Föreläsning 7:
Specialization, Rational, Traits, Multiple inheritance mm

1 Specialisering

Se SpecDot.hpp

```
#include <vector>
#include <iterator>
using namespace std;


template <class IT>
typename iterator_traits<IT>::value_type DotProduct2(IT aBegin, IT aEnd
, IT bBegin, IT bEnd) {
    iterator_traits<IT>::value_type sum = iterator_traits<IT>::value_ty
pe();
    for (auto it = aBegin, it2 = bBegin; it != aEnd && it2 != bEnd; ++i
t, ++it2) {
        sum += *it * *it2;
    }
    return sum;

}

int DotProduct2(bool* aBegin, bool* aEnd, bool* bBegin, bool* bEnd) {
    int sum = 0;
    for (auto it = aBegin, it2 = bBegin; it != aEnd && it2 != bEnd; ++i
t, ++it2) {
        sum += *it * *it2;
    }
    return sum;

}

void TestDot() {
    vector<int> a = { 1, 2, 3, 4 };
    vector<int> b = { -1, 2, -3, 4 };
    //int dot = DotProduct(0, a, b);
    int dot2 = DotProduct2(a.begin(), a.end(), b.begin(), b.end());

    bool r[] = { 0, 1, 1, 0 };
    bool s[] = { 1, 1, 0, 1 };
    int bit2 = DotProduct2(r, r+4, s, s+4);
}
```

2 Rational

Lite allmänt om rationella tal
Något om att Rationella tal ska vara kompatibla med vanliga tal.
Samt behovet att räkna med fler bitar än man har i representationen:
Rational<char>: 127/2 + 125/2 blir 126/1 som kan representeras som ett Rational<char> men mellanresultaten kommer att bli större än vad som kan representeras i en char.
Se `NumberTrait<char>::NextSizeType representeras`
Se NumberTrait.hpp för hur detta kan göras:
`NumberTrait<char>::NextSizeType` representerar ett ”större” tal.

```
template <typename TNumber>
struct NumberTrait {
    static const int bits = 8 * sizeof(TNumber);
    typedef TNumber NextSizeType;
};

template<>
struct NumberTrait < char > {
    static const int bits=8;
    typedef short NextSizeType;
};

template<>
struct NumberTrait < int > {
    static const int bits = 32;
    typedef long long NextSizeType;
};

void Test() {
    NumberTrait<char>::NextSizeType sum = 0;
    NumberTrait<int>::NextSizeType sum2 = 0;
}
```

2.1 Traits

Allmänt prat om Traits.

3 Multipe inheritance

Se InterfaceDiamondExample.zip projektet
Interface,h definierar två interfact IDrawable och IUpdatable
Object.h definierar ett enkelt exempel på hur interfacen kan användas

Diamond.h tar upp problemet med att samma object kan förekomma två gånger i en objektgraf och då ofta bör ärvas virtuellt.

Något om hur det implementeras togs upp. (virtual pointers for virtual objects)

4 Casts

Alla fyra varianterna beskrivs

static_cast	för det som kan kontrolleras statiskt
dynamic_cast	för det som kräver runtime check
const_cast	för att ta bort const (sällsynt att det är en vettig idé)
reinterprete_cast	innebär att ”bitarna” tolkas på annat sätt (ytterligt sällsynt att det kan vara en vettig idé, ett exempel som är rimligt är en hashfunction som går på bitmönster).

Läs mer i boken.