**7.1数组和结构体**

**结构体是把多种变量打包成一个“对象”；结构体变量可以放进数组管理；enum 是给状态起名字，typedef 是让你写得更快。**

**结构体数组 = 数据的二维化 + 类型封装，非常强大**

**typedef**

* 用于给已有的类型取别名，本质是“**类型的命名**”。
* 可以让复杂的类型名称简洁易读，比如函数指针、结构体、数组等。

**命名基础型：**

***typedef int MyInt;* MyInt x; 就是 int x;**

**命名数组类型：**

***typedef int MyArray[10];* MyArray a; 就是 int a[10];**

**命名结构体类型：（使用 typedef可以少写 struct，更方便地写）**

***typedef struct StudentInfo Student;***

**命名枚举（使用 typedef可以少写enum，但c++中自带此功能）**

***typedef enum { TITLE，PLAY} SCENE;***

**SCENE currentScene = TITLE 就是enum SCENE currentScene = TITLE;**

**复习. Enum{}**

* **枚举整数常量，本质是“值的命名”。**
* **常用于表示状态（move，attack）、选项（start，end）、场景（title，play）等有限取值的变量。**

**数组（配列Array）**

* **用来集中管理多个类型相同、功能相近的变量。**
* **所有元素共享同一个名字，通过索引（index）区分，是连续内存块。**

***int scores[5] = {90, 85, 88, 92, 87};***

**声明和初始化规则**

1，可初始化要素等于相同的值

*Int enemyX* ***[ 5 ]*** *=* ***{ 0 }***

2，可分别给每个要素初始化值

*Int enemyX* ***[ 5 ]*** *= {* ***10, 10, 10, 10, 10*** *}*

3，不指定要素数的话，由初始化值的数量决定数组长度

*Int enemyX [ ] = {* ***10 , 10 , 10*** *} //****元素数 = 3***

4，初始化值数量少于要素数量：自动填充0

*int bulletX [ 5 ] = { 10, 10 }*

//第0、1个要素为10，第2个及之后的要素自动填充为0*( bulletX [ 2 – 4 ] = 0 )*

5，初始化值数量多于要素数量：错误

int bulletY [ 5 ] = { 10, 10, 10, 10, 10, 10, 10}

**索引(添え字index)**

用于管理数组各要素的编号

*bulletY [ 5 ]、bulletY [ 4 ]……*

*bulletY [ 5 ] = 10* //给5号元素赋值10

注意区分宣言时的*int bulletY [ 5 ]* //声明0号到4号5个要素

**要素数使用常量**

Int enemyMax = 5;

*Int enemy [ enemyMax ];* // enemyMax非常量，错误

常量使用**const int**

**sizeof()**

因为数组的每个元素在内存中所占的大小必定相同

用**sizeof**可获取对象或类型所占内存大小（byte）。

可计算数组的**要素数**

**要素数 = sizeof(数组) / sizeof(数组中元素的类型)**

*int total\_size = sizeof(nums); // 数组总字节数*

*int element\_size = sizeof(nums[0]); // 单个元素的字节数*

*int count = total\_size / element\_size; // 元素个数（要素数）*

如果数组元素个数是固定且明确的，一般不需要用sizeof来计算元素个数。

但是当数组元素很多，或者元素个数是通过常量、宏、函数等间接定义而不明确元素数时，用此公式可以快速准确地得到元素数量。

**配列[]とFOR**

数组通常和for搭配使用

例如复制数组操作

*For ( int i = 要素数 ; i < 要素数 ; i++ )*

*groupB [ i ] = groupA [ i ]；*

（也可使用memcpy()函数实现相同功能，见下）

其他操作同理

Enemy enemies[10];

不使用for循环需10个if

*if (checkCollision(player, enemies[0])) { /\* 处理碰撞 \*/ }*

*if (checkCollision(player, enemies[1])) { /\* 处理碰撞 \*/ }*

*……………… 重复至enemies[9]*

使用for (3行)

*for (int i = 0; i < 5; i++) {*

*if ( checkCollision(player, enemies [ I ] ) )*

*// 统一处理所有敌人的碰撞逻辑*

*handleCollision();*

}

**Memcpy（）**

需要 *#include <string.h>*

实在不清楚拷贝字节数时使用Sizeof

*Memcpy ( prekeys , keys , 256 )*

　目标数列 拷贝数列 拷贝byte数

屏幕上有字

AI 生成的内容可能不正确。

**4. struct{}**

**结构体（struct）就是你设计的“数据打包模板”：**

***typedef struct {***

***char name[32];* //这些都是模板内的member**

***int age;***

***float hp;***

***} Enemy;***

**这就像“敌人信息模板”。**

**用此模板，声明一个结构体变量，就是造一个多种变量打包成的“对象”**

**给这个结构体变量赋值，初始化操作需要注意：**

***Enemy enemy1 = { "Slime", 3, 100.0f };*** //结构体整体初始化：少写了的member会根据自己的类型变为0，NULL，\n等，成员顺序错误会报错

**也可以这样一个个赋值/覆写：**

***enemy1.hp = 80.0f;* //**【.hp】：点运算符成员访问，*= 80.0f*结构体成员赋值

结构体成员是字符串数组（char name[32]）时，**不能这样直接赋值字符串字面量**，需要用 **strcpy()**（需要#include <string.h>）（指针类型char\*除外）

**写法**：*Strcpy ( enemy1.hp , ”小明”)*

**常量数组可以作为结构体的member，结构体变量也可以组成数组**

**(结构体数组 = 数据的二维化 + 类型封装，非常强大。)**

***Enemy enemies[3] = {***

***{ "Slime", 3, 100.0f },*** //结构体整体初始化

***{ "Goblin", 5, 120.0f },***

***{ "Dragon", 10, 300.0f }***

***};***

***enemies[0] = (Enemy){ "Slime", 3, 100.0f };*** //数组元素单独赋值

***Enemy enemy[3];***

***for (int i = 0; i < 3; i++) {*** //for批量操作

***enemies[i].hp = 100.0f***

***}***

***结构体也可包含结构体***

*struct Vector2 {*

*float x;*

*float y;*

*};*

*struct Player {*

*Vector2 position;*

*Vector2 speed;*

*int isAlive;*

*};*

**不能直接访问成 player.x，而是必须明确指出它属于Vector2：**

**Player player1;**

**player1.Vector2.x = 100;**

**player1.Vector2.y = 100;**

**附录：结构体使用注意点：**

**名称与成员不匹配 / 成员名重复结构体名 / 数组化每个字段“结构体拆散”**

**图形用户界面, 文本

AI 生成的内容可能不正确。**

**文本

AI 生成的内容可能不正确。**

**图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。**