

Lambda Expressions and Capture Continued Solutions

Capture by Reference

- What effect does capture by reference have?
 - It allows the body of the lambda expression to modify the captured variable
- What syntax is used?
 - To capture a variable by reference, we put an & before it
- Write a simple program which demonstrates capture by reference

Lambda capture implementation

- How is capture by reference implemented?
 - A lambda with capture by reference is implemented as a "functor with state"
 - The member variable is a reference
 - The captured variable is passed to the functor constructor by reference
 - The operator () can then modify the captured variable through the reference

Equivalent code with functor

- Rewrite the program in the first slide to use a functor

Implicit Capture

- What is meant by implicit capture?
 - In an implicit capture, all variables in scope are captured by the lambda expression
- What syntax is used for implicit capture?
 - [=] for implicit capture by value
 - [&] for implicit capture by reference

Implicit Capture Contd

- Is implicit capture safe?
 - Implicit capture by value is safe
 - Implicit capture by reference allows the lambda to change any variable in scope. This can be confusing and is potentially dangerous, so should be avoided
- Can implicit capture be made safer?
 - We can exclude some variables from implicit capture
 - [=, &x] will capture x by reference, all others by value. Only x can be modified by the lambda
 - [&, =a, =b] will capture a and b by value, all others by reference. a and b cannot be modified by the lambda

Lambda Functions and Member Functions

- If a lambda expression is defined inside a member function, can it access other members of the object?
 - Yes. This can be done by capturing "this"
 - The lambda will have access to all the data members and member functions of the object
- Write a simple program in which a lambda expression is defined inside a member function and modifies a data member of the class

Capturing "this" Continued

- If your compiler supports C++17, rewrite your last solution to use capture by value
- Why does this not compile?
 - The lambda body has a const copy of the object
- Make the code compile and run. Explain your results
 - The code will compile if the lambda is declared mutable
 - However, the lambda's body uses a copy of the object. Any changes to this copy will not affect the data member of the original object