Министерство науки и высшего образования
Российской Федерации
Московский физико-технический институт
(национальный исследовательский университет)
Заочная физико-техническая школа

ИНФОРМАТИКА и ИКТ

Основы языка программирования

Задание №1 для 9-х классов

(2020 – 2021 учебный год)

Долгопрудный, 2020

Составители: В.В. Мерзляков, ассистент кафедры информатики СУНЦ МГУ, А.Е Дербышев, выпускник ФОПФ МФТИ, младший научный сотрудник лаборатории теоретической физики Объединенного института ядерных исследований.

Информатика: задание №1 для 9-х классов (2020 – 2021 учебный год), 2020, 18 с.

Дата отправления задания – 10 октября 2020 г.

Составители:
Мерзляков Василий Владимирович
Дербышев Андрей Евгеньевич

Подписано 07.07.20. Формат 60×90 1/16.
Бумага типографская. Печать офсетная.
Усл. печ. л. 1,12. Уч.-изд. л. 1,00.

Заочная физико-техническая школа
Московского физико-технического института
(национального исследовательского университета)

Институтский пер., 9, г. Долгопрудный, Москов. обл., 141700.
ЗФТШ, тел/факс (495) 408-51-45 – **заочное отделение,**
тел./факс (498)744-63-51 – **очно-заочное отделение,**
тел. (499) 755-55-80 – **очное отделение.**

e-mail: zftsh@mail.mipt.ru

Наш сайт: www.school.mipt.ru

Введение

Мы рады приветствовать вас на курсе Информатики. Данный курс рассчитан на три года обучения. Ориентировочно, это девятый, десятый и одиннадцатый классы средней школы. Поэтому в ходе изложения будут использоваться соответствующие знания из курсов математики.

Данный курс состоит из трёх больших частей, которые будут чередоваться по заданиям. Первая часть курса – теоретическая. В ней будут рассматриваться общие знания, которые необходимы любому человеку, собирающемуся связать свою жизнь с техническими специальностями. В частности, будут рассматриваться особенности представления информации различного вида (числовая, текстовая, графическая и т. д.), алгебра логики, математическая теория информации, теория алгоритмов и многое другое.

Вторая часть курса – программистская. В этой части мы будем рассматривать основные концепции языков программирования и учиться писать полноценные программы. Вас ждёт большое множест-во задач самого разного вида. По завершении курса вы сможете не просто писать программы на одном языке программирования, но уже будете обладать достаточными знаниями, чтобы самостоятельно легко изучать другие языки.

Третья часть курса – технологическая. Здесь будут рассматриваться информационно-коммуникационные технологии. В частности, компьютерные сети, обработка баз данных, работа с электронными таблицами и т. д.

На первом году обучения теоретических заданий не будет, а основная масса будет посвящена программистской части курса. Основная цель первого года обучения – овладеть языком программирования как инструментом для дальнейшего использования. В качестве учебного языка программирования мы выберем язык Pascal. Тому есть несколько причин. Во-первых, этот язык изначально создавался для обучения программированию, и в нём нет большого количества сложных тонкостей, для понимания которых требуются глубокие специальные знания (как, например, в языках C/C++). Во-вторых, концептуально Pascal является каноническим языком процедурной парадигмы программирования, и после него можно очень легко переходить на любые другие языки.

Также нам будет интересно не просто решить конкретную задачу, а овладеть стратегиями для решения больших классов задач. Нужно будет

научиться видеть общие моменты в предлагаемых задачах и сформировать определённые шаблоны для быстрого написания программ. Кроме того, особое внимание мы будем уделять красоте и эффективности алгоритмов.

Для выполнения заданий программистской части курса вам будет необходимо установить себе среду программирования, либо воспользоваться online-компилятором. Имейте в виду, что даже для языка Pascal различные среды программирования могут серьёзно различаться (например, free pascal и Pascal ABC.NET). Мы будем обращать внимание на данные различия по ходу изложения материала.

В рамках первого задания мы будем изучать основы языка программирования и особенности выполнения арифметических операций.

§1. Алфавит языка Pascal

Изучение любого нового языка всегда начинается с алфавита. В алфавит языка Pascal входят следующие элементы:

1) Заглавные и строчные латинские буквы, символ подчёркивания (по грамматике языка символ подчёркивания считается буквой): _, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

2) Цифры: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0.

3) Знаки операций: + (плюс), – (минус), * (умножить), / (разделить), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), = (равно), <> (не равно). Последний знак состоит из знаков «меньше» и «больше», записанных без пробелов.

4) Знаки пунктуации, специальные символы:

{ } или (* *)	Скобки комментариев
[ ]	Выделение индексов массивов, элементов множеств
' '	Выделение символа или строковой константы
( )	Выделение выражений, списков параметров
:=	Знак оператора присваивания
;	Разделение операторов и объявлений
:	Отделение переменной или константы от типа. Отделение метки от оператора

=	Отделение имени типа от описания типа. Отделение константы от её значения
,	Запятая для разделения элементов в списке
. .	Разделение границ диапазона
.	Конец программы, отделение целой части от дробной
#	Обозначение символа по его коду

В таблице приведены не все знаки пунктуации, а лишь те, которые будут использоваться при дальнейшем изложении.

5) Служебные (ключевые) зарезервированные слова.

Некоторые слова имеют предопределённое значение и используются в качестве элементов при построении сложных конструкций языка. Приведём некоторые примеры зарезервированных служебных слов, которые нам понадобятся в дальнейшем:

`and, array, begin, case, const, div, do, downto, end, for, if, mod, not, of, or, program, repeat, string, then, to, type, until, var, while, xor.`

В дальнейшем, условимся при написании текстов программ выделять жирным шрифтом служебные зарезервированные слова.

§2. Структура программы

Рассмотрим общую структуру программы на языке Pascal. Программа состоит из 2 частей: разделов описаний и раздела действий (команд, операторов). Раздел операторов представляет собой некую последовательность команд (операторов)**,** которые должен выполнить компьютер. Другими словами, раздел действий – это собственно программа. В разделах описаний программист сообщает компьютеру, какие объекты он будет использовать в своей программе, сколько памяти под них нужно выделить и т. д.

Из всего перечисленного выше обязательным для каждой программы является лишь раздел действий. Разделов описания в про-грамме может и не быть (хотя в содержательных задачах они будут). Раздел действий начинается со служебного зарезервированного слова `begin`, далее записывается список операторов (собственно программа) и в конце пишется служебное зарезервированное слово `end` и ставится точка.

Внутри списка все операторы отделяются друг от друга спе-циальным символом – точкой с запятой. В этом задании мы познако-мимся со следующими операторами:

1) пустой оператор;
2) оператор присваивания;
3) операторы вывода;
4) операторы ввода.

Начнём с пустого оператора. В языке Pascal пустой оператор – это просто ничего. Он **не содержит никаких символов и не выполняет никакого действия.** Рассмотрим пример программы состоящей из одного пустого оператора:

```
begin
end.
```

Эта программа начинается и сразу же заканчивается, не выполняя никаких содержательных действий. Рассмотрим другой пример:

```
begin
; ; ;
end.
```

Эта программа состоит уже из четырёх пустых операторов, отделённых друг от друга точками с запятой.

Для того чтобы познакомиться с разделами описаний, сначала нужно изучить объекты, которые мы можем использовать в программе.

§3. Константы и переменные

Познакомимся с двумя важнейшими в программировании понятиями.

Определение 1. ***Константой*** *называется объект, который получает значение до начала выполнения программы и не может менять его в ходе выполнения.*

Определение 2. ***Переменной*** *называется объект, который может менять своё значение в ходе выполнения программы.*

С каждой переменной, используемой в программе, связывается область в оперативной памяти компьютера, размер которой зависит от типа объекта. Каждая константа и каждая переменная имеет своё имя **(идентификатор)**, по которому мы и обращаемся к этим объектам, чтобы с ними работать. Каждое имя должно быть уникальным, чтобы не возникала ситуация неопределённости – какой объект выбирать. Если в

программе две переменные имеют одно и то же имя, компьютер откажется выполнять данную программу.

Правила именования в языке Pascal следующие: именем может являться любая последовательность латинских букв и цифр, начинающаяся с буквы. При этом заглавные и строчные латинские буквы не различаются, то есть имена aBBA и AbBa на самом деле обозначают один и тот же объект. Ещё одна интересная особенность построения имён заключается в том, что символ подчёркивания (_) считается латинской буквой, поэтому он также может входить в состав имени и, более того, имя с него может начинаться. Последнее правило именования заключается в том, что имена переменных или констант не должны совпадать со служебными зарезервированными словами языка.

Каждая переменная и каждая константа помимо имени ещё имеет свой тип. Тип определяет три вещи:

1) размер области оперативной памяти, отводимой под соответствующую переменную;

2) множество значений, которые может принимать соответствующая константа или переменная;

3) набор операций, которые можно выполнять с соответствующей константой или переменной.

Для того чтобы работать в программе с константой или переменной, её нужно сначала описать до раздела действий.

Переменные описываются следующим образом: сначала записывается служебное зарезервированное слово var, затем ставится пробел, указывается имя, которое мы хотим дать нашей переменной, ставится двоеточие и записывается тип переменной. После этого ставится точка с запятой. Например, запись

```
var x:integer;
```

означает, что в нашей программе будет использоваться переменная с именем x, имеющая тип integer (целое число). Если в программе будет несколько переменных одного типа, то можно до двоеточия перечислить их имена через запятую, а не выписывать отдельную строчку для каждой переменной. Пример: `var x,y:integer;`

Для описания константы необходимо записать служебное зарезервированное слово const, затем указать имя константы (отделив

его как минимум одним пробелом от слова const), поставить знак равенства и тут же задать её значение, например:

```
const N=1000;
```

После окончания описания константы также ставится точка с запятой.

В программе принято сначала описывать константы (если они есть), а уже затем переменные (а они есть практически в любой программе).

Замечание. Термином «константа» в программировании принято обозначать ещё одно понятие – если в программе встречается некоторое конкретное значение (например, число 1000)**,** то оно также называется константой соответствующего типа.

§4. Числовые типы переменных. Оператор присваивания

Рассмотрим два основных числовых типа переменных.

1) INTEGER. Этот тип характеризует целые числа. Переменные этого типа занимают в оперативной памяти 4 байта и могут принимать значения из диапазона [–2147483648,2147483647].Точное значение запоминать необязательно, главное помнить, что переменная этого типа может вмещать целые числа примерно до 2 миллиардов по модулю. В средах программирования free pascal, а также в устаревшем семействе BP тип integer считается не четырёх, а двухбайтовым с диапазоном значений [–32768, 32767]. В этих средах четырёхбайтовый целый тип имеет название longint

2) REAL. Этот тип предназначен для работы с вещественными (действительными) числами. Переменные этого типа занимают в оперативной памяти 8 байт. При записи констант этого типа целая часть числа отделяется от дробной точкой, а не запятой, как в математике, например: 3.14.

Обратите внимание, что названия типов (integer, longint, real) не являются служебными зарезервированными словами языка. То есть, теоретически можно создать переменные с такими именами, но тогда работать с данными стандартными типами мы уже не сможем. Поэтому, лучше этого не делать ☺.

Оператор присваивания позволяет изменить значение любой переменной программы (присвоить ей новое значение). Этот оператор выглядит следующим образом: записывается имя переменной, затем знак присваивания (:=), а потом значение, которое мы хотим присвоить

переменной. Присвоить можно константу или выражение соответствующего типа.

Пример 1.

```
X:=5; {в переменную X присвоили число 5}
Y:=X; {в переменную Y присвоили текущее значение переменной X}
Z:=X+Y; {в переменную Z присвоили сумму текущих значений переменных X и Y}
```

Если переменной присвоено некоторое значение, то в дальнейшем в программе при вычислениях вместо её имени будет подставляться это значение, пока мы не присвоим ей новое.

Пример 2.

```
X:=5;
Y:=X+4; {в переменную Y запишется число 9, так как текущее значение переменной X равно 5}
```

При использовании операторов присваивания необходимо соблюдать **правило совместимости типов**. Это правило заключается в том, что тип присваиваемого значения должен соответствовать типу переменной, которой мы хотим это значение присвоить. Есть исключение из этого правила: переменной типа `real` можно присвоить целое значение.

§5. Арифметические выражения

Арифметические выражения состоят из операций и операндов. В языке программирования Pascal существует шесть операций: сложение (обозначается знаком «+»), вычитание (обозначается знаком «−»), умножение (обозначается знаком «*»), деление (обозначается знаком «/»), деление нацело (обозначается словом «`div`») и взятие остатка от деления нацело (обозначается словом «`mod`»). Слова `div` и `mod` являются служебными зарезервированными.

Важным понятием в арифметике является понятие операнда. Операндами называются те объекты, над которыми выполняется арифметическая операция. В математике различные операции могут иметь разное количество операндов, но все арифметические имеют два операнда. Операндом для операции может являться как одиночное число или имя переменной, так и целое арифметическое выражение. Рассмотрим выражение (2+2)*2. У операции сложения операндами являются два числа 2, а у операции умножения правый операнд – это

число 2, а левый – это выражение в скобках (2+2). Прежде чем выполнять операцию, необходимо вычислить оба её операнда.

Приоритет операций в Паскале точно такой же, как и в математике. Сначала выполняются операции умножения, деления, `div` и `mod` (это тоже операции деления), а потом операции сложения и вычитания. Операции одного приоритета выполняются слева направо. Для изменения порядка действий можно использовать круглые скобки. Операции в скобках имеют более высокий приоритет, чем операции вне скобок. Так при вычислении выражения 2+2*2 получается число 6, потому что операция умножения имеет более высокий приоритет, чем сложение, и, следовательно, выполняется первой. Если же записать выражение (2+2)*2, то при вычислении получается число 8, потому что сложение в скобках выполняется раньше умножения.

Рассмотрим, как определить тип результата при вычислении арифметического выражения. Операции сложения, вычитания и умножения выдают целый результат, если оба их операнда целые, и вещественный, если хотя бы один из операндов – вещественный. Операция деления «/» **всегда** выдаёт вещественный результат. Даже если мы 4 делили на 2, всё равно в итоге получается нецелое число. На первый взгляд это кажется странным, но в отличие от математики в программировании каждое число кроме значения ещё имеет тип, и если типы у чисел не совпадают, то они НЕ считаются равными. Нужно уяснить, что **1 ≠ 1.0.** Это несложно понять, если помнить, что раз числа 1 и 1.0 имеют различные типы, то будут представлены совершенно разными последовательностями битов. Операции `div` и `mod` всегда выдают целый результат и, в отличие от всех остальных арифметических операций, могут иметь только целые операнды. Попытка применить данные операции к вещественным числам приведёт к тому, что программа просто не будет работать.

Давайте подробнее познакомимся с двумя последними операциями. Операция `a div b` выдаёт целую часть от деления числа `a` на число `b`. То есть `5 div 2 = 2`, а `3 div 7 = 0`. Операция `a mod b` выдаёт остаток от деления `a` на `b` по следующему закону

```
a mod b = a - ((a div b) * b)
```

Приведём примеры выполнения этих их операций для всех возможных знаков операндов:

```
 5 div  3 =  1;  5 mod  3 =  2;
-5 div  3 = -1; -5 mod  3 = -2;
 5 div -3 = -1;  5 mod -3 =  2;
-5 div -3 =  1; -5 mod -3 = -2;
```

Операндами в арифметическом выражении также могут быть стандартные математические функции, которые приведены в таблице ниже.

Функция	Комментарий	Тип аргумента	Тип Результата
abs(x)	$\lvert x \rvert$ — модуль x	integer, real	совпадает с типом аргумента
sqr(x)	x^2	integer, real	совпадает с типом аргумента
sqrt(x)	$\sqrt{x}$ — корень квадратный из x	integer, real	real
Pi	3.1415926535897932385	нет	real
sin(x)	sin$x$	integer, real	real
cos(x)	cos$x$	integer, real	real
trunc(x)	отсекание дробной части x	real	integer
round(x)	округление x до ближайшего целого. Половины округляются к ближайшему чётному числу (Pascal ABC.NET) или в сторону увеличения модуля (free pascal)	real	integer

Необходимо отметить, что функциям sin и cos угол следует подавать в радианах, а не в градусах!

§6. Операторы вывода. Модификаторы формата

Операторы вывода являются важнейшей частью языка программирования, ведь только благодаря им, мы можем увидеть на экране компьютера результат работы нашей программы. В языке Pascal существует два оператора вывода: `write` и `writeln`. Правило их использования одно и тоже: после слова `write` или `writeln` в скобках через запятую перечисляются параметры, которые мы хотим вывести

(называемые списком вывода). Число этих параметров не ограничено. Разделителем между параметрами служит запятая:

```
writeln(параметр, параметр,...,параметр)
```

Существует три вида параметров: *константы*, *переменные* и *выражения* (например, арифметические выражения). Константы бывают числовые (это просто различные числа — целые и вещественные), логические (будут рассмотрены в следующем задании) и строковые. Любой текст, набранный с клавиатуры и заключённый в апострофы (одиночные кавычки), называется *строковой константой*. Если в текст нам нужно поместить апостроф, например, в слове O'key, на этом месте нужно набить два апострофа подряд вместо одного: `write('O''key')`. Все параметры в `write` или `writeln` независимы друг от друга, поэтому в одном и том же операторе могут встречаться параметры разных типов, в произвольном порядке.

При выполнении оператора вывода все параметры будут выведены в одной строке в том же порядке, в каком они перечислены в списке параметров. Любая константа, числовая или строковая, будет выведена так, как вы её написали в вызове `write` или `writeln` (в строковой константе начальный и конечный апострофы отображаться на экране не будут, а вместо двух апострофов, расположенных в строковой константе подряд, на экране появится в этом месте один); вместо переменной на экране появится её значение, а вместо арифметического выражения — результат его вычисления.

Между `write` и `writeln` существует единственное различие: после выполнения `writeln` курсор переходит на новую строку, а после выполнения `write` курсор остаётся в той же строке, и новый вывод данных с помощью `write` или `writeln` или ввод данных при помощи операторов ввода данных будут проходить в той же строке.

При выводе параметров пробелы между ними автоматически не вставляются, например, при печати чисел 1, 2, 3 с помощью `writeln(1,2,3)` все они сольются в одно число — `123`. Чтобы разделить выводимые элементы, можно поместить между ними символ пробела, например, `writeln(1,' ',2,' ',3)` или отформатировать вывод, поставив после каждого элемента списка вывода двоеточие и целое число (называемое **модификатором ширины поля**), которое указывает, сколько позиций на экране должна занимать выводимая величина, например, `writeln(1:3,2:3,3:3)`. Отметим,

что элемент дополняется начальными пробелами слева с тем, чтобы соответствовать указанной после двоеточия величине. Результаты выполнения двух последних операторов будут выглядеть так (символ подчёркивания обозначает пробел):

```
1_2_3
__1__2__3
```

Если указанное в модификаторе ширины поля число меньше, чем необходимо, то модификатор ширины поля игнорируется.

При выдаче на экран значений вещественных выражений в формате вывода полезно использовать ещё один модификатор, который записывается через двоеточие после модификатора ширины поля и называется **модификатором точности**. Он будет обозначать количество символов после десятичной точки, которое мы хотим вывести. Например, при выводе результата стандартной функции `pi`, которая с машинной точностью выдаёт значение числа π, оператор `write(pi:0:0,pi:6:2,pi/2:2:0)` выдаст на экран:

```
3  3.14 2
```

Заметим, что при печати фиксированного количества цифр вещественного числа оно предварительно округляется по правилам математики. Если вещественное число содержит после десятичной точки меньше цифр, чем количество символов для печати, указанное в модификаторе точности, то число выводится с незначащими нулями, например, оператор `write(3.14:3:4)` выдаст на экран:

```
3.1400
```

Модификатор точности можно применять только к параметрам вещественного типа. Использование модификатора точности с параметрами других типов является критической ошибкой (программа не будет работать). Модификатор ширины поля можно использовать с любым типом параметра вывода.

§7. Операторы ввода

Операторы `read` и `readln` предназначены для задания значений переменным путём ввода их с клавиатуры. Правило их применения одно и то же: после слова `read` или `readln` в скобках через запятую перечисляются имена переменных, значения которых мы хотим ввести (список ввода). Число этих имён не ограничено. Запятая служит разделителем между именами переменных:

```
readln(имя, имя, ..., имя)
```

При срабатывании оператора `read` или `readln` выполнение программы будет приостановлено до тех пор, пока пользователь не введёт соответствующее количество значений. Вводимые значения должны быть того же типа, что и переменные. Если в `read` или `readln` переменных несколько, то они могут быть набиты в одной строке, но одно число от другого должно отделяться пробелом или переводом строки. Чтобы выполнить оператор `read` или `readln` после набивания значений с клавиатуры, нужно нажать клавишу «Enter». В результате переменные приобретут заданные вами значения. Между `read` и `readln` существует единственное различие: после выполнения `readln` курсор переходит на новую строку, игнорируя всю оставшуюся информацию в прежней строке, а после выполнения `read` курсор остаётся в той же строке, и новая набивка данных для `read` или `readln` будет проходить в той же строке. Но, так как после нажатия клавиши «Enter» курсор в любом случае переходит на новую строчку, для однократного ввода значений переменных разницу между операторами `read` и `readln` заметить невозможно. Тем не менее, в данном случае лучше использовать `readln`. Оператор `readln` можно использовать и без параметров вообще. Тогда программа просто будет находиться в режиме ожидания, пока пользователь не нажмёт клавишу «Enter». Такой оператор, например, удобно ставить самым последним оператором в программе, при работе в средах free pascal. Тогда можно сразу посмотреть результат работы программы, а потом нажать «Enter», и только тогда работа программы завершится.

Замечание. Перед вводом данных с клавиатуры рекомендуется выдавать на экран приглашение, например:

```
write('Введите число a => ');
readln(a);
```

§8. Примеры простейших программ

Теперь воспользуемся всеми полученными знаниями и рассмотрим примеры простейших программ. Напомним, что заглавные и строчные буквы не различаются, а служебные зарезервированные слова выделяются жирным шрифтом.

Пример 1. *Идёт k-ая секунда суток (k вводится). Вывести, сколько полных часов h и полных минут m прошло с начала суток.*

Решение. В этой задаче все параметры целые. Решается она с помощью операций `div` и `mod`. Эти операции можно использовать для

«срезания периодов» при переводе мелких единиц измерения в более крупные (например, секунд в минуты). Операция `div` нам выдаст количество полных периодов (сколько полных минут содержится в большом количестве секунд), а операция `mod` – количество единиц в последнем неполном периоде (сколько секунд не укладывается в полное количество минут). Приведём полный текст решения.

```
var k,h,m:integer;
begin
  write('Введите номер секунды в сутках');
  readln(k);
  h:=k div 3600;
  m:=k mod 3600 div 60;
  writeln('С начала суток прошло ',h,' часов и
  ',m,' минут');
end.
```

Пример 2. *Вводится четырёхзначное число. Вывести произведение его цифр.*

Решение. Эта задача показывает ещё одно применение операций `div` и `mod` – выделение цифр из целого числа. Описанное ниже решение работает только для случая, когда количество цифр в числе заранее известно. В противном случае придётся использовать цикли-ческий алгоритм. Приведём текст решения.

```
var N,c1,c2,c3,c4:integer;
begin
  write('Введите целое четырёхзначное число');
  readln(N);
    c1:=N div 1000;
    c2:=N div 100 mod 10;
    c3:=N div 10 mod 10;
    c4:=N mod 10;
  writeln('Произведение цифр вашего числа равно
  ',c1*c2*c3*c4);
End.
```

Правила контроля

Теперь вам будут предложены контрольные вопросы и задачи. За каждый правильный ответ будут ставиться баллы. Максимальное количество баллов за задание указано в скобках после его номера. Если задание стоит более одного балла, то возможно получить частичный балл за частично верное решение. Имейте в виду, что более объёмные и сложные задания стоят дороже. Итоговая оценка будет определяться по сумме набранных баллов. Желаем успеха!

Контрольные вопросы

1(1). Какие из приведённых последовательностей символов могут быть именами переменных или констант в языке программирования Паскаль? Запятые являются разделителями между последовательностями.

_1 , 1_ , 20.01.1980 , Program , const , const_1 , repaet , max:1 , integer .

2(1). Какой результат получится после выполнения следующего алгоритма:

```
var a,b,c,d: real;
begin
    a:=5;
    b:=a-3;
    c:=a*(b-1);
    d:=(a+b+c) mod 2;
   writeln(b);
end.
```

3(1). Что будет выведено на экран после выполнения следующих операторов, если с клавиатуры были введены числа 1,2,3,4,5,6,7,8:

```
readln(c,b,a,f,c,b,a1,b);
writeln(a:2,b,c:4);
```

4(1). Уберите из выражения лишние скобки (то есть те, которые не изменяют порядок действий при вычислениях).

((7 – x)*(2-y)) – (y –(- z))*(13 mod (4*2))

5(3). Запишите по правилам Паскаля выражение:

a) $\frac{\sin 30^\circ}{\sqrt[4]{x^3 - \cos 50^\circ}}$, b) $\frac{|x \| y| \left(\pi/4\right)^2}{1 + \frac{|y|^2}{1+x^2}}$, c) $\frac{\{5x\}[y]}{1 + \frac{1}{1 + \frac{1}{1+y^5}}}$

6(2). Есть целая переменная n и вещественная переменная y. Какие из следующих операторов присваивания верные, а какие нет и почему?

y:=n , n:=y , y:=5,41 , y:=n mod 4 , n:=(5*n)/5 , n=5*n , n:= trunc(n/4) , y:=y mod n.

7(2). Возможно ли вычислить данный оператор присваивания при каких-либо типах переменных x и y? Если да, то укажите типы, если нет, то объясните, почему.

y:=(trunc(y)mod x+ cos(x) +y*x mod 2

8(3).Что выведут данные программы?

a)
```
var a,b,c:integer;
 begin
a:=1 + 22 mod 2;
b:=2 + 24 mod 3;
c:=2.1 + 26 mod 4;
writeln(a,' ',b,' ',c)
 end.
```

b)
```
var a,b,c:integer;
begin
a:=trunc (3.14);
 b:=a*21 div 9;
c:=b*a mod 3;
 writeln(a:3,' ',b,' ',c)
end.
```

c)
```
var k:integer;
begin
K:=135 mod 11 div 5 mod 7*2;
write(k);
end.
```

9(1). Какое число необходимо подать на вход программе, чтобы было выведено число 5311?

```
 var n:integer;
begin readln(n);
writeln(n div 200 mod 20*1000 + n div 10 mod 10*100 + n mod 10*15 + n div 1000)
end.
```

10(2). Что делает данная программа?

```
var n:integer;
 begin
readln(n);
writeln(1-(n mod 3+1)div 2)
 end.
```

Задачи

(Использование массивов, строк, условных операторов не допускается)

1(2). Напишите программу, на вход которой подается с клавиатуры два натуральных числа n и m и которая выводит 1, если n делится на m и 0 в противном случае. (Подсказка в вопросе 10).

2(2). Напишите программу, на вход которой подается с клавиатуры шестизначное число, и она выводит чисто с перевернутыми тройками цифр. Например, из числа 123456 должно получиться число 321654.Строки не использовать.

3(2). Петя пошел в магазин за молоком. Мама дала ему n 50 рублевых купюр и сказала купить молока на все деньги . Молоко стоит 52р 50к. Но в магазине закончилась сдача и продавщица вместо сдачи выдает спички по 1р10к.(сумма меньше 1р10к округляется до целого коробка). Сколько коробков спичек принесет Петя домой?

4(3). С клавиатуры вводится вещественное число с четырьмя значащими цифрами. Вывести сумму этих цифр.

5(3). С клавиатуры вводится натуральное пятизначное число. Вывести сумму четных цифр, если число четное и сумму нечетных цифр, если число не четное.

6(4). С клавиатуры вводится натуральное число n, не превосходящее 180. Определите n-ую цифру в ряду: 1011121314…9899. В данном ряду подряд выписаны все двузначные числа. На экран вывести одно число – искомую цифру.

7(4). С клавиатуры вводится целое четырехзначное число n и цифра m. Вывести сколько раз в записи n встречается m. Например: при вводе 2324 и 2 на выходе 2, а при вводе 2333 и 3 – 3. (Подсказка: использовать решение задачи 1, а так же тот факт, что 0 делится нацело на любые числа).