Sumário

1 PRG22

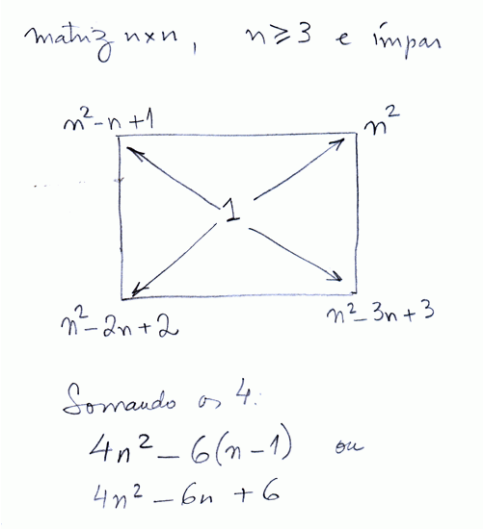
1.1 vivo035c - soma das diagonais crescentes

SOMA DAS DIAGONAIS (vivo035c) - EULER28 - Começando com o número 1 e movendo-se para a direita no sentido horário, uma espiral de 5 por 5 é formada da seguinte forma:

21	22	23	24	25
20	7	8	9	10
19	6	1	2	11
18	5	4	3	12
17	16	15	14	13

Pode-se verificar que a soma dos números nas diagonais é 101.

Qual é a soma dos números nas diagonais em uma espiral de 1001 por 1001 formada da mesma maneira?

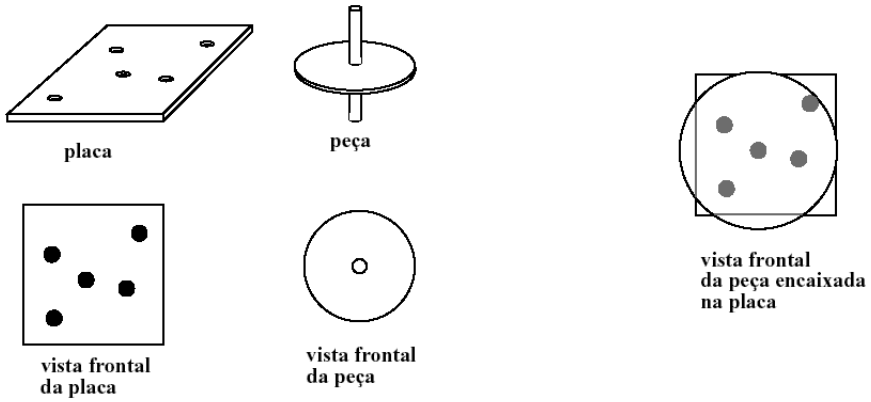



```
#include<iostream>
using namespace std;
long long int eulr(long long int k) {
   if (k>2) {
      return ((4*k*k) - (6*(k-1)) + eulr(k-2));
   }
   else {
      return 1;
   }
}
long long int euli(long long int k) {
   int j, s=1;
   for (j=3;j<=k;j=j+2) {
      s=s+(4*j*j)-(6*(j-1));
   }
   return s;
}
main() {
   long long int n,s=0;
   cout<<"Informe tam matriz (impar e > 2)";
   cin>>n;
```

```
   cout<<eulr(n)<<endl; // recursivo
   cout<<euli(n)<<endl;  } // iterativo,
   //   ambos dao o mesmo valor
```

1.2 VIVOm34 - cubra os furos

Exercício de maratona: CUBRA OS FUROS Uma placa de aço retangular contém N furos circulares de 5 mm de diâmetro, localizados em pontos distintos, não sobrepostos – ou seja, o centro de cada furo está a uma distância maior ou igual a 5 mm do centro de todos os outros furos. Uma peça de forma circular, tendo em seu centro um eixo de 5 mm de diâmetro, deve ser colocada sobre a placa, de modo que o eixo encaixe-se em um de seus furos. Tarefa: Você deve escrever um programa para determinar o diâmetro mínimo que a peça deve ter de tal forma que, com seu eixo encaixado em um dos furos da placa, a parte circular cubra completamente todos os outros furos da placa. Deve informar também qual é o furo que receberá a placa.



Estratégia de solução:

- Ache a distância de cada furo para todos os outros
- Guarde a maior destas distâncias, já que com a chapa disposta neste furo que está sendo examinado, a maior distância é a única que cobre todos os demais.
- Com o vetor preenchido, escolha o menor valor. Seu índice é o furo que deve ser usado.
- Ao guardar a distância, não esquecer de somar 2.5 (o diâmetro do furo) e multiplicar por 2 para passar de raio a diâmetro.

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
   int n,i,j,isal;
   float pt[n][2];
   float md[n];
   float maxd,mind,d,p1,p2;
   cin>>n;
   for (i=0;i<n;i++){cin>>pt[i][0]>>pt[i][1];}
   for (i=0;i<n;i++){
        maxd=-99999;
      for (j=0;j<n;j++){
        if (i!=j){
                p1=pow((pt[i][0]-pt[j][0]),2);
                p2=pow((pt[i][1]-pt[j][1]),2);
                d=sqrt(p1+p2);
                if (d>maxd){maxd=d;}
        }
      }
      md[i]=(maxd+2.5)*2;
   }
   mind=9999999;
   for(i=0;i<n;i++){
        if (md[i]<mind){mind=md[i]; isal=i;}
   }
```

```
   cout<<mind<<" "<<isal;
}
```

Para testar: 6, 48,78; 65,18; 43,24; 17,71; 50,53 e 70,50. Resposta: 81.157, furo 4.
Mais um: 4, 54,47; 51,31; 61,22 e 28,27. Resposta: 51.690, furo 1.

1.3 vivom44 - Quantos fibonaccis ?

QUANTOS FIBONACCIS ? (vivom44) Relembre-se a definição dos números de Fibonacci
$f_1 = 1$
$f_2 = 1$
$f_n = f_{n-1} + f_{n-2}, (n \geq 3)$
Dados dois números a e b calcule quantos números de Fibonacci existem no intervalo $[a, b]$. Note que deve usar *long long* por que estes números crescem rapidamente.

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
   long long i,j;
   long long a,c,b,qtd=0;
   cin>>i>>j;
   a=1;
   b=1;
   c=2;
   while(c<=j){
      c=a+b;
      if ((c>=i)&&(c<=j)){qtd++;}
      a=b;
      b=c;
   }
   cout<<"deu: "<<qtd;  }
```

Para testar: 10,100, resposta 5
1234567890,9876543210, resposta 4

1.4 UVA136 e vivom54 - números feios

NÚMEROS FEIOS (vivom54) - Originalmente UVA136 -

Números feios-UVA:136 O exercício a seguir não é igual ao 136 da UVA. Está apenas baseado nele. Números feios são aqueles cujos únicos fatores primos são 3 primos previamente determinados.

Por exemplo, se escolhermos os primos 2, 3 e 5, os primeiro 11 números feios são: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ... Por convenção, o 1 é sempre o primeiro.

Se os primos forem 5, 7 e 11, os 11 primeiros serão: 1, 5, 7, 11, 25, 35, 49, 55, 77, 121, 125, ...

Se os primos forem 2, 13 e 19, os 11 primeiros serão: 1, 2, 4, 8, 13, 16, 19, 26, 32, 38, 52, ...

Você deve escrever um programa que receba os 3 primos e o número de ordem do feio a gerar e calcule o feio pedido.

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
   int p1,p2,p3,n,i=2,iaux,qt=1;
   cin>>p1>>p2>>p3>>n;
   cout<<"1 ";
   while (1==1){
      if (qt==n){break;}
      iaux=i;
      while (0==i%p1){i=i/p1;}
      while (0==i%p2){i=i/p2;}
      while (0==i%p3){i=i/p3;}
      if (i==1){cout<<iaux<<" ";qt++;}
      i=iaux+1;
   }   }
```

Para testar: 5 3 13 55 é 6591
2 3 19 44 é 456
3 5 19 52 é 7695

1.5 Desvio padrão

DESVIO PADRÃO: Escreva um programa que leia n e depois essa quantidade de números reais, e ao final imprima o desvio padrão da distribuição, cuja formulação é

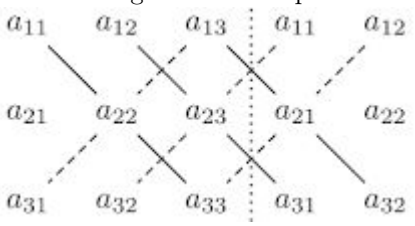
$$\sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
   int n,i;
   float x[100];
   float md,su=0,aux;
   cin>>n;
   for (i=0;i<n;i++){
      cin>>x[i];
      su=su+x[i];
   }
   md=su/n;
   su=0;
   for (i=0;i<n;i++){
      su=su+((x[i]-md)*(x[i]-md));
   }
   su=sqrt(su/n);
   cout<<su;    }
```

Para testar: 5, 1 2 3 10 20. Resposta: 7.138 Mais: 6, 20 50 50 60 70 90. Resposta: 21.343

1.6 Sarrus

SARRUS - Escreva um programa C++ que receba uma matriz 3×3, calcule e imprima o determinante desta matriz usando a regra de Sarrus para efetuar o cálculo.




```
#include<iostream>
using namespace std;
float a[3][3]={{2,4,5},{3,7,0},{1,1,5}};
float sarrus(float x[3][3]){
   float a1,a2;
   a1=x[0][0]*x[1][1]*x[2][2];
   a1=a1+x[0][2]*x[1][0]*x[2][1];
   a1=a1+x[2][0]*x[0][1]*x[1][2];
   a2=x[0][2]*x[1][1]*x[2][0];
   a2=a2+x[0][1]*x[1][0]*x[2][2];
   a2=a2+x[0][0]*x[2][1]*x[1][2];
   return a1-a2;
}
main() {
   cout<<sarrus(a);
}
```

1.7 (matsimetrica)

Escreva uma função que receba uma matriz de ordem 9 e devolva 1 se a matriz é simétrica e 0 senão.

```
#include<iostream>
using namespace std;
int msim(int a[9][9]){
   int e=1;
   int i,j;
   for (i=0;i<9;i++){
      for (j=0;j<9;j++){
          if (a[i][j]!=a[j][i]) {
             e=0;
          }
      }
   }
   return (e);
}
int main() {
   int x[9][9];
   int mi,mj;
   for (mi=0;mi<9;mi++){
      for (mj=0;mj<9;mj++){
         x[mi][mj]=0;
      }
   }
   x[3][5]=99;
   cout<<msim(x)<<endl;
   x[3][5]=0;
   cout<<msim(x)<<endl;
}
```