Sumário

1 PRG21

1.1 USP7.7 - quadrado mágico

USP7.7 Dizemos que uma matriz quadrada é um quadrado mágico se a soma dos elementos de cada linha, a soma dos elementos de cada coluna e a soma dos elementos das diagonais principais e secundária são todos iguais. Por exemplo, a matriz A
$\begin{pmatrix} 8 & 0 & 7 \\ 4 & 5 & 6 \\ 3 & 10 & 2 \end{pmatrix}$ é um quadrado mágico. Dado um inteiro n e uma matriz quadrada $n \times n$ verifique se A é um quadrado mágico.

```
#include<iostream>
using namespace std;
int main(){
   int n,i,j,quadr=1,somac=0,sl=0,sc=0;
   float a[10][10];
   cin>>n;
   for (i=0;i<n;i++){
      for (j=0;j<n;j++){
         cin>>a[i][j];
      }
   }
   for (i=0;i<n;i++){somac=somac+a[i][i];}
// calculo das linhas
   for (i=0;i<n;i++){
       sl=0;sc=0;
       for (j=0;j<n;j++){
           sl=sl+a[i][j];
           sc=sc+a[j][i];
       }
       if ((sl!=somac)||(sc!=somac)){quadr=0;}
   }
   sl=0;
   for (i=0;i<n;i++){
      sl=sl+a[i][(n-1)-i];
   }
   if (sl!=somac){quadr=0;}
   cout<<quadr; }
```

1.2 USP7.8 - Triângulo de pascal

USP7.8 Dado um inteiro positivo n imprimir as n primeiras linhas do Triângulo de Pascal.

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5  10 10 5  1
```

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
```

```
   int n,i,j;
   float p[20][20]={0};
   cin>>n;
   for (i=0;i<n;i++){
      p[i][i]=p[i][0]=1;
   }
   for (i=1;i<n;i++){
      for (j=1;j<i;j++){
        p[i][j]=p[i-1][j]+p[i-1][j-1];
      }
   }
   for (i=0;i<n;i++){
    for (j=0;j<n;j++){
        cout<<setw(5)<<p[i][j];
    }
    cout<<endl;
   }   }
```

O mesmo enunciado, usando apenas dois vetores

```
int main(){
   int n,i,j;
   int p[20]={0};
   int ant[20]={0};
   cin>>n;
   p[0]=1; ant[0]=1;
   for (i=1;i<n;i++){
      cout<<"   1";
      for(j=1;j<i;j++){
          p[j]=ant[j]+ant[j-1];
          cout<<setw(4)<<p[j];
      }
      cout<<endl;
      for (j=0;j<n;j++)
         ant[j]=p[j];
      }   }
```

1.3 USP8.8d - matriz identidade ?

USP8.8d - Faça uma função que verifica se uma matriz $m \times m$ é a matriz identidade

```
int ident(int a[m][m]){
  int i,j,e=1;
  for (i=0;i<m;i++){
    for (j=0;j<m;j++){
      if ((i!=j)&&(a[i][j]!=0)){e=0;}
      if ((i==j)&&(a[i][j]!=1)){e=0;}
    }
  }
  return e;
}
```

1.4 USP8.16c - multiplicação matricial

USP8.16c - Escreva uma função que receba duas matrizes quadradas A e B de ordem m e devolva (também via parâmetro) a multiplicação matricial de A e B.

```
#include<iostream>
#include<iomanip>
#define m 3
using namespace std;
void mm(int a[m][m], int b[m][m], int r[m][m]){
  int i,j,k,s;
  for (i=0;i<m;i++){
    for (j=0;j<m;j++){
```

```
      s=0;
      for (k=0;k<m;k++){
        s=s+a[i][k]*b[k][j];
      }
      r[i][j]=s;
    }
  }
}
int  main (){
  int a[3][3]={1,2,3,4,5,6,7,8,9};
  int b[3][3]={2,4,6,8,10,12,14,16,18};
  int r[3][3]={0};
  mm(a,b,r);
  int i,j;
  for (i=0;i<3;i++){
      for(j=0;j<3;j++){
        cout<<setw(4)<<r[i][j];
      }
      cout<<endl;
  }  }
----------------- resposta é ------------
 60   72   84
132  162  192
204  252  300
```

1.5 Lagrange

LAGRANGE - Existe um algoritmo de interpolação, denominado Lagrange, que recebendo dois vetores, denominados x e fx de mesmo comprimento e um novo valor chamado nx, calcula o polinômio interpolador (a partir de x e fx) e depois aplica o valor nx a este polinômio imprimindo o valor de $fx(nx)$.

$$P_n(x) = \sum_{k=0}^{n} y_k L_k(x)$$

e

$$L_k(x) = \frac{\prod_{\substack{j=0 \\ j\neq k}}^{n} (x - x_j)}{\prod_{\substack{j=0, \\ j\neq k}}^{n} (x_k - x_j)}$$

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
  int n,i,j;
  float x[100];
  float fx[100];
  float nx;
  cin>>n;
  cout<<"----- valores de x ------"<<endl;
  for (i=0;i<n;i++){cin>>x[i];}
  cout<<"----- valores de fx ------"<<endl;
  for (i=0;i<n;i++){cin>>fx[i];}
  cout<<" Valor de nx ";
  cin>>nx;
  float yp=0.0;
  for (i=0;i<n;i++){
    float p=1.0;
    for (j=0;j<n;j++){
        if (i!=j){p=p*(nx-x[j])/(x[i]-x[j]);}
    }
    yp=yp+p*fx[i];
```

```
  }
  cout<<yp;   }
```

1.6 Mult. matr. é comutativa ?

Sabe-se que, em geral, a multiplicação matricial não é comutativa, isto é: A.B $\neq$ B.A, embora em alguns casos seja. Escreva uma função que receba 2 matrizes A e B ambas de ordem n ($n < 20$) e responda 1 se sua multiplicação matricial for comutativa e 0 senão.

```
#include<iostream>
#define m 4
using namespace std;
void mm(int a[m][m], int b[m][m], int r[m][m]){
  int i,j,k,s;
  for (i=0;i<m;i++){
    for (j=0;j<m;j++){
      s=0;
      for (k=0;k<m;k++){
        s=s+a[i][k]*b[k][j];
      }
      r[i][j]=s;
    }
  }
}
int main(){
   int n,i,j;
   int a[4][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
   int b[4][4]={0,0,1,2,3,4,5,6,0,0,7,8,9,0,0,0};
   // cin>>n;
   // for(i=0;i<n;i++){
   //   for(j=0;j<n;j++){
   //      cin>>a[i;j]>>b[i;j]; }}
   n=4;
   int rab[4][4];
   int rba[4][4];
   mm(a,b,rab);
   mm(b,a,rba);
   int e=1;
   for (i=0;i<n;i++){
      for(j=0;j<n;j++){
         if (rab[i][j]!=rba[i][j]){e=0;}
      }
   }
   cout<<e;
}
```