C++ Cast

HCI Programming 2 (321190)
2007년 가을학기
10/4/2007
박경신

Overview

- C++의 cast
  - static_cast
  - reinterpret_cast
  - const_cast
  - dynamic_cast
- 실시간 타입 정보 (Real-Time Type Information, RTTI)
- RTTI 사용 예
- cast

새로운 cast

- static_cast
  - 가장 일반적인 상황에서 사용할 수 있는 캐스팅
  - static_cast를 사용하기 위해서는 변환하려는 인자와 변환할 타입 사이에 납득할 만한 관계이어야 함 – 포인터끼리 바꾼다거나, enum이나 float를 int로 바꾸는 것 등등

```
int nVal;
short nShort = static_cast<short> (nVal);

void *pPtr;
int *pVal = static_cast<int *> (pPtr);
```


새로운 cast

- reinterpret_cast
  - 관계없는 속성으로 형을 바꿀 때 사용
  - int를 포인터로 바꾼다거나 그 반대의 경우에 사용
  - 가능하면 사용하지 않도록 함

```
int nVal;
Class CTest { … };
CTest c;
nVal = reinterpret_cast<int>(&c); // CTest 클래스의 포인터를 int형으로 변환
```

새로운 cast

- const_cast
  - 어떠 객체의 const 속성을 제거하는데 사용

```
const int * pConstVal;
int * pNotConstInt = const_cast<int *>(pConstVal);
```


새로운 cast

- dynamic_cast
  - C++에 RTTI (Real-Time Type Information) 기능을 사용해서 캐스팅이 가능하면 하고, 불가능하면 하지 않음
  - 포인터나 래퍼런스에 대해서만 사용가능
  - 캐스팅을 하지 못할 때 원하는 타입이 포인터였다면 캐스팅의 결과로 NULL 포인터를 반환하고, 원하는 타입이 레퍼런스였다면 bad_cast expection을 줌

실시간 타입 정보 (RTTI)

- 실시간 타입 정보 (Real-Time Type Information, RTTI)의 중요한 3가지 키워드
  - dynamic_cast – 동적 캐스팅. 반드시 polymophic class 에 사용해야 함. polymophic class가 아닌 클래스에 사용할 경우에는 컴파일 오류 생성.
  - typeid – 연산자. 클래스의 객체나 연산의 결과에 대해서 const type_info&를 반환
  - type_info – typeinfo.h에 선언되어 있는 클래스

type_info 클래스

```
class type_info {
public:
    virtual ~type_info(); // is polymorphic bool
    operator==(const type_info&) const; // can be compared
    bool operator!=(const type_info&) const;
    bool before(const type_info&) const; // ordering

    const char* name() const; // name of type

private:
    type_info(const type_info&); // prevent copying
    type_info& operator=(const type_info&); // prevent copying
// ...
};
```

RTTI 사용 예

```
#include <typeinfo>
#include <iostream>
using namespace std;
class Parent {
public:
    Parent() { cout << “Parent contructed." << endl; }
    virtual ~Parent() { cout << “Parent destructed." << endl; }
    virtual void Print() { cout << “I am Parent.” << endl; }
};
class Child: public Parent {
public:
    Child() { cout << “Child contructed." << endl; }
    ~Child() { cout << “Child destructed." << endl; }
    virtual void Print() { cout << “I am Child.” << endl; }
};
int main() {
    Child* pC = new Child;
    Parent* pP = pC;
    Child* pC1 = dynamic_cast<Child*>(pP);
    pP->Print();
    pC1->Print();
    if( typeid(*pP) == typeid(Child) )
        cout << "OK!" << endl;
    else
        cout << "No!" << endl;
    delete pP;
}
```

```
Parent constructed.
Child constructed.
I am Child.
I am Child.
OK!
Child destructed.
Parent destructed.
```

캐스트

- 업캐스트 (upcast)
  - 현재 클래스의 직접적인 부모로의 형변환을 의미
- 다운캐스트 (downcast)
  - 현재 클래스의 파생 클래스로 형변환하는 경우
  - 실제로 현재 클래스의 포인터가 가리키고 있는 것이 그것의 정확한 파생 클래스라면 dynamic_cast를 써서 형변환이 가능
  - 만약 현재 클래스의 포인터가 가리키고 있는 것이 파생 클래스가 아니라 원래 클래스라면 dynamic_cast가 실패
- 크로스캐스트 (crosscast)
  - 예를 들어 A를 계승한 B와 C를 계승한 D가 있고 B, D를 동시에 계승하는 E가 있는데, B의 포인터를 C의 포인터로 캐스팅하려는 경우 dynamic_cast를 사용하여 형변환이 가능