变量作用：给一段指定的内存空间起名，方便操作这段内存。

常量：用于记录程序中不可更改的数据

#define宏常量 #define 常量名 常量值 通常在文件上方定义

const修饰的变量

常量和宏常量不可修改

再给变量或者常量起名的时候不能使用C++关键字，否则会产生歧义

标识符命名规则：1、标识符不可以是关键字

2、标识符是由字母、数字、下划线构成

3、标识符是区分大小写的

4、标识符第一个字符只能是字母或下划线

数据类型存在意义：给变量分配合适的内存空间。

Short:2字节（-2^15 ~2^15 - 1）

int：4字节

long：win—>4字节

long long：8字节

整形：整型变量表示的是整数类型的数据

上面几个的区别是所占内存空间不同

基本情况下用int就足够了

Sizeof关键字：求出数据类型占用内存大小

语法：sizeof(数据类型/变量)

浮点型：用于表示小数

单精度float 4字节 7为有效数字

双精度double 8字节 15~16位有效数字

默认情况下，输出一个小数，会显示出6为有效数字

科学计数法

3e2 代表 3 \* 10^2

3e-2 代表 3 \* 0.1 ^2

字符型：字符型变量用于显示单个字符

语法：char ch = ‘a’;

注意：在显示字符型变量时，用单引号将字符括起来，不要用双引号

单引号内只能有一个字符，不可以是字符串

C/C++中字符型变量只占用1个字节

字符型变量并不是把字符本身存放到内存中存储，而是将对应的ASCLL编码放入到存储单元

转义字符：用于表示一些不能显示出来的ASCLL字符

换行符 \n

反斜杠 \\

水平制表符 \t 一个\t占8个位置，作用是整齐的输出数据

字符串型：用于表示一串字符

C语言 char 变量名[] = “字符串值” 一定要加[]，一定是双引号包含字符串

C++风格 string 变量名 = “字符串值” 用之前记得加头文件#include<string>

布尔类型bool

True真——1 false假——0

除了非0其它都是真

Bool类型占1字节大小

数据输入：用于从键盘获取数据 cin >> 变量

只要是非0的数都为真。

初始化指创建变量并给它赋初始值，而赋值则是擦除对象的当前值并用新值代替。

三目运算符

语法：表达式1 ？ 表达式2 ： 表达式3

解释：

如果表达式1的值为真，执行表达式2，并返回表达式2的结果

如果表达式1的值为假，执行表达式3，并返回表达式3的结果

在C++中三目运算符返回的是变量，可以继续赋值

Switch缺点：，判断时候只能是整形或者字符型，不可以是一个区间

Switch优点：结构清晰，执行效率高

总结：与if语句对比，对于多条件判断时，switch的结构清晰，执行效率高，缺点是switch不可以判断区间。

While循环语句：只要循环条件的结果为真，就执行循环语句。

变量的定义用于为变量分配存储空间，也可以为变量指定初始值。

#include<ctime>

//添加随机数种子，作用利用当前系统时间生成随机数，防止每次随机数都一样

srand((unsigned int)time(NULL));

do…while循环语句:与while的区别在于do…while会先执行一次循环语句，在判断循环条件。

For循环的执行

Break的使用时机 1、出现在switch语句中 2、出现在循环语句中 3、出现在嵌套循环中

Continue语句：在循环语句中，跳过本次循环中余下尚未执行的语句（不执行continue下面的代码），继续执行下一次循环

Break直接退出本循环

数组：一个集合，里面存放相同数据类型的数据元素

特点1、数组中的每个数据元素都是相同的数据类型

数组是由连续的内存位置组成的

如果不在定义的时候不加入值也不加入数组的长度，这个是错误的，会报错的。所以定义数组的时候，必须有初始长度。

数组的定义方式：

1、数据类型 数组名[数组长度];

2、数据类型 数组名[数组长度] = {值1， 值2，…};

3、数据类型 数组名[] = {值1， 值2 …}

一维数组名称的用途：

可以统计整个数组在内存中的长度 sizeof(arr)

可以获取数组在内存中的首地址 cout << arr <<endl;

数组起始下标位置：int start = 0;

数组末尾下标位置：int end = sizeof(arr)/sizeof(arr[0]) – 1;

冒泡排序：

比较相邻的元素，如果第一个比第二个大，就交换他们两个。

对每一对相邻元素做同样的工作，执行完毕后，找到第一个最大值

重复以上的步骤，每次比较次数-1，直到不需要比较。

每次都是找到它的最大值！

总共排序轮数是元素个数 – 1

//内层循环对比次数 = 元素个数 - 当前轮数 - 1

//如果第一个数字，比第二个数字大，交换两个数字

在定义二维数组时，如果初始化了数据，可以省略行数。

函数：将一段经常使用的代码封装起来，减少重复代码

函数的定义的步骤：

返回值类型

函数名

参数列表

函数体语句

return表达式

语法

返回值类型 函数名 参数列表

{

函数体语句

return 表达式

}

值传递：就是函数调用时实参将数值传入给形参。值传递时，如果形参发生改变，并不会影响实参。

常见的函数样式4种

无参无返

有参无返

无参有返

有参有返

函数声明：告诉编译器函数名称及如何调用函数，函数的实际主体可以单独定义。

函数的声明（告诉编译器存在而已）可以多次，但是函数的定义只能一次。

函数的分文件编写作用：让代码结构更加清晰

函数分文件编写一般有4个步骤

创建后缀名为.h的头文件

创建后缀名为.cpp的源文件

在头文件中写函数的声明

在源文件中写函数的定义

指针的作用：可以通过指针间接访问内存。

内存编号是从0开始记录的，一般用十六进制数字表示

可以利用指针变量保存地址

在32为操作系统下：指针占用4个字节空间

64 8

空指针：指针变量指向内存中编号为0的空间

用途：初始化指针变量

注意：空指针指向的内存是不可以访问的

野指针：指针变量指向非法的内存空间

空指针和野指针都不是我们申请的空间，因此不要访问

const修饰指针有三种情况

1、const修饰指针 --常量指针

2、const修饰常量 --指针常量

3、const即修饰指针，又修饰常量

常量指针

const int \* p = &a

特点：指针的指向可以修改，但是指针指向的值不可以改

指针常量

int \* const p = &a;

特点：指针的指向不可以修改，指针指向的值可以改

const既修饰指针又修饰常量

const int \* const p = &a;

特点：指针的指向和指针指向的值都不可以改

指针和函数作用：利用指针做函数参数，可以修改实参的值

结构体属于用户自定义的数据类型，允许用户存储不同的数据类型

结构体的定义和使用

语法：struct 结构体名{结构体成员列表}：

通过结构体创建变量的方式有三种：

Struct结构体名 变量名

Struct结构体名 变量名 = {成员1值， 成员2值…}

定义结构体时顺便创建变量

定义结构体时关键字是struct，不可省略

创建结构体变量时，关键字struct可以省略

结构体变量利用操作符“.“访问成员

结构体数组作用：将自定义的结构体放入到数组中方便维护

语法：struct 结构体名 数组名[元素个数] = { {}, {}, {}…}

结构体指针作用：通过指针访问结构体中的成员

利用操作符->可以通过结构体指针访问结构体属性

结构体嵌套结构体作用：结构体中的成员可以是另一个结构体

结构体做函数参数作用：将结构体作为参数向函数中传递

传递方式有两种：值传递、地址传递

值传递修饰形参，但是实参不会改变

地址传递修饰形参，实参会跟着发生改变

所以如果不想修改主函数中的数据，用值传递，反之用地址传递。

结构体中const使用场景作用：用const来防止误操作

//随机数种子 记得加上头文件#include<ctime>

srand((unsigned int)time(NULL));

C++程序在执行时，将内存大方向划分为4个区域

代码区：存放函数体的二进制代码，由操作系统进行管理

全局区：存放全局变量和静态变量以及常量

栈区：由编译器自动分配释放，存放函数的参数值，局部变量等

堆区：由程序员分配和释放，若程序员不释放，程序结束后由操作系统回收

内存四区意义：

不同区域存放的数据，赋予不同的生命周期，给我们更大的灵活编程。

栈区数据注意事项——不要返回局部变量的地址

栈区的数据由编译器管理开辟和释放

栈区的数据在函数执行完后自动释放

堆区：由程序员分配释放，若程序员不释放，程序结束时由操作系统回收

在C++中主要利用new在堆区开辟内存

new返回的是该数据类型的指针

利用new创建的数据，会返回该数据对应的类型的指针

堆区的数据 由程序员管理开辟，程序员管理释放

如果想释放堆区的数据，利用关键字delete

释放数组的时候 要加上[ ]才可以。

引用作用：给变量起别名

语法： 数据类型 & 别名 = 原名

注意事项：引用必须要初始化

引用一旦初始化后，就不可以更改了

引用左函数参数作用：函数传参时，可以利用引用的技术让形参修饰实参

只需要在函数定义的时候对形参变量在前面加上& 符号。

优点：可以简化指针修改实参

通过引用参数产生的效果同按地址传递是一样的。

引用做函数的返回值

注意：不要返回局部变量的引用 输出第一次的结果正确，是因为编译器做了保留，第二次结果错误，因为它的内存已经释放。

静态变量是存放在全局区的 全局区上的数据在程序结束后系统释放

如果函数的返回值是引用，这个函数调用可以作为左值

引用的本质在C++内部实现是一个指针常量

常量引用主要是用来修饰形参，防止误操作

在函数形参列表中，可以加const修饰形参，防止形参改变实参

引用必须引一块合法的内存空间

int &a = 10; 这句话是个错误，引用本身需要一个合法的内存空间

但是加入const就可以了，编译器优化代码，int temp = 10; const int & a = temp;

const int &a = 10; 这句话就可以运行

函数默认参数

在C++中，函数的形参列表中的形参是可以有默认值的。

语法：返回值类型 函数名 （参数 = 默认值）{ }

注意事项：如果某个位置已经有了默认参数，那么从这个位置往后，从左到右都必须有默认值。

例如 int func(a = 10 ,int b, int c) 这是错误的，因为a有了默认值，b和c都在a的右边所以b和c都必须要有默认值。

int func(int a, int b, int c = 10) 这样是可以的。

如果函数声明有默认参数，函数实现就不能有默认参数。

C++中函数的形参列表里可以有占位参数，用来做占位，调用函数时必须填补该位置。

语法：返回值类型 函数名 （数据类型）{ }

函数重载作用：函数名可以相同，提高复用性

函数重载满足条件：

同一个作用域下

函数名称相同

函数参数类型不同，或者个数不同或者顺序不同

注意：函数的返回值不可以作为函数重载的条件

注意事项：函数的返回值不可以作为函数重载的条件。

函数重载的注意事项：引用作为重载条件

函数重载碰到函数默认参数 碰到默认参数产生歧义，需要避免

类和对象

C++面向对象的三大特性：封装、继承、多态

封装的意义一：

将属性和行为作为一个整体，表现生活中的事物

将属性和行为加以权限控制

Class代表设计一个类，类后面紧跟着类的名字

实例化：通过类创建具体的类的对象。

语法：class 类名 { 访问权限： 属性 / 行为 };

类中的属性和行为 统称为成员

属性——成员属性 成员变量

行为——成员函数 成员方法

封装的意义二：

类在设计时，可以把属性和行为放在不同的权限下，加以控制

访问权限有三种：

1、public 公共权限 类内可以访问 类外可以访问

2、protected 保护权限 类内可以访问 类外不可以访问 子类可以访问父类的保护内容

3、private 私有权限 类内可以访问 类外不可以访问 子类不可以访问父类的私有内容

在C++中struct和class唯一的区别在于默认的访问权限不同

区别：

struct 默认权限为公共 即类外可以访问

class 默认权限为私有 即类外不能访问

成员属性设置为私有优点

将所有成员属性设置为私有，可以自己控制读写权限

对于写权限，我们可以检测数据的有效性

.h文件中就留下类的声明

.cpp文件中是定义记得加上成员函数，例如下图中的Circle::

构造函数和析构函数

如果我们不提供构造和析构，编译器会提供，但是编译器提供的构造函数和析构函数时空实现。

构造函数：主要作用在于创建对象时为对象的成员属性赋值，构造函数由编译器自动调用，无须手动调用。

析构函数：主要作用于对象销毁前系统自动调用，执行一些清理工作。

构造函数语法：类名（）{ }

构造函数，没有返回值也不写void

函数名称与类名相同

构造函数可以有参数，因此可以发生重载

程序在调用对象时候会自动调用构造，无需手动调用，而且只会调用一次

析构函数语法：~类名（）{ }

析构函数，没有返回值也不写void

函数名称与类名相同，在名称前加上符号~

析构函数不可以有参数，因此不可以发生重载

程序在对象销毁前会自动调用析构，无需手动调用，而且只会调用一次。

构造函数的分类及调用

两种分类方式：

按参数分为：有参构造和无参构造

按类型分为：普通构造和拷贝构造

三种调用方式：

括号法 显示法 隐式转换法

注意事项：调用默认构造函数的时候，不要加（）。因为如果加上了（），编译器会认为是一个函数的声明。

拷贝构造写法：和自己一样的数据类型，用引用方式传递，前面加上const，例如

Person(const Person& p)

拷贝构造函数调用时机

C++中拷贝构造函数调用时机通常有三种情况

使用一个已经创建完毕的对象来初始化一个新对象

值传递的方式给函数参数传值

以值方式返回局部对象

构造函数调用规格

默认情况下，c++编译器至少给一个类添加三个函数

1、默认构造函数（无参，函数体为空）

2、默认析构函数（无参，函数体为空）

3、默认拷贝构造函数，对属性进行值拷贝

构造函数调用规则如下

如果用户定义有参构造函数，c++不再提供默认无参构造，但是会提供拷贝构造

如果用户定义拷贝构造函数，c++不会再提供其他构造函数

深拷贝与浅拷贝

浅拷贝：简单的赋值拷贝操作

深拷贝：在堆区重新申请空间，进行拷贝操作

浅拷贝带来的问题就是堆区的内存重复释放，浅拷贝的问题，要利用深拷贝进行解决。

如果属性有在堆区开辟的，一定要自己提供拷贝构造函数，防止浅拷贝带来的问题。

初始化列表

作用：C++提供了初始化列表语法，用来初始化属性

语法：构造函数（）：属性1（值1），属性2（值2）…{ }

类对象作为类成员

C++类中的成员可以是另一个类的对象，我们称该成员为对象成员。

当其他类对象作为本类成员构造时候先构造类对象，再构造自身，析构的顺序与构造相反。

静态成员就是在成员变量和成员函数前加上关键字static，成为静态成员

静态成员分为：

静态成员变量

所有对象共享同一份数据

在编译阶段分配内存

类内声明，类外初始化 类外初始化格式 数据类型 类名：：变量名 = 数据

静态成员函数

所有对象共享同一个函数

静态成员函数只能访问静态成员变量

静态成员变量有两种访问方式：

通过对象进行访问

通过类名进行访问 格式：类名：：变量名

静态成员变量也是有访问权限的，公有的情况下，类外是能够访问的，私有的权限下，类外是不能够访问的。

静态成员函数也是有访问权限的同静态成员。

C++中，类内的成员变量和成员函数分开存储

只有非静态成员变量才属于类的对象上

空对象占用内存空间为1，C++编译器会给每个空对象也分配一个字节空间，是为了区分空对象占内存的位置，每个空对象也应该有一个独一无二的内存地址

每一个非静态成员函数只会诞生一份函数实例，也就是说多个同类型的对象会共用一块代码

问题：这一块代码如何区分哪个对象调用自己？

C++通过提供特殊的对象指针，this指针，解决问题

this指针指向被调用的成员函数所属的对象

this指针是隐含每一个非静态成员函数内的一种指针

this指针不需要顶底，直接可使用

this指针的用途：

当形参和成员变量同名时，可用this指针来区分

在类的非静态成员函数中返回对象本身，可以使用return \*this

这个最前面的带上引用和不带有引用是有区别的。如果用值的方式返回，它返回的一定是创建一个新的对象，如果是用引用的方式返回，它不会创建新的对象。

当形参和成员变量同名时，可用this指针来区分。