Главное управление образования Гомельского облисполкома

Учреждение образования

«Гомельский государственный машиностроительный колледж»

Учебная дисциплина

«Конструирование программ и языки программирования»

**Инструкция**

по выполнению лабораторной работы №2

«Разработка линейных программ»

Гомель 2018

Составитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гаврилова В.М.

Обсуждено и одобрено на заседании цикловой комиссии «Программное обеспечение информационных технологий»

Протокол от «01» сентября 2018 № 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Ю. Лавская

**Лабораторная работа №2**

**Тема работы: «Разработка линейных программ»**

1. **Цель работы**

Сформировать умения разрабатывать линейные программы с объявлением переменных и констант, строить выражения на основе приоритета операций.

1. **Задание**
2. Разработать линейную программу согласно варианту по списку в журнале

|  |  |  |
| --- | --- | --- |
| Вариант | Вычислить | Расчетные формулы |
| 1 | Площадь прямоугольного треугольника с гипотенузой *с* и одним из катетов *a* |  |
| 2 | Периметр и площадь прямоугольного треугольника с катетами *a, b* |  |
| 3 | Высоту и площадь равнобедренного треугольника с основанием *a* и углом при основании |  |
| 4 | Радиус круга, описанного вокруг треугольника со сторонами *a, b, c* |  |
| 5 | Периметр и площадь параллелограмма со сторонами *a, b* и острым углом *α* | *P =* 2(*a+b*) |
| 6 | Площадь прямоугольной трапеции с основаниями *a*, *b* (*b>a*) и углом при большем основании α |  |
| 7 | Сопротивление проводника длиной l, площадью поперечного сечения *S* и удельным сопротивлением ρ |  |
| 8 | Расстояние между точками с координатами *x*1*, y*1 и *x*2*, y*2 |  |
| 9 | Путь, пройденный за время *t* со скоростью *v* | *S = vt* |
| 10 | Емкость плоского конденсатора C с площадью поверхности одной пластины *S*, расстоянием между пластинами *d* и диэлектрической проницаемостью материала ε |  |
| 11 | Площадь круга и длину окружности радиуса *r* |  |
| 12 | Площадь и угол при основании равнобедренного треугольника с основанием *a* и высотой *h* |  |
| 13 | Площадь и периметр прямоугольника со сторонами *a, b* | *S = abP =* 2(*a+b*) |
| 14 | Скорость в конце пути и путь, пройденный за время *t*с ускорением *a* при *v*0 *=* 0 | *v = at* |
| 15 | Площадь и периметр квадрата со стороной *а* | *S = a*2 *P =* 4*a* |
| 16 | Объем и площадь боковой поверхности параллелепипеда со сторонами *а, b, c* | *V = abc S =* 2(*a+b*)*с* |
| 17 | Площадь кольца с внешним радиусом *R* и внутренним *r* | *S =* π(*R*2*– r*2) |
| 18 | Площадь боковой поверхности и объем цилиндра с радиусом основания *r* и высотой *h* | *S =* 2π*rhV =* π*r*2*h* |
| 19 | Площадь и периметр прямоугольного треугольника с катетами *a, b*игипотенузой*с* | *P=a+b+c* |
| 20 | Объем и площадь поверхности куба со стороной *а* | *V = a*3 *S =* 6*a*2 |
| 21 | Периметр и площадь треугольника со сторонами *a, b, c* | *P = a+b+c =* 2*p* |
| 22 | Площадь основания и объем цилиндра с радиусом основания *r* и высотой *h* | *S =* π*r 2 V=Sh* |
| 23 | Объем и площадь основания параллелепипеда со сторонами *а, b, c* | *V = abc S = ab* |
| 24 | Площадь основания и объем конуса с радиусом основания *r* и высотой *h* | *S =* π*r* 2 |
| 25 | Гипотенузу и площадь прямоугольного треугольника с катетами *a, b* |  |

**2.** Разработать линейную программу для вычисления значения выражения по формулам

| Вариант | Вид функции | Вариант | Вид функции |
| --- | --- | --- | --- |
| 1 |  | 14 |  |
| 2 |  | 15 |  |
| 3 |  | 16 |  |
| 4 |  | 17 |  |
| 5 |  | 18 |  |
| 6 |  | 19 |  |
| 7 |  | 20 |  |
| 8 |  | 21 |  |
| 9 |  | 22 |  |
| 10 |  | 23 |  |
| 11 |  | 24 |  |
| 12 |  | 25 |  |
| 13 |  |  |  |

1. **Оснащение работы**

Технические средства обучения:

* IBM – совместимый компьютер;

Электронные средства обучения:

* MSVisualStudio;
* MSOffice.

1. **Основные теоретические сведения**

***Линейной***называется программа, все операторы которой выполняются последовательно в том порядке, в котором они записаны. Простейшим примером линейной программы является программа расчета по заданной формуле. Она состоит из трех этапов: ввод исходных данных, вычисление по формуле и вывод результатов.

*Переменная* - именованный участок памяти для хранения данных определенного типа. Значение переменной (информация в соответствующих ячейках памяти) в ходе выполнения программы может быть изменено.

*Константами* называются величины, значение которых в ходе выполнения программы изменено быть не может. Конкретные переменные и константы представляют собой объекты уникальные и отличаются друг от друга именем.

*Тип переменной* указывает на то, какие данные могут быть сохранены в этом участке памяти, и в каких действиях эта переменная может участвовать.

Любой встроенный тип языка С# соответствует стандартному классу библиотеки .NET.

*Встроенные типы языка С#*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Ключевое слово** | **Тип .NET** | **Диапазон** | **Описание** | **Размер, битов** |
| Логический тип | bool | Boolean | true, false |  |  |
| Целые типы | sbyte | SByte | -128..127 | Со знаком | 8 |
| byte | Byte | 0..255 | Без знака | 8 |
| short | Int16 | -32768..32767 | Со знаком | 16 |
| ushort | UInt16 | 0..65535 | Без знака | 16 |
| int | Int32 | -2\*10^9..2\*10^9 | Со знаком | 32 |
| uint | UInt32 | 0..4\*10^9 | Без знака | 32 |
| long | Int64 | -9\*10^18..9\*10^18 | Со знаком | 64 |
| ulong | UInt64 | 0..18\*10^18 | Без знака | 64 |
| Символьный тип | char | Char | U+0000..U+ffff | Unicode-символ | 16 |
| Вещественные типы | float | Single | 1.5\*10^-45..3.4\*10^38 | 7 цифр | 32 |
| double | Double | 5.0\*10^-324..1.7\*10^308 | 15-16 цифр | 64 |
| Финансовый тип | decimal | Decimal | 1.0\*10^-28..7.9\*10^28 | 28-29 цифр | 128 |
| Строковый тип | string | String | Длина ограничена объемом доступной памяти | Строка из Unicode-символов |  |
| Тип object | object | Object | Можно хранить все что угодно | Всеобщий порядок |  |

Пример описания целой переменной с именем а и вещественной переменной х:

inta; floatx;

*Идентификатор (*имяпеременной) служит для обращения к области памяти, в которой хранится *значение* переменной. Имя дает программист. Оно должно соответствовать пра­вилам именования идентификаторов С#, отражать смысл хранимой величины и быть легко распознаваемым.

*Тип переменной выбирается, исходя из диапазона и требуемой точности представления данных.*

При объявлении можно присвоить переменной некоторое начальное значение, то есть *инициализировать* ее.

Так называемая *область действия* переменной, то есть область программы, где можно использовать переменную, начинается в точке ее описания и длится до конца блока, внутри которого она описана.

*Блок —* это код, заключенный в фигурные скобки. Основное назначение блока — группировка операторов. В С# любая переменная описана внутри какого-либо блока: класса, метода или блока внутри метода.

Имя переменной должно быть уникальным в области ее действия.

**Пример описания локальных переменных**

usingSystem;

namespaceConsoleApplicationl

{classClassl

{ static void Main()

{

inti = 3;

doubleу = 4.12;

decimal d = 600m;

string s = "Вася";

Console.Write( "i = " ); Console.WriteLine( i );

Console.Write( "y = " ); Console.WriteLine(у );

Console.Write( "d = " ); Console.WriteLine( d );

Console.Write( "s = " ); Console.WriteLine( s ); } } }

Переменные создаются при входе в их область действия (блок) и уничтожаются при выходе. Это означает, что после выхода из блока значение переменной не сохра­няется. При повторном входе в этот же блок переменная создается заново.

**Именованные константы**

Можно запретить изменять значение переменной, задав при ее описании ключе­вое слово const, например:

constintb = 1;

constfloatx = 0.1, у = 0.1f; // const распространяется на обе переменные

Такие величины называют *именованными константами,* или просто ***константами.***Они применяются для того, чтобы вместо значений констант можно было использовать в программе их имена. Это делает программу более понятной и об­легчает внесение в нее изменений, поскольку изменить значение достаточно только в одном месте программы.

## Основная структура программы

Для консольного приложения автоматически создается структура:

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespace ConsoleApplication2

{

classProgram

{

staticvoid Main(string[] args)

{

// здесь пишется текст программы

}

}

}

Имя программы выбирается программистом самостоятельно в соответствии с правилами построения идентификаторов и совпадает с именем файла на диске.

## Комментарии

Помимо отступов, большие логически замкнутые блоки программы удобно разделять строками-комментариями, содержащими информацию о смысле последующего блока.

// Двойная косая черта до конца строки

/\* многострочный

комментарий \*/

## Операторы ввода

Для того чтобы получить данные, вводимые пользователем вручную (то есть с консоли), применяются команды

<строковая переменная>=Console.ReadLine()

Для того чтобы преобразовать к нужному типу данных используем объект Convert:

<переменная целого типа> = Convert.ToInt32(<строковая переменная>);

<переменная действ типа> = Convert.ToDouble(<строковая переменная>);

<переменная логического типа> = Convert.ToBoolean(<строковая переменная>);

<переменная целого типа> = Convert.ToByte(<строковая переменная>);

<переменная символьного типа> = Convert.ToChar(<строковая переменная>);

<переменная целого типа> = Convert.ToInt64(<строковая переменная>);

***Пример***

Int32 x; Double y; string s;

s = Console.ReadLine();

x = Convert.ToInt32(s);

//без промежуточной строковой переменной

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

Типы вводимых значений должны совпадать с типами указанных переменных, иначе возникает ошибка.

Для того, чтобы обеспечить ожидание ввода любой клавиши в конце программы, используют

Console.ReadKey();

## Операторы вывода

Для того чтобы вывести на экран какое-либо сообщение, воспользуйтесь процедурой

Console.WriteилиConsole.WriteLine.

Первая из них, напечатав на экране все, о чем ее просили, оставит курсор в конце выведенной строки, а вторая переведет его в начало следующей строчки.

Console.WriteLine(s);// переменная

Console.WriteLine(55.3);// константа

Console.WriteLine(y\*3+7);// выражение

Console.Write(z);// переменная

Console.Write(-5.3);// константа

Console.Write(i\*3+7/j);// выражение

Чтобы при выводе данных они не склеивались, нужно позаботиться о пробелах между выводимыми переменными, также можно выводить списком несколько переменных:

Console.WriteLine("Это число А={0} далее B={1} и, наконец их сумма {2}", a, b, a + b);

Для форматирования числовых результатов можно использовать метод String.Format или метод Console.Write, вызывающий метод String.Format. Формат задается с помощью строк формата. Спецификация формата следующая: **{N,M:Axx}**, где N указывает позицию элемента в списке выводимых переменных (нумерация начинается с 0); M - задаёт ширину области, в которую будет помещено форматированное значение, если M отсутствует или отрицательно, значение будет выровнено влево, в противном случае - вправо; Axx - является необязательной строкой форматирующих кодов, которые используются для управениея форматированием чисел, даты и времени, денежных знаков и т.д.

***Пример простейшей программы***

string s1;

Console.Write("Введитевашеимя ");

s1 = Console.ReadLine();

Console.WriteLine("Мы рады вас приветствовать, {0:g}, в нашей программе",s1);

Console.ReadKey();

Во время работы этой программы на экране появится примерно следующее:

Введите ваше имя Джон

Мы рады вас приветствовать, Джон, в нашей программе

Самым простым действием над переменной является занесение в нее величины соответствующего типа. Иногда говорят об этом, как о присвоении переменной конкретного значения. Такая команда (оператор) в общем виде выглядит следующим образом:

**<Имя переменной> = <Выражение>;**

Тип переменной и тип выражения должны быть совместимы. Выражение, указанное справа от знака "=", должно приводить к значению того же типа, какого и сама переменная, или типа, совместимого с переменной относительно команды присваивания. Например, переменной типа double можно присвоить значение типа Int или byte (впрочем, наоборот делать нельзя). Выражение будет сначала вычислено, затем, его результат будет положен в ячейки памяти, отведенные для переменной.

*Выражение* — это правило вычисления значения. В выражении участвуют *опе­ранды,* объединенные *знаками операций.* Операндами простейшего выражения могут быть константы, переменные и вызовы функций.

Например

A = X; // переменная

B =15; // целая константа

D = Math.Sin(X); // вызов функции

E = X \* Y; // произведение X на Y

F = Z / (1 - Z); // отношение Z к (1 - Z)

G =(X == 1.5); // логическое выражение

H = ! J; // отрицание логической переменной

Также можно делать множественное присваивание (во все переменные заносится самое правое значение):

a = b = c = d = 18;

Пробелы внутри знака операции, состоящей из нескольких сим­волов, не допускаются. По количеству участвующих в одной операции опе­рандов операции делятся на *унарные, бинарные* и *тернарную.* Операции С# приведены в таблице.

Таблица - Операции С#

|  |  |  |
| --- | --- | --- |
| **Категория** | **Знак операции** | **Название** |
| **Первичные** | **.** | Доступ к элементу |
| **X()** | Вызов метода или делегата |
| **X[ ]** | Доступ к элементу |
| **X++** | Постфиксный инкремент |
| **new** | Выделение памяти |
| **X--** | Постфиксный декремент |
| **typeof** | Получение типа |
| **checked** | Проверяемый код |
| **unchecked** | Непроверяемый код |
| **Унарные** | **+** | Унарный плюс |
| **-** | Унарный минус (арифметическое отрицание) |
| **!** | Логическое отрицание |
| **~** | Поразрядное отрицание |
| **++X** | Префиксный инкремент |
| **--X** | Префиксный декремент |
| **(тип)x** | Преобразование типа |
| **Мультипликативные**  **(типа умножения)** | **\*** | Умножение |
| **/** | Деление |
| **%** | Остаток отделения |
| **Аддитивные**  **(типа сложения)** | **+** | Сложение |
| **-** | Вычитание |
| **Сдвига** | **<<** | Сдвиг влево |
| **>>** | Сдвиг вправо |
| **Отношения и проверки типа** | **<** | Меньше |
| **>** | Больше |
| **<=** | Меньше или равно |
| **>=** | Больше или равно |
| **is** | Проверка на принадлежность типу |
| **as** | Приведение типа |
| **Проверка на равенство** | **= =** | Равно |
| **!=** | Не равно |
| **Поразрядные логические** | **&** | Поразрядное И (конъюнкция) |
| **^** | Поразрядное исключающее ИЛИ |
| **|** | Поразрядное ИЛИ (дизъюнкция) |
| **Условные логические** | **&&** | Логическое И |
| **||** | Логическое ИЛИ |
| **Тернарная операция** | **?:** | Условная операция |

Операции в выражении выполняются в определенном порядке в соответствии с *приоритетами,* как и в математике. В приведенной выше таблице операции расположены по убыванию приоритетов.

**Инкремент и декремент**

Операции инкремента (++) и декремента (--), называемые также операциями увеличения и уменьшения на единицу, имеют две формы записи — *префиксную,* когда знак операции записывается перед операндом, и *постфиксную.* В префиксной форме сначала изменяется операнд, а затем его значение становится результирующим значением выражения, а в постфиксной форме значением выражения является исходное значение операнда, после чего он изменяется.

***Пример операций инкремента и декремента***

usingSystem;

namespace ConsoleApplication1

{ classProgram

{ static void Main()

{intx = 3, у = 3;

Console.Write( "Значение префиксного выражения: "');

Console.WriteLine( ++x );

Console,Write( "Значение*х*послеприращения: " );

Console.WriteLine(x);

Console.Write( "Значение постфиксного выражения: " ).;

Console.WriteLine( y++ );

Console.Write("Значение у после приращения: " );

Console.WriteLine(у );} } }

*Результат работы программы:*

*Значение префиксного выражения: 4*

*Значение х после приращения: 4*

*Значение постфиксного выражения: 3*

*Значение у после приращения: 4*

***Операция остатка от деления (%)***также интерпретируется по-разному для целых, вещественных и финансовых величин. Если оба операнда целочисленные, результат операции вычисляется по формуле х - (х / у) \* у. Если делитель равен нулю, генерируется исключение System. DivideByZeroException. Тип результата oneрации равен «наибольшему» из типов операндов, но не менее int. Если хотя бы один из операндов вещественный, результат операции вычисляется по формуле х - n \* у, где n — наибольшее целое, меньшее или равное результату деления х на у.

Для финансовых величин (тип decimal) при получении остатка от деления на 0 и при переполнении генерируются соответствующие исключения, при исчезновении порядка результат равен 0. Знак результата равен знаку первого операнда.

***Пример операций умножения, деления и получения остатка***

using System;

namespace ConsoleApplication1

{ classClassl

{ static void Main()

{int x = 11, у = 4;

float z = 4;

Console.WriteLine( z \*y ); // Результат 16

Console.WriteLine (z \* le308 ); // Результат "бесконечность"

Console.WriteLine( x / у ); // Результат 2

Console.WriteLine( x / z ); // Результат 2,75

Console.WriteLine( x*%* у ); // Результат 3

Console.WriteLine(le-324 / le-324 ); // Результат NaN – результат не число

} } }

**Тернарная (условная) операция**

Условная операция (? :) — тернарная, то есть имеет три операнда. Ее формат:

операнд1 ? операнд2 : операнд3

***Пример применения тернарной операции***

usingSystem;

namespaceConsoleApplicationl

{ classClassl

{ static void Main()

{ int a = 11, b = 4;

int max = b > a ? b : a;

Console.WriteLine(max ); // Результат 11

} } }

*Другой пример применения условной операции:* требуется, чтобы некоторая целая величина увеличивалась на 1, если ее значение не превышает n, а иначе принимала значение 1. Это удобно реализовать следующим образом:

i = (i<n) ?i + 1: 1;

**Операции присваивания**

***Операции присваивания*(=, +=, -=, \*=**и т. д.) задают новое значение переменной. Эти операции могут использоваться в программе как законченные операторы. Механизм выполнения операции присваивания такой: вычисляется выражение и его результат заносится в память по адресу, который определяется именем переменной, находящейся слева от знака операции. То, что ранее хранилось в этой области памяти, естественно, теряется.

***Примеры операторов присваивания:***

а = b + с / 2;

b = а;

а = b;

х = 1;

х+ = 0.5;

Обратите внимание: b=а и а=b— это совершенно разные действия!

Присваивание — это передача данных «налево».

Результатом операции присваивания является значение, записанное в левый операнд. Тип результата совпадает с типом левого операнда.

***В сложных операциях присваивания*( +=, \*=,** /= и т. п.) при вычислении выражения, стоящего в правой части, используется значение из левой части. Например, при *сложении с присваиванием* ко второму операнду прибавляется первый, и результат записывается в первый операнд, то есть выражение а *+=* b является более компактной записью выражения а = а + b.

Результатом операции сложного присваивания является значение, записанное в левый операнд.

Операции присваивания выполняются справа налево, в отличие от большинства других операций (а = b = с означает а = (b=c)).

**Математические функции – класс Math**

Все математические функции реализованы как методы класса Math, основные из которых представлены в таблице.

Таблица - Основные поля и методы класса Math

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Результат** | **Пояснения** |
| Abs | Модуль | перегружен | |x| записывается как Abs(x) |
| Acos | Арккосинус | double | Acos(double x) |
| Asin | Арксинус | double | Asin(double x) |
| Atan | Арктангенс | double | Atan(double x) |
| Atan2 | Арктангенс | double | Atan2(double x, doubley)— угол, тангенс которого есть результат деления y на x |
| BigMul | Произведение | long | BigMul(int x, int y) |
| Ceiling | Округление до большего целого | double | Ceiling(double х) |
| Cos | Косинус | double | Сos(double x) |
| Cosh | Гиперболический косинус | double | Cosh(double x) |
| DivRem | Деление и остаток | перегружен | DivRem(x, y, rem) |
| E | База натурального логарифма (число *е*) | double | 2,71828182845905 |
| Exp | Экспонента | double | ex записывается как Exp(x) |
| Floor | Округление до меньшего целого | double | Floor(double х) |
| IEEERemainder | Остаток от деления | double | IEEERemainder(double x, double y) |
| Log | Натуральный логарифм | double | logex записывается как Log(x) |
| Log10 | Десятичный логарифм | double | log10x записывается как Log10(x) |
| Max | Максимум из двух чисел | перегружен | Max(x, y) |
| Min | Минимум из двух чисел | перегружен | Min(x, y) |
| PI | Значение числа π | double | 3,14159265358979 |
| Pow | Возведение в степень | double | xy записывается как Pow(x, y) |
| Round | Округление | перегружен | Round(3.1) даст в результате 3 Round (3.8) даст в результате 4 |
| Sign | Знак числа | int | аргументы перегружены |
| Sin | Синус | double | Sin(double x) |
| Sinh | гиперболический синус | double | Sinh(double x) |
| Sqrt | Квадратный корень | double | √x записывается как Sqrt(x) |
| Tan | Тангенс | double | Tan(double x) |
| Tanh | Гиперболический тангенс | double | Tanh(double x) |

**Пример 1.**  Вычислить значение выражения по формуле

**

using System;

namespace ConsoleApplication1

{

classProgram

{

staticvoid Main(string[] args)

{

doublex,a,y; string s; // описываем переменные

Console.WriteLine("Введите x"); //Выдаём приглашение для ввода х

s = Console.ReadLine(); //Вводимстроку s

x = Convert.ToDouble(s); //Преобразовываем строку в х

Console.WriteLine("Введите alfa");

s=Console.ReadLine();

a=double.Parse(s);

double y=Math.Sqrt(Math.PI\*x)-Math.Exp(0.2\*Math.Sqrt(a))+

2 \* Math.Tan(2 \* a) + 1.6e3 \* Math.Log10(Math.Pow(x, 2)); //Вычисляем y согласноформуле

Console.WriteLine("Для х= {0} и alfa ={1}", x, a); //Печатаем введенные значения переменных

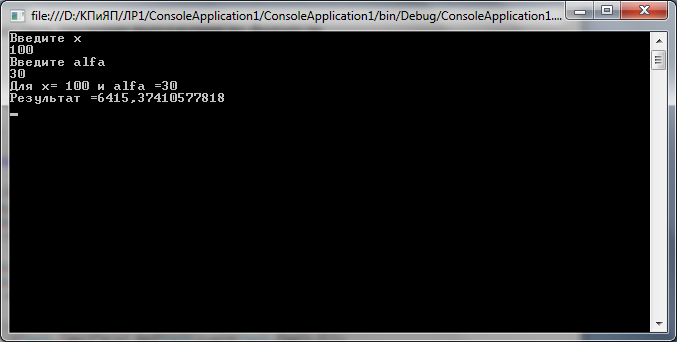
Console.WriteLine("Результат =" + y); //Печатаем полученный результат

Console.ReadKey(); //Ожидаем нажатия на любую клавишу в конце программы

}

}

}



## Пример №2

С начала суток прошло N секунд (N — целое). Записать текущее время выраженное в часах, минутах, секундах. Даже если прошло несколько суток, отобразить время последних суток.

usingSystem;

namespaceConsoleApplication1

{

classProgram

{

staticvoid Main(string[] args)

{

int N, hours, minutes, seconds; string s;

Console.Write("Введите число секунд прошедших от начала суток N=");

s = Console.ReadLine(); //вводим произвольное число секунд

N = Convert.ToInt32(s);

hours = N / 3600 % 24; // находим число часов, и оставляем только

//часы в последних сутках, если их больше 24

minutes = N % 3600 / 60; // определяем число минут

seconds = N % 3600 % 60; // определяем число секунд

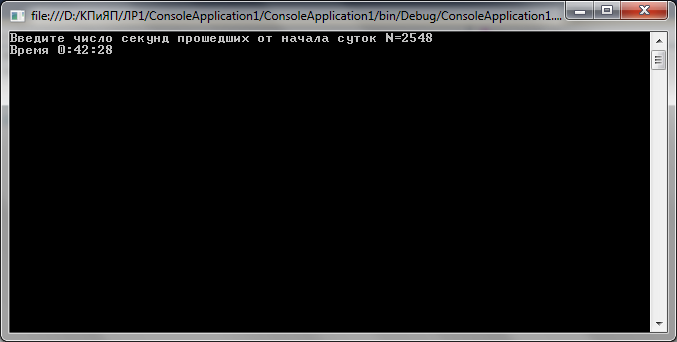
Console.Write("Время {0}:{1}:{2}", hours, minutes, seconds);

Console.ReadKey();

}

}

}



1. **Порядок выполнения работы**

1 Разработайте алгоритм решения задачи в виде блок-схемы

2 Создайте консольное приложение в среде Microsoft Visual Studio

3 Осуществите ввод данных в разрабатываемой программе

4 Вычислите значение выражение согласно варианту

5 Осуществите вывод результата на консоль

6 Оформите отчет о работе

1. **Форма отчета о работе**

*Номер учебной группы \_\_\_\_\_\_\_\_*

*Фамилия, инициалы обучающегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Задание: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

1. **Контрольные вопросы и задания**
2. Дайте определение линейной программы?
3. Как работать с математическими функциями в C#?
4. Как осуществляется ввод данных в C#?
5. Перечислите значимые типы данных?

5 Перечислите основные операции в С#?

**Рекомендуемая литература**

**Албахари, Д.** C# 5.0. Справочник. Полное описание языка / Д. Албахари, Б. Албахари : [пер. с англ. Ю.Н.Артеменко]. М., 2013.

**Гриффитс, И.** Программирование на С# 5.0 / И. Гриффитс ; [пер. с англ. И.М. Райтмана]. М., 2014.

**Павловская, Т.А.** C#. Программирование на языке высокого уровня / Т.А. Павловская. СПб., 2015.

**Стиллмен, Э.** Изучаем С# / Э. Стиллмен, Дж. Грин. СПб., 2014.