软件工程 一班 马启凡 202430551135

**实验名称：类和对象**

**实验内容：**

请设计一个精灵类Spirit，其成员要求如下：

1.具有private的data members

（1）name，精灵的名字，键盘输入

（2）x，精灵的位置坐标X轴的值，键盘输入

（3）y，精灵的位置坐标Y轴的值，键盘输入

（4）health，精灵的health生命力初值设为1000

（5）aliveState，精灵的生存状态，初值设为true

2.具有静态数据成员

（1）number，记录已经创建的精灵数量，初值为0

（2）alivenumber，记录生存状态为true的精灵数量，初值为0

3.具有的public的member function

（1）构造函数1，无参数的构造函数，创建精灵对象并初始化，number+1，alivenumber+1

（2）构造函数2，有参数的构造函数（精灵的名字和位置坐标），创建对象并初始化，number+1，alivenumber+1

（3）takeDamage函数：实现对精灵的伤害，首先需要判断aliveState的值，如果是true，将health - damage，若伤害后的精灵health<=0，则health设为0，将aliveState设为false，alivenumber-1

（4）setPosition函数：需要判断新位置是否在0-200之间，如果是的话，就设置精灵新位置，否则报错，仍保持原位置，并输出错误提示信息。

（5）getPositionX函数：返回精灵的位置坐标X

（6）getPositionY函数：返回精灵的位置坐标Y

（7）getName函数：返回精灵的名字

（8）getHealth函数：返回精灵的生命值

（9）getNumber函数：返回精灵世界一共有多少精灵

（10）getAliveNumber函数：返回精灵世界一共有多少alive的精灵

（11）getInfo函数：输出精灵的各种信息，包括姓名、生命值、生存状态、坐标等

4.main函数功能

（1）在main函数中通过不同的构造函数创建不同精灵。

（2）调用takeDamage函数，测试精灵的生存状态。

（3）测试类中定义的所有函数。比如设置新位置，输出精灵的各种信息等。

（4）设置对象数组管理10个精灵并初始化

（5）用随机数模拟10次随机事件，即对精灵i的k点伤害，输出相关信息，随机事件结束后，输出所有对象的基本信息

5.选做任务

        （1）尝试在屏幕上用符号（比如“\*”）代表精灵，以精灵坐标为屏幕的行和列，在屏幕上“画”出所有精灵

        （2）对应各种事件在屏幕画中展现相应变化，比如新增精灵，精灵消亡，精灵移动

        （3）添加精灵事件，任何你希望精灵具备的技能

**实验过程**

**1.代码展示**

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

int number = 1;

int alivenumber = 1;

//创建精灵类

class Spirit

{

public:

Spirit()

{

}

Spirit(string name1, int X1, int Y1)

{

name = name1;

X = X1;

Y = Y1;

}

void takeDamage(int damage)

{

if (health > damage)

{

health = health - damage;

state = 0;

}

else

{

health = 0;

aliveState = false;

alivenumber--;

state = 0;

}

}

void setPosition(int x, int y)

{

X = x;

Y = y;

}

void getPositionX()

{

cout <<"X坐标：" << X << endl;

}

void getPositionY()

{

cout<<"Y坐标：" << Y << endl;

}

void getName()

{

cout <<"姓名：" << name << endl;

}

void getHealth()

{

cout<<"生命值：" << health << endl;

}

static void getNumber()

{

cout << number << endl;

}

static void getAliveNumber()

{

cout << alivenumber << endl;

}

void getInfo()

{

getName();

getHealth();

getPositionX();

getPositionY();

cout <<"生存状态：" << aliveState << endl;

}

bool state = 1;

private:

string name;

int X = rand() % 201;

int Y = rand() % 201;

int health = 1000;

bool aliveState = true;

};

int main()

{

//设定随机数种子

srand((unsigned int)time(NULL));

Spirit array[100];

while (1)

{

cout << "请尊贵的主人输入数字，完成对精灵世界的创造!" << endl;

cout << "1.创造一个可怜的没名字又迷路的小精灵" << endl;

cout << "2.创造一个幸福的有名字又有家的小精灵" << endl;

cout << "3.查看一位小精灵的全部信息" << endl;

cout << "4.重新为一位小精灵找到家" << endl;

cout << "5.哗啦一下子创造十个活蹦乱跳的小精灵并将它们排排序" << endl;

cout << "6.精灵世界遭受邪恶力量的入侵，你创造的小精灵有十个遭了殃" << endl;

cout << "7.向主人汇报精灵世界诞生过多少精灵，又有多少精灵仍然快乐地生活着" << endl;

cout << "8.退出精灵世界" << endl;

cout << "请输入数字：" << endl;

int i = 0;

cin >> i;

switch (i)

{

case 1:

{

cout << "成功！" << endl;

cout << "这是精灵世界存在过的第" << endl;

cout << number << endl;

cout << "个精灵" << endl;

number++;

alivenumber++;

system("pause");

system("cls");

break;

}

case 2:

{

string name;

int X;

int Y;

cin >> name >> X >> Y;

if (X>= 0 && X <= 200 && Y >= 0 && Y <= 200)

{

Spirit s(name, X, Y);

array[number -1] = s;

cout << "成功！" << endl;

cout << "这是精灵世界存在过的第" << endl;

cout << number << endl;

cout << "个精灵" << endl;

number++;

alivenumber++;

system("pause");

system("cls");

}

else

{

cout << "哎呀，输错了，请再输一次！" << endl;

system("pause");

system("cls");

}

break;

}

case 3:

{

cout << "您已创建了" << endl;

cout << number -1 << endl;

cout << "个精灵" << endl;

cout << "请输入要访问的精灵的序号" << endl;

int i;

cin >> i;

array[i - 1].getInfo();

system("pause");

system("cls");

break;

}

case 4:

{

cout << "您已创建了" << endl;

cout << number - 1 << endl;

cout << "个精灵" << endl;

cout << "请输入要访问的精灵的序号" << endl;

int i;

cin >> i;

cout << "请输入要改变的坐标：" << endl;

while (1)

{

int x, y;

cin >> x >> y;

if (x >= 0 && x <= 200 && y >= 0 && y <= 200)

{

array[i-1].setPosition(x, y);

cout << "更改成功！" << endl;

system("pause");

system("cls");

break;

}

else

{

cout << "您输入的位置不符合规范，请重新输入" << endl;

}

}

break;

}

case 5:

{

cout << "创建10个小精灵成功！" << endl;

number = number + 10;

alivenumber = alivenumber + 10;

system("pause");

system("cls");

break;

}

case 6:

{

if (alivenumber < 11)

{

cout << "小精灵数量不足，请重新尝试！" << endl;

}

else

{

for (int k = 0; k < number - 1; k++)

{

array[k].state = 1;

}

for (int i = 0; i < 10; i++)

{

while (1)

{

int j = rand() % (number - 1);

if (array[j].state)

{

array[j].takeDamage(rand() % 1500 + 1);

break;

}

}

}

cout << "邪恶降临啦！小精灵们快跑啊！" << endl;

}

system("pause");

system("cls");

break;

}

case 7:

{

cout << "精灵世界诞生过精灵数量：" << number - 1 << endl;

if (alivenumber > 1)

{

cout << "仍然快乐地生活着的精灵数量：" << alivenumber - 1 << endl;

}

else

{

alivenumber = 1;

cout << "精灵世界生灵涂炭，期待久违的新生！" << endl;

}

system("pause");

system("cls");

break;

}

case 8:

{

cout << "主人再见！" << endl;

system("pause");

return 0;

}

default:

cout << "哎呀，输错了，请再输一次！" << endl;

system("pause");

system("cls");

}

}

}

**2.代码运行示例**

请尊贵的主人输入数字，完成对精灵世界的创造!

1.创造一个可怜的没名字又迷路的小精灵

2.创造一个幸福的有名字又有家的小精灵

3.查看一位小精灵的全部信息

4.重新为一位小精灵找到家

5.哗啦一下子创造十个活蹦乱跳的小精灵并将它们排排序

6.精灵世界遭受邪恶力量的入侵，你创造的小精灵有十个遭了殃

7.向主人汇报精灵世界诞生过多少精灵，又有多少精灵仍然快乐地生活着

8.退出精灵世界

请输入数字：

1. **编程思路分析**

**在初次阅读题目要求后，我首先想到的是设计一个交互界面，通过陈述一个情境的方式，引导用户输入数字进行“精灵类”的调用和互动。**

**首先，我们需要设计一个“精灵”Spirit类的大纲，包括精灵的各种属性：姓名、坐标、生命值、生存状态等。需要注意的是，我并没有完全按照题目要求设立两个静态变量alivenumber和number，而是将这两个值设为全局变量进行处理，并且取消了每次调用构造函数时将alivenumber和number加一的设定，而是在下方while循环中的switch特定case下将两者加1.**

**为什么这样设定呢？其实一开始我是尝试按照题目要求将这两个变量设为类的静态变量的，但是我发现一个问题，那就是如果我们在用户输入“创建一个对象”的指令后再创建一个对象，那我们无法控制每次创建的对象的名称不同，也就是说，即使每次实例化的对象的名称、坐标等都不同，可是实际上只有一个对象在被重复创建，也就无法达到“查找某一个具体的精灵所有信息”和“创建一个含10个精灵的数组”、“实现对精灵的伤害”这样的目标，那从本质上来说。这种类的实例化就只是形式的而非实际的。**

**因此，我产生了“将alivenumber和number定义为全局变量”的想法，这个想法的前提是我们先于用户实例化了若干个（例如100个）的精灵类对象的数组，这样可以在每次输入创建指令后按照操作的先后有序地使用拷贝构造函数的方式将先前使用默认构造函数的对象进行重定义，达到实际意义上的“创建一个不同的精灵类对象”的目的。**

**按照这个思路，进行代码功能的细节化就容易很多了。比如，我在每次用户输入参数前会提示输入的顺序，在成功创建对象以及成功完成对于对象的行为后也会设计提示词，这一方面提高了程序的交互性，也对于代码后期的维护以及场景的渲染起到了很好的提示作用。**

**那么，在大纲确定后的功能实现中，真正具有挑战性，或者说是趣味性的就是“实现对随机十个精灵的随机数值攻击”，我的实现方式是在一个执行10次的for循环中，每一次在已创建的精灵中用种子为系统时间的随机数来选择一位精灵进行范围为0~1500的同样的随机伤害，我选择1500是因为如果最大值低于精灵的最大血量，那就无法出现“一击致命”的场景，不符合“随机灾难性事件”的描述；另一方面，如果最大伤害太高，那就很容易出现“每次消失的精灵个数恒定”的情况，那就实际上不是“意外”而是“屠杀”，也同样降低了游戏的内涵。**

**当然，这种设计思路也出现了另一个“不公平”的情况，那就是有可能一位“倒霉”的精灵在一次灾难中遭受两次甚至多次伤害，虽然理论上由于“随机数”的平均性，这种情况出现的几率很小，但还是在一定程度上背离了题目要求的“对10位精灵的伤害”的设定。可是这又似乎很难解决，我的想法是在每次对一位精灵造成伤害后将其标记，然后将其移出下一次选择的范围。但是如何实现呢？我正在寻找方法。**

**现在，我找到了方法，那就是在“精灵类”对象中附加一个变量state，判断其在一次对10位精灵的伤害中是否已经遭受过一次伤害，如果是的话，就重新进行一次随机选择。当然，在每一次随机伤害之前，我们还要将state全部归为0，否则前一次被伤害的精灵就再也不会受到伤害了。**

1. **实验总结**

**关于类和对象的实际应用，虽然我在之前已经接触过一些，但本次“精灵类”对象的设计却颇具启发性与趣味性。首先这是一个相对独立的程序设计，在用户界面设计、情景设计、逻辑设计等方面均有一定的层次性，其次综合运用了<cstdlib>、<ctime>等库的函数，实现了以系统时间为种子的随机数取样，最后，这也拓展了我的思维，使我综合了解了类的调用，并且意识到类的定义与调用对于程序设计以及运行的重要性。**