CS601: Software Development for Scientific Computing

Autumn 2024

Week11: Intermediate C++, Structured Grids

this

- Implicit variable defined by the compiler for every class
  - E.g. `MyVec *this;`
- All member functions have `this` as an implicit first argument
  - E.g.

```
int MyVec::GetVecLen() const;
```

would actually be:

```
int MyVec::GetVecLen(MyVec* this) const;
```

Overloading +=

- `MyVec v1;`
  `v1+=3;`
- `MyVec& MyVec::operator+=(double)`

Overloading +=

- `MyVec v1;`

  `v1+=3;`
  - `MyVec& MyVec::operator+=(double)`
- `MyVec v2;`

  `v2+=v1;`
  - `MyVec& MyVec::operator+=(const MyVec& rhs)`
  - What if you make the return value above `const`?

    Disallow: `(v2+=v1)+=3;`

Overloading +

- `v1=v1+3;`

  Single-argument constructors: *allow implicit conversion from a particular type to initialize an object.*

  – `const MyVec MyVec::operator+(double val)`
- `v3=v1+v2;`

  1. `const MyVec MyVec::operator+(const MyVec& vec2) const;`

  OR

  2. `friend const MyVec operator+(const MyVec& lhs, const MyVec& rhs);`

v1=3+v1 is compiler error! Why?

Operator Overloading - Guidelines

- If a binary operator accepts operands of different types and is commutative, both orders should be overloaded
- Consistency:
  - If a class has ==, it should also have !=
  - += and + should result in identical values
  - define your copy assignment operator if you have defined a copy constructor

Refer to demo example

Class Templates

```
//CS601: example code used to show class declaration.
#ifndef MYVEC_H
#define MYVEC_H
class MyVec{
        //private attributes
        double* data;
        int vecLen;
public:
        MyVec(){data=0;vecLen=0;}
```

What if user wants to have a MyVec class with integer `data`?

Class Templates

- Like function templates but for templating classes

Refer to demo example

Standard Template Library (STL)

- Large set of frequently used data structures and algorithms
  - Defined as *parametrized* data types and functions
  - Types to represent complex numbers and strings, algorithms to sort, get random numbers etc.
- Convenient and bug free to use these libraries
- E.g. `vector, map, queue, pair, sort` etc.
- Use your own type only for efficiency considerations - *only if you are sure!*