最后一次课考试 14周

第一次课

计算机五个部分

输入、输出、运算器、控制器、存储器

机器语言执行速度最快

技术路线：

L A M P 开源

W IIS IIS 服务器

.NET

平台（platform）

二进制文件.exe可以在操作系统上直接运行

Unmanaged Applications非托管代码

而c#是Managed Applications托管代码，在虚拟机上运行，生成的.exe文件是个文本文件（类似于Java的字节码文件）

CLR Common Language Runtime相当于JAVA的虚拟机（JVM）

CLS Common Language Specification 语言规范

命名空间——等同于包

XML 可扩展的标记语言

编译语言 写完再编译

解释语言 写一句一编译，效率较低，但是出错可以迅速发现，后端往往是解释的

manifest程序清单

metadata元数据（描述数据的数据）

.exe文件要运行需要用加载器

assembly 程序集

JIT语言

源文件后缀名 .cs

visual stdio

进程是资源分配的最小单位

第二次课

编译器会以main函数所在的类名命名生成的可执行文件

dll文件，动态链接库，在编译自己的文件时当需要引用时编译进来，可执行文件需要dll对应的类时才去编译

四种符号：

标识符

关键字

界符

常量

标识符

正则表达式 小括号（组合），中括号[可有可无]，大括号{重复}

只能以字母/下划线（\_）/@符号开头，后边跟若干字母或数字或下划线（\_）

关键字是标识符的子集

使用Unicode码，变量起名可以以中文起名，不推荐

word中按Alt+X可以实现Unicode码和中文或者其他语言的转换

第三次课

.cs源程序：

using ....

namespace \*\*\*\*\*{

class{}

Enum{}

struct{}...

}

类名前写/// 自动生成summary（注释，方便写文档），写文档时可以用命令自动生成xml文档

第四次课

string类型

是引用类型

比较可以用"=="比较是否相等

vs中在string类型处右键点开可以跳转

动态数组：

sort 按照英文字母的顺序排

关联数组（哈希表）：

方括号内套双引号——关键字（键），不允许重复

关联数组名+关键字 赋值，形成键值对结构

foreach（\*\*\* x in \*\*\*(数组名)） 循环

NoSQL数据库——非关系型数据库，包含关系型数据库，也包含其他的非关系型数据库，采用哈希表的键值对形式存储

结构体

值类型，在栈里存储，调用速度快，等同于栈上的类

（与之相对，类存在堆里，是引用类型）

不允许声明没有参数的构造函数，因为没有参数的构造函数是默认自带的，不能够覆盖，调用时对所有的数据成员赋初值

（对类来讲，声明的构造函数可以有参数，可以没有参数）

类的空间分配和构造过程

声明 Rectangle r;

r = new Rectangle();

第一条，在栈顶分配一个字节左右的小空间，里边是空的

第二条，在堆的顶部分配一个空间，并把首地址填到第一步执行时的空间中

参照c#笔记图，在stack和heap增长到接触时，会向操作系统请求分配更多的空间

结构体会把以上过程全都在堆中进行

结构体不能继承类，可以继承接口

不提倡写析构函数

Object对象

Object类是所有类的基类，既可以指向值类型，也可以指向引用类型

装箱（Boxing）和拆箱(Unboxing)

boxing

把一个栈上的值类型拷贝一个放入堆中，把首地址交给object

例如：值类型转换成引用类型

unboxing

强制转换，把堆里的东西拷贝进栈中

例如：引用类型强制转换为值类型

队列

一个队列中如果包含多种数据类型，出队时需要强制类型转换

表达式

expression

operand操作数

运算时两个变量的长度不一时，运算完成后得到长度较大的类型，如果此类型比int占用的字节数少，那么保存为int

is操作符，判断一个变量是否属于某个类型，a is object

在一个数据类型的最大值时+1会变成更大的数据类型

左右移位

checked运算符，可以把后边括起来的地方检查是否有溢出，有溢出会报错（抛出异常）

typeof 类型

sizeof，判断一个类型占用的字节多少，慎用，检查程序是否正确时使用，用unsafe+csc的编译命令来操作，不太安全

声明

declaration

某一个局部变量的生命周期

在声明语句处开始生命周期，在语句块完成时生命周期结束

内层和外层出现同名的变量，内层的会把外层的覆盖

internal修饰符，程序集内可见

两个程序拥有同名的namespace，两个程序可以合并，同名的namespace会合并，namespace相当于容器，把这些容器中的东西纳入考虑范围，即使用namespace中的A时，可以直接调用，而使用不在namespace中的B时，要写\*\*\*.B

namespace的子空间必须写明，Util和Util.Figures不能通过一个using Util全包括了，不能够访问子名字空间

语句块

第六周

语句

赋值语句 左值提供地址，右值提供值

string Split分割，split（','）以,分割

string String.Join("", 分割的string数组)连接

switch语句

判断条件不能是浮点数，比较采用的是减法，浮点数相减可能有精度问题

有限自动机 匹配上的是可以识别一门语言的所有语句，可以用来判断语句的合法性，符合的语言是正则语言

循环

foreach 只能读不能写，对集合进行遍历

跳转

goto语句 在switch语句块中可以直接跳转到某个case，可能会破坏程序的结构化特点，不能从循环体外跳进循环里（不能跳进块里）或者try/catch的finally块

return 语句

函数和过程：函数有返回值，过程没有返回值，返回void的函数可看做过程，return可以直接跳出函数/过程

栈里有局部变量（活动记录区）， 堆里有全局变量

Main函数里的返回值返回给了操作系统（与java不同）

控制台交互

输出格式 占位符 0开始递增表示第几个变量

中括号表示可选的

"{" n ["," width] [":" format [precision]] "}"

width宽度，format格式，precision保留的小数位数（精度）

format

d 整数

f 浮点数（默认2位）

n 逗号（自动在每三位加逗号，千，百万，十亿）

e 科学计数法

c 货币

x 16进制

g 默认

FileStream 数据流，输出到文件，建立连接

StreamWriter 写到文件

先建立连接，再写/读

数据流用完要关闭，否则一直占有资源

同一时刻只能由一个StreamWriter

键盘输入 ReadLine可以读入整个一行的内容（读到第一个回车/换行），Read会读到第一个空格

文件读入如果用Read()，会读到EOF（end of file）文件的最后会有EOF

读的写法把StreamWriter 换成StreamReader

在控制台运行程序，文件名 \*&@&^%(@

后边的东西是参数，就是static void Main(string[] args)的args，按照空格拆分，每遇到一个空格认为是一个参数

类和结构体

类 默认private，引用结构，分配在堆上，指针放在栈上

结构体 分配在栈上，效率高，不需要垃圾回收

第七周

类和结构体

类 字段和方法组成

结构体和类的最主要区别 不能够定义无参数的构造函数

对类：

在namespace中类不写可见性public,private，默认的可见性在namespace里，在其他namespace里使用写声明类的namespace名.类名 或者 using声明类的namespace名

const关键字，定义常量，无法改变，在声明的时候赋值

常量不属于对象，属于类

readonly关键字，只读，在构造函数赋值或声明时赋值，之后不可修改，属于对象

静态static，属于类而不属于对象，类名加点调用

常量默认是static，不能再加

mean均值

调用者 caller callee（被调用者）

传值

传地址（引用调用） ref关键字（被调用者和调用者都要写），表示引用，传进去的是地址，引用调用

out传递 单向传递，只能由被调用者传向调用者

传入的形参可以用params 关键字实现动态变化（用数组，放在形参最后）

例如： void add(out int sum, params int[] val){}数组长度动态变化

overload 重载，同名函数参数不同，不能根据返回值判断

override 重写

低精度到高精度可以匹配，如果有多个可以匹配的重载函数可以调用时（最接近的不止一个函数时），编译器会报错

构造函数，给字段赋初值

冒号:继承

缺省构造函数

给所有字段全部赋初值

有了有参数的构造函数，编译器就不会再生成无参数的构造函数

结构体的字段不能在声明时赋初值

构造函数里必须把所有字段全部赋初值

静态成员需要静态构造函数，在第一次声明对象时调用静态构造函数

析构函数

在c#里慎用，把对象杀死，如果对象已经被杀死可能会再生成一个对象再杀死

获取private字段 属性，类似get set函数

索引器 Indexers，用index把复合的变成数组

public int this[int i]{...}，函数体内是get set

运算符重载

必须是static, public static \*\*\* operator +(A a, B b){}

重载了+也要重载-

重载要成对出现，返回的是个函数，比如一系列操作+-\*/

转换操作符

长的赋给短的要强制类型转换，长的转换成短的

代码溢出检查 check关键字

继承

class B : A //B继承A，除了A中的private内容和A的构造函数，其他所有的函数和变量B中都有

Override重写 子类中和父类完全同名

Overlap 交叉、重复

c#只支持单继承，需要多继承时用接口实现

只能继承自类，不能继承自结构体，结构体只能实现接口

不写明父类时父类为Object类

B是A的子类

A a = new A() a为静态类型（没有override）

A a = new B() a为动态类型（override）

用is操作符类型检查，判断一个对象是否属于某个类

null值可以转换成任一个类的对象

as操作符

是不是某个类型，如果是就等于，如果否就为空

A a = new C();

B b = a as B; // if (a is B) b = (B)a; else b = null;

override 重写

virtual 虚函数 可以在子类中重写，不写virtual也可以执行

静态函数不能重写

隐藏

在子类中声明方法时 public new int x() 让编译器知道父类可以被子类覆盖掉,不会报warnning

动态绑定

从声明时最父类静态方法向子类中往下（动态类型）找，找到非override的方法就调用，否则一直往下找，不管是否被子类覆盖

可以解决Java语言中比较脆弱的基类问题，对调用的函数如果不能分清是应该调用父类的函数还是子类的函数，比如父类的某个函数里调用了父类的另一个函数而且这个被调用的函数在子类中也存在，在调用时没有警告，可能会引起巨大的问题。（低级格式化、高级格式化问题）

子类和父类的构造函数

子类构造时会先调用父类的缺省构造函数，再执行子类的构造函数，如果父类没有无参的构造函数，会报错，解决方法：给父类定义无参构造函数；2.在子类构造函数执行时先显式定义父类构造函数，可以这样写

class B:A{ public B(int x) :base(x){}} 调用了A的有参构造函数A(int x){ }

internal关键字，在程序集内可见

protected 关键字，自身和子类可见

protected internal 在自身和子类以及程序集里可见

抽象类

抽象方法 abstract，只有声明没有定义

抽象方法一定是虚方法，必须被重写，不再用virtual修饰

有抽象方法、抽象属性（get set不写函数体）的类就是抽象类

封装类

sealed关键字，不允许再被继承

Object类

复制，深复制和浅复制， 浅拷贝拷贝对象的本身，不拷贝对象引用的内容，深拷贝还拷贝对象引用的内容

输出对象时自动调用tostring()

=号和!=号如果重载应该同时都重载

接口

声明的接口一律以i打头，且i和i后的一个字母都大写

所有的方法只有定义没有实现（纯抽象类），没有析构函数，构造，运算符重载、嵌套类型

成员不能是静态

不可以含有成员变量

结构体可以实现方法，但是不叫继承

如果一个类实现的多个接口中出现了同名函数，在实现时要写明：

接口名.函数名（）

注意：这种情况不能把同名函数写成Public，会报错，是CLR的问题

代理（或叫委托）和事件

方法类型

返回类型 + 函数名 + (参数列表) 返回类型和参数组成方法的签名

delegate + 返回类型 + 委托名 + (参数列表) 声明委托类型

委托名 委托变量名 ;//声明一个委托类型的委托变量

方法签名和委托类型里的签名（返回类型和参数类型）一致

把委托变量赋值为方法名（也可以new一个委托类型的变量new A(函数名)），则委托变量相当于一个函数指针了，可以调用函数

多播委托

"委托变量 += 方法"可以再绑定一个方法，所以在执行委托时会执行两个函数

可以"委托变量 -= 方法"解绑方法

如果绑定的方法是函数，即有返回值，则最后委托变量的返回值是最后一个函数的返回值，如果参数有ref，参数是分发式，所以不会发生后边的函数要调用的参数被前边的函数修改的情况（即先传入了所有的参数再返回，都是原有的参数）

事件

特殊的委托

把方法的调用和修改隔离开，只有声明方法的类可以触发event域，外部只能通过+=或者-=来改变event域，可以让委托变量不会被外部随意修改

异常

c#抛出异常以后可以不处理，会抛给操作系统

鲁棒性

一层一层往调用者处抛，直到被catch或抛给操作系统

多播委托

命名空间和程序集

命名空间 容器，里边放的是各种类型，包含类、接口、结构体、枚举类型、委托……

可以嵌套

同样一个namspace在两个.cs中都可以用

不写明时默认的namespace是所在项目的工程名

上下级的目录（父目录和子目录）并不互相知道，不能通用，需要用目录名后加点，来调用目录中的变量

有没有using namespace对目标文件的大小没有影响

c#的.exe文件是文本文件，需要类似于JVM的东西来在操作系统上跑，是CLR

而C的.exe文件是二进制文件，可以直接在操作系统上跑

一个c#工程文件可以有多个namespace，不同于java的每个工程只能有一个package

动态链接库 在大型项目中使用，执行到的时候调用

外包

程序集

公有程序集 dll文件

强名字 签名

数字签名

非对称加密算法 对很长的密码来说不实用 密钥 分公开密钥和私有密钥 私有密钥很关键

主要应用领域：公钥加密，私钥签名

采用版本控制方法

property 属性

attribute 特性

在类、方法等前边有方括号时，里边是特性，写版本号、文化、密钥等信息

**签名**

**sn.exe 获取私钥文件**

**MD5算法 2进制长度128位**

**对文件进行计算，MD5值如果相同说明没有被篡改 防篡改**

**哈希函数，单向函数，不能逆向求解（已经被攻破，通过’碰撞’，去找同哈希值的内容）**

**验证签名：**

**私有密钥生成签名，如果公有密钥和签名执行出来的md5值和私有密钥生成的md5相同，可以证明是这个密钥生成的签名**

**internal**

**程序集内可见，在不同的程序集里即使是同一个namespace也不可见，在同一个程序集里即使不在同一个namespace，通过using所在的namespace也可见**

**Attributes**

**特性**

**元信息**

**public sealed abstract等都是预定义的特性**

**一定要用方括号括起来，里边是attribute**

**[Obsolete(参数1，参数2)]只能传入两个参数**

**放在类型、成员、程序集等的前边**

**进程**

**调度的最小单位 （资源分配最小单位是线程）**

**线程 共享一个数据区**

**并行编程**

**线程的状态：**

**开始之前 unstarted**

**start 开始 running**

**suspend 挂起 suspended**

**resume 重新恢复 running**

**abort 退出 stopped**

**线程.join 等待线程结束再继续执行**

**调用sleep 即使是0，也会导致线程退出cpu不再占用**

**互斥**

**复习**

**c#生成的.exe文件只是文本文件，与c++和java不同，不能够直接执行，需要类似于jvm的framework**

**基础、引用、结构体、类、继承、委托、线程**

**多线程**

**装箱拆箱 引用类型和值类型的相互转化**

**泛型 抽象，“模板”,类似于c++中的模板**

**有尖括号的类是泛型类，可以用于类、结构体、接口、泛型方法，Element叫占位符，可以被替代**

**class Buffer<Element>{**

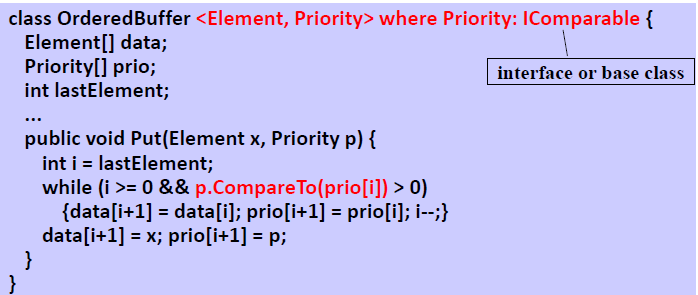
**}**

**使用的时候实例化，Buffer<int>a = new Buffer<int>(100)**

**还可以带多个参数**

**class Buffer <Element, Priority>，两个参数可以是不同类型**

**还可以对参数添加限制，比如要求多个参数可以进行比较（比如同类型数字）**



**根据优先级插值排序**

**泛型的父类和子类问题**

**泛型子类可以继承自泛型类，也可以继承自已经实例化的泛型（其实已经不是泛型了），也可以继承自普通类**

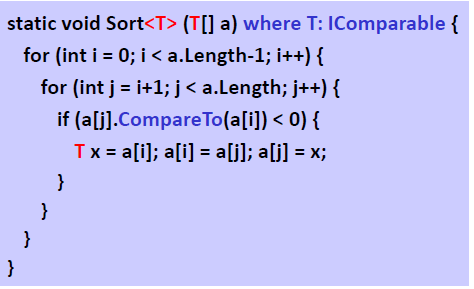
**子类和父类的泛型可以不同（参数个数）**

**不允许父类是泛型类而子类不是泛型类**

**泛型类必须包含泛型方法**

**泛型的重载**

**父类的虚函数的参数可以是Element（占位符），子类里重载的时候必须把泛型实例化，不能再用占位符**



**泛型实现的冒泡排序**

**泛型方法里赋值必须是default(T) T为占位符 不能赋值为0/null等**

**JIT即时编译器**

**泛型类只在实例化为新的类型（此处指引用、值等，一次引用编译可以让以后都不必再编译引用类型，一次int编译可以让以后不需要再编译int类型）的才会编译，否则只在第一次出现的时候编译**

**LINQ**

**Lambda表达式**

**委托值**

**无名的委托**

**有‘’=>‘’为Lambda表达式**