C++中除了简单数据类型，还有什么数据类型

C++中的简单数据类型（内置基础类型）主要包括`int`、`float`、`char`、`bool`等，仅能满足基础数据存储需求。除此之外，C++提供了复合类型、自定义类型及标准库封装类型，这些类型功能更强大，是实现复杂逻辑的核心，下面分大类详细说明。

一、复合类型（基于基本类型组合）

复合类型由基本类型“组装”而成，复用基础类型特性，适配多数据、间接操作等场景。

1. 数组（Array）

用于存储多个相同类型元素，内存连续排列，大小固定，可通过下标直接访问。

```
int scores[5] = {90, 85, 95, 88, 92}; // 整数数组
char name[10] = "Tom"; // C风格字符串（字符数组）
```

2. 指针（Pointer）

存储变量的内存地址，支持间接操作变量、动态内存管理，是C++核心特性之一。

```
int num = 10;
int* p = &num; // 指针p存储num的地址
*p = 20; // 通过指针修改num的值（num变为20）
```

3. 引用（Reference）

变量的“别名”，本质简化指针使用，必须初始化且无法更换指向的变量，常用作函数参数（避免拷贝）或返回值。

```
int a = 5;
int& ref = a; // ref是a的别名
ref = 10; // 操作ref等价于操作a，a变为10
```

4. 结构体（Struct）

将不同类型数据组合为一个整体，适合描述实体（如学生、商品），C++中支持定义成员函数。

```
struct Student {
    std::string name; // 姓名
    int id; // 学号
    float score; // 成绩
};
Student s1 = {"张三", 1001, 90.5}; // 定义并初始化结构体变量
```

5. 共用体（Union）

与结构体类似，但所有成员共享同一块内存，同一时间仅能使用一个成员，可节省内存，大小为最大成员的大小。

```
union Data {
    int i; // 整数
    float f; // 浮点数
    char c; // 字符
};
Data d;
d.i = 10;
d.f = 3.14; // 覆盖i的值，共享4字节内存
```

二、自定义类型（开发者自主定义）

根据业务需求创建，灵活性极高，是面向对象编程和规范化开发的关键。

1. 枚举（Enum/Enum Class）

定义一组命名常量，替代“魔法数字”提升代码可读性，分为普通枚举和C++11新增的强类型枚举（更安全，避免命名冲突）。

```
// 普通枚举，默认值从0开始递增
enum Weekday { Mon, Tue, Wed };
Weekday day = Tue; // day的值为1

// 强类型枚举，需加作用域访问
enum class Color { Red, Green, Blue };
Color c = Color::Red;
```

2. 类（Class）

面向对象编程的核心，封装数据（成员变量）和操作数据的方法（成员函数），支持封装、继承、多态三大特性。

```
class Circle {
private:
    double radius; // 私有成员（半径，仅类内可访问）
public:
    Circle(double r) : radius(r) {} // 构造函数（初始化半径）
    double getArea() { return 3.14 * radius * radius; } // 计算面积
};
Circle c1(5.0);
double area = c1.getArea(); // 调用成员函数，面积为78.5
```

三、标准库封装类型（现成可用的高级类型）

C++标准库（STL）封装了大量实用类型，无需重复开发，兼顾安全性和效率。

1. 字符串（std::string）

替代C风格字符数组，支持拼接、查找、替换等操作，更安全易用，需包含头文件`<string>`。

```
#include <string>
std::string str = "Hello";
str += " World"; // 拼接字符串，结果为"Hello World"
int len = str.length(); // 获取长度，结果为11
```

2. STL容器

用于存储一组元素，支持动态扩容、增删改查，适配不同场景，需包含对应头文件。

```
#include <vector> // 动态数组（最常用）
#include <map>    // 键值对集合

std::vector<int> vec = {1,2,3};
vec.push_back(4); // 新增元素，vec变为{1,2,3,4}

std::map<std::string, int> person;
person["张三"] = 20; // 存储姓名-年龄键值对
```

3. 其他常用标准库类型

- **std::pair**：存储一对不同类型的值，如`std::pair<int, std::string>`（学号-姓名）；
- **std::tuple**：存储多个不同类型的值，比`pair`更通用；

- **函数指针/仿函数**：函数指针指向函数地址，仿函数是重载`()`的类，常用于算法回调。

总结

C++中除简单数据类型外，核心类型可归纳为三类：复合类型（基于基本类型组合）、自定义类型（开发者自主创建）、标准库封装类型（现成高效）。其中类、指针、STL容器是实际开发中最常用的核心，掌握这些类型是编写高效C++代码的基础。

（注：文档部分内容可能由 AI 生成）