Polymorphism in Inheritance

Idea

- Use base class as framework for derived classes usage.
- Define member functions with virtual keyword.
- Override base class functions with new implementations in derived classes.
- If virtual keyword not used, overloading won't occur.

Polymorphism comes from the fact that you could call the based method of an object belonging to any class that derived from it, without knowing which class the object belonged to.

Inheritance

Example (Matrix Base Class)

```
class matrix {
   protected:
      int rows, cols;
      double *elements;
   public:
      matrix(int r, int c);
      ~matrix();
      int get_rows();
      int get_cols();
      virtual void fill(double value);
};
```

Inheritance

Example (Square Matrix Derived Class)

```
class squarematrix :  public matrix {
   private:
   protected:
   public:
      squarematrix(int r, int c) :  matrix(r,c) {
         if(r!=c) std::cerr<<"not a square matrix"; exit(1);
      }
      double trace();
      void fill(double value) {
         for (int i=0; i < rows*cols; i++)
            elements[i] = value;
      }
};
```

Inheritance

Example (non-virtual)

```
squarematrix Q(5,5);
Q.fill(1.6);
std::cout<<" Trace = "<<Q.trace();
```

Example (virtual)

```
matrix *Q;
Q = new squarematrix(5,5);
Q->fill(1.6);
std::cout<<" Trace = "<<Q->trace();
```

SciNet compute • calcul CANADA

Inheritance

Gotcha:

- Virtual functions are run-time determined
- Equivalent cost to a pointer dereference
- Not as efficient as compile time determined (ie non-virtual)
- Should be avoided for many use functions

Gotcha:

- Friend keyword allows non-member functions access to private data.
- Useful but does break OOP and will cause problems in inheritance.
- "friends are your enemies"

SciNet compute • calcul CANADA

HANDS-ON:

Modify your derived and base class using the virtual keyword.