第一章 第一个C#程序

C#常用的数据类型

|  |  |  |
| --- | --- | --- |
| 常用数据类型 | java | C# |
| 整型 | int | Int |
| 浮点型 | float | float |
| 双精度型 | double | double |
| 字符串型 | String | string |
| 布尔型 | boolean | bool |

C#变量命名规则：

组成：52个英文字母（A~Z，a~z）,10个数字（0~9），下划线（-），除此之外不能含有其它字符

开头：只能以字母下划线开头；

不能使用的：不能是C#中的关键字，

C#中的常量：

Const 数据类型常量名称=值；

常量命名规范：

1.常量名称必须具有一定的额实际意义

2.常量名称最好以大写字母来命名，中间可根据意义的连接性作用下划线做连接，每个常量定义的右侧最好有简单的注释，说明其作用。

3.常量名称长度最好不要超过25个字符，否则可读性差。

C#向控制台输出：

Console.writeline（）输出格式：

1. console.writeline();
2. Console.writeline(要输出的值);
3. Console.writeline(“格式字符串”，变量列表);

向控制台输入：

Console.readline（）；

类和对象

自定义方法：

[访问修饰符] 返回值类型 方法名（参数列表）

｛

//方法的主体

｝

[访问修饰符] class 类名

｛

//类的主体

｝

Visual studio 调试技巧

1. 设置断点(breakpoint)

设置一个断点，采取的方法时将光标停在这一行，按F9键，

程序调试时常用的快捷键

* F5：开始调试；
* Shift+F5：停止调试
* F9：设置活取消断点；
* Ctrl+F9：取消断点；
* F10：单步执行；
* F2：转到所调用过程或变量的定义；
* Ctrl+F2：将焦点转移到类的下拉列表框；

第二章 C#语法快速热身

If结构

（与java相同）

Switch 结构

Java中 break可以没有

C# 中 break 必须有

数组与循环

一维数组：语法;

数据类型[] 数组名；

数据类型 数组名[]；

Foreach循环

foreach（元素类型 元素变量名 in数组）

{

//代码块；

}

冒泡排序：

class program

{

static void Main(string[] args)

{

int[] socre=new int[5];

int i,j;

Int temp;

Console.WriteLine(“请输入5个学员的成绩”);

for(i=0;i<score.Length;i++)

{

Console.WriteLine(“请输入第{0}个学员的成绩”，i+1);

score[i]=int.Parse(Console.ReadLine());

}

//开始使用冒泡排序

For(i=0;i<socre.Length-1;i++)

{

For(j=0;j<socre.Length-1;j++)

{

If(score[j]>score[j+1])

{

Temp=score[j];

Score[j]=score[j+1];

Score[j+1]=temp;

}

}

}

Console.WriteLine(“升序后的成绩为：”);

For（i=0;i<score.length;i++）

{

Console.Write(“{0}\t”,score[i]);

}

Console.ReadLine();

}

}

(口诀：（升序）

* N个数字来排队，两两相比小靠前
* 外层循环N-1，内层循环N-1-i
* 如果要降序，只要把程序中的大于号换成小于号就行了

)

第三章 使用属性

C#中的访问修饰符

|  |  |  |
| --- | --- | --- |
| 访问修饰符 | 访问权限 | 访问级别 |
| public | 不受任何权限 | 最高 |
| private | 只有类的内部使用 | 最低 |

This 关键字

使用this关键字可以解决成员变量和局部变量名称冲突的问题。

用属性实现字段封装

private string\_name;

public string Name

{

get{return \_name;}

set{\_name=value;}

}

值传递和引用传递

值传递是将变量的值复制一份传递给方法，使得该方法的形参和实参的值相同。在调用的方法中修改形参也只是对实参复制品的数据做更新，并没有真正改变实参的值。

引用传递：将要传递的对象的引用复制给方法的形参，使得被调用的方法直接对引用对象进行更改，会影响实参原来的值。

引用传递使用ref修饰；

第四章 深入C#的String类

运算“==”和Equals（）方法

“==”判断的是两个字符串对象在内存中的首地址，即判断是否是同一个字符串对象，而Equals（）方法判断的是两个字符串对象的值是否相等

“”和String.Empty：

“”：为String对象分配一个长度为零的存储空间。

String.Empty：表示空字符串，不会为对象分配存储空间。

Format格式化

String 变量名=string.format（“格式字符串”，参数列表）；

类型转换：

使用Convert类型进行转化

|  |  |
| --- | --- |
| 方法 | 说明 |
| Convert.ToInt3() | 转换为int型 |
| Convert.ToSingle() | 转换为float型 |
| Convert.ToDouble() | 转换为double型 |
| Convert.ToString | 转换为string型 |

第六章 程序数据集散地：数据库

当今常用的数据库

Oracle SQL Sever IBM DB2 MySQL

数据库管理系统和数据库系统

数据库管理系统：是一种系统软件，由一个互相关联的数据集合和一组访问数据的程序构成。

数据库系统：是一个实际可运行的系统，可以对系统提供的数据进行存储、维护和应用，它是由存储介质、处理对象和管理系统共同组成的集合体，通常由软件、数据库和数据库管理员组成。

数据库由数据库管理系统同一管理，数据的插入、修改和检索都要通过数据库管理系统进行。数据库管理员在数据库中负责创建、监控和维护整个数据库，使数据能被有权限使用的人有效使用。

数据的冗余：在数据库系统中，数据重复的现象就是数据的冗余。

第七章 用表组织数据

SQL Sever 创建表的过程是设置数据列属性的过程，同时也是实施数据完整性（包括实体完整性、引用完整性和域完整性）的过程。

实体完整性要求数据不能有重复，每一行数据都由主键来唯一确定。

域完整性实现了对输入到特定列的数值的限制。

创建数据表需要确定表的列名，数据类型，是否允许为空，还需确定主键，必要的默认值，标识列和检查约束。

如果创建了主表和从表的关系，有一下三种情况：

1. 从表中相关项目的数据在主表中必须存在。
2. 主表中相关项的数据更改了，则从表对应的数据项也应当随之更改。
3. 在删除从表之前，不能够删除主表。

第八章 用SQL语句操作数据

SQL语言是数据库能够识别的通用的指令集。

在T-SQL中，where 用来限制条件，其后紧跟条件表达式

一次插入多行数据，可以使用insert、select语句、select

Into语句或union关键字来实现

使用update更新数据，一般都有限制条件

使用delete删除数据时，不能删除主键值被其它数据表引用的数据行。

数据库的导入/到出功能可以实现与文本.Excel等文件交换数据

1. 数据查询基础

查询将逐行筛选表中的数据，最后将符合要求的记录重新组合称“记录集”记录集的结构类似与表结构

使用IS null判断一行中的某列是否为空

使用order by 进行查询记录及的排序，并且可以按照多个列进行排序

在查询中，可以使用常量、表达式、运算符

在查询中使用函数，能够向程序中那样处理数据

第十章 模糊查询和聚合函数

通配符是一类字符，它可以代替一个或多个正真的字符，查找信息时作为替代字符出现

“-”和“%”分别匹配一个字符或多个字符

使用like、 between、in关键字，能进行模糊查询

聚合函数能够针对列进行计算，对于分析和统计非常有用

常用的聚合函数有sum（），avg（），max（），min（），count（）。

1. 连接查询和分组查询

分组查询时针对表中不同的组分类统计的，group by子句通常会结合聚合函数一起使用。

Having子句能够在分组的基础上，再次进行筛选

多表之间的查询可以使用连接查询，连接查询可以分为内连接和外连接查询

最常见的连接查询时内连接查询（inner join ......on），通常会在相关表之间提取引用列的数据项。

1. 使用ADo.net访问数据库

ADO.Net结构，利用ADO.Net技术可以实现应用程序对数据库的访问操作

ADO.Net是.Net Framework中的一组允许应用程序与数据库交互的类

ADO.Net的两个主要组件是.Net framwork数据提供程序和Dataset。

.Net framwork 数据提供程序包括四个核心对象；Connection、Command、DataAdapter、DataReader。

Connection对象用于建立应用程序和数据库之间的连接、需要定义连接字符串、必须显示关闭和打开数据库连接。

Command对象允许向数据库传递请求，以及检索和操作数据库中的数据。

Command对象的ExecuteScalar（）方法可以检索数据库并返回一个值。

数据库操作过程中可能出现异常，可以使用try-catch-finally语句处理异常。

1. 使用ADO.Net查询和操作数据

StringBuilder类与System。String类的主要区别；Stirng类具有不变性；StringBuilder类常用来处理字符串的修改操作

DataReader对象是一个只进，只读的数据流，每次从数据源中提取一条记录

使用DataReader对象可以获得查询的数据

通过Command对象的ExcuteReader（）方法返回一个DataReader对象。

读取DataReader对象中的数据时，每调用一次Read（）方法将获的一行数据

完成数据查询后，要掉用Close（）方法关闭DataReader。

使用Command对象的ExecuteNonQuery（）方法可以执行数据源数据的增、删、改操作

使用Command的对象的ExecuteNonQuery（）方法返回受影响的行数。