C++常见编程错误

一　过程式编程

1．名字空间

1)

```
namespace foo {
    void f() {/* f's body */    }
    int x;
}
using namespace foo::f() ;
```

2.

```
#include <iostream>
using std;
using std::ostream;
using std::cout;
void f(){
   int x;
   using std::cin>>x;
}
```

3. 输入输出流运算符<< 和 >>

1）.　std::cin　<< x, y ;

2)

4.　变量及类型、作用域

1)

```
f( ){
        a = 25.6;
        c = a + b;
}
```

2）

```
int a;
string s;
a = s;
```

3)

```
string str;
for ( int i = 0 ; i < 10; i++) {
        cin>>str;
        if(str=="Exit") break;
}
cout<<" break on :" << i << '\n';
```

5. 函数

1）

```
void main(){
    int a = 3, b = 4;
    swap( a , b);
}
void swap(int &x, int &y){
    int t = a ; a = b;    b = t;
}
```

2) //有 2 处错误

```
int add(int a, int b){
     return A+b;
}
int main(){
    std::cout<<add(34.5, 56.5) ;
    std::cout<<add(34.5) ;
    return 0;
}
```

3) //有 2 处错误

```
#include <iostream>
float triangle_area(a, b, c){
   float s = (a+b+c)/2;
   float area = sqrt( s*(s-a)* (s-b) *(s-c) );
}
int main(){
        float x = x, y=4, z = 5;
        std::cout<<"the aresa of triangle with edges ("<<a<<" " <<b<<" "<<c<<") is:"
                << triangle_area(x,y,z)<<'\n';
        return 0;
}
```

4) inline 应用于声明而不是定义!

```
void    swap(int &x, int &y);
void main(){
        int a = 3, b = 4;
        swap( a , b);
    }
inline void    swap(int &x, int &y){
        int t = a ; a = b;    b = t;
}
```

5)

```
int f();
void main(){
```

```
        f(6);
    }
    int f() { return 3; }
```

6）

```
    int f( int a = 9, double b){
        return 0;
    }
```

7)

```
    void print ( int , int ) ;
    void main ( ) {
       print( );
    }
    void print ( int int a = 6, int int b = 8 ) {
        //...
    }
```

8)

```
     int f ( int );
     double f( int ) ;
     void main ( ) {
          int i , j ;                  double d;
          d = f( i );
     }
```

6. std::string

1)

```
      string s1, s2 = "Hello";
      char s3[30] = "char array";
      s1 += s2;
      s1 += "World";
  s3 += s1;
```

7. new 和 delete

1)

```
    int *ip = new int [10];
    //...
    delete ip ;
```

2)

```
    int *ip;
    cin >> *ip;
    cout<<ip;
    cout *ip;
```

8. 引用和 const （常量）

1)

```
int b = 3;
int &a;
int &c = 5;
c = b;
```

2)

```
void fun ( int &x ) { x++; }
void g() {
    int a;
    cin>>a;
    fun( &a );
}
```

3)

```
void fun (coinst int &x ) { x++; }
void g() {
    int a;
    const int b;
    cin>>a;
    cin>>b;
    fun( a );
    fun( b );
}
```

9. 变量作用域

1） 发现并改正下列程序的错误

```
int factorial (int n) {
        int i = 1;
        while (n > 1) {
                i = i * n;
                int n = n - 1;
        }
        return i;
}

int main (int argc, char *argv[]) {
        int fac4 = factorial(4);
        int fac5 = factorial(5);
        printf("4! = %d, 5! = %d\n", fac4, fac5);
        return 0;
}
```

10 传值和传引用

```
#include <stdio.h>
```

```
void square(int num) {
        num = num * num;
}

int main() {
        int x = 4;
        square(x);
        printf("%d\n", x);
        return 0;
}
```

不修改 square 函数的返回值类型，通过传引用方式，得到并输出正确的平方数。

二 类

1. 有 2 个错误

```
C c1, c2;
class C {
};
```

2.

```
class C{
 public:
     C(int ) { }
};
void main ( ){
    C c;
}
```

3. 私有或保护成员只能在类的成员函数或 f 友元(friend) 函数中被访问; （非静态的）类成员必须通过具体对象才能访问.

```
class   C {
    int a;
  public:
    m ( ) { / *f's body */}
};
void main(){
    C c;
    c. m ( ) ;
    c.a;
    m ( );
}
```

4.

```
class C{
  public:
        inline void f() ;
};
inline void C::f( ) { /*    f' s body */ }
```

5.

```
class C {
  public:
        void C( ) {    }
        void ~C* {      }
        C ( int ) {        }
          ~C ( int ) {      }
};
```

6.

```
class C {
public:
     C ( C c) {    }
     C ( C c, int a) {        }
};
```

7.

```
class C {
   const int c;
public:
   C( ) { c = 0; }
};
```

8.

```
class C {
public:
   void f ( )    {        }
};
void fun( ) {
        C c;
        C* p;
        p = &c;
        p.f( );
        c.f( ) ;
}
void g( C& c) {
     c->f( );
}
```

9. 类的静态成员仅仅在类中声明，还必须在类体外定义！如

```
class C {
public:
   int n;
   static int c; //类 C 的静态数据成员 c 的声明
};
int C::c ;   // //类 C 的静态数据成员 c 的定义，不需要再加关键字 static
int main () {
      C c;
      cin>>c.n;
      cout<<c.n<<"\n";;
      cin>>C::c;
      cout<<C::c <<'\n';
      rerturn 0;
}
```

10. 类的静态成员仅存在于在整个类中而不是一个具体的类对象中。而非静态的成员对于每个具体的对象都有一个单独的拷贝！

```
   class C {
   public:
         void f() { /* f's body    */    }
         void g() { /*g's body    */    }
   } ;
   int main( ) {
      C c;
      c.f();
      c.g();
      C::f();
      C::g() ;
      return 0;
   }
```

三 继承

1. 如果派生类的成员函数与基类的成员函数同名，则称派生类的该成员函数覆盖了基类的同名函数。也就是说通过派生类的对象将不能再调用基类的这个函数了。

```
class B{
  public:
      void f( double ) { /* f's body */}
};
class D: public B{
```

```
  public:
       void f( char ) { /* f's body */}
};
int main(){
     D d;
     d.f( 3.14 );
}
```

2. 如果基类没有默认构造函数，则派生类的构造函数必须显式地在其初始化成员列表中调用基类构造函数，以便对继承下来的基类成员进行初始化。

```
class B{
       int x;
  public:
       B( int a ) { x = a; }
};
class D: public B{
    double y;
  public:
       void D( int a ) { y = a; }
};
int main(){
     D d (3);
     return 0;
}
```

3. 私有成员只能被该类的成员函数或该类的友元访问，即使派生类也无权访问基类的私有数据！

```
    class B{
       int x;
    public:
       B( int a    = 0) { x = a; }
       int get_x()    { return x;}
    };
    class D: public B{
       double y;
      public:
         void D( int a = 9 ):B(a) { }
         int get_x() { return x;}
  };
```