C#复习

1. 字符和字符串函数

1.1. struct 结构

1.2. IsUpper 字符是否是大写字母

1.3. IsLower 字符是否是小写字母

1.4. Length 长度

1.5. Trim 去掉字符串前后空格

1.6. EndsWith 判断字符串是否以匹配项结尾

1.7. IndexOf 查找首个匹配项的索引

1.8. LastIndexOf 查找最后一个匹配项的索引

1.9. Substring 截取字符串

1.10. Split 分割字符串

1.11. Replace 替换字符串

1.12. 字符类型Char

1.12.1. IsControl 指示指定的Unicode字符是否属于控制字符类别

1.12.2. IsDigit 指示某个Unicode字符是否属于十进制数字类别

1.12.3. IsLetter 指示某个Unicode字符是否属于字母类别

1.12.4. IsLower 指示某个Unicode字符是否属于小写字母类别

1.12.5. IsUpper 指示某个Unicode字符是否属于大写字母类别

1.12.6. IsWhiteSpace 指示某个Unicode字符是否属于空白类别

1.12.7. Parse 将指定字符串的值转换为它的等效Unicode字符

1.12.8. ToString 将此实例的值转换为其等效的字符串表示

1.13. 转义字符

1.13.1. \a 响铃　ASCLL码值 007

1.13.2. \b 退格(BS)，将当前位置移到前一列 008

1.13.3. \t 水平制表(HT)（跳到下一个TAB位置） 009

1.13.4. \n 换行(LF)，将当前位置移到下一行开头 010

1.13.5. \v 垂直制表(VT) 011

1.13.6. \f 换页(FF)，将当前位置移到下页开头 012

1.13.7. \r 回车(CR)，将当前位置移到本行开头 013

1.13.8. \\ 代表一个反斜线字符"\' 092

1.13.9. \' 代表一个单引号（撇号）字符 039

1.13.10. \" 代表一个双引号字符 034

1.13.11. \0 空字符(NULL) 000

1.13.12. \ddd 1到3位八进制数所代表的任意字符 三位八进制

1.13.13. \xhh 1到2位十六进制所代表的任意字符 二位十六进制

2. 函数

2.1. return　返回

2.1.1. 一个函数只允许有一个返回值

2.1.2. 带返回值函数体中可以出现多个return语句，但是只有一个会被执行

2.2. parameter　参数

2.2.1. 形式参数

2.2.2. 实际参数

2.3. ref　使用引用传递参数时的关键字

2.4. 函数重载

2.4.1. overload　重载

2.4.2. params　定义动态数组参数时的关键字

2.5. 系统函数

2.5.1.

2.6. 用户自定义

2.7. 函数作用

2.7.1. 1、定义一次，多次调用，提升代码重用性

2.7.2. 2、将逻辑独立的代码定义成一个函数在调用

2.7.3. 子主题 3

2.8. 语法：【访问修饰符】[static] 返回值类型 函数名 （形参列表）{
函数体代码块 }

2.8.1. 调用： 函数名 （实际参数）

3. 冒泡排序

3.1. Bubble 冒泡

3.1.1. int[] sortArray = new int[]{ 6,2,4,1,5,9 };
Console.WriteLine("初始数组中元素的值为："); foreach (int i in sortArray) {
Console.Write("{0}\t", i); } //冒泡排序的实现 //获取数组的长度 int n =
sortArray.Length; //总共比较n-1趟

3.2. Sort 排序

3.2.1. for (int i = 1; i < n; i++) { //每趟从第一个元素到第n-
i个元素进行两两比较 for (int j = 0; j < n - i; j++) {
//两两比较，如果前一个数大于后一个数，则交换位置 if (sortArray[j] >
sortArray[j + 1]) { int t = sortArray[j]; sortArray[j] =
sortArray[j + 1]; sortArray[j + 1] = t; } } }

3.3. Quick Sorting 快速排序

3.4. Selection Sorting 选择排序

3.5. Insertion Sorting 插入排序

3.6. 算法原理 **1.** 比较相邻的元素。如果第一个比第二个大，就交换它们两个。 **2.**
对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。 **3.**
针对所有的元素重复以上的步骤，除了最后一个。 **4.**
持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

4. 数组

4.1. 一维数组

4.1.1. 声明数组

语法：数据类型 [ ] 数组名；

4.1.2. 创建数组

数组名 = new 数据类型 [数组长度]；

4.1.3. 初始化数组

数组名 [索引] = 值；

4.2. 二维数组

4.2.1. 声明二维数组

数据类型[,] 数组名;

4.2.2. 创建二维数组

数组名 = new 数据类型[长度1，长度2];

4.2.3. 初始化二维数组

数据类型[,] 数组名= new 数据类型[长度1,长度2];

5. 语法元素

5.1. 语句

5.2. 注释

5.3. 关键字

5.4. 标识符

5.5. 括号

5.5.1. 大括号"{。。。}"代码块使用

5.5.2. 小括号"（.....）"函数使用

5.5.3. 中括号"[....]"数组，集合使用

5.6. 空白字符

6. 输入，输出

6.1. Console.WriteLine()函数 输出

6.2. Console.ReadLine()函数 输入

6.3. Read（）函数 返回ASCLL码值

6.4. ReadLine 输入并返回字符串

6.5. Parse 转换

7. 数据类型

7.1. object 所有类型的基类（所有类型都可以转换成object）

7.2. 值类型

7.2.1. 简单类型

有符号整数型：sbyte short int long

无符号整型：byte ushort uint ulong

Unicode字符：char

表示一个16位（Unicode）字符

单精度浮点型：float

双精度浮点型：double

高精度浮点型：decimal

布尔型：bool

表示true或false

7.2.2. 枚举类型

enum E{.....}形式的用户定义类型

7.2.3. 结构类型

strut S{.....}形式的用户定义类型

7.3. 引用类型

7.3.1. 类类型

所有其它类型的基类：object

C#的基类，其他所有类型都是由它派生而来的

Unicode字符串：string

表示一个Unicode字符串

class C{....}形式的用户定义类型

7.3.2. 数组类型

一堆数组和多维数组，如int[]和int[,]

7.3.3. 接口类型

interface I{.....}形式的用户定义类型

7.3.4. 委托类型

delegate TD {.....}形式的用户定义类型

7.4. 本章小结

7.4.1. C#中的数据类型分为值类型和应用类型。值类型数据存储在堆栈中，运行效率高，但只能处理固定大小的简单数据。引用类型数据在堆中分配内存，能处理任意大小的复杂数据，但运行效率低。

7.4.2. Object是引用类型，是所有类型的基类

7.4.3. 常用的数据类型包含整数，浮点数（小数），布尔，字符，字符串。整数一般用int表示，字符型用char表示，布尔型用bool表示，小数用double表示，字符串用string表示。

8. 变量和常量

8.1. 变量

8.1.1. 1、声明变量：根据数据的类型，开辟内存空间

8.1.2. 2、初始化变量：将数据初始值存入内存空间

8.1.3. 3、访问变量：读取变量的值或重新为变量赋值

8.1.4. 作用域：1、变量的作用域是指可以访问该变量的代码区域，也就是能够获取或者设置这个变量值的区域

2、局部变量的作用域：是声明变量的语句快内，位于声明变量之后的区域。3、同名的局部变量不能在同一作用域内声明两次，否则会出现编码错误。

8.2. 常量

8.2.1. 定义语法：CONST 数据类型 常量的名称=值

8.2.2. 1、字符常量

8.2.3. 2、符号常量

8.2.4. 3、枚举类型

8.3. 子主题 4

9. 程序调试

9.1. 错误类型

9.1.1. 语法错误

9.1.2. 语义错误

9.2. 调试基本过程

9.2.1. 设置断点，启动调试，单步执行，观察变量，发现问题，停止调试修改代码，重新调试

9.3. Ctrl+F5 表示步调试程序，F5表示启动调试

9.4. F10 表示逐过程调试程序，F11表示逐语句调试程序

10. 运算符

10.1. 算术运算符

10.1.1. + - * / %

结果：数值

10.2. 自运算

10.2.1. ++i（先增减，后使用）

10.2.2. i++（先使用，后增减）

10.3. 关系运算符

10.3.1. > >= < <= == !=

结果类型：布尔值（**true**或**false**）

10.4. 逻辑运算符

10.4.1. && ||和！

结果类型：布尔值（**true**或**false**）

10.5. 条件运算符

10.5.1. 三元运算符：？ ：

10.6. 赋值运算符

10.6.1. = += -= *= /= %=

10.7. 运算符的运算顺序

10.7.1. 当一个表达式包含多种运算符时，运算符的优先级控制着运算顺序，运算符级别高的先运算。

10.7.2. 当同一级运算符同时出现时按照运算符的结合性（从左向右或从右向左）运算

10.7.3. 我们还可以加小括号提高运算符的优先级

10.8. 本章小结：表达式是指由操作数和运算符组成的用于完成某种运算功能的语句

2、常用的运算符包括算术运算符，赋值运算符，自运算符，关系运算符和逻辑运算符，条件运算符。**3、**自运算符**++**或**--**，表示变量自身加**1**或减**1**，和其他运算符结合使用时，区分前后缀，前锥时“先运算，后使用”，后缀时 “先使用 后运算”。

11. 运算符的优先级顺序

11.1. 初级运算符

11.1.1. （） **[ ]**

左关联（从左边向右边）

11.2. 一元运算符

11.2.1. + - ！ ++ --

右关联（从右向左）

11.3. 算术运算符

11.3.1. 先算 *** / %** 再算 **+ -**

左关联（从左向右）

11.4. 关系运算符

11.4.1. 先算**<> <= >=** 再算 **== ！ =**

左关联（从左向右）

11.5. 逻辑运算符

11.5.1. 先算&& 后算 ||

左关联（从左向右）

11.6. 条件运算符

11.6.1. ？ ：

右关联（从右向左）

11.7. 赋值运算符

11.7.1. = *= /= %= += -=

右关联（从右向左）

12. 类型转换

12.1. Parse

12.2. ToString

12.3. Convert

13. 结构化程序

13.1. 顺序语句

13.2. 选择语句

13.2.1. IF

1、单分支if语句

if （布尔表达式）{【如果条件成立执行的语句块】}

2、双分支if语句

if （布尔表达式）{【条件成立执行语句块1】； } else {
【条件不成立执行的语句块2】； }

3、多分支if语句

if （）{ }else if { } else { }

4、嵌套if语句

13.2.2. Switch

break 中断 退出

break用于退出当前switch语句，不再执行switch结构中
剩余代码，而是去执行switch语句结构之后的代码。

default 默认

default语句表示，所有case的值都不与switch表达式的
值相等时，就去执行的代码，可以省略。

switch 开关 互斥分支语句关键字

case 情况 状况

goto

goto语句一般会不标签一起使用，标签用于标出代码中的一个位置，而got
o 语句则在需要的时候跳转到某个标签标示的位置。
定义标签应在独立的一行代码中。其格式如下： 标签名：代码行;
使用goto语句时，其格式如下： goto 标签名;

switch之后表达式结果的类型只能是各种整数、字符串、
字符类和布尔型。 多分支选择语句——switch，通过判断某个表达式的值不
case后常量的值是否相等，来选择是否执行该case后的语
句块，执行完后，跳出switch语句

continu

continue语句只能用在循环体中，循环中的语句遇到continue语句时，会停止当前循环体代码的执行，并重新开始下一次的循环。在有的需求逻辑里， continue可以实现循环加速的功能

13.3. 循环语句

13.3.1. while， do...while

13.3.2. 相同点：1、都是循环语句，有固定的语法。2、都是由循环条件和循环体构成。 3、循环的核心都是控制好循环的次数，而次数又由三要
素决定：初始部分，终止条件，迭代部分。三要素巧
妙结合，控制循环正常执行

13.3.3. for

13.4. 小结：1、程序的三种基本结构：顺序结构，选择结构，循环结构。实现这些结构的各种流程控制语句分别时选择语句，循环语句和跳转语句

13.5. 2、顺序结构程序：按程序的书写顺序自顶向下，依次执行

13.6. 3、选择语句根据不同条件，选择性执行部分代码，包括：if 语句 和 switch语句

13.7. 4、if语句有4种形式：单分支if形式；双分支if...else形式；多分支 if...else if ;嵌套if 形式。

14. 流程控制语句

14.1. 选择语句

14.1.1. IF

单，双，多，嵌套

14.1.2. Switch

14.2. 循环语句

14.3. 跳转语句

15. 捕获异常

15.1. try { } catch(Exception) { throw； } finally { }

15.2. try

15.3. catch

15.4. throw

15.5. finally