**北京邮电大学课程设计报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | **面向对象程序设计实践（C++）** | | **学 院** | **计算机学院** | **指导教师** | **王洪波** |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| **2018211304** | **16** | **2018211208** | | **谢睿** |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | 1. 基础实验 2. 编写C++程序完成“矩阵”相加减 3. (1)编写C++程序完成“圆形”及判断相交   (2)使用“类”2 编写C++程序完成“矩阵”类以下功能   1. 编写C++程序完成“形状”及其派生类 2. I/O 流实验 编写 C++程序完成猜价格游戏 3. (1)为“形状”定义虚函数；定义基类为抽象类   (2)对Point类重载＋＋和――运算符   1. 综合实验   宠物小精灵对战  涉及：cs模式通信，用户注册，用户登陆，用户登出；精灵对战，精灵升级，精灵释放技能；获取精灵，送出精灵。 | | | | | |
| **学生**  **课程设计**  **报告**  （附页） | 见附页 | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | 遵照实践教学大纲并根据以下四方面综合评定成绩：  1、课程设计目的任务明确，选题符合教学要求，份量及难易程度  2、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果  3、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范  **评语**:        **成绩**:  指导教师签名：  年 月 日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

目录

[一．基础实验讲解 1](#_Toc25172)

[1.“类与对象实验”中圆形对象以及 Point 类成员的构造函数与析构函数的调用： 1](#_Toc28879)

[2.“继承与派生实验”中创建派生类的对象，观察构造函数、析构函数调用次序 2](#_Toc15501)

[3.“I/O 流实验”注意检查输入的合法性 3](#_Toc16566)

[4.“重载实验”中将【形状】 中的基类计算面积的方法定义为虚函数，比较与【形状（A）】程 序的差异： 4](#_Toc22334)

[5.“重载实验”中将【形状】中的基类定义为抽象类，比较与【形状（A）】程序的差异： 5](#_Toc24335)

[6.“重载实验”中对Point类重载＋＋和――运算符： 5](#_Toc2066)

[二．综合实验讲解 6](#_Toc9113)

[1.精灵的加入 6](#_Toc17794)

[2.用户注册与登陆 7](#_Toc6587)

[3.游戏对战的设计 9](#_Toc26856)

[4.附加设计 11](#_Toc27335)

[三．所有题目代码 13](#_Toc24099)

[1.基础1 13](#_Toc920)

[2.基础2.1 15](#_Toc5454)

[3.基础2.2 17](#_Toc6105)

[4.基础3 22](#_Toc28516)

[5.基础4 26](#_Toc30894)

[7.基础5.1(2) 30](#_Toc25793)

[8.基础5.2 34](#_Toc8610)

[9.综合（服务器） 36](#_Toc23773)

[文件1：header.h 36](#_Toc13828)

[文件2：Skills.h 37](#_Toc10424)

[文件3：server.h 40](#_Toc14866)

[文件4：PokemonList.h 62](#_Toc24015)

[文件5：main.cpp 101](#_Toc28331)

[10.综合（客户端） 123](#_Toc32146)

[文件：main.cpp 123](#_Toc11098)

[四．参考资料 134](#_Toc30133)

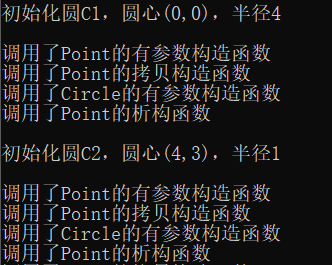
# 一．基础实验讲解

## 1.“类与对象实验”中圆形对象以及 Point 类成员的构造函数与析构函数的调用：

主函数如下：

1. cout << endl << "初始化圆C1，圆心(0,0)，半径4" << endl << endl;
2. Circle C1(Point(0, 0), 4);
3. cout << endl << "初始化圆C2，圆心(4,3)，半径1" << endl << endl;
4. Circle C2(Point(4, 3), 1);
5. **int** flagCross = C1.isCross(C1, C2);
6. **switch** (flagCross)
7. {
8. **case** 1 :
9. cout << endl << "相交" << endl << endl;
10. **break**;
11. **case** 0 :
12. cout << endl << "相切" << endl << endl;
13. **break**;
14. **case** -1 :
15. cout << endl << "相离" << endl << endl;
16. **break**;
17. **default** :
18. cout << endl << "ERROR in Judge!" << endl << endl;
19. }

程序运行结果如下：



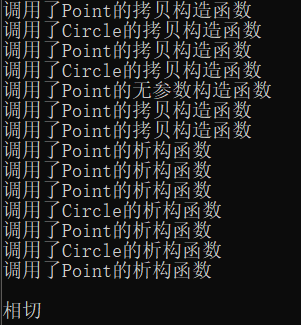
这两部分的调用是由于声明对象C1和C2时产生的。

第一句Point的有参数构造函数生成传入的参数；

第二句Point拷贝构造函数时传递参数Point(0,0)；

第三句Circle有参数构造函数是C1本身调用；

第四句Point的析构函数是释放传入的参数Point。



这部分判断相交，

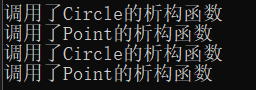
第一三句Point拷贝构造函数与第二四句Circle拷贝构造函数是传入参数C1，C2；

第五句Point无参数构造函数是生成函数isCross内声明的临时对象；

第六七句是传入两个参数Point；

第八九十句是释放临时Point以及传入的参数Point；

最后四句是释放传入的参数Circle；

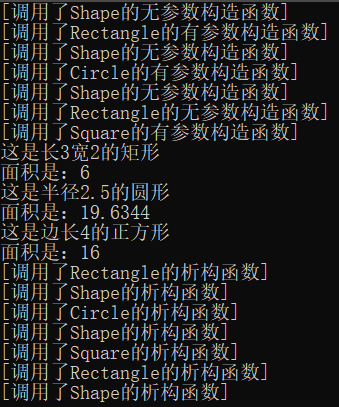


这部分是程序结束时释放C1和C2

先调用Circle再调用Point，因为Point是Circle的一个成员

## 2.“继承与派生实验”中创建派生类的对象，观察构造函数、析构函数调用次序

程序运行结果如下：



调用顺序是：

构造时先构造基类，析构时先析构派生类，顺序相反

## 3.“I/O 流实验”注意检查输入的合法性

使用cin.good()判断输入合法性：

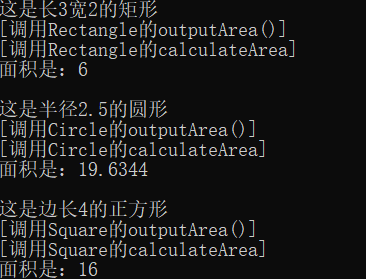
1. cin >> guess;
2. **if** (cin.good())//如果输入是整数
3. {
4. **if** (guess >= 1 && guess <= 1000)//如果范围正确
5. {
6. **if** (guess < price) cout << "没有这么便宜哦,再试一次吧：";
7. **else** **if** (guess == price)
8. {
9. cout << "恭喜！猜对啦~~" << endl;
10. continueFlag = 0;
11. }
12. **else** **if** (guess > price) cout << "没有这么贵哦，再试一次吧：";
13. }
14. **else**//如果范围不正确
15. {
16. cout << "您输入的数超过商品价值范围了，请重新输入：";
17. }
18. }
19. **else**//如果输入不是整数
20. {
21. cout << "输入不是整数，请重新输入：";
22. cin.clear();//清除错误标志
23. }
24. cin.ignore(numeric\_limits<streamsize>::max(), '\n');//清除输入缓冲区

## 4.“重载实验”中将【形状】 中的基类计算面积的方法定义为虚函数，比较与【形状（A）】程 序的差异：

定义为虚函数，则派生类重写的此同名函数可以由指向基类的指针调用。例如使用指向基类的指针p1，p2，p3计算相应图形的面积：

1. Rectangle\* pR = **new** Rectangle(3, 2);
2. Circle\* pC = **new** Circle(2.5);
3. Square\* pS = **new** Square(4);
4. Shape\* p1 = pR;
5. Shape\* p2 = pC;
6. Shape\* p3 = pS;
7. (\*pR).output();
8. (\*p1).outputArea();
9. (\*pC).output();
10. (\*p2).outputArea();
11. (\*pS).output();
12. (\*p3).outputArea();
13. **delete** pR;
14. **delete** pC;
15. **delete** pS;

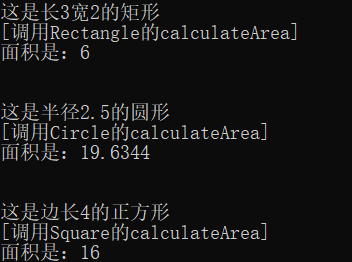
运行结果如下：



## 5.“重载实验”中将【形状】中的基类定义为抽象类，比较与【形状（A）】程序的差异：

抽象类中的纯虚函数必须由派生类实现，起到定义接口，规范程序的作用。

运行结果类似：



## 6.“重载实验”中对Point类重载＋＋和――运算符：

前置运算符和后置运算符的重载方式略有不同：

1. Point& operator --()//--p重载
2. {
3. X--;
4. Y--;
5. **return** \***this**;
6. }
7. Point& operator ++()//++p重载
8. {
9. X++;
10. Y++;
11. **return** \***this**;
12. }
13. Point operator --(**int**)//p--重载
14. {
15. Point tmp(X, Y);
16. X--;
17. Y--;
18. **return** tmp;
19. }Point operator ++(**int**)//p++重载
20. {
21. Point tmp(X, Y);
22. X++;
23. Y++;
24. **return** tmp;
25. }

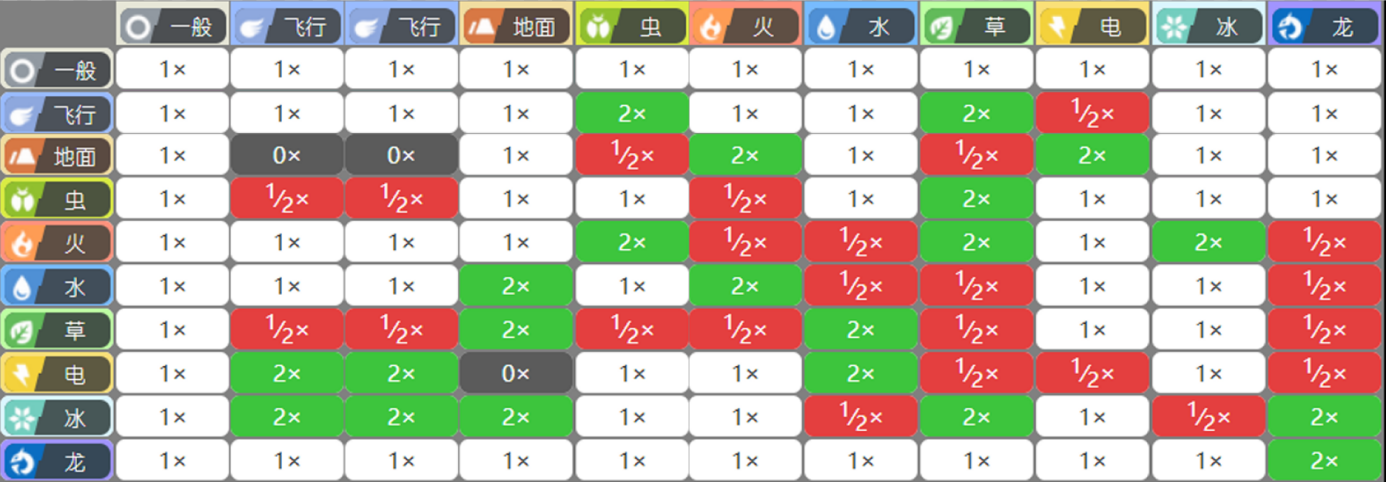
# 二．综合实验讲解

## 1.精灵的加入

1. 精灵共有四种种类：力量型：高 攻击； 肉盾型：高生命值； 防御型：高防御； 敏捷型：低攻击间隔。
2. 此外，还额外设置精灵的元素属性，共十种：
3. NORMAL 0 //普通
4. FIRE 1      //火
5. GRASS 2     //草
6. WATER 3     //水
7. BUG 4       //虫
8. FLY 5       //飞
9. GROUND 6    //地
10. ELECTRIC 7  //电
11. ICE 8       //冰
12. DRAGON 9    //龙

元素间存在克制关系，导致伤害增加或减少。

属性相克表如下：

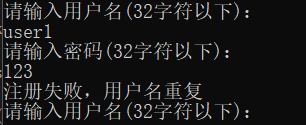


除去普通，每种元素有一只精灵，共9只：

1. BLUBASAUR 1 //妙蛙种子
2. CHARIZARD 2 //喷火龙
3. SQUIRTLE 3  //杰尼龟
4. CATERPIE 4  //绿毛虫
5. PIDGEOTTO 5 //比比鸟
6. DIGLETT 6   //地鼠
7. PIKACHU 7   //皮卡丘
8. SNORUNT 8   //雪童子
9. DRAGONITE 9 //快龙
10. 随着等级提高，精灵可以学习技能，最多学会4种技能。每种元素的精灵可以学会的技能有不同，详见代码”Skills.h”中的定义。
11. 全部精灵由基类“Pokemon”继承而来。
12. 精灵升级会增加2点攻击，5点HP，1点防御；四种属性还会额外增加对应的属性。

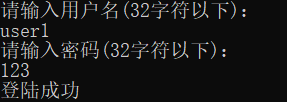
## 2.用户注册与登陆

1. 注册：需输入用户名及密码，用户名不得与现有用户重复，若重复则展示提示信息并要求重新输入。注册成功后随机分发3个精灵给用户。

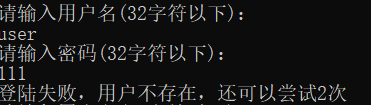


1. 登陆：登陆成功以及登陆失败都有对应提示

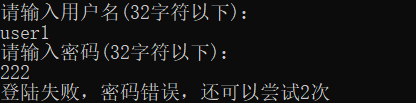
登陆成功：



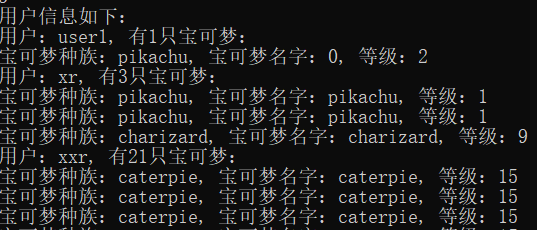
登陆失败：用户不存在



登陆失败：密码错误



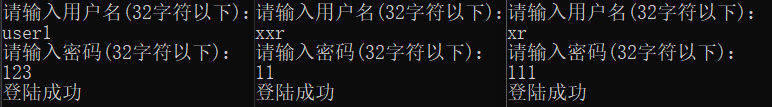
1. 查看所有用户：



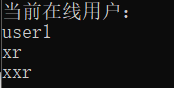
...

1. 查看在线用户：

首先开启三个客户端，分别登陆user1，xr，xxr；

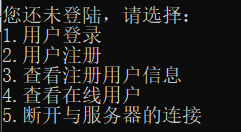


然后再打开一个客户端，查询在线用户如下：

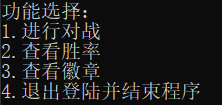


1. 交互反馈：系统有两级菜单，操作都有对应的反馈逻辑

一级菜单是登陆前：

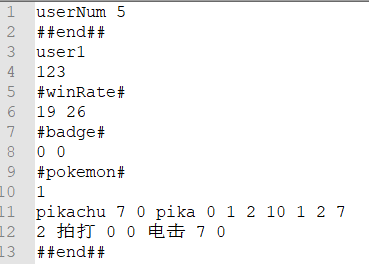


二级菜单是登陆后：



1. 用户信息存储方式使用TXT文本文档：

例：



第一行记录已注册用户数量；

第二行是结束标志；

第三四行是一个用户的用户名及密码；

第五行是胜率（战斗记录）开始标志；

第六行记录胜场及总场数；

第七行是徽章记录开始标记；

第八行记录是否拥有两种徽章以及徽章材质（金银铜）

第九行是拥有的精灵数据开始标志；

第十行是拥有就精灵数量；

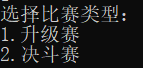
第十一行是第一只精灵属性信息：“种族 元素 属性 昵称 经验值 等级 攻击力 生命值 防御力 速度 图鉴索引值”；

第十二行是这只精灵的技能；

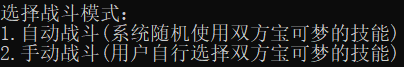
第十三行是结束标志。

## 3.游戏对战的设计

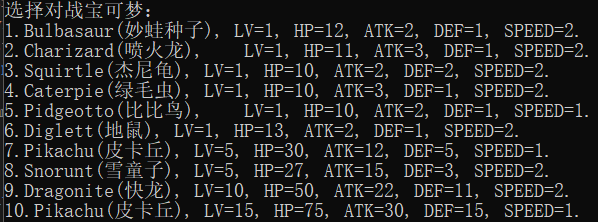
1. 对战分升级赛和决斗赛：



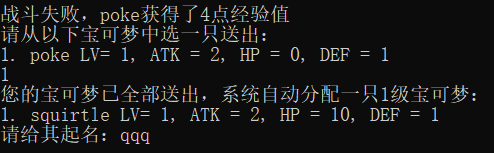
1. 战斗可以由系统自动随机进行，也可以用户自行选择出招：



1. 服务器上有一个虚拟精灵列表：

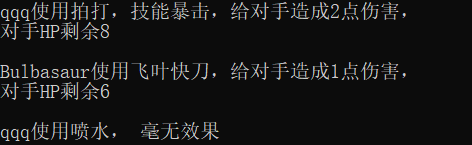


1. 全部送出后系统会分配一只精灵：



1. 随机性：

精灵有复数个招式可以使用，服务器随机选择出招；攻击有暴击与闪避的随机性：







1. 查看胜率：



1. 查看徽章：

精灵较少的用户基本没有徽章：



精灵较多较强的用户有徽章：



1. 服务器使用多进程监听并响应客户端，相互独立地与各个客户端通信

系统有服务线程：

1. //服务线程
2. **DWORD** WINAPI ServerThread(**LPVOID** lpParameter)

主函数负责监听客户端请求并建立新线程：

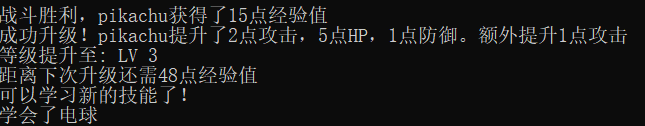
1. **if** (\*clntSock == SOCKET\_ERROR)
2. {
3. cout << "接收连接请求失败" << endl;
4. closesocket(\*clntSock);
5. }
6. **else**
7. {
8. cout << endl << "一个客户端尝试连接到服务器，socket是：" << \*clntSock << endl;
9. CreateThread(NULL, 0, &ServerThread, nowClient, 0, NULL);
10. }

## 4.附加设计

1. 精灵升级可以学习新技能。

精灵最多学会4个技能，初始会2个技能。在3，6，9，12，15级分别可以学习一个新技能。

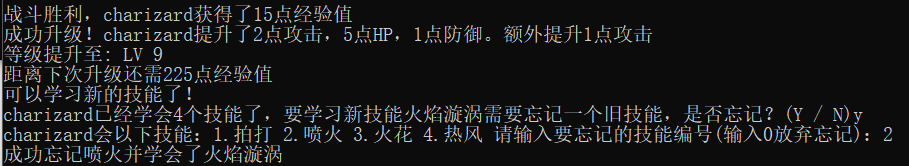
例如下图，精灵pikachu在升级到LV3时学会了技能“电球”：



1. 精灵可以遗忘旧技能，是否遗忘由用户选择。

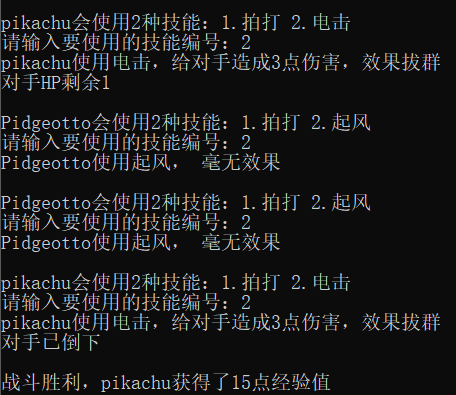
当已学会的技能数达到上限，则学习新技能需要遗忘旧技能。

例如下图显示，精灵charizard升到LV9时，可以学习技能“火焰漩涡”，但因为已经学会了4个技能，所以选择忘记一个技能“喷火”，这样就成功学会了“火焰漩涡”：



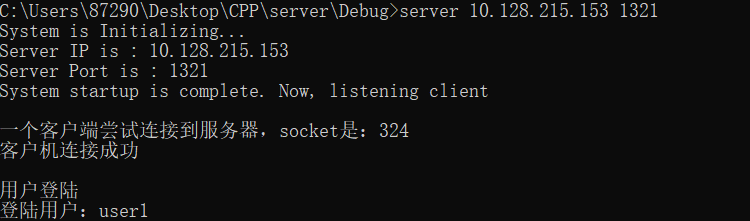
1. 除了自动战斗，用户还可以选择手动出招，体验通过策略战胜属性值高于自己的精灵。

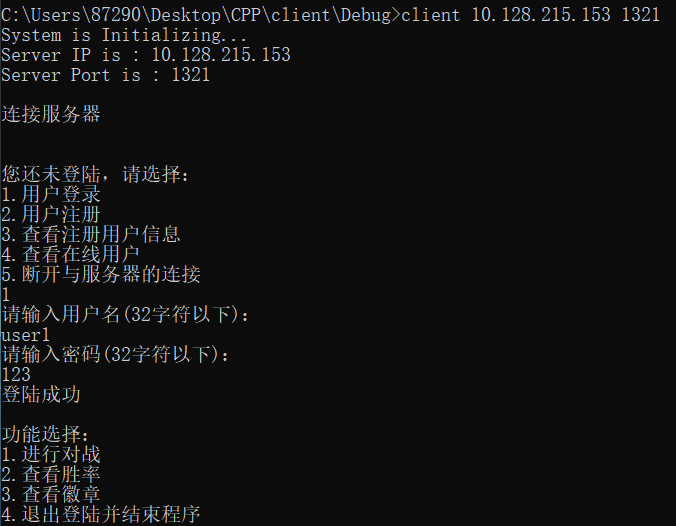
例如下图，精灵pikachu通过使用克制对方精灵pidgetto的技能，成功战胜属性较高的对手：



1. 可以指定服务器ip与端口号，即便于将服务器和客户端置于不同设备上。

首先查看服务器IP，为119.3.234.165，使用通过命令行启动服务器，IP设为119.3.234.165，端口号随机设为1321；同时客户机采用相同地址与端口号运行，即可成功连接：





# 三．所有题目代码

## 1.基础1

1. #include<iostream>
2. **using** **namespace** std;
3. //不用对象
4. **void** Madd(**int**\*\* pA1, **int**\*\* pA2, **int**\*\* pA3)//pA1 + pA2 -> pA3
5. {
6. **for** (**int** i = 0; i < 4; i++)
7. {
8. **for** (**int** j = 0; j < 5; j++)
9. {
10. pA3[i][j] = pA1[i][j] + pA2[i][j];
11. }
12. }
13. }
14. **void** Msub(**int**\*\* pA1, **int**\*\* pA2, **int**\*\* pA3)//pA1 - pA2 -> pA3
15. {
16. **for** (**int** i = 0; i < 4; i++)
17. {
18. **for** (**int** j = 0; j < 5; j++)
19. {
20. pA3[i][j] = pA1[i][j] - pA2[i][j];
21. }
22. }
23. }
24. **void** Mout(**int**\*\* p)//输出矩阵
25. {
26. **for** (**int** i = 0; i < 4; i++)
27. {
28. **for** (**int** j = 0; j < 5; j++)
29. {
30. cout << p[i][j] << "\t";
31. }
32. cout << endl;
33. }
34. }
35. **void** Mfree(**int**\*\* p)//释放矩阵
36. {
37. **for** (**int** i = 0; i < 4; i++)
38. {
39. free(p[i]);
40. }
41. free(p);
42. }
43. **int** main()
44. {
45. **int**\*\* pA1 = **new** **int**\* [4];
46. **int**\*\* pA2 = **new** **int**\* [4];
47. **int**\*\* pA3 = **new** **int**\* [4];
48. **for** (**int** i = 0; i < 4; i++)
49. {
50. pA1[i] = **new** **int**[5];
51. pA2[i] = **new** **int**[5];
52. pA3[i] = **new** **int**[5];
53. }
54. cout << "初始化A1，A2：" << endl;
55. cout << "A1:" << endl;
56. **for** (**int** i = 0; i < 4; i++)
57. {
58. **for** (**int** j = 0; j < 5; j++)
59. {
60. cin >> pA1[i][j];
61. }
62. }
63. cout << "A2:" << endl;
64. **for** (**int** i = 0; i < 4; i++)
65. {
66. **for** (**int** j = 0; j < 5; j++)
67. {
68. cin >> pA2[i][j];
69. }
70. }
71. Madd(pA1, pA2, pA3);
72. cout << "A3 = A1 + A2:" << endl;
73. Mout(pA3);
74. Msub(pA1, pA2, pA3);
75. cout << "A3 = A1 - A2:" << endl;
76. Mout(pA3);
78. Mfree(pA1);
79. Mfree(pA2);
80. Mfree(pA3);
82. **return** 0;
83. }

## 2.基础2.1

1. #include<iostream>
2. #include<math.h>
3. **using** **namespace** std;
5. **class** Point
6. {
7. **private**:
8. **double** X;
9. **double** Y;
10. **public**:
11. Point();
12. Point(**double** newX, **double** newY);
13. Point(**const** Point& p);
14. ~Point();
15. **double** distance(Point A,Point B);
16. };
17. Point::Point()
18. {
19. cout << "调用了Point的无参数构造函数" << endl;
20. X = 0;
21. Y = 0;
22. }
23. Point::Point(**double** newX, **double** newY)
24. {
25. cout << "调用了Point的有参数构造函数" << endl;
26. X = newX;
27. Y = newY;
28. }
29. Point::Point(**const** Point& p)
30. {
31. cout << "调用了Point的拷贝构造函数" << endl;
32. X = p.X;
33. Y = p.Y;
34. }
35. Point::~Point()
36. {
37. cout << "调用了Point的析构函数" << endl;
38. }
39. **double** Point::distance(Point A,Point B)
40. {
41. **double** disSquare = 0;
42. disSquare = (A.X - B.X) \* (A.X - B.X) + (A.Y - B.Y) \* (A.Y - B.Y);
43. **return** sqrt(disSquare);
44. }
46. **class** Circle
47. {
48. **private**:
49. Point center;
50. **double** radius;
51. **public**:
52. Circle();
53. Circle(Point newCenter, **double** newRad);
54. Circle(**const** Circle& c);
55. ~Circle();
56. **int** isCross(Circle A,Circle B);
57. };
58. Circle::Circle()
59. :center(0, 0)
60. {
61. cout << "调用了Circle的无参数构造函数" << endl;
62. radius = 0;
63. }
64. Circle::Circle(Point newCenter, **double** newRad)
65. :center(newCenter)
66. {
67. cout << "调用了Circle的有参数构造函数" << endl;
68. radius = newRad;
69. }
70. Circle::Circle(**const** Circle& c)
71. :center(c.center)
72. {
73. cout << "调用了Circle的拷贝构造函数" << endl;
74. radius = c.radius;
75. }
76. Circle::~Circle()
77. {
78. cout << "调用了Circle的析构函数" << endl;
79. }
80. **int** Circle::isCross(Circle A, Circle B)
81. {
82. Point tmp;
83. **double** dis = tmp.distance(A.center, B.center);
84. **double** radSum = A.radius + B.radius;
85. **if** (dis < radSum) **return** 1;
86. **else** **if** (dis == radSum) **return** 0;
87. **else** **return** -1;
88. }
89. **int** main()
90. {
91. **int** tmp;
92. cout << endl << "初始化圆C1，圆心(0,0)，半径4" << endl << endl;
93. Circle C1(Point(0, 0), 4);
94. cout << endl << "初始化圆C2，圆心(4,3)，半径1" << endl << endl;
95. Circle C2(Point(4, 3), 1);
96. **int** flagCross = C1.isCross(C1, C2);
97. **switch** (flagCross)
98. {
99. **case** 1 :
100. cout << endl << "相交" << endl << endl;
101. **break**;
102. **case** 0 :
103. cout << endl << "相切" << endl << endl;
104. **break**;
105. **case** -1 :
106. cout << endl << "相离" << endl << endl;
107. **break**;
108. **default** :
109. cout << endl << "ERROR in Judge!" << endl << endl;
110. }
112. **return** 0;
113. }

## 3.基础2.2

1. #include<iostream>
2. **using** **namespace** std;
4. **class** Matrix
5. {
6. **public**:
7. Matrix();
8. Matrix(**int** r, **int** c);
9. Matrix(**const** Matrix& m);
10. ~Matrix();
11. Matrix& operator =(**const** Matrix& m)//赋值重载
12. {
13. **if** (**this** != &m)//自赋值判断
14. {
15. **this**->~Matrix();//先释放原空间
16. row = m.row;
17. col = m.col;
18. data = **new**  **int**\* [row];
19. **for** (**int** i = 0; i < row; i++)
20. {
21. data[i] = **new** **int**[col];
22. **for** (**int** j = 0; j < col; j++)
23. {
24. data[i][j] = m.data[i][j];
25. }
26. }
27. }
28. **return** \***this**;
29. }
30. **void** Minit();//输入函数
31. **void** Mshow();//输出函数
32. Matrix Madd(Matrix m1, Matrix m2, Matrix& m3);//m1 + m2 -> m3
33. Matrix Msub(Matrix m1, Matrix m2, Matrix& m3);//m1 - m2 -> m3
34. **private**:
35. **int** row, col;
36. **int**\*\* data;
37. };
38. Matrix::Matrix()
39. {
40. row = 4;
41. col = 5;//默认4\*5
42. data = **new**  **int**\* [row];
43. **for** (**int** i = 0; i < row; i++)
44. {
45. data[i] = **new** **int** [col];
46. **for** (**int** j = 0; j < col; j++)
47. {
48. data[i][j] = 0;//默认初始化为零矩阵
49. }
50. }
51. }
52. Matrix::Matrix(**int** r, **int** c)
53. {
54. row = r;
55. col = c;
56. data = **new**  **int**\* [row];
57. **for** (**int** i = 0; i < row; i++)
58. {
59. data[i] = **new** **int**[col];
60. **for** (**int** j = 0; j < col; j++)
61. {
62. data[i][j] = 0;//默认初始化为零矩阵
63. }
64. }
65. }
66. Matrix::Matrix(**const** Matrix& m)
67. {
68. row = m.row;
69. col = m.col;
70. data = **new**  **int**\* [row];
71. **for** (**int** i = 0; i < row; i++)
72. {
73. data[i] = **new** **int**[col];
74. **for** (**int** j = 0; j < col; j++)
75. {
76. data[i][j] = m.data[i][j];
77. }
78. }
79. }
80. Matrix::~Matrix()
81. {
82. **for** (**int** i = 0; i < row; i++)
83. {
84. **delete**[] data[i];
85. }
86. **delete**[] data;
87. }
88. **void** Matrix::Minit()
89. {
90. cout << "请输入矩阵初始化值（大小" << row << "\*" << col << "）。空格分隔，回车换行：" << endl;
91. **for** (**int** i = 0; i < row; i++)
92. {
93. **for** (**int** j = 0; j < col; j++)
94. {
95. cin >> data[i][j];
96. }
97. }
98. }
99. **void** Matrix::Mshow()
100. {
101. cout << "矩阵为：" << endl;
102. **for** (**int** i = 0; i < row; i++)
103. {
104. **for** (**int** j = 0; j < col; j++)
105. {
106. cout << data[i][j] << "\t";
107. }
108. cout << endl;
109. }
110. }
111. Matrix Matrix::Madd(Matrix m1, Matrix m2, Matrix& m3)
112. {
113. **if** (m1.row == m2.row && m1.col == m2.col)
114. {
115. m3.~Matrix();//清除m3原数据并使得m3大小符合计算要求
116. m3.row = m1.row;
117. m3.col = m1.col;
118. m3.data = **new**  **int**\* [row];
119. **for** (**int** i = 0; i < row; i++)
120. {
121. m3.data[i] = **new** **int**[col];
122. **for** (**int** j = 0; j < col; j++)
123. {
124. m3.data[i][j] = m1.data[i][j] + m2.data[i][j];
125. }
126. }
127. //m3.Mshow();
128. }
129. **else**
130. {
131. cout << "相加的两矩阵大小不同，无法运算" << endl;
132. }
133. **return** m3;
134. }
135. Matrix Matrix::Msub(Matrix m1, Matrix m2, Matrix& m3)
136. {
137. **if**(m1.row == m2.row && m1.col == m2.col)
138. {
139. m3.~Matrix();//清除m3原数据并使得m3大小符合计算要求
140. m3.row = m1.row;
141. m3.col = m1.col;
142. m3.data = **new**  **int**\* [row];
143. **for** (**int** i = 0; i < row; i++)
144. {
145. m3.data[i] = **new** **int**[col];
146. **for** (**int** j = 0; j < col; j++)
147. {
148. m3.data[i][j] = m1.data[i][j] - m2.data[i][j];
149. }
150. }
151. //m3.Mshow();
152. }
153. **else**
154. {
155. cout << "相减的两矩阵大小不同，无法运算" << endl;
156. }
157. **return** m3;
158. }
160. **int** main()
161. {
162. **int** A1row, A1col, A2row, A2col;
163. cout << "请输入矩阵A1大小,格式\"行数 空格 列数\"：" << endl;
164. cin >> A1row >> A1col;
165. cout << "请输入矩阵A2大小,格式\"行数 空格 列数\"：" << endl;
166. cin >> A2row >> A2col;
168. //普通声明：
169. Matrix A1(A1row,A1col), A2(A2row,A2col), A3;
170. cout << "已采用普通声明建立两个矩阵A1，A2；以及默认大小的矩阵A3。接下来对A1和A2进行矩阵初始化" << endl;
171. A1.Minit();
172. A2.Minit();
173. cout << "计算 A3 = A1 + A2: " << endl;
174. A3 = A3.Madd(A1, A2, A3);
175. A3.Mshow();
176. cout << "计算 A3 = A1 - A2: " << endl;
177. A3 = A3.Msub(A1, A2, A3);
178. A3.Mshow();
179. //此种声明由于new操作在对象内，所以系统会自动调用析构函数，无需显式地释放空间
181. //用new声明：
182. Matrix\* pA1 = **new** Matrix(A1row, A1col);
183. Matrix\* pA2 = **new** Matrix(A2row, A2col);
184. Matrix\* pA3 = **new** Matrix;
185. cout << "已用new声明建立两个矩阵pA1，pA2；以及默认大小的矩阵pA3。接下来对pA1和pA2进行矩阵初始化" << endl;
186. (\*pA1).Minit();
187. (\*pA2).Minit();
188. (\*pA3) = (\*pA3).Madd(\*pA1, \*pA2, \*pA3);
189. cout << "计算 pA3 = pA1 + pA2: " << endl;
190. (\*pA3).Mshow();
191. (\*pA3) = (\*pA3).Msub(\*pA1, \*pA2, \*pA3);
192. cout << "计算 pA3 = pA1 - pA2: " << endl;
193. (\*pA3).Mshow();
194. **delete** pA1;
195. **delete** pA2;
196. **delete** pA3;
197. //此种声明由于对象外还有new操作，所以需要显式地释放空间
199. **return**  0;
200. }

## 4.基础3

1. #include<iostream>
2. #define PI 3.1415//圆周率近似值
3. **using** **namespace** std;
5. **class** Shape
6. {
7. **public**:
8. Shape(){ cout << "[调用了Shape的无参数构造函数]" << endl; }
9. Shape(**const** Shape& p){ cout << "[调用了Shape的拷贝构造函数]" << endl; }
10. ~Shape(){ cout << "[调用了Shape的析构函数]" << endl; }
11. **double** calculateArea() { cout << "基类无法计算面积" << endl; **return** -1; }
12. };
14. **class** Rectangle :**public** Shape
15. {
16. **public**:
17. Rectangle();
18. Rectangle(**double** a, **double** b);
19. Rectangle(**const** Rectangle& p);
20. ~Rectangle() { cout << "[调用了Rectangle的析构函数]" << endl; }
21. **double** calculateArea() { **return** length \* width; }
22. **void** output() { cout << "这是长" << length << "宽" << width << "的矩形" << endl; }
23. **private**:
24. **double** length;//长
25. **double** width;//宽
26. };
27. Rectangle::Rectangle()
28. {
29. length = 0;
30. width = 0;
31. cout << "[调用了Rectangle的无参数构造函数]" << endl;
32. }
33. Rectangle::Rectangle(**double** a, **double** b)
34. {
35. length = a;
36. width = b;
37. cout << "[调用了Rectangle的有参数构造函数]" << endl;
38. }
39. Rectangle::Rectangle(**const** Rectangle& p)
40. {
41. length = p.length;
42. width = p.width;
43. cout << "[调用了Rectangle的拷贝构造函数]" << endl;
44. }

47. **class** Circle :**public** Shape
48. {
49. **public**:
50. Circle();
51. Circle(**double** a);
52. Circle(**const** Circle& p);
53. ~Circle() { cout << "[调用了Circle的析构函数]" << endl; }
54. **double** calculateArea() { **return** PI \* radius \* radius; }
55. **void** output() { cout << "这是半径" << radius << "的圆形" << endl; }
56. **private**:
57. **double** radius;//半径
58. };
59. Circle::Circle(**double** a)
60. {
61. radius = a;
62. cout << "[调用了Circle的有参数构造函数]" << endl;
63. }
64. Circle::Circle()
65. {
66. radius = 0;
67. cout << "[调用了Circle的无参数构造函数]" << endl;
68. }
69. Circle::Circle(**const** Circle& p)
70. {
71. radius = p.radius;
72. cout << "[调用了Circle的拷贝构造函数]" << endl;
73. }
75. **class** Square :**public** Rectangle
76. {
77. **public**:
78. Square();
79. Square(**double** a);
80. Square(**const** Square& p);
81. ~Square() { cout << "[调用了Square的析构函数]" << endl; }
82. **double** calculateArea() { **return** edge \* edge; }
83. **void** output() { cout << "这是边长" << edge << "的正方形" << endl; }
84. **private**:
85. **double** edge;//边长
86. };
87. Square::Square(**double** a)
88. {
89. edge = a;
90. cout << "[调用了Square的有参数构造函数]" << endl;
91. }
92. Square::Square()
93. {
94. edge = 0;
95. cout << "[调用了Square的无参数构造函数]" << endl;
96. }
97. Square::Square(**const** Square& p)
98. {
99. edge = p.edge;
100. cout << "[调用了Square的拷贝构造函数]" << endl;
101. }
103. **int** main()
104. {
105. Rectangle\* pR = **new** Rectangle(3, 2);
106. Circle\* pC = **new** Circle(2.5);
107. Square\* pS = **new** Square(4);
109. (\*pR).output();
110. cout << "面积是：" << (\*pR).calculateArea() << endl;
111. (\*pC).output();
112. cout << "面积是：" << (\*pC).calculateArea() << endl;
113. (\*pS).output();
114. cout << "面积是：" << (\*pS).calculateArea() << endl;
116. **delete** pR;
117. **delete** pC;
118. **delete** pS;
120. **return** 0;
121. }

## 5.基础4

1. #include <iostream>
2. #include <cstdlib>
3. #include <ctime>
4. **使用命名空间** std;
6. **int** main（）
7. {
8. **int** continueFlag = 1;
9. srand（（**int**）time（0））;
10. **整型** 价格= rand（）％1001 + 1;
11. **INT** 猜测;
12. cout <<价格<< endl;
13. cout <<  “我们有一件商品，请您猜测其价格（1-1000的整数）”  << endl;
14. cout <<  “请输入您猜测的价格：” ;
15. **而** （continueFlag）
16. {
17. cin >>猜;
18. **if** （cin.good（））//如果输入是整体
19. {
20. **if** （猜测> = 1 &&猜测<= 1000）//如果范围正确
21. {
22. **if** （guess <price）cout <<  “没有这么便宜哦，再试一次吧：” ;
23. **否则，如果** （猜测==价格）
24. {
25. cout <<  “恭喜！猜对啦~~”  << endl;
26. ContinueFlag = 0;
27. }
28. **否则if** （guess> price）cout <<  “没有这么贵哦，再试一次吧：” ;
29. }
30. **else**//如果范围不正确
31. {
32. cout <<  “您输入的数超过商品价值范围了，请重新输入：” ;
33. }
34. }
35. **else**//如果输入不是整数
36. {
37. cout <<  “输入不是整数，请重新输入：” ;
38. cin.clear（）; //清除错误标志
39. }
40. cin.ignore（numeric\_limits <streamsize> :: max（），  '\ n' ）; //清除输入监狱
41. }
43. **返回** 0;
44. }

#### 6.基础5.1(1)

1. #include<iostream>
2. #define PI 3.1415//圆周率近似值
3. **using** **namespace** std;
5. **class** Shape
6. {
7. **public**:
8. Shape() { cout << "[调用了Shape的无参数构造函数]" << endl; }
9. Shape(**const** Shape& p) { cout << "[调用了Shape的拷贝构造函数]" << endl; }
10. ~Shape() { cout << "[调用了Shape的析构函数]" << endl; }
11. **virtual** **double** calculateArea() { cout << "基类无法计算面积" << endl; **return** -1; }
12. **virtual** **void** outputArea();
13. };
14. **void** Shape::outputArea()
15. {
16. cout << "[调用基类outputArea()]" << endl;
17. cout << "面积是：" << calculateArea() << endl << endl;
18. }
20. **class** Rectangle :**public** Shape
21. {
22. **public**:
23. Rectangle();
24. Rectangle(**double** a, **double** b);
25. Rectangle(**const** Rectangle& p);
26. ~Rectangle() { cout << "[调用了Rectangle的析构函数]" << endl; }
27. **virtual** **double** calculateArea() { cout << "[调用Rectangle的calculateArea]" << endl; **return** length \* width; }
28. **void** output() { cout << "这是长" << length << "宽" << width << "的矩形" << endl; }
29. **void** outputArea();
30. **private**:
31. **double** length;//长
32. **double** width;//宽
33. };
34. Rectangle::Rectangle()
35. {
36. length = 0;
37. width = 0;
38. cout << "[调用了Rectangle的无参数构造函数]" << endl;
39. }
40. Rectangle::Rectangle(**double** a, **double** b)
41. {
42. length = a;
43. width = b;
44. cout << "[调用了Rectangle的有参数构造函数]" << endl;
45. }
46. Rectangle::Rectangle(**const** Rectangle& p)
47. {
48. length = p.length;
49. width = p.width;
50. cout << "[调用了Rectangle的拷贝构造函数]" << endl;
51. }
52. **void** Rectangle::outputArea()
53. {
54. cout << "[调用Rectangle的outputArea()]" << endl;
55. cout << "面积是：" << calculateArea() << endl << endl;
56. }
58. **class** Circle :**public** Shape
59. {
60. **public**:
61. Circle();
62. Circle(**double** a);
63. Circle(**const** Circle& p);
64. ~Circle() { cout << "[调用了Circle的析构函数]" << endl; }
65. **double** calculateArea() { cout << "[调用Circle的calculateArea]" << endl; **return** PI \* radius \* radius; }
66. **void** output() { cout << "这是半径" << radius << "的圆形" << endl; }
67. **void** outputArea();
68. **private**:
69. **double** radius;//半径
70. };
71. Circle::Circle(**double** a)
72. {
73. radius = a;
74. cout << "[调用了Circle的有参数构造函数]" << endl;
75. }
76. Circle::Circle()
77. {
78. radius = 0;
79. cout << "[调用了Circle的无参数构造函数]" << endl;
80. }
81. Circle::Circle(**const** Circle& p)
82. {
83. radius = p.radius;
84. cout << "[调用了Circle的拷贝构造函数]" << endl;
85. }
86. **void** Circle::outputArea()
87. {
88. cout << "[调用Circle的outputArea()]" << endl;
89. cout << "面积是：" << calculateArea() << endl << endl;
90. }
92. **class** Square :**public** Rectangle
93. {
94. **public**:
95. Square();
96. Square(**double** a);
97. Square(**const** Square& p);
98. ~Square() { cout << "[调用了Square的析构函数]" << endl; }
99. **double** calculateArea() { cout << "[调用Square的calculateArea]" << endl; **return** edge \* edge; }
100. **void** output() { cout << "这是边长" << edge << "的正方形" << endl; }
101. **void** outputArea();
102. Square& operator =(**const** Square& m)//赋值重载
103. {
104. **if** (**this** != &m)//自赋值判断
105. {
106. edge = m.edge;
107. }
108. **return** \***this**;
109. }
110. **private**:
111. **double** edge;//边长
112. };
113. Square::Square(**double** a)
114. {
115. edge = a;
116. cout << "[调用了Square的有参数构造函数]" << endl;
117. }
118. Square::Square()
119. {
120. edge = 0;
121. cout << "[调用了Square的无参数构造函数]" << endl;
122. }
123. Square::Square(**const** Square& p)
124. {
125. edge = p.edge;
126. cout << "[调用了Square的拷贝构造函数]" << endl;
127. }
128. **void** Square::outputArea()
129. {
130. cout << "[调用Square的outputArea()]" << endl;
131. cout << "面积是：" << calculateArea() << endl << endl;
132. }
134. **int** main()
135. {
136. Rectangle\* pR = **new** Rectangle(3, 2);
137. Circle\* pC = **new** Circle(2.5);
138. Square\* pS = **new** Square(4);
139. Shape\* p1 = pR;
140. Shape\* p2 = pC;
141. Shape\* p3 = pS;

144. (\*pR).output();
145. (\*p1).outputArea();
146. (\*pC).output();
147. (\*p2).outputArea();
148. (\*pS).output();
149. (\*p3).outputArea();
151. **delete** pR;
152. **delete** pC;
153. **delete** pS;
155. **return** 0;
156. }

## 7.基础5.1(2)

1. #include<iostream>
2. #define PI 3.1415//圆周率近似值
3. **using** **namespace** std;
5. **class** Shape
6. {
7. **public**:
8. Shape() { cout << "[调用了Shape的无参数构造函数]" << endl; }
9. Shape(**const** Shape& p) { cout << "[调用了Shape的拷贝构造函数]" << endl; }
10. ~Shape() { cout << "[调用了Shape的析构函数]" << endl; }
11. **virtual** **double** calculateArea() = 0;//纯虚函数
12. };
14. **class** Rectangle :**public** Shape
15. {
16. **public**:
17. Rectangle();
18. Rectangle(**double** a, **double** b);
19. Rectangle(**const** Rectangle& p);
20. ~Rectangle() { cout << "[调用了Rectangle的析构函数]" << endl; }
21. **double** calculateArea() { cout << "[调用Rectangle的calculateArea]" << endl; **return** length \* width; }
22. **void** output() { cout << "这是长" << length << "宽" << width << "的矩形" << endl; }
23. **private**:
24. **double** length;//长
25. **double** width;//宽
26. };
27. Rectangle::Rectangle()
28. {
29. length = 0;
30. width = 0;
31. cout << "[调用了Rectangle的无参数构造函数]" << endl;
32. }
33. Rectangle::Rectangle(**double** a, **double** b)
34. {
35. length = a;
36. width = b;
37. cout << "[调用了Rectangle的有参数构造函数]" << endl;
38. }
39. Rectangle::Rectangle(**const** Rectangle& p)
40. {
41. length = p.length;
42. width = p.width;
43. cout << "[调用了Rectangle的拷贝构造函数]" << endl;
44. }
46. **class** Circle :**public** Shape
47. {
48. **public**:
49. Circle();
50. Circle(**double** a);
51. Circle(**const** Circle& p);
52. ~Circle() { cout << "[调用了Circle的析构函数]" << endl; }
53. **double** calculateArea() { cout << "[调用Circle的calculateArea]" << endl; **return** PI \* radius \* radius; }
54. **void** output() { cout << "这是半径" << radius << "的圆形" << endl; }
55. **private**:
56. **double** radius;//半径
57. };
58. Circle::Circle(**double** a)
59. {
60. radius = a;
61. cout << "[调用了Circle的有参数构造函数]" << endl;
62. }
63. Circle::Circle()
64. {
65. radius = 0;
66. cout << "[调用了Circle的无参数构造函数]" << endl;
67. }
68. Circle::Circle(**const** Circle& p)
69. {
70. radius = p.radius;
71. cout << "[调用了Circle的拷贝构造函数]" << endl;
72. }
74. **class** Square :**public** Rectangle
75. {
76. **public**:
77. Square();
78. Square(**double** a);
79. Square(**const** Square& p);
80. ~Square() { cout << "[调用了Square的析构函数]" << endl; }
81. **double** calculateArea() { cout << "[调用Square的calculateArea]" << endl; **return** edge \* edge; }
82. **void** output() { cout << "这是边长" << edge << "的正方形" << endl; }
83. Square& operator =(**const** Square& m)//赋值重载
84. {
85. **if** (**this** != &m)//自赋值判断
86. {
87. edge = m.edge;
88. }
89. **return** \***this**;
90. }
91. **private**:
92. **double** edge;//边长
93. };
94. Square::Square(**double** a)
95. {
96. edge = a;
97. cout << "[调用了Square的有参数构造函数]" << endl;
98. }
99. Square::Square()
100. {
101. edge = 0;
102. cout << "[调用了Square的无参数构造函数]" << endl;
103. }
104. Square::Square(**const** Square& p)
105. {
106. edge = p.edge;
107. cout << "[调用了Square的拷贝构造函数]" << endl;
108. }
110. **int** main()
111. {
112. Rectangle\* pR = **new** Rectangle(3, 2);
113. Circle\* pC = **new** Circle(2.5);
114. Square\* pS = **new** Square(4);
115. //Shape\* p = pS;
117. cout << endl;
118. (\*pR).output();
119. cout << "面积是：" << (\*pR).calculateArea() << endl;
120. cout << endl;
121. cout << endl;
122. (\*pC).output();
123. cout << "面积是：" << (\*pC).calculateArea() << endl;
124. cout << endl;
125. cout << endl;
126. (\*pS).output();
127. cout << "面积是：" << (\*pS).calculateArea() << endl;
128. cout << endl;
130. **delete** pR;
131. **delete** pC;
132. **delete** pS;
134. **return** 0;
135. }

## 8.基础5.2

1. #include<iostream>
2. #include<math.h>
3. **using** **namespace** std;
5. **class** Point
6. {
7. **private**:
8. **double** X;
9. **double** Y;
10. **public**:
11. Point();
12. Point(**double** newX, **double** newY);
13. Point(**const** Point& p);
14. ~Point();
15. Point& operator --()//--p重载
16. {
17. X--;
18. Y--;
19. **return** \***this**;
20. }
21. Point& operator ++()//++p重载
22. {
23. X++;
24. Y++;
25. **return** \***this**;
26. }
27. Point operator --(**int**)//p--重载
28. {
29. Point tmp(X, Y);
30. X--;
31. Y--;
32. **return** tmp;
33. }Point operator ++(**int**)//p++重载
34. {
35. Point tmp(X, Y);
36. X++;
37. Y++;
38. **return** tmp;
39. }
40. **void** show();
41. };
42. Point::Point()
43. {
44. //cout << "调用了Point的无参数构造函数" << endl;
45. X = 0;
46. Y = 0;
47. }
48. Point::Point(**double** newX, **double** newY)
49. {
50. //cout << "调用了Point的有参数构造函数" << endl;
51. X = newX;
52. Y = newY;
53. }
54. Point::Point(**const** Point& p)
55. {
56. //cout << "调用了Point的拷贝构造函数" << endl;
57. X = p.X;
58. Y = p.Y;
59. }
60. Point::~Point()
61. {
62. //cout << "调用了Point的析构函数" << endl;
63. }
64. **void** Point::show()
65. {
66. cout << "Point is at (" << X << "," << Y << ")." << endl << endl;
67. }
69. **int** main()
70. {
71. Point p(0, 1);
72. cout << "At first,";
73. p.show();
74. p--;
75. cout << "Then \"p--\",";
76. p.show();
77. p++;
78. cout << "Then \"p++\",";
79. p.show();
80. --p;
81. cout << "Then \"--p\",";
82. p.show();
83. ++p;
84. cout << "Then \"++p\",";
85. p.show();
86. }

## 9.综合（服务器）

出于兴趣，将对战功能最先完成，导致不便分开三个独立的程序。故将综合实验三个题目写到了一起，验收时按照助教指示分别演示了各项功能。

此处代码为实现三个题目所有功能的代码。

### 文件1：header.h

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. /\* @课程名：        C++课程设计 题目二 用户注册与平台登录   \*/
3. /\* @作者；         谢睿（2018211208）                      \*/
4. /\* @教师：         王洪波                                 \*/
5. /\* @最后修改时间： 2020/08/30 19：06                        \*/
6. /\* @文件名：        header.h                                \*/
7. /\* @文件说明：       定义少量宏定义                         \*/
8. /\*----------------------------------------------------------\*/
9. /\* @备注：         本程序遵循任天堂公司习惯用语，使用“宝可\*/
10. /\*                  梦(Pokemon)”代指宠物小精灵              \*/
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. #ifndef Header\_h
14. #define Header\_h
16. //#define MAX\_CLIENT 16//最大客户端数量（未使用）
17. #define MAX\_BUF\_SIZE 1024//缓冲区长度
18. #define MAX\_32\_INT 2147483647//设定的最大INT值（不是实际最大INT）
20. #define UNLOGIN 0//未登录
21. #define LOGINED 1//已登陆
22. #define BATTLE 2//对战中（未使用）
24. #define DODGE\_RATE 5//闪避率（百分比）
25. #define CRIT\_RATE 15//暴击率（百分比）
27. #endif

### 文件2：Skills.h

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. /\* @课程名：        C++课程设计 题目二 用户注册与平台登录   \*/
3. /\* @作者；         谢睿（2018211208）                      \*/
4. /\* @教师：         王洪波                                 \*/
5. /\* @最后修改时间： 2020/08/22 17：52                        \*/
6. /\* @文件名：        Skills.h                                \*/
7. /\* @文件说明：       定义各种元素属性以及可学会的技能        \*/
8. /\*----------------------------------------------------------\*/
9. /\* @备注：         本程序遵循任天堂公司习惯用语，使用“宝可\*/
10. /\*                  梦(Pokemon)”代指宠物小精灵              \*/
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. #ifndef Skills\_h
14. #define Skills\_h
16. //元素定义
17. #define NORMAL 0//普通
18. #define FIRE 1//火
19. #define GRASS 2//草
20. #define WATER 3//水
21. #define BUG 4//虫
22. #define FLY 5//飞
23. #define GROUND 6//地
24. #define ELECTRIC 7//电
25. #define ICE 8//冰
26. #define DRAGON 9//龙
28. **class** Skill//技能
29. {
30. **private**:
31. **char** name[32];//技能名
32. **int** elem;//元素属性
33. **int** ex;//额外伤害加成
34. **public**:
35. Skill();//构造函数
36. Skill(**int** type, **const** **char**\* newName, **int** newEx);//构造函数
37. Skill(**const** Skill& s);//拷贝构造函数
38. ~Skill() {}//析构函数
39. Skill& operator =(**const** Skill& m)//赋值重载
40. {
41. **if** (**this** != &m)//自赋值判断
42. {
43. elem = m.elem;
44. strcpy(name, m.name);
45. ex = m.ex;
46. }
47. **return** \***this**;
48. }
49. **int** getElem() { **return** elem; };//获取元素属性
50. **char**\* getName() { **return** name; };//获取技能名
51. **int** getEx() { **return** ex; };//获取额外伤害值
52. };
53. Skill::Skill()//默认招式为普通系攻击“拍打”,无攻击加成
54. {
55. elem = 0;
56. strcpy(name, "拍打");
57. ex = 0;
58. }
59. Skill::Skill(**int** newType, **const** **char**\* newName, **int** newEx)
60. {
61. elem = newType;
62. strcpy(name, newName);
63. ex = newEx;
64. }
65. Skill::Skill(**const** Skill& s)
66. {
67. elem = s.elem;
68. strcpy(name, s.name);
69. ex = s.ex;
70. }
72. **typedef** **struct** skillDataItem//结构体：记录构成一个技能的数据，便于传递
73. {
74. **char** name[32];
75. **int** elem;
76. **int** ex;
77. }SKILL\_DATA\_ITEM;
79. SKILL\_DATA\_ITEM grassSkillDataBase[5] =//草系可学会的技能，初始是：飞叶快刀
80. { {"藤鞭",GRASS,1},
81. {"吸取",GRASS,0},
82. {"日光",GRASS,2},
83. {"花瓣舞",GRASS,1},
84. {"种子机关枪",GRASS,4} };
85. SKILL\_DATA\_ITEM fireSkillDataBase[5] =//火系可学会的技能，初始是：喷火
86. { {"火花",FIRE,1},
87. {"热风",FIRE,0},
88. {"喷射火焰",FIRE,2},
89. {"火焰漩涡",FIRE,1},
90. {"大字爆炎",FIRE,4} };
91. SKILL\_DATA\_ITEM waterSkillDataBase[5] =//水系可学会的技能，初始是：喷水
92. { {"水枪",WATER,1},
93. {"潮漩",WATER,0},
94. {"冲浪",WATER,2},
95. {"泡沫",WATER,1},
96. {"高压水炮",WATER,4} };
97. SKILL\_DATA\_ITEM bugSkillDataBase[5] =//虫系可学会的技能，初始是：撕咬
98. { {"双针",BUG,1},
99. {"飞弹针",BUG,0},
100. {"吸血",BUG,2},
101. {"连斩",BUG,1},
102. {"毒液喷射",BUG,4} };
103. SKILL\_DATA\_ITEM flySkillDataBase[5] =//飞行系可学会的技能，初始是：起风
104. { {"展翅击",FLY,1},
105. {"啄击",FLY,0},
106. {"回旋击",FLY,2},
107. {"狂风",FLY,1},
108. {"神鸟喙",FLY,4} };
109. SKILL\_DATA\_ITEM groundSkillDataBase[5] =//地面系可学会的技能，初始是：击地
110. { {"泼沙",GROUND,1},
111. {"地震",GROUND,0},
112. {"撒菱",GROUND,2},
113. {"掷泥",GROUND,1},
114. {"地裂",GROUND,4} };
115. SKILL\_DATA\_ITEM electricSkillDataBase[5] =//电系可学会的技能，初始是：电击
116. { {"雷电拳",ELECTRIC,1},
117. {"电球",ELECTRIC,0},
118. {"打雷",ELECTRIC,2},
119. {"电磁波",ELECTRIC,1},
120. {"十万伏特",ELECTRIC,4} };
121. SKILL\_DATA\_ITEM iceSkillDataBase[5] =//冰系可学会的技能，初始是：吐冰
122. { {"冰冻拳",ICE,1},
123. {"冰冻光束",ICE,0},
124. {"暴风雪",ICE,2},
125. {"极光束",ICE,1},
126. {"急冻光束",ICE,4} };
127. SKILL\_DATA\_ITEM dragonSkillDataBase[5] =//龙系可学会的技能，初始是：龙爪
128. { {"吐息",DRAGON,1},
129. {"逆鳞",DRAGON,0},
130. {"龙之怒",DRAGON,2},
131. {"流星群",DRAGON,1},
132. {"龙之波动",DRAGON,4} };
134. #endif

### 文件3：server.h

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. /\* @课程名：        C++课程设计 题目二 用户注册与平台登录   \*/
3. /\* @作者；         谢睿（2018211208）                      \*/
4. /\* @教师：         王洪波                                 \*/
5. /\* @最后修改时间： 2020/08/23 17：07                        \*/
6. /\* @文件名：        server.h                                \*/
7. /\* @文件说明：       定义服务端操作                         \*/
8. /\*----------------------------------------------------------\*/
9. /\* @备注：         本程序遵循任天堂公司习惯用语，使用“宝可\*/
10. /\*                  梦(Pokemon)”代指宠物小精灵              \*/
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. #ifndef Server\_h
14. #define Server\_h
16. #include"PokemonList.h"
17. #include"header.h"
18. #include<iostream>
19. #include<string>
20. #include<cstdlib>
21. #include<ctime>
22. #include<stdio.h>
23. #include<string.h>
24. **using** **namespace** std;
26. **extern** **int** userNum;//外部变量
27. **extern** **int** activeUserNum;//外部变量
29. #define GOLD 1//金
30. #define SILVER 2//银
31. #define COPPER 3//铜


35. **class** User//用户
36. {
37. **private**:
38. **char** username[32];//用户名
39. **char** password[32];//密码
40. Pokemon\* pokemonOwned[32];//拥有的精灵，背包最多32只
41. **int** pokemonNum;//拥有精灵数量
42. **int** quantityBadge;//精灵数量徽章
43. **int** advancedBadge;//高级精灵徽章
44. **int** winBattleNum;//胜场数
45. **int** allBattleNum;//总场数
46. **void** distributePokemon(**int** type, **char**\* newName, **int**\* data, **int** index, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//随机分配宝可梦
47. **int** sameUserJudge(**const** **char**\* uName);//重名检测
48. **void** saveData(**const** **char**\* uName, **const** **char**\* uPass);//新建用户数据存入文件
49. **public**:
50. User();//构造
51. User(**const** **char**\* uName, **const** **char**\* uPass, **int** type, **int**& result);//构造
52. User(**const** User& u);//构造
53. ~User();//析构
54. Pokemon\* usingPokemon;//正在使用的宝可梦
55. **int** login(**const** **char**\* uName, **const** **char**\* uPass);//登陆函数
56. **int** loadPokemon();//载入宝可梦
57. **void** showPokemon();//展示宝可梦（测试用）
58. **void** selectPokemon(SOCKET\* clntSock);//选择出战宝可梦
59. **void** getUsername(**char**\* thisName);//获取用户名
60. **void** lossPokemon(SOCKET\* clntSock);//丢弃宝可梦
61. **void** getPokemon(Pokemon\* aPokemon);//获得宝可梦
62. **void** winBattlePlus();//胜场加一
63. **void** allBattlePlus();//总场数加一
64. **void** saveAllToFile();//用户数据储存
65. **double** seeWinRate(**char**\* uName);//展示胜率
66. **int** getquantityBadge();//获取数量徽章
67. **int** getadvancedBadge();//获取高级徽章
68. };
70. **typedef** **struct** clientData//用户简略信息结构
71. {
72. User\* clientUser;//User对象指针
73. SOCKET\* clientSocket;//套接字指针
74. **int** state;//客户状态 0，1，2：未登录，已登陆，对战中
75. clientData\* prevData;//指向上一个客户的指针
76. clientData\* nextData;//指向下一个客户的指针
77. }CLIENT\_DATA;
79. **extern** CLIENT\_DATA\* allClients;//所有在线用户的链表
81. **void** User::distributePokemon(**int** type, **char**\* newName, **int**\* data, **int** index, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
82. {
83. **int** tmpData[9] = { NORMAL,POWER,0,1,2,10,1,2,0 };//初始化数据
84. **char** tmpName[32] = { 0 };
85. **int** tmpStatus[7] = { 0 };
86. **int** typeIndex = 0;
87. **if** (type == 0)//随机分配
88. {
89. typeIndex = \*data;
90. }
91. **else** **if** (type == 1 || type == 2)//按输入数据分配
92. {
93. typeIndex = data[8];
94. strcpy(tmpName, newName);
95. **for** (**int** i = 0; i < 9; i++)
96. {
97. tmpData[i] = data[i];
98. }
99. }
101. **switch** (typeIndex)
102. {
103. **case** 1:
104. tmpData[8] = BLUBASAUR;
105. **if** (type == 0 || type == 2)
106. {
107. strcpy(tmpName, "bulbasaur");
108. pokemonOwned[index] = **new** Bulbasaur(tmpName, tmpData);
109. }
110. **else** **if** (type == 1)
111. {
112. pokemonOwned[index] = **new** Bulbasaur(tmpName, tmpData, newSkillNum, skills);
113. }
114. **break**;
115. **case** 2:
116. tmpData[8] = CHARIZARD;
117. **if** (type == 0 || type == 2)
118. {
119. strcpy(tmpName, "charizard");
120. pokemonOwned[index] = **new** Charizard(tmpName, tmpData);
121. }
122. **else** **if** (type == 1)
123. {
124. pokemonOwned[index] = **new** Charizard(tmpName, tmpData, newSkillNum, skills);
125. }
126. **break**;
127. **case** 3:
128. tmpData[8] = SQUIRTLE;
129. **if** (type == 0 || type == 2)
130. {
131. strcpy(tmpName, "squirtle");
132. pokemonOwned[index] = **new** Squirtle(tmpName, tmpData);
133. }
134. **else** **if** (type == 1)
135. {
136. pokemonOwned[index] = **new** Squirtle(tmpName, tmpData, newSkillNum, skills);
137. }
138. **break**;
139. **case** 4:
140. tmpData[8] = CATERPIE;
141. **if** (type == 0 || type == 2)
142. {
143. strcpy(tmpName, "caterpie");
144. pokemonOwned[index] = **new** Caterpie(tmpName, tmpData);
145. }
146. **else** **if** (type == 1)
147. {
148. pokemonOwned[index] = **new** Caterpie(tmpName, tmpData, newSkillNum, skills);
149. }
150. **break**;
151. **case** 5:
152. tmpData[8] = PIDGEOTTO;
153. **if** (type == 0 || type == 2)
154. {
155. strcpy(tmpName, "pidgeotto");
156. pokemonOwned[index] = **new** Pidgeotto(tmpName, tmpData);
157. }
158. **else** **if** (type == 1)
159. {
160. pokemonOwned[index] = **new** Pidgeotto(tmpName, tmpData, newSkillNum, skills);
161. }
162. **break**;
163. **case** 6:
164. tmpData[8] = DIGLETT;
165. **if** (type == 0 || type == 2)
166. {
167. strcpy(tmpName, "diglett");
168. pokemonOwned[index] = **new** Diglett(tmpName, tmpData);
169. }
170. **else** **if** (type == 1)
171. {
172. pokemonOwned[index] = **new** Diglett(tmpName, tmpData, newSkillNum, skills);
173. }
174. **break**;
175. **case** 7:
176. tmpData[8] = PIKACHU;
177. **if** (type == 0 || type == 2)
178. {
179. strcpy(tmpName, "pikachu");
180. pokemonOwned[index] = **new** Pikachu(tmpName, tmpData);
181. }
182. **else** **if** (type == 1)
183. {
184. pokemonOwned[index] = **new** Pikachu(tmpName, tmpData, newSkillNum, skills);
185. }
186. **break**;
187. **case** 8:
188. tmpData[8] = SNORUNT;
189. **if** (type == 0 || type == 2)
190. {
191. strcpy(tmpName, "snorunt");
192. pokemonOwned[index] = **new** Snorunt(tmpName, tmpData);
193. }
194. **else** **if** (type == 1)
195. {
196. pokemonOwned[index] = **new** Snorunt(tmpName, tmpData, newSkillNum, skills);
197. }
198. **break**;
199. **case** 9:
200. tmpData[8] = DRAGONITE;
201. **if** (type == 0 || type == 2)
202. {
203. strcpy(tmpName, "dragonite");
204. pokemonOwned[index] = **new** Dragonite(tmpName, tmpData);
205. }
206. **else** **if** (type == 1)
207. {
208. pokemonOwned[index] = **new** Dragonite(tmpName, tmpData, newSkillNum, skills);
209. }
210. **break**;
211. **default**:
212. **break**;
213. }
214. pokemonOwned[index]->getStatus(tmpName, tmpStatus);
215. **if** (type == 0 || type == 2)
216. {
217. cout << "获得精灵：" << tmpName << endl << "LV = " << tmpStatus[1] << ", ATK = " << tmpStatus[2] << ", HP = " << tmpStatus[3] << ", DEF = " << tmpStatus[4] << endl;
219. }
220. **else** **if** (type == 1)//按输入数据分配
221. {
222. cout << "载入精灵：" << tmpName << endl << "LV = " << tmpStatus[1] << ", ATK = " << tmpStatus[2] << ", HP = " << tmpStatus[3] << ", DEF = " << tmpStatus[4] << endl;
224. }
225. }
226. **int** User::sameUserJudge(**const** **char**\* uName)
227. {
228. **char** cmpName[32] = { 0 };
229. ifstream userFile("./DATA/userData.txt");
230. userFile.getline(cmpName, 31);
231. userFile.getline(cmpName, 31);
232. **for** (**int** i = 0; i < userNum ; i++)
233. {
234. userFile >> cmpName;
235. //cout << cmpName << endl;
236. **if** (strcmp(uName, cmpName) == 0)//重名
237. {
238. userFile.close();
239. **return** 1;
240. }
241. userFile >> cmpName;
242. **while** (strcmp("##end##", cmpName) != 0)
243. {
244. userFile >> cmpName;
245. //cout << cmpName << endl;
246. }
247. }
248. userFile.close();
249. **return** 0;
250. }
251. **void** User::saveData(**const** **char**\* uName, **const** **char**\* uPass)
252. {
253. ofstream destFile("./DATA/userData.txt", ios::app);
254. destFile << uName << endl;
255. destFile << uPass << endl;
256. destFile << "#winRate#" << endl;
257. destFile << "0 0" << endl;
258. destFile << "#badge#" << endl;
259. destFile << "0 0" << endl;
260. destFile << "#pokemon#" << endl;
261. destFile << "0" << endl;
262. destFile << "##end##" << endl;
263. destFile.close();
265. ifstream userFile("./DATA/userData.txt");
266. ofstream newFile("./DATA/newUserData.txt", ios::out);
267. **char** cmpData[1024];
268. **int** oldNum = 0;
269. userFile >> cmpData;
270. userFile >> oldNum;
271. newFile << "userNum ";
272. newFile << oldNum + 1;
273. **while** (userFile.getline(cmpData, **sizeof**(cmpData)))
274. {
275. newFile << cmpData << endl;
276. }
277. userFile.close();
278. newFile.close();
279. remove("./DATA/userData.txt");
280. //cout << endl << errno << endl;
281. rename("./DATA/newUserData.txt", "./DATA/userData.txt");//重命名
282. //cout << endl << errno << endl;
283. }
284. User::User()
285. {
286. strcpy(username, "user");
287. **char** index[5];
288. itoa(userNum, index, 10);
289. strcat(username, index);
290. strcpy(password, "123456");
291. cout << "创建用户：" << username << " ，密码为：" << password << endl;
292. winBattleNum = 0;
293. allBattleNum = 0;
294. pokemonNum = 0;
295. //srand((int)time(0));
296. **int** randPokemon;
297. **for** (**int** i = 0; i < 3; i++)
298. {
299. randPokemon = rand() % 9 + 1;
300. distributePokemon(0, NULL, &randPokemon, i, 0, NULL);
301. }
302. }
303. User::User(**const** **char**\* uName, **const** **char**\* uPass, **int** type, **int**& result)
304. {
305. **if** (type == 0)//注册
306. {
307. **if** (sameUserJudge(uName))
308. {
309. cout << "用户名重复" << endl;
310. result = 1;//注册失败
311. }
312. **else**
313. {
314. strcpy(username, uName);
315. strcpy(password, uPass);
316. cout << "创建用户：" << username << " ，密码为：" << password << endl;
317. winBattleNum = 0;
318. allBattleNum = 0;
319. saveData(username, password);
320. pokemonNum = 0;
321. quantityBadge = 0;
322. advancedBadge = 0;
323. //srand((int)time(0));
324. **int** randPokemon;
325. **for** (**int** i = 0; i < 3; i++)
326. {
327. randPokemon = rand() % 9 + 1;
328. distributePokemon(0, NULL, &randPokemon, i, 0, NULL);
329. pokemonNum++;
330. }
331. saveAllToFile();
332. result = 0;//注册成功
333. }
334. }
335. **else** **if** (type == 1)//登陆
336. {
337. strcpy(username, uName);
338. strcpy(password, uPass);
339. cout << "登陆用户：" << username << endl;
340. pokemonNum = 0;
341. result = login(uName, uPass);
342. }
344. }
345. User::User(**const** User& u)
346. {
347. cout << "调用user的拷贝构造函数" << endl;
348. system("pause");
350. strcpy(username, u.username);
351. strcpy(password, u.password);
352. pokemonNum = u.pokemonNum;
353. }
354. User::~User()
355. {
356. **for** (**int** i = 0; i < pokemonNum; i++)
357. {
358. //      delete[] pokemonOwned[i];
359. }
360. }
361. **int** User::login(**const** **char**\* uName, **const** **char**\* uPass)
362. {
363. **char** cmpName[500] = { 0 };
364. **char** cmpPass[32] = { 0 };
365. **int** multiLoginFlag = 0;
366. ifstream userFile("./DATA/userData.txt");
367. userFile.getline(cmpName, 500);
368. userFile.getline(cmpName, 500);
369. **for** (**int** i = 0; i < userNum; i++)
370. {
371. userFile >> cmpName;
372. **if** (strcmp(uName, cmpName) == 0)//存在此用户
373. {
374. userFile >> cmpPass;
375. **if** (strcmp(uPass, cmpPass) == 0)//如果密码正确
376. {
377. CLIENT\_DATA\* showData = allClients;
378. **char** thisName[32] = { 0 };
379. **while** (showData)
380. {
381. **if** (showData->state == LOGINED)
382. {
383. showData->clientUser->getUsername(thisName);
384. **if** (strcmp(thisName, uName) == 0)
385. {
386. multiLoginFlag = 1;
387. **break**;
388. }
389. }
390. showData = showData->nextData;
391. }
392. **if** (multiLoginFlag == 0)
393. {
394. userFile >> cmpName;
395. **while** (strcmp("#winRate#", cmpName) != 0)//定位比赛场数
396. {
397. userFile >> cmpName;
398. }
399. userFile >> winBattleNum;
400. userFile >> allBattleNum;
401. **while** (strcmp("#badge#", cmpName) != 0)//定位徽章
402. {
403. userFile >> cmpName;
404. }
405. userFile >> quantityBadge;
406. userFile >> advancedBadge;
407. //cout << "登陆成功！欢迎回来：" << uName << endl;
408. userFile.close();
409. **return** 0;
410. }
411. **else**
412. {
413. cout << "重复登陆" << endl;
414. userFile.close();
415. **return** 3;
416. }
417. }
418. **else**
419. {
420. cout << "密码错误" << endl;
421. userFile.close();
422. **return** 2;
423. }
424. }
425. userFile >> cmpName;
426. **while** (strcmp("##end##", cmpName) != 0)
427. {
428. userFile >> cmpName;
429. }
430. }
431. cout << "用户不存在" << endl;
432. userFile.close();
433. **return** 1;
434. }
435. **int** User::loadPokemon()
436. {
437. **char** cmpName[32] = { 0 };
438. **char** pokeName[32] = { 0 };
439. **int** data[9] = { 0 };
440. SKILL\_DATA\_ITEM skills[4] = { 0 };
441. **int** skillNum = 0;
442. **char** skillName[32] = { 0 };
443. ifstream userFile("./DATA/userData.txt");
444. userFile.getline(cmpName, 31);
445. userFile.getline(cmpName, 31);
446. **for** (**int** i = 0; i < userNum; i++)
447. {
448. userFile >> cmpName;
449. //cout << cmpName << endl;
450. **if** (strcmp(username, cmpName) == 0)//寻找用户数据位置
451. {
452. **while** (strcmp("#pokemon#", cmpName) != 0)//定位宝可梦数据位置
453. {
454. userFile >> cmpName;
455. }
456. **break**;
457. }
458. **while** (strcmp("##end##", cmpName) != 0)
459. {
460. userFile >> cmpName;
461. }
462. }
463. userFile >> pokemonNum; //cout << pokemonNum << endl;
464. **for** (**int** i = 0; i < pokemonNum; i++)
465. {
466. userFile >> pokeName; //cout << pokeName << endl;
467. userFile >> data[0]; //cout << data[0] << endl; //元素
468. userFile >> data[1];//类型
469. userFile >> pokeName; //cout << pokeName << endl;
470. userFile >> data[2];//EXP
471. userFile >> data[3];//LV
472. userFile >> data[4];//ATK
473. userFile >> data[5];//HP
474. userFile >> data[6];//DEF
475. userFile >> data[7];//SPEED
476. userFile >> data[8];//Index
478. userFile >> skillNum;
479. **for** (**int** j = 0; j < skillNum; j++)
480. {
481. userFile >> skillName;// cout << skillName << endl;
482. strcpy(skills[j].name, skillName);
483. userFile >> skills[j].elem;
484. userFile >> skills[j].ex;
485. }
486. distributePokemon(1, pokeName, data, i, skillNum, skills);
487. }
488. userFile.close();
489. **return** 0;
490. }
491. **void** User::showPokemon()
492. {
493. **char** tmpName[32] = { 0 };
494. **int** tmpStatus[7] = { 0 };
495. **for** (**int** i = 0; i < pokemonNum; i++)
496. {
497. pokemonOwned[i]->getStatus(tmpName, tmpStatus);
498. cout << "宝可梦：" << tmpName << endl << "LV = " << tmpStatus[1] << ", ATK = " << tmpStatus[2] << ", HP = " << tmpStatus[3] << ", DEF = " << tmpStatus[4] << endl;
500. }
501. }
502. **void** User::selectPokemon(SOCKET\* clntSock)
503. {
504. **int** sendLen;
505. **int** receiveLen;
506. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
507. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
508. **char** tmpStr[100];
510. **char** tmpName[32] = { 0 };
511. **int** tmpStatus[7] = { 0 };
512. **int** select = 0;
513. **int** flag = 1;
514. cout << "用户选择宝可梦：\n";
515. //cout << "您有以下宝可梦：" << endl;
516. strcpy(sendBuf, "您有以下宝可梦：\n");
517. **for** (**int** i = 0; i < pokemonNum; i++)
518. {
519. pokemonOwned[i]->getStatus(tmpName, tmpStatus);
520. //cout << i+1 << ". " << tmpName << " " << "LV = " << tmpStatus[1] << ", ATK = " << tmpStatus[2] << ", HP = " << tmpStatus[3] << ", DEF = " << tmpStatus[4] << endl;
521. itoa(i + 1, tmpStr, 10);
522. strcat(sendBuf, tmpStr);
523. memset(tmpStr, 0, **sizeof**(tmpStr));
524. strcat(sendBuf, ". ");
525. strcat(sendBuf, tmpName);
526. strcat(sendBuf, " LV= ");
527. itoa(tmpStatus[1], tmpStr, 10);
528. strcat(sendBuf, tmpStr);
529. memset(tmpStr, 0, **sizeof**(tmpStr));
530. strcat(sendBuf, ", ATK = ");
531. itoa(tmpStatus[2], tmpStr, 10);
532. strcat(sendBuf, tmpStr);
533. memset(tmpStr, 0, **sizeof**(tmpStr));
534. strcat(sendBuf, ", HP = ");
535. itoa(tmpStatus[3], tmpStr, 10);
536. strcat(sendBuf, tmpStr);
537. memset(tmpStr, 0, **sizeof**(tmpStr));
538. strcat(sendBuf, ", DEF = ");
539. itoa(tmpStatus[4], tmpStr, 10);
540. strcat(sendBuf, tmpStr);
541. memset(tmpStr, 0, **sizeof**(tmpStr));
542. strcat(sendBuf, "\n");
543. }
544. //cout << "选择一只宝可梦进行对战：";
545. strcat(sendBuf, "选择一只宝可梦进行对战：");
546. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
547. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
548. **while** (flag)
549. {
550. //cin >> select;
551. //select--;
552. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
553. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
554. select = atoi(receiveBuf);
555. select--;
556. **if** (select >= 0 && select < pokemonNum)
557. {
558. flag = 0;
559. }
560. **else**
561. {
562. //cout << "编号错误，请重新输入：";
563. strcat(sendBuf, "编号错误，请重新输入：");
564. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
565. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
566. Sleep(10);
567. }
568. }
569. cout << "选择了" << select + 1 << "号宝可梦" << endl;
570. usingPokemon = pokemonOwned[select];
571. strcat(sendBuf, "#selectEnd#");
572. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
573. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
574. Sleep(10);
575. }
577. **void** User::getUsername(**char**\* thisName)
578. {
579. strcpy(thisName, username);
580. }
582. **void** User::lossPokemon(SOCKET\* clntSock)
583. {
584. **int** sendLen;
585. **int** receiveLen;
586. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
587. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
588. **char** tmpStr[100];
590. **char** tmpName[32] = { 0 };
591. **int** tmpStatus[7] = { 0 };
592. **int** select = 0;
593. **int** flag = 1;
594. **int** decided = -1;
595. **int** sendOutNum = 0;
596. //cout << "您有以下宝可梦：" << endl;
598. strcpy(sendBuf, "#lossPokemon#");
599. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
600. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
601. Sleep(10);
603. strcpy(sendBuf, "请从以下宝可梦中选一只送出：\n");
604. **int** sendOut[3] = { -1,-1,-1 };
605. **if** (pokemonNum >= 3)
606. {
607. sendOutNum = 3;
608. //srand((int)time(0));
609. **for** (**int** j = 0; j < 3; j++)
610. {
611. flag = 1;
612. **while** (flag)
613. {
614. flag = 0;
615. sendOut[j] = rand() % pokemonNum;
616. **for** (**int** k = 0; k < j; k++)
617. {
618. **if** (sendOut[j] == sendOut[k]) flag = 1;
619. }
620. }
621. }
622. }
623. **else** **if** (pokemonNum == 2)
624. {
625. sendOutNum = 2;
626. sendOut[0] = 0;
627. sendOut[1] = 1;
628. }
629. **else** **if** (pokemonNum == 1)
630. {
631. sendOutNum = 1;
632. sendOut[0] = 0;
633. }
634. **for** (**int** i = 0; (i < 3 && (sendOut[i] != -1)); i++)
635. {
636. pokemonOwned[sendOut[i]]->getStatus(tmpName, tmpStatus);
637. itoa(i + 1, tmpStr, 10);
638. strcat(sendBuf, tmpStr);
639. memset(tmpStr, 0, **sizeof**(tmpStr));
640. strcat(sendBuf, ". ");
641. strcat(sendBuf, tmpName);
642. strcat(sendBuf, " LV= ");
643. itoa(tmpStatus[1], tmpStr, 10);
644. strcat(sendBuf, tmpStr);
645. memset(tmpStr, 0, **sizeof**(tmpStr));
646. strcat(sendBuf, ", ATK = ");
647. itoa(tmpStatus[2], tmpStr, 10);
648. strcat(sendBuf, tmpStr);
649. memset(tmpStr, 0, **sizeof**(tmpStr));
650. strcat(sendBuf, ", HP = ");
651. itoa(tmpStatus[3], tmpStr, 10);
652. strcat(sendBuf, tmpStr);
653. memset(tmpStr, 0, **sizeof**(tmpStr));
654. strcat(sendBuf, ", DEF = ");
655. itoa(tmpStatus[4], tmpStr, 10);
656. strcat(sendBuf, tmpStr);
657. memset(tmpStr, 0, **sizeof**(tmpStr));
658. strcat(sendBuf, "\n");
659. }
660. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
661. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
662. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
663. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
664. decided = atoi(receiveBuf);
665. decided--;
666. **while** (1)
667. {
668. **if** (decided >= 0 && decided < sendOutNum)
669. {
670. **break**;
671. }
672. **else**
673. {
674. strcpy(sendBuf, "输入错误，请重新输入：");
675. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
676. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
677. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
678. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
679. decided = atoi(receiveBuf);
680. }
681. }
682. //delete pokemonOwned[sendOut[decided]];
683. **for** (**int** j = sendOut[decided]; j < pokemonNum; j++)
684. {
685. pokemonOwned[j] = pokemonOwned[j + 1];
686. }
687. pokemonOwned[pokemonNum - 1] = NULL;
688. pokemonNum--;
689. **if** (pokemonNum == 0)
690. {
691. pokemonNum++;
692. //srand((int)time(0));
693. **int** randPokemon= rand() % 9 + 1;
694. distributePokemon(0, NULL, &randPokemon, 0, 0, NULL);
695. strcpy(sendBuf, "您的宝可梦已全部送出，系统自动分配一只1级宝可梦：\n");
696. **for** (**int** i = 0; i < pokemonNum; i++)
697. {
698. pokemonOwned[i]->getStatus(tmpName, tmpStatus);
699. //cout << i + 1 << ". " << tmpName << " " << "LV = " << tmpStatus[1] << ", ATK = " << tmpStatus[2] << ", HP = " << tmpStatus[3] << ", DEF = " << tmpStatus[4] << endl;
700. itoa(i + 1, tmpStr, 10);
701. strcat(sendBuf, tmpStr);
702. memset(tmpStr, 0, **sizeof**(tmpStr));
703. strcat(sendBuf, ". ");
704. strcat(sendBuf, tmpName);
705. strcat(sendBuf, " LV= ");
706. itoa(tmpStatus[1], tmpStr, 10);
707. strcat(sendBuf, tmpStr);
708. memset(tmpStr, 0, **sizeof**(tmpStr));
709. strcat(sendBuf, ", ATK = ");
710. itoa(tmpStatus[2], tmpStr, 10);
711. strcat(sendBuf, tmpStr);
712. memset(tmpStr, 0, **sizeof**(tmpStr));
713. strcat(sendBuf, ", HP = ");
714. itoa(tmpStatus[3], tmpStr, 10);
715. strcat(sendBuf, tmpStr);
716. memset(tmpStr, 0, **sizeof**(tmpStr));
717. strcat(sendBuf, ", DEF = ");
718. itoa(tmpStatus[4], tmpStr, 10);
719. strcat(sendBuf, tmpStr);
720. memset(tmpStr, 0, **sizeof**(tmpStr));
721. strcat(sendBuf, "\n");
722. }
723. strcat(sendBuf, "请给其起名：");
724. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
725. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
726. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
727. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
728. pokemonOwned[0]->setName(receiveBuf);
729. }
730. strcpy(sendBuf, "#lossEnd#");
731. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
732. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
733. Sleep(10);
734. }
735. **void** User::getPokemon(Pokemon\* aPokemon)
736. {
737. **char** tmpName[32] = { 0 };
738. **int** tmpStatus[9] = { 0 };
739. aPokemon->getAllInf(tmpName, tmpStatus);
740. distributePokemon(2, tmpName, tmpStatus, pokemonNum, 0, NULL);
741. pokemonNum++;
742. }
744. **void** User::winBattlePlus()
745. {
746. winBattleNum++;
747. }
748. **void** User::allBattlePlus()
749. {
750. allBattleNum++;
751. }
752. **void** User::saveAllToFile()
753. {
754. ifstream userFile("./DATA/userData.txt");
755. ofstream newFile("./DATA/newUserData.txt", ios::out);
756. **char** cmpData[1024];
757. **int** finded = 0;
758. **int** lv15Num = 0;
759. **char** tmpName[32] = { 0 };
760. **int** tmpStatus[9] = { 0 };
761. **int** tmpInt;
762. SKILL\_DATA\_ITEM skills[4] = { 0 };
763. **while** (userFile.getline(cmpData, **sizeof**(cmpData)))
764. {
765. newFile << cmpData << endl;
766. **if** (strcmp("##end##", cmpData) == 0)
767. {
768. userFile.getline(cmpData, **sizeof**(cmpData));
769. newFile << cmpData << endl;//用户名
770. **if** (strcmp(username, cmpData) == 0)//找到用户数据
771. {
772. userFile.getline(cmpData, **sizeof**(cmpData));
773. newFile << cmpData << endl;//密码
774. newFile << "#winRate#" << endl;
775. newFile << winBattleNum << " " << allBattleNum << endl;
776. newFile << "#badge#" << endl;
777. **if** (pokemonNum >= 0 && pokemonNum < 5)
778. {
779. newFile << "0" << " ";
780. }
781. **else** **if** (pokemonNum >= 5 && pokemonNum < 10)
782. {
783. newFile << "3" << " ";
784. }
785. **else** **if** (pokemonNum >= 10 && pokemonNum < 20)
786. {
787. newFile << "2" << " ";
788. }
789. **else** **if** (pokemonNum >= 20)
790. {
791. newFile << "1" << " ";
792. }
793. **for** (**int** i = 0; i < pokemonNum; i++)//计算15级宝可梦个数
794. {
795. pokemonOwned[i]->getStatus(tmpName, tmpStatus);
796. **if** (tmpStatus[1] == 15) lv15Num++;
797. }
798. **if** (lv15Num >= 0 && lv15Num < 5)
799. {
800. newFile << "0" << endl;
801. }
802. **else** **if** (lv15Num >= 5 && lv15Num < 10)
803. {
804. newFile << "3" << endl;
805. }
806. **else** **if** (lv15Num >= 10 && lv15Num < 20)
807. {
808. newFile << "2" << endl;
809. }
810. **else** **if** (lv15Num >= 20)
811. {
812. newFile << "1" << endl;
813. }
814. newFile << "#pokemon#" << endl;
815. newFile << pokemonNum << endl;
816. **for** (**int** i = 0; i < pokemonNum; i++)//存储每个宝可梦
817. {
818. pokemonOwned[i]->getAllInf(tmpName, tmpStatus);
819. **switch** (tmpStatus[8])
820. {
821. **case**  BLUBASAUR:
822. newFile << "blubasaur ";
823. **break**;
824. **case** CHARIZARD:
825. newFile << "charizard ";
826. **break**;
827. **case** SQUIRTLE:
828. newFile << "squirtle ";
829. **break**;
830. **case** CATERPIE:
831. newFile << "caterpie ";
832. **break**;
833. **case** PIDGEOTTO:
834. newFile << "pidgeotto ";
835. **break**;
836. **case** DIGLETT:
837. newFile << "diglett ";
838. **break**;
839. **case** PIKACHU:
840. newFile << "pikachu ";
841. **break**;
842. **case** SNORUNT:
843. newFile << "snorunt ";
844. **break**;
845. **case** DRAGONITE:
846. newFile << "dragonite ";
847. **break**;
848. dafault:
849. **break**;
850. }
851. newFile << tmpStatus[0]; //cout << data[0] << endl; //元素
852. newFile << " ";
853. newFile << tmpStatus[1];//类型
854. newFile << " ";
855. newFile << tmpName; //cout << pokeName << endl;
856. newFile << " ";
857. newFile << tmpStatus[2];//EXP
858. newFile << " ";
859. newFile << tmpStatus[3];//LV
860. newFile << " ";
861. newFile << tmpStatus[4];//ATK
862. newFile << " ";
863. newFile << tmpStatus[5];//HP
864. newFile << " ";
865. newFile << tmpStatus[6];//DEF
866. newFile << " ";
867. newFile << tmpStatus[7];//SPEED
868. newFile << " ";
869. newFile << tmpStatus[8];//Index
870. newFile << "\n";
872. pokemonOwned[i]->getSkills(tmpInt, skills);
873. newFile << tmpInt << " ";
874. **for** (**int** j = 0; j < tmpInt; j++)
875. {
876. newFile << skills[j].name << " ";
877. newFile << skills[j].elem << " ";
878. newFile << skills[j].ex << " ";
879. }
880. newFile << "\n";
881. }
882. newFile << "##end##" << endl;
883. userFile.getline(cmpData, **sizeof**(cmpData));
884. **while** (strcmp("##end##", cmpData) != 0)
885. {
886. userFile.getline(cmpData, **sizeof**(cmpData));
887. }
888. }
889. }
890. }
891. userFile.close();
892. newFile.close();
893. remove("./DATA/userData.txt");
894. //cout << endl << errno << endl;
895. rename("./DATA/newUserData.txt", "./DATA/userData.txt");//重命名
896. //cout << endl << errno << endl;
897. }
899. **double** User::seeWinRate(**char**\* uName)
900. {
901. **char** cmpName[32] = { 0 };
902. ifstream userFile("./DATA/userData.txt");
903. userFile.getline(cmpName, 31);
904. userFile.getline(cmpName, 31);
905. **int** winNum = 0;
906. **int** allNum = 0;
907. **int** findFlag = 0;
908. **double** rate = 0;
909. **for** (**int** i = 0; i < userNum; i++)
910. {
911. userFile >> cmpName;
912. **if** (strcmp(uName, cmpName) == 0)//寻找用户数据位置
913. {
914. findFlag = 1;
915. **while** (strcmp("#winRate#", cmpName) != 0)//定位战斗数据位置
916. {
917. userFile >> cmpName;
918. }
919. **break**;
920. }
921. **while** (strcmp("##end##", cmpName) != 0)
922. {
923. userFile >> cmpName;
924. }
925. }
926. **if** (findFlag == 1)
927. {
928. userFile >> winNum;
929. userFile >> allNum;
930. **if** (allNum == 0)
931. {
932. **return** 0;
933. }
934. **else**
935. {
936. rate = (**double**)winNum / (**double**)allNum;
937. **return** rate;
938. }
939. }
940. **else**
941. {
942. **return** -1;
943. }
944. }
945. **int**  User::getquantityBadge()
946. {
947. **return** quantityBadge;
948. }
949. **int**  User::getadvancedBadge()
950. {
951. **return** advancedBadge;
952. }
954. #endif

### 文件4：PokemonList.h

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. /\* @课程名：        C++课程设计 题目二 用户注册与平台登录   \*/
3. /\* @作者；         谢睿（2018211208）                      \*/
4. /\* @教师：         王洪波                                 \*/
5. /\* @最后修改时间： 2020/08/21 20：09                        \*/
6. /\* @文件名：        PokemonList.h                           \*/
7. /\* @文件说明：       定义各种宝可梦类                        \*/
8. /\*----------------------------------------------------------\*/
9. /\* @备注：         本程序遵循任天堂公司习惯用语，使用“宝可\*/
10. /\*                  梦(Pokemon)”代指宠物小精灵              \*/
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. #ifndef Pokemon\_List\_h
14. #define Pokemon\_List\_h
16. #include<iostream>
17. #include<string>
18. #include<cstdlib>
19. #include<ctime>
20. #include<string.h>
21. #include"Skills.h"
22. #include"header.h"
24. **using** **namespace** std;
26. //属性定义
27. #define POWER 0//力量型
28. #define BEARABLE 1//肉盾型
29. #define DEDENSIVE 2//防御型
30. #define AGILE 3//敏捷型
32. //速度定义
33. //#define SUPER 0//极快
34. #define QUICK 1//快
35. #define COMMON 2//一般
36. #define SLOW 3//缓慢
38. //常量定义
39. #define MAX\_LEVEL 15//最大等级
40. #define MAX\_EXP 4190//最大经验值
41. #define MAX\_SKILL 4//最大学习技能数
42. #define MAX\_TYPE 2//宝可梦种类
44. //图鉴
45. #define BLUBASAUR 1 //妙蛙种子
46. #define CHARIZARD 2 //喷火龙
47. #define SQUIRTLE 3  //杰尼龟
48. #define CATERPIE 4  //绿毛虫
49. #define PIDGEOTTO 5 //比比鸟
50. #define DIGLETT 6   //地鼠
51. #define PIKACHU 7   //皮卡丘
52. #define SNORUNT 8   //雪童子
53. #define DRAGONITE 9 //快龙
55. //外部变量定义
56. **extern** **int** expNeeded[14];
57. **extern** **double** effectiveness[10][10];
59. **class** Pokemon//宝可梦
60. {
61. **protected**:
62. **int** elem;//元素属性
63. **int** type;//宝可梦属性(4种)
64. **char** name[32];//宝可梦名字
65. **int** EXP;//宝可梦经验值
66. **int** LV;//宝可梦等级
67. **int** ATK;//宝可梦攻击属性值
68. **int** HP;//宝可梦生命属性值
69. **int** DEF;//宝可梦防御属性值
70. **int** SPEED;//宝可梦速度（敏捷）属性值
71. Skill\* skill[4];//每只宝可梦同时只能学会四个技能
72. **int** skillNum;//记录已学会的技能数
73. **int** globalIndex;//图鉴索引号
74. **int** selectSkill(SOCKET\* clntSock, **int** atkMode);//选择使用的技能
75. **int** deleteSkill(SOCKET\* clntSock);//选择删除的技能
76. **void** showEffect(**const** **double** effect);//显示效果
77. **private**:
78. **int** fullHP;//满生命的值
79. **public**:
80. Pokemon();//构造函数
81. Pokemon(**const** **char**\* newName, **const** **int**\* data);//构造函数
82. Pokemon(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
83. Pokemon(**const** Pokemon& p);//拷贝构造函数
84. ~Pokemon();//析构函数
85. **int** getElem() { **return** elem; };//获取元素属性
86. **int** getType() { **return** type; };//获取属性
87. **int** getExp() { **return** EXP; };//获取经验值
88. **int** getLevel() { **return** LV; };//获取当前等级
89. **int** getATK() { **return** ATK; };//获取当前攻击力
90. **int** getHP() { **return** HP; };//获取当前生命值
91. **int** getDEF() { **return** DEF; };//获取当前防御值
92. **int** getSPEED() { **return** SPEED; };//获取当前速度值
93. **void** changeLevel(**int** d) { LV += d; };//等级变更d级
94. **void** changeATK(**int** d) { ATK += d; };//攻击变更d点
95. **void** changeHP(**int** d) { HP += d; };//生命变更d点
96. **void** changeDEF(**int** d) { DEF += d; };//防御变更d点
97. **void** changeSPEED(**int** d) { SPEED += d; };//速度变更d点
98. **void** getSkills(**int**& getNum, SKILL\_DATA\_ITEM\* getskill);
99. **void** getStatus(**char**\* thisName, **int**\* thisStatus);//获取宝可梦信息
100. **void** getAllInf(**char**\* thisName, **int**\* thisStatus);//获取宝可梦全部信息
101. **virtual** **void** attack(Pokemon& target, **int** atkMode, SOCKET\* clntSock);//攻击函数 (参数：目标)
102. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
103. **virtual** **void** levelUp(SOCKET\* clntSock, **int** getExp);//升级函数 (参数：获取的经验值)
104. **void** recover();//恢复函数
105. **void** setName(**char**\* newName);//设置宝可梦昵称（新获得的初级LV1）
106. };
107. Pokemon::Pokemon()
108. {
109. elem = NORMAL;//默认为普通元素属性
110. type = POWER;//默认为力量型
111. strcpy(name, "noName");
112. EXP = 0;
113. LV = 1;
114. ATK = 1;
115. HP = 4;
116. DEF = 0;
117. SPEED = SLOW;
118. fullHP = HP;
119. skill[0] = NULL;
120. skill[1] = NULL;
121. skill[2] = NULL;
122. skill[3] = NULL;
123. skillNum = 0;//不给予任何技能
124. globalIndex = 0;
125. }
126. Pokemon::Pokemon(**const** **char**\* newName, **const** **int**\* data)
127. {
128. elem = data[0];
129. type = data[1];
130. strcpy(name, newName);
131. EXP = data[2];
132. LV = data[3];
133. ATK = data[4];
134. HP = data[5];
135. DEF = data[6];
136. SPEED = data[7];
137. fullHP = HP;
138. skill[0] = **new** Skill(0, "拍打", 0);//给予初始技能“拍打”
139. skillNum = 1;
140. globalIndex = data[8];
141. }
142. Pokemon::Pokemon(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
143. {
144. elem = data[0];
145. type = data[1];
146. strcpy(name, newName);
147. EXP = data[2];
148. LV = data[3];
149. ATK = data[4];
150. HP = data[5];
151. DEF = data[6];
152. SPEED = data[7];
153. globalIndex = data[8];
154. fullHP = HP;
155. skillNum = newSkillNum;
156. **for** (**int** i = 0; i < skillNum; i++)
157. {
158. skill[i] = **new** Skill(skills[i].elem, skills[i].name, skills[i].ex);
159. }
160. }
161. Pokemon::Pokemon(**const** Pokemon& p)
162. {
163. elem = p.elem;
164. type = p.type;
165. strcpy(name, p.name);
166. EXP = p.EXP;
167. LV = p.LV;
168. ATK = p.ATK;
169. HP = p.HP;
170. DEF = p.DEF;
171. SPEED = p.SPEED;
172. fullHP = HP;
173. **for** (**int** i = 0; i < p.skillNum; i++)
174. {
175. skill[i] = **new** Skill();
176. (\*skill[i]) = \*(p.skill[i]);
177. }//正确性待验证
178. skillNum = p.skillNum;
179. globalIndex = p.globalIndex;
180. }
181. Pokemon::~Pokemon()
182. {
183. **for** (**int** i = skillNum - 1; i >= 0; i--)
184. {
185. **delete**[] skill[i];
186. }
187. }
188. **int** Pokemon::selectSkill(SOCKET\* clntSock, **int** atkMode)
189. {
190. **int** sendLen;
191. **int** receiveLen;
192. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
193. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
194. **char** tmpStr[100];
196. **int** skillIndex = 0;
198. **if** (atkMode == 0)
199. {
200. //cout << name << "会使用" << skillNum << "种技能：";
201. strcpy(sendBuf, name);
202. strcat(sendBuf, "会使用");
203. itoa(skillNum, tmpStr, 10);
204. strcat(sendBuf, tmpStr);
205. memset(tmpStr, 0, **sizeof**(tmpStr));
206. strcat(sendBuf, "种技能：");
207. **for** (**int** i = 0; i < skillNum; i++)
208. {
209. //cout << i + 1 << "." << skill[i]->getName() << " ";
210. itoa(i + 1, tmpStr, 10);
211. strcat(sendBuf, tmpStr);
212. memset(tmpStr, 0, **sizeof**(tmpStr));
213. strcat(sendBuf, ".");
214. strcat(sendBuf, skill[i]->getName());
215. strcat(sendBuf, " ");
216. }
217. //cout << endl << "请输入要使用的技能编号：";
218. strcat(sendBuf, "\n请输入要使用的技能编号：");
219. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
220. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
221. **while** (1)
222. {
223. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
224. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
225. skillIndex = atoi(receiveBuf);
226. skillIndex--;
227. **if** (skillIndex >= 0 && skillIndex < skillNum) **break**;//如果范围正确
228. **else**
229. {
230. strcpy(sendBuf, "您输入了错误的编号，请重新输入：");
231. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
232. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
233. Sleep(10);
234. }
236. //测试用（服务器输入）
237. /\*
238. cin >> skillIndex;
239. skillIndex--;
240. if (cin.good())//如果输入是整数
241. {
242. if (skillIndex >= 0 && skillIndex < skillNum)//如果范围正确
243. {
244. notAttackedFlag = 0;
245. }
246. else//如果范围不正确
247. {
248. cout << "您输入了错误的编号，请重新输入：";
249. }
250. }
251. else//如果输入不是整数
252. {
253. cout << "输入不是整数，请重新输入：";
254. cin.clear();//清除错误标志
255. }
256. cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
257. \*/
258. }
259. strcpy(sendBuf, "#skillOver#");
260. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
261. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
262. }
263. **else**
264. {
265. Sleep(3);
266. //srand((int)time(0));
267. skillIndex = rand() % skillNum;
268. }
269. **return** skillIndex;
270. }
271. **int** Pokemon::deleteSkill(SOCKET\* clntSock)
272. {
273. **int** sendLen;
274. **int** receiveLen;
275. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
276. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
277. **char** tmpStr[100];
279. **int** skillChange = 0;
280. **int** flag = 1;
281. //cout << name << "会以下技能：";
282. strcpy(sendBuf, name);
283. strcat(sendBuf, "会以下技能：");
284. **for** (**int** i = 0; i < skillNum; i++)
285. {
286. //cout << i + 1 << "." << skill[i]->getName() << " ";
287. itoa(i + 1, tmpStr, 10);
288. strcat(sendBuf, tmpStr);
289. memset(tmpStr, 0, **sizeof**(tmpStr));
290. strcat(sendBuf, ".");
291. strcat(sendBuf, skill[i]->getName());
292. strcat(sendBuf, " ");
293. }
294. //cout << endl << "请输入要忘记的技能编号(输入0放弃忘记)：";
295. strcat(sendBuf, "请输入要忘记的技能编号(输入0放弃忘记)：");
296. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
297. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
299. **while** (flag)
300. {
301. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
302. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
303. skillChange = atoi(receiveBuf);
304. skillChange--;
305. **if** (skillChange >= -1 && skillChange < skillNum)//如果范围正确
306. {
307. flag = 0;
308. }
309. **else**//如果范围不正确
310. {
311. //cout << "您输入了错误的编号，请重新输入：";
312. strcpy(sendBuf, "您输入了错误的编号，请重新输入：");
313. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
314. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
315. }
317. }
318. **return** skillChange;
319. }
320. **void** Pokemon::showEffect(**double** effect)
321. {
322. **if** (effect == 1)
323. {
324. //cout << endl;
325. }
326. **else** **if** (effect == 2)
327. {
328. //cout << "效果拔群" << endl;
329. }
330. **else** **if** (effect == 0.5)
331. {
332. //cout << "效果一般" << endl;
333. }
334. }
335. **void** Pokemon::getSkills(**int**& getNum, SKILL\_DATA\_ITEM\* getskill)
336. {
337. getNum = skillNum;
338. **for** (**int** j = 0; j < skillNum; j++)
339. {
340. strcpy(getskill[j].name, skill[j]->getName());
341. getskill[j].elem = skill[j]->getElem();
342. getskill[j].ex = skill[j]->getEx();
343. }
344. }
345. **void** Pokemon::getStatus(**char**\* thisName, **int**\* thisStatus)
346. {
347. strcpy(thisName, name);
348. thisStatus[0] = type;
349. thisStatus[1] = LV;
350. thisStatus[2] = ATK;
351. thisStatus[3] = HP;
352. thisStatus[4] = DEF;
353. thisStatus[5] = SPEED;
354. thisStatus[6] = EXP;
355. }
356. **void** Pokemon::getAllInf(**char**\* thisName, **int**\* thisStatus)
357. {
358. strcpy(thisName, name);
359. thisStatus[0] = elem;
360. thisStatus[1] = type;
361. thisStatus[2] = EXP;
362. thisStatus[3] = LV;
363. thisStatus[4] = ATK;
364. thisStatus[5] = HP;
365. thisStatus[6] = DEF;
366. thisStatus[7] = SPEED;
367. thisStatus[8] = globalIndex;
368. }
369. **void** Pokemon::attack(Pokemon& target, **int** atkMode, SOCKET\* clntSock)
370. {
371. **int** sendLen;
372. **int** receiveLen;
373. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
374. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
375. **char** tmpStr[100];
377. **int** skillIndex = selectSkill(clntSock,atkMode);//选择技能
378. **double** effect = effectiveness[skill[skillIndex]->getElem()][target.getElem()];//效果
380. **int** damage = (**int**)((ATK + skill[skillIndex]->getEx()) \* effect - target.getDEF());//计算造成的伤害值
382. Sleep(3);
383. //srand(time(0));
384. **int** dodge = rand() % 100;//闪避
385. **int** crit = rand() % 100;//暴击
387. **if** (damage <= 0)
388. {
389. damage = 0;
390. //cout << "毫无效果" << endl;
391. strcpy(sendBuf, name);
392. strcat(sendBuf, "使用");
393. strcat(sendBuf, skill[skillIndex]->getName());
394. strcat(sendBuf, "， 毫无效果\n");
395. }
396. **else**
397. {
398. **if** (dodge <= DODGE\_RATE)//闪避
399. {
400. damage = 0;
401. strcpy(sendBuf, name);
402. strcat(sendBuf, "使用");
403. strcat(sendBuf, skill[skillIndex]->getName());
404. strcat(sendBuf, "，但被对方闪避而没有命中，造成0点伤害\n");
405. }
406. **else**
407. {
408. **if** (crit <= CRIT\_RATE)//暴击
409. {
410. damage = damage \* 2;
411. strcpy(sendBuf, name);
412. strcat(sendBuf, "使用");
413. strcat(sendBuf, skill[skillIndex]->getName());
414. strcat(sendBuf, "，技能暴击，给对手造成");
415. itoa(damage, tmpStr, 10);
416. strcat(sendBuf, tmpStr);
417. memset(tmpStr, 0, **sizeof**(tmpStr));
418. strcat(sendBuf, "点伤害，");
419. }
420. **else**
421. {
422. strcpy(sendBuf, name);
423. strcat(sendBuf, "使用");
424. strcat(sendBuf, skill[skillIndex]->getName());
425. strcat(sendBuf, "，给对手造成");
426. itoa(damage, tmpStr, 10);
427. strcat(sendBuf, tmpStr);
428. memset(tmpStr, 0, **sizeof**(tmpStr));
429. strcat(sendBuf, "点伤害，");
430. }
431. **if** (effect == 1)
432. {
433. strcat(sendBuf, "\n");
434. }
435. **else** **if** (effect == 2)
436. {
437. strcat(sendBuf, "效果拔群\n");
438. }
439. **else** **if** (effect == 0.5)
440. {
441. strcat(sendBuf, "效果一般\n");
442. }
443. }
444. //cout << name << "使用" << skill[skillIndex]->getName() << "，给对手造成" << damage << "点伤害，";
446. target.changeHP(-damage);
447. showEffect(effect);

450. **if** (target.getHP() > 0)
451. {
452. //cout << "对手HP剩余" << target.getHP() << endl;
453. strcat(sendBuf, "对手HP剩余");
454. itoa(target.getHP(), tmpStr, 10);
455. strcat(sendBuf, tmpStr);
456. memset(tmpStr, 0, **sizeof**(tmpStr));
457. strcat(sendBuf, "\n");
458. }
459. **else**
460. {
461. //cout << "对手已倒下" << endl;
462. strcat(sendBuf, "对手已倒下\n");
463. target.changeHP(-target.getHP());//HP置零
464. }
465. }
466. //cout << endl;
467. strcat(sendBuf, "\n");
468. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
469. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
470. }
471. **void** Pokemon::learnSkill(SOCKET\* clntSock)
472. {
473. cout << "\*\*技能学习由子类定义，基类无法学习技能\*\*" << endl;
474. }
475. **void** Pokemon::levelUp(SOCKET\* clntSock, **int** getExp)
476. {
477. **int** sendLen;
478. **int** receiveLen;
479. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
480. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
481. **char** tmpStr[100];
483. **int** nextLevelExp = 0;
484. **int** thisLevelExp = expNeeded[0];
485. **if** (LV < MAX\_LEVEL)//如果未满级
486. {
487. strcpy(sendBuf, "成功升级！");
488. **for** (**int** i = 0; i < LV; i++)//计算升级所需经验值
489. {
490. nextLevelExp += expNeeded[i];
491. thisLevelExp = expNeeded[i+1];
492. }
493. EXP += getExp;
494. **if** (EXP > MAX\_EXP)//如果经验溢出则置为最大值
495. {
496. EXP = MAX\_EXP;
497. }
498. **if** (EXP >= nextLevelExp)//如果可以升级
499. {
500. ATK += 2;
501. HP += 5;
502. DEF += 1;
503. ///cout << this->name << "提升了2点攻击，5点HP，1点防御。";
504. strcat(sendBuf, **this**->name);
505. strcat(sendBuf, "提升了2点攻击，5点HP，1点防御。");
506. **switch** (type)
507. {
508. **case** POWER:
509. ATK += 1;
510. //cout << "额外提升1点攻击" << endl;
511. strcat(sendBuf, "额外提升1点攻击\n");
512. **break**;
513. **case** BEARABLE:
514. HP += 4;
515. //cout << "额外提升4点生命" << endl;
516. strcat(sendBuf, "额外提升4点生命\n");
517. **break**;
518. **case** DEDENSIVE:
519. DEF += 1;
520. //cout << "额外提升1点防御" << endl;
521. strcat(sendBuf, "额外提升1点防御\n");
522. **break**;
523. **default**:
524. //cout << "无额外提升" << endl;
525. strcat(sendBuf, "无额外提升\n");
526. **break**;
527. }
528. LV++;
529. fullHP = HP;
530. //cout << "等级提升至: LV " << LV << endl;
531. strcat(sendBuf, "等级提升至: LV ");
532. itoa(LV, tmpStr, 10);
533. strcat(sendBuf, tmpStr);
534. memset(tmpStr, 0, **sizeof**(tmpStr));
535. strcat(sendBuf, "\n");

538. **if** (LV == MAX\_LEVEL)
539. {
540. //cout << this->name << "已满级" << endl;
541. strcat(sendBuf, "已满级\n");
542. }
543. **else**
544. {
545. //cout << "距离下次升级还需" << nextLevelExp + thisLevelExp - EXP << "点经验值" << endl;
546. strcat(sendBuf, "距离下次升级还需");
547. itoa(nextLevelExp + thisLevelExp - EXP, tmpStr, 10);
548. strcat(sendBuf, tmpStr);
549. memset(tmpStr, 0, **sizeof**(tmpStr));
550. strcat(sendBuf, "点经验值\n");
551. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
552. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
553. Sleep(10);
554. }
555. **if** (LV % 3 == 0)
556. {
557. learnSkill(clntSock);
558. }
559. }
560. }
561. **else**
562. {
563. //cout << "已达等级上限，无法继续升级" << endl;
564. strcpy(sendBuf, "已达等级上限，无法继续升级\n\n");
565. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
566. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
567. Sleep(10);
568. }
569. //cout << endl;
570. }
571. **void** Pokemon::recover()
572. {
573. HP = fullHP;
574. }
575. **void** Pokemon::setName(**char**\* newName)
576. {
577. strcpy(name, newName);
578. }
580. **class** Bulbasaur :**public** Pokemon//妙蛙种子
581. {
582. **protected**:
583. **public**:
584. Bulbasaur();//构造函数
585. Bulbasaur(**const** **char**\* newName, **const** **int**\* data);//构造函数
586. Bulbasaur(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
587. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
588. };
589. Bulbasaur::Bulbasaur()
590. {
591. globalIndex = BLUBASAUR;
592. elem = GRASS;
593. skillNum = 2;
594. skill[1] = **new** Skill(GRASS, "飞叶快刀", 0);
595. }
596. Bulbasaur::Bulbasaur(**const** **char**\* newName, **const** **int**\* data)
597. :Pokemon(newName, data)
598. {
599. globalIndex = BLUBASAUR;
600. elem = GRASS;
601. skillNum = 2;
602. skill[1] = **new** Skill(GRASS, "飞叶快刀", 0);
603. }
604. Bulbasaur::Bulbasaur(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
605. :Pokemon(newName, data, newSkillNum, skills)
606. {
607. globalIndex = BLUBASAUR;
608. elem = GRASS;
609. }
610. **void** Bulbasaur::learnSkill(SOCKET\* clntSock)
611. {
612. **int** sendLen;
613. **int** receiveLen;
614. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
615. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
616. **char** tmpStr[100];
618. **int** skillIndex = 0;
619. **int** skillChange = 0;
620. **char** forget = 'N';
621. **int** flag = 1;
622. skillIndex = (LV / 3) % 5;
623. //cout << "可以学习新的技能了！" << endl;
624. strcpy(sendBuf, "可以学习新的技能了！\n");
625. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
626. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
627. Sleep(10);
628. **if** (skillNum < MAX\_SKILL)
629. {
630. skill[skillNum] = **new** Skill(grassSkillDataBase[skillIndex].elem, grassSkillDataBase[skillIndex].name, grassSkillDataBase[skillIndex].ex);
631. //cout << "学会了" << grassSkillDataBase[skillIndex].name << endl;
632. strcpy(sendBuf, "学会了");
633. strcat(sendBuf, grassSkillDataBase[skillIndex].name);
634. strcat(sendBuf, "\n");
635. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
636. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
637. Sleep(10);
638. skillNum++;
639. }
640. **else**//已会4个技能，需选择是否遗忘某个技能
641. {
642. strcpy(sendBuf, "#forget#");
643. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
644. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
645. Sleep(10);
646. //cout << name << "已经学会4个技能了，要学习新技能" << grassSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
647. strcpy(sendBuf, name);
648. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
649. strcat(sendBuf, grassSkillDataBase[skillIndex].name);
650. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
651. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
652. Sleep(10);
653. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
654. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
655. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
657. //cin >> forget;
658. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
659. **if** (strcmp("Y", receiveBuf) == 0|| strcmp("y", receiveBuf) == 0)
660. {
661. skillChange = deleteSkill(clntSock);
662. strcpy(sendBuf, "#learnEnd#");
663. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
664. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
665. Sleep(10);
666. **if** (skillChange == -1)
667. {
668. //cout << "放弃学习" << endl;
669. strcpy(sendBuf, "放弃学习\n");
670. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
671. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
672. Sleep(10);
673. }
674. **else**
675. {
676. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << grassSkillDataBase[skillIndex].name << endl;
677. strcpy(sendBuf, "成功忘记");
678. strcat(sendBuf, skill[skillChange]->getName());
679. strcat(sendBuf, "并学会了");
680. strcat(sendBuf, grassSkillDataBase[skillIndex].name);
681. strcat(sendBuf, "\n");
682. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
683. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
684. Sleep(10);
685. skill[skillChange] = **new** Skill(grassSkillDataBase[skillIndex].elem, grassSkillDataBase[skillIndex].name, grassSkillDataBase[skillIndex].ex);
686. }
687. }
688. **else**
689. {
690. strcpy(sendBuf, "#allEnd#");
691. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
692. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
693. Sleep(10);
694. //cout << "放弃学习新技能" << endl;
695. strcpy(sendBuf, "放弃学习新技能\n");
696. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
697. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
698. Sleep(10);
699. }
700. }
701. }
703. **class** Charizard :**public** Pokemon//喷火龙
704. {
705. **protected**:
706. **public**:
707. Charizard();//构造函数
708. Charizard(**const** **char**\* newName, **const** **int**\* data);//构造函数
709. Charizard(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
710. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
711. };
712. Charizard::Charizard()
713. {
714. globalIndex = CHARIZARD;
715. elem = FIRE;
716. skillNum = 2;
717. skill[1] = **new** Skill(FIRE, "喷火", 0);
718. }
719. Charizard::Charizard(**const** **char**\* newName, **const** **int**\* data)
720. :Pokemon(newName, data)
721. {
722. globalIndex = CHARIZARD;
723. elem = FIRE;
724. skillNum = 2;
725. skill[1] = **new** Skill(FIRE, "喷火", 0);
726. }
727. Charizard::Charizard(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
728. :Pokemon(newName, data, newSkillNum, skills)
729. {
730. globalIndex = CHARIZARD;
731. elem = FIRE;
732. }
733. **void** Charizard::learnSkill(SOCKET\* clntSock)
734. {
735. **int** sendLen;
736. **int** receiveLen;
737. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
738. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
739. **char** tmpStr[100];
741. **int** skillIndex = 0;
742. **int** skillChange = 0;
743. **char** forget = 'N';
744. **int** flag = 1;
745. skillIndex = (LV / 3) % 5;
746. //cout << "可以学习新的技能了！" << endl;
747. strcpy(sendBuf, "可以学习新的技能了！\n");
748. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
749. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
750. Sleep(10);
751. **if** (skillNum < MAX\_SKILL)
752. {
753. skill[skillNum] = **new** Skill(fireSkillDataBase[skillIndex].elem, fireSkillDataBase[skillIndex].name, fireSkillDataBase[skillIndex].ex);
754. //cout << "学会了" << fireSkillDataBase[skillIndex].name << endl;
755. strcpy(sendBuf, "学会了");
756. strcat(sendBuf, fireSkillDataBase[skillIndex].name);
757. strcat(sendBuf, "\n");
758. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
759. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
760. Sleep(10);
761. skillNum++;
762. }
763. **else**//已会4个技能，需选择是否遗忘某个技能
764. {
765. strcpy(sendBuf, "#forget#");
766. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
767. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
768. Sleep(10);
769. //cout << name << "已经学会4个技能了，要学习新技能" << fireSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
770. strcpy(sendBuf, name);
771. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
772. strcat(sendBuf, fireSkillDataBase[skillIndex].name);
773. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
774. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
775. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
776. Sleep(10);
777. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
778. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
780. //cin >> forget;
781. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
782. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
783. {
784. skillChange = deleteSkill(clntSock);
785. strcpy(sendBuf, "#learnEnd#");
786. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
787. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
788. Sleep(10);
789. **if** (skillChange == -1)
790. {
791. //cout << "放弃学习" << endl;
792. strcpy(sendBuf, "放弃学习\n");
793. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
794. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
795. Sleep(10);
796. }
797. **else**
798. {
799. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << fireSkillDataBase[skillIndex].name << endl;
800. strcpy(sendBuf, "成功忘记");
801. strcat(sendBuf, skill[skillChange]->getName());
802. strcat(sendBuf, "并学会了");
803. strcat(sendBuf, fireSkillDataBase[skillIndex].name);
804. strcat(sendBuf, "\n");
805. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
806. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
807. Sleep(10);
808. skill[skillChange] = **new** Skill(fireSkillDataBase[skillIndex].elem, fireSkillDataBase[skillIndex].name, fireSkillDataBase[skillIndex].ex);
809. }
810. }
811. **else**
812. {
813. strcpy(sendBuf, "#allEnd#");
814. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
815. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
816. Sleep(10);
817. //cout << "放弃学习新技能" << endl;
818. strcpy(sendBuf, "放弃学习新技能\n");
819. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
820. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
821. Sleep(10);
822. }
823. }
824. }
826. **class** Squirtle :**public** Pokemon//杰尼龟
827. {
828. **protected**:
829. **public**:
830. Squirtle();//构造函数
831. Squirtle(**const** **char**\* newName, **const** **int**\* data);//构造函数
832. Squirtle(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
833. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
834. };
835. Squirtle::Squirtle()
836. {
837. globalIndex = SQUIRTLE;
838. elem = WATER;
839. skillNum = 2;
840. skill[1] = **new** Skill(WATER, "喷水", 0);
841. }
842. Squirtle::Squirtle(**const** **char**\* newName, **const** **int**\* data)
843. :Pokemon(newName, data)
844. {
845. globalIndex = SQUIRTLE;
846. elem = WATER;
847. skillNum = 2;
848. skill[1] = **new** Skill(WATER, "喷水", 0);
849. }
850. Squirtle::Squirtle(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
851. :Pokemon(newName, data, newSkillNum, skills)
852. {
853. globalIndex = SQUIRTLE;
854. elem = WATER;
855. }
857. **void** Squirtle::learnSkill(SOCKET\* clntSock)
858. {
859. **int** sendLen;
860. **int** receiveLen;
861. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
862. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
863. **char** tmpStr[100];
865. **int** skillIndex = 0;
866. **int** skillChange = 0;
867. **char** forget = 'N';
868. **int** flag = 1;
869. skillIndex = (LV / 3) % 5;
870. //cout << "可以学习新的技能了！" << endl;
871. strcpy(sendBuf, "可以学习新的技能了！\n");
872. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
873. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
874. Sleep(10);
875. **if** (skillNum < MAX\_SKILL)
876. {
877. skill[skillNum] = **new** Skill(waterSkillDataBase[skillIndex].elem, waterSkillDataBase[skillIndex].name, waterSkillDataBase[skillIndex].ex);
878. //cout << "学会了" << waterSkillDataBase[skillIndex].name << endl;
879. strcpy(sendBuf, "学会了");
880. strcat(sendBuf, waterSkillDataBase[skillIndex].name);
881. strcat(sendBuf, "\n");
882. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
883. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
884. Sleep(10);
885. skillNum++;
886. }
887. **else**//已会4个技能，需选择是否遗忘某个技能
888. {
889. strcpy(sendBuf, "#forget#");
890. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
891. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
892. Sleep(10);
893. //cout << name << "已经学会4个技能了，要学习新技能" << waterSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
894. strcpy(sendBuf, name);
895. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
896. strcat(sendBuf, waterSkillDataBase[skillIndex].name);
897. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
898. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
899. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
900. Sleep(10);
901. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
902. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
904. //cin >> forget;
905. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
906. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
907. {
908. skillChange = deleteSkill(clntSock);
909. strcpy(sendBuf, "#learnEnd#");
910. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
911. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
912. Sleep(10);
913. **if** (skillChange == -1)
914. {
915. //cout << "放弃学习" << endl;
916. strcpy(sendBuf, "放弃学习\n");
917. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
918. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
919. Sleep(10);
920. }
921. **else**
922. {
923. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << waterSkillDataBase[skillIndex].name << endl;
924. strcpy(sendBuf, "成功忘记");
925. strcat(sendBuf, skill[skillChange]->getName());
926. strcat(sendBuf, "并学会了");
927. strcat(sendBuf, waterSkillDataBase[skillIndex].name);
928. strcat(sendBuf, "\n");
929. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
930. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
931. Sleep(10);
932. skill[skillChange] = **new** Skill(waterSkillDataBase[skillIndex].elem, waterSkillDataBase[skillIndex].name, waterSkillDataBase[skillIndex].ex);
933. }
934. }
935. **else**
936. {
937. strcpy(sendBuf, "#allEnd#");
938. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
939. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
940. Sleep(10);
941. //cout << "放弃学习新技能" << endl;
942. strcpy(sendBuf, "放弃学习新技能\n");
943. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
944. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
945. Sleep(10);
946. }
947. }
948. }
950. **class** Caterpie :**public** Pokemon//绿毛虫
951. {
952. **protected**:
953. **public**:
954. Caterpie();//构造函数
955. Caterpie(**const** **char**\* newName, **const** **int**\* data);//构造函数
956. Caterpie(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
957. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
958. };
959. Caterpie::Caterpie()
960. {
961. globalIndex = CATERPIE;
962. elem = BUG;
963. skillNum = 2;
964. skill[1] = **new** Skill(BUG, "撕咬", 0);
965. }
966. Caterpie::Caterpie(**const** **char**\* newName, **const** **int**\* data)
967. :Pokemon(newName, data)
968. {
969. globalIndex = CATERPIE;
970. elem = BUG;
971. skillNum = 2;
972. skill[1] = **new** Skill(BUG, "撕咬", 0);
973. }
974. Caterpie::Caterpie(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
975. :Pokemon(newName, data, newSkillNum, skills)
976. {
977. globalIndex = CATERPIE;
978. elem = BUG;
979. }
980. **void** Caterpie::learnSkill(SOCKET\* clntSock)
981. {
982. **int** sendLen;
983. **int** receiveLen;
984. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
985. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
986. **char** tmpStr[100];
988. **int** skillIndex = 0;
989. **int** skillChange = 0;
990. **char** forget = 'N';
991. **int** flag = 1;
992. skillIndex = (LV / 3) % 5;
993. //cout << "可以学习新的技能了！" << endl;
994. strcpy(sendBuf, "可以学习新的技能了！\n");
995. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
996. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
997. Sleep(10);
998. **if** (skillNum < MAX\_SKILL)
999. {
1000. skill[skillNum] = **new** Skill(bugSkillDataBase[skillIndex].elem, bugSkillDataBase[skillIndex].name, bugSkillDataBase[skillIndex].ex);
1001. //cout << "学会了" << bugSkillDataBase[skillIndex].name << endl;
1002. strcpy(sendBuf, "学会了");
1003. strcat(sendBuf, bugSkillDataBase[skillIndex].name);
1004. strcat(sendBuf, "\n");
1005. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1006. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1007. Sleep(10);
1008. skillNum++;
1009. }
1010. **else**//已会4个技能，需选择是否遗忘某个技能
1011. {
1012. strcpy(sendBuf, "#forget#");
1013. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1014. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1015. Sleep(10);
1016. //cout << name << "已经学会4个技能了，要学习新技能" << bugSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
1017. strcpy(sendBuf, name);
1018. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
1019. strcat(sendBuf, bugSkillDataBase[skillIndex].name);
1020. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
1021. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1022. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1023. Sleep(10);
1024. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
1025. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
1027. //cin >> forget;
1028. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
1029. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
1030. {
1031. skillChange = deleteSkill(clntSock);
1032. strcpy(sendBuf, "#learnEnd#");
1033. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1034. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1035. Sleep(10);
1036. **if** (skillChange == -1)
1037. {
1038. //cout << "放弃学习" << endl;
1039. strcpy(sendBuf, "放弃学习\n");
1040. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1041. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1042. Sleep(10);
1043. }
1044. **else**
1045. {
1046. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << bugSkillDataBase[skillIndex].name << endl;
1047. strcpy(sendBuf, "成功忘记");
1048. strcat(sendBuf, skill[skillChange]->getName());
1049. strcat(sendBuf, "并学会了");
1050. strcat(sendBuf, bugSkillDataBase[skillIndex].name);
1051. strcat(sendBuf, "\n");
1052. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1053. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1054. Sleep(10);
1055. skill[skillChange] = **new** Skill(bugSkillDataBase[skillIndex].elem, bugSkillDataBase[skillIndex].name, bugSkillDataBase[skillIndex].ex);
1056. }
1057. }
1058. **else**
1059. {
1060. strcpy(sendBuf, "#allEnd#");
1061. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1062. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1063. Sleep(10);
1064. //cout << "放弃学习新技能" << endl;
1065. strcpy(sendBuf, "放弃学习新技能\n");
1066. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1067. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1068. Sleep(10);
1069. }
1070. }
1071. }
1073. **class** Pidgeotto :**public** Pokemon//比比鸟
1074. {
1075. **protected**:
1076. **public**:
1077. Pidgeotto();//构造函数
1078. Pidgeotto(**const** **char**\* newName, **const** **int**\* data);//构造函数
1079. Pidgeotto(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
1080. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
1081. };
1082. Pidgeotto::Pidgeotto()
1083. {
1084. globalIndex = PIDGEOTTO;
1085. elem = FLY;
1086. skillNum = 2;
1087. skill[1] = **new** Skill(FLY, "起风", 0);
1088. }
1089. Pidgeotto::Pidgeotto(**const** **char**\* newName, **const** **int**\* data)
1090. :Pokemon(newName, data)
1091. {
1092. globalIndex = PIDGEOTTO;
1093. elem = FLY;
1094. skillNum = 2;
1095. skill[1] = **new** Skill(FLY, "起风", 0);
1096. }
1097. Pidgeotto::Pidgeotto(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
1098. :Pokemon(newName, data, newSkillNum, skills)
1099. {
1100. globalIndex = PIDGEOTTO;
1101. elem = FLY;
1102. }
1103. **void** Pidgeotto::learnSkill(SOCKET\* clntSock)
1104. {
1105. **int** sendLen;
1106. **int** receiveLen;
1107. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
1108. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
1109. **char** tmpStr[100];
1111. **int** skillIndex = 0;
1112. **int** skillChange = 0;
1113. **char** forget = 'N';
1114. **int** flag = 1;
1115. skillIndex = (LV / 3) % 5;
1116. //cout << "可以学习新的技能了！" << endl;
1117. strcpy(sendBuf, "可以学习新的技能了！\n");
1118. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1119. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1120. Sleep(10);
1121. **if** (skillNum < MAX\_SKILL)
1122. {
1123. skill[skillNum] = **new** Skill(flySkillDataBase[skillIndex].elem, flySkillDataBase[skillIndex].name, flySkillDataBase[skillIndex].ex);
1124. //cout << "学会了" << flySkillDataBase[skillIndex].name << endl;
1125. strcpy(sendBuf, "学会了");
1126. strcat(sendBuf, flySkillDataBase[skillIndex].name);
1127. strcat(sendBuf, "\n");
1128. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1129. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1130. Sleep(10);
1131. skillNum++;
1132. }
1133. **else**//已会4个技能，需选择是否遗忘某个技能
1134. {
1135. strcpy(sendBuf, "#forget#");
1136. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1137. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1138. Sleep(10);
1139. //cout << name << "已经学会4个技能了，要学习新技能" << flySkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
1140. strcpy(sendBuf, name);
1141. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
1142. strcat(sendBuf, flySkillDataBase[skillIndex].name);
1143. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
1144. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1145. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1146. Sleep(10);
1147. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
1148. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
1150. //cin >> forget;
1151. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
1152. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
1153. {
1154. skillChange = deleteSkill(clntSock);
1155. strcpy(sendBuf, "#learnEnd#");
1156. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1157. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1158. Sleep(10);
1159. **if** (skillChange == -1)
1160. {
1161. //cout << "放弃学习" << endl;
1162. strcpy(sendBuf, "放弃学习\n");
1163. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1164. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1165. Sleep(10);
1166. }
1167. **else**
1168. {
1169. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << flySkillDataBase[skillIndex].name << endl;
1170. strcpy(sendBuf, "成功忘记");
1171. strcat(sendBuf, skill[skillChange]->getName());
1172. strcat(sendBuf, "并学会了");
1173. strcat(sendBuf, flySkillDataBase[skillIndex].name);
1174. strcat(sendBuf, "\n");
1175. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1176. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1177. Sleep(10);
1178. skill[skillChange] = **new** Skill(flySkillDataBase[skillIndex].elem, flySkillDataBase[skillIndex].name, flySkillDataBase[skillIndex].ex);
1179. }
1180. }
1181. **else**
1182. {
1183. strcpy(sendBuf, "#allEnd#");
1184. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1185. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1186. Sleep(10);
1187. //cout << "放弃学习新技能" << endl;
1188. strcpy(sendBuf, "放弃学习新技能\n");
1189. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1190. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1191. Sleep(10);
1192. }
1193. }
1194. }
1196. **class** Diglett :**public** Pokemon//地鼠
1197. {
1198. **protected**:
1199. **public**:
1200. Diglett();//构造函数
1201. Diglett(**const** **char**\* newName, **const** **int**\* data);//构造函数
1202. Diglett(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
1203. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
1204. };
1205. Diglett::Diglett()
1206. {
1207. globalIndex = DIGLETT;
1208. elem = GROUND;
1209. skillNum = 2;
1210. skill[1] = **new** Skill(GROUND, "击地", 0);
1211. }
1212. Diglett::Diglett(**const** **char**\* newName, **const** **int**\* data)
1213. :Pokemon(newName, data)
1214. {
1215. globalIndex = DIGLETT;
1216. elem = GROUND;
1217. skillNum = 2;
1218. skill[1] = **new** Skill(GROUND, "击地", 0);
1219. }
1220. Diglett::Diglett(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
1221. :Pokemon(newName, data, newSkillNum, skills)
1222. {
1223. globalIndex = DIGLETT;
1224. elem = GROUND;
1225. }
1226. **void** Diglett::learnSkill(SOCKET\* clntSock)
1227. {
1228. **int** sendLen;
1229. **int** receiveLen;
1230. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
1231. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
1232. **char** tmpStr[100];
1234. **int** skillIndex = 0;
1235. **int** skillChange = 0;
1236. **char** forget = 'N';
1237. **int** flag = 1;
1238. skillIndex = (LV / 3) % 5;
1239. //cout << "可以学习新的技能了！" << endl;
1240. strcpy(sendBuf, "可以学习新的技能了！\n");
1241. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1242. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1243. Sleep(10);
1244. **if** (skillNum < MAX\_SKILL)
1245. {
1246. skill[skillNum] = **new** Skill(groundSkillDataBase[skillIndex].elem, groundSkillDataBase[skillIndex].name, groundSkillDataBase[skillIndex].ex);
1247. //cout << "学会了" << groundSkillDataBase[skillIndex].name << endl;
1248. strcpy(sendBuf, "学会了");
1249. strcat(sendBuf, groundSkillDataBase[skillIndex].name);
1250. strcat(sendBuf, "\n");
1251. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1252. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1253. Sleep(10);
1254. skillNum++;
1255. }
1256. **else**//已会4个技能，需选择是否遗忘某个技能
1257. {
1258. strcpy(sendBuf, "#forget#");
1259. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1260. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1261. Sleep(10);
1262. //cout << name << "已经学会4个技能了，要学习新技能" << groundSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
1263. strcpy(sendBuf, name);
1264. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
1265. strcat(sendBuf, groundSkillDataBase[skillIndex].name);
1266. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
1267. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1268. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1269. Sleep(10);
1270. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
1271. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
1273. //cin >> forget;
1274. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
1275. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
1276. {
1277. skillChange = deleteSkill(clntSock);
1278. strcpy(sendBuf, "#learnEnd#");
1279. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1280. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1281. Sleep(10);
1282. **if** (skillChange == -1)
1283. {
1284. //cout << "放弃学习" << endl;
1285. strcpy(sendBuf, "放弃学习\n");
1286. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1287. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1288. Sleep(10);
1289. }
1290. **else**
1291. {
1292. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << groundSkillDataBase[skillIndex].name << endl;
1293. strcpy(sendBuf, "成功忘记");
1294. strcat(sendBuf, skill[skillChange]->getName());
1295. strcat(sendBuf, "并学会了");
1296. strcat(sendBuf, groundSkillDataBase[skillIndex].name);
1297. strcat(sendBuf, "\n");
1298. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1299. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1300. Sleep(10);
1301. skill[skillChange] = **new** Skill(groundSkillDataBase[skillIndex].elem, groundSkillDataBase[skillIndex].name, groundSkillDataBase[skillIndex].ex);
1302. }
1303. }
1304. **else**
1305. {
1306. strcpy(sendBuf, "#allEnd#");
1307. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1308. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1309. Sleep(10);
1310. //cout << "放弃学习新技能" << endl;
1311. strcpy(sendBuf, "放弃学习新技能\n");
1312. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1313. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1314. Sleep(10);
1315. }
1316. }
1317. }
1319. **class** Pikachu :**public** Pokemon//皮卡丘
1320. {
1321. **protected**:
1322. **public**:
1323. Pikachu();//构造函数
1324. Pikachu(**const** **char**\* newName, **const** **int**\* data);//构造函数
1325. Pikachu(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
1326. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
1327. };
1328. Pikachu::Pikachu()
1329. {
1330. globalIndex = PIKACHU;
1331. elem = ELECTRIC;
1332. skillNum = 2;
1333. skill[1] = **new** Skill(ELECTRIC, "电击", 0);
1334. }
1335. Pikachu::Pikachu(**const** **char**\* newName, **const** **int**\* data)
1336. :Pokemon(newName, data)
1337. {
1338. globalIndex = PIKACHU;
1339. elem = ELECTRIC;
1340. skillNum = 2;
1341. skill[1] = **new** Skill(ELECTRIC, "电击", 0);
1342. }
1343. Pikachu::Pikachu(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
1344. :Pokemon(newName, data, newSkillNum, skills)
1345. {
1346. globalIndex = PIKACHU;
1347. elem = ELECTRIC;
1348. }
1349. **void** Pikachu::learnSkill(SOCKET\* clntSock)
1350. {
1351. **int** sendLen;
1352. **int** receiveLen;
1353. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
1354. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
1355. **char** tmpStr[100];
1357. **int** skillIndex = 0;
1358. **int** skillChange = 0;
1359. **char** forget = 'N';
1360. **int** flag = 1;
1361. skillIndex = (LV / 3) % 5;
1362. //cout << "可以学习新的技能了！" << endl;
1363. strcpy(sendBuf, "可以学习新的技能了！\n");
1364. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1365. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1366. Sleep(10);
1367. **if** (skillNum < MAX\_SKILL)
1368. {
1369. skill[skillNum] = **new** Skill(electricSkillDataBase[skillIndex].elem, electricSkillDataBase[skillIndex].name, electricSkillDataBase[skillIndex].ex);
1370. //cout << "学会了" << electricSkillDataBase[skillIndex].name << endl;
1371. strcpy(sendBuf, "学会了");
1372. strcat(sendBuf, electricSkillDataBase[skillIndex].name);
1373. strcat(sendBuf, "\n");
1374. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1375. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1376. Sleep(10);
1377. skillNum++;
1378. }
1379. **else**//已会4个技能，需选择是否遗忘某个技能
1380. {
1381. strcpy(sendBuf, "#forget#");
1382. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1383. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1384. Sleep(10);
1385. //cout << name << "已经学会4个技能了，要学习新技能" << electricSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
1386. strcpy(sendBuf, name);
1387. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
1388. strcat(sendBuf, electricSkillDataBase[skillIndex].name);
1389. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
1390. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1391. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1392. Sleep(10);
1393. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
1394. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
1396. //cin >> forget;
1397. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
1398. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
1399. {
1400. skillChange = deleteSkill(clntSock);
1401. strcpy(sendBuf, "#learnEnd#");
1402. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1403. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1404. Sleep(10);
1405. **if** (skillChange == -1)
1406. {
1407. //cout << "放弃学习" << endl;
1408. strcpy(sendBuf, "放弃学习\n");
1409. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1410. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1411. Sleep(10);
1412. }
1413. **else**
1414. {
1415. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << electricSkillDataBase[skillIndex].name << endl;
1416. strcpy(sendBuf, "成功忘记");
1417. strcat(sendBuf, skill[skillChange]->getName());
1418. strcat(sendBuf, "并学会了");
1419. strcat(sendBuf, electricSkillDataBase[skillIndex].name);
1420. strcat(sendBuf, "\n");
1421. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1422. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1423. Sleep(10);
1424. skill[skillChange] = **new** Skill(electricSkillDataBase[skillIndex].elem, electricSkillDataBase[skillIndex].name, electricSkillDataBase[skillIndex].ex);
1425. }
1426. }
1427. **else**
1428. {
1429. strcpy(sendBuf, "#allEnd#");
1430. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1431. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1432. Sleep(10);
1433. //cout << "放弃学习新技能" << endl;
1434. strcpy(sendBuf, "放弃学习新技能\n");
1435. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1436. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1437. Sleep(10);
1438. }
1439. }
1440. }
1442. **class** Snorunt :**public** Pokemon//雪童子
1443. {
1444. **protected**:
1445. **public**:
1446. Snorunt();//构造函数
1447. Snorunt(**const** **char**\* newName, **const** **int**\* data);//构造函数
1448. Snorunt(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
1449. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
1450. };
1451. Snorunt::Snorunt()
1452. {
1453. globalIndex = SNORUNT;
1454. elem = ICE;
1455. skillNum = 2;
1456. skill[1] = **new** Skill(ICE, "吐冰", 0);
1457. }
1458. Snorunt::Snorunt(**const** **char**\* newName, **const** **int**\* data)
1459. :Pokemon(newName, data)
1460. {
1461. globalIndex = SNORUNT;
1462. elem = ICE;
1463. skillNum = 2;
1464. skill[1] = **new** Skill(ICE, "吐冰", 0);
1465. }
1466. Snorunt::Snorunt(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
1467. :Pokemon(newName, data, newSkillNum, skills)
1468. {
1469. globalIndex = SNORUNT;
1470. elem = ICE;
1471. }
1472. **void** Snorunt::learnSkill(SOCKET\* clntSock)
1473. {
1474. **int** sendLen;
1475. **int** receiveLen;
1476. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
1477. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
1478. **char** tmpStr[100];
1480. **int** skillIndex = 0;
1481. **int** skillChange = 0;
1482. **char** forget = 'N';
1483. **int** flag = 1;
1484. skillIndex = (LV / 3) % 5;
1485. //cout << "可以学习新的技能了！" << endl;
1486. strcpy(sendBuf, "可以学习新的技能了！\n");
1487. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1488. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1489. Sleep(10);
1490. **if** (skillNum < MAX\_SKILL)
1491. {
1492. skill[skillNum] = **new** Skill(iceSkillDataBase[skillIndex].elem, iceSkillDataBase[skillIndex].name, iceSkillDataBase[skillIndex].ex);
1493. //cout << "学会了" << iceSkillDataBase[skillIndex].name << endl;
1494. strcpy(sendBuf, "学会了");
1495. strcat(sendBuf, iceSkillDataBase[skillIndex].name);
1496. strcat(sendBuf, "\n");
1497. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1498. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1499. Sleep(10);
1500. skillNum++;
1501. }
1502. **else**//已会4个技能，需选择是否遗忘某个技能
1503. {
1504. strcpy(sendBuf, "#forget#");
1505. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1506. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1507. Sleep(10);
1508. //cout << name << "已经学会4个技能了，要学习新技能" << iceSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
1509. strcpy(sendBuf, name);
1510. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
1511. strcat(sendBuf, iceSkillDataBase[skillIndex].name);
1512. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
1513. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1514. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1515. Sleep(10);
1516. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
1517. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
1519. //cin >> forget;
1520. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
1521. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
1522. {
1523. skillChange = deleteSkill(clntSock);
1524. strcpy(sendBuf, "#learnEnd#");
1525. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1526. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1527. Sleep(10);
1528. **if** (skillChange == -1)
1529. {
1530. //cout << "放弃学习" << endl;
1531. strcpy(sendBuf, "放弃学习\n");
1532. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1533. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1534. Sleep(10);
1535. }
1536. **else**
1537. {
1538. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << iceSkillDataBase[skillIndex].name << endl;
1539. strcpy(sendBuf, "成功忘记");
1540. strcat(sendBuf, skill[skillChange]->getName());
1541. strcat(sendBuf, "并学会了");
1542. strcat(sendBuf, iceSkillDataBase[skillIndex].name);
1543. strcat(sendBuf, "\n");
1544. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1545. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1546. Sleep(10);
1547. skill[skillChange] = **new** Skill(iceSkillDataBase[skillIndex].elem, iceSkillDataBase[skillIndex].name, iceSkillDataBase[skillIndex].ex);
1548. }
1549. }
1550. **else**
1551. {
1552. strcpy(sendBuf, "#allEnd#");
1553. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1554. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1555. Sleep(10);
1556. //cout << "放弃学习新技能" << endl;
1557. strcpy(sendBuf, "放弃学习新技能\n");
1558. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1559. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1560. Sleep(10);
1561. }
1562. }
1563. }
1565. **class** Dragonite :**public** Pokemon//快龙
1566. {
1567. **protected**:
1568. **public**:
1569. Dragonite();//构造函数
1570. Dragonite(**const** **char**\* newName, **const** **int**\* data);//构造函数
1571. Dragonite(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills);//构造函数
1572. **virtual** **void** learnSkill(SOCKET\* clntSock);//学习技能
1573. };
1574. Dragonite::Dragonite()
1575. {
1576. globalIndex = DRAGONITE;
1577. elem = DRAGON;
1578. skillNum = 2;
1579. skill[1] = **new** Skill(DRAGON, "龙爪", 0);
1580. }
1581. Dragonite::Dragonite(**const** **char**\* newName, **const** **int**\* data)
1582. :Pokemon(newName, data)
1583. {
1584. globalIndex = DRAGONITE;
1585. elem = DRAGON;
1586. skillNum = 2;
1587. skill[1] = **new** Skill(DRAGON, "龙爪", 0);
1588. }
1589. Dragonite::Dragonite(**const** **char**\* newName, **const** **int**\* data, **int** newSkillNum, **const** SKILL\_DATA\_ITEM\* skills)
1590. :Pokemon(newName, data, newSkillNum, skills)
1591. {
1592. globalIndex = DRAGONITE;
1593. elem = DRAGON;
1594. }
1595. **void** Dragonite::learnSkill(SOCKET\* clntSock)
1596. {
1597. **int** sendLen;
1598. **int** receiveLen;
1599. **char** sendBuf[MAX\_BUF\_SIZE] = { 0 };
1600. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };
1601. **char** tmpStr[100];
1603. **int** skillIndex = 0;
1604. **int** skillChange = 0;
1605. **char** forget = 'N';
1606. **int** flag = 1;
1607. skillIndex = (LV / 3) % 5;
1608. //cout << "可以学习新的技能了！" << endl;
1609. strcpy(sendBuf, "可以学习新的技能了！\n");
1610. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1611. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1612. Sleep(10);
1613. **if** (skillNum < MAX\_SKILL)
1614. {
1615. skill[skillNum] = **new** Skill(dragonSkillDataBase[skillIndex].elem, dragonSkillDataBase[skillIndex].name, dragonSkillDataBase[skillIndex].ex);
1616. //cout << "学会了" << dragonSkillDataBase[skillIndex].name << endl;
1617. strcpy(sendBuf, "学会了");
1618. strcat(sendBuf, dragonSkillDataBase[skillIndex].name);
1619. strcat(sendBuf, "\n");
1620. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1621. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1622. Sleep(10);
1623. skillNum++;
1624. }
1625. **else**//已会4个技能，需选择是否遗忘某个技能
1626. {
1627. strcpy(sendBuf, "#forget#");
1628. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1629. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1630. Sleep(10);
1631. //cout << name << "已经学会4个技能了，要学习新技能" << dragonSkillDataBase[skillIndex].name << "需要忘记一个旧技能，是否忘记？(Y/N) ";
1632. strcpy(sendBuf, name);
1633. strcat(sendBuf, "已经学会4个技能了，要学习新技能");
1634. strcat(sendBuf, dragonSkillDataBase[skillIndex].name);
1635. strcat(sendBuf, "需要忘记一个旧技能，是否忘记？(Y / N)");
1636. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1637. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1638. Sleep(10);
1639. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
1640. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
1642. //cin >> forget;
1643. //cin.ignore((numeric\_limits<streamsize>::max)(), '\n');//清除输入缓冲区
1644. **if** (strcmp("Y", receiveBuf) == 0 || strcmp("y", receiveBuf) == 0)
1645. {
1646. skillChange = deleteSkill(clntSock);
1647. strcpy(sendBuf, "#learnEnd#");
1648. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1649. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1650. Sleep(10);
1651. **if** (skillChange == -1)
1652. {
1653. //cout << "放弃学习" << endl;
1654. strcpy(sendBuf, "放弃学习\n");
1655. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1656. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1657. Sleep(10);
1658. }
1659. **else**
1660. {
1661. //cout << "成功忘记" << skill[skillChange]->getName() << "并学会了" << dragonSkillDataBase[skillIndex].name << endl;
1662. strcpy(sendBuf, "成功忘记");
1663. strcat(sendBuf, skill[skillChange]->getName());
1664. strcat(sendBuf, "并学会了");
1665. strcat(sendBuf, dragonSkillDataBase[skillIndex].name);
1666. strcat(sendBuf, "\n");
1667. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1668. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1669. Sleep(10);
1670. skill[skillChange] = **new** Skill(dragonSkillDataBase[skillIndex].elem, dragonSkillDataBase[skillIndex].name, dragonSkillDataBase[skillIndex].ex);
1671. }
1672. }
1673. **else**
1674. {
1675. strcpy(sendBuf, "#allEnd#");
1676. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1677. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1678. Sleep(10);
1679. //cout << "放弃学习新技能" << endl;
1680. strcpy(sendBuf, "放弃学习新技能\n");
1681. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
1682. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
1683. Sleep(10);
1684. }
1685. }
1686. }
1688. #endif

### 文件5：main.cpp

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. /\* @课程名：        C++课程设计 题目二 用户注册与平台登录   \*/
3. /\* @作者；         谢睿（2018211208）                      \*/
4. /\* @教师：         王洪波                                 \*/
5. /\* @最后修改时间： 2020/09/01 19：22                        \*/
6. /\* @文件名：        main.cpp                                \*/
7. /\* @文件说明：       主功能代码，用于测试以及功能实现        \*/
8. /\*----------------------------------------------------------\*/
9. /\* @备注：         本程序遵循任天堂公司习惯用语，使用“宝可\*/
10. /\*                  梦(Pokemon)”代指宠物小精灵              \*/
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. #include<iostream>
14. #include<fstream>
15. #include<string>
16. #include<winsock2.h>
17. #include<string.h>
18. #include"header.h"
19. #include"PokemonList.h"
20. #include"server.h"
21. **using** **namespace** std;
23. #pragma comment (lib, "ws2\_32.lib")  //加载 ws2\_32.dll

26. //全局变量区
27. //int pokemonNumOwned = 0;//玩家拥有的宝可梦数
28. **int** expNeeded[14] = { 20,40,60,90,120,160,200,250,300,350,400,500,700,1000 };//升级所需经验值
29. **double** effectiveness[10][10] = { {1,   1,   1,   1,   1,   1,   1,   1,   1,   1  },
30. {1,   0.5, 2,   0.5, 2,   1,   1,   1,   2,   0.5},
31. {1,   0.5, 0.5, 1,   0.5, 0.5, 2,   1,   1,   0.5},
32. {1,   2,   0.5, 0.5, 1,   1,   2,   1,   1,   0.5},
33. {1,   0.5, 2,   1,   1,   0.5, 1,   1,   1,   1  },
34. {1,   1,   2,   1,   2,   1,   1,   0.5, 1,   1  },
35. {1,   2,   0.5, 1,   0.5, 0,   1,   2,   1,   1  },
36. {1,   1,   0.5, 2,   1,   2,   0,   0.5, 1,   0.5},
37. {1,   0.5, 2,   0.5, 1,   2,   2,   1,   0.5, 2  },
38. {1,   1,   1,   1,   1,   1,   1,   1,   1,   2  } };//元素相克表
39. **int** userNum = 0;//所有注册用户数
40. **int** activeUserNum = 0;//当前在线用户
41. CLIENT\_DATA\* allClients = NULL;//所有在线用户的链表
42. **char**\* virtualPokemonList;//用户可选择对战的虚拟精灵列表
43. **char**\* virtualPokemonName[10];//用户可选择对战的虚拟精灵名字
44. **int**\* virtualPokemonData[10];//用户可选择对战的虚拟精灵属性

47. //服务线程
48. **DWORD** WINAPI ServerThread(**LPVOID** lpParameter)
49. {
50. //传入参数处理
51. CLIENT\_DATA\* pmd = (CLIENT\_DATA\*)lpParameter;
52. SOCKET\* clntSock = pmd->clientSocket;
53. User\* client = pmd->clientUser;
54. **int** state = pmd->state;
56. **int** menuFlag = 1;//退出菜单标志位
57. **int** tmpData[9] = { FIRE,POWER,0,1,2,10,1,2,2 };//初始化数据
58. **char** tmpStr[100] = { 0 };//用于拷贝以及比较的临时数组
59. **int** sendLen;//成功发送长度
60. **int** receiveLen;//成功接收长度
61. **char** sendBuf[MAX\_BUF\_SIZE] = "This is a message from server\n";//发送缓冲区
62. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };//接收缓冲区
63. **int** round = 0;//出招轮数
64. **double** winRate = 0;//胜率
66. srand(time(0));
68. //主菜单以及登陆工作
69. **char** tmpBuf[32] = { 0 };
70. ifstream userFile("./DATA/userData.txt");
71. userFile >> tmpBuf;
72. userFile >> userNum;
73. userFile.close();
75. **char** uName[32] = { 0 };
76. **char** uPass[32] = { 0 };
77. **int** result = 0;
78. cout << "客户机连接成功" << endl << endl;
80. **while** (menuFlag)
81. {
82. strcpy(sendBuf, "\n您还未登陆，请选择：\n"
83. "1.用户登录\n"
84. "2.用户注册\n"
85. "3.查看注册用户信息\n"
86. "4.查看在线用户\n"
87. "5.断开与服务器的连接\n");
88. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
89. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
90. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
91. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
92. **if** (receiveLen < 0)
93. {
94. cout << "接收失败,客户机已断开" << endl;
95. menuFlag = 0;
96. **break**;
97. }
98. **else**
99. {
100. **if** (strcmp("1", receiveBuf) == 0)//登陆
101. {
102. cout << "用户登陆" << endl;
103. **int** tryCount = 2;
104. **while** (tryCount >= 0)
105. {
106. strcpy(sendBuf, "请输入用户名(32字符以下)：\n");
107. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
108. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
109. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
110. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
111. strcpy(uName, receiveBuf);
112. strcpy(sendBuf, "请输入密码(32字符以下)：\n");
113. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
114. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
115. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
116. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
117. strcpy(uPass, receiveBuf);
119. client = **new** User(uName, uPass, 1, result);
120. pmd->clientUser = client;
121. **if** (result == 0)
122. {
123. pmd->state = LOGINED;
124. menuFlag = 0;
125. strcpy(sendBuf, "登陆成功\n");
126. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
127. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
128. Sleep(10);
129. **break**;
130. }
131. **else** **if** (result == 1)
132. {
133. itoa(tryCount, tmpStr, 10);
134. strcpy(sendBuf, "登陆失败，");
135. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
136. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
137. Sleep(10);
138. strcpy(sendBuf, "用户不存在，还可以尝试");
139. strcat(sendBuf, tmpStr);
140. strcat(sendBuf, "次\n");
141. memset(tmpStr, 0, **sizeof**(tmpStr));
142. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
143. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
144. **delete** client;
145. Sleep(10);
146. }
147. **else** **if** (result == 2)
148. {
149. itoa(tryCount, tmpStr, 10);
150. strcpy(sendBuf, "登陆失败，");
151. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
152. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
153. Sleep(10);
154. strcpy(sendBuf, "密码错误，还可以尝试");
155. strcat(sendBuf, tmpStr);
156. strcat(sendBuf, "次\n");
157. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
158. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
159. **delete** client;
160. Sleep(10);
161. }
162. **else** **if** (result == 3)
163. {
164. itoa(tryCount, tmpStr, 10);
165. strcpy(sendBuf, "登陆失败，");
166. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
167. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
168. Sleep(10);
169. strcpy(sendBuf, "请勿重复登陆，还可以尝试");
170. strcat(sendBuf, tmpStr);
171. strcat(sendBuf, "次\n");
172. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
173. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
174. **delete** client;
175. Sleep(10);
176. }
177. tryCount--;
178. }
179. }
180. **else** **if** (strcmp("2", receiveBuf) == 0)//注册
181. {
182. cout << "用户注册" << endl;
183. **while** (1)
184. {
185. strcpy(sendBuf, "请输入用户名(32字符以下)：\n");
186. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
187. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
188. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
189. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
190. strcpy(uName, receiveBuf);
191. strcpy(sendBuf, "请输入密码(32字符以下)：\n");
192. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
193. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
194. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
195. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
196. strcpy(uPass, receiveBuf);
197. client = **new** User(uName, uPass, 0, result);
198. **if** (result == 0)//注册成功
199. {
200. pmd->clientUser = client;
201. pmd->state = LOGINED;
202. menuFlag = 0;
203. strcpy(sendBuf, "注册成功，已登陆\n");
204. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
205. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
206. Sleep(10);
207. userNum++;
208. strcpy(sendBuf, "获得3只初始宝可梦\n");
209. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
210. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
211. Sleep(10);
212. **break**;
213. }
214. **else** **if** (result == 1)//注册失败，用户名重复
215. {
216. strcpy(sendBuf, "注册失败，");
217. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
218. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
219. Sleep(10);
220. strcpy(sendBuf, "用户名重复\n");
221. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
222. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
223. Sleep(10);
224. }
225. }
226. }
227. **else** **if** (strcmp("3", receiveBuf) == 0)//查看用户信息
228. {
229. strcpy(sendBuf, "用户信息如下：\n");
230. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
231. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
232. Sleep(10);
233. **char** dataName[100] = { 0 };
234. **int** dataNum = 0;
235. **int** bufNum = 0;
236. ifstream userFile("./DATA/userData.txt");
237. userFile.getline(dataName, 100);
238. userFile.getline(dataName, 100);
239. **for** (**int** i = 0; i < userNum; i++)
240. {
241. userFile >> dataName;
242. //cout << "用户：" << dataName << ", ";
243. strcpy(sendBuf, "用户：");
244. strcat(sendBuf, dataName);
245. strcat(sendBuf, ", ");
246. **while** (strcmp("#pokemon#", dataName) != 0)//定位宝可梦数据位置
247. {
248. userFile >> dataName;
249. }
250. userFile >> dataNum;
251. //cout << "有" << dataNum << "只宝可梦：" << endl;
252. itoa(dataNum, tmpStr, 10);
253. strcat(sendBuf, "有");
254. strcat(sendBuf, tmpStr);
255. strcat(sendBuf, "只宝可梦：\n");
256. memset(tmpStr, 0, **sizeof**(tmpStr));
257. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
258. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
259. Sleep(10);
260. **for** (**int** j = 0; j < dataNum; j++)
261. {
262. //cout << j + 1 << ".";
263. userFile >> dataName; //cout << "宝可梦种族：" << dataName << ", ";
264. strcpy(sendBuf, "宝可梦种族：");
265. strcat(sendBuf, dataName);
266. strcat(sendBuf, ", ");
267. userFile >> bufNum; //cout << data[0] << endl; //元素
268. userFile >> bufNum;//类型
269. userFile >> dataName; //cout << "宝可梦名字：" << dataName << ", ";
270. strcat(sendBuf, "宝可梦名字：");
271. strcat(sendBuf, dataName);
272. strcat(sendBuf, ", ");
273. userFile >> bufNum;//EXP
274. userFile >> bufNum; //cout << "等级：" << bufNum << endl;//LV
275. itoa(bufNum, tmpStr, 10);
276. strcat(sendBuf, "等级：");
277. strcat(sendBuf, tmpStr);
278. strcat(sendBuf, "\n");
279. userFile >> bufNum;//ATK
280. userFile >> bufNum;//HP
281. userFile >> bufNum;//DEF
282. userFile >> bufNum;//SPEED
283. userFile >> bufNum;//Index
284. userFile >> dataName;
285. userFile.getline(dataName, 100);
286. memset(tmpStr, 0, **sizeof**(tmpStr));
287. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
288. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
289. Sleep(10);
290. }
291. //cout << endl;
292. userFile >> dataName;
293. userFile.getline(dataName, 100);
294. }
295. Sleep(10);
296. strcpy(sendBuf, "###end###");
297. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
298. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
299. userFile.close();
300. Sleep(10);
301. }
302. **else** **if** (strcmp("4", receiveBuf) == 0)//查看在线用户
303. {
304. //cout << "当前在线用户：" << endl;
305. CLIENT\_DATA\* showData = allClients;
306. **char** thisName[32] = { 0 };
307. **while** (showData)
308. {
309. **if** (showData->state == LOGINED)
310. {
311. showData->clientUser->getUsername(thisName);
312. strcpy(sendBuf, thisName);
313. strcat(sendBuf, "\n");
314. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
315. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
316. //cout << thisName << endl;
317. Sleep(10);
318. }
319. showData = showData->nextData;
320. }
321. Sleep(10);
322. strcpy(sendBuf, "###end###");
323. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
324. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
325. Sleep(10);
326. }
327. **else** **if** (strcmp("5", receiveBuf) == 0)//退出
328. {
329. pmd->state = UNLOGIN;
330. menuFlag = 0;
331. **break**;
332. }
333. **else**
334. {
335. cout << "指令错误" << endl;
336. }
337. }
338. }
339. //system("pause");
341. //退出主菜单，进入子菜单或结束进程
342. **int** battleWith = 0;//与服务器对战的宝可梦编号
343. Pokemon\* servPokem;//给服务器新建宝可梦
344. Pokemon\* clntPokem;//给用户新建宝可梦
345. **if** (pmd->state == LOGINED)//登陆成功
346. {
347. **while** (1)
348. {
349. strcpy(sendBuf, "功能选择：\n"
350. "1.进行对战\n"
351. "2.查看胜率\n"
352. "3.查看徽章\n"
353. "4.退出登陆并结束程序\n");
354. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
355. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
356. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
357. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
358. **if** (receiveLen < 0)
359. {
360. cout << "接收失败,客户机已断开" << endl;
361. **break**;
362. }
363. **else**
364. {
365. **if** (strcmp("1", receiveBuf) == 0)//对战
366. {
367. //client->loadPokemon();
368. //client->selectPokemon(clntSock);
369. //system("pause");
370. **while** (1)
371. {
372. **char** tmpName[32] = { 0 };
373. **int** tmpStatus[7] = { 0 };
374. **int** autoBattle = 0;//是否自动对战 0否1是
375. **int** battleType = 0;//战斗类型 0升级1决斗
377. strcpy(sendBuf, "选择比赛类型：\n"
378. "1.升级赛\n"
379. "2.决斗赛\n");
380. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
381. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
382. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
383. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
384. **if** (receiveLen < 0) cout << "接收失败,客户机已断开" << endl;
385. **else**
386. {
387. **if** (strcmp("1", receiveBuf) == 0)battleType = 0;
388. **else** **if** (strcmp("2", receiveBuf) == 0)battleType = 1;
389. **else** cout << "指令错误" << endl;
390. }
391. strcpy(sendBuf, "选择战斗模式：\n"
392. "1.自动战斗(系统随机使用双方宝可梦的技能)\n"
393. "2.手动战斗(用户自行选择双方宝可梦的技能)\n");
394. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
395. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
396. Sleep(10);
397. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
398. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
399. **if** (receiveLen < 0)
400. {
401. cout << "接收失败,客户机已断开" << endl;
402. **break**;
403. }
404. **else**
405. {
406. **if** (strcmp("1", receiveBuf) == 0)autoBattle = 1;
407. **else** **if** (strcmp("2", receiveBuf) == 0)autoBattle = 0;
408. **else** cout << "指令错误" << endl;
409. }
410. strcpy(sendBuf, "选择对战宝可梦：\n");
411. strcat(sendBuf, virtualPokemonList);
412. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
413. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
414. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
415. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
416. **if** (receiveLen < 0)
417. {
418. cout << "接收失败,客户机已断开" << endl;
419. **break**;
420. }
421. **else**
422. {
423. battleWith = atoi(receiveBuf);
424. battleWith--;
425. }
426. **switch** (battleWith)
427. {
428. **case** 0:
429. servPokem = **new** Bulbasaur(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
430. **break**;
431. **case** 1:
432. servPokem = **new** Charizard(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
433. **break**;
434. **case** 2:
435. servPokem = **new** Squirtle(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
436. **break**;
437. **case** 3:
438. servPokem = **new** Caterpie(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
439. **break**;
440. **case** 4:
441. servPokem = **new** Pidgeotto(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
442. **break**;
443. **case** 5:
444. servPokem = **new** Diglett(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
445. **break**;
446. **case** 6:
447. servPokem = **new** Pikachu(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
448. **break**;
449. **case** 7:
450. servPokem = **new** Snorunt(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
451. **break**;
452. **case** 8:
453. servPokem = **new** Dragonite(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
454. **break**;
455. **case** 9:
456. servPokem = **new** Pikachu(virtualPokemonName[battleWith], virtualPokemonData[battleWith]);
457. **break**;
458. **default**:
459. servPokem = **new** Bulbasaur(virtualPokemonName[0], virtualPokemonData[0]);
460. **break**;
461. }
463. client->loadPokemon();//载入宝可梦
464. client->selectPokemon(clntSock);//选择出战宝可梦
465. clntPokem = client->usingPokemon;//使用宝可梦
467. **while** ((\*clntPokem).getHP() > 0 && (\*servPokem).getHP() > 0)//如果双方均可继续战斗
468. {
469. //竞速制
470. **if** (round % (\*servPokem).getSPEED() == 0)
471. {
472. (\*servPokem).attack((\*clntPokem), autoBattle, clntSock);
473. **if** ((\*clntPokem).getHP() == 0) **break**;
474. Sleep(101);
475. }
476. **if** (round % (\*clntPokem).getSPEED() == 0)
477. {
478. (\*clntPokem).attack((\*servPokem), autoBattle, clntSock);
479. **if** ((\*servPokem).getHP() == 0) **break**;
480. Sleep(101);
481. }
482. **if** (round < MAX\_32\_INT)round++;
483. **else** round = 0;
485. /\*回合制
486. if ((\*servPokem).getSPEED() <= (\*clntPokem).getSPEED())//速度快的宝可梦先手，一样则服务器先手
487. {
488. cout << "服务器先手" << endl;
489. (\*servPokem).attack((\*clntPokem));
490. if ((\*clntPokem).getHP() == 0) break;
491. (\*clntPokem).attack((\*servPokem));
492. cout << "回合结束，进入下一回合" << endl << endl;
493. }
494. else
495. {
496. cout << "客户端先手" << endl;
497. (\*clntPokem).attack((\*servPokem));
498. if ((\*servPokem).getHP() == 0) break;
499. (\*servPokem).attack((\*clntPokem));
500. cout << "回合结束，进入下一回合" << endl << endl;
501. }
502. \*/
503. }
504. strcpy(sendBuf, "#battleOver#");
505. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
506. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
507. Sleep(10);
508. //对战结束，收益计算
509. **if** ((\*clntPokem).getHP() == 0)//如果客户宝可梦无法战斗，则服务器胜利
510. {
511. client->allBattlePlus();
512. **if** (battleType == 0)
513. {
514. (\*clntPokem).getStatus(tmpName, tmpStatus);
515. cout << "客户战败，" << tmpName << "获得了4点经验值" << endl;
516. strcpy(sendBuf, "战斗失败，");
517. strcat(sendBuf, tmpName);
518. strcat(sendBuf, "获得了4点经验值\n");
519. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
520. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
521. Sleep(10);
522. (\*clntPokem).levelUp(clntSock, 4);
523. }
524. **else** **if** (battleType == 1)
525. {
526. (\*clntPokem).getStatus(tmpName, tmpStatus);
527. cout << "客户战败，" << tmpName << "获得了4点经验值" << endl;
528. strcpy(sendBuf, "战斗失败，");
529. strcat(sendBuf, tmpName);
530. strcat(sendBuf, "获得了4点经验值\n");
531. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
532. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
533. Sleep(10);
534. (\*clntPokem).levelUp(clntSock, 4);
535. client->lossPokemon(clntSock);
536. }
537. }
538. **else**//客户胜利
539. {
540. client->allBattlePlus();
541. client->winBattlePlus();
542. (\*servPokem).recover();
543. **if** (battleType == 0)
544. {
545. (\*clntPokem).getStatus(tmpName, tmpStatus);
546. cout << "客户胜利，" << tmpName << "获得了15点经验值" << endl;
547. strcpy(sendBuf, "战斗胜利，");
548. strcat(sendBuf, tmpName);
549. strcat(sendBuf, "获得了15点经验值\n");
550. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
551. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
552. Sleep(10);
553. (\*clntPokem).levelUp(clntSock, 15);
554. }
555. **else** **if** (battleType == 1)
556. {
557. (\*clntPokem).getStatus(tmpName, tmpStatus);
558. cout << "客户胜利，" << tmpName << "获得了15点经验值" << endl;
559. strcpy(sendBuf, "战斗胜利，");
560. strcat(sendBuf, tmpName);
561. strcat(sendBuf, "获得了15点经验值\n");
562. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
563. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
564. Sleep(10);
565. (\*clntPokem).levelUp(clntSock, 15);
566. client->getPokemon(servPokem);
567. strcpy(sendBuf, "已获得对战宝可梦\n");
568. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
569. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
570. Sleep(10);
571. }
572. }
573. //恢复宝可梦
574. (\*servPokem).recover();
575. (\*clntPokem).recover();
576. client->saveAllToFile();
577. strcpy(sendBuf, "继续下一场战斗？（Y/N）：");
578. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
579. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
580. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
581. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
582. **if** (strcmp("N", receiveBuf) == 0 || strcmp("n", receiveBuf) == 0)
583. {
584. **break**;
585. }
586. }
587. }
588. **else** **if** (strcmp("2", receiveBuf) == 0)//查询胜率
589. {
590. **while** (1)
591. {
592. strcpy(sendBuf, "输入要查询胜率的用户名：");
593. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
594. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
595. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
596. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
597. winRate = client->seeWinRate(receiveBuf);
598. **if** (winRate != -1)
599. {
600. winRate = winRate \* 100;
601. **char** tmpRate[1024] = { 0 };
602. strcpy(sendBuf, "#seeStart#");
603. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
604. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
605. Sleep(10);
606. ofstream tmpFile("./TMP/tmp.txt", ios::out);
607. tmpFile << winRate;
608. tmpFile.close();
609. ifstream tmpFile1("./TMP/tmp.txt");
610. tmpFile1.getline(tmpRate, **sizeof**(tmpRate));
611. strcpy(sendBuf, "胜率是");
612. strcat(sendBuf, tmpRate);
613. strcat(sendBuf, "%\n");
614. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
615. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
616. **break**;
617. }
618. **else**
619. {
620. strcpy(sendBuf, "查无此人，是否重新输入？（Y/N）：");
621. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
622. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
623. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
624. receiveLen = recv(\*clntSock, receiveBuf, **sizeof**(receiveBuf), NULL);
625. **if** (strcmp("N", receiveBuf) == 0 || strcmp("n", receiveBuf) == 0)
626. {
627. **break**;
628. }
629. }
630. }
631. }
632. **else** **if** (strcmp("3", receiveBuf) == 0)//查询徽章
633. {
634. **switch**(client->getquantityBadge())
635. {
636. **case** GOLD:
637. strcpy(sendBuf, "拥有金质“宠物个数徽章”，");
638. **break**;
639. **case** SILVER:
640. strcpy(sendBuf, "拥有银质“宠物个数徽章”，");
641. **break**;
642. **case** COPPER:
643. strcpy(sendBuf, "拥有铜质“宠物个数徽章”，");
644. **break**;
645. **default**:
646. strcpy(sendBuf, "没有“宠物个数徽章”，");
647. }
648. **switch** (client->getadvancedBadge())
649. {
650. **case** GOLD:
651. strcat(sendBuf, "拥有金质“高级宠物徽章”\n");
652. **break**;
653. **case** SILVER:
654. strcat(sendBuf, "拥有银质“高级宠物徽章”\n");
655. **break**;
656. **case** COPPER:
657. strcat(sendBuf, "拥有铜质“高级宠物徽章”\n");
658. **break**;
659. **default**:
660. strcat(sendBuf, "没有“高级宠物徽章”\n");
661. }
662. sendLen = send(\*clntSock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
663. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
664. }
665. **else** **if** (strcmp("4", receiveBuf) == 0)//退出
666. {
667. **break**;
668. }
669. }
670. }
671. }
672. **else**
673. {
674. cout << "用户未登录" << endl;
675. }
677. //善后工作，清除此客户
678. CLIENT\_DATA\* tmpClientData = NULL;
679. tmpClientData = pmd;
680. **if** (pmd->prevData)//如果前面还有客户
681. pmd->prevData->nextData = pmd->nextData;
682. **else**
683. allClients = pmd->nextData;
684. **if** (pmd->nextData)//如果后面还有客户
685. pmd->nextData->prevData = pmd->prevData;
686. free(pmd);
687. activeUserNum--;
689. //输出当前活跃用户数
690. cout << "activeUserNum:" << activeUserNum << endl;
691. cout << "进程结束" << endl << endl;
692. **return** 0;
693. }
695. //主函数
696. **int** main(**int** argc, **char**\* argv[])
697. {
698. cout << "System is Initializing..." << endl;
700. CLIENT\_DATA\* preClient = NULL;//上一个操作的客户端
701. CLIENT\_DATA\* nowClient = NULL;//当前操作的用户客户端
703. //socket相关初始化
704. **int** error;//错误警告位
705. **int** sendLen;//成功发送长度
706. **int** receiveLen;//成功接收长度
707. **char** sendBuf[MAX\_BUF\_SIZE] = "s";//发送缓冲区
708. **char** receiveBuf[MAX\_BUF\_SIZE] = { 0 };//接收缓冲区
709. u\_short port = 1145;//端口 默认1145
710. **char** ip[16] = "127.0.0.1";//ip 默认本机(127.0.0.1)
712. **if** (argc > 1)//获取非默认ip与端口
713. {
714. strcpy(ip, argv[1]);
715. cout << "Server IP is : " << ip << endl;
716. **if** (argc > 2)
717. {
718. port = std::stoi(argv[2]);
719. cout << "Server Port is : " << port << endl;
720. }
721. **else**
722. {
723. cout << "Server Port is : 1145 by default" << endl;
724. }
725. }
726. **else**
727. {
728. cout << "Server IP is : 127.0.0.1 by default" << endl;
729. cout << "Server Port is : 1145 by default" << endl;
730. }
732. //初始化虚拟精灵列表
733. virtualPokemonList = (**char**\*)malloc(1000 \* **sizeof**(**char**));
734. **for** (**int** k = 0; k < 10; k++)
735. {
736. virtualPokemonName[k] = (**char**\*)malloc(100 \* **sizeof**(**char**));
737. virtualPokemonData[k] = (**int**\*)malloc(9 \* **sizeof**(**int**));
738. }
739. strcpy(virtualPokemonList,  "1.Bulbasaur(妙蛙种子), LV=1, HP=12, ATK=2, DEF=1, SPEED=2.\n"
740. "2.Charizard(喷火龙),  LV=1, HP=11, ATK=3, DEF=1, SPEED=2.\n"
741. "3.Squirtle(杰尼龟), LV=1, HP=10, ATK=2, DEF=2, SPEED=2.\n"
742. "4.Caterpie(绿毛虫), LV=1, HP=10, ATK=3, DEF=1, SPEED=2.\n"
743. "5.Pidgeotto(比比鸟),  LV=1, HP=10, ATK=2, DEF=1, SPEED=1.\n"
744. "6.Diglett(地鼠), LV=1, HP=13, ATK=2, DEF=1, SPEED=2.\n"
745. "7.Pikachu(皮卡丘), LV=5, HP=30, ATK=12, DEF=5, SPEED=1.\n"
746. "8.Snorunt(雪童子), LV=5, HP=27, ATK=15, DEF=3, SPEED=2.\n"
747. "9.Dragonite(快龙), LV=10, HP=50, ATK=22, DEF=11, SPEED=2.\n"
748. "10.Pikachu(皮卡丘), LV=15, HP=75, ATK=30, DEF=15, SPEED=1.\n");
749. strcpy(virtualPokemonName[0], "Bulbasaur");
750. strcpy(virtualPokemonName[1], "Charizard");
751. strcpy(virtualPokemonName[2], "Squirtle");
752. strcpy(virtualPokemonName[3], "Caterpie");
753. strcpy(virtualPokemonName[4], "Pidgeotto");
754. strcpy(virtualPokemonName[5], "Diglett");
755. strcpy(virtualPokemonName[6], "Pikachu");
756. strcpy(virtualPokemonName[7], "Snorunt");
757. strcpy(virtualPokemonName[8], "Dragonite");
758. strcpy(virtualPokemonName[9], "Pikachu(Powerful)");
759. virtualPokemonData[0][0] = GRASS;
760. virtualPokemonData[0][1] = BEARABLE;
761. virtualPokemonData[0][2] = 0;
762. virtualPokemonData[0][3] = 1;
763. virtualPokemonData[0][4] = 2;
764. virtualPokemonData[0][5] = 12;
765. virtualPokemonData[0][6] = 1;
766. virtualPokemonData[0][7] = 2;
767. virtualPokemonData[0][8] = BLUBASAUR;
768. virtualPokemonData[1][0] = FIRE;
769. virtualPokemonData[1][1] = POWER;
770. virtualPokemonData[1][2] = 0;
771. virtualPokemonData[1][3] = 1;
772. virtualPokemonData[1][4] = 3;
773. virtualPokemonData[1][5] = 11;
774. virtualPokemonData[1][6] = 1;
775. virtualPokemonData[1][7] = 2;
776. virtualPokemonData[1][8] = CHARIZARD;
777. virtualPokemonData[2][0] = WATER;
778. virtualPokemonData[2][1] = DEDENSIVE;
779. virtualPokemonData[2][2] = 0;
780. virtualPokemonData[2][3] = 1;
781. virtualPokemonData[2][4] = 2;
782. virtualPokemonData[2][5] = 10;
783. virtualPokemonData[2][6] = 2;
784. virtualPokemonData[2][7] = 2;
785. virtualPokemonData[2][8] = SQUIRTLE;
786. virtualPokemonData[3][0] = BUG;
787. virtualPokemonData[3][1] = POWER;
788. virtualPokemonData[3][2] = 0;
789. virtualPokemonData[3][3] = 1;
790. virtualPokemonData[3][4] = 3;
791. virtualPokemonData[3][5] = 10;
792. virtualPokemonData[3][6] = 1;
793. virtualPokemonData[3][7] = 2;
794. virtualPokemonData[3][8] = CATERPIE;
795. virtualPokemonData[4][0] = FLY;
796. virtualPokemonData[4][1] = AGILE;
797. virtualPokemonData[4][2] = 0;
798. virtualPokemonData[4][3] = 1;
799. virtualPokemonData[4][4] = 2;
800. virtualPokemonData[4][5] = 10;
801. virtualPokemonData[4][6] = 1;
802. virtualPokemonData[4][7] = 1;
803. virtualPokemonData[4][8] = PIDGEOTTO;
804. virtualPokemonData[5][0] = GROUND;
805. virtualPokemonData[5][1] = BEARABLE;
806. virtualPokemonData[5][2] = 0;
807. virtualPokemonData[5][3] = 1;
808. virtualPokemonData[5][4] = 2;
809. virtualPokemonData[5][5] = 13;
810. virtualPokemonData[5][6] = 1;
811. virtualPokemonData[5][7] = 2;
812. virtualPokemonData[5][8] = DIGLETT;
813. virtualPokemonData[6][0] = ELECTRIC;
814. virtualPokemonData[6][1] = AGILE;
815. virtualPokemonData[6][2] = 0;
816. virtualPokemonData[6][3] = 5;
817. virtualPokemonData[6][4] = 12;
818. virtualPokemonData[6][5] = 30;
819. virtualPokemonData[6][6] = 5;
820. virtualPokemonData[6][7] = 1;
821. virtualPokemonData[6][8] = PIKACHU;
822. virtualPokemonData[7][0] = ICE;
823. virtualPokemonData[7][1] = POWER;
824. virtualPokemonData[7][2] = 0;
825. virtualPokemonData[7][3] = 5;
826. virtualPokemonData[7][4] = 15;
827. virtualPokemonData[7][5] = 27;
828. virtualPokemonData[7][6] = 3;
829. virtualPokemonData[7][7] = 2;
830. virtualPokemonData[7][8] = SNORUNT;
831. virtualPokemonData[8][0] = DRAGON;
832. virtualPokemonData[8][1] = POWER;
833. virtualPokemonData[8][2] = 0;
834. virtualPokemonData[8][3] = 10;
835. virtualPokemonData[8][4] = 22;
836. virtualPokemonData[8][5] = 50;
837. virtualPokemonData[8][6] = 11;
838. virtualPokemonData[8][7] = 2;
839. virtualPokemonData[8][8] = DRAGONITE;
840. virtualPokemonData[9][0] = ELECTRIC;
841. virtualPokemonData[9][1] = AGILE;
842. virtualPokemonData[9][2] = 0;
843. virtualPokemonData[9][3] = 15;
844. virtualPokemonData[9][4] = 30;
845. virtualPokemonData[9][5] = 75;
846. virtualPokemonData[9][6] = 15;
847. virtualPokemonData[9][7] = 1;
848. virtualPokemonData[9][8] = PIKACHU;

851. //初始化 DLL
852. WSADATA wsaData;
853. error = WSAStartup(MAKEWORD(2, 2), &wsaData);
854. **if** (error != 0)
855. {
856. cout << "初始化套接字库失败" << endl;
857. **return** 0;
858. }
860. //创建套接字
861. SOCKET servSock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);
863. //绑定套接字
864. sockaddr\_in sockAddr;
865. memset(&sockAddr, 0, **sizeof**(sockAddr));  //每个字节都用0填充
866. sockAddr.sin\_family = PF\_INET;  //使用IPv4地址
867. sockAddr.sin\_addr.s\_addr = inet\_addr(ip);  //本机
868. sockAddr.sin\_port = htons(port);  //端口
869. **if** (bind(servSock, (SOCKADDR\*)&sockAddr, **sizeof**(SOCKADDR)) == SOCKET\_ERROR)
870. {
871. cout << "绑定套接字失败" << endl;
872. closesocket(servSock);
873. WSACleanup();
874. **return** 0;
875. }
877. //进入监听状态
878. **if** (listen(servSock, 100) < 0)
879. {
880. cout << "进入监听失败" << endl;
881. closesocket(servSock);
882. WSACleanup();
883. **return** 0;
884. }
886. cout << "System startup is complete. Now, listening client" << endl;
888. **int** tmpData[9] = { FIRE,POWER,0,1,2,10,1,2,2 };//初始化数据
890. allClients = (CLIENT\_DATA\*)malloc(**sizeof**(CLIENT\_DATA));
891. **while** (1)
892. {
893. //循环接收客户端连接请求并创建服务线程
894. SOCKET\* clntSock = **new** SOCKET;
895. SOCKADDR clntAddr;
896. **int** nSize = **sizeof**(SOCKADDR);
897. \*clntSock = accept(servSock, (SOCKADDR\*)&clntAddr, &nSize);
899. **if** (activeUserNum == 0)
900. {
901. allClients = (CLIENT\_DATA\*)malloc(**sizeof**(CLIENT\_DATA));
902. nowClient = allClients;
903. nowClient->clientSocket = NULL;
904. nowClient->clientUser = NULL;
905. nowClient->nextData = NULL;
906. nowClient->prevData = NULL;
907. nowClient->state = UNLOGIN;
908. }
909. **else**
910. {
911. nowClient = (CLIENT\_DATA\*)malloc(**sizeof**(CLIENT\_DATA));
912. nowClient->clientSocket = NULL;
913. nowClient->clientUser = NULL;
914. nowClient->nextData = NULL;
915. nowClient->prevData = preClient;
916. nowClient->state = UNLOGIN;
917. preClient->nextData = nowClient;
918. }
920. nowClient->clientSocket = clntSock;
922. **if** (\*clntSock == SOCKET\_ERROR)
923. {
924. cout << "接收连接请求失败" << endl;
925. closesocket(\*clntSock);
926. }
927. **else**
928. {
929. cout << endl << "一个客户端尝试连接到服务器，socket是：" << \*clntSock << endl;
930. CreateThread(NULL, 0, &ServerThread, nowClient, 0, NULL);
931. }
933. activeUserNum++;
934. preClient = nowClient;
935. }
936. //关闭套接字
937. closesocket(servSock);
939. //终止 DLL 的使用
940. WSACleanup();
942. //释放申请的空间
943. free(virtualPokemonList);
944. **for** (**int** k = 0; k < 10; k++)
945. {
946. free(virtualPokemonName[k]);
947. free(virtualPokemonData[k]);
948. }
950. **return** 0;
951. }

## 10.综合（客户端）

### 文件：main.cpp

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2. /\* @课程名：        C++课程设计 题目二 用户注册与平台登录   \*/
3. /\* @作者；         谢睿（2018211208）                      \*/
4. /\* @教师：         待定                                  \*/
5. /\* @最后修改时间： 2020/08/22 17：50                        \*/
6. /\* @文件名：        main.cpp                                \*/
7. /\* @文件说明：       主功能代码，用于测试                  \*/
8. /\*----------------------------------------------------------\*/
9. /\* @备注：         本程序遵循任天堂公司习惯用语，使用“宝可\*/
10. /\*                  梦(Pokemon)”代指宠物小精灵              \*/
11. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
13. #include<iostream>
14. #include<fstream>
15. #include<string>
16. #include <winsock2.h>
17. #include <string.h>
18. **using** **namespace** std;
20. #pragma comment (lib, "ws2\_32.lib")  //加载 ws2\_32.dll

23. //主函数
24. **int** main(**int** argc, **char**\* argv[])
25. {
26. cout << "System is Initializing..." << endl;
28. **char** tmpStr[100] = { 0 };
29. **int** continueFlag = 1;
30. **int** loginFlag = 0;
32. //socket相关初始化
33. **int** error;//错误警告位
34. **int** sendLen;//成功发送长度
35. **int** receiveLen;//成功接收长度
36. **char** sendBuf[1000] = "c";//发送缓冲区
37. **char** receiveBuf[1000] = { 0 };//接收缓冲区
38. u\_short port = 1145;//端口 默认1145
39. **char** ip[16] = "127.0.0.1";//ip 默认本机(127.0.0.1)
41. **if** (argc > 1)//获取非默认ip与端口
42. {
43. strcpy(ip, argv[1]);
44. cout << "Server IP is : " << ip << endl;
45. **if** (argc > 2)
46. {
47. port = std::stoi(argv[2]);
48. cout << "Server Port is : " << port << endl;
49. }
50. **else**
51. {
52. cout << "Server Port is : 1145 by default" << endl;
53. }
54. }
55. **else**
56. {
57. cout << "Server IP is : 127.0.0.1 by default" << endl;
58. cout << "Server Port is : 1145 by default" << endl;
59. }
61. //初始化 DLL
62. WSADATA wsaData;
63. error = WSAStartup(MAKEWORD(2, 2), &wsaData);
64. **if** (error != 0)
65. {
66. cout << "初始化套接字库失败" << endl;
67. **return** 0;
68. }
70. //创建套接字
71. SOCKET sock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);
73. //向服务器发起请求
74. sockaddr\_in sockAddr;
75. memset(&sockAddr, 0, **sizeof**(sockAddr));  //每个字节都用0填充
76. sockAddr.sin\_family = PF\_INET;
77. sockAddr.sin\_addr.s\_addr = inet\_addr(ip);
78. sockAddr.sin\_port = htons(port);
79. connect(sock, (SOCKADDR\*)&sockAddr, **sizeof**(SOCKADDR));
81. cout << endl << "连接服务器" << endl << endl;
83. //功能选择
84. **while**(1)
85. {
86. //接收服务器传回的数据
87. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
88. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
89. **if** (receiveLen < 0)
90. {
91. cout << "接收失败,服务器已断开" << endl;
92. **break**;
93. }
94. **else**
95. {
96. cout << receiveBuf;
97. }
99. //向服务器发送数据
100. cin >> sendBuf;
101. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
102. strcpy(tmpStr, sendBuf);
103. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
104. **if** (sendLen < 0) cout << "发送失败" << endl;
105. **if** (strcmp("1", tmpStr) == 0)//登陆
106. {
107. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
108. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
109. **if** (receiveLen < 0)
110. {
111. cout << "接收失败,服务器已断开" << endl;
112. **break**;
113. }
114. **else**
115. {
116. cout << receiveBuf;
117. }
118. **while** (strcmp("登陆成功\n", receiveBuf) != 0)
119. {
120. cin >> sendBuf;
121. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
122. strcpy(tmpStr, sendBuf);
123. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
124. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
125. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
126. **if** (receiveLen < 0)
127. {
128. cout << "接收失败,服务器已断开" << endl;
129. **break**;
130. }
131. **else**
132. {
133. cout << receiveBuf;
134. }
135. **if** (strcmp("登陆失败，", receiveBuf) == 0)
136. {
137. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
138. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
139. cout << receiveBuf;
140. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
141. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
142. cout << receiveBuf;
143. }
144. }
145. loginFlag = 1;
146. **break**;
147. }
148. **else** **if** (strcmp("2", tmpStr) == 0)//注册
149. {
150. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
151. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
152. **if** (receiveLen < 0)
153. {
154. cout << "接收失败,服务器已断开" << endl;
155. **break**;
156. }
157. **else**
158. {
159. cout << receiveBuf;
160. }
161. **while** (strcmp("注册成功，已登陆\n", receiveBuf) != 0)
162. {
163. cin >> sendBuf;
164. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
165. strcpy(tmpStr, sendBuf);
166. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
167. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
168. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
169. **if** (receiveLen < 0)
170. {
171. cout << "接收失败,服务器已断开" << endl;
172. **break**;
173. }
174. **else**
175. {
176. cout << receiveBuf;
177. }
178. **if** (strcmp("注册失败，", receiveBuf) == 0)
179. {
180. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
181. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
182. cout << receiveBuf;
183. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
184. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
185. cout << receiveBuf;
186. }
187. }
188. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
189. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
190. cout << receiveBuf;
191. loginFlag = 1;
192. **break**;
193. }
194. **else** **if** (strcmp("3", tmpStr) == 0)
195. {
196. **while** (1)
197. {
198. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
199. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
200. **if** (strcmp("###end###", receiveBuf) == 0) **break**;
201. **else** cout << receiveBuf;
202. }
203. }
204. **else** **if** (strcmp("4", tmpStr) == 0)
205. {
206. cout << "当前在线用户：" << endl;
207. **while** (1)
208. {
209. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
210. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
211. **if** (strcmp("###end###", receiveBuf) == 0) **break**;
212. **else** cout << receiveBuf;
213. }
214. }
215. **else** **if** (strcmp("5", tmpStr) == 0)
216. {
217. cout << "已断开" << endl;
218. **break**;
219. }
220. cout << endl;
221. }
223. //登陆成功
224. **while** (loginFlag)
225. {
226. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
227. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
228. cout << endl << receiveBuf;
229. cin >> sendBuf;
230. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
231. strcpy(tmpStr, sendBuf);
232. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
233. **if** (sendLen < 0) cout << "发送失败" << endl;
234. **if** (strcmp("1", tmpStr) == 0)//对战
235. {
236. //开始对战
237. **int** autoBattle = 0;//是否自动对战 0否1是
238. **int** battleType = 0;//战斗类型 0升级1决斗
239. **while** (continueFlag)
240. {
241. //记录战斗模式
242. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
243. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
244. cout << receiveBuf;
245. cin >> sendBuf;
246. battleType = atoi(sendBuf);
247. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
248. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
249. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
250. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
251. cout << receiveBuf;
252. cin >> sendBuf;
253. **if** (strcmp("1", sendBuf) == 0)autoBattle = 1;
254. **else** **if** (strcmp("2", sendBuf) == 0)autoBattle = 0;
255. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
256. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
258. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
259. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
260. cout << receiveBuf;
261. cin >> sendBuf;
262. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
263. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
265. **while** (1)
266. {
267. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
268. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
269. **if** (strcmp("#selectEnd#", receiveBuf) == 0)
270. {
271. **break**;
272. }
273. **else**
274. {
275. cout << receiveBuf;
276. cin >> sendBuf;
277. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
278. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
279. }
280. }
282. **if** (autoBattle == 0)//非自动
283. {
284. **while** (1)
285. {
286. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
287. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
288. **if** (strcmp("#battleOver#", receiveBuf) == 0)
289. {
290. **break**;
291. }
292. **else**
293. {
294. **while** (1)
295. {
296. **if** (strcmp("#skillOver#", receiveBuf) == 0)
297. {
298. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
299. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
300. cout << receiveBuf;
301. **break**;
302. }
303. **else**
304. {
305. cout << receiveBuf;
306. cin >> sendBuf;
307. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
308. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
309. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
310. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
311. }
312. }
313. }
314. }
315. }
316. **else**//自动
317. {
318. **while** (1)
319. {
320. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
321. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
322. **if** (strcmp("#battleOver#", receiveBuf) == 0)
323. {
324. **break**;
325. }
326. **else**
327. {
328. cout << receiveBuf;
329. }
330. }
331. }
333. //战斗结束
334. **while** (1)
335. {
336. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
337. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
338. **if** (strcmp("#forget#", receiveBuf) == 0)
339. {
340. **while** (1)
341. {
342. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
343. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
344. **if** (strcmp("#learnEnd#", receiveBuf) == 0 || strcmp("#allEnd#", receiveBuf) == 0)
345. {
346. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
347. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
348. cout << receiveBuf;
349. **break**;
350. }
351. **else**
352. {
353. cout << receiveBuf;
354. cin >> sendBuf;
355. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
356. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
357. }
358. }
359. **break**;
360. }
361. **else** **if** (strcmp("继续下一场战斗？（Y/N）：", receiveBuf) == 0)
362. {
363. cout << endl;
364. cout << receiveBuf;
365. cin >> sendBuf;
366. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
367. **if** (strcmp("N", sendBuf) == 0 || strcmp("n", sendBuf) == 0)
368. {
369. continueFlag = 0;
370. }
371. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
372. cout << endl;
373. **break**;
374. }
375. **else** **if** (strcmp("#lossPokemon#", receiveBuf) == 0)
376. {
377. **while** (1)
378. {
379. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
380. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
381. **if** (strcmp("#lossEnd#", receiveBuf) == 0) **break**;
382. **else**
383. {
384. cout << receiveBuf;
385. cin >> sendBuf;
386. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
387. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
388. }
389. }
390. }
391. **else**
392. {
393. cout << receiveBuf;
394. }
395. }
396. cout << endl;
397. }
398. continueFlag = 1;
399. }
400. **else** **if** (strcmp("2", tmpStr) == 0)
401. {
402. **while** (1)
403. {
404. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
405. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
406. **if** (strcmp("#seeStart#", receiveBuf) == 0)
407. {
408. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
409. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
410. cout << receiveBuf;
411. **break**;
412. }
413. **else**
414. {
415. cout << receiveBuf;
416. cin >> sendBuf;
417. sendLen = send(sock, sendBuf, strlen(sendBuf) + **sizeof**(**char**), NULL);
418. memset(sendBuf, 0, **sizeof**(sendBuf));  //清空缓冲区
419. }
420. }
421. }
422. **else** **if** (strcmp("3", tmpStr) == 0)
423. {
424. memset(receiveBuf, 0, **sizeof**(receiveBuf));  //清空缓冲区
425. receiveLen = recv(sock, receiveBuf, **sizeof**(receiveBuf), NULL);
426. cout << receiveBuf;
427. }
428. **else** **if** (strcmp("4", tmpStr) == 0)
429. {
430. **break**;
431. }
432. }


436. //关闭套接字
437. closesocket(sock);
439. //终止使用 DLL
440. WSACleanup();
442. cout << "程序结束，感谢使用" << endl;
444. **return** 0;
445. }

# 四．参考资料

1. 宝可梦列表：[宝可梦列表（按全国图鉴编号） - 神奇宝贝百科，关于宝可梦的百科全书](https://wiki.52poke.com/zh-hans/%E5%AE%9D%E5%8F%AF%E6%A2%A6%E5%88%97%E8%A1%A8%EF%BC%88%E6%8C%89%E5%85%A8%E5%9B%BD%E5%9B%BE%E9%89%B4%E7%BC%96%E5%8F%B7%EF%BC%89) <https://wiki.52poke.com/zh-hans/%E5%AE%9D%E5%8F%AF%E6%A2%A6%E5%88%97%E8%A1%A8%EF%BC%88%E6%8C%89%E5%85%A8%E5%9B%BD%E5%9B%BE%E9%89%B4%E7%BC%96%E5%8F%B7%EF%BC%89>
2. 宝可梦招式列表：[招式列表 - 神奇宝贝百科，关于宝可梦的百科全书](https://wiki.52poke.com/wiki/%E6%8B%9B%E5%BC%8F%E5%88%97%E8%A1%A8) <https://wiki.52poke.com/wiki/%E6%8B%9B%E5%BC%8F%E5%88%97%E8%A1%A8>
3. <https://blog.csdn.net/m0_37316917/article/details/82712017>
4. <https://www.cnblogs.com/Ybossy/p/13023600.html>
5. <https://blog.csdn.net/mao0514/article/details/85066811>
6. <https://blog.csdn.net/weixin_36049506/article/details/93333549>