В. М. Казиев ***РАЗВИВАЮЩИЕ ЗАДАЧИ (ЭТЮДЫ)***

цикл статей "Развивающие задачи"; журнал "Информатика и образование", N 3, 1997; N2, 1998; приложения "Информатика в уроках и задачах" к этому журналу, N2, 1998; N1, 1999.)

В последнее время междисциплинарные аспекты, проблемы гармонизации, гуманитаризации и гуманизации образования становятся более явными, сложными и актуальными. В то же время преподавание информатики классически тяготеет к технократическому варианту, "к преподаванию основ вычислительной техники, программирования и пользовательского интерфейса" (мы ни в коей мере не умаляем значимость такого подхода на определённом уровне подготовки и для определённых категорий обучаемых), что уже не адекватно стоящим перед наукой "Информатика" проблемам.

При решении задач образовательной информатики в соответствии с указанной парадигмой важное значение приобретает обучение на развивающих, междисциплинарных специально подобранных дидактических единицах, называемых далее развивающими задачами или этюдами (по аналогии с замечательной книгой "Этюды для программистов" Ч. Уэзерелла).

Задачи и решения могут быть использованы для проведения научной работы, олимпиад, обучения. Замечания, отзывы, пожелания и другие задачи указанного цикла (он постоянно пополняется автором) присылать по электронной почте: kaziev@ns.kbsu.ru. Ссылки на задачи обязательны.

1. Информация: кодирование, шифрование, измерение. Информационно–логические задачи

Задачи с решениями

1.1 Возможна ли ЭВМ с быстродействием 10^{20} операций в секунду? Ответ подтвердить убедительными вычислениями.

Решение. Самая большая скорость переноса электронов ≈ скорость света в вакууме $v=3\beta 10^{10}$ см/сек. При выполнении одной операции свет (электрон) должен пройти расстояние не меньшее диаметра атома водорода $s=10^{\approx 8}$ см. Время этого пути равно:

$$t = s/v = 10^{\approx 8}\ см/(3\beta 10^{10}\ см/сек) = 10^{\approx 18}/3\ сек.$$

Следовательно, в идеальном варианте (если вообще удастся технически это когда–то реализовать) быстродействие компьютеров на принципах переноса электронов, не может превышать $3\beta 10^{18}$ операций в секунду.

1.2. Привести пример лексикографически упорядоченной, а также неупорядоченной последовательностей. Добавьте слова, изменяющие упорядоченность (неупорядоченность), если это возможно.

Решение. Слова "*арбуз*", "*банан*", "*киви*", "*лимон*", "*тыква*" ≈ лексикографически упорядочены по входному алфавиту русского языка. Добавляемое в конце слово – "*яблоко*". Слова двоичного алфавита "*101*", "*011*", "*010*", "*000*" лексикографически неупорядочены. Добавить (вставить) слова упорядочивающие слова ≈ невозможно (почему?).

1.3. Описать словесно и математически закон формирования чисел данного ряда (начиная с третьего): *1, 1, 2, 5, 29, 866, ...*

Решение. Нетрудно заметить, что начальный отрезок похож на ряд Фибоначчи, но это обманчиво. На самом деле, каждый член ряда (с третьего) получается не сложением двух предыдущих, а сложением их квадратов. Математически это записывается в виде:

$$x_n = (x_{n\approx 1})^2 + (x_{n\approx 2})^2\ ,\ n=3, 4, \ldots .$$

1.4 Найти систему кодировки (т.е. правило шифровки) текста, если текст "*АРГУМЕНТФУНКЦИИ*" зашифрован с помощью этого кода как "*БСДФНЖОУХФОЛЧКК".*

Решение. Сравнивая число символов исходного и зашифрованного текста (они равны) и символы, стоящие на одинаковых местах в исходном и шифрованном тексте, замечаем, что этот текст закодирован кодом Цезаря: каждый символ алфавита (любого выбранного множества символов) кодируется следующим за ним по порядку (определенном в алфавите) символом, причем последний символ кодируется первым символом алфавита т.е. алфавит "закольцован".

1.5 С некоторой планеты поступают импульсы. Отождествив один импульс с полубайтом, получаем последовательность сообщений вида:

X[1]=0001000010010101011000 01,
X[2]=0001000100001001010101010110000 1,
X[3]=0001000100010000100101010101010110000 1 и т.д.

Выдвинута гипотеза: если сообщения $X[i]$, $i=1,2,3,...$ подчиняются некоторому простому закону, то на планете живут разумные существа. Разумные ли существа живут на планете?

Решение. Переводя полубайты в десятичную систему, получаем сообщения вида:

X[1]=109561, X[2]=11095561, X[3]=1110955561, ...

Нетрудно видеть, что $X[n] = 111...11109555...55561$. ($n$ единиц n пятерок). Представим его в виде:

$$X[n]=11...1\times 10^{n+4}+9\times 10^{n+2}+55...5\times 10^{2}+61=(10^{n}\approx 1)/9\times 10^{n+4}+9\times 10^{n+2}+$$

$$+5(10^{n}\approx 1)10^{2}/9+61=(10^{2n+4}\approx 14\times 10^{n+2}+49)/9=[(10^{n+2}\approx 7)/3]^{2}.$$

Следовательно, последовательность сообщений подчиняется закону:

$X[1]=331^2$, $X[2]=3331^2$, $X[3]=33331^2$, ... Вывод: на планете живут разумные существа.

1.6. Найти все решения (цифры) *S, A, P, B, D,* если: *SP4+APD=DBAB*. Решение найти за минимальное число рассуждений, если одно рассуждение – простое утвердительное повествовательное предложение.

Решение. Наши рассуждения таковы:

1) так как при сложении двух десятичных цифр возможен перенос в старший разряд только *1*, то однозначно определяем *D=1*;

2) тогда из младшего разряда получаем однозначно *B=5;*

3) из *2*–го и *3*–го разрядов получаем две возможности:

а) $P+P = A$ *и* $S+A\approx 10 = 5$ *(2P=A, S+A=15);*

б) $P+P\approx 10 = A$ *и* $S+A+1\approx 10 = 5$ *(2P=A+10, S+A=14);*

если рассматривать случай а), то отсюда следует, что $A \approx$ четно т.е. $A=2, 6$ или 8, а $S \approx$ нечетно и $S > 5$, т.е. $S=7$ или 9, т.е. подходят только значения $A=6, S=9$ (иначе получаем $P=4$ при $A=8$ и $S=7$, что невозможно ≈ цифра *4* явно присутствует в примере). Аналогично рассматривается случай б).

1.7. Найти и записать закономерность в последовательности A вида:

1, 10, 11, 100, 111, 1000, 1111, 10000, ...

Рассмотреть отдельно случаи: а) числа заданы в десятичной системе;
б) числа заданы в двоичной системе.

Решение. Рассмотрим случай "а", так как случай "б" легче. Из сравнения членов $A[i]$ *(i=1,2,...)* последовательности стоящих на четных местах и на нечётных местах (отдельно) видно, что:

1) элемент на нечётном месте, получается из элемента на предыдущем нечётном месте добавлением единицы справа к нему;

2) каждый элемент на чётном месте, получается из элемента на предыдущем чётном месте добавлением справа к нему нуля.

Этот словесно описанный закон можно записать на математическом языке, в аналитическом виде. Получим отдельно для случаев 1) или 2):

A[2n]=10× A[2n≈2], A[2n≈1]=10× A[2n≈1]+1, n=1, 2, ...

Можно записать формулу, объединяющую обе эти формулы в одну:

A[2n+m]=10× A[2n+n≈2]+m , где m=0 или m=1.

Существует лучшая, но менее наглядная форма записи: A[2n+mod(n,2)]=10A[2n+mod(n,2)≈2]+ mod(n,2).

1.8. *Петя, Вася, Дима, Коля* и *Гена* хотят (каждый по отдельности) пойти в кинотеатры, где они не были. При этом:

– Петя не был в "*Победе*", "*Востоке*", "*Авроре*", "*Мире*";
– Вася был в "*Авроре*", "*Мире*";
– Дима не был лишь в "*Авроре*";
– Коля не был в "*Победе*", "*Мире*";
– Гена был в "*Востоке*", "*Мире*", "*Юности*".

Найти минимумом рассуждений, кто и куда должен пойти (в перечисленные кинотеатры). За одно рассуждение принять одно повествовательное предложение относительно кинотеатра или юноши.

Решение. Эту задачу решим без составления таблицы (для краткости). Первое рассуждение: *Дима* не был только в "*Авроре*", поэтому он идет в "*Аврору*". Второе рассуждение: так как *Гена* не был только в "*Авроре*" и "*Победе*", то он идет в "*Победу*". Третье рассуждение: так как *Коля* не может уже идти в "*Победу*", то он идет в "*Мир*". Четвёртое рассуждение: так как *Петя* не может пойти уже в "*Победу*", "*Аврору*" или в "*Мир*", то он идет в "*Восток*". Пятое рассуждение: *Васе* однозначно "остается" "*Юность*".

1.9. Друзья *X, Y, Z, U, V* должны поехать в разные города *А, Б, В*, *Г, Д, Е*. При этом : *X* может ехать только в *А, Б, Д*; *Y* может ехать только в *Б* и *Г*; *Z* может ехать только один и в *В*; *U* не может ехать вместе с *Y*; *V* может ехать только с *X* и *Z,* но не в *D*. В каком городе мог быть каждый из них, если оказалось, что вдвоем они не были ни в одном городе.

Решение. Составим таблицу возможностей каждого, пометив города куда может ехать каждый знаком "+", а куда не может – знаком "–":

	X	**Y**	**Z**	**U**	**V**
А	+	–	–	+	+
Б	+	+	–	–	+
В	–	–	+	+	+
Г	–	+	–	–	–
Д	+	–	–	+	–
Е	–	–	–	+	–

Из анализа этой таблицы следует:

1) *Z* может ехать только в *В*;
2) *U* должен ехать только в *Е* (иначе в *Е* никто не едет);
3) *Y* должен ехать только в *Г* (иначе в *Г* никто не едет);
4) *X* должен ехать только в *Д* (так как *U* в *Д* уже не может ехать);
5) *V* может ехать только в *А* или в *Б* (т.е. задача имеет два решения).

1.10. Задан квадрат *ABCD* со стороной *n*, около которого описана и в который вписана две окружности. Вписанная окружность разделена на *4* равные части. Как раскрасить тремя цветами (сделать "заливку" ≈ как, например, в графическом редакторе) все полученные части большей окружности, чтобы никакие две соседние части не имели бы одинаковый цвет по общей у них протяжённой границе? Что скажете о решении, если на *4* равные части делится описанная окружность?

Решение. Одно из возможных решений приведено на рисунке ниже (номер внутри области фигуры означает номер цвета закраски):

1
2 3
1 2

3 1
2 3
1 2
3

При делении большей окружности на *4* равные части, можно раскрасить тремя цветами все полученные части фигур так, что никакие две соседние фигуры, имеющие протяжённую границу не будут окрашены одинаковым цветом.

Задачи без решений

1a1 Сколько различных символов в сообщениях (один символ ≈ *1* байт):

```
11100010000011111111000011100011,
11111000111110001110001000000011,
10111000000011111011000011100010.
```

1a2. Лазерный диск (*CD*) вмещает *600 М*. Сколько страниц учебника можно записать на *1* диск, если на *1* странице учебника можно записать *50* строк по *40* букв (символов) в строке? Каков должен быть объём *CD* для того, чтобы на нём помещалось ровно *1 Г* информации? Сколько бит информации помещается на *1* странице учебника указанного формата?

1a3. Запишите условие существования треугольника с заданными вершинами *A(a;b), B(c;d), C(m;n)* с помощью минимума условий. Запишите условие попадания заданной точки *M(x;y)* в этот треугольник. Условия должны быть наиболее компактными и полными. Сколько бит это условие занимает?

1a4. Сколько осмысленных предложений можно составить из слов данного предложения: "Поздним вечером переполненный автобус вез вахтенных рабочих домой".

1a5. Сколько всего байт необходимо для запоминания одного экрана в памяти ЭВМ, если каждая точка может быть одного из *8 (16, 32, 200, 2^n)* различных цветов. Каждый цвет кодируется одним байтом, а экран дисплея может вмещать *1024x640* точек?

1a6. Найти и записать словесно и формульно закон формирования чисел в последовательностях:

а) *10, 11, 100, 121, 1000, 1331*, ... ;
б) *1, 3, 15, 105, 945, ...* .

1a7. Текст "*Сегодня хороший ден?"* закодирован цифрами в виде

```
"10020304050607000904011120005020613".
```

Расшифровать код и определить код и символ, помеченный знаком "?". Ответ нужно обосновать за минимум утвердительных рассуждений.

1a8. Задан квадрат *ABCD* со стороной *10* см. около которого описана и в который вписана окружность. Вписанная окружность разделена на *4* равные части. Как можно раскрасить тремя цветами все полученные при этом части описанной окружности, чтобы никакие две соседние части не были бы окрашены в одинаковый цвет. Можно ли ограничиться двумя цветами?

1a9. Найти всевозможные решения (наборы цифр) *S, M, A, P, B, C*:

а) SAM + MAP = MPPM; б) A6A ≈ BB = CA2.

Решение найти за минимум рассуждений; одно рассуждение ≈ одно утвердительное высказывание относительно одного разряда чисел.

1a10. У *Петрова, Иванова, Семенова* и *Николаева* в фамилиях содержатся часть (корневая) их имён, причем только у *Николаева* они совпадают, т.е. *Николаева* зовут *Николай,* а *Семенова* зовут не *Петром.* Указать фамилии и имена каждого за минимум рассуждений об имени и/или фамилии.

1a11. *Петя, Миша, Ваня, Коля, Дима* должны одновременно поехать в города *Нальчик, Москва, Серпухов*, *Тольятти, Норильск*. При этом:

– *Петя* должен ехать только в *Нальчик, Москву* или *Норильск*;
– *Миша* должен ехать только в *Москву* или *Тольятти*;
– *Ваня* должен ехать только в *Серпухов* или *Тольятти*;
– *Коля* может ехать в любой город;
– *Дима* не может ехать вместе с *Ваней* или *Петей* в *Москву*.

В каком городе мог быть каждый, если оказалось, что они не нарушили ни одно из этих условий. Ответ обосновать минимумом рассуждений.

2. Системы счисления (*задачи с решениями)*

2.1. Вычислить наибольшее и наименьшее *5*–разрядное целое число в системе счисления с основанием *4*.

Решение. Наибольшее целое *n*–разрядное число, которое возможно записать в системе счисления с основанием *p*, очевидно, равно:

$$x_p^{\max} = \sum_{i=0}^{n-1}(p-1)\,p^i = \sum_{i=0}^{n-1} p^{i+1} - \sum_{i=0}^{n-1} p^i = p^n - 1$$

Наименьшее целое n – разрядное число в этой системе равно $x_{min} = \approx x^{max} = 1 \approx p^n$. Таким образом, в системе счисления с основанием *4* и числом разрядов *5* представим диапазон следующих чисел:

$$\approx 1023 = 1 \approx 4^5 \le (x)_4 \le 4^5 \approx 1 = 1023.$$

Этим формулам можно придать более компактный вид с использованием комбинаторики (или добавляя и затем отнимая *1*). Например, для двоичной системы

$a = 1\ 1\ 1\ \ldots\ 1\ 1\ 1_2 = (2^n \approx 1)_{10}$, $b = (1 \approx 2^n)_{10}$, n разрядов

а в восьмеричной системе счисления эти числа определяются в виде:

$a = 7\ 7\ 7\ \ldots\ 7\ 7\ 7_8 = (8^n \approx 1)_{10}$, $b = (1 \approx 8^n)_{10}$. n разрядов

2.2 Привести основные общие (различные) стороны множества чисел *(–1;1)* в обычной и машинной *n*–разрядной алгебре (арифметике).

Решение. С точки зрения обычной арифметики в интервале *(–1;1)* имеется бесконечное множество "плотно" расположенных точек, причем в любой окрестности каждой такой точки имеется хотя бы одна точка из этого множества. Такую арифметику называют часто *регулярной арифметикой.* Машинная арифметика имеет следующие особенности. Она нерегулярна – точки интервала сгущаются около нуля; кроме того, в этом интервале точка *x* "изолирована" – если взять её любую окрестность *(x–a; x+a),* где *a* – число, которое не превосходит машинный нуль, то в этом интервале нет других точек (отличных от *x*). Есть и другие особенности этих множеств (связанные с выполнением операций, например), но указанные особенности– основные.

2.3. Что такое числа (арифметика) с плавающей запятой, какие неудобства чисел с ограниченной разрядностью можно указать?

Решение. Эта форма представления чисел была предложена в *1937* году *Конрадом Цузе* для увеличения диапазона чисел, представимых в арифметике двоичных чисел, а также для повышения точности этого представления чисел. Например, в *16*–разрядной арифметике двоичных чисел можно представить диапазон целых *x* (старший разряд – под знак числа): $1 \approx 2^{15} < x < 2^{15} \approx 1$.

Если же в этой арифметике (не меняя её разрядность) отвести *7* разрядов под мантиссу, а 7 разрядов – под порядок, то уже представим диапазон чисел: $\approx 127 \times 2^{127} < x < 127 \times 2^{127}$ (два разряда – под знак числа и знак порядка; несколько упрощена и общая картина представления – для наглядности).

К "неудобствам" этой формы представления чисел можно отнести возможность возникновения следующих "особо опасных" ситуации:

а) если число достаточно мало, например, *a=0.12E+00,* то оно может быть представлено любым числом из наименьшего интервала включающего *a*, в частности, числом *0.120000001* или *0.199999999* и в этом случае сравнивать на равенство "в лоб" нельзя;

б) порядок выполнения операций может влиять на результат, например, в четырехразрядной арифметике с фиксированной запятой имеем:

`20.0000+0.0001=20.0001,` но при этом `0.2000E+02+0.1000E–05=0.2000E+02;`

в) может возникнуть так называемая ситуация "переполнения порядка" при сложении (умножении) очень больших чисел или "исчезновения порядка" при сложении (умножении) "очень малых чисел", например,

0.6000E+39× 0.1200E+64=0.9999E+99 (или неопределенно),
0.6000E≈35× 0.0200E≈65=0.9999E≈99 (или неопределенно)

при соответствующим образом определенной разрядности десятичной арифметики.

2.4. Выяснить, в какой системе счисления было выполнено сложение, где * обозначены неизвестные числа (возможно, разные), т.е. найти основание системы счисления *p*:

$(33*5*)_p + (1*643)_p = (52424)_p$.

Решение. Так как во втором разряде *(5+4)* получается число меньшее *9*, то основание системы будет меньше *9*, а так как в третьем разряде *(*+6)* есть цифра *6* (определенная только в системах счисления с $p \geq 7$), то $7 \leq p \leq 8$. Из второго разряда *(5+4=2)* следует, что был перенос в старший разряд единицы основания, т.е. $5+4 \approx p=2$ или $p=7$.

2.5. В саду *100* фруктовых деревьев – *14* яблонь и *42* груши. В какой системе счисления посчитаны деревья ?

Решение. Из условия задачи и переводом указанных чисел в систему счисления с пока неизвестным основанием *p* можно записать равенство:

$$1\times p^2 + 0\times p^1 + 0 = 1\times p^1 + 4 + 4\times p^1 + 2$$

или $p^2 \approx 5p \approx 6=0$. Из корней этого уравнения подходит в качестве основания системы только один – $p=6$. Деревья посчитаны в 6–ной системе счисления.

2.6. Найти число *x,* записанное в системе счисления с основанием *p,* если оно совпадает с его обратным кодом. При каких *p* возможно решение задачи?

Решение. Число в системе счисления с основанием *p* имеет вид:

$$x=x_0 x_1 \ldots x_{n\approx 1} x_n = x_{0\times} p^n + x_{1\times} p^{n\approx 1} + \ldots + x_n .$$

Обратный код числа *y* имеет вид:

$$y= y_0 y_1 \ldots y_{n\approx 1} y_n \,,\; y_i = p \approx 1 \approx x_i \,,\; i=0, 1, \ldots , n \,.$$

Так как по условию задачи $x=y$, то $y_i = x_i$, $i=0, 1, \ldots , n$. Отсюда получаем: $x_i = p \approx 1 \approx x_i$ или $x_i = (p\approx 1)/2$, $i=0, 1, \ldots , n$. Из последней системы равенств видно, что решение задачи возможно лишь в системах счисления с нечётным основанием (в противном случае x_i – не целое, что невозможно). У решения в любой системе счисления с нечётным основанием все цифры прямого и обратного кода совпадают.

2.7 Доказать, что $mod(x,2)=0$, если число $x=17C$ рассматривается в системе счисления с основанием p, $11<p<36$. Чем может быть объяснены задаваемые границы на число *p* (приведите одну убедительную причину)?

Решение. Так как $x = p^2 + 7p + 12 = (p + 3)(p + 4)$,

то один из этих двух последовательных множителей является четным т.е. $mod(x, 2)=0$. Границы *p* заданы для того, чтобы не выходить за пределы алфавита из десятичных цифр и букв латинского алфавита, используемых в этом примере и они могут быть легко изменяемы.

2.8 В факториальной системе счисления целые числа записывают как линейную комбинацию факториалов, например,

$$2457 = 2\times 3! + 4\times 2! + 5\times 1! + 7\times 0! \;(\; 0! = 1,\; n! = 1\times 2\times 3\times \ldots \times n)$$

а) Позиционна ли эта система и почему (обосновать)?
б) Записать формулу перевода из факториальной системы в систему десятичную.

Решение. а) Эта система (условно) позиционна. Так как $0!=1!=1$, то два младших разряда *n* – разрядного числа $x = x_n x_{n-1} \ldots x_2 x_1$ $(n>1)$ в разложении этого числа по факториалам представимы как $x_2 \times 1! + x_{1\times} 0! = x_{1\times} 1! + x_2 \times 0!$ и поэтому веса этих разрядов не зависят от позиции (поэтому при $n>1$ это число можно считать непозиционным лишь условно).

б) Формула перевода из факториальной системы счисления в десятичную систему:

$$x = x_n x_{n\approx 1} \ldots x_1 = x_{n\times}(n\approx 1)! + x_{n-1\times}(n\approx 2)! + \ldots + x_{1\times} 0! \,.$$

Задачи без решений

2a1. Имеются ящики: *4* черных, *3* красных, *2* желтых и *1* зеленый (посчитаны в десятичной системе). В каждом черном ящике – $(21)_p$ шара, красном – $(23)_p$ шара, жёлтом – $(23)_p$ шара, зелёном – $(111)_p$ шара. Определить основание *p* системы счисления, в которой были посчитаны шары, если всего было $(244)_p$ шаров. Чему равно общее число шаров в восьмеричной системе?

2a2. Числа $A=(100)_x$ и $B=(411)_y$ заданы, соответственно, в системах счисления с основаниями *x* и *y* $(0<x, y<17)$. Найти *x* и *y*, если известно, что в десятичной системе счисления *x* в *2* раза больше *y,* а если от *B* отнять *A,* то получаем десятичное число *7.* Найти *A* в шестнадцатеричной системе счисления.

2a3. Число $x=(176)_p$ (рассматриваемая в системе счисления с основанием *p,* $1<p<20$) делится нацело на *7.* Найти *p* (не перебирая все указанные *p*). Вычислить *x* в восьмеричной системе.

2a4. Какова разрядность двоичной системы счисления, в которой представим лишь диапазон чисел из интервала (*–255; 255*) ?

2a5. Найти числа, соответствующие понятиям "*нуль*" и "*бесконечность*" в *n*–разрядной системе счисления и сравнить их. Указать эти числа для двоичной, восьмеричной и шестнадцатеричной систем.

2a6. Сформулировать какие–то признаки делимости на *8* (на *11*) числа x записанного в системе счисления с основанием *p=12*, не переводя эти числа в десятичную систему. Рассмотреть другое значение *p.*

2a7. Доказать, что число $x=x_n x_{n\approx 1}...x_1 x_0$ (в системе счисления с основанием *p*) кратно числу $p \approx 1$ только тогда, когда $x_n x_{n\approx 1}...x_1 x_0$ кратно числу $p\approx 1$.

2a8. Найти число *x*, записанное в системе счисления с основанием *p*, если оно совпадает со своим дополнительным кодом. При каких *p* это возможно?

2a9. В факториальной системе счисления целые числа записывают как линейную комбинацию факториалов, например, число *2457* в этой системе:

$2457_! = 2\times 3! + 4\times 2! + 5\times 1! + 7\times 0!$ ($0! = 1$, $n! = 1\times 2\times 3\times ...\times n$)

Описать процедуры (формулы) переводов из десятичной системы в факториальную и обратно.

3. Алгебра высказываний и предикатов (*Задачи с решениями)*

3.1. Заданы два предиката вида *p* = "число *x* делится нацело на *5*" и *q*="*y* ≈ день недели". Найти множество истинности предикатов *p* и *q*, если $x\in$ {*1, 4, 6, 16, 20, 26, 30, 35*}, $y\in$ {*первый, вторник, среда, 1999*, *зима, выходной, праздник, воскресенье*}.

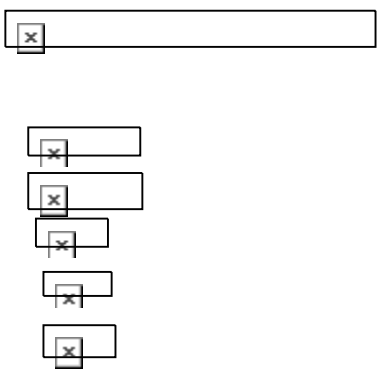
Решение. *D(p)={20, 30, 35}, D(q)={вторник, среда, воскресенье}*.

3.2 Изобразить множество истинности двухместного предиката вида *p(x,y)*="*модуль числа x равен модулю числа y*", если задана область изменения аргументов: *x, y*$\in$ [*0;1*].

Решение.

$E(p) = \{(x,y): |x|=|y|\} = \{(x,y): (x=y) \vee (x=\approx y)\}==\{(x,y): x=y\} \cup \{(x,y): x=\approx y\} = E(p_1) \cup E(p_2)$.

Смысл предикатов $p_1(x,y)$ и $p_2(x,y)$ очевиден. Множество изображается графиком функции $y=|x|$:

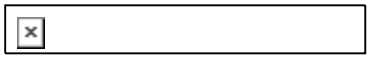


$\bar{x} \vee v = 1$

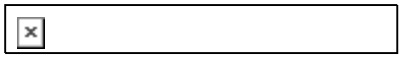
3.3 *Брауну, Джонсу* и *Смиту* предъявлено обвинение в соучастии в ограблении банка. В ходе следствия *Браун* сказал, что преступники были на синем "*Бьюике*", *Джонс* сказал, что это был черный "*Крайслер*", *Смит* утверждал, что это "*Форд*", но не синий. Все они указывали неправильно либо марку, либо цвет автомобиля. Определите цвет и марку автомобиля.

Решение. Рассмотрим простые высказывания вида: *x*="*машина ≈ синяя*", *y*="*машина ≈ Бьюик*", *z*="*машина ≈ черная*", *u*="*машина ≈ Крайслер*", *v*="*машина ≈ Форд*". Высказывание *Брауна* можно записать в виде сложного логического выражения вида , высказывание *Джонса* – , высказывание *Смита* – . Так как в каждом из этих выражений одна из переменных принимает значение 1 ("истина"), то истинны и дизъюнкции вида: , , . По определению конъюнкции, . Упростим логическое выражение:

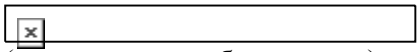
Здесь было использовано, что одновременно не могут быть истинными два высказывания относительно цвета или два высказывания относительно марки машины. Так как конъюнкция равна "*истина*" только тогда, когда *y*=1, *z* =1, =1, то заключаем, что автомобиль был *черным "Бьюиком*".



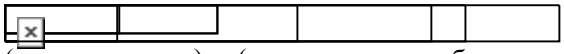
3.4 Вычислить значение логической функции *u*, если $x \vee y \vee z$=0,



Решение. Преобразуем функцию, используя аксиомы алгебры логики: (аксиома де Моргана) =



(аксиома дистрибутивности) =



(соотношение) = (аксиома дистрибутивности и соотношение вида) (соотношения и)

Из условия $x \vee y \vee z = 0$ по определению (таблице истинности) дизъюнкции получаем, что $x=0, y=0, z=0$. Следовательно, $u=1$.

Самое лучшее и короткое решение задачи тем не менее ≈ такое: так как $x \vee y \vee z=0$, то первое слагаемое в исходном выражении *u* равно *1*, поэтому все выражение $u=1$.

3.5 Можно ли данные функции $z = x \vee x \vee y \vee y$, $w = x \vee y \vee x \vee y$ реализовать одним и тем же наименьшим числом логических схем (вентилей) и почему ? Ответ кратко и полно обосновать.

Решение. Используя аксиомы де Моргана и поглощения получаем:

$z = x \vee x \vee y \vee y = x \vee x \wedge y \vee y = x \vee y$,
$w = x \vee y \vee x \vee y = x \vee y \wedge x \vee y = x \vee y$.

Следовательно, обе функции реализуются двумя одинаковыми вентилями: одним инвертором и одним дизъюнктором.

3.6 Упростить, используя как можно меньше аксиом алгебры предикатов и за как можно меньшее число их применений: $z = x \vee y \vee x \wedge y \vee x$.

Решение. Используем аксиому поглощения: $x \wedge y \vee x = x$. Применяя её к последним слагаемым, получим: $x \wedge y \vee x = x$. Итак, $z = x \vee y \vee x$. По аксиоме де Моргана $z = x \wedge y \vee x$. По аксиоме поглощения, получим: $z = x \wedge y \vee x = x$. Использованы 2 аксиомы (поглощения ≈ 2 раза и де Моргана ≈ 1 раз).

3.7 Упростить, используя минимум аксиом алгебры предикатов:

$$z = x \vee x \vee y \vee y \wedge x \wedge y .$$

Решение. В скобках будем записывать аксиому, с помощью которой из выражения слева получено выражение справа:

$z = x \vee x \vee y \vee y \wedge x \wedge y$ = (аксиома де Моргана) =
$= x \vee x \vee y \vee y \vee x \wedge y$ = (аксиома инволюции) =
$= x \vee x \vee y \vee y \vee x \wedge y$ = (аксиома де Моргана) =
$= x \wedge y \vee x \wedge y \vee x \vee y$ = (аксиома поглощения) $= 1 \vee x \vee y = 1$.

3.8 "*Черный ящик*" – устройство, содержимое которого неизвестно и может быть определено только по входу – выходу ящика. В "черном ящике" находится логическая схема, которая в ответ на последовательность входных логических констант выдает последовательность логических констант, получаемых после выполнения логической схемы из "черного ящика". Определить логическую функцию в "черном ящике", если операции выполняются с логическими константами для входных последовательностей (поразрядно). Схема "ящика" такова:

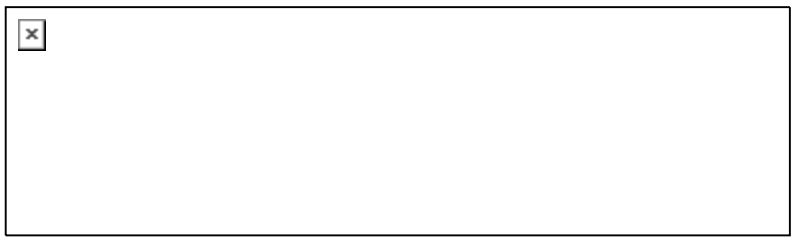
```
x=00011101
z=f(x,y) z=11100000
y=11100001
```

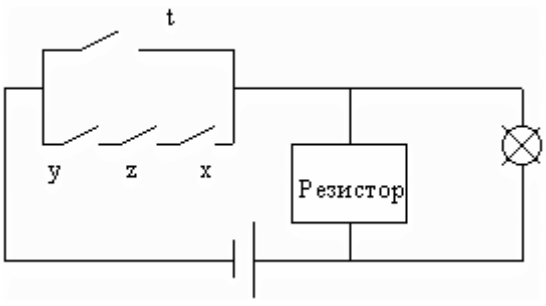
Решение. Из анализа входных значений x, y и поразрядного сравнения логических констант в этих сообщениях с константами в z ≈ результате выполнения функции в "черном ящике", видно, что подходит, например, функция вида $z=(x \vee y) \wedge x$. Действительно, при данных x и y :

$x \vee y = 00011101 \vee 11100001 = 11111101$,
$(x \vee y) \wedge x = 11111101 \wedge 11100010 = 11100000 = z$.

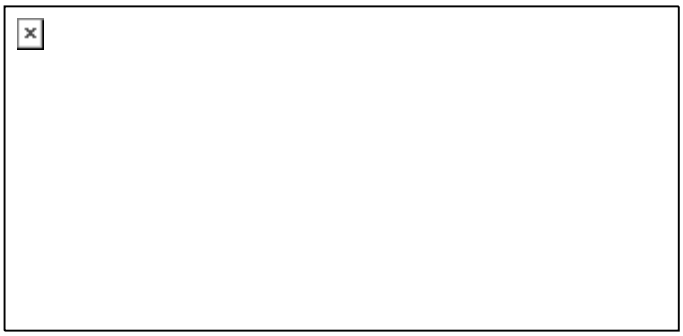
3.9 Построить логическую и электрическую схемы для функции $u = x \wedge y \wedge z \vee t$.

Решение. Искомые схемы имеют вид:





3.10 Определить логическую функцию *u=f(x,y,z,t)* реализуемую логической схемой вида:



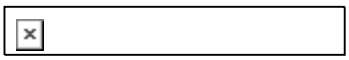
Решение. Искомая логическая функция имеет вид: $u = \overline{x \vee y} \vee (\bar{z} \wedge t)$.

Задачи без решений

3а1 Определить множество истинности предиката *p* на множестве *X*:

p(x)="число x кратно 4", X=[2; 10].

3а2. Составить таблицу истинности и затем упростить функцию:



3а3. Обсуждая свои возможности по поступлению в вуз абитуриенты *А*, *Б, В* высказали предположения: *А* ≈ "Я не смогу поступить, а *В* – поступит"; *Б*≈ "*В* не поступит, а *А* ≈ поступит"; *В* ≈ "Или я не поступлю, или *В* ≈ не поступит". После сдачи экзаменов выяснилось, что каждый высказал одно верное и одно ложное простое утверждения. Кто поступил в вуз ?

3а4. Построить логическое выражение определяемое таблицей истинности:

x	y	?	?	?	?	?
0	0	1	1	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	1	0	1	1	1	0

3а5. Подобрать три функции эквивалентные данной, но более простого вида (с меньшим числом операций и операндов):

$z = x \wedge y \vee x \vee y \wedge y \vee x$.

3а6. Из указанных ниже функций отметить (с обоснованием всех своих рассуждений) эквивалентные:

а) $z = x \vee x \vee y \vee y \vee x \wedge y \vee x \vee y \vee y \vee x \wedge y$;
б) $u = x \vee x \vee y \vee y \vee x \wedge y$;
в) $w = x \vee x \vee y \vee y \vee x \wedge y$;
г) $s = x \vee x \vee y \vee y \vee x \wedge y$;
д) $s = x \vee y \vee x \wedge y \vee y \vee x \wedge y$.

3а7. Упростить логическое выражение *z*, затем построить таблицу истинности и указать хотя бы одну функцию, эквивалентную данной:

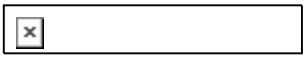
1) $z = x \wedge y \wedge y \vee x \vee y \wedge y \vee y$;
2) $z = x \wedge y \wedge x \vee y \vee x$;
3) $z = x \wedge y \wedge x \vee y \vee y$.

3а8. Определите логическое выражение внутри "черного ящика" вида:

x=11110001
z=f(x,y) z=01111100
y=11110000

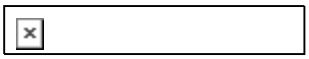
Всегда ли можно однозначно решать подобные задачи и почему?

3а9. Придумать один черный ящик, внутри которого находится двухместная функция и описать ее идентификацию.

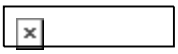


3а10. Построить логические схемы, соответствующие функциям:

а) ;



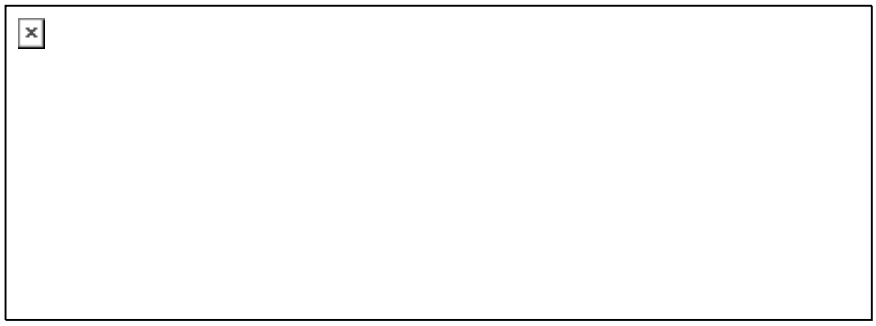
б) ;



в) .

Построить эквивалентные логические схемы с минимумом инверторов, конъюнкторов, дизъюнкторов. Покажите вычислениями, что построенная схема оптимальна по сравнению с эквивалентными ей по входу–выходу.

3а11. Определить логическую функцию, реализуемую схемой:



4. Алгоритмы и алгоритмизация (*Задачи с решениями)*

4.1 Формализовать (записать алгоритм) сложения двух целых положительных чисел и оценить число операций алгоритма.

Решение. Пусть числа заданы своими разрядами в виде: $a=a_n a_{n-1}...a_1 a_0$, $b=b_m b_{m-1}...b_1 b_0$, где a_i и b_j ≈ *i*–ый и *j*–ый разряды этих чисел. Разряды нумеруются справа налево. Алгоритм состоит из следующих пунктов:

1) Найти k – минимальное из n и m т.е. выполнить операцию $k:=min(n,m)$.

2) Положить $i=0$ (т.е. выбрать младший разряд у складываемых чисел).

3) Если $a_i+b_i<10$, то в *i*–ый разряд формируемой суммы $c=a+b$ записать цифру $c_i=a_i+b_i$, в противном случае – записать цифру суммы $c_i=a_i+b_i\approx10$.

4) Увеличить следующий, $(i+1)$– ый разряд числа a на 1 $(a_{i+1}:=a_{i+1}+1)$.

5) Если $i\geq k$, то проверить, был ли перенос в предыдущем пункте 3 (в разряде $i\approx1$) и, если он был, то в следующем разряде суммы записать цифру $c_i=a_i+1$, а все остальные разряды c_j положить равными a_j, $j=i+2, i+3, ... , n$ (при $k=m$) или записать цифру $c_i=c_i+1$, а все остальные разряды c_j положить равными b_j, $j=i+2, i+3,..., m$ (при $k=n$); если же $i<k$, то перейти к п.7.

6) Заменить текущее i на $i+1$ $(i:=i+1)$ и перейти к п.3.

7) Записать сумму: $c=c_n c_{n-1}...c_0$ (или же $c=c_{n+1}c_n...c_0$) – при $k=m$, или $c=c_m c_{m-1}...c_0$ (или же $c=c_{m+1}c_m...c_0$) – при $k=n$. Интересен этот пример и тем, что обычно и интуитивно просто исполняемая нами процедура оказалась не очень простой при её формализации (здесь полную формализацию, т.е. за-

пись процедуры на формальном языке, например, на учебном алгоритмическом мы не провели – оставляем это как упражнение для читателя).

4.2 Составить алгоритм выделения знака и всех цифр заданного натурального n–разрядного числа x. Цифры выдавать пользователю с младшего разряда. Составить достаточный на ваш взгляд набор тестов. Указать число, функцию, выражение или способ для оценки эффективности этого алгоритма. Как можно решить эту задачу, если *n* не задается? Изменится ли сложность алгоритма по выбранной вами оценке или оценкам при этом и почему?

Решение. Алгоритм имеет вид:

алг Выделение цифр (арг цел *n, x*);

дано | натуральное *n*–разрядное число *x*, которое может быть | записано в виде, например, своими цифрами: *x=x*[*1*]*x*[*2*]...*x*[*n*]

надо | выделить и вывести по мере выделения все цифры этого числа

нач | начало тела алгоритма

цел *i*, | переменная цикла выделения (номер выделяемой цифры)

x, | заданное число
y | выделенная очередная цифра

если *x>0* | число положительно (можно и через sign(x)) ?
то вывод ("знак числа: + ") | да, положительно
иначе | остальные случаи

если *x=0* | число равно нулю ?
то вывод ("число *0*") | да, это нуль
иначе вывод ("знак числа: – ") | нет, отрицательно

все | закончили с неположительными числами
все | закончили со всеми числами

нц для *i* от *1* до *n* | заголовок цикла для выделения цифр
y:=*mod*(*x*,*10*) | выделяем последнюю цифру текущего

| значения числа *x* (оно по ходу будет изменяться так, как это указано ниже)

вывод (*y*) | выдача очередной цифры числа

x:=*div*(*x*,*10*) | заменяем *x* числом, получаемым из *x* "отбрасыванием" последней цифры: | она была уже выше найдена

кц | конец тела цикла выделения цифр
кон | конец тела алгоритма .

Набор тестов должен включать, как минимум, тесты типа:

а) все разряды числа *x* – равны (в том числе и нулю);
б) среди разрядов есть равные между собой (в том числе и идущие последовательно);
в) случай *n=1*.

Функцию эффективности можно строить исходя из различных критериев эффективности, например, из числа операций, операндов, команд и т.д. В частности, показателем эффективности может служить простая интегральная оценка вида:

1) операций присваивания – *3n* (*2* явные операции в теле и одна неявная – в заголовке цикла и они повторяются *n* раз);

2) операций проверки простых предикатов – *n+2* (в заголовке цикла, неявно, – *n* и явно в условных командах – *2*);

3) вычислений функций – *2n*;

4) логическая сложность, например, оцениваемая по глубине вложения циклов (равна *1*) и условных команд (равна *2*).

Если *n* неизвестно, то для решения задачи вместо цикла типа "до" можно использовать цикл типа "пока":

n:=0 | начальное значение счетчика разрядов

нц пока *x>0* | пока "есть цифры в числе"

y:=mod(x,10) | выделяем очередную цифру
x:=div(x,10) | "отбрасываем" найденную цифру
n:=n+1 | переходим к следующей цифре

вывод (*n*,"-я цифра:", *y*) | выдача очередной цифры

кц | конец цикла поиска следующих цифр в числе .

Оценки 2) – 4) при этом не изменяются, а число присваиваний станет равно *3n+1* (все явные). Это демонстрирует тот факт, что недостаток информации для решения задачи (в исходной форме) может увеличивать хаос и сложность.

4.3 Составить эффективный алгоритм решения квадратного уравнения (с наименьшим числом операций умножений, возведений в степень, делений, сложений, вычитаний, операндов, переменных и констант).

Решение. Алгоритм РКУ – Решение Квадратного Уравнения имеет следующий вид:

алг РКУ (арг вещ *a, b, c* рез лит *s*, вещ *x1, x2*)

дано | квадратное уравнение с коэффициентами *a, b, c*
надо | решить оптимальным способом это уравнение

нач | начало тела алгоритма

если *a=0* | является ли уравнение квадратным на самом деле?
то если *b=0* | является ли уравнением на самом деле?
то если *c=0* | является ли соотношение тождеством?
то *s*:="*Уравнение выродилось в тождество 0*=0"

иначе *s:="Такого уравнения не существует*"
все | конец проверки вырождения уравнения

иначе *s:="Уравнение – уравнение первой степени*"

x1:=≈c/b | решаем полученное уравнение
все | конец проверки параметров уравнения

иначе *x1:= – b/(2*a)* | вспомогательное выражение

*x2:=x1*x1≈ c/a* | выражение для дискриминанта

если *x2>0* | существуют ли два различных корня ?
то *s:="Уравнение имеет два различных корня*" | да
x2:=sqrt(x2) | оптимизированный дискриминант

x1:=x1 ≈ x2 | находим первый корень
*x2:=x1+2*x2* | находим второй корень

иначе | нет (двух различных корней)

если *x2=0* | есть два равных корня ?
то *s:="Уравнение имеет 2 равных корня*" | да

x2:=x1 | находим второй равный корень
иначе *s:="Уравнение не имеет решений*" | нет

все | конец проверки различных корней
все | конец проверки наличия двух корней уравнения
все | конец проверки всего параметров заданного уравнения

кон | конец всего алгоритма .

Легко видеть, что затрачено меньше операций и операндов, чем в классически используемом алгоритме. (Подсчитайте эти числа для данного и классического алгоритма и сравните их для каждой операции и операндов).

4.4 Заданы натуральные числа *m* и *p*. Составить эффективный алгоритм для получения чисел $x_1, x_2, \ldots, x_n$ по правилу:

x_1 – остаток от деления *m* на *p*,
x_2 – остаток от деления произведения $m*x_1$ на *p*,
x_3 – остаток от деления произведения $m*x_2$ на *p* и т. д.

Решение. Алгоритм имеет вид:

алг Последовательность (арг цел *n, p, m*, рез цел таб *x*[*1: n*])

дано | натуральные *n, m, p*

надо | найти *x*[*i*]– остатки от деления *m***x*[*i–1*] на *p, i=1,2,...*

нач | начало тела алгоритма

цел *i* | переменная цикла вычисления искомых чисел

i:=1 | начальное значение счетчика количества чисел

x[*1*]*:=mod(m,p)* | вычисление первого числа

нц пока (*i<n+1*) и (*x[i]>0*) | цикл вычисления чисел

i:=i+1 | переход к новому числу

x[*i*]*:=mod(x*[*i–1*]**m, p)* | вычисление нового числа

кц | конец цикла

кон | конец алгоритма .

В алгоритмах не использующих проверку второго условия заголовка цикла могут быть произведены лишние, бесполезные повторения цикла, поэтому такие решения не могут являться полными и правильными (по условию задачи).

4.5 Числа $x_1, x_2, \ldots, x_n$ упорядочены по убыванию. Составить алгоритм вставки в этот ряд заданного числа без нарушения этой упорядоченности. Составить достаточный на ваш взгляд набор тестов для тестирования этого алгоритма.

Решение. Алгоритм имеет вид:

алг Вставка (арг цел *n*, вещ *y,* таб *x*[*1:n*], рез вещ таб *x*[*1: n+1*])

| возможно и такое описание: арг ... вещ ... таб *x*[*1: n+1*]

дано | упорядоченный по убыванию ряд чисел *x*[*1*], *x*[*2*],...,*x*[*n*]

| число *y* (все числа вещественные)

надо | поставить *y* в ряд, не нарушив упорядоченности ряда

нач | начало тела алгоритма и описания переменных

цел *i*, | переменная цикла просмотра всего ряда

j, | переменная для фиксирования места *y* в ряде

s | литерная переменная для признака: найдено ли место?

s:="не найдено" | начальное значение "флага" *s*

i:=1 | начальное значение номера испытуемого места

если (*x*[*1*]<*y*) | искомое место – начало ряда

то *j:=0* | сдвигать все элементы с номера *j=1*

иначе если (*x*[*n*]>*y*) | искомое место – конец ряда

то *j:=n* | сдвигать не нужно

иначе | вставка "внутрь" ряда

нц пока (*s*="*не найдено*") и (*i*<=*n*) | цикл просмотра если(*x*[*i*]>=*y*) и (*y*>=*x*[*i+1*]) | если найдено место *y* то *s*:="*найдено*" | то изменяем *s* и затем *j:=i* | фиксируем это место *y*

все | закончено испытание места номер *i*

i:=i+1 | переход к испытанию нового места

кц | конец цикла просмотра

все | конец внутренней условной структуры

все | конец внешней условной структуры

i:=n | определяем позицию для сдвига

нц пока (*i*>=*1*) и (*x*[*i*]<*y)*

x[*i*+1*]:=x*[*i*]

i:=i–1

кц

x[*i+1*]*:=y* .

кон | конец алгоритма .

Набор тестов должен включать, как минимум, тесты вида:

а) все числа x_i ($i=1,2,...,n$) равны (в том числе и нулю);
б) среди чисел есть равные (в том числе и в разных местах ряда);
в) $n=1$;
г) $y>x_1$ и $x_n>y$ ($y \geq x_1$ и $x_n \geq y$).

4.6 Составить алгоритм генерирования случайного десятичного целого положительного числа так, чтобы его изображение в системе счисления с основанием *p* было бы *n*–разрядным. Все разряды получаемого числа должны быть значащими.

Решение. Пусть $a=x_0x_1...x_{n-1}x_n \approx$ изображение этого числа в системе счисления с основанием *p*. Тогда верно разложение: $a = x_0p^n + x_2p^{n-1} + ... + x_{n-1}p + x_n$.

Чтобы цифра x_0 была значащей, необходимо выполнение условия: $0<x_0<p$, $0 \leq x_i < p$, $i = 1, 2, ... , n$.

Рассмотрим наиболее простой и понятный алгоритм, не ставя пока задачу нахождения самого эффективного решения задачи. Будем генерировать все x_i с помощью функции *rnd(p≈1),* но будем брать целую часть. Под *rnd(p≈1)* понимается датчик случайных чисел из интервала *(0;p≈1)*. При этом, если x_0 окажется равным *0,* то заменяем его, например, на *1*. Обычный датчик случайных чисел из *(0;1)* или *rnd(1)* легко трансформируется в *rnd(p≈1)* по формуле *rnd(p≈1)=(p≈1)*rnd(1).* Разряды будем сохранять в массиве *x*, а затем найдем *a* по схеме Горнера. Итак, алгоритм имеет вид:

алг Датчик (арг цел *p, n,* рез цел *a*)
дано | натуральные *p , n*
надо | найти случайное *n* – разрядное целое число *a*

нач | начало тела алгоритма

цел *i*, | переменная цикла генерирования разрядов числа *a*

таб *x*[*0: n*] | массив генерируемых разрядов числа *a*

a:=int(rnd(p≈1)) | генерируем старший разряд числа

если *a=0* | проверяем, он значащий или нет?

то *a=1* | если – не значащий, то меняем его

| (сразу брать равным *1* – "не интересно")

все | конец этой проверки

x[*0*]*:=a* | запоминаем это число

нц для *i* от *1* до *n* | цикл для схемы Горнера

x[*i*]*:=int(rnd(p≈1))* | генерируем, запоминаем разряд *a*

*a:=a*p+x*[*i*] | вычисляем *a* по схеме Горнера

кц | конец генерирования числа

кон | конец алгоритма .

4.7 Составить алгоритм вычисления номеров элементов заданного ряда в упорядоченном ряде из этих чисел без сортировки ряда.

Решение. Алгоритм имеет вид:

алг Номера (арг цел *n*, вещ таб *x*[*1*:*n*])

дано | ряд вещественных чисел *x*[*1*], *x*[*2*], ..., *x*[*n*]

надо | найти номера *x*[*i*], *i=1,2,...,n* в упорядоченном по возрастанию

| ряде из этих чисел, не меняя ряд т.е. сохраняя ряд – оригинал

нач | описания вспомогательных переменных

цел *j*, | *j*–номер рассматриваемого текущего члена ряда *x*[*j*]

i, | *i* – номер члена ряда, с которым сравнивается *x*[*j*]

m, | *m* – номер *x*[*j*] в упорядоченном ряде

k | *k* – число членов ряда равных *x*[*j*]

нц для *j* от *1* до *n* | цикл нахождения номера каждого элемента

m:*=1* | в начале искомый номер полагаем равным *1*

k:*=0* | в начале число членов ряда равных *x*[*j*] полагаем *0*

нц для *i* от *1* до *n* | цикл сравнения текущего *x*[*j*] с *x*[*i*]

если *x*[*j*]=*x*[*i*] | если эти два числа равны

то если *j*>*i* | и *x*[*j*] находится после *x*[*i*],

то *k*:=*k+1* | то увеличиваем *k* на *1*

все | конец проверки на равенство

иначе | если же эти два числа не равны

если *x*[*j*]<*x*[*i*] | и "за *x*[*j*] находится *x*[*i*] "

то *m*:=*m+1* | то увеличиваем *m* на *1*

все | конец проверки на неравенство

все | общий конец проверок на (не)равенство

кц | конец цикла нахождения номера для текущего *x*[*j*]

m:=*m*+*k* | искомый номер равен числу расположенных за

| *x*[*j*] больших или равных *x*[*j*] элементов ряда

вывод ("Номер *x*["; *j*;"] равен "; *m*) | выдача номера *x*[*j*]

кц | конец цикла нахождения номеров всех членов ряда

кон | конец алгоритма .

Этот не самый оптимальный, но "понятный" алгоритм – пример решения задачи с ограничениями на ресурсы (на операционную среду). Ограничения ресурсов могут изменить сложность (часто и сам алгоритм). С этой позиции можно сравнить этот алгоритм с алгоритмом сортировки, например, обменами и оценить их по одним и тем же критериям.

4.8 Составить алгоритм нахождения наибольшего *m* и наименьшего *k* из целочисленной последовательности чисел $x_1,x_2,...,x_n$, но только среди элементов на чётных местах и только до тех пор, пока не встретится нечётный элемент (первый в ряду).

Решение. Так как нам необходимо перебирать только элементы на чётных местах, то цикл типа "до" (без задаваемого шага по переменной цикла, точнее – с шагом равным *1*) нас не устраивает и поэтому необходимо использовать цикл типа "пока". Алгоритм имеет вид:

алг Мин_мах_из_ряда_до_нечёт(арг цел *n*, цел таб *x*[*1:n*], рез цел *m, k*)

дано | ряд целых чисел *x*[*i*], *i=1, 2, ..., n*

надо | найти минимум *m* и максимум *k* среди нечётных элементов данного ряда, но только среди элементов на чётных местах

нач | начало алгоритма и описания вспомогательных переменных

цел *i* | *i* – номер текущего чётного места

i:=2 | начальное значение *i* полагаем первому чётному номеру
k:=x[*2*] | начальное значение минимального элемента ряда
m:=x[*2*] | начальное значение максимального элемента ряда

нц пока (*i<= n*) и (*mod(x*[*i*],*2)=0*) | цикл перебора всех (первый | предикат заголовка) элементов на чётных местах до первого | нечётного ("пока идут чётные" – | второй предикат цикла)

m:=max(m,x[*i*]*)* | выбираем наибольший из текущего элемента на чётном месте и из наибольшего среди всех | предшествовавших ему таких же элементов

k:=min(k,x[*i*]*)* | выбираем наименьший из текущего элемента на чётном месте и из наименьшего среди всех | предшествовавших ему таких же элементов

i:=i+2 | переход к следующему элементу на чётном месте ряда

кц | конец цикла перебора требуемых по условию элементов ряда

кон | конец алгоритма.

4.9 Составить алгоритм последовательной перестановки всех элементов строки номер *m* и элементов столбца номер *k* заданной матрицы *A* из *n* строк и *n* столбцов.

Решение. Необходимо осуществить обмен элементов: *a*[*m,1*] с *a*[*1,k*], *a*[*m,2*] с *a*[*2,k*], ..., *a*[*m,n*] с *a*[*n,k*]. Как нетрудно видеть, такую перестановку можно осуществить с помощью одного цикла от *1* до *n*, в котором одинаково ("синхронно") изменяются первый индекс (для строки) и второй индекс (для столбца) элементов. Алгоритм имеет вид:

алг Обмен(арг цел *n,m,k,* вещ таб *a*[*1:n,1:n*],рез вещ таб *a*[*1:n,1:n*])

дано | матрица A из элементов $a[i,j]$, $i=1, 2, ..., n; j=1, 2, ..., n$

надо | переставить элементы строки номер m с элементами столбца k

нач | начало алгоритма и описания вспомогательных переменных
цел i, | i – номер текущего элемента m–ой строки (k–го столбца) A

b | b – "буферная" переменная для обмена двух элементов

нц для i от 1 до n | цикл перебора элементов строки m (столбца k)

$b:=a[m,i]$ | запоминаем в "буфере" b текущий элемент строки
$a[m,i]:=a[i,k]$ | элемент строки заменяем элементом столбца
$a[i,k]:=b$ | элемент столбца меняем на элемент строки (точнее на его "копию" сохранённой в буфере b)

кц | конец цикла перебора строки m и столбца k
кон | конец алгоритма.

4.10 Дана квадратная матрица $x[1:n,1:n]$. Составить алгоритм нахождения значения такого элемента и его индексов, который является наибольшим среди элементов главной (ведущей из левого верхнего в правый нижний угол матрицы) и побочной (ведущей из правого верхнего угла в нижний левый угол) диагоналей.

Решение. Элементы главной диагонали имеют совпадающие индексы строки и столбца: $j=i$. Элементы побочной диагонали также имеют общее свойство: если индекс строки такого элемента равен i, то индекс столбца равен $j=n–i+1$. Таким образом, зная номер строки любого элемента этих двух диагоналей, можно "автоматически" определять и номер столбца, на пересечении которых стоит этот элемент. Поэтому при работе с этими элементами нет необходимости организовывать второй цикл – по столбцам матрицы, достаточен по строкам (в котором вычисляем индекс столбца j). С учётом этого алгоритм имеет вид:

алг Макс_элем_диагон(арг цел n, вещ таб $x[1:n,1:n]$, рез цел k,l,вещ m)
дано | квадратная матрица x размерности n строк и n столбцов
надо | найти максимальный элемент m среди элементов главной и побочной диагоналей матрицы, а также его индексы k и l

нач | начало алгоритма и описание вспомогательных переменных
цел i, | i – переменная цикла по строкам матрицы

$m:=x[1,1]$ | начальное значение максимального элемента – это значение "первого встречного" элемента матрицы; | его потом можем изменять при нахождении более | "достойной кандидатуры" – большего его элемента

$k:=1$ | номер строки исходного "первого встречного"
$l:=1$ | номер столбца исходного "первого встречного"

нц для i от 1 до n | цикл перебора всех строк матрицы

если $m<x[i,i]$ | если старое значение m меньше значения текущего элемента главной диагонали, то
то $m:=x[i,i]$ | меняем его значение на значение текущего элемента и фиксируем

$k:=i$ | номер его строки и
$l:=i$ | номер его столбца

если $m<x[i,n–i+1]$ | если старое значение m меньше значения текущего элемента побочной диагонали, то
то $m:=x[i,n–i+1]$ | меняем его значение на значение текущего элемента и фиксируем

$k:=i$ | номер его строки и
$l:=n–i+1$ | номер его столбца

все | конец сравнения текущих элементов с m

кц | конец цикла перебора строк матрицы
кон | конец алгоритма.

4.11 Дана квадратная целочисленная матрица *x*[*1:n,1:n*]. Найти среднее арифметическое нечётных элементов, расположенных на и над главной диагональю (т.е. включая и элементы диагонали).

Решение. Нетрудно заметить, что на и над главной диагональю все элементы *x*[*i,j*] обладают следующим свойством: индекс столбца элемента больше или равно индексу строки, т.е. если индекс *i* строки таких элементов может меняться от *i=1* до *i=n,* то индекс *j* столбца любого элемента строки *i* в указанном множестве элементов может изменяться соответственно от *j=i* до *j=n* (от *i=1* идёт начало из–за того, что элементы самой диагонали включаются в рассматриваемое множество). Таким образом, нет никакой необходимости перебирать все *n*n* элементов матрицы – достаточно перебирать *n*(n+1)/2* элементов (почему?), что при больших *n* значительно может сократить число операций. С учётом сказанного, алгоритм примет вид:

алг Среднее_арифм_нечёт (арг цел *n,* цел таб *x*[*1:n,1:n*], рез вещ *s*)

дано | квадратная матрица *x* размерности *n* строк и *n* столбцов

надо | вычислить *s* – среднее арифметическое нечётных элементов, расположенных на и над главной диагональю матрицы

нач | начало алгоритма и описание вспомогательных переменных

цел *i*, | *i* – переменная цикла по строкам матрицы

j, | *j* – переменная цикла по столбцам матрицы
k | *k* – счётчик числа положительных элементов по условию
s:=0 | начальное значение суммы элементов по условию
k:=0 | начальное значение количества элементов по условию

нц для *i* от *2* до *n* | цикл перебора всех строк матрицы
нц для *j* от *i* до *n* | цикл рассмотрения всех элементов текущей строки – начиная от элемента строки на главной диагонали и до конца этой строки

если *mod*(*x*[*i,j*],*2*)*=1* | если текущий элемент нечётен,
то *s:=s+x*[*i,j*] | то изменяем старое значение суммы *s*

k:=k+1 | и увеличиваем значение *k* на *1*

все | конец проверки на нечётность текущего элемента
кц | конец цикла перебора всех строк матрицы
кц | конец цикла перебора элементов текущей строки

если *k>0* | если был найден хоть один элемент по условию задачи
то *s:=s/k* | то находим среднее арифметическое (без этого условия возможно деление на нуль!);

все | конец нахождения среднего арифметического
кон | конец алгоритма.

4.12 Составить алгоритм подсчёта числа несовпадающих символов на одинаковых позициях в двух заданных текстах *a* и *b*. Длины текстов не задаются.

Решение. Алгоритм имеет вид:

алг Подсчёт несовпадений в текстах (арг лит *a*, *b*, рез цел *k*);

дано | тексты *a* и *b* произвольной допустимой длины

надо | подсчитать *k* – число несовпадающих символов на одинаковых позициях в текстах *a* и *b*.

нач | начало тела алгоритма.

цел *i*, | *i* – переменная цикла сравнения символов на одинаковых позициях в текстах *a* и *b*

j | *j* – минимальное из длин (в символах) *a* и *b*

k:=*abs*(*длина*(*a*)–*длина*(*b*)) | начальное значение счётчика полагаем равным числу символов, на которое текст *a(b)*длиннее *b(a)*

j:=*min*(*длина*(*a*),*длина*(*b*)) | выбираем минимум из длин *a* и *b*.
нц для *i* от *1* до *j* | заголовок цикла для сравнения символов *a,b*, стоящих на одинаковых, *i*-х позициях *a,b*

если *a*[*i*] ≠ *b*[*i*] | если они не равны, то
то *k*:=*k*+*1* | увеличиваем счётчик *k* на *1*.

всё | конец условной структуры
кц | конец цикла
кон | конец алгоритма.

4.13 Составить алгоритм нахождения в заданном тексте числа подтекстов заданной длины. Все подтексты данного текста разделены только пробелами.

Решение. Алгоритм имеет вид:

алг Число подтекстов (арг цел *d*, лит *t*, рез цел *n*)

дано | задан текст *t* состоящий из подтекстов разделённых пробелами

надо | найти число подтекстов заданной длины *d* в заданном тексте *t*

нач | начало тела алгоритма

| описание типов используемых в теле алгоритма переменных

цел *i*, | переменная цикла для перебора всех символов текста *t*

m, | переменная подсчёта символов выделяемого подтекста

сим *x* | переменная, которой присваивается текущее значение

| выбранного символа из текста *t*

m:=*0* | начальное значение счётчика символов в выделяемом

| подтексте полагаем равным нулю, так как на данный

| момент времени в подтексте число символов равно *0*

нц для *i* от *1* до *длина*(*t*) | заголовок цикла перебора символов *t*

| тело цикла выделения подтекста

x:=*t*[*i*] | присваиваем вспомогательной

| переменной *x* значение символа,

| стоящего на *i*–ом месте в тексте *t* и

| "вырезаемого" из текста *t*

если *x*≠"" | если символ *x* не пробел,

то *m*:=*m*+*1* | то количество символов в текущем

| подтексте увеличивается,

иначе | иначе (*x*–пробел) – выполняем

| блок сравнения длины подтекста с

| заданной длиной *d*

если *m*=*d* | если длина подтекста равна *d*, то

то *n*:=*n*+*1* | увеличиваем число подтекстов

всё | конец проверки длины текста

m:=*0* | обнуляем длину подтекста, для

| выделения новых подтекстов остатка *t*

всё | конец условной структуры

кц | конец цикла

кон | конец алгоритма.

4.14 Составить алгоритм вставки в заданный текст *t* заданного текста *p* с заданной позиции *n* т.е. первый символ текста *p* должен стать *n*-м символом нового текста (образуемого после вставки).

Решение. Алгоритм имеет вид:

алг Вставка текста в позицию (арг цел *n*, лит *t, p*)

дано | текст *t*, вставляемый текст (подтекст) *p*, позиция вставки *n*

надо | вставить в текст *t* с позиции номер *n* заданный текст *p*

нач | начало тела алгоритма

| описание типа вспомогательной переменной

цел *m* | *m* – переменная запоминающая число символов в *t*

m:= *длина*(*t*) | находим число символов в *t*, включая и пробелы

t:=*t*[*1*:*n–1*]+*p*+*t*[*n*:*m*] | "вырезаем" из *t* его подтекст *t*[*1*:*n–1*] от

| первого до (*n–1*)–го символа, а затем к

| нему "цепляем" вставляемый

| текст *p* и затем к полученному "цепляем"

| подтекст *t*[*n*:*m*] текста *t* от *n*–го до

| последнего символа данного текста

кон | конец алгоритма.

4.15 Составить алгоритм удаления из заданного текста *t* его подтекста от заданной позиции *n* до заданной позиции *m*.

Решение. Алгоритм имеет вид:

алг Удаление подтекста (арг цел *n,m*, лит *t*)

дано | текст *t*, позиция *n* начала и позиция *m* конца удаляемого текста

надо | удалить из текста *t* его подтекст от позиции *n* äî ïîçèöèè *m*

нач | начало тела алгоритма

| описание типа вспомогательной переменной

цел *k* | *k* – переменная запоминающая число символов в *t*

k:= *длина*(*t*) | находим число символов в *t*, включая и пробелы

t:=*t*[*1*:*n–1*]+*t*[*n*:*m*] | "вырезаем" из *t* его подтекст *t*[*1*:*n–1*] от

| первого до (*n–1*)–го символа, а затем к

| к этой части текста *t* "цепляем"

| подтекст *t*[*m+1*:*k*] текста *t* от *m+1*–го до

| последнего символа данного текста

кон | конец алгоритма.

4.16 Задан текст длины не более *255* символов. Составить алгоритм эффективного (по числу команд, операций, операндов) нахождения текста, записанного в обратном порядке.

Решение. Алгоритм имеет вид:

алг Инвертирование текста (арг лит *a*)

дано | произвольный текст *a* (*длина*(*a*)<=*255)*

надо | записать *a* в обратном порядке (наиболее эффективно)

нач | начало тела алгоритма

цел *i*, | переменная цикла

n, | вычисляемая длина текста

сим *b* | переменная для запоминания одного из переставляемых

| симметричных относительно середины элементов

n:=*длина*(*a*) | вычисляем длину текста

если *n>1* | если "есть, что переставлять", то

то нц для *i* от *1* до *int*(*n*/2) | цикл перестановки

b:=*a*[*i*] | запоминаем текущий символ
a[*i*]:=*a*[*n–i+1*] | на его место – симметричный,
a[*n–i*+1]:=*b* | затем передвигаем запомненный элемент на место симметричного

кц | конец цикла
все | конец условной структуры
кон | конец алгоритма.

4.17 Определить функцию фрагмента алгоритма и записать ее кратко; определить значение каждого элемента массива *a* после выполнения этого фрагмента при заданных начальных условиях: *n=4*, $a_1=2$, $a_2=-6.5$, $a_3=2.8$, $a_4=0$. Фрагмент этого алгоритма:

нц для *i* от *1* до *n*

если *a*[*i*]>*0*
то *a*[*i*]:=*1*
иначе если *a*[*i*]<*0*

то *a*[*i*]:=*–1*
иначе *a*[*i*]:=*0*

все
все
кц .

Решение. Алгоритм заменяет каждое число из последовательности чисел $a_1, a_2, \ldots, a_n$ его знаком, т.е. выполняет те же функции, что и функция *sign*(*x*) по отношению к числам $x=a_i$, $i=1,\ldots,n$. После выполнения фрагмента: $a_1=1$, $a_2=-1$, $a_3=1$, $a_4=0$.

4.18 Выполнить трассировку алгоритма поиска индекса наибольшего числа с нечетным индексом в целых числах $x_1, x_2, \ldots, x_n$.

алг Поиск (арг цел *n*, цел таб *x*[*1*:*n*], рез цел *k*, лит *s*);

дано | последовательность целых чисел *x*[*1*], *x*[*2*],..., *x*[*n*]

надо | найти индекс *k* максимального числа *x*[*k*], *k=1,2,...,n*

нач | описания типов вспомогательных переменных

цел *i*, | *i* – вспомогательная переменная (цикла)

вещ *y* | *y*– вспомогательная переменная текущего максимума

s:= "*задача не имеет решения*" | начальное состояние *s*

k:=*0* | начальное состояние *k* (не определено)
y:=*x*[*1*] | начальное состояние *y* (определено)
i: =*1* | начальное состояние *i* (определено)

нц пока (*i*<=*n*) | заголовок цикла поиска *k*
если (*y*<=*x*[*i*]) | проверка условия превышения очередного элемента *x*[*i*] текущего значения *y*

то | если было превышение, то

y:=*x*[*i*] | фиксируем большее значение в *y*
k:=*i* | и его индекс *i* в *k*

все | конец проверки превышения

i:=*i+2* | переходим к следующему элементу

кц | конец цикла

если *k>0* | проверка положительности решения и

то *s*:="*задача имеет решение*" | замена значения *s*

все | конец проверки положительности

кон | конец алгоритма.

Решение. Произведем трассировку этого алгоритма при входных данных: $n=5$, $x_1=7$, $x_2=2$; $x_3=1$; $x_4=10$; $x_5=18$ с помощью таблицы; *s*="*нет* (*да*)" соответствует "*задача уже решена (еще не решена)":*

i	s	k	y	y<=x[i]	i<=n
1	нет	1	7	да	да
3	нет	1	7	нет	да
5	нет	5	18	да	да
7	да	5	18	да	нет

4.19 Составить алгоритм подсчёта числа совпадающих символов на одинаковых позициях в двух заданных текстах *a* и *b*. Длины текстов не задаются. Как можно уточнить условие задачи для того, чтобы эта задача стала бы корректной при реализации на ЭВМ (т.е. поставить эту задачу как корректную задачу на программирование).

Решение. Алгоритм имеет вид:

алг Подсчёт совпадений в текстах (арг лит *a*, *b*, рез цел *k*);

дано | тексты *a* и *b* произвольной допустимой длины

надо | подсчитать *k* – число совпадающих символов на одинаковых в *a* и *b*.

нач | начало тела алгоритма.

цел *i*, | *i* – переменная цикла сравнения символов

| на одинаковых позициях в текстах *a* и *b*

j | *j* – минимальное из длин (в символах) *a* и *b*

k:=*0* | начальное значение счётчика полагаем равным нулю

j:=*min*(*длина*(*a*),*длина*(*b*)) | выбираем минимум из длин *a* и *b*.

нц для *i* от *1* до *j* | заголовок цикла для сравнения символов *a,b*,

| стоящих на одинаковых, *i*-х позициях *a,b*

если *a*[*i*] = *b*[*i*] | если они равны, то

то *k*:=*k+1* | увеличиваем счётчик *k* на *1*.

всё | конец условной структуры

кц | конец цикла

кон | конец алгоритма .

При реализации задачи на ЭВМ нужно оговорить условие на структуру представления текста в памяти ЭВМ, например, можно считать длину текстов не более *255* символов или представить их динамическим массивом (списком или, например, считыванием текста по частям с винчестера).

Задачи без решений

1. В последовательности чисел каждый элемент (начиная со второго) получен по единому строгому правилу. Составить алгоритм для вычисления любого произвольного члена последовательности при заданном его порядковом номере *n: 0, 2, 6, 14, 24, ...* .

2. Составить алгоритм определения, стоит ли в заданном числе *y* на заданной *n*–ой позиции (в разряде номер *n*) заданная цифра *x*. Рассмотреть отдельно два варианта: а) количество цифр числа задаётся; б) количество цифр числа не задаётся.

3. Составить алгоритм, который для заданных целых чисел *a, b, c* позволяет, расставляя между ними знаки +,≈,× находить хотя бы одно арифметическое выражение с *a, b, c*, значение которого равно числу *x*. Рассмотреть отдельно два случая: а) можно использовать только по одному знаку операции каждого типа в одном и том же выражении; б) можно использовать любое количество знаков каждого типа.

4. Составить алгоритм размена заданной суммы купюрами в *100,500,1000, 5000, 10000* и *50000* руб. Рассмотреть отдельно два случая: а) количество имеющихся купюр задано (ограниченно); б) количество купюр неограниченно.

5. Составить алгоритм вычисления значения максимально допустимого в системе счисления с основанием *p* и с максимальной разрядностью *n* целого числа т.е. числа соответствующего понятию "*положительная бесконечность*" в этой системе счисления. Сформулировать и рассмотреть затем аналогичную задачу для чисел с плавающей запятой.

6. Составить наиболее эффективный по числу команд, операций, операндов алгоритм решения квадратного уравнения, если для корней уравнения задана требуемая точность их вычисления (использовать вспомогательный алгоритм вычисления корня с заданной степенью точности).

7. Составить алгоритм выдачи всех цифр и знака заданного целого *n*–разрядного числа *x*. Цифры выдавать со старшего разряда. Составить достаточный набор тестов для обоснования работоспособности алгоритма.

8. Составить алгоритм нахождения количества простых чисел, заключённых между двумя задаваемыми вещественными числами *a* и *b*.

9. Составить алгоритм нахождения суммы и произведения всех цифр данного вещественного числа, если целая часть числа состоит из *n* цифр, а мантисса числа ≈ из *m* цифр.

10. "Квадрат–палиндром" ≈ число, являющееся квадратом натурального числа и читаемое слева и справа одинаково, например, *9, 121, 484*. Составить эффективный (по количеству команд, операций, операндов, времени, памяти) программу поиска всех таких чисел, не превосходящих заданного числа. Составить минимально достаточный, ≈ на ваш взгляд, набор тестов для тестирования программы.

11. Составить наиболее оптимальный (сформулировать различные критерии оптимальности самому) алгоритм, отвечающий на вопрос: образует ли заданная последовательность *n* вещественных чисел $x_1, x_2, ..., x_n$ арифметическую или геометрическую прогрессию (и какую именно).

12. Составить алгоритм нахождения заданного количества наибольших по модулю элементов заданной последовательности вещественных чисел, а также их индексов.

13. Составить алгоритм, который данную последовательность точек на плоскости (т.е. пар координат) располагает по убыванию расстояний между ними.

14. В заданную последовательность вещественных чисел вида $x_1<x_2<...<x_n$ вставить новую произвольную последовательность вещественных чисел $y_1,y_2,...,y_m$ и определить среднее арифметическое всех положительных элементов получаемой новой последовательности. Новый массив (кроме указанных) не вводить. Составить алгоритм.

15. Из заданной последовательности натуральных чисел вида $x_1, x_2,...,x_n$ удалить все числа начинающиеся на заданную цифру a и оканчивающихся на заданную цифру b и запомнить удаляемые числа в отдельном массиве y. Новый массив (кроме указанных x, y) не вводить. Составить алгоритм.

16. Задана последовательность из n вещественных чисел. Составить алгоритм нахождения таких двух равных элементов этого ряда, между которыми заключена возрастающая подпоследовательность наибольшей "длины".

17. Составить алгоритм, который находит сумму и произведение элементов целочисленной матрицы до тех пор, пока не встретится самый первый элемент матрицы, заданный данному целому числу.

18. Составить алгоритм определения наибольшего из сумм строк, сумм столбцов, сумм двух диагоналей квадратной матрицы.

19. Составить алгоритм сортировки по убыванию или по возрастанию (по желанию пользователя алгоритма) элементов задаваемой строки или столбца (по желанию пользователя) заданной матрицы.

20. Составить алгоритм упорядочивания по возрастанию максимальных во всех строках матрицы элементов.

21. Составить алгоритм, который нечетные строки матрицы меняет с последующими четными строками.

22. Заданы два одномерных массива A и B различной размерности. Составить алгоритм выделения всех несовпадающих (неравных) между собой элементов на одинаковых местах массивов A и B в отдельный массив C.

23. Составить алгоритм нахождения наибольшего из суммы элементов в верхней треугольной части (над главной диагональю) и произведения элементов нижней треугольной части заданной квадратной матрицы.

24. Составить алгоритм запоминания (или ввода элементов) квадратной симметричной матрицы в более компактном (оптимальном по числу запоминаемых элементов) виде.

25. Составить алгоритм нахождения суммы и произведения всех чисел в матрице натуральных чисел из n строк и m столбцов, пока не встретится строка с наибольшим (во всей матрице) числом чётных элементов. Вычисление суммы и произведения осуществлять по строкам матрицы (от первой строки ко второй и т.д.).

26. Составить алгоритм подсчета количества вхождений каждого из символов некоторой заданной литерной строки в другую заданную литерную строку. Повторяющиеся несколько раз вхождения считать только один раз.

5. Типы и структуры данных (*Задачи с решениями)*

5.1. Описать полно все основные стандартные типы данных, т.е. не описываемые в алгоритме специально.

Решение. 1) Тип "***целые числа***". Имя типа данных: цел. Область определения типа: целые числа $i\in Z$. Разрешенные для типа операции: :=, +, –, *, **, = , ≠ , <, >, ≥ , ≤ . Функции от аргумента типа: *int, mod, abs, sqrt, sign, sin, cos, tg, ctg, arcsin, arccos, arctg, arcctg. ln, lg, exp, div, rnd.*

2) Тип "***вещественные числа***". Имя типа данных: вещ. Область определения типа: вещественные числа $r\in R$.. Разрешенные для типа операции: :=, +, –, *, **, = , ≠ , <, >, ≥ , ≤ . Функции от аргумента типа: как в типе цел, кроме *mod, div.*

3) Тип "***символьная константа***". Имя типа данных: сим. Область определения типа: литеры (символы) языка. Разрешенные для типа операции: :=, +, = , ≠ , <, >, ≥ , ≤ . Функции от аргумента типа: *длина* , [ ] (*вырезка, сечение*).

4) Тип "***литерная константа***". Имя типа данных: лит. Область определения типа: последовательность литер (символов). Разрешенные для типа операции: :=, +, = , ≠ , <, >, ≥ , ≤ . Функции от аргумента типа: *длина*, [ ] (*вырезка, сечение*).

5) Тип "***логическая константа***". Имя типа данных: лог . Область определения типа: *1* ("*истина*"), *0* ("*ложь*"). Разрешенные для типа операции: ∧ , ∨ , ¬ , = , ≠ , <, >, ≥ , ≤ . Функции от аргумента типа: отсутствуют.

5.2 Дана некоторая абстрактная структура данных *ñ èìåíåì Список_учеников_школы,* состоящая из определенного числа учеников. Привести примеры корректных абстрактных операций для структуры.

Решение. Разрешены, например, абстрактные операции: *добавить ученика; удалить ученика; выдать атрибуты ученика*.

5.3 Пусть *x* – переменная типа цел, *y* – типа вещ, *s* –типа лог. Определить неправильные (с указанием ошибок) команды:

а) *z:=4*x+100;* б) *y:=max(z,100)<y*;
в) *s:=(x<0)∨(y=4);* г) *x:=mod(x,y).*

Решение. Команда а) неверна, логической переменной *z* будет сделана попытка присвоения целочисленного значения; б) неверна, делается попытка присвоения вещественной переменной *y* логического значения (значение отношения неравенства – "*истина*" или "*ложь*"); в) – верное присваивание; г) неверно, аргумент *y* функции *mod* – не целый.

5.4 Описать абстрактные типы данных: *День*, *День_недели, Жилище, Религия*.

Решение. *День=(пасмурный, светлый, теплый, холодный, дождливый);*
День_недели=(понедельник, вторник, среда, четверг, пятница, суббота, воскресенье);
Жилище=(дом, квартира, шалаш, пещера);
Религия=(буддизм, ислам, христианство); .

5.5 Задана некоторая фирма, у которой *n* сотрудников. Составить алгоритм определения оклада заданного сотрудника (по заданным его Ф.И.О).

Решение. Выберем следующую структуру данных*: Ф.И.О.; отдел фирмы*, где он работает; *телефон отдела* и *оклад сотрудника* и примем следующие типы и форматы этих данных*:*

Ф.И.О. – литерная строка не более *30* символов, включая пробелы, например, *Николаев Петр Иванович*;
отдел – целое двухразрядное число, например, отдел *06*;
телефон – литерная строка не более *10* символов, например, *8–88–90*;
оклад (в руб.)– целое, не более *5*–разрядное число, например, *500* руб.

Алгоритм запишем на учебном алгоритмическом языке:

алг Оклад(алг цел *n*, лит *f*);
дано | описанная выше структура данных по каждому из *n* сотрудников
надо | определять оклад заданного сотрудника (по его Ф.И.О.)

нач | начало тела алгоритма
лит *x*, | *x* – переменная "Ф.И.О.",
f, | *f* – задаваемый сотрудник (его Ф.И.О.)

цел *i,* | *i* – счетчик цикла поиска

j, | *j* – "флаг", признак того, что запись найдена

сим таб Сотрудник[*1:n*], | Сотрудник – массив Ф.И.О. сотрудников

цел таб Оклад[*1:n*]; | Оклад – массив окладов сотрудников

i:=1; | начальное значение номера записи
j:=0; | начальное значение флага: *j=0* – запись не найдена

нц пока (*i*≤*n*) или (*j=0*) | проверяем условие выполнения цикла
если Сотрудник[*i*]=*f* | проверка нахождения сотрудника
то *вывод* ("Оклад "; *f* ;" равен: "; Оклад[*i*]);
j:=1; | изменяем значение флага для окончания цикла
иначе *i:=i+1* | если сотрудник не найден – к следующему
все кц кон

Задачи без решений

5а1. Составить абстрактную структуру данных *Выпускаемая_продукция*, в которой отразить данные: год выпуска, код продукции, выпускающая страна. Отобразить затем ее на некоторую структуру памяти ЭВМ.

5а2. Составить абстрактную структуру данных *Абитуриент* с атрибутами: факультет, группа, домашний адрес, телефон, год рождения, учебное заведение, баллы на экзаменах и указать операции, определенные для этой структуры.

5а3. Задана группа из *n* учеников и их оценки. Составить алгоритм нахождения всех учеников, имеющих оценки ниже среднего по группе и всех отличников по задаваемой дисциплине.

5а4. Задана группа из *n* учеников и их оценки по *m* предметам. Составить алгоритм выдачи учеников: а) занимающихся на "*4*" и "*5*"; б) имеющих áîëåå òð¸õ îöåíîê "*2*"; â) èìåþùèõ õîòü îäíó "*2*" ïî çàäàííîìó ïðåäìåòó.

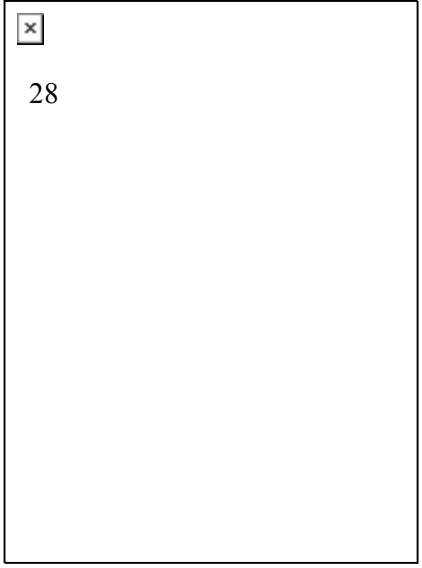
5а5. Заданы текстовые массивы *A* (эталон диктанта), *B* (диктант ученика) из не более чем в *n* предложений, каждое из предложений – не более, чем из *m* знаков. Составить алгоритм и структуру данных проверки диктанта для каждого из *k* учеников. Выдавать фамилию и инициалы ученика и его оценку.

5а6. Задана группа из *n* учеников и их оценки по *m* предметам. Составить алгоритм выдачи учеников занимающихся выше средней успеваемости по группе.

6. Исполнители алгоритмов (*Задачи с решениями)*

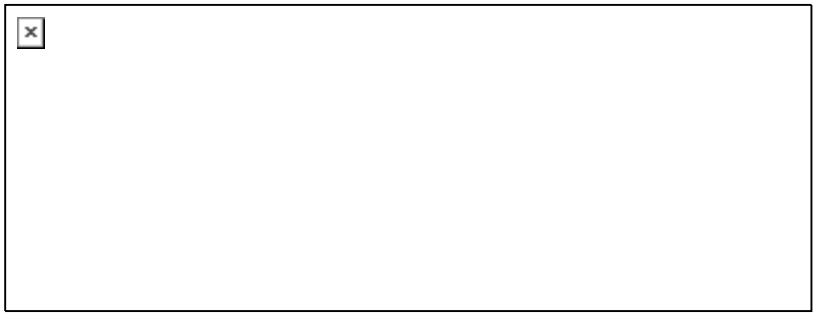
1. Описать функционирование автомата для продажи газированной воды.

Решение. Введем множества (входное – *X*, выходное – *Y*, состояния – *S*): *X={1, 3, Г, ∅}, Y={B, C,0}, S={S_0, S_1, S_2, S_3},* где ниже обозначены следующие события: *1*– "опустить 1 коп.", *3*– "опустить 3 коп.", *Г* – "опустить гнутую монету", *B* – "выдача воды газированной без сиропа", *C* – "выдача газированной воды с сиропом", *0* – "отказ выдать воду", ∅ – "монета не опущена", S_0 – "начальное состояние", S_1– "первое рабочее состояние (или обработка *1* коп.)", S_2 – "второе рабочее состояние (обработка *3* коп.)", S_3 – "состояние неисправности". Работу автомата можно описать графом вида:



2. Нарисовать структуру исполнителя – машины Тьюринга, выполняющей некоторую программу над алфавитом $X=\{x_1, x_2, ..., x_{10}\}$.

Решение.



3. Исполнитель "*Чертёжник*" может выполнять команды: "*налево(a)"*≈ поворот налево на угол в *a* радиан; "*направо(a)"* ≈ поворот направо на *a* радиан; "*вперед(a)"* ≈ движение вперед на *a* см; "*назад(a)"*≈ движение назад на *a* см; "*поднять*" ≈ поднять (от бумаги) карандаш; "*опустить*" ≈ опустить конец карандаша для рисования. Составить последовательность команд "*Чертежника*", которая позволяет нарисовать рисунок (эскиз) квадрата со стороной *5* см. на плоскости. Начальное положение исполнителя≈ начало координат, а его начальный "взгляд"≈ по лучу оси *x* (*x>0*).

Решение. Последовательность (программа для исполнителя) такова: *опустить*

вперед(5)
направо(π /2)
вперед(5)
направо(π /2)
вперед(5)
направо(π /2)
вперед(5);
поднять .

Как видно отсюда, эту последовательность можно записать и лучше:

i:=*1*
опустить
нц пока *i*<5
вперед (5)
направо (π /*2*)
i:=*i+1*
кц
поднять

Задачи без решений

1. Исполнитель "*Робот*" может выполнять команды: "*вперед*(*a*);" √ вперед на *a* см.; "*назад*(*a*);"≈ назад на *a* см. (*a* ≈ вещественное); "*поворот*(*a*,*b*);≈ поворот на угол в *a* радиан направо (при *b=1*) или налево (при *b=0*); "*поднять*;"≈ поднять груз; "*опустить*;" ≈ опустить груз. Записать наименьшую последовательность команд "*Робота*" для того, чтобы можно было переместить грузы, расположенные в вершинах квадрата со стороной в *x* см. и в серединах его сторон в центр квадрата. Каждый груз можно поднимать только за *1* раз. Составить и решить новую задачу для данного исполнителя. Модифицировать команды "*Робота*" таким образом, чтобы они позволяли ему двигаться с различной скоростью и брать несколько грузов сразу. Решить для этого исполнителя одну новую задачу.

2. Исполнитель "*Чертежник"* может выполнять команды: "*налево(a);"*– поворот налево на угол в *a* радиан; "*направо(a);"* – поворот направо на *a* радиан; "*вперед(a);"* – движение вперед на *a* см.; "*назад(a);"* – движение назад; "*поднять;"* – поднять от бумаги карандаш; "*отпустить;"* – отпустить на бумагу конец карандаша для начала рисования. Составить наименьшую последовательность команд "*Чертежника",* которая позволяет нарисовать рисунок (эскиз) куба со стороной *d* см. на плоскости. Все ребра куба должны быть видны. Начальное положение "*Чертежника*" – начало координат, а его начальный "взгляд" – в направлении луча *Ox (x>0)*. Составить и решить новую задачу для этого исполнителя, модифицировать операционную среду исполнителя.

3. Исполнитель "*Робот"* может выполнять команды: "*вперед(a);"* – вперед на *a* см.; "*поворот(a);"* – поворот на угол в *a* радиан направо; "*поднять;"* – поднять груз; "*опустить;"* – опустить груз. Записать наименьшую последовательность команд "*Робота"* для того, чтобы можно было переместить грузы, расположенные в вершинах квадрата со стороной *d* см. и в серединах его сторон в центр квадрата. Каждый груз поднимать только за один раз. Начальное положение "*Робота*" – начало координат, а его начальный "взгляд" – в направление луча *Ox (x>0)*. Составить и решить новую задачу, модифицировать операционную среду исполнителя.

7. Основы компьютера (*Задачи без решений)*

1. Имеется трехадресная ЭВМ с кодом операции сложения равным *02*; рассмотреть выполнение команды *02 150 102 010* в ЭВМ, которая хранится в ячейке памяти номер *105* так, как это показано ниже:

адрес		операция	операнды операции		
105		02	150	102	010
поле адреса ячейки		поле кодовой части	поле первого адреса	поле второго адреса	поле третьего адреса

Решение. После считывания из *105*–ой ячейки памяти информации вида *02150102010* (представленной в двоичном виде, по полям!), ЭВМ определяет, что нужно сложить (код операции – *02*!) число из *150*–ой ячейки памяти с содержимым *102*–ой ячейки и полученную сумму записать в ячейку с номером *010*.

2. Запишите числа 13_{10}, 17_{10}, -31_{10} в формате целых чисел в восьмиразрядную ячейку памяти, изобразив ее схематически.

Решение. Старший разряд ячейки памяти (нулевой) будет содержать бит знака числа: *1*≈ если число отрицательно, *0*≈если число положительно. Учитывая, что $13_{10}=1101_2$, $17_{10}=10001_2$, $-31_{10}=-11111_2$, получаем:

0	0	0	0	1	1	0	1	,	0	0	0	1	0	0	0	1	,	1	0	0	1	1	1	1	1

3. Записать в формате с фиксированной между шестым и седьмым разрядами запятой числа -13.75_{10} и 16.34_{10} в тринадцатиразрядную ячейку памяти.

Решение. Записывая знак числа в старший, нулевой разряд ячейки памяти и учитывая, что $-13.75_{10}=1101.11_2$, $16.34_{10} = 10000.0101011100001\ldots_2$ (перевод точно нельзя сделать!), получаем:

0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	1	0	1	1	1	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	0	0	0	1	0	1	0	1

4. Записать в формате с плавающей запятой числа 90_{10}, 6.25_{10}, -0.0023_{10}, если в тринадцатиразрядной ячейке памяти знак числа и знак порядка записываются, соответственно, в нулевой и первый старшие разряды ячейки, мантисса записывается в разряды *2–8,* порядок ≈ в разряды *9–12*.

Решение. Учитывая, что указанные числа в нормализованной форме √ в форме с плавающей запятой записываются в виде: а) $90 = 0.9\times 10^{2}$, б) $6.2_{10} = 0.625\times 10^{1}$, в) $-0.0023 = -0.23\times 10^{-2}$, а мантиссы чисел в 2–ной системе будут равны: а) $0.9_{10} = 0.1110011100..._2$, б) $0.625_{10} = 0.101_2$, в) $0.23_{10}=0.0011101011100..._2$, получаем следующие представления чисел для указанных в условии задачи ячеек памяти:

(см. исходник!)

5. Некоторая трехадресная ЭВМ имеет, например, команды вида: *01*– сложить содержимое адресов *1* и *2* и результат поместить в адрес *3; 02* – переслать содержимое адреса *3* в адрес *2; 03* – обнулить содержимое адреса *1; 04* – сравнить содержимое адресов *1* и *2* и, если они равны, перейти к выполнению команды, извлекаемой из ячейки, номер которой указан в адресе *3*. Расшифровав команды ЭВМ, определить результат выполнения программы приведенной ниже, если ячейки *100,101,102* содержат, соответственно, числа *5,10,20*. Программа: *01100102103, 02101104103, 01104101104, 03101103102, 04101103104*.

Решение. Команда *01100102103* означает: "Сложить содержимое ячейки номер *100* с содержимым ячейки номер *102* и результат записать в ячейку номер *103*"; в результате выполнения этой команды в ячейке *103* будет записано число *25=5* (содержимое ячейки *100*)+*20* (содержимое ячейки *102*). Команда *02101 104103* означает: "Переслать содержимое ячейки номер *103* в ячейку номер *104*" ; в результате выполнения этой команды ячейка номер *104* будет содержать копию содержимого ячейки *103*, т.е. число *25*, содержимое ячеек *103* и *101* – не изменяются. Следующая команда – *01 104101 104* означает: "Сложить содержимое ячеек номер *104* и *101* и результат поместить в ячейку номер *104*"; в результате выполнения этой команды в ячейке номер *104* будет уже находиться число *25 + 10 = 35*. Команды *03 101 103 102* означает: "Обнулить содержимое ячейки номер *101*; в результате выполнения этой команды, текущее содержимое ячейки *101*, т.е. *10* будет заменено на ноль. Команда *04101 103104* означает: "Если содержимое ячейки номер *101* равно содержимому ячейки номер *103*, то перейти к выполнению команды, номер которой находится в ячейке *104*, т.е. команды извлекаемой из ячейки *104*"; в данном случае это условие не выполнено.

6. Какие байты образуют полуслова, слова, двойные слова, если заданы байты с адресами (номерами) ячеек *120–130, 301–305*.

Решение. Так как адрес *120* левого байта кратен двум, то байты *120–121*, *122–128, 124–125, 126–127, 128–129* образуют *5* полуслов с адресами *120*, *122, 124, 126, 128*. Далее, байты *120–123, 124–127* образуют слова с адресами *120, 124* соответственно, а байты *120–127* образуют двойное слово с адресом *120*. Имеем полуслова из ячеек *302–303, 304–305* (адрес полуслова *302* – не кратен *4*!), двойного слова из этих байтов нет.

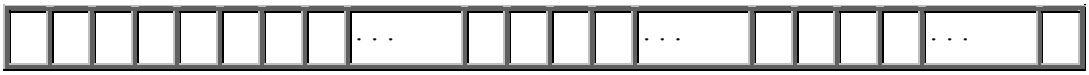
7. Какой минимально разрядности должна быть ячейка памяти у условной двухадресной ЭВМ, если ее память однородна и ЭВМ имеет набор из *50* различных операций и *4* мегабайт памяти; адресуется только машинное слово из двух байт?

Решение. Чтобы адресовать $4M=2^{22}$ байт памяти или 2^{21} машинных слов необходимо иметь адреса от *0* до $2^{21}–1=11111111111111111111$ т.е. под произвольный адрес нужно *21* разрядов. Всего для двухадресной ЭВМ нужно иметь $2\times 21 =42$ разрядов под адресную часть команд. Так как ЭВМ имеет *50* различных операций, то под кодовую часть команды необходимо не менее 6 разрядов*:* $100000_2=32_{10}<50_{10}<63_{10}=111111_2$. Следовательно, ЭВМ должна иметь ячейку памяти, которая, – в случае хранения в них команд, была бы разрядности *48=42+6*.

8. Спроектировать память условной трехадресной ЭВМ фон–Неймана (найти минимально достаточную разрядность, определить структуру ячейки ОЗУ и регистров, сделать рисунок), если ячейки и все регистры ЭВМ – одинаковой длины (память ЭВМ – однородна), ЭВМ реализует *n* различных машинных операций, ОЗУ ЭВМ равно 2^m Мегабайт, а адресуется всегда слово из двух байт (три адреса – для двух операндов и результата операции).

Решение. Чтобы адресовать 2^m (М) = 2^{20+m} (байт) памяти или 2^{19+m} машинных слов необходимо иметь адреса от *0* до $2^{19+m}\approx 1=111...111$ (*19+m* единиц) т.е. под произвольный адрес нужно *19+m* двоичных разрядов. Всего же для рассматриваемой трехадресной ЭВМ нужно иметь *57+3m* разрядов под адресную часть команд. Так как ЭВМ имеет *n* различных операций, то под кодовую часть команды необходимо отвести не менее $[log_2(n)] + 1$ или же *int(ln(n)/ln(2))+1* разрядов. Следовательно, "наша ЭВМ" должна иметь ячейку памяти, которая, – в случае хранения в них команд (имеющих кодовую и адресную части), была бы разрядности равной *57+3m+ int(ln(n)/ ln(2)) + 1*.

Структура ячейки (регистра) при *m=0, n=30* (адресуется *1* М, а ЭВМ имеет *30* различных машинных команд) изображена ниже:



0 1 2 3 4 5 6 7 23 24 42 43 61

Код операции	*Адрес 1–го операнда*	*Адрес 2–го операнда*	*Адрес результата*

Задачи без решений

1. Какое максимальное положительное и минимальное отрицательное число можно записать в 32–разрядную ячейку памяти?

2. Какой минимальной разрядности должна быть ячейка памяти у условной трехадресной ЭВМ с однородной памятью, если она имеет набор из *50* различных команд и *2* мегабайта адресуемой памяти, а адресуется байт.

3. Какова максимальная память (ОЗУ) ЭВМ, если ее разрядность равна восьми, число выполняемых команд – не более ста, адресность всех типов команд равна двум? Привести подтверждающие краткие расчеты.

4. Возможен ли теоретически процессор ЭВМ фон–Неймановского типа с быстродействием 10^{16} операций в секунду и вмещающийся в квадратную микросхему не более *1* кв. см?

5. Укажите содержимое ячеек *100, 101, 102, 103, 104* после выполнения команд трехадресной условной ЭВМ, если она имеет следующие команды: *01* – сложить содержимое ячеек, указанных в адресах первого и второго операнда и результат поместить в ячейку адреса номер *3; 02* – обнулить содержимое ячейки первого адреса*; 03* – если равны содержимое ячеек первого и второго адресов, то закончить программу безусловно, а если они не равны, то перейти к следующей команде; *04* – переслать число из ячейки, номер которой указан в первом адресе, в ячейку по второму адресу. В начале выполнения программы, ячейка номер *100* содержит число *5*. Программа имеет вид: *04100101101, 01101100100, 02104101101, 01104104104, 04104103103, 01101103103, 03104103103, 01101103102* .

8. Алгоритмические языки и методы трансляции (*Задачи с решениями)*

1. Отметить основные (в историческом плане) языки.

Решение.

1) ФОРТРАН, язык научно–технических расчетов, разработан в 1955 году фирмой IBM;
2) АЛГОЛ или АЛГОЛ–60, язык вычислительного характера, разработан в 1960 году Международным комитетом ученых;
3) ЛИСП, язык символьных вычислений (для преобразований со списками), разработан в Массачусетском технологическом институте в 1960–1961 годах;
4) КОБОЛ, язык для экономических расчетов и бизнеса, разработан в 1961 году в США по заданию правительства страны;
5) БЕЙСИК, язык для начинающих, разработан в 1965 году в Дортмунде;
6) ПЛ–1, разрабатывался как универсальный язык фирмой IBM в 1964–1968 г.г.;
7) ПАСКАЛЬ, разрабатывался как язык учебного и исследовательского характера в 1969 году в Цюрихе;
8) С, разработан лабораторией BELL (США, 1972 г.) в качестве языка поддержки и программирования известной операционной системы UNIX;
9) АДА, разработан как язык моделирования по заданию Пентагона в 1979 году;

10) ПРОЛОГ, язык для логического программирования а также для преобразований с деревьями (язык ЭВМ пятого поколения), разработан в 1971–1973 годах.

2. Записать на учебном языке выражение

Решение: *y=2**(n**2)+(x–2*sin(x+π))/exp(x)/x*

3. Перевести из записи на учебном языке в математическую:

*w = sqrt(b*b+c*c+2/a*x/(c+g))*

Решение:

4. Вычислить значение выражения:

*b=ln(exp(3))+max(min(3,2.5),3.6)+ mod(13,4)*int(–1 .5)*

Решение. Результат: *b=3 + max(2.5,3.6) + 1*(–2) = 3 + 3.6 – 2= 4.6*.

5. Указать порядок выполнения операций в выражении

*A = 3*ln(x)/y**2*(x–1)+exp(y).*

Решение. Вычисления подвыражений (выражения) происходит в порядке:

*x–1, ln(x), exp(y), y**2, 3**ln(x), 3*ln(x)/y**2, 3*ln(x)/y**2*(x–1), A.*

Задачи без решений

1. Определить указанные функции через другие операции и функции:

*max(x,y), min(x,y), abs(x), sign(x), int(x), div(x,y), mod(x,y), x**y.*

2. Даны описания переменных: цел *x, y,* вещ *a, b, c,* сим *u*, лит *s*, лог *r.* Указать все неправильные места, причины (условия появления) этих ошибок в выражениях:

а) *y:=sin(x)*s≈длина(s)+длина(u)≈int(x/y)+mod(x,a)≈div(sign(x),int(b));*
б) *a:=exp(u)+s≈sqrt(x+y):(длина(u)≈1);*
в) *r:=(x>y)+(a<длина(s))*.

3. Вычислить значение выражения:

а) *y:=sin(arcsin(1))*длина(sort)+int(2.6/5)+mod(123,3)≈div(5,int(4.8));*
б) *a:=exp(ln(4))+sqrt(6.25+sign(sin(0)))*.
в) *x = ln(exp(3))+max(min(6,3.5),3.8)+mod(13,4)*int(≈4.8)≈*
abs(≈6) div(int(5.99), 3* sign(4.7))*.

Указать порядок вычисления всех объектов в этих выражениях.

4. Записать условие попадания точки *A(x,y)* в область ограниченную окружностью с центром в точке *B(a,b)* и радиусом *r* и полосы, ограниченной параллельными прямыми, проходящими через точки *C(0;b), D(a;2b)* и *M(a;0), N(2a;b)* соответственно. При каких *r* это условие примет значение "истина", а при каких – "ложь".