实验 2　类与对象的深入讨论

班级 信安 20-2　　学号 20101110201　　姓名 李天昊

[实验目的]

1、进一步加深对类和对象的理解。
2、掌握类的构造函数和析构函数的概念和使用方法。
3、掌握对象的数组、对象的指针及其使用方法。
4、掌握定义静态数据成员和静态成员函数的方法及使用。
5、掌握友元的概念和使用。

[实验要求]

给出以下各实验内容的源程序代码，并把编译、运行过程中出现的问题以及解决方法填入实验报告中，按时上交。

[实验学时] 2 学时。

[实验内容]

1、对象数组及对象指针应用。

动态建立一个对象数组，内放 5 个学生的数据(学号、1 门成绩)。利用指针，统计平均成绩并输出第 1，3，5 个学生的数据。

- exp2-1.cpp

```
#include <iostream>
#include <string>

using namespace std;

class CStudent {
private:
        string m_id;
        double m_score;
        static double sm_total_score;
        static size_t sm_total_number;
public:
        CStudent() = default;
        CStudent(const CStudent &s);
        ~CStudent();

        inline void set_value(string id, double score);
        inline string get_id();
```

```
-           inline double get_score();
-           inline static double get_average();
-       };
-
-       double CStudent::sm_total_score = 0;
-       size_t CStudent::sm_total_number = 0;
-
-      CStudent::CStudent(const CStudent &s): m_id(s.m_id), m_score(s.m_score)
{
-           ++sm_total_number;
-           sm_total_score += s.m_score;
-       }
-       CStudent::~CStudent() {
-           --sm_total_number;
-           sm_total_score -= m_score;
-       }
-
-       inline void CStudent::set_value(string id, double score) {
-           m_id = id;
-           m_score = score;
-           ++sm_total_number;
-           sm_total_score += m_score;
-       }
-
-       inline string CStudent::get_id() {
-           return m_id;
-       }
-
-       inline double CStudent::get_score() {
-           return m_score;
-       }
-
-       inline double CStudent::get_average() {
-           return sm_total_score / sm_total_number;
-       }
-
-       int main() {
-           const int N = 5;
```

```
-       CStudent *stu = new CStudent[N];
-       for (int i = 0; i < N; ++i) {
-           string id;
-           double score;
-           cin >> id >> score;
-           (stu + i)->set_value(id, score);
-       }
-       for (int i = 0; i < N; i += 2) {
-           cout << " > The score of student " << (stu + i)->get_id() << " is "
<< (stu + i)->get_score() << ". " << endl;
-       }
-       cout << " > The average score is " << CStudent::get_average() << ". "
<< endl;
-       delete [] stu;
-       return 0;
-
```

- 编译成功，测试通过：

```
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ g++ exp2-1.cpp -o main
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ ./main
101 89.5
102 77.1
103 98.1
104 69.3
105 84.9
 > The score of student 101 is 89.5.
 > The score of student 103 is 98.1.
 > The score of student 105 is 84.9.
 > The average score is 83.78.
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$
```

2、用静态数据成员和静态成员函数设计程序。

商店销售某一商品，当天公布统一的折扣(discount)，商品价格为 22.5 元。现已知 3 个销售员销售情况为：

销售员号(num)	销货件数(quantity)
101	5
102	12
103	100

请编程序，输入当天折扣，计算出当日此商品的总销售款 sum 及每件商品的实际售价。

提示：将折扣 discount、总销售款 sum 和商品销售总件数 n 声明为静态数据

成员，定义静态成员函数 average()求平均售价，定义 display()函数输出结果。

- exp2-2.cpp

```
#include <iostream>

using namespace std;

class CSale {
private:
	string num;
	size_t quantity;

	static double sm_discount;
	static double sm_sum;
	static size_t sm_total_quantity;
	static double sm_avg_price;
	const double PRICE = 22.5;

public:
	CSale() = default;
	~CSale();

	inline static void set_discount(double dis);
	inline void set_sales_data(string n, size_t q);
	static void Average();
	inline static void Display();

};

double CSale::sm_discount = 1.0;
double CSale::sm_sum = 0.0;
size_t CSale::sm_total_quantity = 0;
double CSale::sm_avg_price = 0.0;

CSale::~CSale() {
	--sm_total_quantity;
	sm_sum -= quantity * PRICE * sm_discount;
```

```
-   }
-
-   inline void CSale::set_discount(double dis) {
-       sm_discount = dis;
-   }
-
-   inline void CSale::set_sales_data(string n, size_t q) {
-       num = n;
-       quantity = q;
-       sm_total_quantity += q;
-       sm_sum += quantity * PRICE * sm_discount;
-   }
-
-   void CSale::Average() {
-       sm_avg_price = sm_sum / sm_total_quantity;
-   }
-
-   inline void CSale::Display() {
-       Average();
-       cout << "The actual price is " << sm_avg_price << ". " << endl;
-       cout << "The total sale revenue is " << sm_sum << ". " << endl;
-   }
-
-   int main() {
-       CSale s[3];
-       cout << " > Set Discount: ";
-       double d;
-       cin >> d;
-       CSale::set_discount(d);
-       for (size_t i = 0; i != sizeof(s) / sizeof(CSale); ++i) {
-           cout << " > Set Num and Quantity: " << endl;
-           string num;
-           size_t quan;
-           cin >> num >> quan;
-           s[i].set_sales_data(num, quan);
-       }
-       CSale::Display();
-       return 0;
```

- }
- 编译成功，测试通过：

```
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ g++ exp2-2.cpp -o main
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ ./main
 > Set Discount: 0.95
 > Set Num and Quantity:
101 18
 > Set Num and Quantity:
102 15
 > Set Num and Quantity:
103 20
The actual price is 21.375.
The total sale revenue is 1132.88.
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$
```

3、设计程序。

已知点类 Point，包括两个数据成员：x(横坐标)，y(纵坐标)；若干成员函数。其中计算两点间距离的函数分别采用以下两种方法设计：

① 将 pdistance( )作为 Point 类的成员函数。

② 将 cdistance( )作为 Point 类的友元函数。(cdistance( )是类外的一个普通函数)

注：以点(0，0)和(3，4)作为测试数据，求出它们之间的距离。

- exp2-3.cpp

```
- #define NDEBUG
- #include <iostream>
- #include <cmath>
-
- using namespace std;
-
- class CPoint {
- private:
-     double m_x;
-     double m_y;
- public:
-     CPoint() = default;
-     CPoint(double x, double y);
-     ~CPoint();
-
-     double pdistance(const CPoint &p);
-     friend double cdistance(const CPoint &lhs, const CPoint &rhs);
```

```
-     };
-
-     CPoint::CPoint(double x, double y): m_x(x), m_y(y) {    }
-     CPoint::~CPoint() {    }
-
-     double CPoint::pdistance(const CPoint &p) {
-          return sqrt(pow((m_x - p.m_x), 2) + pow((m_y - p.m_y), 2));
-     }
-
-     double cdistance(const CPoint &lhs, const CPoint &rhs) {
-          return sqrt(pow((lhs.m_x - rhs.m_x), 2) + pow((lhs.m_y -
rhs.m_y), 2));
-     }
-
-     int main() {
-
-          #ifndef NDEBUG
-          CPoint p1(0, 0);
-          CPoint p2(3, 4);
-          if ((p1.pdistance(p2) - cdistance(p1, p2) < 1e-6) &&
p1.pdistance(p2) == 5) cout << "[ TEST OK ] distance = " <<
p1.pdistance(p2) << endl;
-          else cout << "[ TEST FAIL ]" << endl;
-          return 0;
-          #endif
-
-          double x1, y1;
-          cout << "Enter x1 y1: ";
-          cin >> x1 >> y1;
-          cout << endl;
-          CPoint p1(x1, y1);
-
-          double x2, y2;
-          cout << "Enter x2 y2: ";
-          cin >> x2 >> y2;
-          cout << endl;
-          CPoint p2(x2, y2);
-
```

```
-           cout << "Distance between two points: " << p1.pdistance(p2) <<
endl;
-
-           return 0;
-   }
```

- 编译成功，测试通过：

```
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ g++ exp2-3.cpp -o main
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ ./main
[ TEST OK ] distance = 5
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$
```

```
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ g++ exp2-3.cpp -o main
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$ ./main
Enter x1 y1: 3.08 7.33

Enter x2 y2: 1.19 32.87

Distance between two points: 25.6098
david@ThinkpadT480s:/mnt/d/#Programming/CPP/NCUT/EXP$
```

[实验总结] 给出对本次实验的总结。

1、加深了对类和对象的理解，能够较为熟练地使用类和对象。
2、学会了类的构造函数和析构函数的基本概念和使用方法。
3、掌握对象的数组、对象的指针及其使用方法。
4、掌握定义静态数据成员和静态成员函数的方法及使用。
5、掌握友元的概念和使用。