Introdução à Programação

Curso de Verão - IME-USP

Régis S. Santos

2012

Sumário

Prefácio

Esta apostila foi criada a partir de notas de aula do Curso de Verão *Introdução à Programação* realizado no IME-USP em 2012.

É permitida a reprodução total ou parcial desta apostila desde que indicada a autoria.

Esta apostila foi criada para uso pessoal, portanto, adaptada para tal fim. E está sujeito a conter erros, portanto, são aceitas sugestões e críticas construtivas para melhoria do mesmo.

Os livros aqui citados [1], [2] servem como material complementar para um estudo mais aprofundado.

Links

`http://www.ime.usp.br/~macmulti/`

`http://www.ime.usp.br/~hitoshi/introducao/`

`http://www.ime.usp.br/~mac2166/livros/rede.html`

Régis da Silva Santos
IME-USP, 2012.

Parte I

Introdução

1 Introdução

1.1 Computador a papel

http://www.ime.usp.br/~vwsetzer/comp-papel.html

Exemplo 1.1 Computador a papel

```
1 carregue no acumulador o [end.14]
2 armazene o [AC] no end.12
3 leia um número e armazene no end.13
4 escreva [end.13]
5 carregue no AC o [end.12]
6 some o [end.13] ao acumulador e coloque no acumulador
7 armazene no AC o [end.12]
8 carregue no acumulador o [end.13]
9 se [AC] ≠ 0 desvie para o end.3
10 imprima o [end.12]
11 pare
```

Processador

```
1 lê o comando
2 move o apontador para o próximo
3 executa o comando
```

1.2 Dispositivos de entrada e saída

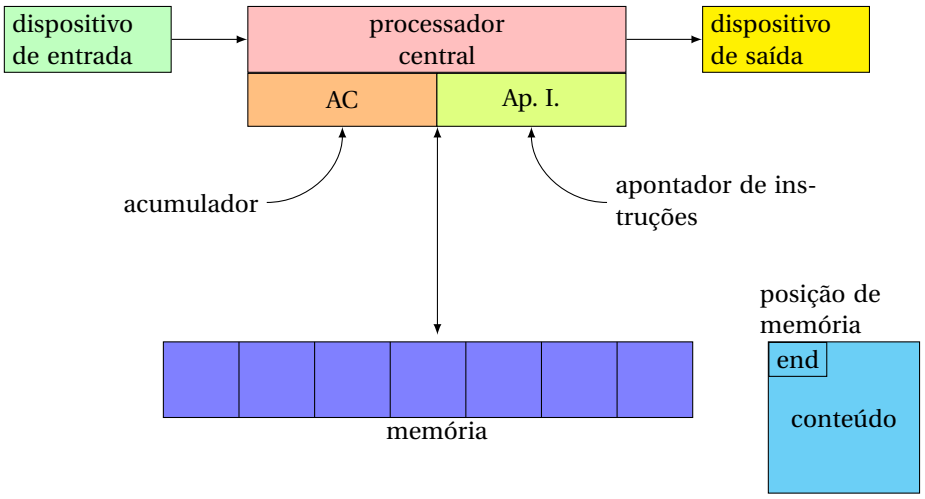


Figura 1.1: Dispositivos de entrada e saída

1.3 Instalando o Code Blocks

http://www.codeblocks.org/
Veja *instalando CodeBlocks no Linux.pdf* e *instalando CodeBlocks no Windows.pdf*.

2 Compilando pelo terminal no Linux

Para compilar pelo *terminal*, se o arquivo for escrito em `C` digite
`gcc -Wall -pedantic -ansi -o <arquivo> <arquivo.c>`
em `C++`
`g++ -Wall -pedantic -ansi -o <arquivo> <arquivo.cpp>`

Para executar o binário digite
`./arquivo`
NOTA: Devido algumas situações particulares do meu HD vou citar aqui os problemas e soluções que eu encontrei durante a execução dos arquivos:
Meu HD está montado com a opção *showexec*.

1. se o arquivo *prob01.cpp* estiver na **pasta pessoal** e for compilado via Code::Blocks será executado um binário *prob01* (sem extensão) e o mesmo será executado normalmente. OK

   a) se este mesmo binário for executado via *terminal* através do comando `./prob01` também será executado normalmente. OK

2. se o arquivo *prob01.cpp* estiver (no meu caso num HD particionado) e for compilado via Code::Blocks será executado um binário *prob01* (sem extensão) e o mesmo NÃO será executado. Permissão negada. FALHOU.

   a) se este mesmo binário for executado via terminal através do comando `./prob01` também NÃO será executado. Permissão negada. FALHOU.

   b) SOLUÇÃO: Para compilar via *terminal* num HD particionado (no meu caso) devemos gerar um *prob01.exe*, então no terminal façamos
   `g++ -Wall -pedantic -ansi -o prob01.exe prob01.cpp`
   Para executar digite
   `./prob01.exe`

`-Wall` retorna possíveis erros durante a compilação. Abreviação para Warnings: all.
`-pedantic` é para fazer uma compilação pedante, ou seja, ser bem estrito na interpretação do código checando tipos nas atribuições de variáveis e afins.
`-ansi` utiliza o padrão ANSI da linguagem. Serve para portabilidade.
`-o`, junto com `-Wall` acusa que variáveis podem não estar inicializadas.

3 Introdução à Programação

Exemplo 3.1 Pseudo-código

```
inteiro soma, numero
soma ← 0
leia (numero)
escreva numero
enquanto (numero ≠ 0) faça
{
    soma ← soma + numero
    leia (numero)
    escreva (numero)
}
escreva soma
```

Simulação 1 simulação

numero	soma	linha	
?	?	1	
?	0	2	
2	0	3	lê o 2
2	0	4	escreve 2
2	0	5	
2	2	6	
13	2	7	lê o 13
13	2	8	escreve 13
13	2	5	
13	15	6	
0	15	7	lê o 0
0	15	8	escreve 0
0	15	5	
0	15	9	escreve 15

Exemplo 3.2 (C++) C++

```cpp
#include <iostream>
using namespace std;
int main()
{
    int numero, soma;
    soma = 0;
    cin >> numero;
    cout << numero << endl;
    while (numero != 0)
    {
        soma = soma + numero;
        cin >> numero;
        cout << numero << endl;
    }
    cout << "soma = " << soma << endl;
    return 0;
}
```

3.1 Esqueleto de um programa em C++

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5   /*declaracao de variaveis*/
6   /*comandos*/
7   return 0;
8 }
```

Problema I.1 Dada uma sequência de números inteiros diferentes de zero, terminada por um zero, calcular a sua soma. Por exemplo, para a sequência 12 7 4 -6 8 0 o seu programa deve escrever o número 35.

Solução:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int numero, soma;
6     soma = 0;
7     cin >> numero;
8     cout << numero << endl;
9     while (numero != 0)
10    {
11        soma = soma + numero;
12        cin >> numero;
13        cout << numero << endl;
14    }
15    cout << "soma = " << soma << endl;
16    return 0;
17 }
```

□

declaração de variáveis (linha 5)

`int numero, soma;` é a declaração de variáveis.
`int a;`
`int a, b, c;`
onde `int` são números inteiros. As variáveis são *posições de memória*.

atribuição (6)

Em `soma = 0;` significa que a variável *soma* está armazenando o valor 0.
soma $\leftarrow 0$
soma $= 0$

entrada e saída (7,8,15)

`cin` é o comando de *entrada.*
`cin >> numero;`
`cin >> numero >> soma;` - requer duas entradas.

`cout` é o comando de *saída.*
```
cout << numero;
cout << numero << endl;
cout << "soma = " << soma << endl;
```

Exemplo 3.3 (Dica) Ao invés de digitar
`12 <enter>`
`17 <enter>`
você pode digitar
`12 17 <enter>`
assim o programa executa os dois números na sequência. Então podemos escrever `12 17 4 -6 8 0 <enter>` que toda a sequência de números será executada.

3.2 laço de repetição

while (9)

while significa *enquanto.*

```
while (condicao)
{
  /*comandos*/
}
```

O *while* é executado enquanto a *condição* for **verdadeira**.

```
    while (numero != 0)
    {
        soma = soma + numero;
        cin >> numero;
        cout << numero << endl;
    }
```

condições

condição	significado
a == b	$a = b$
a != b	$a \neq b$
a < b	$a < b$
a <= b	$a \leqslant b$
a > b	$a > b$
a >= b	$a \geqslant b$
a + b	$a + b$
a - b	$a - b$
a * b	$a.b$
a / b	a/b

comentários

```
/*comentario*/
```

Problema I.2 Dados os números inteiros n e k com $k \geqslant 0$, determinar n^k. Por exemplo, dados os números 3 e 4 o programa deve escrever o número 81.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, k, resultado, potencia;
    cout << "Digite um numero:";
    cin >> n;
    cout << "Digite outro numero:";
    cin >> k;
    resultado = 1;
    potencia = 0;
    while (potencia < k)
    {
        resultado = resultado * n;
        potencia = potencia + 1;
    }
    cout << resultado << endl;
    return 0;
}
```

Outro modo:

```
#include <iostream>
using namespace std;
int main()
{
    int n, k, resultado;
    cout << "Digite um numero:";
    cin >> n;
    cout << "Digite outro numero:";
    cin >> k;
    resultado = 1;
    while (k > 0)
    {
        resultado = resultado * n;
        k = k - 1;
    }
    cout << resultado << endl;
    return 0;
}
```

□

Problema I.3 Dado um número inteiro $n \geqslant 0$, calcular $n!$.

$$n! = \begin{cases} 1 & , \text{se } n = 0 \\ n & , \text{se } n = 1 \\ n(n-1)! & , \text{se } n > 1 \end{cases}$$

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, resultado, i;
    cout << "Digite um numero:";
    cin >> n;
    resultado = 1;
    i = n;
    while (i >= 1)
    {
        resultado = resultado*i;
        i = i - 1;
    }
    cout << resultado << endl;
    return 0;
}
```

Outro modo:

```
#include <iostream>
using namespace std;
int main()
{
    int n, resultado;
    cout << "Digite um numero:";
    cin >> n;
    resultado = 1;
    while (n > 0)
    {
        resultado = resultado*n;
        n = n - 1;
    }
    cout << resultado << endl;
    return 0;
}
```

□

Problema I.4 Dados um número inteiro $n \geqslant 0$, e uma sequência com n inteiros, determinar a soma dos inteiros positivos da sequência. Por exemplo, para a sequência 6 -2 7 0 -5 8 4 o programa deve escrever o número 19.

Solução:

Minha solução: prob04v2.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int n, numero, contador, soma;
    cout << "Digite o comprimento da sequencia: ";
    cin >> n;
    contador = 0;
    soma = 0;
    while (contador < n)
    {
        cout << "Digite o proximo numero: ";
        cin >> numero;
        if (numero > 0)
        {
            soma = soma + numero;
        }
        contador = contador + 1;
    }
    cout << "A soma dos inteiros positivos eh: " << soma <<
        endl;
    return 0;
}
```

Solução do prof.: prob04.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int n, numero, cont, soma;
    cont = 0;
    soma = 0;
    cout << "Digite o comprimento da sequencia: ";
    cin >> n;
    while (cont < n)
    {
        cout << "Digite o proximo numero: ";
        cin >> numero;
        /*
            Neste caso, o while funciona como um
            if desfarcado.
        */
        while (numero > 0)
        {
            soma = soma + numero;
            /* Necessario, senao o while nao sera
```

```
22              testado de novo.*/
23              numero = 0;
24          }
25          cont = cont + 1;
26      }
27      cout << "A soma dos inteiros positivos eh: " << soma <<
            endl;
28      return 0;
29  }
```

□

3.3 if then else

Laço de decisão. *if* significa *se, then* significa *então* e *else* significa *senão.*

```
1  if (condicao)
2  {
3    /*comandos*/
4  }
5  else
6  {
7    /*comandos*/
8  }
```

Problema I.5 Dados um número inteiro $n \geqslant 0$, e uma sequência com n inteiros, determinar a soma dos inteiros positivos e a soma dos inteiros negativos da sequência. Por exemplo, para a sequência 6 -2 7 0 -5 8 4 o programa deve escrever o número 19 e -7.

Solução:

```
1  #include <iostream>;
2  using namespace std;
3  int main()
4  {
5      int n, i, numero, pos, neg;
6      i = 0;
7      pos = 0;
8      neg = 0;
9      cout << "Digite o comprimento da sequencia: ";
10     cin >> n;
11     while (i < n)
12     {
13         cout << "Digite o proximo numero: ";
14         cin >> numero;
15         if (numero > 0)
16         {
17             pos = pos + numero;
18         }
19         else
20         {
21             neg = neg + numero;
```

```
22          }
23          i = i + 1;
24      }
25      cout << "A soma dos inteiros positivos eh: " << pos <<
            endl;
26      cout << "A soma dos inteiros negativos eh: " << neg <<
            endl;
27      return 0;
28  }
```

□

3.4 Operadores

condição	significado
a + b	$a+b$
a - b	$a-b$
a * b	$a.b$
a / b	a/b
a % b	resto de a/b

Problema I.6 Dados um número inteiro $n \geqslant 0$, e uma sequência com n inteiros, determinar quantos números da sequência são pares e quantos são ímpares. Por exemplo, para a sequência 6 -2 7 0 -5 8 4 o programa deve escrever o número 4 para o número de pares 2 para o número de ímpares.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int n, i, numero, par, impar;
6       i = 0;
7       par = 0;
8       impar = 0;
9       cout << "Digite o comprimento da sequencia: ";
10      cin >> n;
11      while (i < n)
12      {
13          cout << "Digite o proximo numero: ";
14          cin >> numero;
15          if (numero % 2 == 0)
16          {
17              par = par + 1;
18          }
19          else
20          {
21              impar = impar + 1;
22          }
23          i = i + 1;
24      }
```

```
25      cout << "A sequencia possui " << par << " inteiros pares
            e " << impar << " inteiros impares." << endl;
26      return 0;
27  }
```

□

3.5 for

Laço de repetição com incremento. Como se fosse um contador.

```
1  for (inicializacao; condicao; atualizacao)
2  {
3    /*comandos*/
4  }
```

Comparando com *while*, temos:

```
1  i = 0;
2  while (i < n)
3  {
4    /*comandos*/
5    i = i + 1;
6  }
```

```
1  for (i = 0; i < n; i = i +
       1)
2  {
3    /*comandos*/
4  }
```

Nota: `i = i + 1` é equivalente a `i++`.

Problema I.7 Dados um número inteiro $n \geqslant 0$, e uma sequência com n inteiros, determinar o maior inteiro da sequência. Por exemplo, para a sequência 6 -2 7 0 -5 8 4 o programa deve escrever o número 8.

Solução:
Minha solução

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n, i, numero, maior;
6      cout << "Digite o comprimento da sequencia: ";
7      cin >> n;
8      cout << "Digite o primeiro numero: ";
9      cin >> maior;
10     i = 1;
11     while (i < n)
12     {
13         cout << "Digite o proximo numero: ";
14         cin >> numero;
15         if (numero > maior)
16         {
17           maior = numero;
18         }
19         i = i + 1;
20     }
21     cout << "O maior numero eh: " << maior << endl;
```

```
22      return 0;
23  }
```

Solução do prof.: prob07.cpp

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int n, i, numero, maior;
6       cout << "Digite o comprimento da sequencia: ";
7       cin >> n;
8       maior = 0;
9       for (i = 0; i < n; i = i + 1)
10      {
11          cout << "Digite o proximo numero: ";
12          cin >> numero;
13          if (numero > maior)
14              maior = numero;
15      }
16      cout << "O maior numero eh: " << maior << endl;
17      return 0;
18  }
```

□

Se você quiser que o usuário digite uma sequência somente com números positivos faça:

```
1   do
2   {
3     cout << "Digite numeros positivos.";
4     cin >> numero;
5   } while (numero < 0)
```

Exemplo 3.4 (extra) Dados um número inteiro $n \geqslant 0$, e um digito d (0 <= d <= 9) determinar quantas vezes d ocorre em n. Por exemplo, para $n = 63453$ e $d = 3$ o programa deve imprimir 2.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int n, d, resto;
6       resto = 0;
7       cout << "Digite um numero: ";
8       cin >> n;
9       cout << "Digite um digito: ";
10      cin >> d;
11      while (n > 0)
12      {
13          if (n % 10 == d)
14          {
```

```
15              resto = resto + 1;
16          }
17          n = n / 10;
18      }
19      cout << "O numero " << d << " aparece "<< resto << "
            vezes."<< endl;
20      return 0;
21  }
```

□

Problema I.8 Dados um número inteiro $n > 0$ e as notas de n alunos, determinar quantos alunos ficaram de recuperação. Um aluno está de recuperação se sua nota estiver entre 30 e 50 (nota máxima 100).

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int n, i, nota, rec;
6       cout << "Digite o numero de alunos: ";
7       cin >> n;
8       rec = 0;
9       for (i = 0; i < n; i = i + 1)
10      {
11          cout << "Digite as notas: ";
12          cin >> nota;
13          if (nota >= 30)
14          {
15              if (nota <= 50)
16              {
17                  rec = rec + 1;
18              }
19          }
20      }
21      cout << rec << " alunos estao de recuperacao." << endl;
22      return 0;
23  }
```

□

Podemos substituir as linhas 13 à 19 por

```
1   if (nota >= 30 && nota <= 50)
2   {
3     rec = rec + 1;
4   }
```

onde && é o operador e lógico.

3.6 Operadores lógicos

&&	e
||	ou
!	não

Exemplo 3.5 `if(!(nota <= 30))` é equivalente a `if(nota > 30)`.

Obs: Note a diferença entre `(a && b) || c` e `a && (b || c)`.

Problema I.9 Dados um número inteiro $n > 0$ e uma sequência com n números inteiros, verificar se a sequência está em ordem crescente.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, numero, anterior, crescente;
    cout << "Digite o comprimento da sequencia: ";
    cin >> n;
    crescente = 1;
    cout << "Digite o primeiro numero: ";
    cin >> anterior;
    for (i = 1; i < n; i = i + 1)
    {
        cout << "Digite o proximo numero: ";
        cin >> numero;
        if (anterior >= numero)
        {
            crescente = 0;
        }
        anterior = numero;
    }
    if (crescente == 1)
    {
        cout << "A sequencia 'e crescente." << endl;
    }
    else
    {
        cout << "A sequencia nao 'e crescente." << endl;
    }
    return 0;
}
```

□

3.7 Indicador de passagem

variável `int`
Com valor 0 ou 1.
0: falso

1: verdadeiro
São variáveis lógicas ou booleanas.

Problema I.10 Dados números inteiros n, i e j, todos maiores que zero, imprimir em ordem crescente os n números naturais que são múltiplos de i ou de j ou ambos. Por exemplo, para $n = 6, i = 2$ e $j = 3$, a saída deverá ser: 0 2 3 4 6 8.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, j, atual, multiplos;
    atual = multiplos = 0;
    cout << "Digite o valor de n: ";
    cin >> n;
    cout << "Digite o valor de i: ";
    cin >> i;
    cout << "Digite o valor de j: ";
    cin >> j;
    while (multiplos < n)
    {
        if (atual % i == 0 || atual % j == 0)
        {
            multiplos = multiplos + 1;
            cout << atual << " ";
        }
        atual = atual + 1;
    }
    cout << endl;
    return 0;
}
```

□

Notações para resumir o código

Podemos atribuir um valor direto na declaração da variável.
`int n, i, j, atual = 0, multiplos = 0;`
Também podemos fazer atribuições múltiplas: `i = j = t = 0;`

Operações

```
i = i + 2;
i += 2;
i *= 2;
i /= 2;
i %= 2;
```
`i++;` soma 1 unidade
`i--;` subtrai 1 unidade

Obs: Seja `int b, a = 3;`

`b = a++;`	`b = ++a;`
`b = 3`	`b = 4`
`a = 4`	`a = 4`

Dica: Como estas notações são confusas é aconselhável que não se use em operações algébricas.

Problema I.11 Dado um número inteiro $n > 0$, verificar se n é primo.

Solução:

n é primo se apenas 1 e n dividem n ou equivalentemente, para todo $m \neq n (m \neq 1)$, $n\%m \neq \emptyset$.

Minha solução: prob11v2.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, primo;
    primo = 0;
    cout << "Digite um numero: ";
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        if (n % i == 0)
        {
            primo = primo + 1;
        }
    }
    if (primo == 2)
    {
        cout << n << " 'e primo." << endl;
    }
    else
    {
        cout << n << " nao 'e primo." << endl;
    }
    cout << endl;
    return 0;
}
```

Solução do prof.: prob11.cpp

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, primo = 1;
    cout << "Digite um numero: ";
    cin >> n;
    if (n == 1)
    {
        primo = 0;
    }
    for (i = 2; i < n; i++)
```

```
13      {
14          if (n % i == 0)
15          {
16              primo = 0;
17          }
18      }
19      if (primo == 1)
20      {
21          cout << n << " 'e primo." << endl;
22      }
23      else
24      {
25          cout << n << " nao 'e primo." << endl;
26      }
27      return 0;
28  }
```

□

Declaração de constantes

Para definir nomes para valores constantes use
`#define <nome> <valor>`

Exemplo 3.6 Dois exemplos
```
#define TRUE 1
#define FALSE 0
```

3.8 Precedência de operadores

prioridade	operador
1	`++, --, -` sinal
2	`*, /, %`
3	`+, -`
4	`<, <=, ==, !=, =, +=, -=, *=, /=, %=`

prioridade	operador
1	`!`
2	`&&`
3	`||`

Problema I.12 Dado um número inteiro $n > 1$, imprimir sua decomposição em fatores primos, indicando também a multiplicidade de cada fator. Por exemplo, para $n = 600$, a saída deverá ser:
fator 2 multiplicidade 3
fator 3 multiplicidade 1
fator 5 multiplicidade 2
Note que $600 = 2^3.3^1.5^2$.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, t, multiplicidade;
    cout << "Digite um numero: ";
    cin >> n;
    t = n;
    for (i = 2; i <= n; i++)
    {
        multiplicidade = 0;
        while (t % i ==0)
        {
            t /= i;
            multiplicidade ++;
        }
        if (multiplicidade > 0)
            cout << "Fator " << i << " multiplicidade " <<
                multiplicidade << endl;
    }
    return 0;
}
```

□

Problema I.13 Dados um inteiro $n > 0$, e uma sequência com n números inteiros maiores do que zero, determinar o máximo divisor comum entre eles. Por exemplo, para a sequência 3 42 30 105 o seu programa deve escrever o número 3.

Solução:

```
#include <iostream>
using namespace std;
int main ()
{
    int n, i, numero, mdc, candidato;
    cout << "Digite o comprimento da sequencia: ";
    cin >> n;
    cin >> mdc;
    for (i = 1; i < n; i++)
    {
        cin >> numero;
        candidato = mdc;
        while (mdc % candidato != 0 || numero % candidato !=
             0)
            candidato --;
        mdc = candidato;
    }
    cout << mdc << endl;
    return 0;
}
```

□

Exemplo 3.7 (extra) Dado um inteiro $n > 0$, e uma sequência com n números inteiros maiores que zero, determinar o fatorial de cada número da sequência.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, j, numero, fat;
    cout << "Digite o comprimento da sequencia: ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        cin >> numero;
        fat = 1;
        for (j = 2; j <= numero; j++)
            fat *= j;
        cout << numero << "!= " << fat << endl;
    }
    return 0;
}
```

□

Problema I.14 Dados um inteiro $n > 0$, e uma sequência com n inteiros, verificar se a sequência é uma progressão aritmética.

Solução:

```
#include <iostream>
#define TRUE 1
#define FALSE 0

using namespace std;
int main()
{
    int n, i, anterior, atual, razao = 0, res = TRUE;
    cout << "Digite o comprimento da sequencia: ";
    cin >> n;
    cin >> anterior;
    if (n > 1)
    {
        cin >> atual;
        razao = atual - anterior;
    }
    for (i = 2; i < n; i++)
    {
        anterior = atual;
        cin >> atual;
        if (atual - anterior != razao)
```

```
22              res = FALSE;
23      }
24      if (res) /*ou res = TRUE*/
25          cout << "'E uma P.A." << endl;
26      else
27          cout << "Nao 'e uma P.A." << endl;
28      return 0;
29  }
```

□

Problema I.15 Sabe-se que cada número da forma n^3 é igual a soma de n ímpares consecutivos. Exemplos:

$$1^3 = 1$$
$$2^3 = 3+5$$
$$3^3 = 7+9+11$$
$$4^3 = 13+15+17+19$$

Dado um inteiro $m > 0$, determinar os ímpares consecutivos cuja soma é igual a n^3, para n assumindo valores de 1 a n.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int m, n, i, candidato, soma, termo;
6       cout << "Digite um numero: ";
7       cin >> m;
8       for (n = 1; n <= m; n++)
9       {
10          candidato = 1;
11          do
12          {
13              soma = 0;
14              termo = candidato;
15              for (i = 0; i < n; i++)
16              {
17                  soma += termo;
18                  termo += 2;
19              }
20              if (soma != n*n*n)
21                  candidato += 2;
22          } while (soma != n*n*n);
23          cout << n << "^3 = ";
24          termo = candidato;
25          for (i = 0; i < n - 1; i++)
26          {
27              cout << termo << " + ";
28              termo += 2;
29          }
```

```
30          cout << termo << endl;
31      }
32      return 0;
33  }
```

Uma segunda solução seria:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int m, n, i, candidato, soma, termo;
6       cout << "Digite um numero: ";
7       cin >> m;
8       for (n = 1; n <= m; n++)
9       {
10          candidato = 1;
11          do
12          {
13              soma = 0;
14              termo = candidato;
15              soma = n * candidato + n * (n - 1);
16              if (soma != n*n*n)
17                  candidato += 2;
18          } while (soma != n*n*n);
19          cout << n << "^3 = ";
20          termo = candidato;
21          for (i = 0; i < n - 1; i++)
22          {
23              cout << termo << " + ";
24              termo += 2;
25          }
26          cout << termo << endl;
27      }
28      return 0;
29  }
```

Terceira solução: resolvendo a equação $n^3 = n * candidato + n(n-1)$ obtemos $candidato = n^2 - n + 1$;

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int m, n, i, candidato, termo;
6       cout << "Digite um numero: ";
7       cin >> m;
8       for (n = 1; n <= m; n++)
9       {
10          candidato = n*n - n + 1;
11          cout << n << "^3 = ";
12          termo = candidato;
13          for (i = 0; i < n - 1; i++)
14          {
15              cout << termo << " + ";
16              termo += 2;
```

```
17          }
18          cout << termo << endl;
19      }
20      return 0;
21  }
```

□

Parte II

Números Reais

4 Números Reais

4.1 Lendo dados num arquivo externo

Podemos criar um arquivo externo, inserir os dados e fazer com que o *terminal* leia os dados.

Para isso crie um *arquivo*, insira os dados nele separado com espaço e/ou quebra de linha.

Ao executar o binário, digite, por exemplo `./recebedados < arquivo`

4.2 Float

A variável `float` representa os números reais.

Exemplo 4.1 `float a = 1.53;`

```
cin >> a;
cout << a;
```

Problema II.1 Dadas $n > 0$ notas de prova, calcular a média aritmética das notas. As notas podem ser fracionárias.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i;
    float nota, soma = 0.0;
    cout << "Digite o numero de notas: ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        cout << "Digite as notas: ";
        cin >> nota;
        soma += nota;
    }
    cout << soma / n << endl;
    return 0;
}
```

□

Problema II.2 Dado um inteiro $n > 0$, determinar o número harmônico H_n dado por $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$.

Solução:

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

```
{
    int n, i;
    float soma = 0.0;
    cout << "Digite um numero: ";
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        soma += 1.0/i;
    }
    cout << soma << endl;
    return 0;
}
```

□

4.3 Casting

Converte um tipo de dados para outro.

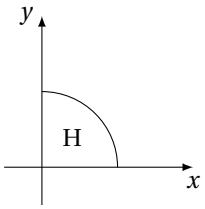
Exemplo 4.2 Transformando a em *int*: `(int) a`

Se a for *float*, então ele trunca o número (arredonda para baixo).

Exemplo 4.3 Considere dois inteiros e um float.

```
int a, b;
float c;
c = (float) a/b;
```

Problema II.3 Os pontos (x, y) que pertencem à figura H (abaixo) são tais que $x \geqslant 0, y \geqslant 0$ e $x^2 + y^2 \leqslant 1$.



Dados n pontos reais (x, y), verificar se cada ponto pertence ou não a H.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, i;
    float x, y;
    cout << "Digite um numero: ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        cin >> x >> y;
```

```
12          if (x >= 0 && y >= 0 && x*x + y*y <= 1)
13              cout << "Pertence a H." << endl;
14          else
15              cout << "Nao pertence a H." << endl;
16      }
17      return 0;
18  }
```

□

Exemplo 4.4 (extra) Dado um inteiro $n \geqslant 0$, calcular o valor da soma $S_n = \frac{1}{n} + \frac{2}{n-1} + \frac{3}{n-2} + \ldots + \frac{n}{1}$.

Solução:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n, i, k;
6      float sn = 0.0;
7      cout << "Digite um numero: ";
8      cin >> n;
9      k = n;
10     for (i = 1; i <= n; i++)
11     {
12         sn = sn + (float)i / k;
13         k--;
14     }
15     cout << sn << endl;
16     return 0;
17 }
```

□

Exemplo 4.5 (Fibonacci) Dado um inteiro $n \geqslant 0$, determinar a sequencia de Fibonacci até n.

Solução:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n, i, aa = 0, a = 1, F;
6      cout << "Digite o numero de termos da sequencia de
           Fibonacci: ";
7      cin >> n;
8      cout << aa << " " << a << " ";
9      for (i = 0; i < n - 2; i++)
10     {
11         F = a + aa;
12         cout << F << " ";
```

```
13          aa = a;
14          a = F;
15      }
16      cout << endl;
17      return 0;
18  }
```

□

Problema II.4 Dados números reais $x \geqslant 0$ e $\varepsilon > 0$, calcular uma aproximação da raiz quadrada de x através da seguinte sequencia:

$$r_0 = x$$
$$r_{n+1} = \frac{1}{2}\left(r_n + \frac{x}{r_n}\right)$$

A aproximação será o primeiro valor r_{n+1} tal que $|r_{n+1} - r_n| < \varepsilon$.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       float x, epsilon, atual, anterior;
6       cout << "Digite um numero: ";
7       cin >> x;
8       cout << "Digite um epsilon: ";
9       cin >> epsilon;
10      atual = x;
11      do
12      {
13          anterior = atual;
14          atual = 0.5 * (anterior + x / anterior);
15      } while (anterior - atual >= epsilon || atual - anterior
            >= epsilon);
16      cout << atual << endl;
17      return 0;
18  }
```

Uma segunda solução seria:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       float x, epsilon, atual, anterior, diferenca;
6       cout << "Digite um numero: ";
7       cin >> x;
8       cout << "Digite um epsilon: ";
9       cin >> epsilon;
10      atual = x;
11      do
12      {
13          anterior = atual;
```

```
14          atual = 0.5 * (anterior + x / anterior);
15          diferenca = atual - anterior;
16          if(diferenca < 0)
17              diferenca = -diferenca; /*pega o modulo de
                    diferenca*/
18      }while(diferenca >= epsilon);
19      cout << atual << endl;
20      return 0;
21  }
```

□

Problema II.5 Dados números reais x e $\varepsilon > 0$, calcular uma aproximação de e^x através da seguinte série infinita:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots$$

Inclua na aproximação todos os termos até o primeiro valor absoluto (módulo) menor do que ε.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int n = 0;
6       float x, epsilon, soma = 1.0, termo = 1.0;
7       cout << "Digite um numero: ";
8       cin >> x;
9       cout << "Digite um epsilon: ";
10      cin >> epsilon;
11      while ((termo > 0 && termo >= epsilon) || (termo < 0 &&
            -termo >= epsilon))
12      {
13          n++;
14          termo *= x / n;
15          soma += termo;
16      }
17      cout << soma << endl;
18      return 0;
19  }
```

Note que

$$\frac{x^2}{2!} = \frac{x^1}{1!} \cdot \frac{x}{2}$$
$$\frac{x^3}{3!} = \frac{x^2}{2!} \cdot \frac{x}{3}$$

□

4.4 Caracteres

`char a;`
`a = 'c';`
Maiúsculas e minúsculas são diferentes. `'a' != 'A'`.
`cin >> a;`
Lê um `char` exceto espaços e/ou quebra de linha.
`cin >> noskipws >> a;` considera espaço
`'.', '!', '?', ' ', '1', 'n', '\n', '\t`
onde `'\n'` é a quebra de linha e `'\t'` é a tabulação.

Problema II.6 Dada uma sequência de caracteres terminada por um ponto `'.'` representando um texto, determinar a frequência relativa de vogais no texto. Por exemplo, no "Em terra de cego quem tem um olho e caolho.", essa frequência é 0.380952(16/42).

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int vogais = 0, tamanho = 0;
    char c;
    cin >> noskipws >> c;
    while (c != '.')
    {
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' ||
            c == 'u' ||
            c == 'A' || c == 'E' || c == 'I' || c == 'O' ||
                c == 'U')
            vogais ++;
        tamanho ++;
        cin >> noskipws >> c;
    }
    cout << (float) vogais / tamanho << endl;
    return 0;
}
```

□

Problema II.7 Dada uma frase terminada por `'.'`, imprimir o comprimento da palavra mais longa.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int maior = 0, atual = 0;
    char c;
```

```
7       cin >> noskipws >> c;
8       while (c != '.')
9       {
10          atual ++;
11          if (c == ' ')
12              atual = 0;
13          if (atual > maior)
14              maior = atual;
15          cin >> noskipws >> c;
16      }
17      cout << maior << endl;
18      return 0;
19  }
```

Segunda solução: prob207v2.cpp

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int maior = 0, atual = 0;
6       char c;
7       cin >> noskipws >> c;
8       while (c != '.')
9       {
10          atual = 0;
11          while (c != ' ' && c != '.')
12          {
13              atual ++;
14              cin >> noskipws >> c;
15          }
16          if (atual > maior)
17              maior = atual;
18          if (c != '.')
19              cin >> noskipws >> c;
20      }
21      cout << maior << endl;
22      return 0;
23  }
```

□

Problema II.8 Dada uma sequência de caracteres terminada por '.', determinar quantas letras minúsculas e maiúsculas aparecem na sequência.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int ma = 0, mi = 0;
6       char a;
7       cin >> noskipws >> a;
8       while (a != '.')
```

```
    {
        if (a >= 'a' && a <= 'z')
            mi ++;
        if (a >= 'A' && a <= 'Z')
            ma ++;
        cin >> noskipws >> a;
    }
    cout << "A frase tem "<< ma << " letras maiusculas." <<
        endl;
    cout << "e "<< mi << " letras minusculas." << endl;
    return 0;
}
```

□

4.5 Funções

Funções são trechos de código que fazem alguma coisa.
`void f(int x, int y)`

```
int funcao (int a, float b, char c)
{
  /* comandos */
  return <valor>
}
```

```
d = funcao(2,3.5,'b');
```

Problema II.9 Dados dois números reais x e y e dois números inteiros positivos a e b, calcular o valor de $x^a + y^b + (x-y)^{a+b}$ usando a função
`float potencia (float base, int expoente);`

Solução:

```
#include <iostream>
using namespace std;

float potencia (float base, int expoente)
{
    int i;
    float mult = 1;
    for (i = 0; i < expoente; i++)
        mult *= base;
    return mult;
}

/* Se a funcao estiver depois do int main */
/* podemos chama-la com o comando a seguir: */
/* float potencia (...); */
int main()
{
    float x, y;
    int a, b;
    cin >> x >> y >> a >> b;
```

```
21      cout << potencia(x,a) + potencia(y,b) + potencia(x-y,a+b
            ) << endl;
22      return 0;
23  }
```

□

Problema II.10

- Escreva uma função que recebe n e devolve $n!$;
- Faça uma função que recebe dois inteiros m e n e usando a função anterior calcula

$$\binom{m}{n} = \frac{m!}{n!(m-n)!}$$

- Faça um programa que lê um inteiro $n > 0$ e imprime os coeficientes da expansão de $(a+b)^n$. Lembrando que $(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}$.

Solução:

```
1   #include <iostream>
2   using namespace std;
3
4   int fatorial (int n)
5   {
6       int mult = 1, i;
7       for (i = 1; i <= n; i++)
8           mult *= i;
9       return mult;
10  }
11
12  int combinacao (int m, int n)
13  {
14      return fatorial(m) / (fatorial(n) * fatorial(m - n));
15  }
16
17  int main()
18  {
19      float n, i;
20      cin >> n;
21      for (i = 0; i <= n; i++)
22          cout << combinacao(n,i) << " ";
23      cout << endl;
24      return 0;
25  }
```

□

4.6 Alterando o valor de uma variável

Quando chamamos uma função $f(a)$, mesmo que o valor da variável que recebe a dentro de f seja alterado, a não é alterado na função onde f foi chamada.

Algumas vezes queremos alterar o valor da variável, para isso usamos *ponteiros*.
OBS: A função `void` é uma função que não retorna nenhum valor.

Exemplo 4.6 Seja

```
#include <iostream>
using namespace std;

void troca1 (int a, int b)
{
    int c;
    c = a;
    a = b;
    b = c;
}

void troca2 (int *a, int *b)
{
    int c = *a;
    *a = *b;
    *b = c;
}

int main ()
{
    int a = 2, b = 3;
    troca1(a,b);
    cout << a << " " << b << endl;
    troca2(&a,&b);
    cout << a << " " << b << endl;
    return 0;
}
```

Problema II.11 Um número a é dito ser permutação de um outro número b se os digitos de a formam uma permutação dos digitos de b. Exemplo, 5412434 é uma permuta de 4321445 mas não é uma permuta de 4312455. Obs: Considere que o digito 0 não aparece nos números.

- Faça uma função *contadigitos* que dados um inteiro n e um inteiro d, $0 \leqslant d \leqslant 9$, devolve quantas vezes d aparece em n.
- Usando a função do item anterior faça um programa que lê dois números a e b e responda se a é permutação de b.

Solução:

```
#include <iostream>
using namespace std;

int contadigitos (int n, int d)
{
    int quant = 0;
    while (n > 0)
```

```
8       {
9           if (n % 10 == d)
10              quant ++;
11          n /= 10;
12      }
13      return quant;
14  }
15
16  int main ()
17  {
18      int a, b, i, perm = 1;
19      cout << "Entre com a e b: ";
20      cin >> a >> b;
21      for (i = 1; i <= 9; i++)
22      {
23          if (contadigitos(a,i) != contadigitos(b,i))
24              perm = 0;
25      }
26      if (perm)
27          cout << "'E permuta." << endl;
28      else
29          cout << "Nao 'e permuta." << endl;
30      return 0;
31  }
```

□

Problema II.12 Dizemos que um inteiro positivo n é *perfeito* se for igual à soma de seus divisores positivos diferentes de n. Exemplo: 6 é perfeito, pois $1+2+3=6$. Faça um programa que verifica se um dado número inteiro positivo é perfeito.

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int n, i, perfeito = 0;
6       cout << "Digite um numero: " << endl;
7       cin >> n;
8       for (i = 1; i < n; i++)
9       {
10          if (n % i == 0)
11              perfeito += i;
12      }
13      if (perfeito == n)
14          cout << "'E perfeito." << endl;
15      else
16          cout << "Nao 'e perfeito." << endl;
17      return 0;
18  }
```

□

Problema II.13 Dizemos que um número natural n é palíndrome se lido da direita para a esquerda ou da esquerda para a direita é o mesmo número. Exemplo: 567765 é palíndrome e 567675 não é palíndrome.

Escreva uma função que recebe um inteiro $n > 0$ e devolve o seu primeiro digito, seu último digito e altera o valor de n removendo seu primeiro e último digitos.

Exemplo:

valor inicial de n	primeiro digito	último digito	valor final de n
14738	1	8	473
78	7	8	0
7	7	7	0

Solução:

```
#include <iostream>
#define TRUE 1
#define FALSE 0
using namespace std;

void encurta (int *n, int *primeiro, int *ultimo)
{
    int copia = *n, pot = 1;
    *ultimo = copia % 10;
    while (copia >= 10)
    {
        copia /= 10;
        pot *= 10;
    }
    *primeiro = copia;
    *n %= pot; /* ou *n = *n - *primeiro * pot; */
    *n /= 10;
}

int main ()
{
    int n, primeiro, ultimo, palindrome = TRUE;
    cout << "Digite um numero: " << endl;
    cin >> n;
    while (n > 0)
    {
        encurta (&n, &primeiro, &ultimo);
        if (primeiro != ultimo)
            palindrome = FALSE;
    }
    if (palindrome)
        cout << "'E palindrome." << endl;
    else
        cout << "Nao 'e palindrome." << endl;
    return 0;
}
```

□

Problema II.14 Escreva uma função com protótipo
`int divide (int *m, int *n, ind d)` que recebe três inteiros positivos como parâmetro e retorna 1 se d divide *pelo menos um* entre `*m` e `*n` e 0, caso contrário. Fora isso, se d divide `*m` então `*m` é dividido (e o resultado é armazenado em `*m`). Faça a mesma coisa para `*n`.

Escreva um programa que lê dois inteiros positivos m e n e calcula, usando a função acima, o *mínimo múltiplo comum* entre m e n.

Exemplo: $3080 = 2^3.5.7.11$ e $1092 = 2^2.3.7.13$ resultado $= 2^3.3.5.7.11.13$

Solução:

```
#include <iostream>
#define TRUE 1
#define FALSE 0
using namespace std;

int divide (int *m, int *n, int d)
{
    int ok = 0;
    if (*m % d == 0)
    {
        ok = 1;
        *m /= d;
    }
    if (*n % d == 0)
    {
        ok = 1;
        *n /= d;
    }
    return ok;
}

int main ()
{
    int m, n, d, mmc;
    cout << "Digite dois numeros: " << endl;
    cin >> m >> n;
    d = 2;
    mmc = 1;
    while (m != 1 || n != 1)
    {
        if (divide (&m, &n, d))
            mmc *= d;
        else
            d++;
    }
    cout << mmc << endl;
    return 0;
}
```

□

Problema II.15 Escreva uma função com protótipo
`void converte (char ch, int *tipo, char *valor)` que recebe um caractere `ch` e devolve em `*tipo`:

0 se o caractere for um número,
1 se o caractere for uma letra e
2, caso contrário.

Além disso, no caso de ser uma letra, converte para maiúscula e coloca em `*valor` e se não for uma letra coloca `ch` inalterado em `*valor`. Escreva um programa que lê uma frase terminada por `'.'`, e imprime apenas os números e letras da frase. Converta as letras para maiúscula antes de imprimir.

Solução:

```
#include <iostream>
using namespace std;

void converte (char ch, int *tipo, char *valor)
{
    *valor = ch;
    if (ch >= '0' && ch <= '9')
        *tipo = 0;
    else if (ch >= 'a' && ch <= 'z')
    {
        *tipo = 1;
        *valor = ch - 'a' + 'A';
    }
    else if (ch >= 'A' && ch <= 'Z')
        *tipo = 1;
    else
        *tipo = 2;
}

int main ()
{
    char c, valor;
    int tipo;
    do
    {
        cin >> noskipws >> c;
        converte (c, &tipo, &valor);
        if (tipo != 2)
            cout << valor;
    } while (c != '.');
    return 0;
}
```

□

Problema II.16 Escreva uma função que decomponha um número em sua parte inteira e sua parte fracionária. A entrada da função é um número real x e a saída é um número inteiro e um número real com os valores da parte inteira e da parte fracionária de x.

Solução:

```
#include <iostream>
using namespace std;

void decompoe (float x, int *xint, float *xfrac)
{
    *xint = (int) x;
    *xfrac = x - *xint;
}

int main ()
{
    float b, x;
    int a;
    cout << "Digite um numero: " << endl;
    cin >> x;
    decompoe(x, &a, &b);
    cout << a << endl;
    cout << b << endl;
    return 0;
}
```

□

Problema II.17 Escreva uma função que transforma um intervalo de tempo dado em segundos para dias, horas, minutos e segundos. A entrada da função é um inteiro t com o valor de um intervalo de tempo em segundos. A saída da função são 4 inteiros com os valores de dias, horas, minutos e segundos, correspondentes ao valor de t.

Exemplo:
para $t = 100\,000$ a saída será 1 (dia), 3 (horas), 46 (minutos) e 40 (segundos).

Solução:

```
#include <iostream>
using namespace std;

void converte (int t, int *dias, int *horas, int *minutos,
    int *segundos)
{
    *segundos = t % 60;
    t /= 60;
    *minutos = t % 60;
    t /= 60;
    *horas = t % 24;
    t /= 24;
    *dias = t;
```

```
13 }
14
15 int main ()
16 {
17     int t, dias, horas, minutos, segundos;
18     cout << "Digite um intervalo de tempo em segundos: " <<
           endl;
19     cin >> t;
20     converte (t, &dias, &horas, &minutos, &segundos);
21     cout << dias << " dia(s), " << horas << " hora(s), " <<
           minutos << " minuto(s), " << segundos << " segundo(s
           )."<< endl;
22     return 0;
23 }
```

□

Parte III

Vetores

5 Vetores

Um *vetor* é um tipo especial de variável que armazena muitas informações de um mesmo tipo. Por exemplo, `int v[10];` significa que `v[0], v[1], ... v[9]` são "variáveis" inteiras.

0	1	2	3	4	5	6	7	8	9

Para declarar um vetor podemos usar apenas números (nunca variáveis). É útil usar constante (`#define`).

```
int v[MAX];
```

Exemplo 5.1 Considere

```
1  int v[10];
2  v[7] = 153;
3  v[2] = 7;
4  cout << v[2] + v[7] << endl;
```

Só podemos atribuir um valor por vez no vetor, então podemos escrever com um laço.

Exemplo 5.2 Considere

```
1  int n, v[MAX], i;
2  cin >> n;
3  for (i = 0; i <= n; i++)
4      cin >> v[i];
```

Obs: Para "zerar" tudo faça: `v[i] = 0;`

Problema III.1 Dados $n > 0$ números inteiros, imprimi-los na ordem inversa de leitura.

Solução:

```
1  #include <iostream>
2  #define MAX 1000
3  using namespace std;
4  int main()
5  {
6      int n, i, v[MAX];
7      cout << "Digite um numero: ";
8      cin >> n;
9      for (i = 0; i < n; i++)
10         cin >> v[i];
11     for (i = n - 1; i >= 0; i--)
12         cout << v[i] << " ";
13     cout << endl;
14     return 0;
15 }
```

□

Problema III.2 Dados $n > 0$ lançamentos de uma roleta (números entre 0 e 36), calcular a frequência de cada número.

Solução:

```
#include <iostream>
#define MAX 37
using namespace std;
int main()
{
    int n, i, numero, histograma[MAX];
    cout << "Digite um numero: ";
    cin >> n;
    for (i = 0; i < MAX; i++)
        histograma[i] = 0;
    for (i = 0; i < n; i++)
    {
        cin >> numero;
        histograma[numero] ++;
    }
    for (i = 0; i < MAX; i++)
        cout << "Frequencia de " << i << " = " << histograma
            [i] << endl;
    return 0;
}
```

□

Problema III.3 Dados dois números naturais m e n e duas sequências ordenadas. Com $m > 0$ e $n > 0$ números inteiros, obter uma única sequência ordenada contendo todos os elementos das sequências originais sem repetição.

Solução:

```
#include <iostream>
#define MAX 1000
using namespace std;
int main()
{
    int m, n, i, j, vetorm[MAX], vetorn[MAX];
    cout << "Digite dois numeros: ";
    cin >> m >> n;
    for (i = 0; i < m; i++)
        cin >> vetorm[i];
    for (i = 0; i < n; i++)
        cin >> vetorn[i];
    i = j = 0;
    while (m > i && n > j)
    {
        if (vetorm[i] < vetorn[j])
        {
            cout << vetorm[i] << " ";
            i++;
```

```
20          }
21          else if (vetorm[i] == vetorn[j])
22          {
23              cout << vetorm[i] << " ";
24              i++;
25              j++;
26          }
27          else
28          {
29              cout << vetorn[j] << " ";
30              j++;
31          }
32      }
33      while (i < m)
34      {
35          cout << vetorm[i] << " ";
36          i++;
37      }
38      while (j < n)
39      {
40          cout << vetorn[j] << " ";
41          j++;
42      }
43      cout << endl;
44      return 0;
45  }
```

□

Problema III.4 a) Escreva uma função com protótipo
`int acha(float v[MAX], int n, float x)` que devolve a posição em que o número real x ocorre no vetor v ou devolve -1 se x não aparece no vetor. O número de elementos do vetor é n.

b) Escreva uma função com protótipo
`void insere(float v[MAX], inr *n, float x)` que insere x na última posição do vetor v e altera o valor de $*n$.

c) Dada uma sequência de n números reais, imprimi-la eliminando as repetições.

Solução:

```
1   #include <iostream>
2   #define MAX 100
3   using namespace std;
4
5   int acha(float v[MAX], int n, float x)
6   {
7       int i;
8       for (i = 0; i < n; i++)
9       {
10          if (v[i] == x)
11              return i;
12      }
```

```
13      return -1;
14  }
15
16  void insere(float v[MAX], int *n, float x)
17  {
18      v[*n] = x;
19      (*n)++;
20  }
21
22  int main()
23  {
24      int n, i, t = 0;
25      float atual, seq[MAX];
26      cout << "Digite um numero: ";
27      cin >> n;
28      for (i = 0; i < n; i++)
29      {
30          cin >> atual;
31          if (acha(seq,t,atual) == -1)
32              insere(seq,&t,atual);
33      }
34      for (i = 0; i < t; i++)
35          cout << seq[i] << " ";
36      cout << endl;
37      return 0;
38  }
```

□

Problema III.5 a) Escreva uma função que recebe um inteiro positivo k e um vetor com os $k+1$ coeficientes reais de um polinômio de grau k e outro real x e devolve o valor do polinômio no ponto x. Considere que o coeficiente de x^i está guardado na posição i do vetor.

Um polinômio $p(x)$ de grau k é dado por

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_k x^k$$

b) Escreva uma função que recebe como parâmetro um inteiro não-negativo k e um vetor p com os $k+1$ coeficientes reais de um polinômio de grau k e altera k e p de modo que eles passem a armazenar o grau e o coeficiente da *derivada* do polinômio p.

A derivada de um polinômio $p(x)$ de grau k é dada por

$$p'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \ldots + k a_k x^{k-1}$$

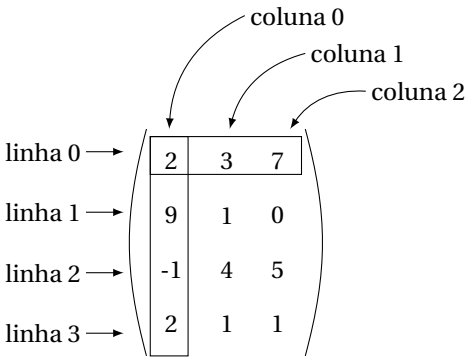
c) Escreva um programa que leia um inteiro não-negativo k e os $k+1$ coeficientes reais de um polinômio p_0 de grau k e calcula $p_0(x_0), p_1(x_1), p_2(x_2)$ onde p_1 e p_2 são respectivamente a primeira e a segunda derivada do polinômio p_0.

Solução:

```
#include <iostream>
#define MAX 1000
using namespace std;

float calcula_poli(int k, float coef[MAX], float x)
{
    float pot = 1, soma = 0;
    int i;
    for (i = 0; i <= k; i++)
    {
        soma += coef[i] * pot;
        pot *= x;
    }
    return soma;
}

void derivada(int *k, float coef[MAX])
{
    int i;
    for (i = 0; i < *k; i++)
        coef[i] = (i + 1) * coef[i + 1];
    (*k)--;
}

int main()
{
    int k, i;
    float coef[MAX], x0, x1, x2;
    cout << "Digite o grau do polinomio: ";
    cin >> k;
    cout << "Digite os coeficientes: ";
    for (i = 0; i <= k; i++)
        cin >> coef[i];
    cout << "Digite os valores de x: ";
    cin >> x0 >> x1 >> x2;
    cout << calcula_poli(k,coef,x0) << endl;
    derivada(&k,coef);
    cout << calcula_poli(k,coef,x1) << endl;
    derivada(&k,coef);
    cout << calcula_poli(k,coef,x2) << endl;
    return 0;
}
```

□

6 Matriz



Problema III.6 Dado n e uma matriz real $A_{n\times n}$ verificar se A é simétrica.

Solução:

```
#include <iostream>
#define MAX 100
#define TRUE 1
#define FALSE 0
using namespace std;

int main()
{
    float A[MAX][MAX];
    int n, i, j, simetrica = TRUE;
    cout << "Digite a ordem da matriz quadrada: ";
    cin >> n;
    cout << "Digite as entradas da matriz: ";
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            cin >> A[i][j];
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (A[i][j] != A[j][i])
                simetrica = FALSE;
        }
    }
    if (simetrica)
        cout << "Simetrica." << endl;
    else
        cout << "Nao simetrica." << endl;
    return 0;
}
```

□

Problema III.7 Imprimir as n primeiras linhas do triângulo de Pascal.

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

Solução:

```
#include <iostream>
#define MAX 100
using namespace std;
int main()
{
    int A[MAX][MAX], n, linha, coluna;
    cout << "Digite um numero: ";
    cin >> n;
    A[0][0] = A[0][1]=1;
    for (linha = 1; linha < n; linha ++)
    {
        for (coluna = 0; coluna <= linha + 1; coluna ++)
        {
            if (coluna == 0)
                A[linha][coluna] = A[linha - 1][coluna];
            else if (coluna == linha + 1)
                A[linha][coluna] = A[linha - 1][coluna - 1];
            else
                A[linha][coluna] = A[linha - 1][coluna] + A[
                    linha - 1][coluna - 1];
        }
    }
    for (linha = 0; linha < n; linha ++)
    {
        for (coluna = 0; coluna <= linha + 1; coluna ++)
            cout << A[linha][coluna] << " ";
        cout << endl;
    }
    return 0;
}
```

□

Problema III.8 Escreva um programa que, dadas duas matrizes $A_{m\times n}$ e $B_{n\times p}$, calcula a matriz $C_{m\times p}$ que é o produto de A por B.

Solução:

□

Problema III.9 a) Escreva uma função que recebe como parâmetros uma matriz inteira $A_{n\times m}$ e uma posição (i, j) da matriz, e conta quantas posições ao redor da posição (i, j) contém o valor -1.

b) Escreva um programa que lê uma matriz $A_{n\times m}$ de 0's (posições livres) e -1's (minas). Utilizando a função do item anterior, o programa deve computar e imprimir a quantidade de minas ao redor de cada posição livre da matriz.

Solução:

```
#include <iostream>
#define MAX 100
using namespace std;

int conta(int A[MAX][MAX], int n, int m, int i, int j)
{
    int soma = 0, l, k;
    for (l = i - 1; l <= i + 1;l++)
    {
        for (k = j - 1; k <= j + 1; k++)
            if (!(l == i && k == j) && A[l][k] == -1)
                soma ++;
    }
    return soma;
}

int main()
{
    int m, n, i, j, A[MAX][MAX];
    cout << "Digite dois numeros: ";
    cin >> n >> m;
    // moldura da matriz
    for (i = 0; i <= n + 1; i++)
        A[i][0] = A[i][m+1] = 0;
    for (j = 0; j <= m + 1; j++)
        A[0][j] = A[m+1][j]=0;
    // lendo a matriz
    for (i = 1; i <= n; i++)
        for (j = 1; j <= m; j++)
            cin >> A[i][j];
    cout << endl;
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= m; j++)
            if (A[i][j] != -1)
                cout << conta(A,n,m,i,j) << " ";
            else
                cout << -1 << " ";
            cout << endl;
    }
    return 0;
}
```

□

Problema III.10 a) Escreva uma função que recebe como parâmetro uma matriz real $A_{n\times m}$ e uma posição (i, j) da matriz e calcula a média aritmética dos vizinhos de (i, j), ou seja, a média entre $A[i-1][j]$, $A[i+1][j]$, $A[i][j-1]$ e $A[i][j+1]$.

Desconsidere os vizinhos que não pertencem a matriz. Por exemplo, os vizinhos de $(0,0)$ são somente $(0,1)$ e $(1,0)$.

b) Escreva uma função que recebe como parâmetro uma matriz real $A_{n\times m}$ e devolve uma matriz $A_{\text{média}}$, onde $A_{\text{média}}[i][j]$ é a média aritmética dos vizinhos de (i,j). Para isso, utilize a função do item anterior.

c) Escreva um programa que lê uma matriz real $A_{n\times m}$ e um número inteiro k; utilizando a função do item anterior, o programa deve transformar a matriz k vezes, imprimindo a matriz inicial e depois de cada transformação.

Solução:

```
#include <iostream>
#define MAX 100
using namespace std;

float media(float A[MAX][MAX], itn n, int m, int i, int j)
{
  float soma = 0;
  int qtde = 0;
  if(i - 1 >= 0)
  {
    soma += A[i-1][j];
    qtde ++;
  }
  if(j - 1 >= 0)
  {
    soma += A[i][j-1];
    qtde ++;
  }
  if(i + 1 < n)
  {
    soma += A[i+1][j];
    qtde ++;
  }
  if(j + 1 < m)
  {
    soma += A[i][j+1];
    qtde ++;
  }
  return soma/qtde;
}

void matriz_media(float A[MAX][MAX], int n, int m, float B[
    MAX][MAX])
{
  int i, j;
  for(i = 0; i < n; i++)
    for(j = 0; j < m; j++)
      B[i][j] = media(A,n,m,i,j);
}

void imprime(float A[MAX][MAX], int n, int m)
```

```
41 {
42   int i,j;
43   for(i = 0; i < n; i++)
44   {
45     for(j = 0; j < m; j++)
46       cout << A[i][j] << " ";
47     cout << endl;
48   }
49 }
50
51 int main()
52 {
53   int n, m, i, j, k, r;
54   float A[MAX][MAX], B[MAX][MAX];
55   cin >> n >> m;
56   for(i = 0; i < n; i++)
57     for(j = 0; j < m; j++)
58       cin >> A[i][j];
59   imprime(A,n,m);
60   for(r = 0; r < k; r++)
61   {
62     if(r % 2 == 0)
63     {
64       matriz_media(A,n,m,B);
65       imprime(B,n,m);
66     }
67     else
68     {
69       matriz_media(B,n,m,A);
70       imprime(A,n,m);
71     }
72   }
73   return 0;
74 }
```

□

Problema III.11 Escreva uma função que recebe um vetor real v, cujos elementos estão ordenadaos em ordem não-decrescente, um inteiro n que indica o tamanho do vetor e um real r, e devolva a posição onde r ocorre no vetor. Se a função não encontrar r, devolva -1.

Solução:

```
1  int buscasimples(float v[MAX], int n, float r)
2  {
3    int i;
4    for(i = 0; i < n; i++)
5    {
6      if(v[i] == r)
7        return i;
8    }
9    return -1;
10 }
```

Busca binária. Localiza o meio do vetor e descarta o trecho que não interessa.

```
1  int buscabinaria(float v[MAX], int n, float r)
2  {
3    int meio;
4    int inicio = 0, fim = n;
5    while(inicial < fim)
6    {
7      meio = (inicio + fim)/2;
8      if(v[meio] == r)
9        return meio;
10     if(v[meio] < r)
11       inicio = meio + 1;
12     else
13       fim = meio;
14   }
15   return -1;
16 }
```

□

Problema III.12 Escreva uma função que recebe um vetor real v e um inteiro n com o tamanho do vetor, e devolve o vetor ordenado em ordem crescente.

Solução:
Existem pelo menos três métodos de seleção:

- selection sort
- insertion sort
- bubble sort

selection sort

```
1  //selection sort
2  int max(float v[MAX], int n)
3  {
4      int i, m = v[0], imaior = 0;
5      for (i = 1; i < n; i++)
6      {
7          if (m < v[i])
8          {
9              m = v[i];
10             imaior = i;
11         }
12     }
13     return imaior;
14 }
15
16 void selecao(float v[MAX], int n)
17 {
18     int maior, aux;
19     while (n > 1)
20     {
21         maior = max(v,n);
22         aux = v[n-1];
23         v[n-1] = v[maior];
24         v[maior] = aux;
25         n--;
26     }
27 }
```

insertion sort

```
1  //insertion sort
2  void insercao(float v[MAX], int n)
3  {
4      int i, j, aux;
5      for (i = 1; i < n; i++)
6      {
7          j = i;
8          while (j > 0 && v[j] < v[j-1])
9          {
10             aux = v[j-1];
11             v[j-1] = v[j];
12             v[j] = aux;
13             j--;
14         }
15     }
16 }
```

buble sort

```
1  //bubble sort
2  void bolha(float v[MAX], int n)
3  {
4      int i, aux, trocou = TRUE;
5      while (trocou)
6      {
7          trocou = FALSE;
8          for (i = 0; i < n - 1; i++)
9          {
10             if (v[i] > v[i+1])
11             {
12                 aux = v[i+1];
13                 v[i+1] = v[i];
14                 v[i] = aux;
15                 trocou = TRUE;
16             }
17         }
18     }
19 }
```

```
//buble sort2
void bolha2(float v[MAX], int n)
{
    int i, aux, j;
    for (j = 0; j < n; j++)
    {
        for (i = 0; i < n - 1; i++)
        {
            if (v[i] > v[i+1])
            {
                aux = v[i+1];
                v[i+1] = v[i];
                v[i] = aux;
            }
        }
    }
}
```

□

Problema III.13 a) Escreva uma função que recebe como parâmetros:

- dois inteiros positivos n e m;
- uma matriz $A_{n\times m}$;
- o índice c de uma coluna;
- os índices k e p de duas linhas;

e ordena entre as linhas k e p da matriz A segundo a coluna c.

b) Dados n datas em uma matriz $DATA_{N\times 3}$, onde a primeira coluna corresponde ao dia, a segunda ao mês e a terceira ao ano, coloque essas datas em ordem cronológica crescente, usando a função acima.

Solução:

```
#include <iostream>
#define TRUE 1
#define FALSE 0
#define MAX 100
using namespace std;

//bubble sort
void bolha_matriz(int n, int m, int A[MAX][MAX], int c, int
    k, int p)
{
    int i, j, aux, trocou = TRUE;
    while (trocou)
    {
        trocou = FALSE;
        for (i = k; i < p; i++)
        {
            if (A[i][c] > A[i+1][c])
            {
```

```
18                  for (j = 0; j < m; j++)
19                  {
20                      aux = A[i+1][j];
21                      A[i+1][j] = A[i][j];
22                      A[i][j] = aux;
23                  }
24                  trocou = TRUE;
25              }
26          }
27      }
28  }
29
30  int main ()
31  {
32      int DATA[MAX][MAX], n, i;
33      cin >> n;
34      for (i = 0; i < n; i++)
35          cin >> DATA[i][0] >> DATA[i][1] >> DATA[i][2];
36      bolha_matriz(n,3,DATA,0,0,n-1);
37      bolha_matriz(n,3,DATA,1,0,n-1);
38      bolha_matriz(n,3,DATA,2,0,n-1);
39      cout << endl;
40      for (i = 0; i < n; i++)
41          cout << DATA[i][0] << " " << DATA[i][1] << " " <<
                DATA[i][2] << endl;
42      return 0;
43  }
```

□

Problema III.14 Escreva um programa que leia

- um inteiro $k > 0$;
- k pares de números inteiros que indicam as posições (ou seja, a linha e a coluna) de k rainhas em um tabuleiro de xadrez;
- um inteiro $n > 0$, e
- n pares de números inteiros indicando posições no tabuleiro.

e para cada uma das n posições determinar se ela está ou não sob ataque de alguma rainha.

Para resolver este problema:

a) Escreva uma função de protótipo
`void inicializa_tabuleiro(char tab[MAX][MAX])` que inicializa a matriz `tab` com todas as posições vazias.

b) Escreva uma função de protótipo
`void imprima_tabuleiro(char tab[MAX][MAX])` que imprime o tabuleiro.

c) Escreva uma função de protótipo
`int atacada(char tab[MAX][MAX], int linha, int coluna, int *linha_r, int *coluna_r)` que devolve 1 se a posição `(linha,coluna)`

está sendo atacada por alguma rainha representada pela letra R no tabuleiro `tab` e devolve 0 caso contrário.

Se a posição estiver sendo atacada, então `(*linha_r,*coluna_r)` devolve a posição de uma rainha que ataca essa posição.

Solução:

```
#include <iostream>
#define MAX 8
using namespace std;

void inicializa_tabuleiro(char tab[MAX][MAX])
{
    int i, j;
    for (i = 0; i < MAX; i++)
    {
        for (j = 0; j < MAX; j++)
            tab[i][j] = '-';
    }
}

void imprima_tabuleiro(char tab[MAX][MAX])
{
    int i, j;
    for (i = 0; i < MAX; i++)
    {
        for (j = 0; j < MAX; j++)
            cout << tab[i][j];
        cout << endl;
    }
}

int atacada(char tab[MAX][MAX], int linha, int coluna, int *
    linha_r, int *coluna_r)
{
    int i, j, d1, d2;
    d1 = linha - coluna;
    d2 = linha + coluna;
    for (i = 0; i < MAX; i++)
    {
        if (tab[linha][i] == 'R')
        {
            *linha_r = linha;
            *coluna_r = i;
            return 1;
        }
        if (tab[i][coluna] == 'R')
        {
            *linha_r = i;
            *coluna_r = coluna;
            return 1;
        }
        j = i - d1;
```

```
46          if (j >= 0 && j < MAX && tab[i][j] == 'R')
47          {
48              *linha_r = i;
49              *coluna_r = j;
50              return 1;
51          }
52          j = d2 - i;
53          if (j >= 0 && j < MAX && tab[i][j] == 'R')
54          {
55              *linha_r = i;
56              *coluna_r = j;
57              return 1;
58          }
59      }
60      return 0;
61  }
62
63  int main()
64  {
65      int n, i, k, l, c, rl, rc;
66      char tab[MAX][MAX];
67      inicializa_tabuleiro(tab);
68      cout << "Digite o numero de rainhas: ";
69      cin >> k;
70      for (i = 0; i < k; i++)
71      {
72          cout << "Digite a posicao da rainha: ";
73          cin >> l >> c;
74          tab[l][c] = 'R';
75      }
76      imprima_tabuleiro(tab);
77      cout << "Digite o numero de posicoes que podem ser
            atacadas: ";
78      cin >> n;
79      for (i = 0; i < n; i++)
80      {
81          cout << "Digite essas posicoes: ";
82          cin >> l >> c;
83          if (atacada(tab,l,c,&rl,&rc))
84              cout << "Atacada por (" << rl << "," << rc << ")
                    ." << endl;
85          else
86              cout << "Nao atacada." << endl;
87      }
88      return 0;
89  }
```

□

6.1 string

string é uma sequência de caracteres terminada por \0.

Problema III.15 Dada uma *string* contar o número de ocorrência de cada letra do alfabeto.

Solução:

```
#include <stdio.h>
#define MAX 100

int main ()
{
    char string[MAX], c;
    int histograma[26], i, pos;
    printf("Digite uma sequencia de caracteres:\n");
    fgets(string,MAX,stdin);
    for (i = 0; i < 26; i++)
        histograma[i] = 0;
    for (i = 0; string[i] != '\0'; i++)
    {
        if (string[i] >= 'A' && string[i] <= 'Z')
            string[i] = string[i] - 'A' + 'a';
        if (string[i] >= 'a' && string[i] <= 'z')
        {
            pos = string[i] - 'a';
            histograma[pos]++;
        }
    }
    for (c = 'a'; c <= 'z'; c++)
    {
        pos = c - 'a';
        printf("%c %d\n",c,histograma[pos]);
    }
    return 0;
}
```

□

Problema III.16 a) Dado um vetor real x com n elementos e um certo índice k, escreva uma função que determina o índice do elemento mínimo entre $x[k]$ e $x[n-1]$.

b) Usando a função do item anterior coloque os elementos de um vetor em ordem crescente.

Solução:

```c
#include <stdio.h>
#define MAX 100

int min(float X[MAX], int n, int k)
{
    int min, imin, i;
    min = X[k]; /* ou tire essa linha */
    imin = k;
    for (i = k + 1; i < n; i++)
        if (min > X[i]) /* ou (X[imin] > X[i]) */
        {
            min = X[i]; /* tire essa linha */
            imin = i;
        }
    return imin;
}

void ordena(float X[MAX], int n)
{
    int i, menor, aux;
    for (i = 0; i < n; i++)
    {
        menor = min(X,n,i);
        aux = X[menor];
        X[menor] = X[i];
        X[i] = aux;
    }
}

int main()
{
    int n, i;
    float X[MAX];
    printf("Digite o tamanho da sequencia: ");
    scanf("%d",&n);
    printf("Digite os valores: ");
    for (i = 0; i < n; i++)
        scanf("%f",&X[i]);
    ordena(X,n);
    for (i = 0; i < n; i++)
        printf("%f ",X[i]);
    printf("\n");
    return 0;
}
```

□

Prova 01

A

1. A partir de um número inteiro $n > 1$ podemos gerar a seguinte sequência de números: se n é par então o próximo número da sequência é $n/2$ e se n é impar então o próximo número da sequência é $3n+1$ e assim sucessivamente. Acredita-se que essa sequência sempre chegará em 1 (e nesse exercício você deve supor que isso acontecerá). Por exemplo, para $n = 3$, temos que a sequência é:

$$3 \quad 10 \quad 5 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

pois 3 é impar e $3*3+1 = 10$, 10 é par e portanto o terceiro número da sequência é 5, 5 é impar e $3*5+1 = 16$ e os números 16, 8, 4 e 2 são todos pares.

Escreva um programa que, dado um número inteiro $n > 1$, imprime qual é o **tamanho** da sequência descrita acima quando começamos com n. Por exemplo, para $n = 3$, seu programa deve imprimir 8.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, tamanho = 1;
    cout << "Digite um numero: ";
    cin >> n;
    while (n > 1)
    {
        tamanho ++;
        if (n % 2 == 0)
            n /= 2;
        else
            n = 3 * n + 1;
    }
    cout << tamanho << endl;
    return 0;
}
```

□

2. Leonardo Fibonacci, estudando o ritmo de crescimento da população de coelhos através, criou uma sequência de números naturais que passou a ser conhecida como *sequência de Fibonacci*. O n-ésimo número da sequência de Fibonacci F_n e é definido pela seguinte fórmula de recorrência:

$$F_n = \begin{cases} 1 & , \text{se } n = 1 \\ 1 & , \text{se } n = 2 \\ F_{n-1} + F_{n-2} & , \text{se } n > 2 \end{cases}$$

Faça um programa que, dado um inteiro $n > 0$, imprima os n primeiros números da sequência de Fibonacci. Por exemplo, para $n = 7$, seu programa deverá imprimir

$$1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13$$

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, anterior, atual, i, temp;
    anterior = atual = 1;
    cout << "Digite o numero de termos da sequencia de
        Fibonacci: ";
    cin >> n;
    cout << anterior << " ";
    if (n > 1)
        cout << atual << " ";
    for (i = 2; i < n; i++)
    {
        temp = atual;
        atual += anterior;
        anterior = temp;
        cout << atual << " ";
    }
    cout << endl;
    return 0;
}
```

□

3. Dado um inteiro $n > 0$, calcule a soma dos números entre 2 e n (inclusive) que são primos. Lembre-se que um número p é primo se ele é maior do que 1 e seus divisores (positivos) são apenas 1 e p.

Solução:

```
#include <iostream>
#define TRUE 1
#define FALSE 0
using namespace std;
int main()
{
    int n, i, divisor, primo, soma = 0;
    cout << "Digite um numero: ";
    cin >> n;
    for (i = 2; i <= n; i++)
    {
        primo = TRUE;
        for (divisor = 2; divisor < i; divisor ++)
        {
            if (i % divisor == 0)
                primo = FALSE;
        }
        if (primo == TRUE)
            soma += i;
    }
```

```
21      cout << soma << endl;
22      return 0;
23  }
```

□

4. O Último Teorema de Fermat afirma que **não** existem inteiros positivos x, y, z e n, onde $n > 2$, tal que $x^n + y^n = z^n$. Nesta questão, o seu objetivo é criar um programa que testa tal teorema. Dados x e y e dois inteiros positivos a e b, seu programa deverá imprimir $x^n + y^n - z^n$ (que pelo teorema será sempre diferente de zero) com z variando entre 1 e a (inclusive) e n variando entre 3 e b (inclusive). Por exemplo, para $x = 2$, $y = 3$, $a = 3$ e $b = 4$ seu programa deverá imprimir 34, 27, 8, 96, 81 e 16, pois

$$\begin{array}{lrcrcr}
x=2,y=3,z=1,n=3: & 2^3+3^3-1^3 & = & 8+27-1 & = & 34 \\
x=2,y=3,z=2,n=3: & 2^3+3^3-2^3 & = & 8+27-8 & = & 27 \\
x=2,y=3,z=3,n=3: & 2^3+3^3-3^3 & = & 8+27-27 & = & 8 \\
x=2,y=3,z=1,n=4: & 2^4+3^4-1^4 & = & 16+81-1 & = & 96 \\
x=2,y=3,z=2,n=4: & 2^4+3^4-2^4 & = & 16+81-16 & = & 81 \\
x=2,y=3,z=3,n=4: & 2^4+3^4-3^4 & = & 16+81-81 & = & 16
\end{array}$$

Solução:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int x, y, z, n, a, b, i, potx, poty, potz;
6       cout << "Digite quatro numeros: ";
7       cin >> x >> y >> a >> b;
8       for (n = 3; n <= b; n++)
9       {
10          for (z = 1; z <= a; z++)
11          {
12              potx = poty = potz = 1;
13              for (i = 1; i <= n; i++)
14              {
15                  potx *= x;
16                  poty *= y;
17                  potz *= z;
18              }
19              cout << potx + poty - potz << endl;
20          }
21      }
22      return 0;
23  }
```

□

B

Prova 02

1. Escreva um programa que recebe um número $\varepsilon > 0$ e calcule uma aproximação de π usando a seguinte fórmula:

$$\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

Você deverá somar até o primeiro termo $\frac{(-1)^k}{2k+1}$ que em módulo seja menor ou igual a ε.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int k = 0, sinal = 1;
    float epsilon, pi = 0, termo;
    cout << "Digite um valor para epsilon: ";
    cin >> epsilon;
    do
    {
        termo = (float) sinal / (2 * k + 1);
        sinal *= -1;
        k++;
        pi += termo;
        if (termo < 0)
            termo =- termo;
        cout << pi << endl;
    } while (termo > epsilon);
    pi *= 4;
    cout << pi << endl;
    return 0;
}
```

□

2. A cifra de César consiste em, dada uma mensagem e um número inteiro $n > 0$, trocar cada letra da mensagem pela n-ésima letra que aparece após essa letra no alfabeto (voltando para o começo do alfabeto se preciso). Por exemplo, para $n = 3$

normal →	A	B	C	…	V	W	X	Y	Z
cifra →	D	E	F	…	Y	Z	A	B	C

Escreva um programa que recebe um inteiro n ($0 \leqslant n \leqslant 26$) e uma mensagem terminada por '.' e aplicar a cifra de César (usando o n dado) na frase.

Solução:

```
#include <iostream>
using namespace std;
int main()
{
    int n, k = 0;
    char c, d;
    cout << "Digite duas letras: ";
    cin >> n >> c;
    while (c != '.')
    {
        d = c + n;
        if (d > 'z')
            d = 'Z' - 'A' + 1;
        cout << d;
        if (k != 25)
            c = 'a' + k;
        else
            c = '.';
        k++;
    }
    cout << "." << endl;
    return 0;
}
```

□

3. Escreva uma função com protótipo `int chuta(int chute)` que recebe um inteiro positivo *chute* e verifica se *chute* é menor ou maior do que uma constante `numero_magico` (imagine que essa constante já está definido anteriormente). Devolva −1 se chute for menor do que `numero_magico`, 0 se for igual e 1 se for maior.

Escreva uma função com protótipo `encontra(int max)` que recebe um inteiro positivo `max` e retorna o valor de `numero_magico` usando apenas a função acima (sem usar a constante `numero_magico`) ou retorna −1 se não encontrar `numero_magico` no intervalo de 1 a `max`.

Escreva um programa que dado um número n, imprime, usando a função anterior, o `numero_magico` se `numero_magico` está entre 1 e n e imprime "não encontrei" caso contrário.

Solução:

```
#include <iostream>
#define NUMERO_MAGICO 10
using namespace std;

int chuta(int chute)
{
    if (chute == NUMERO_MAGICO)
        return 0;
    else if (chute < NUMERO_MAGICO)
        return -1;
```

```
11      else
12          return 1;
13  }
14
15  int encontra(int max)
16  {
17      int i;
18      for (i = 1; i <= max; i++)
19      {
20          if (chuta(i) == 0)
21              return i;
22      }
23      return -1;
24  }
25
26  int main()
27  {
28      int n, res;
29      cout << "Digite um numero: ";
30      cin >> n;
31      res = encontra(n);
32      if (res == -1)
33          cout << "Nao encontrei." << endl;
34      else
35          cout << res << endl;
36      return 0;
37  }
```

□

4. Escreva uma função com protótipo
`void somabit(int b1, int b2, int *vaium, int *soma)` que recebe três bits (inteiros entre 0 e 1) `b1` e `b2` e `vaium` e devolve um bit `*soma` representando a soma dos três e um novo bit "vai um" em `*vaium`.

Escreva um programa que leia dois números em binário e calcula um número em binário. Que é a soma dos dois números dados. Utilize a função acima. A entrada será dada através de dois números inteiros.

Solução:

```
1   #include <iostream>
2   using namespace std;
3
4   void somabit(int b1, int b2, int *vaium, int *soma)
5   {
6       int temp = b1 + b2 + *vaium;
7       *vaium = temp / 2;
8       *soma = temp % 2;
9   }
10
11  int main()
12  {
13      int m, n, vaium = 0, soma, pot = 1, total = 0;
```

```
14      cout << "Digite dois numeros: ";
15      cin >> n >> m;
16      while (n > 0 || m > 0)
17      {
18          somabit (n % 10, m % 10, &vaium, &soma);
19          total += pot * soma;
20          pot *= 10;
21          n /= 10;
22          m /= 10;
23      }
24      total += pot * vaium;
25      cout << total << endl;
26      return 0;
27  }
```

□

Referências Bibliográficas

[1] D. RITCHIE and B. KERNIGHAN. C a linguagem de programação padrão ansi. *Rio de Janeiro: Campus, c1989. 289p*, 1989.

[2] D. Walters. C programming. a modern approach. by kn king. ww norton & company: New york. 1996. 661 pp. isbn 0-393-96945-2. *Journal of Chemical Information and Computer Sciences*, 36(5):1050–1050, 1996.