Sumário

1 PRG20

1.1 matriz inversa

Para acompanhar esta explicação projete o apres_mat_inversa.pdf.

Seja agora o problema clássico de achar a matriz inversa de uma matriz dada. Quem já resolveu este problema em matriz 3×3 sabe o trabalho que dá. Pense agora achar a inversa de matriz 50×50. Vamos lá.

Lembrando o processo, escreve-se a matriz de quem se quer obter a inversa e ao seu lado uma matriz identidade. Depois, mediante operações elementares (i. multiplicação por constante, ii. substituição de uma linha por 2 somadas e iii.troca de linhas) obtém-se a identidade no lugar da matriz original aplicando as mesmas operações naquela que começou como a identidade. Quando a matriz original virar a identidade, a que era a identidade virou a inversa. Vamos ao código

```
 #include<iostream>
#include<iomanip>
#define T 4
using namespace std;
int inversa(float M[T][T], float I[T][T]){
   int i,j,k;
   float pivot,alvo;
   for (i=0;i<T;i++){
    for (j=0;j<T;j++){
        if (i==j){I[i][j]=1;}else{I[i][j]=0;}}}
   for (i=0;i<T;i++){
      for (j=0;j<T;j++){
        if (i!=j){
            pivot=-M[j][i]/M[i][i];
            for (k=0;k<T;k++){
                M[j][k]=M[j][k]+M[i][k]*pivot;
                I[j][k]=I[j][k]+I[i][k]*pivot;
            }
        }
      }
      alvo=M[i][i];
      for(k=0;k<T;k++){
        M[i][k]=M[i][k]/alvo;
        I[i][k]=I[i][k]/alvo;
      }
   }
   return 0;
}
int main(){
    int i,j;
    float m[T][T]={35,20,50,26,14,3,17,47, \
                   18,7,5,17,21,40,2,49};
    float in[T][T];
    inversa(m,in);
    cout.precision(2);
    for (i=0;i<T;i++){
        for (j=0;j<T;j++){
            cout<<setw(10)<<in[i][j];
        }
        cout<<endl;
    }   }
```

1.2 (USP8.14: quadrado latino)

Diz-se que uma matriz $A_{n\times n}$ é um *quadrado latino* de ordem n se em cada linha e em cada coluna aparecem todos os inteiros $1, 2, 3, ...n$ ou seja cada linha e cada coluna é permutação dos inteiros $1, 2, 3, ...n$. Exemplo

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$

a matriz acima é um quadrado latino de ordem 4.

(a) Escreva uma função que receba como parâmetros um inteiro n e um vetor v com n inteiros e verifica se em v ocorrem todos os inteiros de 1 a n.

(b) Dados um inteiro positivo n e uma matriz inteira $A_{n\times n}$ usando a função acima, verifique se a matriz A é um quadrado latino de ordem n.

```
int vetlat(int n, int v[10]){
  int i,j,resp,qt=0;
  for (j=1;j<=n;j++){
    resp=0;
    for (i=0;i<n;i++){
        if (v[i]==j){resp=1;}
    }
    qt=qt+resp;
  }
  return qt==n;
}
int main(){
  int n,i,j,linhas=0,colunas=0,resposta;
  int a[10][10];
  int v[10];
  cin>>n;
  for (i=0;i<n;i++){
    for(j=0;j<n;j++){
        cin>>a[i][j];
    }
  }
  //linhas
  for(i=0;i<n;i++){
    linhas=linhas+vetlat(n,a[i]);
  }
  //colunas
  for (i=0;i<n;i++){
    for(j=0;j<n;j++){
        v[j]=a[j][i];
    }
    colunas=colunas+vetlat(n,v);
  }
  resposta=(linhas==n)&&(colunas==n);
  cout<<resposta<<endl;
}
```

1.3 (USP8.13: média na matriz)

FEA 88

(a) Escreva uma função que receba como parâmetro dois inteiros positivos m e n uma matriz real $A_{m\times n}$ e uma posição i,j da matriz, calcula a média aritmética dos quatro (ou três, ou dois) vizinhos do elemento (i, j) da matriz. Desconsidere os vizinhos que não pertencem a matriz. Por exemplo, os vizinhos de 0,0 são apenas o 1,0 e 0,1.

(b) Escreva uma função que recebe os valores m, n e $A_{m\times n}$ e devolve a matriz A^{media}, onde $A^{media}_{i,j}$ é a média aritmética dos vizinhos de i, j calculados pela função anterior.

(c) Escreva um programa que lê três inteiros, m, n e k e uma matriz $A_{m\times n}$ e utilizando a função anterior transforma a matriz k vezes, imprimindo a matriz inicial e a obtida após cada transformação.

```
#include<iostream>
#include<iomanip>
using namespace std;
float media(int m, int n, int i, int j, float mat[10][10]){
   int qt=0;
   float so=0.0;
   if (i>0){qt=qt+1; so=so+mat[i-1][j];}
   if (i<(m-1)){qt=qt+1;so=so+mat[i+1][j];}
   if (j>0){qt=qt+1; so=so+mat[i][j-1];}
   if (j<(n-1)){qt=qt+1;so=so+mat[i][j+1];}
   return so/qt;
}
void matmed(int m, int n, float mat[10][10], float med[10][10]){
   int i,j;
   for (i=0;i<m;i++){
    for(j=0;j<n;j++){
        med[i][j]=media(m,n,i,j,mat);
    }
   }
}
void print(int m, int n, float mat[10][10]){
  int i,j;
  for (i=0;i<m;i++){
    for(j=0;j<n;j++){
        cout<<setprecision(3);
        cout<<setw(9);
        cout<<mat[i][j]<<" ";
    }
    cout<<endl;
  }
}
void copia(int m, int n, float mat[10][10], float m2[10][10]){
  int i,j;
  for (i=0;i<m;i++){
    for(j=0;j<n;j++){
       m2[i][j]=mat[i][j];
    }
  }
}
int main(){
   int m,n,i,j,k;
   float mat[10][10], m2[10][10];
   cin>>m>>n>>k;
   for (i=0;i<m;i++){
    for(j=0;j<n;j++){
        cin>>mat[i][j];
    }
   }
   print(m,n,mat);
   for(;k>0;k--){
      matmed(m,n,mat,m2);
      cout<<"-----------------"<<endl;
      print(m,n,m2);
      copia(m,n,m2,mat);
   }
}
```

Nota de implementação: A função de cópia de uma matriz para a outra (função `copia` acima), poderia ser pensada de outra maneira, talvez para ser mais eficiente. Há uma discussão grande sobre isso, mas não há nenhuma *bala de prata*. Assim, para manter as coisas simples e 100% funcionais em qualquer ambiente, usa-se a solução óbvia: mover elemento a elemento. Funciona e é confiável.

1.4 USP7.6 Linhas e colunas nulas

USP7.6 Dados dois inteiros positivos $m(\leq 10)$ e $n(\leq 10)$ e uma matriz $A_{m\times n}$ imprimir o número de linhas e de colunas nulas da matriz. Exemplo: $m = 4$ e $n = 4$

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ 4 & 0 & 5 & 6 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$ tem 2 linhas e 1 coluna nulas.

```
#include<iostream>
using namespace std;
int main(){
   int m,n,i,j,col,lin,qtd=0;
   float a[10][10];
   cin>>m>>n;
   for (i=0;i<m;i++){
      for (j=0;j<n;j++){
         cin>>a[i][j];
      }
   }
   for (i=0;i<m;i++){
      lin=0;
      for (j=0;j<n;j++){
         if (a[i][j]!=0){lin=1;}
      }
      if (lin==0){qtd++;}
   }
   for (i=0;i<n;i++){
      col=0;
      for (j=0;j<m;j++){
         if (a[j][i]!=0){col=1;}
      }
      if (col==0){qtd++;}
   }
   cout<<qtd;  }
```

1.5 (trocaelemens)

Escreva um programa C++ que obtenha do usuário uma matriz quadrada (dimensão máxima de 10×10) e troque o maior elemento de cada linha com o elemento da diagonal principal.

```
//---------  trocaelems -----------------
#include<iostream>
using namespace std;
#define T 5
void leitura(int mat[T][T]){
  int i,j;
  cout<<"Forneca "<<T<<" x "<<T<<" valores"<<endl;
  for (i=0;i<T;i++){
    for(j=0;j<T;j++){
        cin>>mat[i][j];
    }
  }
  return;
}
int main(){
  int i,j,mai,salvaj,aux;
//  int mat[T][T];
//  leitura(mat);
  int mat[T][T]={{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15},
  {16,17,18,19,20},{21,22,23,24,25}};
  for (i=0;i<T;i++){
     mai=-9999999;
     for (j=0;j<T;j++){
```

```
        if (mat[i][j]>mai){
            mai=mat[i][j];
            salvaj=j;
        }
     }
     aux=mat[i][i];
     mat[i][i]=mai;
     mat[i][salvaj]=aux;
  }
  for(i=0;i<T;i++){
    for(j=0;j<T;j++){
        cout<<mat[i][j]<<" ";
    }
    cout<<endl;
  }
}
```