1. 基本数据类型：

|  |
| --- |
| void, bool, char, short  Int, float, double, long |

一个char的空间应确保可以存放机器基本字符集中任意字符对应的数字值，也就是说，一个char的大小和一个机器字节一样。

类型char的表现形式有可能是unsinged char，也有可能是signed char，具体是哪种由编译器决定。

1. 变量定义

**基本数据类型** (一组)**类型声明符**

面对一条比较复杂的指针或引用的声明语句时，从右向左阅读有助于弄清楚它的真实含义。

1. 变量初始化

定义于任何函数体之外的内置类型变量被初始化为0；函数体内的内置类型变量如果没有初始化，则其值未定义。

类对象如果没有显示地初始化，则其值由类确定。

建议初始化每一个内置类型的变量。

1. Void\* 指针

Void\* 指针能做的事儿比较有限：拿它和别的指针比较，作为函数的输入或者输出，赋值给另外一个void\* 指针。

以void\* 的视角来看内存空间就仅仅是内存空间，无法访问内存空间中所存的对象。

1. 常量指针、常量引用

所谓指向常量的指针或引用，不过是指针或引用“自以为是”罢了，它们觉得自己指向了常量，所以自觉地不去改变指向对象的值。

1. 类型别名

有两种方法用于定义类型别名：

1. 传统方法，使用关键字**typedef**:

typedef double wages: //wage是double的同义词

1. C++11的新方法，使用别名声明(alias declaration)来定义：

using wages = double;//wage是double的同义词

如果某个类型别名指代的是复合类型或者常量，则把它用到声明语句中就会产生意想不到的后果：

typedef char \*pstring;

const pstring cstr=0; //**等价于char \* const cstr=0;不是const char \*cstr=0;**

1. decltype类型指示符

希望从表达式的类型推断出要定义的变量的类型，但不想用该表达式的值初始化变量。C++11引入decltype，它的作用是选择并返回操作数的数据类型。编译器分析表达式并得到它的类型，却不实际计算表达式的值。

切记：decltype((variable))（注意，是双层括号）的结构永远是引用，而decltype(variable)结果只有当variable本身就是一个引用时才是引用。

1. 自定义数据结构struct

struct Sales\_data{}

string bookNo;

unsigned units\_sold = 0;

double revenue = 0.0;  
}；

C++11新标准规定，可以为数据成员提供一个类内初始化值(in-class initializer)，创建对象时，类内初始化值将用于初始化数据成员，没有初始化值得成员将被默认初始化。