Assignment #1 Discussions

C++ Object Oriented Programming

Pei-yih Ting

94/04 NTOU CS

Design Problems

1. Should readFile(), writeFile(), or print() be a member function? Why or why not?

   Should you pass ostream & as the parameter to print()?
2. Do you think add() or subtract() should return an object? Will it be better if they don't have any side effects?
3. Many functions seem to be not extremely necessary, ex. negative(), clone(), clear(), especially for the required test program? Why should we write something that is not necessary? Should a member function always be of some use?
4. Many statements repeat themselves many times in the program? Have you tried to summarize them into a common function? What are the benefits?

Design Problems

5. Did you see the point that we are trying to make the CFraction class as independent as possible to the test program, i.e. main()? Why are we doing this?
6. What is the advantage to separate the CFraction as an independent module from the "*debug*" point of view? from the "*reuse*" point of view?

   Try separate CFraction.cpp and CFraction.h from your project and put them into another project.

✶ What come to your mind when you think of your CFraction class? A useful object with its versatile functions or a lot of variables and vector template?

Design Problems

7. Did you notice any differences between const function and normal member function? Why should I declare it as const? It is const anyway. I wrote it therefore I know that it does not change anything in the object. Why bother declare it as const? -- enforcement of the encapsulation by suitable interface –
8. Should gcd() be a *public* member function? *private* member function? or a *static file scope* function?
9. When did you start testing your class? after all member functions are written? or right after one function is written? Do not remove codes in your unitTest() function after test finished!!

Technical Problems

1. Differenciate pointers:

```
vector<int> intVector;
vector<int> *ptrToVector;
vector<int *> iPtrVector;
vector<int>::iterator iterIntVector;
vector<int *>::iterator iterIPtrVector;
int *iPtr;
```

2. Reading/writing binary files
   - Binary stream concept
   - Unformatted I/O concept
   - Efficiency and Precision
3. Where should my variable be declared?
   local … member variable … global

Technical Problems

4. No pointer is the best programming practice !???!
   a. Java legend?
   b. Why pointers?
      i. Indirection make your program flexible and powerful. Pointers is source of polymorphism in C. It save space and time.
      ii. In industry, space and time == money! You keep your job if you can meet the requirements under the constraint of budgets.
      iii. System software cares about space and time extremely.
   c. Why no pointers? You are afraid of strange astray bugs. You don't have the correct semantics, i.e. syntax-model mapping
   d. Why can't you detect pointer errors in you previous program assignments? You did not learn well and you did not build safety nets.

How do we find out overflow?

Let's use a C macro
Function call won't work

```
#include <iostream>
using namespace std;

#define printOverflow(x)  _asm{MOV    x, 0 }\
                          _asm{JNO    $+13 }\
                          _asm{MOV    x, 1 }

void main()
{
     int a, b, c, of=-1;

     cout << "before any operation     of=" << of << endl;
     a = 12345678;
     b = 12345678;
     c = a*b;
     printOverflow(of);
     cout << "after 12345678*12345678  of=" << of << " c=" << c << endl;
```

How do we find out overflow?

```
     a = 1;
     c = a*b;
     printOverflow(of);
     cout << "after 1*12345678         of=" << of << " c=" << c << endl;

     a = -1;
     b = -2;
     c = a + b;
     printOverflow(of);
     cout << "after -1 + (-2)          of=" << of << " c=" << c << endl;

     a = -1234567;
     b = 2345678;
     c = a * b;
     printOverflow(of);
     cout << "after -1234567 * 2345678 of=" << of << " c=" << c << endl;
}
```