Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ПРЕДПРОФЕССИОНАЛЬНЫЙ ЭКЗАМЕН
для учащихся инженерных классов (11 класс) города Москвы

Мастер-класс «Решение ситуационных задач практической части предпрофессионального экзамена (Направление программирование)»

Авторы: ***Калмыков Ю.В.,*** *старший преподаватель кафедры СУНЦ-1 «Основы математики и информатики» МГТУ им. Н.Э. Баумана*
Митрофанов М.С.*, ассистент кафедры «Основы математики и информатики» СУНЦ МГТУ им. Н.Э. Баумана*

Москва – 2019

Об экзамене

«Предпрофессиональный экзамен – это форма независимой итоговой оценки с участием представителей вузов, которая проводится по результатам освоения учащимися предпрофессиональных профильных программ в инженерных и медицинских классах образовательных организаций города. Экзамен состоит из двух частей – теоретической и практической. Первую, теоретическую, часть экзамена участники сдают в Московском центре качества образования, где для участников созданы все условия, аналогичные условиям при сдаче ЕГЭ. Вторая часть экзамена пройдет на базе вузов. Участников обеспечат всем необходимым оборудованием и материалами для выполнения практических заданий»

Основные направления задач практической части

- **Программирование + физика.**

Сложное программирование, элементарная физика.

- **Физика + программирование.**

Задача по физике, для решения которой надо програмировать.

Направления подготовки по программированию

- Алгоритмы на графах. Как хранить граф в памяти, поиск пути.
- Численное интегрирование.
- Метод половинного деления.
- Рекурсивный перебор для массива.
- Динамическое программирование.
- Задача коммивояжёра (https://ru.wikipedia.org/wiki/Задача_коммивояжёра).
- Задача о рюкзаке (https://ru.wikipedia.org/wiki/Задача_о_ранце).
- Задача о трубе (Проверить возможность составить из кусков труб трубу нужной длины с нужной пропускной способностью).

- http://algolist.manual.ru/maths/combinat/sequential.php

Направления подготовки по физике

- Конденсаторы
- Колебания
- Резисторы
- Кинематика
- Задачи, в которых меняется ускорение или есть сопротивление среды.

Задача о кузнечике/зайчике/…

Зайчик прыгает по прямой просеке, для удобства разделённой на n клеток. Клетки пронумерованы по порядку натуральными числами от 1 до n. Некоторые клетки заболочены (‘ w ’): если зайчик прыгнет на такую клетку, ему несдобровать. Некоторые другие клетки просеки поросли вкусной зелёной травой (‘ ” ’): прыгнув на такую клетку, зайчик сможет отдохнуть и подкрепиться.

Зайчик начинает свой путь из клетки с номером 1 и хочет попасть в клетку с номером n, по пути ни разу не провалившись в болото и скушав как можно больше вкусной зелёной травы. Конструктивные особенности зайчика таковы, что из клетки с номером k он может прыгнуть лишь в клетки с номерами k + 1, k + 3 и k + 5.

Выясните, какое максимальное количество клеток с травой сможет посетить зайчик на своём пути.

Задача о кузнечике/зайчике/…

1	2	3	4	5	6	7	8
.	.	.	w	“	“	w	.

1	2	3	4	5	6	7	8
0	-1	-1	-1	-1	-1	-1	-1

Задача о кузнечике/зайчике/…

Массив динамики изначально занят -1, кроме первого элемента, который равен 0.

-1 значит, что на эту клетку заяц попасть не смог.

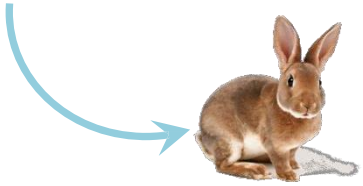
Числа ≥ 0 – это максимальное количество клеток с травой, которые на своём пути посетил заяц.

Логично, что в клетку k заяц может попасть только из клеток k-1, k-3 и k-5. Тогда максимальное количество клеток с травой, посещённых до клетки k включительно, будет равняться максимальному количеству очков среди тех клеток, откуда мы могли прийти, и +1 в том случае, если k – сама клетка с травой.

Задача о кузнечике/зайчике/…

1	2	3	4	5	6	7	8
.	.	.	w	“	“	w	.

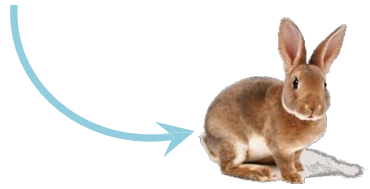
1	2	3	4	5	6	7	8
0	0	-1	-1	-1	-1	-1	-1



Задача о кузнечике/зайчике/…

1	2	3	4	5	6	7	8
.	.	.	W	“	“	W	.

1	2	3	4	5	6	7	8
0	0	0	-1	-1	-1	-1	-1



Задача о кузнечике/зайчике/…

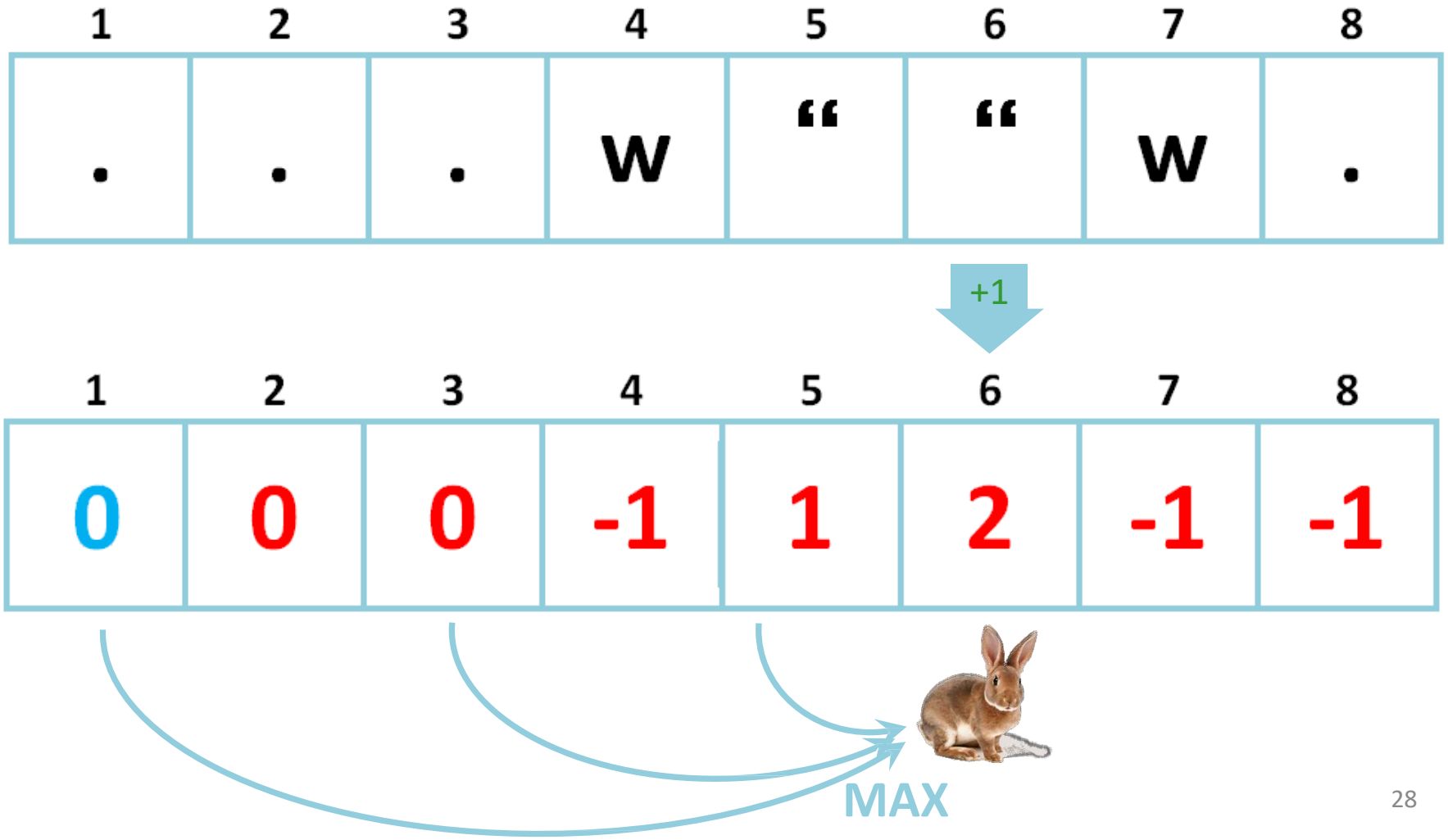
1	2	3	4	5	6	7	8
.	.	.	W	“	“	W	.

+1

1	2	3	4	5	6	7	8
0	0	0	-1	1	-1	-1	-1

MAX

Задача о кузнечике/зайчике/…



Задача о кузнечике/зайчике/…

1	2	3	4	5	6	7	8
.	.	.	W	“	“	W	.

1	2	3	4	5	6	7	8
0	0	0	-1	1	2	-1	1

MAX

Задача о кузнечике/зайчике/…

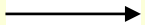
```
vector <int> a(n+2, -1);
char c; cin >> c;
a[1] = 0;
for(int i = 2, i <= n; i++) {
    cin >> c;
    if(c ==‘w’) continue;

    if(i < 3)
        a[i] = a[i-1];
    else if(i < 5)
        a[i] = max( a[i-1], a[i-3] );
    else
        a[i] = max( max( a[i-1], a[i-3] ), a[i-5] );

    if(c == ‘””’) a[i]++;
}

cout << a[n];
```

Морское орудие стреляет снарядом массой m с начальной скоростью V0. Силу сопротивления среды считать равной произведению коэффициента сопротивления среды k и квадрата скорости снаряда V. Угол возвышения составляет α. Определить расстояние, на которое полетит снаряд. Стрельба происходит в море вдали от суши.

Формат ввода

- На вход программы подаётся четыре положительных вещественных числа:
- масса снаряда, его начальная скорость, угол возвышения и коэффициент
- сопротивления среды. Все физические величины приведены в системе СИ и вводятся с
- точностью до двух знаков после запятой.
- Рассчитать расстояние, на которое полетит снаряд, с точностью до 1 м.

```
program ppe19;
 const g=9.81;
 function
move(m,v0,a,k:real):real;
var vx,vy,ax,ay,h,x,y:real;
begin
 vx:=v0*cos(a);
 vy:=v0*sin(a);
 y:=0;
 x:=0;
 h:=0.001;
 while y>=0 do
  begin
   ax:=-k*sqr(vx)/m;
   ay:=-k*sqr(vy)/m;
   vx:=vx+ax*h;
   vy:=vy+(ay-g)*h;
   x:=x+vx*h;
   y:=y+vy*h;
  end;
 move:=x;
end;
```



```
var
m,v0,a,k,l:real;
begin
 read(m);
 read(v0);
 read(a);
 readln(k);

l:=move(m,v0,a,k);
 writeln(l:10:1);
end.
```

Критерии оценивания

- Задача состоит из двух частей: физической и программной. Физическая часть задачи заключается в формализации процесса движения с помощью уравнения. Программная часть задачи заключается в численном расчете искомой величины, используя уравнение, полученное в физической части.
- Физическая часть задачи оценивается максимум на 25 баллов.
- Максимальная оценка ставится за правильно составленное уравнение движения снаряда, в котором учтено влияние сопротивления среды.
- Каждая ошибка в формуле лишает учащегося от 10 баллов.
- Частный случай с нулевым сопротивлением среды оценивается на 5 баллов.
- Программная часть задачи оценивается максимум на 25 баллов.
- Максимальная оценка ставится за правильно написанную программу, в которой численно с требуемой точностью находится искомый параметр.
- Неверная точность из-за настроек метода решения лишает учащегося 5 баллов.
- Каждая содержательная ошибка в программе лишает учащегося 10 баллов.
- Программная реализация частного случая с нулевым сопротивлением среды оценивается на 5 баллов.

Перестановки в строке

Вывести все возможные перестановки символов в строке.

```
procedure perest(head,
tail: string);
…

var
  tail: string;
begin
  readln(tail);
  perest('', tail);
end.
```

```
procedure perest(head, tail:
string);
var newhead, newtail: string; i:
integer;
begin
  if length(tail) = 0 then
writeln(head)
  else begin
        for i := 1 to length(tail)
do
     begin
        newhead := head + copy(tail,
i, 1);
        newtail := tail;
        delete(newtail, i, 1);
        perest(newhead, newtail);
     end;
  end;
end;
```

Перестановки в массиве

Вывести все возможные перестановки элементов в массиве.

```
const  N = 5;
type tmas = array [1 .. N]
of integer;
procedure init(var mas:
tmas);
…
procedure perest(mas: tmas;
cur: integer);
…
var mas: tmas;
begin
 init(mas);
 perest(mas, 1);
end.
```

```
procedure init(var mas:
tmas);
var i: integer;
begin
  for i := 1 to N do
      mas[i] := i;
end;
procedure vivod(mas: tmas);
var i: integer;
begin
  for i := 1 to N do
      write (mas[i], ' ');
  writeln;
end;
```

Перестановки в массиве

```
procedure perest(mas: tmas;
cur: integer);
var i: integer;
begin
  if cur > N then
    vivod(mas)
  else
      for i := cur to N do
begin
     swap(mas[i],
mas[cur]);
     perest(mas, cur + 1);
     swap(mas[i],
mas[cur]);
      end;
end;
```

```
procedure
swap(var x, y:
integer);
var z: integer;
begin
  z:=x;
  x:=y;
  y:=z;
end;
```

Игра "Ханойская башня" состоит в следующем. Есть три стержня. На первый из них надета пирамидка из N колец (большие кольца снизу, меньшие сверху). Требуется переместить кольца на другой стержень. Разрешается перекладывать кольца со стержня на стержень, но класть большее кольцо поверх меньшего нельзя. Составить программу, указывающую требуемые действия.

Напишем рекурсивную процедуру перемещения M верхних колец с A-го стержня на B-ый в предположении, что остальные кольца больше по размеру и лежат на стержнях без движения:

```
procedure Move(M,A,B:integer);
      var C:integer;
    begin
      if M=1 then begin writeln ('сделать ход ',A,'->',B) end
      else
          begin
            C:=6-A-B; {C - третий стержень: сумма номеров
равна 6}
            Move(M-1,A,C);
            Move(1,A,B);
            Move(M-1,C,B)
          end
    end;
```

Сначала переносится пирамидка из M-1 колец на третий стержень C. После этого M-ое кольцо освобождается, и его можно перенести на B. Остается перенести пирамиду из N-1 кольца с C на B. Количество колец стало на единицу меньше. Теперь основную программу можно записать в несколько строк:

```
program Hanoi;
    var N:integer;
    procedure Move(M,A,B:integer);
        .............
  begin
    write('N=');readln(N);
    Move(N,1,2)
  end.
```

Задача: Работа силы

Тело движется вдоль оси Ox под воздействием силы F(x), сонаправленной с осью Ox. В общем виде F(x) определена как $F=a(x+1)^{1/2}+b(x+2)+cx^2+d*\ln(x+1)$, где a, b, c, d – коэффициенты, которые могут меняться на разных отрезках. Гарантируется, что каждое уравнение имеет математический смысл.

Зная количество отрезков, коэффициенты уравнения кривых и границы отрезков, на которых эти кривые применяются, найдите совершенную над телом работу силы f(x) с точностью $\varepsilon = 10^{-5}$.

Точность ε считать достигнутой, когда при вычислении интегральной суммы уменьшение отрезка x вдвое приводит к изменению суммы меньше, чем на ε.

Задача: Работа силы (ввод/вывод)

- В первой строке вводится натуральное число N – число отрезков. N не превышает 5.
- Далее следует N строк, в каждой из которых через пробел записаны шесть вещественных положительных чисел x_i, x_{i+1}, a, b, c, d – соответственно, границы отрезка, и коэффициенты при кривой на этом отрезке.
- Гарантируется, что разрывов нет и каждый следующий отрезок начинается там, где закончился предыдущий.
- Никакие числа не превышают 1000.
- На выходе программа должна выдать вещественное число – совершенную над телом работу силы f(x). Все величины задаются в системе СИ, ответ привести в джоулях.

Задача: Работа силы (решение)

```
function work(x0, x1, a, b, c,
d: real): real;
…
var w, x0, x1, a, b, c, d:
real; i, n: integer;
begin
  w := 0;
  readln(n);
  for i:=1 to n do begin
       readln(x0, x1, a, b, c,
d);
       w := w + work(x0 , x1,
a, b, c, d);
  end;
  writeln(w);
end.
```

```
const
 eps = 0.00001;

function f(x, a, b, c, d:
real): real;
begin
  f := a * sqrt(x + 1) + b *
(x + 2) +
       c * sqr(x) + d * ln(x
+ 1);
end;
```

Задача: Работа силы (решение)

```
function work(x0, x1, a, b, c, d: real): real;
var vt, pv, x, h: real;
begin
  vt := sqr(f((x1 + x0) / 2, a, b, c, d)) * (x1 - x0);
  h := (x1 - x0) / 2;
  repeat
       pv := vt; x := x0; vt := 0;
       while x < x1 do begin
     vt := vt + f(x, a, b, c, d) * h; x := x + h;
   end;
       h := h / 2;
  until abs(vt - pv) < eps;
  work := vt;
end;
```

Задача: Уравновешивание груза

Чтобы уравновесить тяжелый груз, висящий на одном конце легкого стержня, требуется уравновесить его с помощью гирь. Зная массу груза, массу одинаковых гирь из набора и расстояния от точек приложения сил до точки подвеса, подобрать сочетание точек подвеса гирь, которое позволит уравновесить груз.

Груз считается уравновешенным, если моменты сил, действующих на него и на набор гирь, различаются не более чем на 5%.

Задача: Уравновешивание груза (ввод/вывод)

- В первой строке вводится натуральное число N – число гирь и точек подвеса. N не превышает 10.
- Далее через пробел в одной строке вводятся вещественные положительные числа M, L, m_w – соответственно, масса груза, длина стержня от точки подвеса до груза, масса гирь.
- Далее следует N строк, в каждой из которых содержится одно вещественное положительное число li – расстояние от точки, на которой можно закрепить гирю до точки подвеса.
- Никакие числа не превышают 1000.
- На выходе программа должна выдать через запятую номера точек подвеса, или вывести 0, если уравновесить грузы невозможно.

Задача: Уравновешивание груза (решение)

```
var mas: tmasl; nums: tmasn; j, n, kolvo: integer;
    flag: boolean;  m, l, m_w: real;
begin
  readln(n); readln(m, l, m_w);
  for j := 1 to n do begin
      read(mas[j]); nums[j] := j;
  end;
  kolvo := 1; flag := false;
  while (kolvo <= n) and not flag do begin
      perest(nums, mas, 1, kolvo, n, m, l, m_w, flag);
      kolvo := kolvo+1;
  end;
  if not flag then writeln(0);
end.
```

Задача: Уравновешивание груза (решение)

```
const
  MaxN = 10;
  eps = 0.05;
type
  tmasl = array[1..MaxN] of real;
  tmasn = array[1..MaxN] of integer;
procedure output(num: tmasn; kolvo: integer);
var i: integer;
begin
  for i := 1 to kolvo - 1 do
       write(num[i], ', ');
  writeln(num[kolvo]);
end;
```

Задача: Уравновешивание груза (решение)

```
function check(nums: tmasn;  mas: tmasl; kolvo: integer;
                                    m, l, m_w: real): boolean;
var
  w2: real; i: integer;
begin
  w2:=0;
  for i := 1 to kolvo do
       w2 := w2 + mas[nums[i]];
  check := abs(m * l - m_w * w2) / (m * l) <= eps;
end;
procedure swap(var x, y: integer);
…
```

```
procedure perest(nums: tmasn; mas: tmasl;
 cur, kolvo, n: integer; m, l, m_w: real; var flag: boolean);
var i: integer;
begin
  if cur > kolvo then begin
       flag := check(nums, mas, kolvo, m, l, m_w);
       if flag then output(nums,kolvo)
  end
  else begin
    i := cur;
    while (i <= n) and not flag do begin
      swap(nums[i], nums[cur]);
      perest(nums, mas, cur+1, kolvo, n, m, l, m_w, flag);
      swap(nums[i], nums[cur]);
      i := i + 1;
       end;
  end;
end;
```

Задача: Пушка Гаусса

Для работы пушки Гаусса требуется подключить конденсаторы. Из-за ограничений, заданных при разработке оружия, объем набора конденсаторов не должен превышать заданное значение. Зная параметры типовых конденсаторов, которые подключаются параллельно в одну схему, требуется определить несколько вариантов подключения конденсаторов к пушке Гаусса.

Заранее известно следующее.

1. Ограничение по объему V составляет 1000 $см^3$.
2. Всего существует не более пяти видов типовых конденсаторов. Объем каждого из них кратен 0.5 $см^3$, а емкость равна целому числу условных единиц (примечание автора: поскольку мы не знаем, какими будут конденсаторы во времена, когда будут создавать пушку Гаусса, мы остереглись давать конкретные единицы измерения). Запас конденсаторов считать бесконечным.

Задача: Пушка Гаусса (ввод/вывод)

В первой строке на ввод подается целое число N – количество видов конденсаторов.

Далее в N строках задаются одно вещественное и два целых числа: vol, cap и cost – объем, емкость и цена конденсатора. Объем вводится с точностью до одного знака после запятой. (примечание автора: даже во времена, когда будут создавать пушку Гаусса, объем, цена и емкость конденсатора положительны.).

Вопросы:

Задание 1. Найти цену набора конденсаторов, который, помещаясь в заданном объёме, обеспечит наибольшую ёмкость.

Задание 2. Найти емкость наиболее дешевого набора конденсаторов, который целиком займёт заданный объем.

Задача: Пушка Гаусса (решение)

```
const V = 2000; M = 5;
type  tmatr = array [0 .. V, 1 .. 2] of integer;
      tcapa = array [1 .. M, 1 .. 3] of integer;
var capa: tcapa; n, i: integer; vol: real;
begin
  readln(n);
  for i := 1 to n do begin
       readln(vol, capa[i, 2], capa[i, 3]);
       capa[i, 1] := round(2 * vol);
  end;
  writeln(task_cost(capa, n));
  writeln(task_capacity(capa, n));
end.
```

Задача: Пушка Гаусса (решение)

```
procedure init(var matr: tmatr);
var i: integer;
begin
  matr[0, 1] := 0;
  matr[0, 2] := 0;
  for i := 1 to V do begin
        matr[i, 1] := -1;
        matr[i, 2] := 0;
  end;
end;
```

Задача: Пушка Гаусса (решение)

```
function task_cost(capa: tcapa; n: integer): integer;
var i, j, max: integer; matr: tmatr;
begin
  init(matr);
  for i:=1 to V do
       for j:=1 to n do
       if (i - capa[j, 1] >= 0) and
                          (matr[i - capa[j, 1], 1] <> -1) then
       if (matr[i, 1] = -1) or
      (matr[i, 1] < matr[i - capa[j, 1], 1] + capa[j, 2]) then
                                                      begin
       matr[i, 1] := matr[i - capa[j, 1], 1] + capa[j, 2];
       matr[i, 2] := matr[i - capa[j, 1], 2] + capa[j, 3];
       end;
  …
end;
```

Задача: Пушка Гаусса (решение)

```
function task_cost(capa: tcapa; n: integer): integer;
var i, j, max: integer; matr: tmatr;
begin
  …
  max := 0;
  for i := 1 to V do
       if matr[i, 1] > matr[max, 1] then
       max := i;
  task_cost := matr[max, 2];
end;
```

Задача: Пушка Гаусса (решение)

```
function task_capacity(capa: tcapa; n: integer): integer;
var i, j: integer; matr: tmatr;
begin init(matr);
  for i := 1 to V do
       for j := 1 to n do
       if (i - capa[j, 1] >= 0) and
                          (matr[i - capa[j, 1], 1] <> -1) then
       if (matr[i, 1] = -1) or
      (matr[i, 2] > matr[i - capa[j, 1], 2] + capa[j, 3]) then
                                                      begin
       matr[i, 1] := matr[i - capa[j, 1], 1] + capa[j, 2];
       matr[i, 2] := matr[i - capa[j, 1], 2] + capa[j, 3];
       end;
  task_capacity := matr[V, 1];
end;
```

Материалы семинара:

https://vk.com/topic-153393975_38000961

Спасибо за внимание!