**查询语句**

[去重]select DISTINCT(sex) from my\_student;

[在分组前进行查找]select \* from my\_student where name = '哈哈1';

[分组]

selectc\_id,count(\*),max(height),min(height),avg(height),sum(age) from my\_student group by c\_id ;

select c\_id,sex,count(\*),max(height),min(height),avg(height),sum(age)from my\_student group by c\_id ,sex;

selectc\_id,sex,count(\*),max(height),min(height),avg(height),sum(age),GROUP\_CONCAT(name)from my\_student group by c\_id ,sex;（分组后还可以显示 指定字段的所有数据）

[在分组前后进行查找]

selectc\_id,count(\*),max(height),min(height),avg(height),sum(age) from my\_student group by c\_id having COUNT(\*) >= 3;

[排序]select \* from my\_student order by c\_id;

select \* from my\_student order by c\_id,sex;

PS:<asc默认升，desc降>

[限制结果的数量]select top (n-m+1)\* from table where……(选择第m到第n条数据)

用is null或者is not null来判断是否为空行

[找出北京，然后添加地址列来存放北京] select name '北京' as 地址 from　a

[找出姓名] select name as 姓名from a where  gender='男'

[找出前六行] **select top 6 name from a**

**[**模糊查询] select \* from a where name like '赵%'

[范围查询] select \* from a where age between 18 and 20

[在列举中查询] select name from a where address in ('北京','上海','唐山')

**增删改**

【插入单行】insert [into] <表名> [列名] values <列值>

【插入新表】insert into <已有的新表> <列名> select <原表列名> from <原表名>

【删除部分】**delete from <表名> [where <删除条件>]**

**【删除全表】truncate table** <表名>

【更新】**update <表名> set <列名=更新值> [where <更新条件>]**

选择：select \* from table1 where 范围

插入：insert into table1(field1,field2) values(value1,value2)

删除：delete from table1 where 范围

更新：update table1 set field1=value1 where 范围

查找：select \* from table1 where field1 like ’%value1%’ ---like的语法很精妙，查资料!

排序：select \* from table1 order by field1,field2 [desc]

总数：select count as totalcount from table1

求和：select sum(field1) as sumvalue from table1

平均：select avg(field1) as avgvalue from table1

最大：select max(field1) as maxvalue from table1

最小：select min(field1) as minvalue from table1

[集合]UNION：组合其他两个结果表并消去表中任何重复行而派生出一个结果表。当 ALL 随 UNION 一起使用时（即 UNION ALL），不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。

[排除]EXCEPT:包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。当 ALL 随 EXCEPT 一起使用时 (EXCEPT ALL)，不消除重复行。

[交叉]INTERSECT 运算符通过只包括TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。当 ALL 随 INTERSECT 一起使用时 (INTERSECT ALL)，不消除重复行。

唯一索引/主键索引（UNIQUE）：

聚集索引（CLUSTERED）：按顺序进行索引，会改变原有数据的排序

非聚集索引（NONCLUSTERED）：不影响

|  |  |  |
| --- | --- | --- |
| 属性：public 属性类型 属性名  定义属性需要名字和类型。  属性包含两个块：get块和set块。 | 函数：public 返回值类型 函数名（）  有（）跟 返回值类型  构造函数：public 构造函数名（）  有括号，无返回值 | |
| 记忆：变量/字段名.方法名(); | 记忆：类名.方法名(); | |
| 调用实例成员：对象.实例成员  【初始化赋值/////////实例化new】 | 调用静态成员：“类名.静态成员” | |
| 属性：是函数的成员  字段：是类的成员 | 对象：字段/变量  **成员**：属性/方法/索引器 | |
| 创捷类：   1. 生成字段 2. 定义属性 3. 声明构造函数 4. 定义函数 | 结构体：值类型，用于不同类型变量，少用 | 类：引用类型，用于同种类型变量，常用 |
| 接口：一种引用类型，可以制定一组数据但是不引用数据，  默认：接口内部成员是隐式public,不能添加任何修饰；以interface为关键字，I开头命名， | |

|  |  |
| --- | --- |
|  | 1）ArrayList ArrayList是List接口的可变数组非同步实现，并允许包括null在内的所有元素，相当于List < object> 2）List < T > 泛型的List 类是一个**不限长度的集合**类型，它内部实际就是一个数组，初始长度是4，每次数组到达限制，就会把现有容量翻倍，它提供用于对集合进行搜索、排序和操作等方法 List是数组链表，数组链表访问快,复杂度O(1),但是添加删除复杂度O(n) 3）LinkedList LinkedList是List接口的双向链表非同步实现，并允许包括null在内的所有元素。 底层的数据结构是基于双向链表的， LinkedList是指针链表，指针链表访问复杂度是O(n),但是添加删除很快O(1),如果对这个集合在中间的添加删除操作非常频繁的话，就建议使用LinkedList。 4）Dictionary < K, V> 存储键值对的关联性集合，查询等操作速度很快,因为它的时间复杂度是O(1) ，单线程中推荐使用Dictionary,有泛型优势,且读取速度较快,容量利用更充分. 5）Hashtable Hashtable是System.Collections命名空间提供的一个容器，用于处理和表现类似key/value的键值对 **基本概念** Hashtable使用了闭散列法来解决冲突，它通过一个结构体bucket来表示哈希表中的单个元素，这个结构体中有三个成员： 　（1） key ：表示键，即哈希表中的关键字。 　（2） val ：表示值，即跟关键字所对应值。 　（3） hash\_coll ：它是一个int类型，用于表示键所对应的哈希码。  哈希表的所有元素存放于一个名称为buckets（又称为数据桶） 的bucket数组之中 **优点：**  （1）在使用哈希表保存集合元素（一种键/值对）时，首先要根据键自动计算哈希代码，以确定该元素的保存位置，再把元素的值放入相应位置所指向的存储桶中。在查找时，再次通过键所对应的哈希代码到特定存储桶中搜索，这样将大大减少为查找一个元素进行比较的次数  （2）多线程程序中推荐使用Hashtable，对Hashtable进一步调用Synchronized()方法可以获得完全线程安全的类型  **Dictionary**< TKey, TValue> 是 **Hashtable** 的泛型版本，它们之间实现上区别不大，运行效率上有一些差别 Hashtable由于键值类型都object，所以涉及装箱拆箱操作，在添加数据的效率上要差一些，但是频繁使用数据时效率更高，HashTable的优点就在于其索引的方式，速度非常快。如果以任意类型键值访问其中元素会快于其他集合，特别是当数据量特别大的时候，效率差别尤其大。 6）SortedList 表示基于相关的 IComparer 实现按键进行排序的键/值对的集合,与哈希表类似，区别在于SortedList中的Key数组排好序的 7）堆栈（Stack） 表示对象的简单的后进先出非泛型集合。Push方法入栈，Pop方法出栈 8）队列（Queue） 队列先进先出,enqueue方法入队列，dequeue方法出队列 9）SortedList< TKey,TValue> SortedList< TKey,TValue>是支持排序的关联性集合，将数据存储在数组中的。 　　也就是说**添加**和**移除**操作都是线性的，时间复杂度是**O(n)**，因为操作其中的元素可能导致所有的数据移动。 　　但是因为在**查找**的时候利用了**二分搜索**，所以查找的性能会好一些，时间复杂度是**O(log n)**。 　　所以推荐使用场景是这样地：如果你想要快速查找，又想集合按照key的顺序排列，最后这个集合的操作（添加和移除）比较少的话，就是SortedList了。 　　集合中的数据是有序的。可以通过key来匹配数据，也可以通过int下标来获取数据。  添加操作比ArrayList，Hashtable略慢；查找、删除操作比ArrayList快，比Hashtable慢  #####10）SortedDictioanry< TKey,TValue>  SortedDictionary< TKey,TValue>和Dictionary< TKey,TValue>大致上是类似的，但是在实现方式上有一点点区别 　　SortedDictionary< TKey,TValue>用**二叉树**作为存储结构的。并且按key的**顺序排列** 　　**SortedDictionary**< TKey,TValue>相比于**SortedList**< TKey,TValue>其性能优化了 　　SortedList< TKey,TValue>其内部维护的是**数组**而SortedDictionary< TKey,TValue>内部维护的是\*\*红黑树(平衡二叉树)\*\*的一种，因此其占用的内存，性能都好于SortedDictionary< TKey,TValue> 　　**唯一差在不能用下标取值**。 11）HashSet< T> HashSet是一个无序的能够保持唯一性的集合,不支持下标访问。 12）SortedSet< T> SortedSet内部也是一个二叉树，用来支持按顺序的排列元素。 　　算法，存储结构都与哈希表相同，主要是**设计用来做高性能集运算**的，例如对两个集合求交集、并集、差集等。集合中包含一组不重复出现且无特定顺序的元素。 13）BitArray BitArray用于二进制运算，“或”、“非”、“与”、"异或非"等这种操作，只能存true或false； 14）ListDictionary **单向链表**，每次添加数据时都要遍历链表，数据量大时效率较低，数据量较大且插入频繁的情况下，不宜选用 15）HybridDictionary HybridDictionary的类，充分利用了Hashtable查询效率高和ListDictionary占用内存空间少的优点，内置了Hashtable和ListDictionary两个容器，添加数据时内部逻辑如下：  当数据量小于8时，Hashtable为null，用ListDictionary保存数据。  当数据量大于8时，实例化Hashtable，数据转移到Hashtable中，然后将　　ListDictionary置为null。 |
|  |  |
| 属性 | 1、隐藏成员变量，增加程序的灵活性、健壮性。  2、可以实现对数据验证，看数据是否合法。  4、快捷键：再续插入属性的地方输入prop，然后按<Tab>则系统回自动填充属性框架。减少了代码的编写量 |
| 构造函数 | 1、构造函数用于对实例变量的初始化，无返回参数，且不可用void显示表示返回类型为空。其函数名和类名必须完全相同。另外，构造函数可调用该类中其他构造函数（this关键字实现），也可调用基类中构造函数（base关键字实现）。  2、析构函数与构造函数相对应，其在对象被释放之前调用。其函数名表示方法为在类名前加‘~’。其执行时间是不确定的，因为c#中内存回收是通过CLR实现的，当内存可用资源小于下限时，才进行垃圾回收，此时析构函数才将要被执行。 |
| 重载 | 1、方法重载：同一类中，方法名相同，参数类表不同。   2、运算法重载：public static 返回类型 operator 符号（参数列表）{}   3、类型转换运算符：public static explicit（显式转换）|implicit（隐式转换） operator 转换目的类型（参数类表）{} |
| 抽象类 | 1、抽象类由关键字abstract定义，其不能实例化变量。    2、抽象类可含有抽象方法（只含有方法签名，不含有方法体）：访问修饰符 abstract 返回类型 方法名（参数列表）；    3、派生类继承抽象类时，必须重写抽象类中的抽象方法。    4、密封类由关键字sealed定义，其不能有派生类 |
| **this和base** | 1、this表示当前实例，对于静态成员，不能使用this。    2、base表示当前类的基类，可以使用base调用父类的方法。 |
| 简答题 | LinkedList语法和ArrayList一模一样  ArrayList和LinkedList之间的区别？  ArrayList和LinkedList底层数据结构不同 导致优劣势不同  ArrayList：底层是基于数组实现的  优点：随机访问 遍历查找效率高  缺点：添加删除元素的时候效率低  LinkedList：底层是基于链表实现的  优点：添加删除元素的时候效率高。  缺点：随机访问 遍历查找效率低[从下标0开始找起] |

**成员**：属性/字段/方法

静态成员（static）属于类，可以直接通过的方式访问；

非静态成员，必须实例化对象（new）后，通过对象.实例成员来访问;

委托（自定义类）

多态：

（1）子类会以父类的身份出现；且子类特有的方法属性和方法不能使用。  
（2）子类在工作时以自己方式来实现；