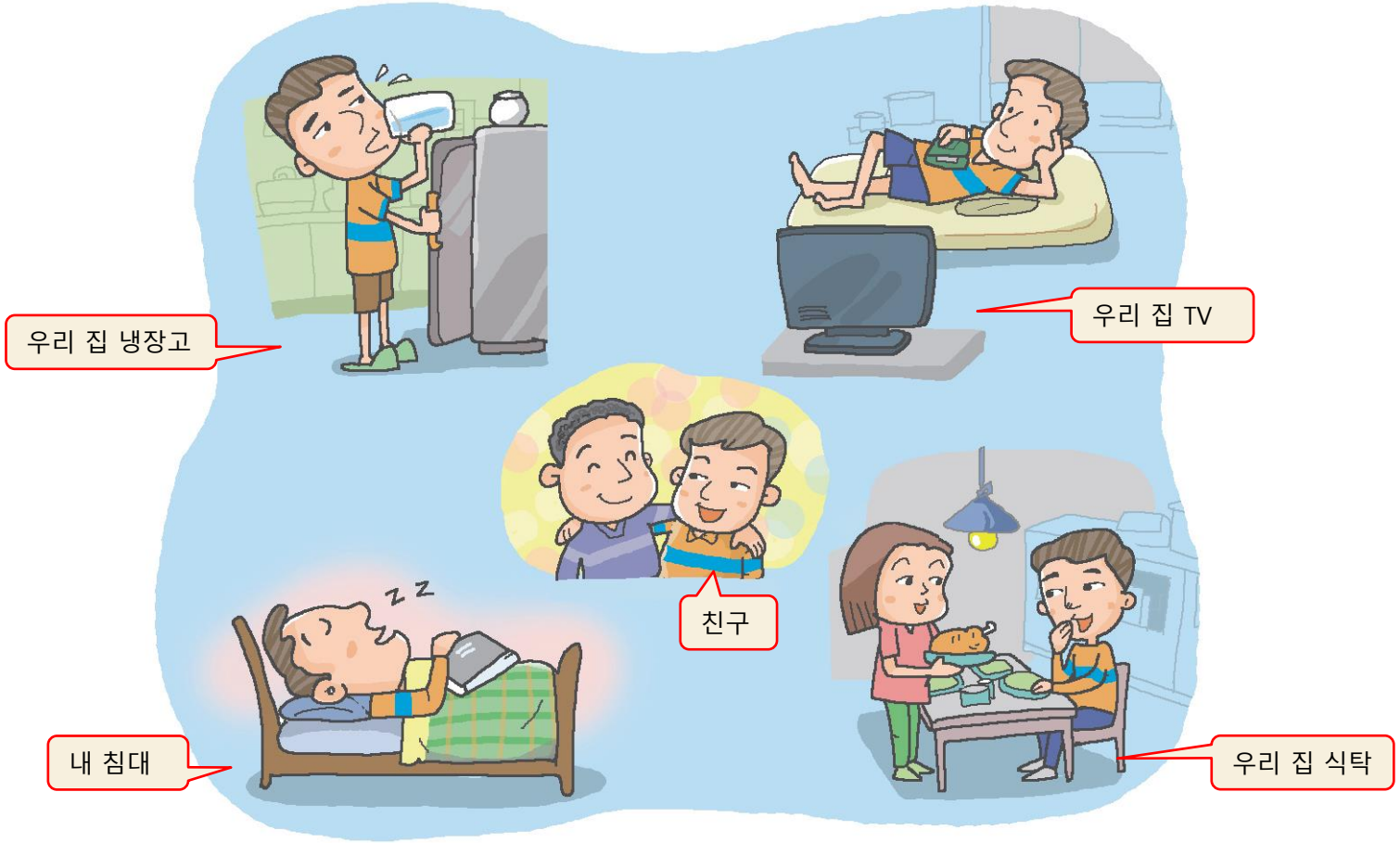
학습 목표

1. C++의 프렌드 함수의 개념을 이해하고, 프렌드 함수를 작성할 수 있다.
2. 연산자 중복의 개념을 이해하고
3. 연산자를 클래스 멤버 함수로 작성할 수 있다.
4. 연산자를 프렌드 함수로 작성할 수 있다.
5. 다양한 이항 연산자를 중복 할 수 있다.
6. 다양한 단항 연산자를 중복 할 수 있다.
7. 단항 연산자에서 전위 연산자와 후위 연산자를 구분하여 작성할 수 있다.

친구란?

친구?

내 가족의 일원은 아니지만 내 가족과 동일한 권한을 가진 일원으로 인정받은 사람



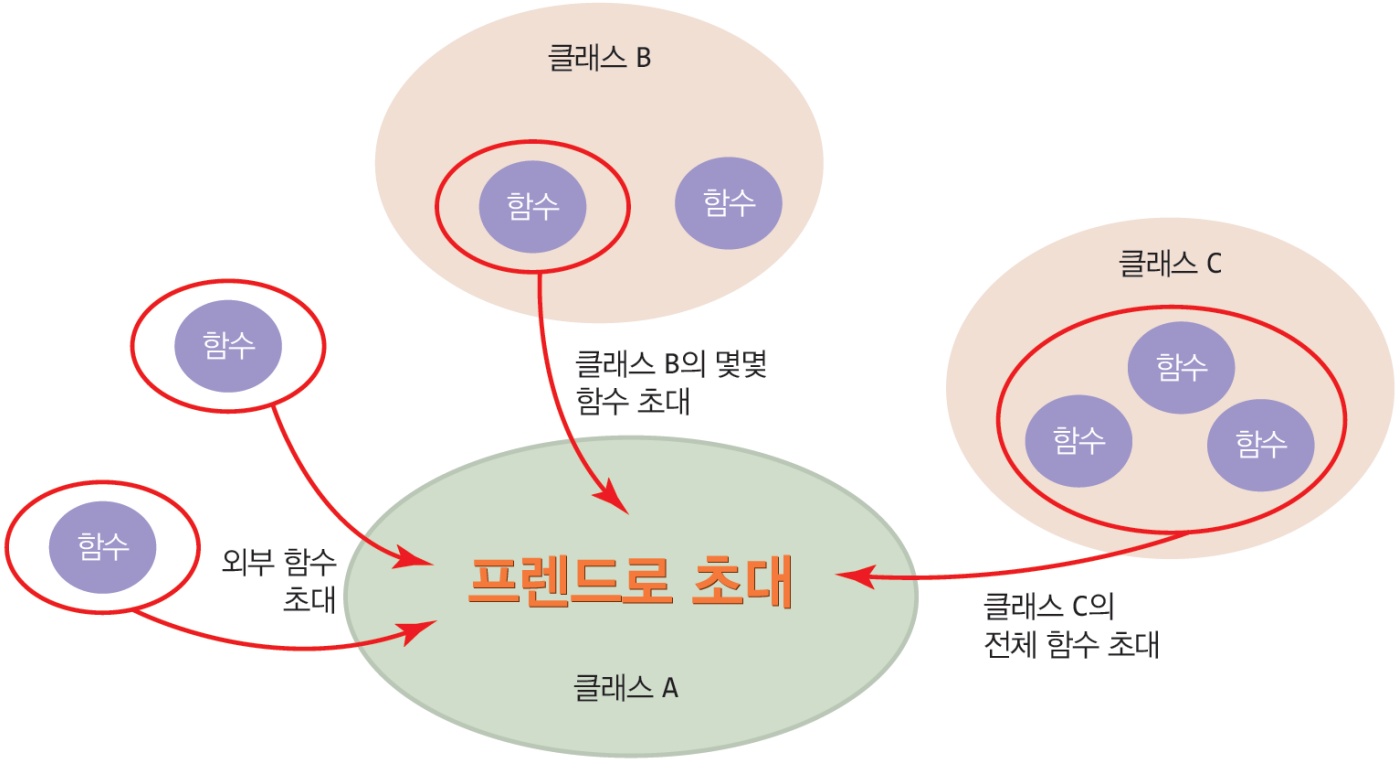
C++ 프렌드

- 프렌드 함수
  - 클래스의 멤버 함수가 아닌 외부 함수
    - 전역 함수
    - 다른 클래스의 멤버 함수
  - friend 키워드로 클래스 내에 선언된 함수
    - 클래스의 모든 멤버를 접근할 수 있는 권한 부여
    - 프렌드 함수라고 부름
  - 프렌드 선언의 필요성
    - 클래스의 멤버로 선언하기에는 무리가 있고, 클래스의 모든 멤버를 자유롭게 접근할 수 있는 일부 외부 함수 작성 시

항목	세상의 친구	프렌드 함수
존재	가족이 아님. 외부인	클래스 외부에 작성된 함수. 멤버가 아님
자격	가족의 구성원으로 인정받음. 가족의 모든 살림살이에 접근 허용	클래스의 멤버 자격 부여. 클래스의 모든 멤버에 대해 접근 가능
선언	친구라고 소개	클래스 내에 `friend` 키워드로 선언
개수	친구의 명수에 제한 없음	프렌드 함수 개수에 제한 없음

프렌드로 초대하는 3 가지 유형

- 프렌드 함수가 되는 3 가지
  - 전역 함수 : 클래스 외부에 선언된 전역 함수
  - 다른 클래스의 멤버 함수 : 다른 클래스의 특정 멤버 함수
  - 다른 클래스 전체 : 다른 클래스의 모든 멤버 함수



프렌드 선언 3 종류

1. 외부 함수 equals()를 Rect 클래스에 프렌드로 선언

```
class Rect { // Rect 클래스 선언
    ...
    friend bool equals(Rect r, Rect s);
};
```

2. RectManager 클래스의 equals() 멤버 함수를 Rect 클래스에 프렌드로 선언

```
class Rect {
    ..............
    friend bool RectManager::equals(Rect r, Rect s);
};
```

3. RectManager 클래스의 모든 멤버 함수를 Rect 클래스에 프렌드로 선언

```
class Rect {
    ..............
    friend RectManager;
};
```

예제 7-1 프렌드 함수 만들기


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

class Rect;
bool equals(Rect r, Rect s); // equals() 함수 선언

class Rect { // Rect 클래스 선언
   int width, height;
public:
   Rect(int width, int height)  {  this->width = width; this->height = height; }
   friend bool equals(Rect r, Rect s);
};

bool equals(Rect r, Rect s) { // 외부 함수
   if(r.width == s.width && r.height == s.height) return true;
   else return false;
}

int main() {
   Rect a(3,4), b(4,5);
   if(equals(a, b)) cout << "equal" << endl;
   else cout << "not equal" << endl;
}
```

Rect 클래스가 선언되기 전에 먼저 참조되는 컴파일 오류(forward reference)를 막기 위한 선언문(forward declaration)

equals() 함수를 프렌드로 선언

equals() 함수는 private 속성을 가진 width, height에 접근할 수 있다.

객체 a와 b는 동일한 크기의 사각형이므로 "not equal" 출력

```
not equal
```

예제 7-2 다른 클래스의 멤버 함수를 프렌드로 선언


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

class Rect;

class RectManager { // RectManager 클래스 선언
public:
    bool equals(Rect r, Rect s);
};

class Rect { // Rect 클래스 선언
    int width, height;
public:
    Rect(int width, int height) { this->width = width; this->height = height; }
    friend bool RectManager::equals(Rect r, Rect s);
};

bool RectManager::equals(Rect r, Rect s) {
    if(r.width == s.width && r.height == s.height) return true;
    else return false;
}

int main() {
    Rect a(3,4), b(3,4);
    RectManager man;

    if(man.equals(a, b)) cout << "equal" << endl;
    else cout << "not equal" << endl;
}
```

Rect 클래스가 선언되기 전에 먼저 참조되는 컴파일 오류(forward reference)를 막기 위한 선언문(forward declaration)

RectManager 클래스의 equals() 멤버를 프렌드로 선언

객체 a와 b는 동일한 크기의 사각형이므로 "equal" 출력

```
equal
```

예제 7-3 다른 클래스 전체를 프렌드로 선언


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

class Rect;

class RectManager { // RectManager 클래스 선언
public:
    bool equals(Rect r, Rect s);
    void copy(Rect& dest, Rect& src);
};

class Rect { // Rect 클래스 선언
    int width, height;
public:
    Rect(int width, int height)  { this->width = width; this->height = height; }
    friend RectManager;
};

bool RectManager::equals(Rect r, Rect s) { // r과 s가 같으면 true 리턴
    if(r.width == s.width && r.height == s.height) return true;
    else return false;
}

void RectManager::copy(Rect& dest, Rect& src) { // src를 dest에 복사
    dest.width = src.width;  dest.height = src.height;
}
```

Rect 클래스가 선언되기 전에 먼저 참조되는 컴파일 오류(forward reference)를 막기 위한 선언문(forward declaration)

RectManager 클래스를 프렌드 함수로 선언

```
int main() {
    Rect a(3,4), b(5,6);
    RectManager man;

    man.copy(b, a); // a를 b에 복사한다.
    if(man.equals(a, b)) cout << "equal" << endl;
    else cout << "not equal" << endl;
}
```

객체 b의 width, height 값이 a와 같아진다.

```
equal
```

man.copy(b,a)를 통해 객체 b와 a의 크기가 동일하므로 "equal" 출력