Элементы языка программирования C#

Обычный разговорный язык состоит из четырех элементов: ***символов***, ***слов***, ***словосочетаний*** и ***предложений***. Алгоритмический язык содержит подобные элементы: слова называют – *элементарными конструкциями*, словосочетания – *выражениями*, предложения – *операторами*.

Описание *алгоритмического языка* - описание 4-х названных элементов.

- Описание **символов** - перечисление допустимых символов языка.
- Описание **элементарных конструкций** - это правила их образования.
- Описание **выражений** - это правила образования любых выражений, имеющих смысл в данном языке.
- Описание **операторов** - правила образования всех типов операторов языка

Описание каждого элемента языка задается его *синтаксисом* и *семантикой*. *Синтаксические* определения устанавливают правила построения элементов языка (большинство команд C# должно заканчиваться символом *«;»* - одно из правил синтаксиса). *Семантика* определяет смысл и правила использования элементов языка, для которых даны синтаксические определения (в C# перед использованием переменные должны быть объявлены; объявление переменных осуществляется с помощью ключевых слов **int**, **double**, **char**, которые обозначают целочисленный, дробный или символьный тип данных).

Символы языка - основные неделимые знаки, в терминах которых пишутся все тексты на языке. ***Элементарные*** конструкции - это минимальные единицы языка, имеющие самостоятельный смысл. Они образуются из основных символов языка. ***Выражение*** в алгоритмическом языке состоит из элементарных конструкций и символов, оно задает правило вычисления некоторого значения. ***Оператор*** задает полное описание некоторого действия, которое необходимо выполнить. Для описания сложного действия операторы объединяются в ***блок*** (группа операторов).

Алфавит языка C# включает (алфавит – совокупность допустимых символов языка):

буквы: латинские, национальных алфавитов и символ подчеркивания «_»;

цифры: арабские цифры от 0 до 9, шестнадцатеричные цифры от A до F;

специальные символы: " { }, 1 [ ] + — %/ \; ' : ? < > = ! & # ~ *-

{ } [ ] ( ) . , : ; + - * / % & | ^ ! ~ =

< > ? ++ -- && << >> == != <= >= += -= *= /= %=

&= |= ^= <<= >>= ->

пробельные символы: пробел, табуляция, переход на новую строку.

Из символов языка составляются: *директивы* препроцессора, *комментарии* и *лексемы (лексема* (token) - *это минимальная единица языка, имеющая самостоятельный смысл).*

Директивы препроцессора (строка в коде программы, которая начинается с символа # и следующего за ним ключевого слова препроцессора) предназначены для указанию компилятору языка выполнить то или иное действие в момент компиляции программы (пришли в C# из языка C/C++). Препроцессор – это предварительная стадия компиляции программы, на которой формируется окончательный вид исходного текста программы (с помощью директив , инструкций, команд препроцессора можно включить или выключить из процесса компиляции фрагменты кода программы).

Комментарии предназначены для записи пояснений и формирования документации. Комментарии представляют -произвольный текст, следующий за символами // до конца текущей строки (однострочные комментарии) или текст, располагающийся на нескольких строках, ограниченных символами /* в начале комментария и */ - в его конце (многострочные комментарии).

Из символов алфавита языка формируются лексемы языка: идентификаторы, ключевые (зарезервированные) слова, знаки операций, константы, разделители (скобки, точка, запятая, пробельные символы). Границы лексем определяются другими лексемами, такими, как разделители или знаки операций. В свою очередь лексемы входят в состав ***выражений*** (выражение задает правило вычисления некоторого значения) и ***операторов*** (оператор задает законченное описание некоторого действия).

Идентификаторы

Имена в программах служат той же цели, что и имена в мире людей, - чтобы обращаться к программным объектам и различать их. Поэтому имена также называют *идентификаторами*. В идентификаторе могут использоваться *буквы*, *цифры* и *символ подчёркивания*. Язык C# чувствителен к регистру и компилятор различает маленькие и большие буквы. Прописные и строчные буквы различаются, например, *sysop*, *SySoP*, *SYSOP* – три разных имени. Идентификатор может состоять из символов (букв латинского и русского алфавитов, больших и маленьких) цифр и знака подчеркивания, но начинаться может только с **буквы** или **знака подчеркивания**. Длина идентификатора не ограничена. Пробелы внутри имён *не* ***допускаются***.

В идентификаторах можно применять escape-последовательности Unicode, - представлять символ с помощью его кода в шестнадцатеричном виде с префиксом \u (\u00F2).

Полное имя пользовательского типа может состоять из идентификатора пространства имен, к которому он принадлежит, и самого идентификатора (имени) пользовательского типа, которые объединяются символом точка ".", например: **Nnnn.Aaaa** (пользовательский тип Aaaa в пространстве имен Nnnn) или **ConsoleApp.Program** (класс Program из приложения ConsoleApp). Для улучшения читабельности кода программным элементам следует давать осмысленные имена, составленные в соответствии с определенными правилами.

Существует несколько видов нотаций - соглашений о правилах создания имен индентификаторов.

Pascal нотация - каждое слово, входящее в идентификатор, начинается с заглавной буквы. Пример: Age, LastName, TimeOfDeath.

Венгерская нотация отличается от предыдущей наличием префикса, соответствующего типу величины. Пример: fAge, sName, iTime.

Camel нотация: с заглавной буквы начинается каждое слово идентификатора, кроме первого. Пример: age, lastName, timeOfDeath.

Наиболее часто используются нотации **Pascal** или **Camel**.

Ключевые (зарезервированные) слова

Ключевые слова - это предварительно определенные зарезервированные идентификаторы, имеющие специальные значения для компилятора. Их нельзя использовать в программе в качестве идентификаторов, если только они не содержат префикс @. Например, **@if** является допустимым идентификатором, но **if** таковым не является, поскольку if - это ключевое слово. В таблице перечислены ключевые слова, являющиеся зарезервированными идентификаторами любой части программы C# и контекстные ключевые слова C#. Контекстные ключевые слова имеют особое значение только в ограниченном программном контексте и могут использоваться в качестве идентификаторов за пределами этого контекста. Обычно новые ключевые слова добавляются в язык C# как контекстные ключевые слова для того, чтобы избежать нарушения выполнения программ, написанных в более ранних версиях языка.

Ключевые (зарезервированные) слова

Название	Описание
abstract	Модификатор **abstract** используется с классами, методами, свойствами, событиями и индексаторами. Модификатор **abstract** (для класса) указывает, что класс используется в качестве базового класса для других классов. Члены, помеченные как абстрактные или включенные в абстрактный класс, должны быть реализованы с помощью классов, производных от абстрактных классов.
as	Оператор **as** используется для выполнения определенных типов преобразований между совместимыми ссылочными типами.
base	Ключевое слово **base** используется для доступа к членам базового класса.
bool	Слово **bool** является псевдонимом свойства System.Boolean. Оно используется для объявления переменных для хранения логических значений, true и false.
break	Оператор break завершает ближайший внешний цикл или оператор switchя. Управление передается оператору, следующему за завершенным оператором (если таковой имеется).
byte, sbyte, Int, uint, long, ulong, short, ushort	Ключевое слово **byte**, **sbyte**, **int**, **uint**, **long**, **ulong**, **short**, **ushort**,обозначает целочисленный тип данных.
case switch	Оператор **switch** - это управляющий оператор, обрабатывающий множественное выделение и перечисления путем передачи управления одному из операторов case в своем тексте, как показано в следующем примере.
char	Слово **char** используется для объявления символа Юникода в диапазоне, указанном в следующей таблице. Символы Юникода -это 16-разрядные символы, используемые для представления известных письменных языков мира.
checked unchecked	Слово **checked** используется для явного включения проверки переполнения при выполнении арифметических операций и преобразований с данными целого типа. Слово **unchecked** используется для подавления проверки переполнения при выполнении арифметических операций и преобразований с данными целого типа.
class	Классы объявляются с помощью ключевого слова **class**
const	Ключевое слово **const** используется для изменения объявления поля или локальной переменной. Оно указывает на то, что значение поля или локальной переменной является постоянным и не может быть изменено.
continue	Оператор continue передает управление на следующую итерацию итерационного оператора, в котором он находится.
decimal	Ключевое слово **decimal** обозначает 128-разрядный тип данных. По сравнению с типом данных с плавающей запятой, тип **decimal** имеет более точный и узкий диапазон, благодаря чему он походит для финансовых расчетов.
default	Ключевое слово **default** используется в инструкции switch или в универсальном коде: Инструкция **switch**: определяет метку по умолчанию. **Универсальный код**: определяет значение параметра типа по умолчанию. Это будет значение NULL для ссылочных типов и ноль для типов значения.
delegate	Объявление типа делегата аналогично подписи метода. Оно имеет возвращаемое значение и некоторое число параметров.

Название	Описание
do	Оператор **do** повторно выполняет оператор или блок операторов, заключенных в {}, пока определенное выражение не примет значение **false**.
double	Ключевое слово **double** обозначает простой тип, используемый для хранения 64-разрядных значений с плавающей запятой.
If else	Оператор **if** выбирает оператор для выполнения на основе значения выражения **Boolean**
enum	Слово **enum** используется для объявления перечисления, состоящего из набора именованных констант, который называется списком перечисления. Обычно лучше всего определять перечисление прямо в пространстве имен, чтобы всем классам в пространстве имен было одинаково удобно получить к нему доступ. Однако перечисление также может быть вложено в классе или структуре.
event	Ключевое слово **event** используется для объявления события в классе издателя.
explicit	Ключевое слово **explicit** служит для объявления оператора преобразования пользовательского типа, который следует вызывать во время приведения.
extern	Модификатор **extern** используется для объявления метода с внешней реализацией. Когда используется вызов неуправляемого кода, модификатор **extern** используется с атрибутом **DllImport**. В этом случае необходимо объявить метод, как **static**
false, true	Используется в качестве перегруженного оператора или литерала.
finally	Блок **finally** позволяет освободить все ресурсы, выделенные в блоке **try**, а также выполнить код, который должен выполняться даже в случае возникновения исключения. Управление всегда передается блоку finally независимо от того, как была завершена обработка блока **try**.
fixed	Оператор fixed не позволяет сборщику мусора переносить перемещаемую переменную. Оператор fixed допускается только в **небезопасном** контексте.
float	Ключевое слово **float** обозначает простой тип, используемый для хранения 32-разрядных значений с плавающей запятой.
for	Цикл **for** повторно выполняет оператор или блок операторов, пока определенное выражение не примет значение **false**. Цикл **for** удобно использовать для итераций в массивах и для последовательной обработки.
foreach in	Оператор **foreach** повторяет группу вложенных операторов для каждого элемента массива или коллекции объектов. Оператор **foreach** используется для итерации коллекции с целью получения необходимой информации.
goto	Оператор **goto** передает управление именованному оператору. Оператор **goto** также используется для выхода из вложенных циклов со сложной структурой.
implicit	Слово **implicit** служит для объявления неявного преобразования пользовательского типа. Оператор обеспечивает неявное преобразование между пользовательским типом и другим типом, если при преобразовании исключается утрата данных.
interface	Интерфейс содержит только сигнатуры **методов**, **свойств**, **событий** или **индексаторов**. Класс или структура, которые реализуют интерфейс, должны реализовать члены этого интерфейса, указанные в его определении.
internal	Ключевое слово internal является модификатором доступа для типов и членов типов. Внутренние типы или члены доступны только внутри файлов в одной и той же сборке.
is	Проверяет совместимость объекта с заданным типом.

Название	Описание
lock	При помощи слова **lock** блок операторов можно пометить как фрагмент для блокировки, получив блокировку взаимного исключения для указанного объекта, выполнив оператор, а затем сняв блокировку.
namespace	Ключевое слово **namespace** используется для объявления области действия.
new	Слово **new** используется в качестве оператора, модификатора или ограничения.
null	Ключевое слово **null** является литералом, представляющим пустую ссылку, которая не ссылается ни на один объект. **null** является значением по умолчанию для переменных ссылочного типа. Обычные типы значений не могут быть равны **null.**
object	Тип **object** - псевдоним для типа **Object** в .NET Framework. В унифицированной системе типов C#: предопределенные, пользовательские, ссылочные и типы значений, наследуют непосредственно или косвенно от типа **Object**. Переменным типа **object** можно назначать значения любых типов. Когда переменная типа значения преобразуется в объект, говорят - она *упаковывается*. Когда переменная типа **object** преобразуется в тип значения, говорят, она *распаковывается*.
operator	Ключевое слово **operator** используется для перегрузки встроенного оператора или выполнения пользовательского преобразования в классе или объявлении структуры.
out	Слово **out** используется для передачи аргументов по ссылке. Оно похоже на слово **ref**, за исключением того, что **ref** требует инициализации переменной перед ее передачей. Для работы с параметром out определение метода и вызывающий метод должны явно использовать ключевое слово **out**.
override	Модификатор **override** требуется для расширения (изменения) абстрактной или виртуальной реализации унаследованного метода, свойства, индексатора, события.
params	Слово **params** определяет **параметр метода**, принимающий аргумент с переменным количеством параметров. В объявлении метода после **params** другие параметры не допускаются, объявлении допускается только одно слово params.
private	Слово **private** является модификатором доступа к члену. Закрытый (**private**) доступ является уровнем доступа с минимальными правами. Доступ к закрытым членам можно получить только внутри тела класса (структуры), в котором они объявлены.
protected	Слово **protected** является модификатором доступа к члену. Доступ к члену **protected** возможен внутри класса и из производных экземпляров класса.
public	Ключевое слово **public** является модификатором доступа для типов и членов типов. Общий (**public**) доступ является уровнем доступа с максимальными правами. Ограничений доступа к общим членам не существует.
readonly	Слово **readonly** - модификатор, который можно использовать для полей. Если объявление поля содержит **readonly**, присвоение значений таким полям может происходить только как часть объявления или в конструкторе в том же классе.
ref	Слово **ref** используется для передачи аргументов по ссылке. В результате все изменения параметра в методе будут отражены в переменной при передаче элемента обратно в вызывающий метод. Для работы с параметром **ref** определение метода и вызывающий метод должны явно использовать ключевое слово **ref**.
return	Выражение **return** прерывает выполнение метода, в котором оно присутствует и возвращает управление вызывающему методу (также возвращает необязательное значение). Если метод является типом **void**, оператор **return** можно опустить. Если оператор **return** находится внутри блока **try**, блок **finally**, если он существует, будет выполняться до возврата управления вызывающему методу.

Название	Описание
sealed	При применении к классу модификатор **sealed** запрещает другим классам наследовать от этого класса. Модификатор **sealed** используется для метода или свойства, которое переопределяет виртуальный метод или свойство в базовом классе. Это позволяет классам наследовать от класса - родителя, запрещая им при этом переопределять определенные виртуальные методы или свойства.
sizeof	Позволяет получить размер в байтах для **типа значения**.
stackalloc	Ключевое слово **stackalloc** используется в небезопасном контексте кода для выделения блока памяти в стеке.
static	Модификатор **static** используется для объявления статического члена, принадлежащего классу, а не конкретному объекту. Модификатор **static** используется для описания классов, полей, методов, свойств, операторов, событий , конструкторов (не используется с индексаторами, деструкторами или типами, отличными от классов).
string	Тип данных **string** - это последовательность, содержащая ни одного или любое число знаков Юникода. В платформе.NET Framework **string** является псевдонимом для **String**. Хотя тип **string** является ссылочным типом, операторы равенства (== и !=) определены для сравнения значений объектов типа **string**, а не ссылок.
struct	Тип **struct** (структура) - это тип значения, который обычно используется для инкапсуляции небольших групп связанных переменных, например координат прямоугольника или характеристик складской номенклатуры.
switch	Оператор **switch** - оператор, обрабатывающий множественное выделение и перечисление путем передачи управления одному из операторов case в тексте
this	Ключевое слово **this** ссылается на текущий экземпляр класса, а также используется в качестве модификатора первого параметра метода расширения.
throw	Оператор **throw** используется для сообщения о случаях аномальных ситуаций (исключений) в ходе выполнения программы.
try	Инструкция **try-catch** состоит из блока **try**, за которым следует одно или несколько предложений **catch**, в которых определяются обработчики для различных исключений. При возникновении исключения среда CLR ищет оператор **catch**, который обрабатывает это исключение. Если выполняющийся в данный момент метод не содержит такого блока **catch**, то среда CLR рассматривает метод, который вызвал текущий метод, и т. д. по стеку вызовов. Если блок **catch** не найден, то среда CLR отображает пользователю сообщение о необработанном исключении и останавливает выполнение программы.
typeof	Используется для получения объекта System.Type для типа (**System.Type type = typeof(int)**).
unsafe	Ключевое слово **unsafe** обозначает небезопасный контекст, необходимый для работы с указателями.
using	Слово **using** имеет два основных применения. 1) В качестве директивы, когда оно используется для создания псевдонима пространства имен или импорта типов, определенных в других **пространствах** имен. 2) В качестве инструкции, когда оно определяет **область**,, в конце которой объект будет удален.
virtual	Слово **virtual** используется для изменения объявлений методов, свойств, индексаторов,событий и разрешения их переопределения в производном классе.

Название	Описание
volatile	Ключевое слово **volatile** указывает, что поле может быть изменено несколькими потоками, выполняющимися одновременно. Поля, объявленные как **volatile**, не проходят оптимизацию компилятором, которая предусматривает доступ посредством отдельного потока. Это гарантирует наличие наиболее актуального значения в поле в любое время.
void	При использовании в качестве возвращаемого типа метода ключевое слово **void** обозначает, что этот метод не возвращает какого-либо значения. Ключевое слово **void** не может входить в список параметров метода.
while	Оператор **while** выполняет оператор или блок операторов, пока определенное выражение не примет значение false.

Контекстные ключевые слова

Название	Описание
from	Выражение запроса должно начинаться с предложения from
get	Ключевое слово get определяет *метод доступа* в свойстве или индексаторе, который извлекает значение свойства элемента индексатора.
group	Предложение group возвращает последовательность объектов IGrouping(Of TKey, TElement), содержащих ноль или более элементов, соответствующих значению ключа группы.
into	Контекстно-зависимое ключевое слово into можно использовать для создания временного идентификатора с целью сохранения результатов предложения group, join или select в новом идентификаторе. Этот идентификатор может стать источником дополнительных команд запросов.
join	Предложение **join** используется для связывания элементов из различных последовательностей источников, которые не имеют прямых связей в объектной модели. Требование – элементы в каждом источнике должны совместно использовать некоторое значение, которое можно сравнить на предмет равенства.
let	В выражении запроса иногда полезно сохранить результат выполнения какой-то составной части выражения, чтобы использовать его в последующих предложениях. Это можно выполнить с помощью ключевого слова let, создающего новую переменную диапазона и инициализирующего ее результатом предоставленного выражения. хранения другого значения.
orderby	В выражении запроса предложение orderby осуществляет сортировку возвращенной последовательности по возрастанию или по убыванию. Для выполнения одной или нескольких операций последующей сортировки можно указать несколько ключей.
partial (тип)	Определения разделяемых типов позволяют разделять определения для классов, структур и интерфейсов на несколько файлов.
partial (метод)	Подпись разделяемого метода определяется в одной части разделяемого типа, а реализация определяется в другой части типа.
select	В выражении запроса предложение select задает тип значений, получаемых при выполнении запроса. Результат основывается на анализе всех предыдущих предложений и на любых выражениях внутри предложения select.
set	Ключевое слово set определяет *метод доступа* в свойстве или индексаторе, который назначает значение свойства элемента индексатора.

Название	Описание
value	Неявный параметр value служит для настройки метода доступа и для добавления или удаления обработчиков событий.
var	Объявляемые в области метода переменные могут иметь неявный тип var. Локальная переменная с неявным типом имеет строгую типизацию, как если бы тип был задан явно, только тип определяет компилятор.
where (ограничение универсального типа)	Предложение where используется в определении универсального типа для указания ограничений типов, которые могут использоваться в качестве аргументов параметра типа, определенного в универсальном объявлении.
where (предложение запроса)	Предложение where используется в выражении запроса для указания элементов, возвращаемых из источника данных, в выражении запроса. Это предложение применяет логическое условие (*предикат*) к каждому исходному элементу (со ссылкой переменной диапазона) и возвращает элементы, для которых заданное условие является истинным. В одном выражении запроса может присутствовать несколько предложений where, а в одном предложении – несколько частей выражения предиката.
yield	Используется в блоке итератора для предоставления значения объекта перечислителя или для сообщения о конце итерации.

Знаки операций и разделители

Знак операции - это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются.

В выражении D+=B знак += является знаком операции, D и B - операндами.

Символы, составляющие знак операций, могут быть как специальными, например, &&, | и <, так и буквенными, такими как as или new.

Операции делятся на ***унарные, бинарные*** и ***тернарную*** по количеству участвующих в них операндов. Один и тот же знак может интерпретироваться по-разному в зависимости от контекста. Все знаки операций, за исключением [ ], ( ) и ? :, представляют собой отдельные лексемы.

Все знаки операций и разделители, использующиеся в C#:

{ } [ ] ( ) . , : ; + - * / % & | ^ ! ~ = < > ? ++ -- && << >> == != <= >= += -= *= /= %= &= |= ^= <<= >>= ->

Литералы (константы)

В программах на языках высокого уровня (C#) литералами называют последовательности входящих в алфавит языка программирования символов, обеспечивающих явное представление значений, которые используются для обозначения начальных значений в объявлении членов классов, переменных и констант в методах класса.

Литералы или ***константы*** - это неизменяемые величины. В C# различают арифметические, логические, символьные (Escape-последовательности) и строковые литералы, а также константе null. Компилятор, выделив константу в качестве лексемы, относит ее к одному из типов данных по ее внешнему виду (программист может задать тип константы самостоятельно).

Литерал (константа)	Описание	Примеры
Логическая	true (истина) или false (ложь)	True False
Целая	***Десятичная***: последовательность десятичных цифр (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), за которой может следовать суффикс (U, u, L, l, UL, Ul, uL, ul, LU, Lu, lU, lu)	8 0 199226 8u0Lu199226L
-	***Шестнадцатеричная***: символы 0x или 0X, за которыми следуют шестнадцатеричные цифры (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F), а за цифрами, в свою очередь, может следовать суффикс (U, u, L, l, UL, Ul, uL, ul, LU, Lu, lU, lu)	0xA 0x1B8 0X00FF 0xAU0x1B8LU 0X00FFl
Вещественная	***С фиксированной точкой***: [цифры][.][цифры][суффикс] Суффикс — один из символов F, f, D, d, M, m	5.7 .001 35. 5.7F.001d35. 5F.001f35m
-	***С порядком***: [цифры][.][цифры]{E|e}[+|–][цифры] [суффикс] Суффикс — один из символов F, f, D, d, M, m	0.2E6 .11e+3 5E-10 0.2E6D.11e–3 5E10
Символьная	Символ, заключенный в апострофы	'A' 'ю' '*' '\0' '\n' '\xF''\x74' '\uA81B'
Строковая	Последовательность символов, заключенная в кавычки	"Здесь был Vasia" "\tЗначение r = \0xF5 \n" "Здесь был \u0056\u0061" "C:\\temp\\file1.txt" @"C:\temp\file1.txt"
Константа null	Ссылка, которая не указывает ни на какой объект	null

Символы Escape-последовательности

Название	Описание
\a	Предупреждение (звонок)
\b	Возврат на одну позицию
\f	Переход на новую страницу
\n	Переход на новую строку
\r	Возврат каретки

Название	Описание
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\0	Ноль
\'	Одинарная кавычка
\"	Двойная кавычка
\\	Обратная косая черта

Управляющая последовательность (**escape**) - определенный символ, предваряемый обратной косой чертой. Эти литералы используется для создания эффектов (звонок), форматирования информации и кодирования символов. **Escape** - последовательность интерпретируется как одиночный символ и используется для представления:

Символьные литералы представляют собой любой символ в кодировке Unicode. Символьные константы записываются в одной из четырех форм:
- «обычный» символ, имеющий графическое представление (кроме - апострофа и символа перевода строки) - **'A', 'ю', '*';**
- управляющая последовательность - **'\0', '\n'**;
- символ в виде шестнадцатеричного кода - **'\xF', '\x74'**;
- символ в виде escape-последовательности Unicode - **'\uA81B'**.

Если требуется вывести несколько строк, можно объединить их в один литерал, отделив одну строку от другой символами \n:

```
string S = "\n **********\n ФИО: Иванов И.И.\nВозраст: 30 лет \n ********** "
```

Этот литерал при выводе будет выглядеть так:

```
**********
ФИО: Иванов И.И
Возраст: 30 лет
**********
```

Дословные строковые литералы (verbatim strings) - последовательность символов и символьных Escape-последовательностей, интерпретируемых компилятором так, как они записаны. Дословные литералы представляются при помощи символа @, который располагается непосредственно перед строковым литералом, заключенным в парные двойные кавычки.
Строковый литерал: **"c:\\My Doc\\sample.txt"** и *дословный литерал:* **@"c:\My Doc\sample.txt"** имеют одно значение: **c:\MyDoc\sample.txt**
Представление двойных кавычек внутри дословных литералов достигается за счет их дублирования: **@""""Focus""""** имеет значение **"Focus"**
Чаще всего дословные литералы применяются при задании полного пути файла. **Строка** может быть **пустой** (записывается парой смежных двойных кавычек ""), пустая **символьная константа недопустима**.

Программа на языке C# - *это текстовый файл (один или несколько), состоящий из* ***операторов*** *языка и* ***комментариев****.*

Операторы *могут содержать* ***зарезервированные*** *слова,* ***идентификаторы****,* ***константы*** *(литералы) и* ***выражения****. Отдельные части оператора отделяются друг от друга пробелами. Несколько следующих друг за другом пробелов считаются одним пробелом. Каждый оператор заканчивается точкой с запятой, что позволяет располагать в одной строке несколько операторов (или один длинный оператор на нескольких строках - в этом случае перенос части оператора на другую строку возможен в любом месте, где может располагаться пробел, кроме строковых выражений).*

Лабораторная работа №2

Написать программу для работы с БД "Студенты", выполняющую следующие действия:

Структура БД:

- № (номер записи - элемента)
- ФИО студента
- Дата рождения (день, месяц, год)
- Институт
- Группа
- Курс
- Средний балл.

Работа БД:

- добавление, изменение, удаление записи (элемента) БД
- прямая и обратная сортировка по полям: ФИО, Дата рождения
- поиск элемента по полям: ФИО, Дата рождения
- нахождение Max, Min, среднего значения и суммы по полю: Средний балл.

Разработать меню для функционирования программы (ввод/вывод данных в файл, ввод/вывод информации, выбор пунктов меню).

Данные БД хранятся в файле.