Advanced Programming Techniques (AdvPT)

Winter Term 2016/17

Sebastian Kuckuk and Martin Bauer
Chair for System Simulation

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Assignment 2

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Function resolving

- Short summary
- More in-depth resources available online, e.g.

  http://en.cppreference.com/w/cpp/language/overload_resolution

  http://en.cppreference.com/w/cpp/language/virtual

  http://en.cppreference.com/w/cpp/language/template_argument_deduction

Function resolving

1. Gather *viable functions*
   1. Gather all functions in the current scope that have the same name as the function called (-> *candidate functions*)
   2. Filter all functions with a non-matching number of parameters
2. Check the number of functions
   1. 0 -> compiler error
   2. 1 -> call that function
   3. >1 -> select best match (see next slide); if there is no best match the compiler will report on an ambiguous function call -> compiler error

Function resolving

- Best match
  - Each parameter type is matched against the types passed in the call
  - In decreasing order of ‘goodness’:
    - An exact match (e.g. double -> double)
    - A promotion (e.g. float -> double)
    - A standard type conversion (e.g. int -> float)
    - A constructor or user-defined type conversion (e.g. int -> class A)
- Choosing a winner
  - Candidates are as strong as their weakest match
  - Candidates with an equivalent number and type of weakest match are compared on their next-weakest (and so on)

Function resolving

- Member functions
  - Candidate functions that are member functions are treated as if they had an extra parameter which represents the object for which they are called; it appears before the first of the actual parameters
  - ⇒ Functions discarding qualifiers (e.g. const functions called on a non-const object) don’t count as valid candidate functions
  - Static functions are handled just as non-static functions for the overload resolution

Function resolving

- Template functions
  - Works basically just like ‘regular’ functions
  - Template arguments are deduces automatically if not provided explicitly
  - In case of ambiguity, the most specialized version is chosen
  - If there is more than one ‘most specialized’ version -> compiler error

The End

Questions ?

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT