**1．本例实现用了4个文件AppCrypt.cpp、Crypter.cpp、Crypter.h和MyRandom.h。（文件个数可随意）**

//=====================================================================

***// Name : AppCrypt.cpp***

// Author : CW

// Version :

// Copyright : Your copyright notice

// Description : Hello World in C++, Ansi-style

//====================================================================

#include <iostream>

using namespace std;

***#include "Crypter.h"***

int main()

{

//初始化字节数组

const int len = 200;

unsigned char data[len];

for(int i=0;i<len;i++)

data[i] = i;

//1)

cout<<"1)"<<endl;

cout<<"加密前：";

Display(data,len);

cout<<"加密后：";

Coder(data,len,123456789);

Display(data,len);

cout<<"解密后：";

Coder(data,len,123456789);

Display(data,len);

cout<<endl;

//2)

cout <<"2)" << endl;

TRandom myRand;

cout<<"加密前：";

Display(data,len);

cout<<"加密后：";

Coder(data,len,myRand,7777777);

Display(data,len);

cout<<"解密后：";

Coder(data,len,myRand,7777777);

Display(data,len);

//3)

//结果没变。但两次调用Coder时，函数体内的rand是不同的对象，只是二者的种子是一样的。

//4）

//不行。Coder函数体内需要修改rand的数据成员(seed)。

//5)

cout <<"5)" << endl;

Crypter machine;

machine.SetKey(88888888);

cout<<"加密前：";

Display(data,len);

cout<<"加密后：";

machine.Encrypt(data,len);

Display(data,len);

cout<<"解密后：";

machine.Decrypt(data,len);

Display(data,len);

return 0;

}

/\*

***\* MyRandom.h***

\* Author: Administrator

\*/

#ifndef MYRANDOM\_H\_

#define MYRANDOM\_H\_

#include <limits.h> //声明INT\_MAX和ULONG\_MAX常量

#include <windows.h> //声明GetTickCount()函数,其返回从开机到现在经过的毫秒数

//#include <stdlib.h> //声明rand和srand函数

//#include <time.h> //声明time函数

class TRandom

{

public:

//缺省使用系统时间(开机后经过的毫秒数)为种子

TRandom (long seed=0) { mSeed=(seed?seed: GetTickCount()); }

//也可以定义自己的种子

void Seed(long seed=0) { mSeed=(seed?seed: GetTickCount( )); }

//取得一个随机的整数

int Integer() { return Next();}

//取得一个在指定范围内的随机整数

int Integer(int min,int max) { return min+Next()%(max-min+1);}

//取得一个随机的（0到1之间的）小数

double Real() {return double(Next())/double(INT\_MAX);}

private:

//使用调和算法

void Change() {mSeed=(314159265\*mSeed+13579)%ULONG\_MAX;}

//取得伪随机数发生器的随机数序列中的下一个随机整数

int Next()

{

int loops=mSeed%3;

for (int i=0;i<=loops;i++)

Change ();

return int(mSeed/2);

}

unsigned long mSeed; //随机数发生器的种子

};

#endif /\* MYRANDOM\_H\_ \*/

/\*

***\* Crypter.h***

\* Author: Administrator

\*/

#ifndef CRYPTER\_H\_

#define CRYPTER\_H\_

#***include "MyRandom.h"***

void Coder(unsigned char data[],int len,unsigned long key);

void Coder(unsigned char data[],int len,TRandom& rand , unsigned long key);

void Display(unsigned char data[],int len);

***class Crypter***

{

public:

void SetKey(unsigned long key)

{

myKey = key;

}

void Encrypt(unsigned char data[],int len)

{

rand.Seed(myKey);

for(int i=0;i<len;i++)

data[i] ^= rand.Integer(0,255);

}

void Decrypt(unsigned char data[],int len)

{

Encrypt(data,len);

}

private:

TRandom rand;

unsigned long myKey;

};

#endif /\* CRYPTER\_H\_ \*/

/\*

***\* Crypter.cpp***

\* Author: Administrator

\*/

#include <iostream>

using namespace std;

***#include "Crypter.h"***

***#include "MyRandom.h"***

void Coder(unsigned char data[],int len,unsigned long key)

{

~~static~~ TRandom staticRand; //非静态也可以吗？

staticRand.Seed(key); //设置随机数发生器的种子

for(int i=0;i<len;i++)

data[i] ^= staticRand.Integer(0,255); //通过异或运算加密

}

void Coder(unsigned char data[],int len,TRandom& rand , unsigned long key)

{

rand.Seed(key); //设置随机数发生器的种子

for(int i=0;i<len;i++)

data[i] ^= rand.Integer(0,255); //通过异或运算加密

}

void Display(unsigned char data[],int len)

{

for(int i=0;i<len;i++)

cout<<(int)(data[i])<<" ";

cout<<endl;

}

**2.本例用了两个文件： Pages.cpp和Book.h, 由于使用了内联实现，所以可以没有Book.cpp.**

//=====================================================================

// Name  ***: Pages.cpp***

// Author : CW

// Version :

// Copyright : Your copyright notice

//=====================================================================

#include <iostream>

using namespace std;

#include "Book.h"

int main()

{

Book book(100);

cout<<"总页数："<<book.GetPages()<<endl;

cout<<"阅读了25页."<<endl;

book.ReadPages(25);

cout<<"剩余未读"<<book.GetUnreadPages()<<endl;

cout<<"又阅读了10页."<<endl;

book.ReadPages(10);

cout<<"总计已阅读:"<<book.GetReadPages()<<endl;

cout<<"剩余未读"<<book.GetUnreadPages()<<endl;

cout << "Over" << endl;

return 0;

}

/\*

***\* Book.h***

\* Author: Administrator

\*/

#ifndef BOOK\_H\_

#define BOOK\_H\_

class Book

{

public:

Book(int pgs) : nTotalPages(pgs),nReadPages(0),nLeftPages(pgs) { }

int GetPages() const { return nTotalPages;}

int GetReadPages() const { return nReadPages;}

int GetUnreadPages() const { return nLeftPages; }

void ReadPages(int pgs) { nReadPages += pgs; nLeftPages -= pgs; }

private:

int nTotalPages;

int nReadPages;

int nLeftPages;

};

#endif /\* BOOK\_H\_ \*/

**3.本例用了三个文件Poker.cpp和Card.h，Card.cpp**

//=====================================================================

// Name : ***Poker.cpp***

// Author : CW

// Version :

// Copyright : Your copyright notice

// Description : Hello World in C++, Ansi-style

//=====================================================================

#include <iostream>

using namespace std;

#include "Card.h"

int main()

{

//Card类的成员较多，本例没有全部使用。

//创建草花3，和红心A，草花A

Card card1(1), card2(38),card3(12);

//或 Card card1(CLUB,THREE),card2(HEART,ACE),card3(CLUB,QUEEN);

cout<<"card1为："<<SuitNames[card1.GetSuit()]<<RankNames[card1.GetRank()]<<endl;

cout<<"card2为："<<SuitNames[card2.GetSuit()]<<RankNames[card2.GetRank()]<<endl;

cout<<"card3为："<<SuitNames[card3.GetSuit()]<<RankNames[card3.GetRank()]<<endl;

cout<<"card1与card2的花色"<<(card1.IsSameSuit(card2)?"相同":"不同")<<endl;

cout<<"card2与card3的等级"<<(card2.IsSameRank(card3)?"相同":"不同")<<endl;

cout << "其他..." << endl; // prints First

return 0;

}

/\*

***\* Card.h:***

\*

\* Author: Administrator

\*/

#ifndef CARD\_H\_

#define CARD\_H\_

//也可使用枚举类(这里使用的是普通枚举类型)

enum SUIT {CLUB=0,DIAMOND,HEART,SPADE};//"草花","方片","红心","黑桃"}

enum RANK {TWO=0,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE,TEN,

JACK,QUEEN,KING,ACE };

extern const char \* SuitNames[]; //= {"草花","方片","红心","黑桃"};

extern const char \* RankNames[]; //= {"2","3","4","5","6","7","8","9","10","J","Q","K","A"};

//上述两个声明

class Card

{

public:

static void SetBackImage(int img){ imgBack=img;}

static void SetTrump(int tsuit,int trank) //设定将牌(有时也称主牌)的花色与大小

{trumpsuit=tsuit;trumprank=trank;}

public:

Card(int nid):id(nid%5~~4~~+1),suit((id-1)/**13**),rank((id-1)%**13**) //构造函数，注意“/”和“%”

{}

Card(SUIT s,Rank r):id(**13\*s+r+1**),suit(s),rank(r) {}

**Card( const Card& rhs)**

:id(rhs.id),suit(rhs.suit),rank(rhs.rank),xpos(rhs.xpos),ypos(rhs.ypos)

//id、suit、rank必须用初始化列表初始化，xpos,ypos是否用初始化列表初始化可选。

{ }

SUIT GetSuit() **const** { return (SUIT)suit;}

RANK GetRank() const { return (RANK)rank;}

bool IsSameSuit(**const** Card& one) { return (suit == one.suit);}

bool IsSameRank(const Card& one) { return (rank == one.rank);}

bool IsSuit(SUIT s) { return (suit == s);}

bool IsRank(RANK r) { return (rank == r);}

bool **operator <** (const Card& rhs) **const**

{ return CaculateValue()<rhs.CaculateValue();}

private:

**//私有的计算分值的函数，这里假设为**

**// 将牌 > 其他牌**

**// 将牌ACE > 将牌KING > ...> 将牌TWO > 其他牌**

**// 其他同花色： ACE > KING > QUEEN > ... > TWO**

int CaculateValue() const

{

if (rank == trumprank)

return rank+**200**;

else if (suit == trumpsuit)

return (rank+**100**);

else

return rank;

}

private:

static int imgBack;

static int trumpsuit;//将牌花色

static int trumprank;//将牌值

**const** int id;

**const** int suit; // 或者 const SUIT suit;

**const** int rank; // 或者 const RANK rank;

**//below are other user data, for example:**

int xpos;

int ypos;

};

#endif /\* CARD\_H\_ \*/

/\*

***\* Card.cpp***

\*Author: Administrator

\*/

#include "Card.h"

//为类变量(类的静态数据成员)分配存储空间

int Card::imgBack = 0; //背面图案

int Card::trumpsuit = SPADE;//将牌花色

int Card::trumprank = TWO;//将牌值

//定义全局常量

const char \* SuitNames[] = {"草花","方片","红心","黑桃"};

const char \* RankNames[] = {"2","3","4","5","6","7","8","9","10","J","Q","K","A"};

**4.本例用了3个文件，AppHero.cpp， Hero.cpp, hero.h**

/\*

***\* Hero.h***

\* Author: Administrator

\*/

#ifndef HERO\_H\_

#define HERO\_H\_

**const** int ABLITTYCOUNT =5; //5种属性值

enum ABLITY {CHARM = 0,REPUTE,ATTACK,DEFENSE,POWER };

enum GOODS {NONE=0,G1,G2,G3,G4,G5,G6}; //6种宝物

**const** int BAGCOUNT = 5; //5个宝物袋

class Hero

{

public:

Hero(int cha,int rep,int att,int def,int pow);

void AddGood(int bagID,GOODS goods);

void RemoveGood(int bagID);

int CurAblity(ABLITY which) const { return curAblities[which]; }

private:

void RecaculateAblities();

private:

//魅力值、声望值、攻击力、防御力、法力

int rawAblities[ABLITTYCOUNT];

int curAblities[ABLITTYCOUNT];

//宝物袋

GOODS bags[BAGCOUNT];

};

#endif /\* HERO\_H\_ \*/

/\*

\* ***Hero.cpp***

\* Author: Administrator

\*/

#include "Hero.h"

Hero::Hero(int cha,int rep,int att,int def,int pow)

{

rawAblities[CHARM] =cha;

rawAblities[REPUTE] =rep;

rawAblities[ATTACK] =att;

rawAblities[DEFENSE]=def;

rawAblities[POWER] =pow;

//各宝物袋初始为空

for(int i=0;i<ABLITTYCOUNT;i++)

bags[i] = NONE;

//各属性初始时，没有宝物加入效果

for(int i=CHARM;i<=POWER;i++)

curAblities[i] = rawAblities[i];

}

void Hero::AddGood(int bagID,GOODS goods)

{

if( bags[bagID] == NONE)

{

bags[bagID] = goods;

RecaculateAblities();

}

}

void Hero::RemoveGood(int bagID)

{

if( bags[bagID] != NONE)

{

bags[bagID] = NONE;

RecaculateAblities();

}

}

void Hero::RecaculateAblities( )

{

//重置能力

for(int i=CHARM;i<=POWER;i++)

curAblities[i] = rawAblities[i];

//1提升魅力2点，2提升声望3点，3提升攻击力1点，...

for(int i=0;i<BAGCOUNT;++i)

{

switch (bags[i])

{

case NONE:

break;

case G1 :

curAblities[CHARM] += 1;

break;

case G2 :

curAblities[REPUTE] += 3;

break;

case G3 :

curAblities[ATTACK] += 1;

break;

case G4 :

//可自己定义

break;

case G5 :

//可自己定义

break;

default:

break;

}

}

}

//=====================================================================

// Name : ***AppHero.cpp***

// Author : CW

// Version :

// Copyright : Your copyright notice

// Description : Hello World in C++, Ansi-style

//====================================================================

#include <iostream>

using namespace std;

#include "Hero.h"

int main()

{

Hero hero(10,20,30,40,50);

cout<<"魅力="<<hero.CurAblity(CHARM)<<endl;

cout<<"声望="<<hero.CurAblity(REPUTE)<<endl;

cout<<"攻击="<<hero.CurAblity(ATTACK)<<endl;

cout<<"防御="<<hero.CurAblity(DEFENSE)<<endl;

cout<<"法力="<<hero.CurAblity(POWER)<<endl;

hero.AddGood(2,G1);

hero.AddGood(3,G3);

cout<<endl<<"增加宝物后："<<endl;

cout<<"魅力="<<hero.CurAblity(CHARM)<<endl;

cout<<"声望="<<hero.CurAblity(REPUTE)<<endl;

cout<<"攻击="<<hero.CurAblity(ATTACK)<<endl;

cout<<"防御="<<hero.CurAblity(DEFENSE)<<endl;

cout<<"法力="<<hero.CurAblity(POWER)<<endl;

hero.RemoveGood(2);

cout<<endl<<"扔掉宝物后："<<endl;

cout<<"魅力="<<hero.CurAblity(CHARM)<<endl;

cout<<"声望="<<hero.CurAblity(REPUTE)<<endl;

cout<<"攻击="<<hero.CurAblity(ATTACK)<<endl;

cout<<"防御="<<hero.CurAblity(DEFENSE)<<endl;

cout<<"法力="<<hero.CurAblity(POWER)<<endl;

cout << "....." << endl;

return 0;

}

**5.单件问题示例**

/\*

\* ***Demo.h***

\*/

#ifndef DEMO\_H\_

#define DEMO\_H\_

class Demo

{

**public**:

**static** Demo\* getDemo() //必须是静态成员函数，只能通过此静态函数创建Demo类的实例

{

if(instance == nullptr)

instance = new Demo;

return instance;

}

**static** void releaseDemo() {

delete instance;

instance = nullptr;

}

public:

void OneFunction(int value) { mNum+=value;}

int Value() const { return mNum; }

**private:**

**static** Demo \* instance; //唯一实例的指针

Demo(int n=0):mNum(n){} //私有化构造函数

Demo(const Demo&); //私有化拷贝构造函数，并只给声明不给定义。

//目的是防止类似调用： Demo two=(Demo::GetOne(1));

Demo& operator= (const Demo&); //私有化赋值函数，并只给声明不给定义。此函数可

//写可不写。如果出现自赋值呢，可能需要这个禁止

//如：\*one=\*two,不写这个赋值函数就允许，写了就不

//许

private:

int mNum;

};

#endif

/\*

\* ***AppMain.cpp***

\*/

#include <iostream.h>

#include "Demo.h"

// 初始化instance

**Demo \* Demo::instance = nullptr;**

int main()

{

Demo \* one = Demo::getDemo();

one->OneFunction(200);

cout<<"Value="<<one->Value()<<endl;

Demo\* two = Demo::getDemo();

One->OneFunction(500);

cout<<"Value="<<two->Value()<<endl;

**//\*one=\*two;//自赋值测试**

Demo::releaseDemo();

return 0;

}

**6.怪物战斗游戏**

/\*

***\* Monster.h***

\* Author: chenwei

\*/

#ifndef MONSTER\_H\_

#define MONSTER\_H\_

class Monster

{

public:

Monster(int spd, int hp, int dam, int def);//速度、生命值、攻击力值、防御力值

bool Fight(Monster& other);

private:

int Attacked(Monster& other) const;

bool PriorTo(const Monster& other) const;

//GetID()函数设置成静态或非静态的 均可；

//目的是取得不重复的ID。

static int GetID()

{

static int theID = 0;

return ++theID;

}

private:

const int id;//号

int speed;//速度

int hitpoint;//生命值

int damage;//攻击力值

int defense;//防御力值

};

#endif /\* MONSTER\_H\_ \*/

/\*

\* ***Monster.cpp***

\* Author: chenwei

\*/

#include "Monster.h"

Monster::Monster(int spd, int hit, int dam, int def) :

id(GetID()),speed(spd), hitpoint(hit), damage(dam), defense(def)

{

}

bool Monster::Fight(Monster& other)

{

if (PriorTo(other))

if (Attacked(other) == 0)

return true;

while (true)

{

if (other.Attacked(\*this) == 0)

return false;

if (Attacked(other) == 0)

return true;

}

}

int Monster::Attacked(Monster& other) const //攻击对方一次，返回对方生命值

{

int harm = damage\*2-other.defense;

if (harm < 1)

harm = 1;

other.hitpoint -= harm;

if (other.hitpoint < 0)

other.hitpoint = 0;

return other.hitpoint;

}

bool Monster::PriorTo(const Monster& other) const //谁先攻击

{

if (speed != other.speed)

return speed>other.speed;

if (hitpoint != other.hitpoint)

return hitpoint > other.hitpoint;

if (damage != other.damage)

return damage > other.damage;

if (defense != other.defense)

return defense > other.defense;

return (id < other.id );

//或直接return true;

}

/\*

\* ***AppMain.cpp***

\* Author: chenwei

\*/

#include <iostream>

using namespace std;

#include "Monster.h"

int main()

{

Monster a(10,200,7,8);

Monster b(10,150,8,7); //改成(10,180,8,7)则战斗失败

if (a.Fight(b))

cout<<"A Win!"<<endl;

else

cout<<"A Lose!"<<endl;

return 0;

}