**Практическая работа № 2.**

**Программирование алгоритмов линейной структуры на языке c#**

**Цель работы**: изучить основные принципы построения программ на языке программирования С#, изучить порядок действий при вычислении выражений; приобрести навыки в записи выражений и использовании стандартных функций; овладеть практическими навыками в программировании линейных алгоритмов и отладке программ, тем самым закрепив полученные теоретические сведения.

|  |  |  |
| --- | --- | --- |
| №  п/п | Ф.И.О. | НОМЕР ВАРИАНТА |
|  | Балицкая Анастасия |  |
|  | Вартик Максим |  |
|  | Вдовиченко Ярослав |  |
|  | Веряскин Иван |  |
|  | Гоцалюк Арзу |  |
|  | Градинар Илья |  |
|  | Гребенщиков Данил |  |
|  | Завтонев Роман |  |
|  | Захаров Станислав |  |
|  | Иванов Богдан |  |
|  | Ильинский Никита |  |
|  | Каплин Егор |  |
|  | Коваленко Михаил |  |
|  | Колак Михаил |  |
|  | Левицкая Мария |  |
|  | Милюкова Екатерина |  |
|  | Никифорова Елизавета |  |
|  | Подолян Евгений |  |
|  | Попов Максим |  |
|  | Родичев Максим |  |
|  | Рознован Роман |  |
|  | Тизул Игорь |  |
|  | Тихончук София |  |
|  | Толоченко Павел |  |
|  | Фокша Кирилл |  |
|  | Чебан Евгений | 15 |
|  | Чебан Илья | 11 |
|  | Юраш Ренат | 4 |

**Теоретическая часть**

Если в программе все операторы выполняются последовательно, один за другим, такая программа называется *линейной*.

Программа на С# состоит из функций. *Функция* — это именованная последовательность операторов. Функция состоит из заголовка и тела. Основная функция программы должна иметь имя main, указывающее, что именно с нее требуется начинать выполнение. За именем функции в скобках обычно следует список передаваемых ей параметров. Перед именем записан тип значения, возвращаемого функцией в точку ее вызова.

* + 1. Состав языка

*Алфавит* – совокупность допустимых в языке символов. Алфавит языка С# включает:

1. прописные и строчные латинские буквы и буквы национальных алфавитов (включая кириллицу);
2. арабские цифры от 0 до 9, шестнадцатеричные цифры от A до F;
3. специальные знаки: " { } , | ; [ ] ( ) + - / % \* . \ ' : ? < = > ! & ~ ^ @ \_
4. пробельные символы: пробел, символ табуляции, символ перехода на новую строку.

Из символов алфавита 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).

*Идентификатор*– это имя программного элемента: константы, переменной, метки, типа, класса, объекта, метода и т.д. Идентификатор может включать латинские буквы и буквы национальных алфавитов, цифры и символ подчеркивания. Прописные и строчные буквы различаются, например, myname, myName и MyName — три различных имени. Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра.

Пробелы внутри имен не допускаются. Язык С# не налагает никаких ограничений на длину имен, однако для удобства чтения и записи кода не стоит делать их слишком длинными.

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.

*Ключевые слова*– зарезервированные идентификаторы, имеющие специальное значение для компилятора, например, main, int и т.д., их можно использовать только по прямому назначению. С ключевыми словами и их назначением можно ознакомиться в справочной системе С#.

* + 1. Типы данных

С# является языком со строгой типизацией. В нем необходимо объявлять тип всех создаваемых программных элементов (например, переменных и т. д.), что позволяет компилятору предотвращать возникновение ошибок, следя за тем, чтобы объектам присваивались значения только разрешенного типа. Тип программного элемента сообщает компилятору о его размере.

В С# типы делятся на две группы: *базовые*типы, предлагаемые языком, и типы, *определяемые пользователем*. Также типы С# разбиваются на две другие категории: *размерные типы*и *ссылочные типы.*Почти все базовые типы являются размерными типами. Исключение составляют типы Object и String. Принципиальное различие между размерными и ссылочными типами состоит в способе хранения их значений в памяти. В первом случае фактическое значение хранится в стеке. Адрес переменной ссылочного типа тоже хранится в стеке, но сам объект хранится в куче.

Основные типы переменных в С#

|  |  |  |
| --- | --- | --- |
| *Тип* | *Описание* | *Диапазон значений* |
| bool | логический | true и false |
| char | символьный | 0 … 65535 |
| string | строка | последовательность символов |
| sbyte | байт | -128 … 127 |
| byte | байт | 0 … 255 |
| short | короткое целое | -32768 … 32767 |
| ushort | короткое целое | 0 … 65535 |
| Int | целое | -2147483648 …2147483647 |
| Uint | целое | 0 … 4294967295 |
| long | длинное целое |  |
| ulong | длинное целое |  |
| float | вещественное | 3,4e-38 … 3,4e+38 |
| double | с повышенной точностью | 1,7e-308 … 1,7e+308 |

*Переменная* представляет собой типизированную область памяти. Программист создает переменную, объявляя ее тип и указывая имя. При объявлении переменной ее можно инициализировать (присвоить ей начальное значение), а затем в любой момент ей можно присвоить новое значение, которое заменит собой предыдущее.

*Константа*- это переменная, значение которой нельзя изменить. Константы бывают трех видов: *литералы, символические константы*и *перечисления.*

* + 1. Организация ввода-вывода данных

Программа при вводе данных и выводе результатов взаимодействует с внешними устройствами. Совокупность стандартных устройств ввода (клавиатура) и вывода (экран) называется *консолью*. В языке С# нет операторов ввода и вывода. Вместо них для обмена данными с внешними устройствами используются специальные объекты. В частности, для работы с консолью используется стандартный класс Console, определенный в пространстве имен System.

*Ввод данных*

Для ввода данных используется метод ReadLine, реализованный в классе Console. Особенностью данного метода является то, что в качестве результата он возвращает строку (string).

*Пример*:

static void Main()

{

string s = Console.ReadLine();

Console.WriteLine(s);

}

Для того чтобы получить числовое значение необходимо воспользоваться преобразованием данных.

*Пример*:

static void Main()

{

string s = Console.ReadLine();

int x = int.Parse(s); //преобразование строки в число

Console.WriteLine(x);

}

Или сокращенный вариант:

static void Main()

{

//преобразование введенной строки в число

int x = int.Parse(Console.ReadLine());

Console.WriteLine(x);

}

Преобразование типов данных осуществляется с помощью класса Convertили методаParse. Для преобразования строкового представления целого числа в типintиспользуется методint.Parse(), который реализован для всех числовых типов данных. Если требуется преобразовать строковое представление в вещественное, то используется методfloat.Parse()илиdouble.Parse(). В случае, если соответствующее преобразование выполнить невозможно, то выполнение программы прерывается и генерируется исключениеSystem.FormatExeption(входная строка имела неверный формат).

*Класс Convert*, определенный в пространстве имен System обеспечивает необходимые преобразования между различными типами. Методыкласса*Convert* поддерживают общий способ выполнения преобразований между типами. *Класс Convert* содержит 15 статических *методов* вида To <Type> (ToBoolean(),...ToUInt64()), где Type может принимать значения от Boolean до UInt64 для всех встроенных типов. Единственным *исключением* является тип object, - *метода* ToObject нет по понятным причинам, поскольку для всех типов существует *неявное преобразование* к типу object.

*Вывод данных*

Для вывода данных используется метод WriteLine, реализованный в классе Console, который позволяет организовывать вывод данных на экран. Однако существует несколько способов применения данного метода:

1. Console.WriteLine(x); //на экран выводится значение идентификатора х
2. Console.WriteLine("x=" + x +"y=" + y); /\* на экран выводится строка, образованная последовательным слиянием строки "x=", значения x, строки "у=" и значения у \*/
3. Console.WriteLine("x={0} y={1}", x, y); /\* на экран выводится строка, формат которой задан первым аргументом метода, при этом вместо параметра {0} выводится значение x, а вместо {1} – значение y\*/

*Использование управляющих последовательностей:*

*Управляющей последовательностью* называют определенный символ, предваряемый обратной косой чертой. Данная совокупность символов интерпретируется как одиночный символ и используется для представления кодов символов, не имеющих графического обозначения (например, символа перевода курсора на новую строку) или символов, имеющих специальное обозначение в символьных и строковых константах (например, апостроф).

Рассмотрим управляющие символы:

|  |  |
| --- | --- |
| *Вид* | *Наименование* |
| \a | Звуковой сигнал |
| \b | Возврат на шаг назад |
| \f | Перевод страницы |
| \n | Перевод строки |
| \r | Возврат каретки |
| \t | Горизонтальная табуляция |
| \v | Вертикальная табуляция |
| \\ | Обратная косая черта |
| \' | Апостроф |
| \" | Кавычки |

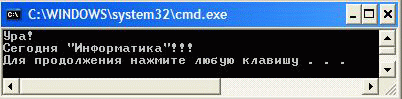
*Пример*:

static void Main()

{

Console.WriteLine("Ура!\nСегодня \"Информатика\"!!!");

}



*Управление размером поля вывода:*

Первым аргументом WriteLine указывается строка вида {n, m} – где n определяет номер идентификатора из списка аргументов метода WriteLine, а m – количество позиций (размер поля вывода), отводимых под значение данного идентификатора. При этом значение идентификатора выравнивается по правому краю. Если выделенных позиций для размещения значения идентификатора окажется недостаточно, то автоматически добавиться необходимое количество позиций.

*Пример*:

static void Main()

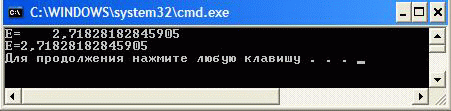
{

double x= Math.E;

Console.WriteLine("E={0,20}", x);

Console.WriteLine("E={0,10}", x);

}



*Управление размещением вещественных данных:*

Первым аргументом WriteLine указывается строка вида {n: ##.###} – где n определяет номер идентификатора из списка аргументов метода WriteLine, а ##.### определяет формат вывода вещественного числа. В данном случае под целую часть числа отводится две позиции, под дробную – три. Если выделенных позиций для размещения целой части значения идентификатора окажется недостаточно, то автоматически добавиться необходимое количество позиций.

*Пример*:

static void Main()

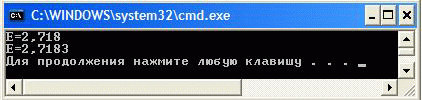
{

double x= Math.E;

Console.WriteLine("E={0:##.###}", x);

Console.WriteLine("E={0:.####}", x);

}



*Управление форматом числовых данных:*

Первым аргументом WriteLine указывается строка вида {n: <спецификатор>m} – где n определяет номер идентификатора из списка аргументов метода WriteLine, <спецификатор> - определяет формат данных, а m – количество позиций для дробной части значения идентификатора. В качестве спецификаторов могут использоваться следующие значения:

|  |  |  |
| --- | --- | --- |
| *Параметр* | *Формат* | *Значение* |
| C или c | Денежный. По умолчанию ставит знак р. Изменить его можно с помощь объекта NumberFormatInfo | Задается количество десятичных разрядов |
| D или d | Целочисленный (используется только с целыми числами) | Задается минималь-ное количество цифр. При необходимости результат дополняется начальными нулями |
| E или e | Экспоненциальное представление чисел | Задается количество символов после запятой. По умолчанию используется 6 |
| F или f | Представление чисел с фиксированной точкой | Задается количество символов после запятой |
| G или g | Общий формат (или экспоненциальный, или с фиксированной точкой) | Задается количество символов после запятой. По умолчанию выводится целая часть |
| N или n | Стандартное форматирование с использованием запятых и пробелов в качестве разделителей между разрядами | Задается количество символов после запятой. По умолчанию – 2, если число целое, то ставятся нули |
| X или x | Шестнадцатеричный формат |  |
| P или p | Процентный |  |

Порядок вычислений определяется приоритетом операций. Основные правила соответствуют принятым в математике. Операции языка С# (с учетом убывания приоритета):

**1. Инкремент (++) и декримент(--).**

Эти операции имеют две формы записи - *префиксную,*когда операция записывается перед операндом, и *постфиксную* - операция записывается после операнда. Префиксная операция инкремента (декремента) увеличивает (уменьшает) свой операнд и возвращает измененное значение как результат. Постфиксные версии инкремента и декремента возвращают первоначальное значение операнда, а затем изменяют его.

*Пример*:

|  |  |
| --- | --- |
| static void Main()  {  int i = 3, j = 4;  Console.WriteLine("{0} {1}", i, j);  Console.WriteLine("{0} {1}", ++i, --j);  Console.WriteLine("{0} {1}", i++, j--);  Console.WriteLine("{0} {1}", i, j);  } | *Результат работы программы:*  3 4  4 3  4 3  5 2 |

**2. Операция new.**Используется для создания нового объекта. С помощью ее можно создавать как объекты ссылочного типа, так и размерные, например:

object z=new object();

int i=new int(); // то же самое, что и int i =0;

3. Отрицание:

* Арифметическое отрицание (-) – меняет знак операнда на противоположный.
* Логическое отрицание (!) – определяет операцию инверсия для логического типа.

*Пример*:

|  |  |
| --- | --- |
| static void Main()  {  int i = 3, j=-4;  bool a = true, b=false;  Console.WriteLine("{0} {1}", -i, -j);  Console.WriteLine("{0} {1}", !a, !b);  } | *Результат работы программы:*  -3 4  False True |

**4. Явное преобразование типа.**Используется для явного преобразования из одного типа в другой. Формат операции:

(тип) выражение;

*Пример*:

|  |  |
| --- | --- |
| static void Main()  {  int i = -4;  byte j = 4;  int a = (int)j; //преобразование без потери точности  byte b = (byte)i; //преобразование с потерей точности  Console.WriteLine("{0} {1}", a, b);  } | *Результат работы программы:*  4 252 |

5. Умножение (\*), деление (/) и деление с остатком (%).

Операции умножения и деления применимы для целочисленных и вещественных типов данных. Для других типов эти операции применимы, если для них возможно неявное преобразование к целым или вещественным типам. При этом тип результата равен "наибольшему" из типов операндов, но не менее int. Если оба операнда при делении целочисленные, то и результат тоже целочисленный.

*Пример*:

|  |  |
| --- | --- |
| static void Main()  {  int i = 100, j = 15;  double a = 14.2, b = 3.5;  Console.WriteLine("{0} {1} {2}", i\*j, i/j, i%j);  Console.WriteLine("{0} {1} {2}", a \* b, a / b, a % b);  } | *Результат работы программы:*  1500 6 10  49.7 4.05714285714286 0.1999999999999999 |

**6. Сложение (+) и вычитание (-).** Операции сложения и вычитания применимы для целочисленных и вещественных типов данных. Для других типов эти операции применимы, если для них возможно неявное преобразование к целым или вещественным типам.

**7. Операции отношения ( <, <=, >, >=, ==, !=)**. Операции отношения сравнивают значения левого и правого операндов. Результат операции логического типа: true - если значения совпадают, false - в противном случае.

*Пример*:

|  |  |
| --- | --- |
| static void Main()  {  int i = 15, j = 15;  Console.WriteLine(i<j); //меньше  Console.WriteLine(i<=j); //меньше или равно  Console.WriteLine(i>j); //больше  Console.WriteLine(i>=j); //больше или равно  Console.WriteLine(i==j); //равно  Console.WriteLine(i!=j); //не равно  } | *Результат работы программы:*  False  True  False  True  True  False |

**8. Логические операции:** И (&&), ИЛИ (||).

Логические операции применяются к операндам логического типа. Результат логической операции И имеет значение истина тогда и только тогда, когда оба операнда принимают значение истина. Результат логической операции ИЛИ имеет значение истина тогда и только тогда, когда хотя бы один из операндов принимает значение истина.

*Пример*:

static void Main()

{

Console.WriteLine("x y x и y x или y");

Console.WriteLine("{0} {1} {2} {3}", false, false, false&&false, false||false);

Console.WriteLine("{0} {1} {2} {3}", false, true, false&&true, false||true);

Console.WriteLine("{0} {1} {2} {3}", true, false, true&&false, true||false);

Console.WriteLine("{0} {1} {2} {3}", true, true, true&&true, true||true);

}

*Результат работы программы:*

x y x и y x или y

False False False False

False True False True

True False False True

True True True True

**9. Операции присваивания:** =, +=, -= и т.д.

Формат операции *простого присваивания* (=): **операнд\_2 = операнд\_1;**

В результате выполнения этой операции вычисляется значение операнда\_1, и результат записывается в операнд\_2. Выражение вида a=b=c=100 выполняется справа налево: результатом выполнения c=100 является число 100, которое затем присваивается переменной b, результатом чего опять является 100, которое присваивается переменной a.

Кроме простой операции присваивания существуют *сложные операции присваивания*, например, умножение с присваиванием (\*=), деление с присваиванием (/=), остаток от деления с присваиванием (%=), сложение с присваиванием (+=), вычитание с присваиванием (-=) и т.д.

В сложных операциях присваивания, например, при *сложении с присваиванием*, к операнду\_2 прибавляется операнд\_1, и результат записывается в операнд\_2. То есть, выражение с += а является более компактной записью выражения с = с + а.

Если в одном выражении соседствуют операции одного приоритета, то операции присваивания и условная операции выполняются справа налево, а остальные наоборот. Если необходимо изменить порядок выполнения операций, то в выражении необходимо поставить круглые скобки.

* + 1. Выражения и преобразование типов

*Выражение* - это синтаксическая единица языка, определяющая способ вычисления некоторого значения. Выражения состоят из операндов, операций и скобок. Каждый операнд является в свою очередь выражением или одним из его частных случаев - константой, переменной или функций.

В языке С# предусмотрено вычисление различных математических функций. Для их использования необходимо обратиться к классу Math, который содержит стандартные математические функции. Этот класс содержит два статических поля, задающих константы E и PI, а также 23 статических метода.

Методы задают:

* тригонометрические функции - Sin, Cos, Tan;
* обратные тригонометрические функции - ASin, ACos, ATan, ATan2 (sinx, cosx);
* гиперболические функции - Tanh, Sinh, Cosh;
* экспоненту и логарифмические функции - Exp, Log, Log10;
* модуль, корень, знак - Abs, Sqrt, Sign;
* функции округления - Ceiling, Floor, Round;
* минимум, максимум, степень, остаток - Min, Max, Pow, IEEERemainder.

*Пример*:Написать программу для расчета функции https://studfile.net/html/2706/27/html_qEXQ_J0uL0.ErTg/img-5_hJra.png.

static void Main(string[] args)

{

double x, y;

Console.WriteLine("Введите значение х:");

x = Convert.ToDouble(Console.ReadLine());

y = Math.Sqrt((x+3)/(x-3));

Console.WriteLine("Результат: {0}", y);

}

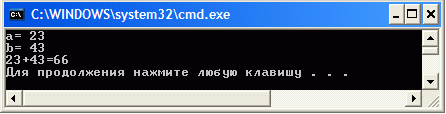
Вопросы для самоконтроля

1. Какая программа называется линейной?
2. Дайте определение термину «функция».
3. Из каких частей состоит функция?
4. Почему необходимо указывать тип используемой переменной при ее описании?
5. Как следует выбирать тип переменных?
6. Какие операции определены в С#?
7. Чем характеризуются переменные?
8. Какие функции ввода-вывода существуют в С#?
9. Каким классом необходимо воспользоваться для реализации математических выражений?
10. Каких математических функций нет в С#?
11. Как определяется порядок вычислений в С#?
12. Для чего используется оператор присваивания?

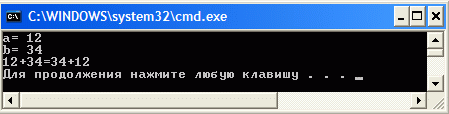
Практическая часть

**Задание №1.** Написать программу, которая реализует диалог с пользователем:

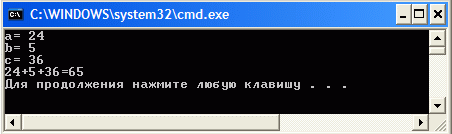
1. запрашивает с клавиатуры два целых числа, и выводит на экран сумму данных чисел:



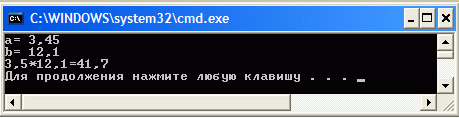
1. запрашивает с клавиатуры три целых числа, и выводит на экран сумму данных чисел в прямом и обратном порядке:



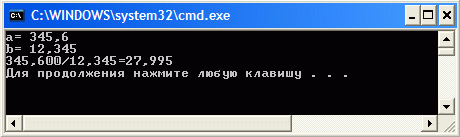
1. запрашивает с клавиатуры три целых числа, и выводит на экран сумму данных чисел:



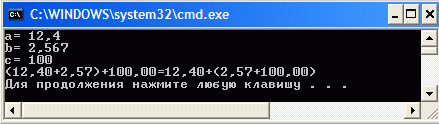
1. запрашивает с клавиатуры два вещественных числа, и выводит на экран произведение данных чисел (вещественные числа выводятся с точностью до 1 знака после запятой):



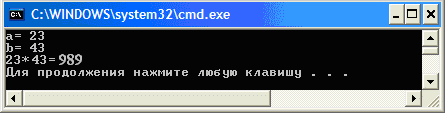
1. запрашивает с клавиатуры два вещественных числа, и выводит на экран результат деления первого числа на второе (вещественные числа выводятся с точностью до 3 знаков после запятой):



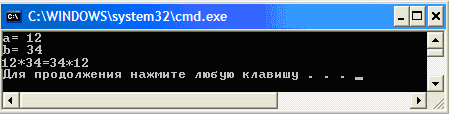
1. запрашивает с клавиатуры три вещественных числа, и выводит на следующее сообщение (вещественные числа выводятся с точностью до 2 знаков после запятой):



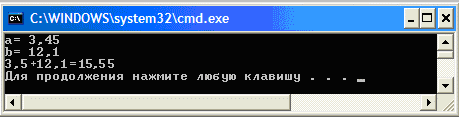
1. запрашивает с клавиатуры два целых числа, и выводит на экран произведение данных чисел:



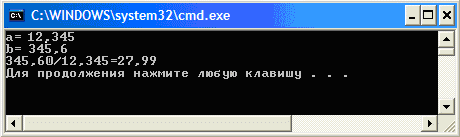
1. запрашивает с клавиатуры три целых числа, и выводит на экран произведение данных чисел в прямом и обратном порядке:



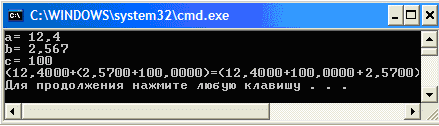
1. запрашивает с клавиатуры два вещественных числа, и выводит на экран сумму данных чисел (вещественные числа выводятся с точностью до 2 знаков после запятой):



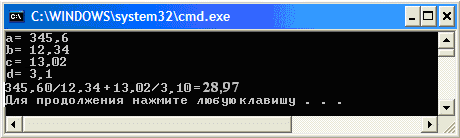
1. запрашивает с клавиатуры два вещественных числа, и выводит на экран результат деления второго числа на первое (вещественные числа выводятся с точностью до 2 знаков после запятой):



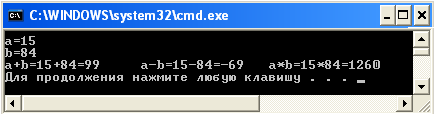
1. запрашивает с клавиатуры три вещественных числа и выводит следующее сообщение (*a+(b+c))=(a+c+b)*(вещественные числа выводятся с точностью до 4 знаков после запятой):



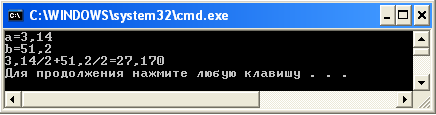
1. запрашивает с клавиатуры четыре вещественных числа, и выводит на экран результат деления первого числа на второе плюс третьего на четвертое (вещественные числа выводятся с точностью до 2 знаков после запятой):



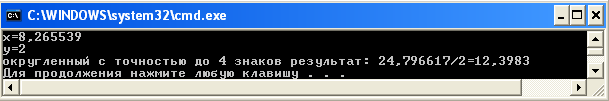
1. запрашивает с клавиатуры два целых числа, и выводит на экран результат их суммы, разности и произведения:



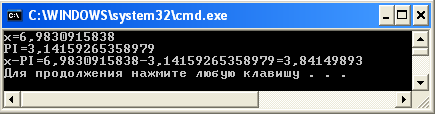
1. переменной вещественного типа *х* присваивает значение, равное полусумме значений а и b с точностью 3 знака после запятой:



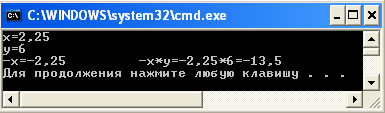
1. утраивает значение вещественной переменной *х* и выполняет деление на целое число *y*, результат округлить до 4 знаков после запятой:



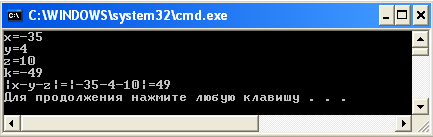
1. переменной вещественного типа *х* присвоить значение на https://studfile.net/html/2706/27/html_qEXQ_J0uL0.ErTg/img-3DO4kE.pngменьшее исходного, результат округлить до 8 знаков после запятой:



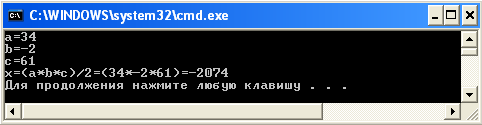
1. меняет знак у переменной *х* на противоположный и выполняет умножение на вещественное число *y*:



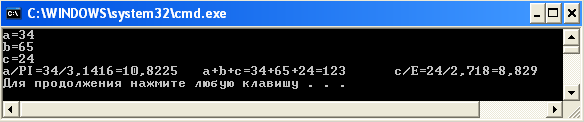
1. запрашивает с клавиатуры три целых числа, и выводит на экран результат модуля разности первого, второго и третьего чисел:



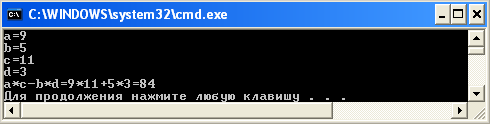
1. переменной целого типа *х* присвоить значение, равное половине произведения значений а, b, с:



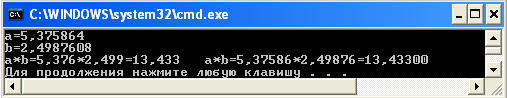
1. запрашивает с клавиатуры три целых числа, и выводит на экран результат деления первого числа на PI (с точностью 4 знака после запятой), сумму данных чисел и результат деления третьего числа на Е (с точностью 3 знака после запятой):



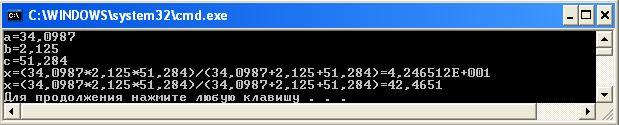
1. запрашивает с клавиатуры четыре целых числа, и выводит на экран результат умножения первого числа на третье минус произведение второго числа на четвертое:



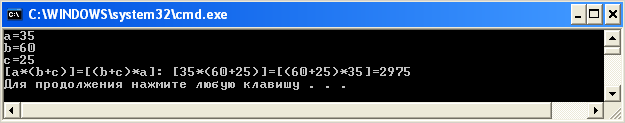
1. запрашивает с клавиатуры два вещественных числа, и выводит на экран результат их умножения друг на друга в два столбика: первый столбик – вывод результата с точностью 3 знака после запятой, второй столбик – с точностью 5 знаков после запятой:



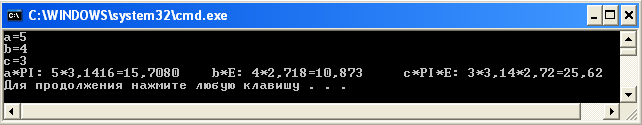
1. переменной вещественного типа *х* присваивает значение произведения чисел *а*, *b* и *с* деленных на их сумму; результат вывести в формате с плавающей и фиксированной точкой с точностью 4 знака после запятой:



1. запрашивает с клавиатуры три целых числа и выводит следующее сообщение [*a\**(*b+c*)]*=*[(*b+c*)*\*a*]:



1. запрашивает с клавиатуры три целых числа, и выводит на экран результат умножения первого числа на PI (с точностью 4 знака после запятой), второго на число Е (с точностью 3 знака после запятой) и третьего числа на произведение числа PI на Е (с точностью 2 знака после запятой):



**Задание №2.** Написать программу:

1. Дано двузначное число. Найти число десятков в нем.
2. Дано двузначное число. Найти число единиц в нем.
3. Дано двузначное число. Найти сумму его цифр.
4. Дано двузначное число. Найти произведение его цифр.
5. Дано трехзначное число. Найти число десятков в нем.
6. Дано трехзначное число. Найти число единиц в нем.
7. Дано трехзначное число. Найти сумму его цифр.
8. Дано трехзначное число. Найти произведение его цифр.
9. Дано четырехзначное число. Найти сумму его цифр.
10. Дано четырехзначное число. Найти произведение его цифр.
11. Дано трехзначное число. Найти число, полученное при прочтении его цифр справа налево.
12. Дано трехзначное число. В нем зачеркнули первую слева цифру и приписали ее в конце. Найти полученное число.
13. Дано трехзначное число. В нем зачеркнули последнюю справа цифру и приписали ее в начале. Найти полученное число.
14. Дано трехзначное число. Найти число, полученное при перестановке первой и второй цифр заданного числа.
15. Дано трехзначное число. Найти число, полученное при перестановке второй и третьей цифр заданного числа.
16. Дано трехзначное число, в котором все цифры различны. Получить шесть чисел, образованных при перестановке цифр заданного числа.
17. Дано четырехзначное число. Найти число, полученное при прочтении его цифр справа налево.
18. Дано четырехзначное число. Найти число, образуемое при перестановке первой и второй, третьей и четвертой цифр заданного числа.
19. Дано четырехзначное число. Найти число, образуемое при перестановке второй и третьей цифр заданного числа.
20. Дано четырехзначное число. Найти число, образуемое при перестановке двух первых и двух последних цифр заданного числа с выделением отдельных цифр заданного числа.
21. Дано четырехзначное число. Найти число, образуемое при перестановке двух первых и двух последних цифр заданного числа без выделения отдельных цифр заданного числа.
22. Дано четырехзначное число. Найти число, образуемое при перестановке первой и последней цифр.
23. Дано пятизначное число. Найти число, образуемое при перестановке первой и третьей, второй и четвертой цифр.
24. Дано пятизначное число. Найти число сотен и тысяч в нем.
25. Дано пятизначное число. Найти число, образуемое при перестановке первой и четвертой, второй и пятой цифр.

**Задание №3.** Написать программу, которая подсчитывает:

* 1. периметр квадрата, площадь которого равна а;
  2. площадь равностороннего треугольника, периметр которого равен p;
  3. расстояние между точками с координатами a, b и с, d;
  4. среднее арифметическое кубов двух данных чисел;
  5. среднее геометрическое модулей двух данных чисел;
  6. гипотенузу прямоугольного треугольника по двум данным катетам a, b.
  7. площадь прямоугольного треугольника по двум катетам a, b.
  8. периметр прямоугольного треугольника по двум катетам a, b.
  9. ребро куба, площадь полной поверхности которого равна s;
  10. ребро куба, объем которого равен v;
  11. периметр треугольника, заданного координатами вершин x1, y1, x2, y2, x3, y3;
  12. площадь треугольника, заданного координатами вершин x1, y1, x2, y2, x3, y3;
  13. радиус окружности, длина которой равна l;
  14. радиус окружности, площадь круга которой равна s;
  15. площадь равнобедренной трапеции с основаниями a и b и углом https://studfile.net/html/2706/27/html_qEXQ_J0uL0.ErTg/img-Zhte4S.pngпри большем основании;
  16. площадь кольца с внутренним радиусом *r*1 и внешним *r*2;
  17. радиус окружности, вписанной в равносторонний треугольник со стороной *а*;
  18. радиус окружности, описанной около равностороннего треугольника со стороной *а*;
  19. площадь треугольника, две стороны которого равны *а* и *b*, а угол между этими сторонами равен у. Считайте, что у - это радианная мера угла
  20. площадь треугольника, две стороны которого равны *а* и *b*, а угол между этими сторонами равен у. Считайте, что у - это градусная мера угла.
  21. расстояние между двумя точками на плоскости с данными координатами (*х*1, *у*1) и (*х*2, *у*2)
  22. длину окружности, площадь круга, объем шара заданного радиуса.
  23. площадь квадрата со стороной *а* (*а* - положительное число).
  24. сумму членов арифметической прогрессии, если известен ее первый член, разность и число членов прогрессии.
  25. сумму членов геометрической прогрессии, если известен ее первый член, знаменатель и число членов прогрессии.

# Контрольные вопросы

1. Какова общая структура программы на С#.
2. Существуют ли ограничения на множество целых чисел, используемых в языке С#?
3. Какие формы записи вещественных чисел используются в языке С#?
4. Из каких символов может состоять идентификатор?
5. Чем вызвана необходимость использования комментариев в программе?
6. Каково назначение методов write и writeline?
7. Перечислите способы вывода данных на дисплей?
8. Перечислите основные управляющие последовательности языка С#?
9. Назовите основные методы класса Math языка С#?
10. Какой метод класса Console используется для того, чтобы вводимые данные читались с новой строки?
11. Что такое выражения в языке С#?
12. Понятие преобразования типов в языке С#?
13. Назовите формы записи операцииинкремента и декремента языка С#?