Concepts i C++0x

Terje Slettebø
Adstate A/S

Presentasjonsoversikt

- Hvorfor ”concepts”?
- Hva er ”concepts”?
- Hva er problemet?
- Tidligere bidrag til løsning
- C++0x ”concepts” forslagene
- Hva går ”concepts” ut på og hvordan brukes de?
- Bibliografi
- Spørsmål

Hvorfor ”concepts”?

”Er ikke C++ komplisert nok som det er?”

Motiverende eksempel

```
int min(int, int);
```

Motiverende eksempel

```
int min(int, int);
double min(double, double);
MyBigNumber min(MyBigNumber, MyBigNumber);
...
```

Motiverende eksempel

Løsning (?): Templates:

```
template<typename T>
const T& min(const T&, const T&);
```

Motiverende eksempel

Men:

```
int *something=...;
int *something_else=...;
...
min(something, something_else); // Krasj med
    kryptisk feilmelding
```

Motiverende eksempel

Løsning!

```
template<LessThanComparable T>
const T& min(const T&, const T&);
```

Hva er ”concepts”?

- ”Concepts” har blitt beskrevet som et ”type-system for typer”.
- Et ”concept” kan også ses som ett ”sett av typer”, dvs. flere typer kan matche samme ”concept”.

Hva er problemet?

”Concepts” fyller et ”hull” i C++s typesystem:

- Vanlige templates kan ikke typesjekkes før instantiering, og det samme gjelder kode som bruker templates.
- Kryptiske feilmeldinger, og det kan ikke minst være et problem for libraries.
- Vanlige templates kan ikke spesifisere krav til typer og verdier som brukes for instantiering; det blir enten konkrete typer (som ”int”), eller ”anything goes” (”class T”).
- Man kan ikke overload’e templates basert på typers egenskaper (f.eks. std::advance).

Tidligere bidrag til løsning

- Boost Concept Check Library
- Concept Traits Library

Konklusjon: Det trengs støtte i selve språket for å kunne utnytte ”concepts” muligheter.

C++ concepts forslagene

- Problemet har vært kjent lenge (siden tidlig C++).
- 2003: Første ”formal proposal” til standardkommiteen om støtte for ”concept” i C++0x av Bjarne Stroustrup og Gabriel Dos Reis.
- 2005: Lillehammer-konferansen: To forslag finnes:
  - Stroustrup/Dos Reis
  - ”The Indiana Team”

  Opphavspersonene bes av kommiteen å komme frem til et felles forslag.

C++ concepts forslagene

- 2007: Et forent forslag finnes (og har fantes siden kort etter Lillehammer).
- En partiell implementasjon av forslaget finnes: ConceptGCC.

Hva går ”concepts” ut på og hvordan brukes de?

Concept definisjon:

```
auto concept LessThanComparable<typename T>
{
   bool operator<(T, T);
}
```

”auto” – implisitt match for typer

Merk at concepts bruker ”pseudo-signaturer”.

Definisjon av concepts, fortsatt

Multi-type concepts:

```
auto concept Convertible<typename T, typename U>
{
    operator U(const T&);
}
template<typename T, typename U>
where Convertible<T, U>
U convert(const T& t)
{
    return t;
}
```

Concept komposisjon

```
concept InputIterator<typename Iter>
{
    where Regular<Iter>;

    typename value_type;
    ...
}
```

Concept refinement

```
concept ForwardIterator<typename Iter, Typename Value> :
    InputIterator<Iter, Value>
{
}
```

Concept map

```
concept_map LessThanComparable<MyType>
{
    bool operator<(const T& x, const T& y)
    {
      return x.my_less_than(y);
    }
}
```

Concept map

Mer realistisk eksempel:

```
concept_map InputIterator<char *>
{
    typedef char value_type;
    typedef char& reference;
    typedef char* pointer;
    typedef std::ptrdiff_t difference_type;
}
```

concept_map kan erstatte bruk av ”traits”.

Concept map

Concept maps kan også være parameterisert:

```
template<typename T>
concept_map InputIterator<T *>
{
    typedef T value_type;
    typedef T& reference;
    typedef T* pointer;
    typedef std::ptrdiff_t difference_type;
}
```

Concept overloading

Automatisk valg av mest effektiv implementasjon:

```
template<EqualityComparable T>
class dictionary
{
    // Langsom lenket liste implementasjon
}

template<LessThanComparable T>
class dictionary
{
    // Balansert tre implementasjon
}

template<Hashable T>
class dictionary
{
    // Hash table implementasjon
}
```

Axioms

```
concept SemiGroup<typename Op, typename T>
{
    axiom Associativity(Op op, Tx, Ty, Tz)
    {
        op(x, op(y, z)) == op(op(x, y), z);
    }
}
```

Mål for concepts

- Enklere programmering
- Ytelse
- Støtte ”Generic Programming”
- Bakoverkompatibilitet
- Forståelig kompileringsmodell (skal ikke kreve doktorgrad for å forstå og bruke...)

Bibliografi

- Concepts (revision 1): http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2081.pdf m.m.
- ”Generic Programming and the STL” - Matt Austern

Spørsmål?