7/15/2013

Type Casting

Bugsquashing Seminar

Jasmin Metzger

dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

type-casting

- Converting an expression of a given type into another type
- Implicit vs. Explicit Conversion

Implicit Conversion

- Standard conversion
- Do not require any operator
- Automatically performed when a value is copied to a compatible type
- May imply a loss of precision
- Example:

```
short a = 2000;
int b;
b = a;
```

- Can also include constructor or operator conversions
- Example:

```
class A {};
class B { public: B (A a) {} };

A a;
B b = a;
```

Explixit Conversion

- Used when different interpretation of the value is imply

- Functional and c-like casting
  - Fundamental data types
  - Example:

```
short a = 2000;
int b;
b = (int) a;      // c-like cast notation
b = int (a);      // functional notation
```

  - Attention: these operators can be applied indiscriminately on classes and pointers to classes → code can be syntactically correct but causes runtime errors

Explixit Conversion

- Four specific casting operators:
  - const_cast
    - Only type conversion, which can manipulate the constness of an object
  - dynamic_cast
    - Can be used only with pointers
    - Cast class to a base class: always successful
    - Special checking when class is polymorphic→ during runtime (performance!)
  - reinterpret_cast
    - any pointer type to any other pointer type, even of unrelated classes
  - static_cast
    - conversions between pointers to related classes
    - also from a base class to its derived→ no safety check

const_cast & reinterpret_cast

- Example: const_cast<T>(*expression*)

```
void print (char * str)
{ cout << str << endl;}

int main () {
  const char * c = "sample text";
  print ( const_cast<char *> (c) );
  return 0;
}
```

- Example: reinterpret_cast<T>(*expression*)

```
class A {};
class B {};
A * a = new A;
B * b = reinterpret_cast<B*>(a);
```

dynamic_cast

- Example: dynamic_cast<T>(*expression*)

```
class CBase { virtual void dummy() {} };
class CDerived: public CBase { int a; };

int main () {
  try {
    CBase* pb new CBase;
    CBase* pba = new CDerived;
    CBase* pbb = new CBase;
    CDerived* pd;

    pb = dynamic_cast<CBase*>(pd); // ok: derived-to-base

    pd = dynamic_cast<CDerived*>(pba); //ok: pointing to a full object of CDerived

    pd = dynamic_cast<CDerived*>(pbb); //fails: pointing to an object of CBase
    if (pd==0) cout << "Null pointer on type-cast" << endl;

  } catch (exception& e) {cout << "Exception: " << e.what();}
  return 0;
}
```

static_cast

- Example: static_cast<T>(*expression*)

```
class CBase {};
class CDerived: public CBase {};
CBase * a = new CBase;
CDerived * b = static_cast<CDerived*>(a);
//valid, but b would point to an incomplete object
```

```
double d=3.14159265;
int i = static_cast<int>(d);
```

typeid

- Check the type of an expression
- Example:

```
#include <typeinfo>

class CBase { virtual void f(){} };
class CDerived : public CBase {};

int main () {
  try {
    CBase* a = new CBase;
    CBase* b = new CDerived;
    cout << "a is: " << typeid(a).name() << '\n';
    cout << "b is: " << typeid(b).name() << '\n';
    cout << "*a is: " << typeid(*a).name() << '\n';
    cout << "*b is: " << typeid(*b).name() << '\n';
  } catch (exception& e) { cout << "Exception: " << e.what() << endl; }
  return 0;
}
```

Ausgabe:

a is: class CBase *
b is: class CBase *
*a is: class Cbase
*b is: class CDerived

Have a look…

- http://www.cplusplus.com/doc/tutorial/typecasting/
- Scott Meyers. 2005. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition)*. Addison-Wesley Professional.