Matrizes e Equações Lineares

O sistema

$$\left\{ \begin{array}{lcl} a_{11}\ x_1 + a_{12}\ x_2 + ... + a_{1n}\ x_n & = & b_1 \\ a_{21}\ x_1 + a_{22}\ x_2 + ... + a_{2n}\ x_n & = & b_2 \\ ... & & \\ a_{m1}\ x_1 + a_{m2}\ x_2 + ... + a_{mn}\ x_n & = & b_m \end{array} \right.$$

é equivalente à seguinte equação matricial

$$\left(\begin{array}{l} a_{11}\ a_{12}\ ...\ a_{1n} \\ a_{21}\ a_{22}\ ...\ a_{2n} \\ ... \\ a_{m1}\ a_{m2}\ ...\ a_{mn} \end{array} \right) \left(\begin{array}{c} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{array} \right) = \left(\begin{array}{c} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{array} \right)$$

no sentido de que toda solução do primeiro é igualmente solução do segundo. Dessa maneira pode-se resumir qualquer dos sistemas acima na forma matricial escrevendo

$$Ax = B$$

Matrizes Inversíveis

Diz-se que uma matriz quadrada é inversível se existe uma matriz B com a propriedade de que $AB = BA = I$ onde I é a matriz unidade.

A matriz B inversa de A é única se existir. A matriz inversa de A é indicada por A^{-1}. Note-se que a relação acima é simétrica, pois se B é a inversa de A então A também é inversa de B.

Mecanismos de cálculo da inversa

Usando a teoria: Usando a teoria, dada uma matriz como por exemplo $\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array} \right)$ Uma possibilidade é querer encontrar escalares x, y, z e w para os quais
$\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array} \right) \left(\begin{array}{cc} x & y \\ z & w \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$ ou
$\left(\begin{array}{cc} 3x+5z & 3y+5w \\ 2x+3z & 2y+3w \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$
ou ainda equações que satisfaçam
$\left\{ \begin{array}{l} 3x+5z=1 \\ 2x+3z=0 \end{array} \right.$ e $\left\{ \begin{array}{l} 3y+5w=0 \\ 2y+3w=1 \end{array} \right.$
Resolvendo estas equações, acha-se a matriz inversa que é $\left(\begin{array}{cc} -3 & 5 \\ -2 & 3 \end{array} \right)$.

Usando determinantes: Outros métodos (extraídos do excelente site `http://mathworld.wolfram.com`) vão a seguir descritos: Dada uma matriz 2×2 representada por $A = \left| \begin{array}{cc} a & b \\ c & d \end{array} \right|$, a inversa é dada por
$A^{-1} = \frac{1}{|A|} \left| \begin{array}{cc} d & -b \\ -c & a \end{array} \right| = \frac{1}{ad-bc} \left| \begin{array}{cc} d & -b \\ -c & a \end{array} \right|$.

Para uma matriz 3×3 veja lá no site, que o espaço aqui é muito pequeno.

Usando Gauss-Jordan: Neste método escreve-se a matriz da qual se quer achar a inversa ao lado esquerdo da matriz identidade, formando este arranjo (matriz escalonada reduzida por linhas).

$$[AI] = \left| \begin{array}{ll} a_{11}\ ...\ a_{1n} & 1\ 0\ ...\ 0 \\ a_{21}\ ...\ a_{2n} & 0\ 1\ ...\ 0 \\ ... & ... \\ a_{n1}\ ...\ a_{nn} & 0\ 0\ ...\ 1 \end{array} \right|$$

Aplica-se agora a este arranjo as técnicas de eliminação de Gauss, sobre o conjunto todo, de maneira a ficar com o seguinte arranjo

$$\left| \begin{array}{ll} 1\ 0\ ...\ 0 & b_{11}\ ...\ b_{1n} \\ 0\ 1\ ...\ 0 & b_{21}\ ...\ b_{2n} \\ ... & ... \\ 0\ 0\ ...\ 1 & b_{n1}\ ...\ b_{nn} \end{array} \right|$$

Neste ponto, a matriz (b_{ij}) é a matriz inversa de A. Ou seja $A^{-1} = B$ e também $AB = I$.

A vantagem deste último método é que ele vale para qualquer dimensão de matriz. É este método que vai ser programado em C++ a seguir.

Para que serve a matriz inversa ?

Inúmeras aplicações, mas vamos focar aqui na solução de sistemas lineares. Dado o sistema $Ax = B$, a matriz A é a dos coeficientes. Se você calcular a inversa A^{-1}, poderá resolver inúmeras instâncias do mesmo sistema, sem outra preocupação, já que $A^{-1} \times B = x$. Vamos ver um exemplo do que se diz aqui. Seja um sistema, dado por
$\left\{ \begin{array}{ll} 3x+2y+z & = \\ x-y+z & = \\ 5x+z & = \end{array} \right\}$
Escrito na forma matricial, o sistema fica $Ax = B$, onde A é a matriz
$\left(\begin{array}{ccc} 3 & 2 & 1 \\ 1 & -1 & 1 \\ 5 & 0 & 1 \end{array} \right)$
Calculando a matriz inversa desta, obtém-se:
$\left(\begin{array}{ccc} -0.1 & -0.2 & 0.3 \\ 0.4 & -0.2 & -0.2 \\ 0.5 & 1 & -0.5 \end{array} \right)$
Agora, a solução para a instância onde $B = \{10, 2, 8\}$ é obtida fazendo-se $A^{-1} \times B$ que dá os valores $x = 1, y = 2, z = 3$. Para outra instância cujo B seja $\{30, -1, 26\}$, ao fazer a mesma multiplicação, obtém-se $x = 5, y = 7, z = 1$ e assim por diante.

Este exercício pede que você implemente o algoritmo de Gauss-Jordan para cálculo da matriz inversa e depois o aplique a uma matriz de ordem suficientemente alta para desencorajar o cálculo manual da mesma. Relembrando as etapas, você deve:

1. Escrever a matriz A e ao seu lado a matriz I.
2. Aplicar operações elementares com linhas a A visando transformá-la em I.
3. Aplicar rigorosamente as mesmas operações sobre a matriz ao lado.
4. Quando A tiver se transformado em I, a matriz que era originalmente I se transformou em A^{-1}.
5. Para conferir, faça a multiplicação matricial entre A e A^{-1}. Tem que dar I.

Vamos aplicar este processo ao sistema acima descrito. O objetivo é produzir um zero no coeficiente do x na segunda equação: Para isso:
$a[1][] = a[1][] + a[0] \times (-a[1][0]/a[0][0])$ onde a notação $a[1][]$ deve ser entendida como a aplicação da operaçao à toda a linha [1] da matriz a. Mas, atenção: isto não vale no compilador C++. Lá, tem que ser por extenso.

Note-se também que o fator $(-a[1][0]/a[0][0])$ é o mesmo para toda a linha e portanto deve ser calculado antes e preservado durante todo o ciclo. O algoritmo vai chamar este fator de *pivot* e ele vai ser previamente calculado e guardado.

para calcular o zero no coeficiente de x na terceira linha, o comando
é $a[2][] = a[2][] + a[0] \times (-a[2][0]/a[0][0])$.

Para a segunda coluna, começa-se achando o zero na primeira linha. O comando agora é
$a[0][] = a[0][] + a[1][] \times (-a[0][1]/a[1][1])$ e depois $a[2][] = a[2][] + a[1][] \times (-a[2][1]/a[1][1])$. A terceira coluna é
$a[0][] = a[0][] + a[2][] \times (-a[0][2]/a[2][2])$ e
$a[1][] = a[1][] + a[2][] \times (-a[1][2]/a[2][2])$.
Com estas operações, os zeros foram alcançados. Agora basta transformar os elementos da diagonal principal na unidade. As operações são:
$a[0][] = a[0][]/a[0][0]$, $a[1][] = a[1][]/a[1][1]$ e $a[2][] = a[2][]/a[2][2]$.

Note que não houve preocupação em determinar se a matriz A é inversível. Isto foi deixado para não complicar demais o algoritmo. Fica como desafio para os mais corajosos.

Eis o algoritmo:

```
#include<iostream>
#include<iomanip>
#define ta 5
using namespace std;
int cami(float x[ta][ta], float y[ta][ta]){
// x é a matriz A e y é a matriz A^-1
  int i,j,k;
  float pivot,alvo;
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (i==j) {
        y[i][j]=1;
      }
      else {
        y[i][j]=0;
      }
    }
  }
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (j!=i){
        pivot=-x[j][i]/x[i][i];
          for (k=0;k<ta;k++){
            x[j][k]=x[j][k]+x[i][k]*pivot;
            y[j][k]=y[j][k]+y[i][k]*pivot;
          }
      }
    }
    alvo=x[i][i];
    for (k=0;k<ta;k++){
      y[i][k]=y[i][k]/alvo;
      x[i][k]=x[i][k]/alvo;
    }
  }
  return 0;
}
int main() {
  float mat[ta][ta]={{...},{...},{},...};
  float res[ta][ta];
  int i,j;
  cami(mat,res);
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      cout<<fixed<<setprecision(5)
            <<res[i][j]<<"  ";
    }
    cout<<endl;
  }
}
```

Um exemplo Para você testar seu algoritmo, use-o nesta matriz:

```
 59 57 44 42 65 23 47 66 78 71
 94 36 83 17 12 86  4 92 43 95
 12 61 18 10 94 45 94 81 30 76
  9 49 92 22 90 22 27  6 21 26
 31 45 96 26 26 75 59 39 47 98
 33 27  9 94 29  5 49 68  9 33
 61 81 17 18 61 74 16 37 34 45
 64 85 46 74 49 81 69 20 18 59
 17 54 66 65 94 54 76 54 30 76
 68 91 61 89  6 57 35 92 52 80
```

Aqui a resposta deverá ser 151.989, já devidamente multiplicada por 1000.

☞ Para você fazer

A seguir uma matriz A de dimensão 10×10 (o que corresponderia a um sistema linear de 10 equações e 10 incógnitas). Você deve calcular a matriz inversa, seguindo o algoritmo acima, e depois responder qual o valor que está localizado na sexta linha e sétima coluna de A^{-1}, (em C: a[5][6]) multiplicado por 1000. Eis a matriz:

```
 23 55 35 63 40 40 68 23 95 79
 62 57 16 96 42 30 93  4 19 28
 31 55 95 40 62  4 49 94 35  5
 80 49 18 31 60 54 50 14 97 24
 25 34  6 91  4 33 79 47  5 83
 63  1 74 56 55 34 68 18 94 91
 21 51 20 99 66  8 74 87  2 18
 91 79 24 66 21 79 67 83 41 64
 79 97 37 80 25 18 36  6 46 41
 74 72 42 10 71 62 54 21 90 83
```

Responda aqui:

$A^{-1}[5][6] \times 1000 =$

Matrizes e Equações Lineares

O sistema

$$\left\{ \begin{array}{lcl} a_{11}\ x_1 + a_{12}\ x_2 + ... + a_{1n}\ x_n & = & b_1 \\ a_{21}\ x_1 + a_{22}\ x_2 + ... + a_{2n}\ x_n & = & b_2 \\ ... & & \\ a_{m1}\ x_1 + a_{m2}\ x_2 + ... + a_{mn}\ x_n & = & b_m \end{array} \right.$$

é equivalente à seguinte equação matricial

$$\left(\begin{array}{l} a_{11}\ a_{12}\ ...\ a_{1n} \\ a_{21}\ a_{22}\ ...\ a_{2n} \\ ... \\ a_{m1}\ a_{m2}\ ...\ a_{mn} \end{array} \right) \left(\begin{array}{c} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{array} \right) = \left(\begin{array}{c} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{array} \right)$$

no sentido de que toda solução do primeiro é igualmente solução do segundo. Dessa maneira pode-se resumir qualquer dos sistemas acima na forma matricial escrevendo

$$Ax = B$$

Matrizes Inversíveis

Diz-se que uma matriz quadrada é inversível se existe uma matriz B com a propriedade de que $AB = BA = I$ onde I é a matriz unidade.

A matriz B inversa de A é única se existir. A matriz inversa de A é indicada por A^{-1}. Note-se que a relação acima é simétrica, pois se B é a inversa de A então A também é inversa de B.

Mecanismos de cálculo da inversa

Usando a teoria: Usando a teoria, dada uma matriz como por exemplo $\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array} \right)$ Uma possibilidade é querer encontrar escalares x, y, z e w para os quais
$\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array} \right) \left(\begin{array}{cc} x & y \\ z & w \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$ ou
$\left(\begin{array}{cc} 3x+5z & 3y+5w \\ 2x+3z & 2y+3w \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$
ou ainda equações que satisfaçam
$\left\{ \begin{array}{l} 3x+5z=1 \\ 2x+3z=0 \end{array} \right.$ e $\left\{ \begin{array}{l} 3y+5w=0 \\ 2y+3w=1 \end{array} \right.$
Resolvendo estas equações, acha-se a matriz inversa que é $\left(\begin{array}{cc} -3 & 5 \\ -2 & 3 \end{array} \right)$.

Usando determinantes: Outros métodos (extraídos do excelente site `http://mathworld.wolfram.com`) vão a seguir descritos: Dada uma matriz 2×2 representada por $A = \left| \begin{array}{cc} a & b \\ c & d \end{array} \right|$, a inversa é dada por
$A^{-1} = \frac{1}{|A|} \left| \begin{array}{cc} d & -b \\ -c & a \end{array} \right| = \frac{1}{ad-bc} \left| \begin{array}{cc} d & -b \\ -c & a \end{array} \right|$.

Para uma matriz 3×3 veja lá no site, que o espaço aqui é muito pequeno.

Usando Gauss-Jordan: Neste método escreve-se a matriz da qual se quer achar a inversa ao lado esquerdo da matriz identidade, formando este arranjo (matriz escalonada reduzida por linhas).

$$[AI] = \left| \begin{array}{ll} a_{11}\ ...\ a_{1n} & 1\ 0\ ...\ 0 \\ a_{21}\ ...\ a_{2n} & 0\ 1\ ...\ 0 \\ ... & ... \\ a_{n1}\ ...\ a_{nn} & 0\ 0\ ...\ 1 \end{array} \right|$$

Aplica-se agora a este arranjo as técnicas de eliminação de Gauss, sobre o conjunto todo, de maneira a ficar com o seguinte arranjo

$$\left| \begin{array}{ll} 1\ 0\ ...\ 0 & b_{11}\ ...\ b_{1n} \\ 0\ 1\ ...\ 0 & b_{21}\ ...\ b_{2n} \\ ... & ... \\ 0\ 0\ ...\ 1 & b_{n1}\ ...\ b_{nn} \end{array} \right|$$

Neste ponto, a matriz (b_{ij}) é a matriz inversa de A. Ou seja $A^{-1} = B$ e também $AB = I$.

A vantagem deste último método é que ele vale para qualquer dimensão de matriz. É este método que vai ser programado em C++ a seguir.

Para que serve a matriz inversa ?

Inúmeras aplicações, mas vamos focar aqui na solução de sistemas lineares. Dado o sistema $Ax = B$, a matriz A é a dos coeficientes. Se você calcular a inversa A^{-1}, poderá resolver inúmeras instâncias do mesmo sistema, sem outra preocupação, já que $A^{-1} \times B = x$. Vamos ver um exemplo do que se diz aqui. Seja um sistema, dado por
$\left\{ \begin{array}{lcl} 3x+2y+z & = & \\ x-y+z & = & \\ 5x+z & = & \end{array} \right\}$
Escrito na forma matricial, o sistema fica $Ax = B$, onde A é a matriz
$\left(\begin{array}{ccc} 3 & 2 & 1 \\ 1 & -1 & 1 \\ 5 & 0 & 1 \end{array} \right)$
Calculando a matriz inversa desta, obtém-se:
$\left(\begin{array}{ccc} -0.1 & -0.2 & 0.3 \\ 0.4 & -0.2 & -0.2 \\ 0.5 & 1 & -0.5 \end{array} \right)$
Agora, a solução para a instância onde $B = \{10, 2, 8\}$ é obtida fazendo-se $A^{-1} \times B$ que dá os valores $x = 1, y = 2, z = 3$. Para outra instância cujo B seja $\{30, -1, 26\}$, ao fazer a mesma multiplicação, obtém-se $x = 5, y = 7, z = 1$ e assim por diante.

Este exercício pede que você implemente o algoritmo de Gauss-Jordan para cálculo da matriz inversa e depois o aplique a uma matriz de ordem suficientemente alta para desencorajar o cálculo manual da mesma. Relembrando as etapas, você deve:

1. Escrever a matriz A e ao seu lado a matriz I.
2. Aplicar operações elementares com linhas a A visando transformá-la em I.
3. Aplicar rigorosamente as mesmas operações sobre a matriz ao lado.
4. Quando A tiver se transformado em I, a matriz que era originalmente I se transformou em A^{-1}.
5. Para conferir, faça a multiplicação matricial entre A e A^{-1}. Tem que dar I.

Vamos aplicar este processo ao sistema acima descrito. O objetivo é produzir um zero no coeficiente do x na segunda equação: Para isso:
$a[1][] = a[1][] + a[0] \times (-a[1][0]/a[0][0])$ onde a notação $a[1][]$ deve ser entendida como a aplicação da operaçao à toda a linha [1] da matriz a. Mas, atenção: isto não vale no compilador C++. Lá, tem que ser por extenso.

Note-se também que o fator $(-a[1][0]/a[0][0])$ é o mesmo para toda a linha e portanto deve ser calculado antes e preservado durante todo o ciclo. O algoritmo vai chamar este fator de *pivot* e ele vai ser previamente calculado e guardado.

para calcular o zero no coeficiente de x na terceira linha, o comando
é $a[2][] = a[2][] + a[0] \times (-a[2][0]/a[0][0])$.

Para a segunda coluna, começa-se achando o zero na primeira linha. O comando agora é
$a[0][] = a[0][] + a[1][] \times (-a[0][1]/a[1][1])$ e depois $a[2][] = a[2][] + a[1][] \times (-a[2][1]/a[1][1])$. A terceira coluna é
$a[0][] = a[0][] + a[2][] \times (-a[0][2]/a[2][2])$ e
$a[1][] = a[1][] + a[2][] \times (-a[1][2]/a[2][2])$.
Com estas operações, os zeros foram alcançados. Agora basta transformar os elementos da diagonal principal na unidade. As operações são:
$a[0][] = a[0][]/a[0][0]$, $a[1][] = a[1][]/a[1][1]$ e $a[2][] = a[2][]/a[2][2]$.

Note que não houve preocupação em determinar se a matriz A é inversível. Isto foi deixado para não complicar demais o algoritmo. Fica como desafio para os mais corajosos.

Eis o algoritmo:

```
#include<iostream>
#include<iomanip>
#define ta 5
using namespace std;
int cami(float x[ta][ta], float y[ta][ta]){
// x é a matriz A e y é a matriz A^-1
  int i,j,k;
  float pivot,alvo;
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (i==j) {
        y[i][j]=1;
      }
      else {
        y[i][j]=0;
      }
    }
  }
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (j!=i){
        pivot=-x[j][i]/x[i][i];
          for (k=0;k<ta;k++){
            x[j][k]=x[j][k]+x[i][k]*pivot;
            y[j][k]=y[j][k]+y[i][k]*pivot;
          }
      }
    }
    alvo=x[i][i];
    for (k=0;k<ta;k++){
      y[i][k]=y[i][k]/alvo;
      x[i][k]=x[i][k]/alvo;
    }
  }
  return 0;
}
int main() {
  float mat[ta][ta]={{...},{...},{},...};
  float res[ta][ta];
  int i,j;
  cami(mat,res);
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      cout<<fixed<<setprecision(5)
            <<res[i][j]<<"  ";
    }
    cout<<endl;
  }
}
```

Um exemplo Para você testar seu algoritmo, use-o nesta matriz:

```
 59 57 44 42 65 23 47 66 78 71
 94 36 83 17 12 86  4 92 43 95
 12 61 18 10 94 45 94 81 30 76
  9 49 92 22 90 22 27  6 21 26
 31 45 96 26 26 75 59 39 47 98
 33 27  9 94 29  5 49 68  9 33
 61 81 17 18 61 74 16 37 34 45
 64 85 46 74 49 81 69 20 18 59
 17 54 66 65 94 54 76 54 30 76
 68 91 61 89  6 57 35 92 52 80
```

Aqui a resposta deverá ser 151.989, já devidamente multiplicada por 1000.

☞ Para você fazer

A seguir uma matriz A de dimensão 10×10 (o que corresponderia a um sistema linear de 10 equações e 10 incógnitas). Você deve calcular a matriz inversa, seguindo o algoritmo acima, e depois responder qual o valor que está localizado na sexta linha e sétima coluna de A^{-1}, (em C: a[5][6]) multiplicado por 1000. Eis a matriz:

```
 46 78 20 75 16 79 64 76 88  5
  4 10 21 61 95 84 23 21 34 84
 37  8 74  3 47 77 73 19  4 97
 83 25 52 97 61 30 73  5 69 30
 61 59 64 24 20 86 45 56 26 36
 33 31 28 40 32 19 97 66 48 77
 59 39 14 53 56 49 94 95 20 53
 96 40 16 37 75 47  7 62 52  3
 99 62 71 28 68 31 67 46 36 81
 22 60 42 79 15 30 60 98 92  3
```

Responda aqui:

$A^{-1}[5][6] \times 1000 =$

Matrizes e Equações Lineares

O sistema

$$\left\{\begin{array}{lcl} a_{11}\ x_1 + a_{12}\ x_2 + ... + a_{1n}\ x_n & = & b_1 \\ a_{21}\ x_1 + a_{22}\ x_2 + ... + a_{2n}\ x_n & = & b_2 \\ ... & & \\ a_{m1}\ x_1 + a_{m2}\ x_2 + ... + a_{mn}\ x_n & = & b_m \end{array}\right.$$

é equivalente à seguinte equação matricial

$$\left(\begin{array}{l} a_{11}\ a_{12}\ ...\ a_{1n} \\ a_{21}\ a_{22}\ ...\ a_{2n} \\ ... \\ a_{m1}\ a_{m2}\ ...\ a_{mn} \end{array}\right)\left(\begin{array}{c} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{array}\right)=\left(\begin{array}{c} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{array}\right)$$

no sentido de que toda solução do primeiro é igualmente solução do segundo. Dessa maneira pode-se resumir qualquer dos sistemas acima na forma matricial escrevendo

$$Ax = B$$

Matrizes Inversíveis

Diz-se que uma matriz quadrada é inversível se existe uma matriz B com a propriedade de que $AB = BA = I$ onde I é a matriz unidade.

A matriz B inversa de A é única se existir. A matriz inversa de A é indicada por A^{-1}. Note-se que a relação acima é simétrica, pois se B é a inversa de A então A também é inversa de B.

Mecanismos de cálculo da inversa

Usando a teoria: Usando a teoria, dada uma matriz como por exemplo $\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array}\right)$ Uma possibilidade é querer encontrar escalares x, y, z e w para os quais
$\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array}\right)\left(\begin{array}{cc} x & y \\ z & w \end{array}\right)=\left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right)$ ou
$\left(\begin{array}{cc} 3x+5z & 3y+5w \\ 2x+3z & 2y+3w \end{array}\right)=\left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right)$
ou ainda equações que satisfaçam
$\left\{\begin{array}{l} 3x+5z=1 \\ 2x+3z=0 \end{array}\right.$ e $\left\{\begin{array}{l} 3y+5w=0 \\ 2y+3w=1 \end{array}\right.$
Resolvendo estas equações, acha-se a matriz inversa que é $\left(\begin{array}{cc} -3 & 5 \\ -2 & 3 \end{array}\right)$.

Usando determinantes: Outros métodos (extraídos do excelente site `http://mathworld.wolfram.com`) vão a seguir descritos: Dada uma matriz 2×2 representada por $A=\left|\begin{array}{cc} a & b \\ c & d \end{array}\right|$, a inversa é dada por
$A^{-1}=\frac{1}{|A|}\left|\begin{array}{cc} d & -b \\ -c & a \end{array}\right|=\frac{1}{ad-bc}\left|\begin{array}{cc} d & -b \\ -c & a \end{array}\right|$.

Para uma matriz 3×3 veja lá no site, que o espaço aqui é muito pequeno.

Usando Gauss-Jordan: Neste método escreve-se a matriz da qual se quer achar a inversa ao lado esquerdo da matriz identidade, formando este arranjo (matriz escalonada reduzida por linhas).

$$[AI]=\left|\begin{array}{ll} a_{11}\ ...\ a_{1n} & 1\ 0\ ...\ 0 \\ a_{21}\ ...\ a_{2n} & 0\ 1\ ...\ 0 \\ ... & ... \\ a_{n1}\ ...\ a_{nn} & 0\ 0\ ...\ 1 \end{array}\right|$$

Aplica-se agora a este arranjo as técnicas de eliminação de Gauss, sobre o conjunto todo, de maneira a ficar com o seguinte arranjo

$$\left|\begin{array}{ll} 1\ 0\ ...\ 0 & b_{11}\ ...\ b_{1n} \\ 0\ 1\ ...\ 0 & b_{21}\ ...\ b_{2n} \\ ... & ... \\ 0\ 0\ ...\ 1 & b_{n1}\ ...\ b_{nn} \end{array}\right|$$

Neste ponto, a matriz (b_{ij}) é a matriz inversa de A. Ou seja $A^{-1} = B$ e também $AB = I$.

A vantagem deste último método é que ele vale para qualquer dimensão de matriz. É este método que vai ser programado em C++ a seguir.

Para que serve a matriz inversa ?

Inúmeras aplicações, mas vamos focar aqui na solução de sistemas lineares. Dado o sistema $Ax = B$, a matriz A é a dos coeficientes. Se você calcular a inversa A^{-1}, poderá resolver inúmeras instâncias do mesmo sistema, sem outra preocupação, já que $A^{-1} \times B = x$. Vamos ver um exemplo do que se diz aqui. Seja um sistema, dado por
$\left\{\begin{array}{lcl} 3x+2y+z & = & \\ x-y+z & = & \\ 5x+z & = & \end{array}\right\}$
Escrito na forma matricial, o sistema fica $Ax = B$, onde A é a matriz
$\left(\begin{array}{ccc} 3 & 2 & 1 \\ 1 & -1 & 1 \\ 5 & 0 & 1 \end{array}\right)$
Calculando a matriz inversa desta, obtém-se:
$\left(\begin{array}{ccc} -0.1 & -0.2 & 0.3 \\ 0.4 & -0.2 & -0.2 \\ 0.5 & 1 & -0.5 \end{array}\right)$

Agora, a solução para a instância onde $B = \{10, 2, 8\}$ é obtida fazendo-se $A^{-1} \times B$ que dá os valores $x = 1, y = 2, z = 3$. Para outra instância cujo B seja $\{30, -1, 26\}$, ao fazer a mesma multiplicação, obtém-se $x = 5, y = 7, z = 1$ e assim por diante.

Este exercício pede que você implemente o algoritmo de Gauss-Jordan para cálculo da matriz inversa e depois o aplique a uma matriz de ordem suficientemente alta para desencorajar o cálculo manual da mesma. Relembrando as etapas, você deve:

1. Escrever a matriz A e ao seu lado a matriz I.
2. Aplicar operações elementares com linhas a A visando transformá-la em I.
3. Aplicar rigorosamente as mesmas operações sobre a matriz ao lado.
4. Quando A tiver se transformado em I, a matriz que era originalmente I se transformou em A^{-1}.
5. Para conferir, faça a multiplicação matricial entre A e A^{-1}. Tem que dar I.

Vamos aplicar este processo ao sistema acima descrito. O objetivo é produzir um zero no coeficiente do x na segunda equação: Para isso:
$a[1][] = a[1][] + a[0] \times (-a[1][0]/a[0][0])$ onde a notação $a[1][]$ deve ser entendida como a aplicação da operaçao à toda a linha [1] da matriz a. Mas, atenção: isto não vale no compilador C++. Lá, tem que ser por extenso.

Note-se também que o fator $(-a[1][0]/a[0][0])$ é o mesmo para toda a linha e portanto deve ser calculado antes e preservado durante todo o ciclo. O algoritmo vai chamar este fator de *pivot* e ele vai ser previamente calculado e guardado.

para calcular o zero no coeficiente de x na terceira linha, o comando
é $a[2][] = a[2][] + a[0] \times (-a[2][0]/a[0][0])$.

Para a segunda coluna, começa-se achando o zero na primeira linha. O comando agora é
$a[0][] = a[0][] + a[1][] \times (-a[0][1]/a[1][1])$ e depois $a[2][] = a[2][] + a[1][] \times (-a[2][1]/a[1][1])$. A terceira coluna é
$a[0][] = a[0][] + a[2][] \times (-a[0][2]/a[2][2])$ e
$a[1][] = a[1][] + a[2][] \times (-a[1][2]/a[2][2])$.
Com estas operações, os zeros foram alcançados. Agora basta transformar os elementos da diagonal principal na unidade. As operações são:
$a[0][] = a[0][]/a[0][0]$, $a[1][] = a[1][]/a[1][1]$ e $a[2][] = a[2][]/a[2][2]$.

Note que não houve preocupação em determinar se a matriz A é inversível. Isto foi deixado para não complicar demais o algoritmo. Fica como desafio para os mais corajosos.

Eis o algoritmo:

```
#include<iostream>
#include<iomanip>
#define ta 5
using namespace std;
int cami(float x[ta][ta], float y[ta][ta]){
// x é a matriz A e y é a matriz A^-1
  int i,j,k;
  float pivot,alvo;
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (i==j) {
        y[i][j]=1;
      }
      else {
        y[i][j]=0;
      }
    }
  }
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (j!=i){
        pivot=-x[j][i]/x[i][i];
          for (k=0;k<ta;k++){
            x[j][k]=x[j][k]+x[i][k]*pivot;
            y[j][k]=y[j][k]+y[i][k]*pivot;
          }
      }
    }
    alvo=x[i][i];
    for (k=0;k<ta;k++){
      y[i][k]=y[i][k]/alvo;
      x[i][k]=x[i][k]/alvo;
    }
  }
  return 0;
}
int main() {
  float mat[ta][ta]={{...},{...},{},...};
  float res[ta][ta];
  int i,j;
  cami(mat,res);
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      cout<<fixed<<setprecision(5)
            <<res[i][j]<<"  ";
    }
    cout<<endl;
  }
}
```

Um exemplo Para você testar seu algoritmo, use-o nesta matriz:

```
 59 57 44 42 65 23 47 66 78 71
 94 36 83 17 12 86  4 92 43 95
 12 61 18 10 94 45 94 81 30 76
  9 49 92 22 90 22 27  6 21 26
 31 45 96 26 26 75 59 39 47 98
 33 27  9 94 29  5 49 68  9 33
 61 81 17 18 61 74 16 37 34 45
 64 85 46 74 49 81 69 20 18 59
 17 54 66 65 94 54 76 54 30 76
 68 91 61 89  6 57 35 92 52 80
```

Aqui a resposta deverá ser 151.989, já devidamente multiplicada por 1000.

☞ Para você fazer

A seguir uma matriz A de dimensão 10×10 (o que corresponderia a um sistema linear de 10 equações e 10 incógnitas). Você deve calcular a matriz inversa, seguindo o algoritmo acima, e depois responder qual o valor que está localizado na sexta linha e sétima coluna de A^{-1}, (em C: a[5][6]) multiplicado por 1000. Eis a matriz:

```
 39 88 87 75 32 72 69 97  7 42
 82 36 49 59 14 72 35 35 96 58
 78 70 69 54 46 15 81 25 50 27
 68 33 93 53 46 14 57 29  6 92
 87 68 93 80 14 39 11 41 78 70
 95 46 59 12 19 27 90 72  9 92
 66 32  5 99 88 95 94 79 57 73
 43 84 20 42 87 35 42  3 14 16
 33 23 84 73 88 41 91 60 65 14
 67 74 24 52 92 48 68 72 23 49
```

Responda aqui:

$A^{-1}[5][6] \times 1000 =$

Matrizes e Equações Lineares

O sistema

$$\left\{\begin{array}{lcl} a_{11}\ x_1 + a_{12}\ x_2 + ... + a_{1n}\ x_n & = & b_1 \\ a_{21}\ x_1 + a_{22}\ x_2 + ... + a_{2n}\ x_n & = & b_2 \\ ... & & \\ a_{m1}\ x_1 + a_{m2}\ x_2 + ... + a_{mn}\ x_n & = & b_m \end{array}\right.$$

é equivalente à seguinte equação matricial

$$\begin{pmatrix} a_{11}\ a_{12}\ ...\ a_{1n} \\ a_{21}\ a_{22}\ ...\ a_{2n} \\ ... \\ a_{m1}\ a_{m2}\ ...\ a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{pmatrix}$$

no sentido de que toda solução do primeiro é igualmente solução do segundo. Dessa maneira pode-se resumir qualquer dos sistemas acima na forma matricial escrevendo

$$Ax = B$$

Matrizes Inversíveis

Diz-se que uma matriz quadrada é inversível se existe uma matriz B com a propriedade de que $AB = BA = I$ onde I é a matriz unidade.

A matriz B inversa de A é única se existir. A matriz inversa de A é indicada por A^{-1}. Note-se que a relação acima é simétrica, pois se B é a inversa de A então A também é inversa de B.

Mecanismos de cálculo da inversa

Usando a teoria: Usando a teoria, dada uma matriz como por exemplo $\begin{pmatrix} 3 & 5 \\ 2 & 3 \end{pmatrix}$ Uma possibilidade é querer encontrar escalares x, y, z e w para os quais
$\begin{pmatrix} 3 & 5 \\ 2 & 3 \end{pmatrix}\begin{pmatrix} x & y \\ z & w \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ ou
$\begin{pmatrix} 3x+5z & 3y+5w \\ 2x+3z & 2y+3w \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
ou ainda equações que satisfaçam
$\left\{\begin{array}{l} 3x+5z=1 \\ 2x+3z=0 \end{array}\right.$ e $\left\{\begin{array}{l} 3y+5w=0 \\ 2y+3w=1 \end{array}\right.$
Resolvendo estas equações, acha-se a matriz inversa que é $\begin{pmatrix} -3 & 5 \\ -2 & 3 \end{pmatrix}$.

Usando determinantes: Outros métodos (extraídos do excelente site `http://mathworld.wolfram.com`) vão a seguir descritos: Dada uma matriz 2×2 representada por $A = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$, a inversa é dada por
$A^{-1} = \frac{1}{|A|}\begin{vmatrix} d & -b \\ -c & a \end{vmatrix} = \frac{1}{ad-bc}\begin{vmatrix} d & -b \\ -c & a \end{vmatrix}$.

Para uma matriz 3×3 veja lá no site, que o espaço aqui é muito pequeno.

Usando Gauss-Jordan: Neste método escreve-se a matriz da qual se quer achar a inversa ao lado esquerdo da matriz identidade, formando este arranjo (matriz escalonada reduzida por linhas).

$$[AI] = \begin{vmatrix} a_{11}\ ...\ a_{1n} & 1\ 0\ ...\ 0 \\ a_{21}\ ...\ a_{2n} & 0\ 1\ ...\ 0 \\ ... & ... \\ a_{n1}\ ...\ a_{nn} & 0\ 0\ ...\ 1 \end{vmatrix}$$

Aplica-se agora a este arranjo as técnicas de eliminação de Gauss, sobre o conjunto todo, de maneira a ficar com o seguinte arranjo

$$\begin{vmatrix} 1\ 0\ ...\ 0 & b_{11}\ ...\ b_{1n} \\ 0\ 1\ ...\ 0 & b_{21}\ ...\ b_{2n} \\ ... & ... \\ 0\ 0\ ...\ 1 & b_{n1}\ ...\ b_{nn} \end{vmatrix}$$

Neste ponto, a matriz (b_{ij}) é a matriz inversa de A. Ou seja $A^{-1} = B$ e também $AB = I$.

A vantagem deste último método é que ele vale para qualquer dimensão de matriz. É este método que vai ser programado em C++ a seguir.

Para que serve a matriz inversa ?

Inúmeras aplicações, mas vamos focar aqui na solução de sistemas lineares. Dado o sistema $Ax = B$, a matriz A é a dos coeficientes. Se você calcular a inversa A^{-1}, poderá resolver inúmeras instâncias do mesmo sistema, sem outra preocupação, já que $A^{-1} \times B = x$. Vamos ver um exemplo do que se diz aqui. Seja um sistema, dado por
$\left\{\begin{array}{lcl} 3x+2y+z & = & \\ x-y+z & = & \\ 5x+z & = & \end{array}\right\}$
Escrito na forma matricial, o sistema fica $Ax = B$, onde A é a matriz
$\begin{pmatrix} 3 & 2 & 1 \\ 1 & -1 & 1 \\ 5 & 0 & 1 \end{pmatrix}$
Calculando a matriz inversa desta, obtém-se:
$\begin{pmatrix} -0.1 & -0.2 & 0.3 \\ 0.4 & -0.2 & -0.2 \\ 0.5 & 1 & -0.5 \end{pmatrix}$
Agora, a solução para a instância onde $B = \{10, 2, 8\}$ é obtida fazendo-se $A^{-1} \times B$ que dá os valores $x = 1, y = 2, z = 3$. Para outra instância cujo B seja $\{30, -1, 26\}$, ao fazer a mesma multiplicação, obtém-se $x = 5, y = 7, z = 1$ e assim por diante.

Este exercício pede que você implemente o algoritmo de Gauss-Jordan para cálculo da matriz inversa e depois o aplique a uma matriz de ordem suficientemente alta para desencorajar o cálculo manual da mesma. Relembrando as etapas, você deve:

1. Escrever a matriz A e ao seu lado a matriz I.
2. Aplicar operações elementares com linhas a A visando transformá-la em I.
3. Aplicar rigorosamente as mesmas operações sobre a matriz ao lado.
4. Quando A tiver se transformado em I, a matriz que era originalmente I se transformou em A^{-1}.
5. Para conferir, faça a multiplicação matricial entre A e A^{-1}. Tem que dar I.

Vamos aplicar este processo ao sistema acima descrito. O objetivo é produzir um zero no coeficiente do x na segunda equação: Para isso:
$a[1][] = a[1][] + a[0] \times (-a[1][0]/a[0][0])$ onde a notação $a[1][]$ deve ser entendida como a aplicação da operaçao à toda a linha [1] da matriz a. Mas, atenção: isto não vale no compilador C++. Lá, tem que ser por extenso.

Note-se também que o fator $(-a[1][0]/a[0][0])$ é o mesmo para toda a linha e portanto deve ser calculado antes e preservado durante todo o ciclo. O algoritmo vai chamar este fator de *pivot* e ele vai ser previamente calculado e guardado.

para calcular o zero no coeficiente de x na terceira linha, o comando
é $a[2][] = a[2][] + a[0] \times (-a[2][0]/a[0][0])$.

Para a segunda coluna, começa-se achando o zero na primeira linha. O comando agora é
$a[0][] = a[0][] + a[1][] \times (-a[0][1]/a[1][1])$ e depois $a[2][] = a[2][] + a[1][] \times (-a[2][1]/a[1][1])$. A terceira coluna é
$a[0][] = a[0][] + a[2][] \times (-a[0][2]/a[2][2])$ e
$a[1][] = a[1][] + a[2][] \times (-a[1][2]/a[2][2])$.
Com estas operações, os zeros foram alcançados. Agora basta transformar os elementos da diagonal principal na unidade. As operações são:
$a[0][] = a[0][]/a[0][0]$, $a[1][] = a[1][]/a[1][1]$ e $a[2][] = a[2][]/a[2][2]$.

Note que não houve preocupação em determinar se a matriz A é inversível. Isto foi deixado para não complicar demais o algoritmo. Fica como desafio para os mais corajosos.

Eis o algoritmo:

```
#include<iostream>
#include<iomanip>
#define ta 5
using namespace std;
int cami(float x[ta][ta], float y[ta][ta]){
// x é a matriz A e y é a matriz A^-1
  int i,j,k;
  float pivot,alvo;
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (i==j) {
        y[i][j]=1;
      }
      else {
        y[i][j]=0;
      }
    }
  }
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (j!=i){
        pivot=-x[j][i]/x[i][i];
          for (k=0;k<ta;k++){
            x[j][k]=x[j][k]+x[i][k]*pivot;
            y[j][k]=y[j][k]+y[i][k]*pivot;
          }
      }
    }
    alvo=x[i][i];
    for (k=0;k<ta;k++){
      y[i][k]=y[i][k]/alvo;
      x[i][k]=x[i][k]/alvo;
    }
  }
  return 0;
}
int main() {
  float mat[ta][ta]={{...},{...},{},...};
  float res[ta][ta];
  int i,j;
  cami(mat,res);
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      cout<<fixed<<setprecision(5)
            <<res[i][j]<<"  ";
    }
    cout<<endl;
  }
}
```

Um exemplo Para você testar seu algoritmo, use-o nesta matriz:

```
 59 57 44 42 65 23 47 66 78 71
 94 36 83 17 12 86  4 92 43 95
 12 61 18 10 94 45 94 81 30 76
  9 49 92 22 90 22 27  6 21 26
 31 45 96 26 26 75 59 39 47 98
 33 27  9 94 29  5 49 68  9 33
 61 81 17 18 61 74 16 37 34 45
 64 85 46 74 49 81 69 20 18 59
 17 54 66 65 94 54 76 54 30 76
 68 91 61 89  6 57 35 92 52 80
```

Aqui a resposta deverá ser 151.989, já devidamente multiplicada por 1000.

☞ Para você fazer

A seguir uma matriz A de dimensão 10×10 (o que corresponderia a um sistema linear de 10 equações e 10 incógnitas). Você deve calcular a matriz inversa, seguindo o algoritmo acima, e depois responder qual o valor que está localizado na sexta linha e sétima coluna de A^{-1}, (em C: a[5][6]) multiplicado por 1000. Eis a matriz:

```
 58 61 59 50 87 48 96 25 27 26
 98 34 68 53 69 72 10 35 79 21
 79 36 53 74 20 35 64 99 87 99
 90 38 69 25 33 70 55 30 24 99
 69 19 12 67 85 67 67 56  5 64
 35  5 61  2 45 17 28 40 15 95
 96 34 91 83 23 66 33 43 80 22
 98  2 65 11 11 85 64 66 64 45
 20 67 51 55 65 23 51 57 97 39
 30 18 13 27 86 70 17 92 99 34
```

Responda aqui:

$A^{-1}[5][6] \times 1000 =$

Prof Dr P Kantek (pkantek@gmail.com)

Matrizes e Equações Lineares

O sistema

$$\left\{ \begin{array}{lcl} a_{11}\ x_1 + a_{12}\ x_2 + ... + a_{1n}\ x_n & = & b_1 \\ a_{21}\ x_1 + a_{22}\ x_2 + ... + a_{2n}\ x_n & = & b_2 \\ ... & & \\ a_{m1}\ x_1 + a_{m2}\ x_2 + ... + a_{mn}\ x_n & = & b_m \end{array} \right.$$

é equivalente à seguinte equação matricial

$$\left(\begin{array}{l} a_{11}\ a_{12}\ ...\ a_{1n} \\ a_{21}\ a_{22}\ ...\ a_{2n} \\ ... \\ a_{m1}\ a_{m2}\ ...\ a_{mn} \end{array} \right) \left(\begin{array}{c} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{array} \right) = \left(\begin{array}{c} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{array} \right)$$

no sentido de que toda solução do primeiro é igualmente solução do segundo. Dessa maneira pode-se resumir qualquer dos sistemas acima na forma matricial escrevendo

$$Ax = B$$

Matrizes Inversíveis

Diz-se que uma matriz quadrada é inversível se existe uma matriz B com a propriedade de que $AB = BA = I$ onde I é a matriz unidade.

A matriz B inversa de A é única se existir. A matriz inversa de A é indicada por A^{-1}. Note-se que a relação acima é simétrica, pois se B é a inversa de A então A também é inversa de B.

Mecanismos de cálculo da inversa

Usando a teoria: Usando a teoria, dada uma matriz como por exemplo $\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array} \right)$ Uma possibilidade é querer encontrar escalares x, y, z e w para os quais
$\left(\begin{array}{cc} 3 & 5 \\ 2 & 3 \end{array} \right) \left(\begin{array}{cc} x & y \\ z & w \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$ ou
$\left(\begin{array}{cc} 3x+5z & 3y+5w \\ 2x+3z & 2y+3w \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$
ou ainda equações que satisfaçam
$\left\{ \begin{array}{l} 3x+5z=1 \\ 2x+3z=0 \end{array} \right.$ e $\left\{ \begin{array}{l} 3y+5w=0 \\ 2y+3w=1 \end{array} \right.$
Resolvendo estas equações, acha-se a matriz inversa que é $\left(\begin{array}{cc} -3 & 5 \\ -2 & 3 \end{array} \right)$.

Usando determinantes: Outros métodos (extraídos do excelente site `http://mathworld.wolfram.com`) vão a seguir descritos: Dada uma matriz 2×2 representada por $A = \left| \begin{array}{cc} a & b \\ c & d \end{array} \right|$, a inversa é dada por
$A^{-1} = \frac{1}{|A|} \left| \begin{array}{cc} d & -b \\ -c & a \end{array} \right| = \frac{1}{ad-bc} \left| \begin{array}{cc} d & -b \\ -c & a \end{array} \right|$.

Para uma matriz 3×3 veja lá no site, que o espaço aqui é muito pequeno.

Usando Gauss-Jordan: Neste método escreve-se a matriz da qual se quer achar a inversa ao lado esquerdo da matriz identidade, formando este arranjo (matriz escalonada reduzida por linhas).

$$[AI] = \left| \begin{array}{ll} a_{11}\ ...\ a_{1n} & 1\ 0\ ...\ 0 \\ a_{21}\ ...\ a_{2n} & 0\ 1\ ...\ 0 \\ ... & ... \\ a_{n1}\ ...\ a_{nn} & 0\ 0\ ...\ 1 \end{array} \right|$$

Aplica-se agora a este arranjo as técnicas de eliminação de Gauss, sobre o conjunto todo, de maneira a ficar com o seguinte arranjo

$$\left| \begin{array}{ll} 1\ 0\ ...\ 0 & b_{11}\ ...\ b_{1n} \\ 0\ 1\ ...\ 0 & b_{21}\ ...\ b_{2n} \\ ... & ... \\ 0\ 0\ ...\ 1 & b_{n1}\ ...\ b_{nn} \end{array} \right|$$

Neste ponto, a matriz (b_{ij}) é a matriz inversa de A. Ou seja $A^{-1} = B$ e também $AB = I$.

A vantagem deste último método é que ele vale para qualquer dimensão de matriz. É este método que vai ser programado em C++ a seguir.

Para que serve a matriz inversa ?

Inúmeras aplicações, mas vamos focar aqui na solução de sistemas lineares. Dado o sistema $Ax = B$, a matriz A é a dos coeficientes. Se você calcular a inversa A^{-1}, poderá resolver inúmeras instâncias do mesmo sistema, sem outra preocupação, já que $A^{-1} \times B = x$. Vamos ver um exemplo do que se diz aqui. Seja um sistema, dado por
$\left\{ \begin{array}{lcl} 3x+2y+z & = & \\ x-y+z & = & \\ 5x+z & = & \end{array} \right\}$
Escrito na forma matricial, o sistema fica $Ax = B$, onde A é a matriz
$\left(\begin{array}{ccc} 3 & 2 & 1 \\ 1 & -1 & 1 \\ 5 & 0 & 1 \end{array} \right)$
Calculando a matriz inversa desta, obtém-se:
$\left(\begin{array}{ccc} -0.1 & -0.2 & 0.3 \\ 0.4 & -0.2 & -0.2 \\ 0.5 & 1 & -0.5 \end{array} \right)$
Agora, a solução para a instância onde $B = \{10, 2, 8\}$ é obtida fazendo-se $A^{-1} \times B$ que dá os valores $x = 1, y = 2, z = 3$. Para outra instância cujo B seja $\{30, -1, 26\}$, ao fazer a mesma multiplicação, obtém-se $x = 5, y = 7, z = 1$ e assim por diante.

Este exercício pede que você implemente o algoritmo de Gauss-Jordan para cálculo da matriz inversa e depois o aplique a uma matriz de ordem suficientemente alta para desencorajar o cálculo manual da mesma. Relembrando as etapas, você deve:

1. Escrever a matriz A e ao seu lado a matriz I.
2. Aplicar operações elementares com linhas a A visando transformá-la em I.
3. Aplicar rigorosamente as mesmas operações sobre a matriz ao lado.
4. Quando A tiver se transformado em I, a matriz que era originalmente I se transformou em A^{-1}.
5. Para conferir, faça a multiplicação matricial entre A e A^{-1}. Tem que dar I.

Vamos aplicar este processo ao sistema acima descrito. O objetivo é produzir um zero no coeficiente do x na segunda equação: Para isso:
$a[1][] = a[1][] + a[0] \times (-a[1][0]/a[0][0])$ onde a notação $a[1][]$ deve ser entendida como a aplicação da operaçao à toda a linha [1] da matriz a. Mas, atenção: isto não vale no compilador C++. Lá, tem que ser por extenso.

Note-se também que o fator $(-a[1][0]/a[0][0])$ é o mesmo para toda a linha e portanto deve ser calculado antes e preservado durante todo o ciclo. O algoritmo vai chamar este fator de *pivot* e ele vai ser previamente calculado e guardado.

para calcular o zero no coeficiente de x na terceira linha, o comando
é $a[2][] = a[2][] + a[0] \times (-a[2][0]/a[0][0])$.

Para a segunda coluna, começa-se achando o zero na primeira linha. O comando agora é
$a[0][] = a[0][] + a[1][] \times (-a[0][1]/a[1][1])$ e depois $a[2][] = a[2][] + a[1][] \times (-a[2][1]/a[1][1])$. A terceira coluna é
$a[0][] = a[0][] + a[2][] \times (-a[0][2]/a[2][2])$ e
$a[1][] = a[1][] + a[2][] \times (-a[1][2]/a[2][2])$.
Com estas operações, os zeros foram alcançados. Agora basta transformar os elementos da diagonal principal na unidade. As operações são:
$a[0][] = a[0][]/a[0][0]$, $a[1][] = a[1][]/a[1][1]$ e $a[2][] = a[2][]/a[2][2]$.

Note que não houve preocupação em determinar se a matriz A é inversível. Isto foi deixado para não complicar demais o algoritmo. Fica como desafio para os mais corajosos.

Eis o algoritmo:

```
#include<iostream>
#include<iomanip>
#define ta 5
using namespace std;
int cami(float x[ta][ta], float y[ta][ta]){
// x é a matriz A e y é a matriz A^-1
  int i,j,k;
  float pivot,alvo;
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (i==j) {
        y[i][j]=1;
      }
      else {
        y[i][j]=0;
      }
    }
  }
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      if (j!=i){
        pivot=-x[j][i]/x[i][i];
          for (k=0;k<ta;k++){
            x[j][k]=x[j][k]+x[i][k]*pivot;
            y[j][k]=y[j][k]+y[i][k]*pivot;
          }
      }
    }
    alvo=x[i][i];
    for (k=0;k<ta;k++){
      y[i][k]=y[i][k]/alvo;
      x[i][k]=x[i][k]/alvo;
    }
  }
  return 0;
}
int main() {
  float mat[ta][ta]={{...},{...},{},...};
  float res[ta][ta];
  int i,j;
  cami(mat,res);
  for (i=0;i<ta;i++){
    for (j=0;j<ta;j++){
      cout<<fixed<<setprecision(5)
            <<res[i][j]<<"  ";
    }
    cout<<endl;
  }
}
```

Um exemplo Para você testar seu algoritmo, use-o nesta matriz:

```
 59 57 44 42 65 23 47 66 78 71
 94 36 83 17 12 86  4 92 43 95
 12 61 18 10 94 45 94 81 30 76
  9 49 92 22 90 22 27  6 21 26
 31 45 96 26 26 75 59 39 47 98
 33 27  9 94 29  5 49 68  9 33
 61 81 17 18 61 74 16 37 34 45
 64 85 46 74 49 81 69 20 18 59
 17 54 66 65 94 54 76 54 30 76
 68 91 61 89  6 57 35 92 52 80
```

Aqui a resposta deverá ser 151.989, já devidamente multiplicada por 1000.

☞ Para você fazer

A seguir uma matriz A de dimensão 10×10 (o que corresponderia a um sistema linear de 10 equações e 10 incógnitas). Você deve calcular a matriz inversa, seguindo o algoritmo acima, e depois responder qual o valor que está localizado na sexta linha e sétima coluna de A^{-1}, (em C: a[5][6]) multiplicado por 1000. Eis a matriz:

```
 86 30 72 71 16 86  6 64 81 26
 94 45 63 63 71 75 74 36 73 44
 23 94  6 87 98 15 10 11 28 68
 27  2 53 39 60 85 36 52 86 71
 55 62 38 89  6 93 71 35 11 92
 93  9  2 94  1  8 43  9 34 18
 31 11 33 68 47 12 86 65 15  4
 62 71 57 98 38 41 93 60 88 39
 89 94 53 10 82 43 38 53 38 37
 54 69 25 21  2 35 78 98  1 93
```

Responda aqui:

$A^{-1}[5][6] \times 1000 =$

```
===== 30/05/2018 10:16:31.4 =====E=PL151
    1    24.467
    2     -.886
    3   -24.063
    4    14.974
    5    23.229
```