**ЧАСТЬ 1**

**Задание №1**

**Дано натуральное число. Верно ли, что цифра a встречается в нём более k раз?**

|  |  |
| --- | --- |
| Дано: | С изменениями:  **program** zadanie1;  **var** n, a, k, i: integer;  **begin**  readln(n,a,k);  i:=0; // инициализация переменной  **while** n>0 **do** // изменено условие  //добавлено служебное слово do  **begin**  **if** a = n **mod** 10 **then** inc(i);  n:=n **div** 10  **end**;  writeln (i);  **if** i > k **then**  writeln ('YES')  **else**  writeln ('NO')  **end**. |

Добавлено: 1) инициализация переменной i 2) служебное слово **do**

Изменено условие продолжения цикла:

Было: **while** (n=0) стало: **while** n>0 **do**

цикл продолжается, пока в числе n есть цифры (n>0)

Убраны лишние операторные скобки

**Задание №2**

**Дано натуральное число. Установить, является ли последовательность его цифр при просмотре их справа налево упорядоченной по возрастанию.**

|  |  |
| --- | --- |
| Дано: | С изменениями:  **program** zadanie2;  **var** n, a, flag: integer;  **begin**  readln(n);  a:=n **div** 10;  writeln (a);  flag:=1; // изменение инициализации переменной  **while** a<>0 **do**  **begin**  **if** a **mod** 10 < n **mod** 10 **then** // изменено условие  flag:=0; // изменено значение  n:=n **div** 10;  a:=a **div** 10;  **end**;  writeln (flag);  **if** flag =1 **then**  writeln ('YES')  **else**  writeln ('NO')  **end**. |

Изменено:

1) начальное значение переменной flag:

Было: flag:=0 стало: flag:=1

Т.е., предположим, что в заданном числе последовательность его цифр припросмотре их справа налево **является упорядоченной по возрастанию** (flag:=1)**.**

2) условие в ветвлении:

Было: a **mod** 10 > n **mod** 10

стало: a **mod** 10 < n **mod** 10

3) значение переменной flag по ветке **then**

Было: flag:=1 стало: flag:=0

Как только встретим цифру слева меньшую цифры справа, то flag:=0, т.е. данные цифры не упорядочены по возрастанию припросмотре их справа налево.

**Задание №3**

**Дан одномерный массив из 15 элементов. Переставить в обратном порядке элементы, расположенные между вторым и десятым (т.е. с 3 по 9).**

|  |  |
| --- | --- |
| Дано: | С изменениями:  **program** zadanie3;  **var** i: integer; //убраны лишние переменные  mas: **array**[1..15] **of** integer;//изменен  //размер массива до 15 чисел  **begin**  **for** i := 1 **to** 15 **do**  readln(mas[i]);  **for** i := 1 **to** 2 **do**  writeln(mas[i]);  **for** i := 9 **downto** 3 **do** //изменен  //заголовок цикла  writeln(mas[i]);  **for** i := 10 **to** 15 **do** //изменено конечное  //значение счетчика цикла  writeln(mas[i]);  **end**. |

Изменено:

1. Из описания величин убраны лишние
2. Размер массива уменьшен до 15 элементов:

Было: **array**[1..16]

стало: **array**[1..15]

1. Изменен заголовок цикла:

Было: **for** i := 9 **to** 3 **do**

стало: **for** i := 9 **downto** 3 **do**

Теперь счетчик цикла изменяется от большего значения 9 к меньшему 3 с шагом -1

4) в последнем цикле конечное значение счетчика изменено на 15:

Было: **for** i := 10 **to** 16 **do**

стало: **for** i := 10 **to** 15 **do**

Т.к. в нашем массиве 15 элементов

ЧАСТЬ 2

Задание 2

**program** zadanie22;

**var** n, d, k, i: integer;

**begin**

readln(n,d);

k:=0;// счетчик количества делителей обнулим

**if** d<n **then**

**begin**

k:=1; //само число - делитель самого себя, поэтому k присваиваем единицу;

//если d больше или равно N, то очевидно, что делителей числа N,больших d не будет

**if** d+1 <= n **div** 2 **then** //если условие не выполняется, то оставшийся делитель, больший d, это само число N

**for** i := d+1 **to** n **div** 2 **do** //цикл от d+1 до половины n

**begin**

**if** n **mod** i =0 **then** //если i делитель

k:=k+1; // то счетчик увеличим на 1

**end**

**end**;

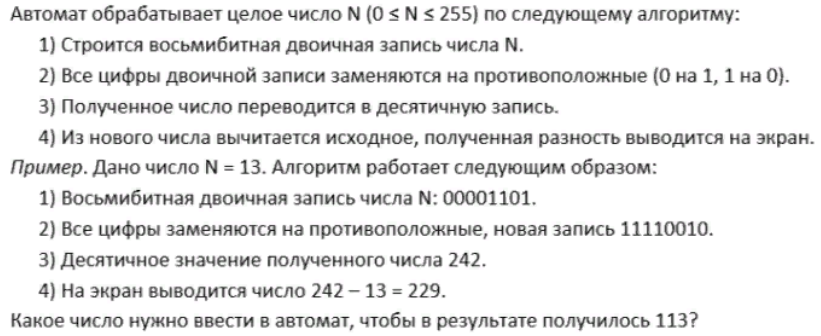
writeln (k);

**end**.

//Можно было облегчить себе жизнь и сразу пройтись по циклу for i от 1 до n div 2,и если число N делится без остатка на i,а также число N больше d,

то увеличиваем счётчик делителей на 1. Но тогда программа не была бы оптимизирована по времени.

**Задание 3.**



Решение

Инверсия двоичной восьмибитной записи числа в сумме с исходным числом дает 11111111, то есть 255. Следовательно, если исходное число равно *N*, то инвертированное число равно 255 − *N*. Затем автомат осуществляет вычитание, вычисляя 255 - 2*N*.

Поэтому, чтобы найти число, которое нужно ввести в автомат для получения 113, нужно решить уравнение:

255 -2*N* = 113

2*N* = 142

*N* = 71

Проверка:

1) 7110= 010001112  2) инвертируем: 10111000 3) 101110002 = 18410 4) 184 – 71 = 113

Ответ: в автомат нужно ввести число **71**, чтобы в результате выполнения алгоритма получилось **113**

**ЗАДАНИЕ 4**

**var**

i, j, t, s, n, sum, maxi: integer;

a: **array** [1..1000] **of** integer;

**begin**

read( s, n);

**for** i := 1 **to** n **do**

read( a[i]);

**for** i := 1 **to** n **do**

**for** j := i + 1 **to** n **do**

**if** a[i] > a[j] **then**

**begin**

t := a[i];

a[i] := a[j];

a[j] := t;

**end**;

sum := 0;

maxi := 1;

**for** i := 1 **to** n **do**

**if** sum + a[i] <= s **then begin**

sum := sum + a[i];

maxi := i;

**end**;

t := a[maxi];

**for** i := maxi **to** n **do**

**if** ((sum - t) + a[i]) <= s **then begin**

sum := sum - t + a[i];

t := a[i];

**end**;

writeln(maxi, ' ', t);

**end**.

Сначала отсортируем массив в порядке возрастания. И теперь, последовательно складывая элементы массива(a) с начала и сравнивая сумму(sum) с размером свободного места на диске(s) получим максимальное количество пользователей, чьи файлы могут поместиться на диске(maxi). Потом, вычитая из найденной суммы наибольший файл в текущей последовательности, будем пробовать прибавлять файлы с большим весом. Если такой файл будет найден, то заменяем значение наибольшего файла, который возможно поместить на диск.

ЗАДАНИЕ 5

**var**

n, a, b, maxa, mina, s, minr, i: int64;

**begin**

s := 0;

minr := 10001; //так как максимальное значение числа по условию 10000, то положим минимальную разницу 10001

readln(N);

**for** i := 1 **to** N **do**

**begin**

readln(a, b);

**if** a > b **then begin** maxa := a; Mina := b **end**

**else begin** Maxa := b; Mina := a **end**;

s := s + Maxa;

**if** ((Maxa - Mina) **mod** 3 > 0) **and** (Maxa - Mina < minr)

**then** minr := Maxa - Mina

**end**;

**if** s **mod** 3 = 0 **then**

s := s - minr;

writeln(s)

**end**.

Чтобы получить максимально возможную сумму, будем брать из каждой пары самое большое число. Если полученная при этом сумма будет делиться на 3, её необходимо уменьшить. Для этого достаточно в одной из пар, где числа имеют разные остатки при делении на 3, заменить ранее выбранное число на другое число из той же пары. При этом разница между числами в паре должна быть минимально возможной. Если во всех парах оба числа имеют одинаковый остаток при делении на 3, получить нужную сумму невозможно.

Программа читает все данные один раз. В каждой паре определяется большее число maxa и разность между большим и меньшим числами пары D. После обработки очередной пары программа хранит два числа: s — сумму всех максимальных элементов прочитанных пар и minr — наименьшую возможную разность D, не кратную 3. Окончательным ответом будет значение s, если оно не делится на 3, и s –minr в противном случае.

ЗАДАНИЕ 6

Код я не успел написать, поэтому тут будет только объяснение решения

Тут динамическое программирование.

Пусть A[i] - наибольшая возрастающая подпоследовательность, заканчивающаяся элементом с индексом i.

Пусть B[i] - наибольшая убывающая подпоследовательность, начинающаяся элементом с индексом i.

Когда эти два массива A и B будут построены, нужно определить, по какому индексу сумма значений будет максимальной. Индекс максимума искомой подпоследовательности будет таким образом найден.

Чтобы суметь восстановить ответ, помимо динамики d[1 .. n] надо также хранить вспомогательный массив p[1 .. n] — то, в каком месте достигся максимум для каждого значения d[i]. Иными словами, индекс p[i] будет обозначать тот самый индекс j, при котором получилось наибольшее значение d[i].

Тогда, чтобы вывести ответ, надо просто идти от элемента с максимальным значением d[i] по массиву **p** до тех пор, пока мы не выведем всю подпоследовательность, т.е. пока не дойдём до элемента со значением d = 1