Misc. Questions

Reminder

- Speaking of exams
  - The date for the Final has been decided:
  - Saturday, November 16th
  - 8am – 10am
  - 01-2000

Announcement

- If you would like a paper copy of Volume 2 of the text, please let me know via e-mail by Wednesday.
  - Cost will be $25.

Before we begin

- Project Notes
  - Quota problems:
    - Use `rm-junk` to clean up your accounts
  - Update design
  - `try` now fixed
  - Clock problem due Oct 16.

Before we begin

- Project Notes
  - About the Parking Lot Problem
    - Cars can only go forward or reverse
    - Cars cannot turn
    - Cars cannot go diagonal
    - Cars cannot move sideways
    - No square cars!

Before we begin

- Misc. Questions
  - Initializing pointers
  - Bad input
  - Virtual operators

Initializing Pointers

- This is okay

```
int a (7);
```

- Evidently, this is not (on Windows)

```
int * a (new int [20]);
```

- However, in initializer list, it's okay

```
Foo::Foo (int size) : intArray (new int[size])
 {}
```

Bad input

- What to do when getting input from cin?

```
float f
cin >> f;
```

- What happens if "Foo" is inputted
  - istream overloads the ! operator
  - istream::operator! will return true if
    - EOF is reached
    - Problem with stream
    - Data formatting error

Bad input

- What to do when getting input from cin?

```
float f
cin >> f;
if (!cin) {
  cerr << "Bad input";
   ...
}
```

Bad input

- Note that istream also defines a method that will convert a stream to a pointer iff it is okay and 0 otherwise.

```
while (cin) { … }
```

Virtual operators

- Q: Can operators be declared as virtual?
  - What if your solver wants to compare 2 abstract Configurations?
  - Make a call to:
    - `Configuration::operator< (const Configuration &C)`
  - This will be overridden by derived classes of Configuration

Virtual operators

- In this case
  - Configuration's operator< must be declared as `virtual`.

```
class Configuration {
public:
      virtual bool operator< (const
                           Configuration &C)
      …
}
```

Virtual operators

- Then..
  - Derived class's signature must be exactly the same:

```
class ClockConfig : public Configuration {
public:
    bool operator< (const Configuration &C)
      …
}
```

Virtual operators

- Problems with this:

```
FarmerConfig F(…);
ClockConfig C(…);

if (C < F) // what does THIS
  mean?
```

Virtual operators

```
/**
* See if this configuration should be ordered
* before another. What that means is up to each
* derived class.
* @pre other is the same type as this.
* @return true iff this should be ordered before
*      other
*/
bool operator<( const Configuration &other );
```

Virtual operators

```
bool ClockConfig::operator<( const Configuration &other
   )
{
   // must cast other to a ClockConfig
   // we know we can if precondition is met
   // should probably check using runtime type checking
   const ClockConfig &CC = (const ClockConfig &)other;

   // compare members with CC, not other
   if (myData < CC.myData) return true;
   ...
}
```

Virtual operators

- Questions

Virtual operators

- Note: operator= cannot be declared virtual

```
class Configuration {
public:
     Configuration & operator= (const
                        Configuration &C)
     …
}

class ClockConfig : public Configuration {
public:
     ClockConfig & operator= (const
                        ClockConfig &C)
     …
}
```

Virtual operators

- ClockConfig's assignment should explicitly call Configuration's assignment

```
const ClockConfig & ClockConfig::operator= (const
  ClockConfig &C)
       {
             // copy ClockConfig data
             ...

             // copy Config data
             Config::operator= (C);

            return (*this);
      }
```

Questions?

A little bit more on testing

- White Box Testing
  - Assure that all code is executed and tested
  - statement coverage:
    - suite of tests executes each statement at least once
  - decision coverage:
    - suite of tests ensures each if/loop/case decision goes every way it possibly can
  - condition coverage:
    - suite of tests ensures that each combination of Boolean outcomes from a single decision is tested

A little bit more on testing

- Software tools
  - Rational Rose includes
    - Purify - detects memory management problems
    - Pure Coverage - statement coverage

A little bit more on testing

- One solution to testing List class
  - Posted on Web site

- Questions?