C++ Practice

프로그래밍 방법론

2007/09/19

Review

- Use the keyword, “class”, for object types.
  - ex) class CTable{…};
- Access control
  - public, private, protected
  - Interfaces are usually public
  - Internal member variables or functions are usually private
- Don’t forget ‘;’at the end of a class declaration.

Class Example

time.h

```
class CTime {
      private:
            int m_iMinute;
      public:
            CTime();
            CTime(int minute);
            ~CTime();
            int GetHour();
            int GetMinute();
            int GetSecond();
};
```

time.cpp

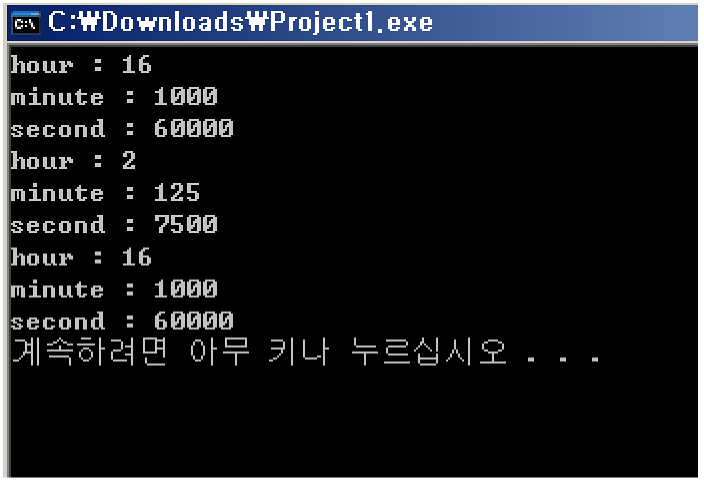
```
#include "time.h"

CTime::CTime() { m_iMinute = 1000; }

CTime::CTime( int minute ) { m_iMinute = minute;}

CTime::~CTime() {}

int CTime::GetHour() { return m_iMinute/60; }

int CTime::GetMinute() { return m_iMinute; }

int CTime::GetSecond() { return m_iMinute*60; }
```

Class Example

main.cpp

```
#include <cstdlib>
#include <iostream>

#include "time.h"

using namespace std;

int main(int argc, char *argv[])
{
    CTime a;
    cout << "hour : "   << a.GetHour() << endl;
    cout << "minute : " << a.GetMinute() << endl;
    cout << "second : " << a.GetSecond() << endl;

    CTime b(125);
    cout << "hour : "   << b.GetHour() << endl;
    cout << "minute : " << b.GetMinute() << endl;
    cout << "second : " << b.GetSecond() << endl;

    CTime* c = new CTime;
    cout << "hour : "   << c->GetHour() << endl;
    cout << "minute : " << c->GetMinute() << endl;
    cout << "second : " << c->GetSecond() << endl;
    delete c;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

result



```
C:₩Downloads₩Project1.exe
hour : 16
minute : 1000
second : 60000
hour : 2
minute : 125
second : 7500
hour : 16
minute : 1000
second : 60000
계속하려면 아무 키나 누르십시오 . . .
```

Overloading

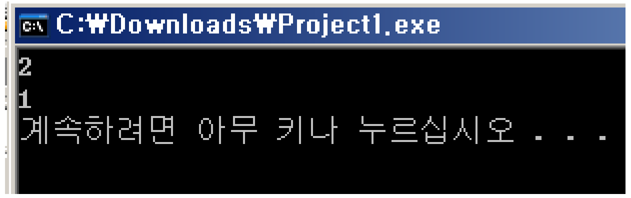
- Most programming languages (C in particular) require a unique identifier for each function
- Function overloading with different argument types
- Operators also can be overloaded in C++

Function overloading

main.cpp

```
#include <cstdlib>
#include <iostream>

using namespace std;

void test(int i) { cout << "1" << endl; }

void test(double i) { cout << "2" << endl; };

int main(int argc, char *argv[])
{
    test(1.0);
    test(1);

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

result



C:₩Downloads₩Project1.exe

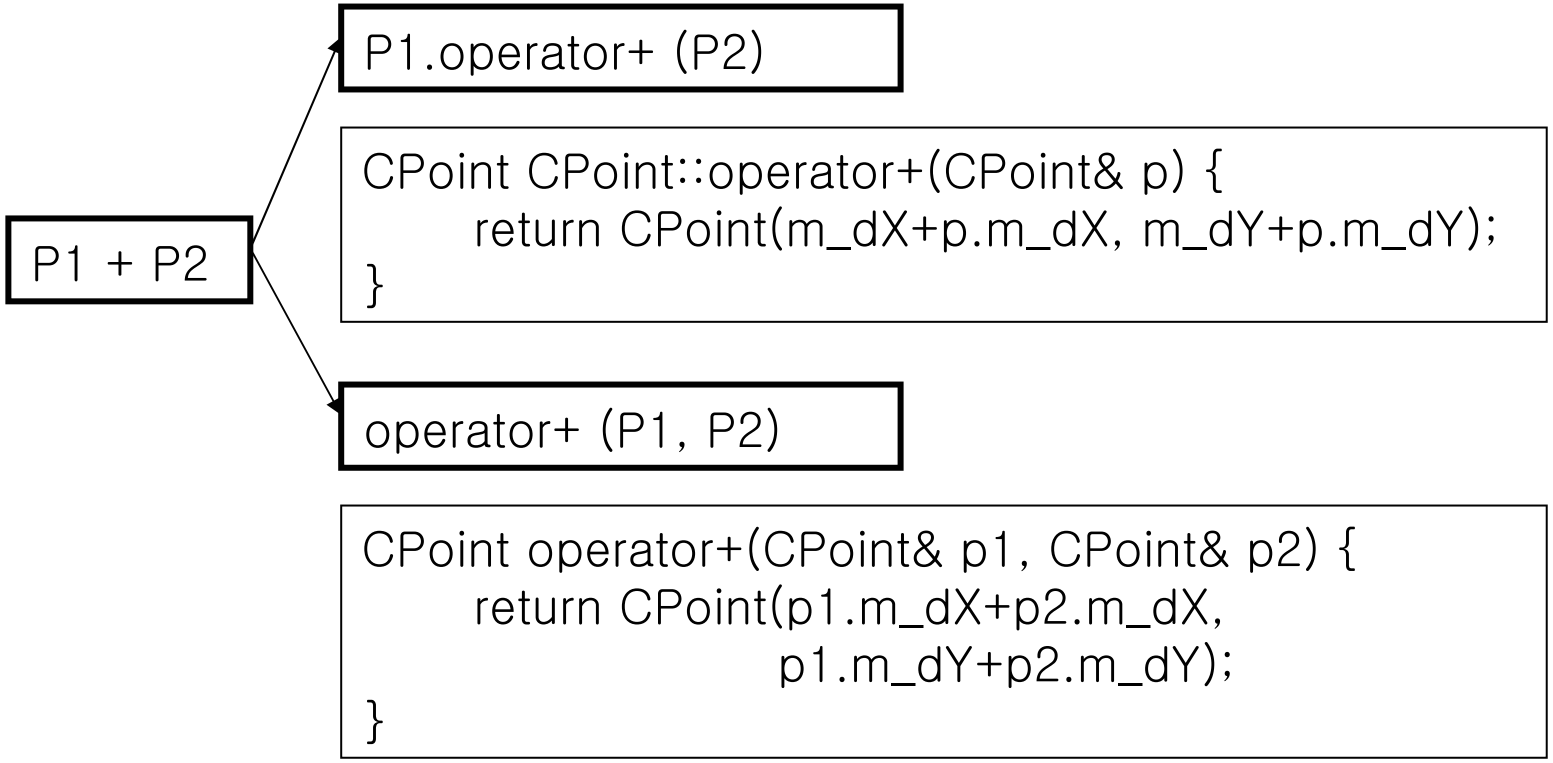
```
2
1
계속하려면 아무 키나 누르십시오 . . .
```

Operator Overloading

- Operator overloading
  - Define behavior of an operator when applied to a class
  - E.g, +/-applies to built-in numerical types
  - When we define our Point class, representing an (x, y) point
    - can overload +/-to behave appropriately
    - (2,3) + (5,5) = (7,8) (calculation of vectors)

Operator Overloading

- Two types



```
CPoint CPoint::operator+(CPoint& p) {
        return CPoint(m_dX+p.m_dX, m_dY+p.m_dY);
}
```

```
CPoint operator+(CPoint& p1, CPoint& p2) {
        return CPoint(p1.m_dX+p2.m_dX,
                      p1.m_dY+p2.m_dY);
}
```

Type 1

point.h

```
class CPoint {
      private:
              double m_dX, m_dY;
      public:
             CPoint() {}
             CPoint(double x, double y) { m_dX=x; m_dY=y; }
             CPoint operator+(CPoint& that);
             CPoint operator-(CPoint& that);
             double GetX();
             double GetY();
};
```

Type 1

point.cpp

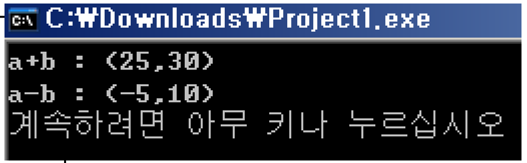
```
#include “point.h"

CPoint CPoint::operator+(CPoint& that) {
	return CPoint(m_dX+that.m_dX, m_dY+that.m_dY);
}

CPoint CPoint::operator-(CPoint& that) {
	CPoint temp;
	temp.m_dX = this->m_dX - that.m_dX;
	temp.m_dY = (*this).m_dY - that.m_dY;
	return temp;
}

double CPoint::GetX() { return m_dX; }
double CPoint::GetY() { return m_dY; }
```

Type 1

main.cpp

```
#include <cstdlib>
#include <iostream>

#include " point.h"

using namespace std;

int main(int argc, char *argv[])
{
    CPoint a(10, 20);
    CPoint b(15, 10);

    CPoint c = a+b;
    CPoint d = a-b;

    cout << "a+b : "   << "(" << c.GetX() << "," << c.GetY() << ")" << endl;
    cout << "a-b : "   << "(" << d.GetX() << "," << d.GetY() << ")" << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

result



C:₩Downloads₩Project1.exe

```
a+b : (25,30)
a-b : (-5,10)
계속하려면 아무 키나 누르십시오
```

Type 2

point.h

```
class CPoint {
	private:
		 double m_dX, m_dY;
	public:
		CPoint() {}
		CPoint(double x, double y) { m_dX=x; m_dY=y; }
		friend CPoint operator+(CPoint& p1, CPoint& p2);
		friend CPoint operator-(CPoint& p1, CPoint& p2);
		double GetX();
		double GetY();
};
```

Type 2

point.cpp

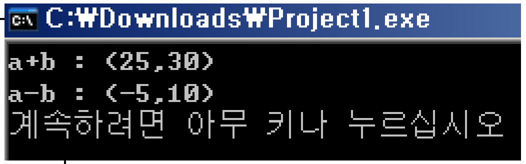
```
#include "point.h"

double CPoint::GetX() { return m_dX; }
double CPoint::GetY() { return m_dY; }
```

Type 2

main.cpp

```
#include <cstdlib>
#include <iostream>

#include "point.h"

using namespace std;

CPoint operator+(CPoint& p1, CPoint& p2) {
      return CPoint(p1.m_dX+p2.m_dX, p1.m_dY+p2.m_dY);
}

CPoint operator-(CPoint& p1, CPoint& p2) {
      return CPoint(p1.m_dX-p2.m_dX, p1.m_dY-p2.m_dY);
}

int main(int argc, char *argv[])
{
   CPoint a(10, 20);
   CPoint b(15, 10);

   CPoint c = a+b;
   CPoint d = a-b;

   cout << "a+b : "   << "(" << c.GetX() << "," << c.GetY() << ")" << endl;
   cout << "a-b : "   << "(" << d.GetX() << "," << d.GetY() << ")" << endl;

   system("PAUSE");
   return EXIT_SUCCESS;
}
```

result



C:₩Downloads₩Project1.exe

```
a+b : (25,30)
a-b : (-5,10)
계속하려면 아무 키나 누르십시오
```

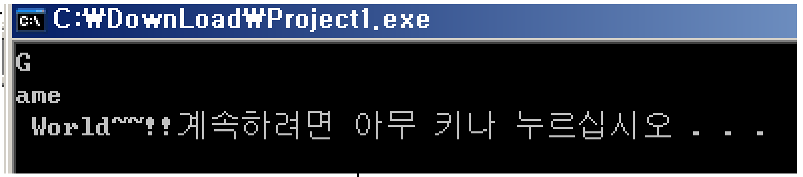
Difference between ‘overloading’ and ‘overriding’

- overloading
  - different functions with same name
  - ex)
    - int calculate(int a);
    - int calculate(double a);

- overriding
  - redefinition of member function in a child class

Bonus : ifstream

```
#include <cstdlib>
#include <iostream>
#include <fstream>

using namespace std;

int main(int argc, char *argv[])
{
    ifstream fin;
    fin.open("test.txt");

    char ch;
    char buf[100];

    fin >> ch; cout << ch << endl;
    fin >> buf; cout << buf << endl;

    fin.getline(buf, 100);
    cout << buf;

    fin.close();
    system("PAUSE");
    return EXIT_SUCCESS;
}
```



```
C:₩DownLoad₩Project1.exe
G
ame
 World~~!!계속하려면 아무 키나 누르십시오 . . .
```