第一讲 函数与指针 & 结构体 & 二维数组

■ 指针与引用

```C++
int a = 10;
int *p = &a;    // 指针保存变量地址
int &r = a;     // 引用是变量别名
```

常见错误

```C++
//    错误：a 不是地址（若 a 为普通变量）
int *p = a;
//    正确：&a 取变量 a 的地址赋值给指针 p
int *p = &a;
```

传参对比

方式	定义	影响
值传递	形参与实参互不影响	不修改原变量
引用传递	形参是实参别名	修改原变量
指针传递	传地址，通过解引用修改	修改原变量

■ 变量作用域

```C++
int x = 5;          // 全局变量：作用域覆盖整个程序
void foo() {
  int x = 10;       // 局部变量：作用域仅在 foo() 内部，覆盖同名全局变量
  cout << x;        // 输出结果：10（局部变量优先）
}
```

核心规则：局部变量优先级高于外层同名变量，函数内部声明会覆盖全局 / 外层变量。

静态变量 static：在函数内声明时，生命周期延长至整个程序（仅初始化一次，函数调用

后值不销毁）。

■ 函数与模块化编程

函数的核心流程：声明 + 定义 + 调用

```C++
// 函数声明：可指定默认参数（默认参数仅能在声明处定义一次）
int add(int a, int b = 1);

// 函数定义：实现具体逻辑（不可重复定义默认参数）
int add(int a, int b) {
  return a + b;
}

int main() {
  cout << add(2);    // 调用：使用默认参数 b=1，输出结果：3
  cout << add(2,3); // 调用：显式传参 b=3，输出结果：5
  return 0;
}
```

■ 数组与多维数组

核心特性：数组名是「常量指针」（指向数组首元素地址，不可修改）

```C++
int arr[3] = {1,2,3};
cout << *(arr + 1); // 等价于 arr[1]，输出结果：2

// 注意：sizeof(arr) 返回整个数组的字节数，而非指针长度
cout << sizeof(arr);  // 输出：12（int 占 4 字节，3 个元素：4*3=12）
```

二维数组指针访问

```C++
int arr[2][3] = {{1,2,3},{4,5,6}};
cout << *(*(arr+1)+2); // 6
```

结构体与复合数据类型

```C++
struct Point {
    int x, y;
};
Point p = {1,2}; //    正确初始化
```

结构体可嵌套或传引用修改成员变量

```C++
void birthday(Cat &c){
    c.age++;
}
```