Today’s lecture

• Inheritance and overloading in C++
Previous Lecture

- Basic features of C++
  - Adds formal **class** concept to C, making it object-oriented
  - **Class** is like a derived type except
    - It is better encapsulated
    - It has “**methods**”
    - Invoking a method is like sending a message to the object, object contains its own logic saying what to do.
  - E.g the String class
    
    String s1;
    s1.set(“Hello”);
    printf(“%s\n”, s1.s());
Inheritance

Want new class uString. Like String except that the strings will be converted and stored in upper case.

e.g.

```
String s;
s.set(“Hello”);
printf(“%s
”,s.s());
⇒ Hello
```

```
uString s;
s.set(“Hello”);
printf(“%s
”,s.s());
⇒ HELLO
```
uString extends String

- No need to write uString from scratch.
- Inherit most code from String.
- Extend String::set to capitalise.
- A uString is a String with some extra feature.

```
String
  set()
  s()
```

Base class

```
uString
```

Derived class
C++ Inheritance Example

• New interface for uString

/* Extend String class to uString */
/* uString stores strings as upper case */
class uString : public String {
    public:
        void set( char *);    /* Set a uString */
};
uString set method

/* Set str to point to a private copy of s */
void uString::set(char *s) {
    int i;
    String::set(s);
    for (i=0;i<strlen(s);++i) {
        if ( str[i] >= 'a' && str[i] <= 'z' ) {
            str[i] = toupper(str[i]);
        }
    }
}

Base class method

“protected”
(not “private”)
main()
{
    String s1;
    uString s2;

    printf("Executable code starting\n");

    s1.set("Hello");
    printf("%s\n",s1.s());
    s2.set("Hello");
    printf("%s\n",s2.s());

    printf("Executable code ending\n");
}
Overloading

Can redefine operators e.g. + to operate on classes e.g.

```cpp
coord p1, p2, p3;
p3 = p1 + p2

This would then do
→ if p1=p2=(1,1,1)  p3 = (2,2,2)
```
Overloading

Have to define the meaning of + and = for a `coord` class object. Language defines meaning for integer, float, double etc but now we can define extra meanings.

class coord{
    public:
        coord operator+(coord );
    private:
        int cx; int cy; int cz;
};

coord coord::operator+ (coord c2)
{
    coord temp;
    temp.cx = cx + c2.cx;
    temp.cy = cy + c2.cy;
    temp.cz = cz + c2.cz;
    return(temp);
}
Homework

• Due Oct 27\textsuperscript{th}