**算术类型**

bool, char, wchar\_t, char16\_t, char32\_t, short, int, long, long long, float, double, long double

类型前加 unsigned 为无符合类型

**指定字面值的类型**

L’a’ // 宽字符

u8”hi” // utf-8字符串

**变量定义与声明**

// 声明：使得该名字为程序所知，要想使用别处定义的名字，只需引入该声明即可

// 定义：负责创建与名字关联的实体

extern int a; // 声明变量a

int b; // 声明并定义变量b

**引用**

// 为对象起了另一个名字，可通过该名字操作对象

int ival = 1024;

int &ival1 = ival;

**指针**

// 存放某个对象地址的变量

// 注意：如下使用变量的 & 与定义变量的 & 没有任何关系，同理 \*

int ival = 1;

int \*p = &ival;

std::cout<< \*p << std::endl; // 输出1

**void\*指针**

// void\* 可存放任意对象的地址

int ival = 1;

void \*p = &ival;

int \*z = &ival;

bool isSame = z == p; // 比较指针

std::cout<< \*p << std::endl; // 错误，不能使用void\*指针

**常量引用**

// 常量引用变量，能通过relival读取ival的值，但不能通过relival改变ival的值

int ival = 1;

const int &relival = ival;

const int &relival1 = ival + 1;

int &relival2 = ival + 1; // 错误，非常量引用不能为表达式

**指向常量的指针**

// 指向常量的指针，不能通过指针修改对象

int ival = 1;

const int \*p = &ival;

\*p = 1; // 错误

**常量指针**

// 常量指针，指针的地址不能改变

int ival = 1;

int a = 1;

int \*const p = &ival;

p = &a; // 错误

**类型别名**

// 类型别名

typedef double d;

using i = int;

**auto类型**

int ival = 1;

auto a = ival; // a为int

**decltype类型**

int ival = 1;

decltype(ival) a = 1; // a为int

**static类型**

static声明的变量只能在当前文件引用

static int a = 1;

**自定义数据结构**

// 定义 StructA

struct StructA

{

int Felid1;

};

// 定义变量

StructA StructA1;

StructA1.Felid1 = 1;

std::cout << StructA1.Felid1 << std::endl;

**编写自己的头文件**

一般每个结构或类都定义在头文件中，且文件名与类名相同，一般定义如下

#ifndef STRUCTA\_H

#define STRUCTA\_H

/\* StructA.cpp文件 \*/

struct StructA

{

int Felid1;

};

#endif