



Capítulo 5

Estruturas homogêneas: vetores e matrizes

CONCEITO GERAIS

Apesar de você ter feito uso das estruturas de repetição que lhe permitiram a entrada de vários dados, ainda não foi possível o armazenamento de todos esses dados, de uma vez só, ou não, pois lhe faltava conhecer a estrutura do vetor.

Um vetor é um arranjo de elementos armazenados na MP, um após o outro, todos com o mesmo nome. A idéia é a mesma de uma **matriz linha** da matemática, isto é, várias colunas e uma linha.

Assumiremos que a primeira posição do vetor é 0.

2	4	5	8	12	3	56	34
0	1	2	3	4	5	6	7
A [2 4 5 8 12 3 56 34]							

Esse é um vetor de 8 elementos, isto é, tem 8 variáveis, todas com o mesmo nome e diferentes por sua posição dentro do arranjo que é indicada por um índice.

Quando se tem somente uma linha, podemos omiti-la e colocar somente a coluna.

$$A_0 = 2 \quad A_1 = 4 \quad A_2 = 5 \quad A_3 = 8 \quad A_4 = 12 \quad A_5 = 3 \quad A_6 = 56 \quad A_7 = 34$$

$A[0] = 2 \ A[1] = 4 \ A[2] = 5 \ A[3] = 8 \ A[4] = 12 \ A[5] = 3 \ A[6] = 56 \ A[7] = 34$

Para dimensionar um vetor, usamos o seguinte comando na declaração de variáveis:

tipo nome[dimensão]

onde **dimensão**, na prática, é o número de elementos:

[5] 5 elementos (de 0 a 4)

tipo poderá ser: **int, real ou string**
nome será o que você dará ao vetor dentro das regras para nomear
 uma variável

No exemplo acima seria:

int A[8]

Em algoritmos também podemos ter vetor caracter e lembre-se: a variável caracter é armazenada na MP como sendo um vetor, mesmo sem a declararmos como tal. Para melhor entendimento, a representação será na vertical.

Linguagens que ASSUMEM a posição

real A[5];
int B[10];
string NOMES[20];

A	0	1	2	3	4

B	0	1	2	3	4	5	6	7	8	9

NOMES

0	1	...			
0					
1					
2					
3					
4					
5					
6					
...					
18					
19					

↳ Um vetor **string**, na verdade, é uma matriz, pois como já falamos cada variável **string** é um vetor.

Um algoritmo com vetor implica vários trechos para que possa funcionar corretamente. Esses trechos são independentes.

- TRECHO DE DIMENSIONAMENTO – já visto anteriormente
- TRECHO DE ENTRADA DE DADOS
 - normalmente, uma estrutura de repetição
 - se for a estrutura do para, deverá ter o valor final igual à **última posição** do vetor.
 - se for a estrutura do enquanto ou faca enquanto, deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

```
para( L<- 0; L < tamanho do vetor ; L++ )
{
  imprima ... ;
  leia nomedovetor [ L ];
}
```

- o trecho anterior poderá ser incrementado com outros comandos;
- **nunca** deverá ter um comando *imprima* tendo como argumento o **nomedovetor [L]**;
- poderá ter: estrutura de seleção, atribuição, outro para etc.

- TRECHO DE SAÍDA
 - normalmente, uma estrutura de repetição;
 - se for a estrutura do para, deverá ter o valor final igual à **última posição** do vetor;
 - se for a estrutura do enquanto ou faca enquanto, deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

```
para( L<- 0; L < tamanho do vetor; L++ )
{
  imprima "\n", nomedovetor [ L ] ;
}
```

- o trecho acima poderá ser incrementado de outros comandos;
- **nunca** deverá ter um comando *leia* tendo como argumento o **nomedovetor [L]**;
- poderá ter: estrutura de seleção, atribuição, outro para etc.

algoritmo 347

Criar um algoritmo que entre com dez nomes e imprima uma listagem contendo todos os nomes.

```
prog lerimp
    int L;
    string nomes[10];
    # trecho de entrada
    para( L<= 0; L <= 9 ; L++)
    {
        imprima "\ndigite ", L + 1, " nome: ";
        leia nomes [L] ;
    }
    # fim do trecho de entrada
    # trecho de saida
    imprima "\n\n";
    para( L<= 0; L <= 9 ; L++)
    {
        imprima "\n", nomes[ L];
    }
    # fim do trecho de saida
    imprima "\n";
fimprog
```

```
digite 1 nome: ANITA
digite 2 nome: JOAO
digite 3 nome: PEDRO
digite 4 nome: MARIA
digite 5 nome: FILIPE
digite 6 nome: ANGELA
digite 7 nome: ANA
digite 8 nome: REGINA
digite 9 nome: LUIZ
digite 10 nome: TEREZA
```

```
ANITA
JOAO
PEDRO
MARIA
FILIPE
ANGELA
ANA
REGINA
LUIZ
TEREZA
```

Quando se têm dados para mais de um vetor relativos a uma pessoa ou a um objeto, os trechos de entrada são separados? NÃO devem ser (exemplo: **prog imprimemedia**).

Quando se têm dados para mais de um vetor que não têm nada em comum, os trechos de entrada são separados? DEVEM ser (exemplo: **prog somavetores**).

algoritmo 348

Criar um algoritmo que armazene nome e duas notas de 5 alunos e imprima uma listagem contendo nome, as duas notas e a média de cada aluno.

```
prog imprimemedia
    int L;
    string nomes[5];
    real pr1[5], pr2[5], media[5];
```

```

# trecho de entrada
para( L<- 0; L <= 4 ; L++)
{
    imprima "\ndigite ", L + 1, " nome: ";
    leia nomes[ L ] ;
    imprima "digite 1a nota: ";
    leia pr1[L];
    imprima "digite 2a nota: ";
    leia pr2[L];
    media[L] <- (pr1[L] + pr2[L])/2;
}
# fim do trecho de entrada
#trecho de saida
imprima "\n\n\n\t\t\tRELACAO FINAL\n";
para( L<- 0; L <= 4 ; L++)
{ imprima "\n", L+1, "- ", nomes[L];
  imprima "\n",pr1[L],"\t", pr2[L], "\t",media[L];}
# fim do trecho de saida
imprima "\n";
fimprog

```

algoritmo 349

Criar um algoritmo que armazene números em dois vetores inteiros de cinco elementos cada. Gere e imprima o vetor soma.

```

prog somavetores
    int L, a[5], b[5], soma[5];
    para( L<- 0; L <= 4 ; L++)
    {
        imprima "\nelemento do vetor
A[", L + 1, "]: ";
        leia a[ L ];
    }
    imprima "\n";
    para( L<- 0; L <= 4 ; L++)
    {
        imprima "\nelemento do vetor
B[", L + 1, "]: ";
        leia b[ L ];
        soma[L] <- a[L] + b[L];
    }
    imprima "\n\n\nVetor Soma\n";
    para( L<- 0; L <= 4 ; L++)
    { imprima "\nsoma[",L + 1, "] - "
    , soma[ L ];}
    imprima "\n";
fimprog

```

*elemento do vetor A[1]: 12
 elemento do vetor A[2]: 10
 elemento do vetor A[3]: 9
 elemento do vetor A[4]: 4
 elemento do vetor A[5]: 5*

*elemento do vetor B[1]: 8
 elemento do vetor B[2]: 10
 elemento do vetor B[3]: 11
 elemento do vetor B[4]: 16
 elemento do vetor B[5]: 15*

Vetor Soma

*soma[1] - 20
 soma[2] - 20
 soma[3] - 20
 soma[4] - 20
 soma[5] - 20*

ORDENANDO VETORES

Compare o elemento que está na 1^a posição com todos os seguintes a ele. Quando for necessário, troque-o de lugar, fazendo uso da variável aux. Repita essa operação até que tenha feito todas as comparações.

1^a Fase

1º passo: Se o nome da posição 0 na ordem alfabética vier depois do nome na posição 1, então inverta-os. Como isso não acontece, tudo fica igual.

NOMES		AUX	NOMES	
0	JOAO		0	JOAO
1	RUI		1	RUI
2	IVO		2	IVO
3	BIA		3	BIA
4	ANA		4	ANA

2º passo: Se o nome da posição 0 na ordem alfabética vier depois do nome na posição 2, então inverta-os.

NOMES		AUX	NOMES	
0	JOAO	JOAO	0	IVO
1	RUI		1	RUI
2	IVO		2	JOAO
3	BIA		3	BIA
4	ANA		4	ANA

3º passo: Se o nome da posição 0 na ordem alfabética vier depois do nome na posição 3, então inverta-os.

NOMES		AUX	NOMES	
0	IVO	IVO	0	BIA
1	RUI		1	RUI
2	JOAO		2	JOAO
3	BIA		3	IVO
4	ANA		4	ANA

4º passo: Se o nome da posição 0 na ordem alfabética vier depois do nome na posição 4, então inverta-os.

NOMES		AUX	NOMES	
0	BIA	B/A	0	ANA
1	RUI		1	RUI
2	JOAO		2	JOAO
3	IVO		3	IVO
4	ANA		4	BIA

- Após essas comparações e trocas, conseguimos colocar na posição 0 o nome ANA.
- Partiremos para a busca do segundo nome da lista.
- Não mais nos preocuparemos com a posição 0.

2ª Fase

1º passo: Se o nome da posição 1 na ordem alfabética vier depois do nome na posição 2, então inverta-os.

NOMES		AUX	NOMES	
0	ANA	RUI	0	ANA
1	RUI		1	JOAO
2	JOAO		2	RUI
3	IVO		3	IVO
4	BIA		4	BIA

2º passo: Se o nome da posição 1 na ordem alfabética vier depois do nome na posição 3, então inverta-os.

NOMES		AUX	NOMES	
0	ANA	JOAO	0	ANA
1	JOAO		1	IVO
2	RUI		2	RUI
3	IVO		3	JOAO
4	BIA		4	BIA

3º passo: Se o nome da posição 0 na ordem alfabética vier depois do nome na posição 4, então inverta-os.

NOMES		AUX	NOMES	
0	ANA	IVO	0	ANA
1	IVO		1	BIA
2	RUI		2	RUI
3	JOAO		3	JOAO
4	BIA		4	IVO

- Após essas comparações e trocas, conseguimos colocar na posição 1 o nome BIA.
- Partiremos para a busca do terceiro nome da lista.
- Não mais nos preocuparemos com as posições 0 e 1.

3ª Fase

1º passo: Se o nome da posição 2 na ordem alfabética vier depois do nome na posição 3, então inverta-os.

NOMES		AUX	NOMES	
0	ANA	RUI	0	ANA
1	BIA		1	BIA
2	RUI		2	JOAO
3	JOAO		3	RUI
4	IVO		4	IVO

2º passo: Se o nome da posição 2 na ordem alfabética vier depois do nome na posição 4, então inverta-os.

NOMES		AUX	NOMES	
0	ANA	JOAO	0	ANA
1	BIA		1	BIA
2	JOAO		2	IVO
3	RUI		3	RUI
4	IVO		4	JOAO

- Após essas comparações e trocas, conseguimos colocar na posição 1 o nome BIA.
- Partiremos para a busca do terceiro nome da lista.
- Não mais nos preocuparemos com as posições 0, 1 e 2.

4^a Fase

1º passo: Se o nome da posição 3 na ordem alfabética vier depois do nome na posição 4, então inverta-os.

NOMES		AUX	NOMES	
0	ANA	RUI	0	ANA
1	BIA		1	BIA
2	IVO		2	IVO
3	RUI		3	JOAO
4	JOAO		4	RUI

Conclusões

- Apesar de termos cinco nomes, só tivemos quatro fases, pois quando compararamos o penúltimo com o último, economizamos uma fase.
- O número de comparações foi diminuindo em cada fase. Isso se explica porque o número de combinações também estava diminuindo, uma vez que diminuía o número de nomes abaixo do que estava em evidência.
- Vamos tentar montar o trecho de ordenação:

1^a linha:

para(L <-0; L < 4 ; L++)

Por que de 0 até < 4?

Porque as fases envolveram as buscas das posições 0,1, 2 e 3.

2^a linha:

{

Inicia o bloco do para

3^a linha:

para(c <- L + 1; c <= 4 ; c ++)

Por que de L + 1 até <= 4?

Porque, em cada fase, comparávamos a posição em evidência com todas as outras posições até a última.

4^a linha:

{

Inicia o bloco do para

5^a linha:

se(nomes[L] > nomes[c])

Pense no L como sendo a fase e, em c como sendo uma das etapas da fase.

6^a linha:

{

Inicia o bloco do se

7^a linha/8^a linha/9^a linha:

```
aux <- nomes[L] ;
nomes[L] <- nomes[c] ;
nomes[c] <- aux;
```

Este trecho permite a troca entre as duas variáveis.

10^a linha:

}

Finaliza o bloco do se

11^a linha:

}

Finaliza o bloco do para

12^a linha:

}

Finaliza o bloco do para

Trecho de ordenação

```
para( L <-0; L < 4 ; L++)
{
    para( c <- L + 1; c <= 4 ; c++)
    {
        se( nomes[L] > nomes[c] )
        {
            aux <- nomes[L] ;
            nomes[L] <- nomes[c] ;
            nomes[c] <- aux;
        }
    }
}
```

↳ Se precisarmos ordenar um vetor e se tivermos outros vetores dependentes dele, precisaremos fazer vários trechos de troca. Isto é, todo trecho que se encontra no retângulo mais interno.

Alguns algoritmos básicos:

algoritmo 350

Criar um algoritmo que armazene 5 nomes em um vetor. Ordenar e imprimir uma listagem.

```
prog ordenanomemevetor
    int L, c;
    string nomes[5], aux;
    para( L<- 0; L <= 4 ; L++)
    {   imprima "\n nome: " ; leia nomes[L]
    ;
    para( L<- 0; L <= 3 ; L++)
    {   para( c<- L + 1; c <= 4 ; c++)
        {   se( nomes[L] > nomes[c] )
            {   aux <- nomes[L];
                nomes[L] <- nomes[c];
                nomes[c] <- aux;   }
        }
    imprima "\n\n ";
    para( L<- 0; L <= 4 ; L++)
    {   imprima "\n", L +1, " - ", nomes[L]
    ;
    imprima "\n";
    fimprog
```

nome: GUTO GARCIA
nome: PEDRO CORREA
nome: ANITA LOPEZ
nome: MARIA CORREA
nome: JOAO BOND

ANITA LOPEZ
GUTO GARCIA
JOAO BOND
MARIA CORREA
PEDRO CORREA

algoritmo 351

Criar um algoritmo que armazene cinco nomes em um vetor e depois possa ser digitado um número que corresponde a uma pessoa e imprimir esse nome.

```
prog procuranomepelonumero
    int L, num ;
    string nomes [5];
    para( L<- 0; L <= 4 ; L++)
    {   imprima "\n nome " , L + 1, ": "; leia
    nomes[L] ;
    #entra com numero para procurar pela
    posicao
    imprima "\nDigite o numero da pessoa: ";
    leia num;
    enquanto(num < 1 || num > 5)
```

nome 1: ANITA
nome 2: GUTO
nome 3: JOAO
nome 4: PEDRO
nome 5: FILIPE

Digite o numero da pessoa: 3

JOAO

```

{
imprima "\n Numero fora do intervalo";
imprima "\n Digite o numero da pessoa: ";
leia num;
}
imprima "\n", nomes[ num -1] ;
imprima "\n";
fimprog

```

algoritmo 352

Criar um algoritmo que armazene cinco nomes em um vetor e depois possa ser digitado um nome e, se for encontrado, imprimir a posição desse nome no vetor; caso contrário, imprimir uma mensagem.

```

prog procuranomepelonome
# procura pelo nome
int L ;
string nome, nomes [5];
para( L<- 0; L <= 4 ; L++)
{ imprima "\nnome: " ;
  leia nomes[L] ;
}
# entra com nome para procura
imprima "\n Digite nome para
procura: " ;
leia nome;
L <-0;
enquanto( L<4 &&
nomes< >nomes[L] ) {
  { L++; }
  se(nome ==nomes[L] )
  {
    imprima "\nencontrei na
posicao: ", L+1;
  }
  senao
  {
    imprima "\nnao encontrei" ;
  }
  imprima "\n";
fimprog

```

nome: ANITA
nome: GUTO
nome: JOAO
nome: PEDRO
nome: FILIPE
Digite nome para procura: ANITA
LOPES
nao encontrei
nome: ANITA
nome: GUTO GARCIA
nome: JOAO
nome: PEDRO
nome: FILIPE
Digite nome para procura: GUTO
GARCIA
encontrei na posicao: 2

EXERCÍCIOS – LISTA 5

LISTA DE VETORES

algoritmo 353

Armazenar 10 nomes em um vetor NOME e imprimir uma listagem numerada.

```
prog vetor1
    int L;
    string nomes[10];
    # trecho de entrada de 10 nomes
    para( L<= 0; L <= 9 ; L++)
    {
        imprima "\nDigite ", L + 1, " nome: ";
        leia nomes[ L ] ;
    }
    # fim do trecho de entrada
    # trecho de saida
    imprima "\nRELACAO DAS PESSOAS\n";
    para( L<= 0; L <= 9 ; L++)
    {
        imprima "\n", L+1, "- ", nomes[ L ];
    }
    # fim do trecho de saida
    imprima "\n";
fimprog
```

algoritmo 354

Armazenar 15 números inteiros em um vetor NUM e imprimir uma listagem numerada contendo o número e uma das mensagens: par ou ímpar.

```
prog vetor2
    int L , num[15];
    # trecho de entrada de 15 elementos
    para( L<= 0; L <= 14 ; L++)
    {
        imprima "\ndigite ", L + 1, " numero: ";
        leia num[ L ] ;
    }
    # fim do trecho de entrada
    # trecho de saida
    imprima "\nRELACAO DOS NUMEROS\n";
    para( L<= 0; L <= 14 ; L++)
    {
        imprima "\n", L+1, "- ", num[L];
        se(num[L] %2==0)
            { imprima " e' PAR";  }
        senao
            { imprima " e' IMPAR";  }
    }
278| }
```

```

# fim do trecho de saída
imprima "\n";
fimprog

```

algoritmo 355

Armazenar 8 números em um vetor e imprimir todos os números. Ao final, temos o total de números múltiplos de seis digitados.

```

prog vetor3
int L , cont, num[8];
cont <- 0;
# trecho de entrada
para( L<- 0; L <= 7 ; L++)
{
    imprima "\ndigite ", L + 1, " numero: ";
    leia num[ L ] ;
    se(num[L] %6==0)
    { cont++;}
}
# fim do trecho de entrada
# trecho de saída
imprima "\nRELACAO DOS NUMEROS\n";
para( L<- 0; L <= 7 ; L++)
{
    imprima "\n", L+1, "- ", num[L];
}
# fim do trecho de saída
imprima "\n Total de multiplos de 6: ",cont;
imprima "\n";
fimprog

```

algoritmo 356

Armazenar nomes e notas das PR1 e PR2 de 15 alunos. Calcular e armazenar a média arredondada. Armazenar também a situação do aluno: AP ou RP. Imprimir uma listagem contendo nome, notas, média e situação de cada aluno, tabulando.

```

prog vetor4
int L ,c ,t, media[15];
string nomes[15], sit[15];
real pr1[15], pr2[15];
# trecho de entrada de 15 nomes
para( L<- 0; L <= 14 ; L++)
{
    imprima "\n\nDigite ", L + 1, " nome: ";
    leia nomes[ L ] ;
    enquanto( strtam(nomes[L]) >30)
    { imprima "\nNomes com ate 30 caracteres";
}

```

```

imprima "\n\nDigite ", L + 1, " nome: ";
leia nomes[ L ] ;
}
# trecho que garante todos os nomes com 30 caracteres para fazer tabulação
t<-30-strtam(nomes[L]);
para(c<-1; c <= t; c++)
{nomes[L]<-strconcat(nomes[L], " ");}
imprima "\ndigit 1 nota: ";
leia pr1[L];
imprima "\ndigit 2 nota: ";
leia pr2[L];
media[L] <- realint((pr1[L] +pr2[L] )/2+0.0001);
se(media[ L ] >= 5 )
{ sit[ L ] <- "AP";   }
senao
{ sit[ L ] <- "RP";   }
}
# fim do trecho de entrada
#trecho de saída
imprima "\n\n\n\t\t\tRELACAO FINAL\n";
para( L<- 0; L <= 14 ; L++)
{ imprima "\n", L+1, "- ",nomes[L]," \t",pr1[L], " \t", pr2[L],
"\t",media[L], "\t",sit[L]; }
# fim do trecho de saída
imprima "\n";
fimprog

```

algoritmo 357

Armazenar nome e salário de 20 pessoas. Calcular e armazenar o novo salário sabendo-se que o reajuste foi de 8%. Imprimir uma listagem numerada com nome e novo salário.

```

prog vetor5
int L,c ,t;
string nomes[20];
real sal[20];
# trecho de entrada de 20 nomes
para( L<- 0; L <= 19 ; L++)
{
    imprima "\ndigit ", L + 1, " nome: ";
    leia nomes[ L ] ;
    enquanto( strtam(nomes[L]) >30)
    {imprima "\nNomes com até 30 caracteres";
     imprima "\n\nDigite ", L + 1, " nome: ";
     leia nomes[ L ] ;
    }
    t<-30-strtam(nomes[L]);
    para(c<-1; c <= t; c++)

```

```

{nomes[L]<-strconcat(nomes[L], " ");}

imprima "\ndigite salario: ";
leia sal[ L ];
sal[L] <-sal[L] * 1.08;
}

# fim do trecho de entrada
# trecho de saida
imprima "\nNOMES\t\t\t\t\t\t";
para( L<- 0; L <= 19 ; L++)
{ imprima "\n", L+1, "- ", nomes[ L ],"\t",sal[L]; }
# fim do trecho de saida
imprima "\n";
fimprog

```

algoritmo 358

Criar um algoritmo que leia o preço de compra e o preço de venda de 100 mercadorias. O algoritmo deverá imprimir quantas mercadorias proporcionam:

- $lucro < 10\%$
- $10\% \leq lucro \leq 20\%$
- $lucro > 20\%$

```

prog vetor6
real precocompra[100],precovenda[100],lucro;
int totlucromenor10,totlucromenor20,totlucromaior20, A;
totlucromenor10 <- 0;
totlucromenor20 <- 0;
totlucromaior20 <- 0;
para(A <- 0; A < 100; A++)
{ imprima "\nPreco de compra: "; leia precocompra [A];
  imprima "\nPreco de venda: "; leia precovenda [A];
}
para(A<-0;A < 100;A++)
{
  lucro <- precovenda[A] - precocompra[A] ;
  se(lucro<10.0)
  {totlucromenor10++; }
  senao
  { se(lucro <= 20.0)
    { totlucromenor20++; }
    senao
    { totlucromaior20++; }
  }
}
imprima "\ntotal de mercadorias com lucro < 10%: ",totlucromenor10;
imprima "\ntotal de mercadorias com 10% <= lucro <= 20%:
",totlucromenor20;
imprima "\ntotal de mercadorias com lucro > 20%: ",totlucromaior20;

```

```

imprima "\n";
fimprog

```

algoritmo 359

Criar o algoritmo que deixe entrar com nome e idade de 20 pessoas e armazene em um vetor todos os nomes que comecem pela letra do intervalo L - S.

```

prog vetor7
string nome[20], LS[20];
int i, c;
c <-0;
para(i <-0; i <= 19; i++)
{ imprima "\ndigite ", i + 1, " nome: ";
leia nome[i];
se(strprim(nome[i]) >= "L" && strprim(nome[i]) <= "S" ||
strprim(nome[i]) >= "l" && strprim(nome[i]) <= "s")
{ LS[c] <- nome[i]; c++; }
}
se(c < > 0)
{ imprima "\n\nRelacao dos nomes que comecam por letra no intervalo L - S\n";
para(i <-0; i <= c -1; i++)
{ imprima "\n",LS[i]; }
}
senao
{ imprima "\n\nNenhum nome encontrado"; }
imprima "\n";
fimprog

```

algoritmo 360

Criar um algoritmo que imprima o horóscopo de várias pessoas, a partir de sua data de nascimento (ddmm). O fim é determinado quando se digita 9999 para data; considere que a data foi digitada corretamente.

Mês	Último dia	Signo
01	20	Capricórnio
02	19	Aquário
03	20	Peixes
04	20	Áries
05	20	Touro
06	20	Gêmeos
07	21	Câncer
08	22	Leão
09	22	Virgem
10	22	Libra
11	21	Escorpião
12	21	Sagitário

```

prog vetor8
    int ultdia [ 12 ] , data, i, dia, mes;
    string signo[ 12 ] ;
    para( i<- 0; i< 12; i++)
    { imprima "\ndigite signo: ";
        leia signo[i];
        imprima "\ndigite ultimo dia: ";
        leia ultdia [ i ] ;
    }
    imprima "\ndigite data no formato ddmm ou 9999 para terminar: ";
    leia data;
    enquanto( data < > 9999)
    { dia <- data div 100;
        mes <- data % 100;
        mes--;
        se( dia > ultdia [ mes ] )
        { mes <- (mes + 1) % 12; }
        imprima "\nsigno: ",signo[ mes ], "\n" ;
        imprima "\ndigite data no formato ddmm ou 9999 para terminar: ";
        leia data;
    }
    imprima "\n";
fimprog

```

algoritmo 361

Armazenar código, nome, quantidade, valor de compra e valor de venda de 30 produtos. A listagem pode ser de todos os produtos ou somente de um ao se digitar o código.

```

prog vetor9
    int L,c,t,op,codigo[30],cod, qtde[30];
    string nomes[30];
    real vc[30], vv[30];
    # trecho de entrada de 30 produtos
    para( L<- 0; L <= 29 ; L++)
    {
        imprima "\ndigite ", L + 1, " nome do produto: ";
        leia nomes[ L ] ;
        enquanto( strtam(nomes[L]) >20)
        {imprima "\nNomes com ate 20 caracteres";
            imprima "\n\nDigite ", L + 1, " nome: ";
            leia nomes[ L ] ;
        }
        t<-20-strtam(nomes[L]);
        para(c<-1; c <= t; c++)
        {nomes[L]<-strconcat(nomes[L], " ")};
        imprima "\ndigite codigo do produto: ";
        leia codigo[L] ;
        imprima "\ndigite quantidade: ";
        leia qtde[L] ;
    }

```

```

imprima "\ndigite valor de compra: ";
leia vc[L] ;
imprima "\ndigite valor de venda: ";
leia vv[L] ;
}
# fim do trecho de entrada
# trecho de saida
faca
{
    imprima "\n\n\nMenu\n";
    imprima "\n1-Todos os produtos";
    imprima "\n2-So um produto";
    imprima "\n3-Sair";
    imprima "\nOPCAO: ";
    leia op;
    se(op==1)
    { imprima "\nCodigo\tNome\t\t\tQtde\tValor-compra\tValor-venda";
        para( L<- 0; L <= 29; L++)
        {
            imprima "\n", codigo[L], "\t", nomes[L], "\t", qtde[L], "\t", vc[L],
            "\t\t", vv[L]; # continuacao
        }
    }
    senao
    {se(op==2)
     { c<-0;
       imprima "\nDigite codigo do produto: ";
       leia cod;
       enquanto(cod < >codigo[c] && c<29)
       {c++;}
       se(cod==codigo[c])
       {imprima "\n\n", codigo[c], "\t", nomes[c], "\t", qtde[c], "\t",
        vc[c], "\t", vv[c];} # continuacao
       senao
       { imprima "\n\nCodigo Inexistente";}
     }
     senao
     {se(op==3)
      {saia;}
     senao
     {imprima "\nOpcao Invalida";}}}
    }
    enquanto(op< >3)
# fim do trecho de saida
imprima "\n";
fimprog

```

algoritmo 362

284 Criar um algoritmo que leia dois conjuntos de números inteiros, tendo cada um 10 e 20 elementos e apresente os elementos comuns aos conjuntos. Lem-

bre-se de que os elementos podem se repetir mas não podem aparecer repetidos na saída.

```
prog vetor10
    int vet1[10], vet2[20], vetc[10], i, c, d, L;
    L <- 0;
    imprima "\nentrada de dados do vetor 1 ( 10 elementos)";
    para(i <-0 ; i <= 9; i++)
    { imprima "\nentre com ", i + 1, " elemento: ";
        leia vet1[i];
    }
    imprima "\nentrada de dados do vetor 2 ( 20 elementos)";
    para(i <-0 ; i <= 19; i++)
    { imprima "\nentre com ", i + 1, " elemento: ";
        leia vet2[i];
    }
    para(i <-0 ; i <= 9; i++)
    { vetc[i] <- -999999999; } # inicializando com valores fora do contexto
    para(i <-0 ; i <= 9; i++)
    { c <- 0;
        enquanto(vet1[i] < > vet2[c] && c < 19)
        { c++; }
        se( vet1[i] == vet2[c] )
        { d <- 0;
            enquanto(vet1[i] < > vetc[d] && d < L)
            {d++; }
            se( d == L)
            { vetc[d] <- vet1[i]; L++; }
        }
    }
    se( L < > 0)
    {
        imprima "\n\nElementos comuns\n\n";
        para(i <-0 ; i <= L -1 ; i++)
        { imprima "\n", vetc[i]; }
    }
    senao
    { imprima "\n\nNao existem elementos comuns"; }
    imprima "\n";
fimprog
```

algoritmo 363

Criar um algoritmo que leia vários números inteiros e positivos. A leitura se encerra quando encontrar um número negativo ou quando o vetor ficar completo. Saber-se que o vetor possui, no máximo, 10 elementos. Gerar e imprimir um vetor onde cada elemento é o inverso do correspondente do vetor original.

```
prog vetor11
    int cont,L;
    real num, v[10], v1[10];
```

```

cont <-0;
imprima "\ndigite numero positivo ou 0. para terminar: ";
leia num;
enquanto( cont < 10 && num > 0. )
{
    v[cont] <- num;
    v1[cont] <- 1 / num ;
    cont++;
    imprima "\ndigite numero positivo ou 0. para terminar: ";
    leia num;
}
se( cont == 0 )
{   imprima "\nVetor nao tem dados"; }
senao
{   cont--;
    para(L <- 0; L <= cont; L++)
    {   imprima "\n", v1[L]; }
}
imprima "\n";
fimprog

```

algoritmo 364

Ler um vetor vet de 10 elementos e obter um vetor w cujos componentes são os fatoriais dos respectivos componentes de v.

```

prog vetor12
int vet[10], w[10], L, f, fat;
para(L <- 0; L <10 ; L++)
{   imprima "\ndigite elemento ", L + 1,": ";
    leia vet[L];
    w[L] <-1;
    para(f<-2 ; f <= vet[L]; f++)
    {   w[L] <- w[L] * f;}
}
imprima "\nVetor fatorial\n";
para(L <- 0; L <10 ; L++)
{   imprima "\n", w[L]; }
imprima "\n";
fimprog

```

algoritmo 365

Uma pessoa muito organizada gostaria de fazer um algoritmo para armazenar os seguintes dados de um talonário após a utilização total do mesmo: nº do cheque, valor, data e destino. Sabendo-se que o número de cheques pode ser variável e não ultrapassa 20, construa esse algoritmo de tal maneira que possa gerar um relatório no vídeo.

```

prog vetor13
  string num[20], valor[20], destino[20], data[20], resp;
  int nc, k;
  imprima "\nnumero de cheques do talonario: ";
  leia nc;
  para( k <- 0; k < nc; k++)
  { imprima "\nnumero do cheque: ";
    leia num[k];
    imprima "\nvalor do cheque: ";
    leia valor[k];
    imprima "\ndata do cheque ddmmaa: ";
    leia data[k];
    imprima "\ndestino do cheque: ";
    leia destino[k];
  }
  imprima "\nRELACAO dos CHEQUES\n";
  para( k <-0; k < nc; k++)
  { imprima "\nNumero do cheque: ", num[k];
    imprima "\nValor do cheque: ", valor[k];
    imprima "\nData do cheque: ", data[k];
    imprima "\nDestino do cheque: ", destino[k];
    imprima "\n\nPressione enter para ver outro cheque: ";
    leia resp; # necessario pois a tela so tem 25 linhas
  }
  imprima "\n";
fimprog

```

algoritmo 366

*Criar um algoritmo que armazene em dois vetores nome e profissão de 20 pessoas.
Deverá sair uma listagem, no vídeo, que tenha a seguinte forma:*

<i>1^a coluna</i>	<i>41^a coluna</i>
<i>NOME</i>	<i>PROFISSAO</i>
<i>1- ...</i>	<i>...</i>
<i>20- ...</i>	<i>...</i>

Total de dentistas: --

```

prog vetor14
  string nomes[20], prof[20];
  int L, tam, c, dent;
  dent <- 0;
  para( L<- 0; L < 20 ; L++)
  { imprima "\nDigite ", L+1, " nome (com ate 30 caracteres ): " ;
    leia nomes[L] ;
    enquanto(strtam(nomes[L] ) > 30)
    { imprima "\nNome com ate 30 caracteres. Digite ", L+1, " nome: ";

```

```

"b=0", k +1, "elemento do vetor 2: ", v2[k], "b=b", v4[k];
}
imprima "\n";
fimprog

```

algoritmo 369

Criar um algoritmo para gerenciar um sistema de reservas de mesas em uma casa de espetáculo.

A casa possui 30 mesas de 5 lugares cada. O algoritmo deverá permitir que o usuário escolha código de uma mesa (100 a 129) e forneça a quantidade de lugares desejados. O algoritmo deverá informar se foi possível realizar a reserva e atualizar a reserva. Se não for possível, o algoritmo deverá emitir uma mensagem. O algoritmo deve terminar quando o usuário digitar o código 0 (zero) para uma mesa ou quando todos os 150 lugares estiverem ocupados.

```

prog vetor17
int mesa[30], i, qtde[30], lugares, codigo, lugmesa;
para(i <- 0; i < 30 ;i++)
{ mesa[i] <- 100 + i;
  qtde[i] <- 5;
}
lugares <- 150;
imprima "\nentre com codigo (100 - 129) ou 0 para terminar: ";
leia codigo;
enquanto(codigo > 0 && lugares < > 0)
{ i <- 0;
  enquanto(codigo < > mesa[i] && i < 29)
  { i++; }
  se( codigo == mesa[i] )
  { imprima "\nquantidade de lugares a reservar: ";
    leia lugmesa;
    se( qtde[i] >= lugmesa)
    { qtde[i] <- qtde[i] - lugmesa; lugares <- lugares - lugmesa; }
    senao
    { imprima "\nnao ha lugares a reservar"; }
  }
  senao
  { imprima "\ncodigo de mesa invalido"; }
  imprima "\nentre com codigo (100 - 129) ou 0 para terminar: ";
  leia codigo;
}
se(lugares == 0)
{imprima "\nLotacao esgotada";}
senao
{ imprima "\nLugares vagos\n";
  para(i <- 0; i < 30 ;i++)
  { se(qtde[i]< >0)
    {imprima "\nmesa: ", mesa[i], " total de lugares: ", qtde[i];
    }
  }
}

```

```

    }
}

imprima "\n";
fimprog

```

algoritmo 370

Criar um algoritmo que realize as reservas de passagem aéreas de uma companhia. Além da leitura do número de vôos e da quantidade de lugares disponíveis, leia vários pedidos de reserva, constituídos do número da carteira de identidade do cliente e do número do vôo desejado.

Para cada cliente, verificar se há possibilidade no vôo desejado. Em caso afirmativo, imprimir o número da identidade do cliente e o número do vôo, atualizando o número de lugares disponíveis. Caso contrário, avisar ao cliente a inexistência de lugares.

```

prog vetor18
int nv, i, voos[1000];# numero superestimado
string nome[1000], id, nvd;
imprima "\nEntre com o numero de voos:";
leia nv;
para (i <- 0; i < nv; i++)
{ imprima "\nEntre com a identificacao do vo ", i + 1, " : ";
  leia nome[i];
  imprima "\nEntre com a quantidade de lugares disponiveis no voo ",
  nome[i], " : ";
  leia voos[i];
}
imprima "\nEntre com o numero do voo desejado ou 0 para terminar: ";
leia nvd;
enquanto( nvd < > "@")
{ i <- 0;
  enquanto(nvd < > nome[i] && i < nv-1)
  {i++;}
  se(nome[i] == nvd)
  { se(voos[i] > 0)
    { voos[i]--;
      imprima "\nQual o numero da identidade do cliente? ";
      leia id;
      imprima "\nIdentidade:", id, " Voo: ", nvd, "\n";
    }
    senao
    { imprima "\nNao existem mais lugares neste voo.\n"; }
  }
  senao
  { imprima "\nNao existe este voo\n"; }
  imprima "\n\nEntre com o numero do voo desejado ou 0 para terminar: ";
  leia nvd;
}
imprima "\n";
fimprog

```

algoritmo 371

Criar um algoritmo que leia dois conjuntos de números inteiros, tendo cada um 10 e 20 elementos e apresentar os elementos que não são comuns aos dois conjuntos.

```
prog vetor19
    int vet1[10], vet2[20], vetc[30], i, c, d, L;
    L <- 0;
    imprima "\nentrada de dados do vetor 1 ( 10 elementos)";
    para(i <-0 ; i <= 9; i++)
    { imprima "\nentre com ", i + 1, " elemento: ";
        leia vet1[i];
    }
    imprima "\nentrada de dados do vetor 2 ( 20 elementos)";
    para(i <-0 ; i <= 19; i++)
    { imprima "\nentre com ", i + 1, " elemento: ";
        leia vet2[i];
    }
    para(i <-0 ; i <= 29; i++)
    { vetc[i] <- -999999999; } # inicializado com valores fora do contexto
    para(i <-0 ; i <= 9; i++)
    { c <- 0;
        enquanto(vet1[i] < > vet2[c] && c < 19)
        { c++; }
        se( vet1[i] < > vet2[c])
        { d <- 0;
            enquanto(vet1[i] < > vetc[d] && d < L)
            {d++; }
            se( d == L)
            { vetc[d] <- vet1[i]; L++; }
        }
    }
    para(i <-0 ; i <= 19; i++)
    { c <- 0;
        enquanto(vet2[i] < > vet1[c] && c < 9)
        { c++; }
        se( vet2[i] < > vet1[c])
        { d <- 0;
            enquanto(vet2[i] < > vetc[d] && d < L)
            {d++; }
            se( d == L)
            { vetc[d] <- vet2[i]; L++; }
        }
    }
    se( L < > 0)
    {
        imprima "\n\nElementos nao comuns";
```

```

    para(i < 0 ; i <= L - 1 ; i++)
    { imprima "\n", vetc[i]; }
}
senao
{ imprima "\n\nNao existem elementos nao comuns" ;}
imprima "\n";
fimprog

```

algoritmo 372

Num torneio de futsal, rodada simples, inscreveram-se 12 times. Armazenar os nomes dos times e imprimir a tabela de jogos.

```

prog vetor20
int L, c,t;
string times[12];
#trecho de entrada
para( L<- 0;L <= 11; L++)
{
    imprima "\n\nDigite ", L + 1, " time: ";
    leia times[L];
enquanto( strtam(times[L]) >20)
    {imprima "\nNomes com ate 20 caracteres";
     imprima "\n\nDigite ", L + 1, " time: ";
     leia times[ L ] ;
    }
    t<-20-strtam(times[L]);
    para(c<-1; c <= t; c++)
    {times[L]<-strconcat(times[L], "b");}
}
imprima "\n Rodada Simples";
para( L<- 0;L <= 10; L++)
{
    para( c<- L +1;c <= 11; c++)
    {
        imprima "\n", times[L ],"\t",times[c ];
    }
}
imprima "\n";
fimprog

```

algoritmo 373

Entrar com nomes de cinco times de futebol e armazená-los em um vetor de nome TIMES. Imprimir uma tabela para rodada dupla.

```

prog vetor21
int L, c,t;
string times[5];
para( L<- 0;L <= 4; L++)

```

```

{
    imprima "\ndigite nome do time:";
    leia times[L];
    enquanto( strtam( times[L] ) >20 )
    { imprima "\nNomes com ate 20 caracteres";
        imprima "\n\nDigite ", L + 1, " time: ";
        leia times[ L ];
    }
    t<-20-strtam(times[L]);
    para(c<-1; c <= t; c++)
    {times[L]<-strconcat(times[L], "b");}
}
imprima "\n Rodada Dupla";
para( L<- 0;L <= 4; L++)
{
    para( c<- 0;c <= 4; c++)
    {
        se(L < > c)
        {   imprima "\n", times[L ],"\t",times[ c ]; }
    }
}
imprima "\n";
fimprog

```

algoritmo 374

Entrar com números inteiros em um vetor A [50]. Gerar e imprimir o vetor B onde cada elemento é o quadrado do elemento, na respectiva posição, do vetor A.

```

prog vetor22
int L,a[50], b[50];
#trecho de entrada para 50 numeros
para( L<- 0;L <= 49 ; L++)
{
    imprima "\ndigite numero :";
    leia a[L];
    b[L] <- a[L] ^ 2;
}
#trecho de saida
imprima "\nOriginal\tQuadrado";
para( L<- 0;L <= 49; L++)
{   imprima "\n",a[L], "\t\t",b [L] ;}
imprima "\n";
fimprog

```

algoritmo 375

Entrar com números reais para dois vetores A e B de dez elementos cada. Gerar e imprimir o vetor diferença.

```

prog vetor23
    int L;
    real a[30], b[30], dif[30];
    #trecho de entrada para 30 numeros
    para( L<- 0; L <= 29; L++)
    {
        imprima "\ndigite elemento ",L+1, " do vetor A:";
        leia a[L];
    }
    imprima "\n\n";
    para( L<- 0; L <= 29 ; L++)
    {
        imprima "\ndigite elemento ",L+1, " do vetor B:";
        leia b[L];
        #Gera o vetor diferença
        dif [L] <- a[L] - b[L];
    }
    #trecho de saída
    imprima "\nVetor A\t\tVetor B\t\tVetor Diferença";
    para( L<- 0; L <= 29; L++)
    {
        imprima "\n",a[L], "\t\t",b [L], "\t\t", dif[L];
    }
    imprima "\n";
fimprog

```

algoritmo 376

Criar um algoritmo que leia um vetor A de dez valores e construa outro vetor B, da seguinte forma:

<i>Ex.: Vetor A</i>	3	8	4	2	...	5
<i>Vetor B</i>	9	4	12	1	...	15

```

prog vetor24
    int L ;
    real a[10], b[10];
    #vetor com 10 elementos
    para( L<- 0; L <= 9 ; L++)
    {
        imprima "\ndigite numero :";
        leia a[L];
        se(L % 2 == 0)
        { b[L] <- a[L] / 2; }
        senao
        { b[L] <- a[L] * 3; }
    }

```

```

imprima "\n\n Vetores\n";
para( L<- 0; L <= 9; L++)
{
    imprima a [L], " ";
}
imprima "\n";
para( L<- 0; L <= 9; L++)
{
    imprima b [L], " ";
}
imprima "\n";
fimprog

```

algoritmo 377

Criar um algoritmo que leia dois vetores A e B, contendo, cada um, 25 elementos inteiros. Intercalle esses dois conjuntos (A[1] / B[1] / A[2] / B[2] /..), formando um vetor V de 50 elementos. Ordene de forma decrescente. Imprima o vetor V.

```

prog vetor25
int L, a[25], b[25], inter[50];
#25 elementos cada
para( L<- 0; L <= 24; L++)
{
    imprima "\ndigite ", L+1, " elemento do vetor A :";
    leia a[L];
}
para( L<- 0; L <= 24; L++)
{
    imprima "\ndigite ", L+1, " elemento do vetor B :";
    leia b[L];
    #Gera o vetor intercalado
    inter[2 * L] <- a[L];
    inter[2 * L + 1] <- b[L];
}
imprima "\nVetor Intercalado";
para( L<- 0; L <= 49; L++)
{
    imprima "\n", inter[L];
}
imprima "\n";
fimprog

```

algoritmo 378

Entrar com vários números, até digitar o número 0. Imprimir quantos números iguais ao último número foram lidos. O limite de números é 100.

```

prog vetor26
int L, c, d, num[100];
# trecho de entrada de 100
L<- 0;
imprima "\ndigite ", L + 1, " numero ou 0 para terminar: ";

```

```

leia num[ L ] ;
enquanto(L<=98 && num[L]< >0)
{ L++;
  imprima "\ndigite ", L + 1, " numero ou 0 para terminar: ";
  leia num[ L ] ;
}
# fim do trecho de entrada
se(num[L] ==0)
{L--;}
c<-0;
para(d<-0; d<=L-1;d++)
{se(num[d] ==num[L])
 {c++;}
}
# trecho de saida
imprima "\nTotal de numeros iguais ao ultimo: ", c;
# fim do trecho de saida
imprima "\n";
fimprog

```

algoritmo 379

Fazer um algoritmo para ler um conjunto de 100 números reais e informar:

1. quantos números lidos são iguais a 30
2. quantos são maior que a média
3. quantos são iguais à média

```

prog vetor27
  real num[100],soma, media;
  int cm,cmm, c30, L;
  soma <-0.;
  c30 <-0;
  para(L <-0;L<=99;L++)
  {
    imprima "\ndigite numero: ";
    leia num[L];
    soma <-soma +num[L];
    se(num[L]==30.)
    {c30++;}
  }
  media <-soma/100;
  cm <-0;
  cmm <-0;
  para(L <-0;L<=99;L++)
  {
    se(num[L] > media)
    {cmm++;}
    senao
  }

```

```

{se(num[L]==media)
{cm++;}}
}
imprima "\nTotal iguais a 30: ",c30;
imprima "\nTotal iguais a media: ",cm;
imprima "\nTotal maiores que a media: ",cmm;
imprima "\n";
fimprog

```

algoritmo 380

Criar um algoritmo que leia um conjunto de 30 valores inteiros, armazene-os em um vetor e escreva-os ao contrário da ordem de leitura.

```

prog vetor28
int L, num[30];
para(L<-0; L<=29;L++)
{imprima "\ndigite numero: ";
leia num[L];
}
imprima "\n\n\n";
imprima "\n Vetor ao Contrario\n";
para(L<-29; L>=0;L--)
{imprima "\n",num[L];}
imprima "\n";
fimprog

```

algoritmo 381

Armazenar dez nomes em um vetor NOME e imprimir uma listagem numerada e ordenada.

```

prog vetor29
int L, c;
string nomes[10], aux;
para( L<- 0; L <= 9 ; L++)
{
    imprima "\n nome: " ;
    leia nomes[L] ;
}
para( L<- 0; L <= 8 ; L++)
{
    para( c<- L + 1; c <= 9 ; c++)
    {
        se ( nomes[L] > nomes[c] )
        {
            aux <- nomes[L];
            nomes[L] <- nomes[c];
            nomes[c] <- aux;
        }
    }
}

```

```

        }
    }
}

imprima "\n\n\n";
imprima "\n Relação dos nomes ordenados\n " ;
para( L<- 0; L <= 9 ; L++)
{
    imprima "\n", L +1, " - ", nomes[L] ;
}
imprima "\n";
fimprog

```

algoritmo 382

Entrar com dados para um vetor VET do tipo inteiro com 20 posições, onde podem existir vários elementos repetidos. Gere o vetor VET1 que também será ordenado e terá somente os elementos do vetor VET que não são repetidos.

```

prog vetor30
int L, c,vet[20], vet1[20], aux;
para( L<- 0; L <= 19 ; L++)
{
    imprima "\n numero: " ;
    leia vet[L] ;
}
para( L<- 0; L <= 18 ; L++)
{
    para( c<- L + 1; c <= 19 ; c++)
    { se ( vet[L] > vet[c] )
        { aux <- vet[L]; vet[L] <- vet[c] ; vet[c] <- aux; }
    }
}
imprima "\n\n\n";
imprima "\n Relação dos numeros ordenados\n " ;
para( L<- 0; L <= 19 ; L++)
{ imprima "\n", L +1, " - ", vet[L] ;}
vet1[0]<- vet[0];
c<-1;
para( L<- 1; L <= 19 ; L++)
{ se(vet[L] < > vet1[c-1])
    { vet1[c] <- vet[L]; c++; }
}
imprima "\n\n\n";
imprima "\n Relação dos numeros nao repetidos\n " ;
para( L<- 0; L <= c-1 ; L++)
{ imprima "\n", L +1, " - ", vet1[L] ;}
imprima "\n";
fimprog

```

algoritmo 383

Criar um algoritmo que leia os elementos de um vetor com 20 posições e escreva-o. Em seguida, troque o primeiro elemento pelo último, o segundo pelo penúltimo, o terceiro pelo antepenúltimo, e assim sucessivamente. Mostre o vetor depois das trocas.

```
prog vetor31
    int L, c,vet[20], aux;
    para( L<- 0; L <= 19 ; L++)
    { imprima "\n numero: " ; leia vet[L] ;}
    para( L<- 0; L <= 9 ; L++)
    { aux <- vet[L]; vet[L] <- vet[19-L]; vet[19-L] <- aux; }
    imprima "\n\n\n";
    imprima "\nVetor trocado\n" ;
    para( L<- 0; L <= 19 ; L++)
    { imprima "\n", L +1, " - ",vet[L];}
    imprima "\n";
fimprog
```

algoritmo 384

Em um concurso público inscreveram-se 5.000 candidatos para 100 vagas. Cada candidato fez 3 provas, tendo cada uma pesos 2, 3 e 5 respectivamente, na ordem em que foram feitas. Fazer um algoritmo que leia nome, matrícula e os pontos obtidos pelos candidatos em cada prova; apresentar a classificação, a matrícula e o nome dos candidatos aprovados, ordenados pela classificação.

```
prog vetor32
    int L, c,tam,p1[5000], p2[5000],p3[5000], pontos[5000], mat[5000],auxn;
    string nomes[5000], auxc;
    para( L<- 0; L <= 4999 ; L++)
    {
        imprima "\n nome: " ;
        leia nomes[L] ;
        enquanto(strtam(nomes[L])>30)
        { imprima "\nNomes com ate 30 caracteres. Digite nome: ";
            leia nomes[L];}
        tam <-strtam(nomes[L]);
        se(tam<30)
        { para(c<-0;c<=30 - tam; c++)
            { nomes[L]<-strconcat(nomes[L], "b");}
        imprima "\n matricula: " ; leia mat[L] ;
        imprima "\n pontos da 1 prova: " ; leia p1[L] ;
        imprima "\n pontos da 2 prova: " ; leia p2[L] ;
        imprima "\n pontos da 3 prova: " ; leia p3[L] ;
        pontos[L]<-p1[L]+p2[L]+p3[L];
    }
}
```

```

para( L<- 0; L <= 4998 ; L++)
{
  para( c<- L + 1; c <= 4999 ; c++)
  {
    se ( pontos[L] < pontos[c] )
    {
      auxn <- pontos[L];
      pontos[L] <- pontos[c];
      pontos[c] <- auxn;
      auxn <- p1[L];
      p1[L] <- p1[c];
      p1[c] <- auxn;
      auxn <- p2[L];
      p2[L] <- p2[c];
      p2[c] <- auxn;
      auxn <- p3[L];
      p3[L] <- p3[c];
      p3[c] <- auxn;
      auxn <- mat[L];
      mat[L] <- mat[c];
      mat[c] <- auxn;
      auxc <- nomes[L];
      nomes[L] <- nomes[c];
      nomes[c] <- auxc;
    }
  }
  imprima "\n\n\n";
  imprima "\nCLAS.\tMAT\tNOME\t\t\t\tP1\tP2\tP3\tSOMA\n" ;
  para( L<- 0; L <= 99 ; L++)
  {
    imprima "\n",L +1," \t",mat[L], "\t",nomes[L]," \t", p1[L], "\t", p2[L],
    "\t", p3[L], "\t",pontos[L];
  }
  imprima "\n";
fimprog

```

algoritmo 385

No vestibular de uma universidade, no curso de Informática, inscreveram-se 1.200 pessoas. Criar um algoritmo que leia o gabarito da prova que tinha 100 questões, sendo o valor de cada questão igual a 1 ponto. Exiba o número de inscrição, o nome e as 100 respostas de cada candidato. O algoritmo deverá imprimir: o número de inscrição, o nome e a nota de cada candidato.

↳ Só é necessário guardar a soma dos pontos de cada candidato, o número de inscrição e o nome.

```

prog vetor33
int L,c, soma,tam, insc[1200],pontos[100];
string nomes[1200], gab[100], resp;
para(L<-0;L<=99;L++)
{imprima "\nDigite gabarito (a/b/c/d/e) da questao ", L+1,": ";
leia gab[L];}
para(L<-0;L<=1199;L++)
{imprima "\nDigite numero de inscricao do candidato ", L+1, " : ";
leia insc[L];
imprima "\nDigite nome do candidato: "; leia nomes[L];
enquanto(strtam(nomes[L]) > 30)
{imprima "\nNomes com ate 30 caracteres. Digite nome: ";
leia nomes[L];}
tam <-strtam(nomes[L] );
se(tam<30)
{para(c<-0;c<=30 - tam; c++)
{nomes[L]<-strconcat(nomes[L], "b");}
}
soma <-0;
para(c<-0;c<=99;c++)
{imprima "\nDigite resposta da ", c+1," questao: ";
leia resp;
se(resp==gab[c])
{ soma++;}
}
pontos[L]<-soma;
}
imprima "\n\n\n";
imprima "\nINSC.\tNOME\t\t\t\tPONTOS\n";
para(L<-0;L<=1199;L++)
{imprima "\n",insc[L], "\t",nomes[L]," \t",pontos[L];}
imprima "\n";
fimprog

```

algoritmo 386

Suponha três vetores de 30 elementos cada, contendo: nome, endereço, telefone. Fazer um trecho que se possa buscar pelo nome e imprimir todos os dados.

```

prog vetor34
int L ;
string nome, nomes [30], end[30], tel[30];
para( L<- 0; L <= 29 ; L++)
{
    imprima "\n nome ", L+1, " : " ; leia nomes[L] ;
    imprima "\n endereco: " ; leia end[L] ;
    imprima "\n telefone: " ; leia tel[L] ;
}

```

```

# entra com nome para procura
imprima "\n Digite nome para procura: " ;
leia nome;
L <-0;
enquanto ( L<29 && nome< >nomes[L] ) 
{ L++; }
se (nome ==nomes[L] )
{
    imprima "\nnome : ", nomes[L];
    imprima "\nendereço: ", end[L];
    imprima "\ntelefone: ", tel[L];
}
senao
{ imprima "\nnao encontrei" ; }
imprima "\n";
fimprog

```

algoritmo 387

Fazer um algoritmo que leia a matrícula e a média de 1.000 alunos. Ordene da maior nota para a menor e imprima uma relação contendo todas as matrículas e médias.

```

prog vetor35
int L, c, mat[1000], auxi ;
real media[1000], auxr;
para( L<- 0; L <= 999 ; L++)
{
    imprima "\n matricula do aluno ",L+1," : " ;
    leia mat[L] ;
    imprima "\n media: " ;
    leia media[L] ;
}
para( L<- 0; L <= 998 ; L++)
{
    para( c<- L + 1; c <= 999; c++)
    {
        se ( media[L] < media[c] )
        {
            auxr <- media[L];
            media[L] <- media[c];
            media[c] <- auxr;
            auxi <- mat[L];
            mat[L] <- mat[c];
            mat[c] <- auxi;
        }
    }
}

```

```

imprima "\nORDENADOS\n";
imprima "\nMATRICULA\tMEDIA\n" ;
para( L<- 0; L <= 999 ; L++)
{   imprima "\n", mat[L], "\t\t", media[L] ;}
imprima "\n";
fimprog

```

algoritmo 388

Criar um algoritmo que armazene 10 números em um vetor. Na entrada de dados, o número já deverá ser armazenado na sua posição definitiva em ordem decrescente. Imprimir o vetor logo após a entrada de dados.

Exemplo:

entrada	vetor	entrada	vetor	entrada	vetor	entrada	vetor
7	7	7	9	7	9	12	12
		9	7		7	9	9
				2	2	7	7
						12	2
					

```

prog vetor36
int L, c,d,n, a[10];
para( L<- 0; L <= 9 ; L++)
{a[L]<-0; }
para( L<- 0; L <= 9 ; L++)
{
    imprima "elemento do vetor A[", L + 1, "]: "; leia n;
    c<-0;
    enquanto(n<=a[c])
    {c++;}
    se(L>0)
    {para(d<-L;d>=c+1;d--)
        {a[d]<-a[d-1];}
    }
    a[c]<-n;
}
imprima "\n";
imprima "\n\n\nVetor Ordenado\n";
para( L<- 0; L <= 9 ; L++)
{
    imprima "\na[",L + 1, "]- ", a[L];
}
imprima "\n";
fimprog

```

algoritmo 389

Criar um algoritmo que receba a temperatura média de cada mês do ano, em centígrados, e armazene essas temperaturas em um vetor; imprimir as temperaturas de todos os meses, a maior e a menor temperatura do ano e em que mês aconteceram.

```
prog vetor37
    int L, maiorM, menorM;
    real temp[12];
    maiorM <- 0;
    menorM <- 0;
    para( L<- 0; L <= 11 ; L++)
    {
        imprima "\n temperatura do mes ", L+1,": " ;
        leia temp[L] ;
        se ( temp[L] > temp[maiorM] )
            { maiorM <- L; }
        senao
        {
            se ( temp[L] < temp[menorM] )
                { menorM <- L; }
        }
    }
    imprima "\n\n\n";
    imprima "\n Relação das temperaturas\n " ;
    para( L<- 0; L < 12 ; L++)
    {
        imprima "\nmes: ", L +1, " - temperatura ", temp[L] ;
        imprima "\nMAIOR TEMPERATURA - ", temp[maiorM], " no mes: ", maiorM + 1;
        imprima "\nMENOR TEMPERATURA - ", temp[menorM], " no mes: ", menorM + 1;
        imprima "\n";
    }
fimprog
```

algoritmo 390

Ler nome, CPF e profissão de 100 pessoas. Imprimir qual(ais) a(s) profissão(ões) que mais se repete(m) e quantas pessoas têm essa(s) profissão(ões).

```
prog vetor38
    int L,c,x, conta[100];
    string nome[100],cpf[100],prof[100],cprof[100];
    para( L<-0; L <= 99;L++)
    {
        imprima "\ndigite nome: "; leia nome[L];
        imprima "\ndigite CPF: "; leia cpf[L];
        imprima "\ndigite profissao somente no sexo masculino: ";
        leia prof[L];
```

```

}

/*impressao das profissoes de todas as pessoas so para entendimento*/
imprima "\n";
para( L<-0;L <= 99;L++)
{   imprima "\n",L+1," - ",prof[L]; }
/*inicializa os vetores */
para( L<-0;L <= 99; L++)
{   conta[L] <-0; cprof[L]<"";
cprof[0] <- prof[0];
conta[0]++;
c <-1;
para( L <- 1;L <= 99;L++)
{
x <-0;
enquanto(prof[L] < > cprof[x]  &&  x < c)
{ x++; }
se( x == c)
{ cprof[c] <- prof[L];  conta[c]++;  c++;}
senao
{conta[x]++;}
}
imprima "\n";
/*impressao das profissoes so para entendimento*/
imprima "\nPROFISSOES(A0) ENCONTRADA(S)";
para( L <-0;L <= c-1; L++)
{
    imprima "\n",L+1, " - ",cprof[L];
}
/*descobre o maior*/
imprima "\n";
x <- 0;
para(L <- 1;L <= c-1; L++)
{
    se( conta[L] > conta[x])
    { x <- L; }
}
/*imprime os maiores*/
imprima "\nPROFISSAO(OES) E QUANTIDADE(S)";
para( L <- 0;L <= c-1; L++)
{
    se(conta[L] == conta[x])
    {imprima "\n", cprof[L]," - ",conta[L];}
}
imprima "\n";
fimprog

```

algoritmo 391

Criar um algoritmo que leia um vetor de 30 números inteiros e imprima o número de elementos da maior sublista ordenada crescentemente.

Exemplo 1: \ 8 \ 9 \ 1 \ 7 \ 8 \ 17 \ 3 \

Maior sublista: 1 \ 7 \ 8 \ 17 \ logo o tamanho é 4.

Exemplo 2: \ 18 \ 9 \ 5 \ 3 \ 1

Maior sublista: não existe mais de um elemento, logo o tamanho é 1.

```
prog vetor39
    int M[30], L, C, Y;
    para( L <- 0; L <= 29 ; L++ )
    {   imprima "\nDigite o elemento ", L + 1,": ";
        leia M[L];
        Y <- 1;
        C <- 0;
        para( L <- 0; L <= 28 ; L++ )
        {
            se( M[L + 1] > M[L] )
            {   Y ++; }
            senao
            {
                se( Y > C )
                {   C <- Y; }
                Y <- 1;
            }
        }
        se( C > Y )
        {   imprima "\ntotal: ", C; }
        senao
        {   imprima "\ntotal: ", Y; }
        imprima "\n";
    fimprog
```

algoritmo 392

Criar um algoritmo que implemente o seguinte menu de opções:

- Ler notas e nomes de 100 candidatos;
- Exibir média geral de todos os candidatos;
- Exibir uma lista com o nome e a nota de todos os candidatos em ordem decrescente de nota;
- Ler um nome e buscar esse candidato imprimindo o nome e sua nota; caso não seja encontrado candidato com o nome lido, imprimir a mensagem "Candidato não encontrado".

```
prog vetor40
    string nomes[100], auxnome, nomep;
```

```

real notas[100], mediag, auxnota;
int opcao, c, L;
faca
{
    imprima "\n\n\n\t MENU";
    imprima "\n 1 - Ler Nomes e Notas";
    imprima "\n 2 - Media Geral";
    imprima "\n 3 - Classificacao";
    imprima "\n 4 - Busca Candidato";
    imprima "\n 5 - Sair";
    imprima "\n\t OPCAO: ";
    leia opcao;
    se(opcao == 1)
    { para(c <- 0 ; c <= 99; c++)
        { imprima "\n Digite o nome: "; leia nomes[c];
         imprima "\n Digite a nota com ponto: "; leia notas[c];
        }
    }
    senao
    { se(opcao == 2)
        { mediag <- 0.;
         para(c <- 0; c <= 99; c++)
         { mediag <- mediag + notas[c];
         }
         imprima "\n Media Geral = ", mediag/100;
        }
    }
    senao
    { se(opcao == 3)
        { para( L<- 0; L <= 98 ; L++)
            { para( c<- L + 1; c <= 99 ; c++)
                { se( notas[L] < notas[c] )
                    { auxnota <- notas[L];
                     notas[L] <- notas[c];
                     notas[c] <- auxnota;
                     auxnome <- nomes[L];
                     nomes[L] <- nomes[c];
                     nomes[c] <- auxnome;
                    }
                }
            }
            imprima "\n Relacao em ordem decrescente de nota";
            para( c<- 0; c <= 99 ; c++)
            { imprima "\n", c+1, " - Nome: ", nomes[c], "\t Nota: ",
              notas[c];
            }
        }
    }
    senao
    { se(opcao == 4)
        { imprima "\n Digite o nome a ser procurado"; leia nomep;
    }
}

```

```

L <- 0;
enquanto( L < 99 && nomep < > nomes[L] )
{ L <- L + 1;
se(nomep ==nomes[L])
{ imprima "\n NOME: ", nomes[L], "\nNOTA:", notas[L];
senao
{ imprima "\n Nao encontrei" ;
imprima "\n";
}
senao
{ se(opcao <> 5 )
{ imprima "\n Opcao Invalida";
}
}
}
}
}
enquanto(opcao<>5)
imprima "\n";
fimprog

```

algoritmo 393

Um sistema de controle de estoque armazena nome, quantidade em estoque e preço unitário de 40 mercadorias. Fazer um menu que exiba as seguintes opções:

MENU

- 1 – Cadastra mercadorias
 - 2 – Exibe valor total em mercadorias da empresa
 - 3 – Sai
- OPCAO:

```

prog vetor41
string nomes[40];
int op, x, y, cadastrou,estoque[40];
real auxhoras, auxiliar,preco[40], soma;
cadastrou <- 0;
faca
{
    imprima "\n1 - Cadastra mercadorias";
    imprima "\n2 - Exibe valor total em mercadorias da empresa";
    imprima "\n3 - Sai";
    imprima "\nDigite sua opcao:";
    leia op;
    se(op == 1)
    { para( x <- 0; x <= 39; x++)
        { imprima "\ndigite nome da mercadoria: ";

```

```

leia nomes[x];
imprima "\ndigite quantidade em estoque: ";
leia estoque[x];
imprima "\ndigite preco da mercadoria: ";
leia preco[x];
}
cadastrou <- 1;
}
senao
{ se(op == 2)
{ se(cadastrou == 0)
{ imprima "\nNao ha dados cadastrados, escolha opcao 1";}
senao
{ soma <- 0. ;
para( x <- 0; x <= 39; x++)
{ soma <- soma + estoque[x] * preco[x]; }
imprima "\nvalor total em estoque = ",soma;
}
}
senao
{ se(op == 3)
{ imprima "\nEncerrar algoritmo";}
senao
{ imprima "\nOpcao invalida";}
}
}
imprima "\n\n";
}
enquanto(op < > 3)
imprima "\n";
fimprog

```

algoritmo 394

Criar um algoritmo que possa armazenar nome, duas notas e média de 50 alunos. A média será calculada segundo o critério: peso 3 para a primeira nota e peso 7 para a segunda. A impressão deverá conter nome, duas notas e a média.

ESCOLA VIVA

- 1 - Entrar nomes
 - 2 - Entrar 1^a nota
 - 3 - Entrar 2^a nota
 - 4 - Calcular média
 - 5 - Listar no display
 - 6 - Sair
- opcao

```

prog vetor42
    int L, c,tam, flag,flag1, flag2;
    real nota1[50], nota2[50], media[50];
    string nomes[50],op;
    flag <-0;
    flag1 <-0;
    flag2 <-0;
    faca
        {imprima "\n\n\n";
        imprima "\n MENU\n";
        imprima "\n1 - ENTRAR NOMES";
        imprima "\n2 - ENTRAR 1 NOTA";
        imprima "\n3 - ENTRAR 1 NOTA";
        imprima "\n4 - CALCULAR MEDIA";
        imprima "\n5 - LISTAR NO DISPLAY";
        imprima "\n6 - SAIR";
        imprima "\nOPCAO:";
        leia op;
        se(op=="1")
        {flag<-1;
        para( L<- 0; L <= 49 ; L++)
        {
            imprima "\nDigite ", L+1, " nome (com ate 30 caracteres e todas as
            letra maiusculas): " ; # na mesma linha anterior
            leia nomes[L] ;
            enquanto(strtam(nomes[L]) >30)
            {imprima "\nNome com ate 30 caracteres. Digite ", L+1, " nome (todas
            as letras maiusculas): ";# na mesma linha anterior
            leia nomes[L];}
            tam <-strtam(nomes[L]);
            se(tam <30)
            {para(c<-0;c<=30 - tam; c++)
            {nomes[L]<-strconcat(nomes[L], "b");}
            }
        }
        senao
        { se(op=="2")
        { se(flag==0)
        {imprima "\n NAO TEM NOME CADASTRADO";}
        senao
        {
            para( L<- 0; L <= 49 ; L++)
            { imprima "\nDigite 1 nota: ";
            leia nota1[L];
            }flag1<-1;
            }
        }
        senao

```

```

{ se(op=="3")
{ se(flag==0)
{ imprima "\n NAO TEM NOME CADASTRADO";}
senao
{
  para( L<- 0; L <= 49 ; L++)
  { imprima "\nDigite 2 nota: ";
    leia nota2[L];
    }flag2<-1;
}
}
senao
{se(op=="4")
{ se(flag==0 || flag1==0 || flag2==0)
{ imprima "\n NEM TODOS OS DADOS ESTAO CADASTRADOS";}
senao
{ para( L<- 0; L <= 49 ; L++)
{ media[L]<-(3*nota1[L] + 7*nota2[L])/10;
}
}
senao
{se(op=="5")
{ se(flag==0 || flag1==0 || flag2==0)
{ imprima "\n NEM TODOS OS DADOS ESTAO CADASTRADOS";}
senao
{ imprima "\nNOME\t\t\tNOTA1\tNOTA2\tMEDIA\n";
  para( L<- 0; L <= 49 ; L++)
  { imprima "\n", nomes[L], "\t", nota1[L], "\t", nota2[L],
    "\t", media[L];
  }
}
}
senao
{se(op=="6")
{saia;}
senao
{imprima "\nOPCAO NAO DISPONIVEL";
} } } } } }
}
enquanto(op< >"6")
imprima "\n";
fimprog

```

algoritmo 395

Criar um algoritmo que possa armazenar nomes e salários de 15 pessoas. A listagem deverá conter nomes e salários, tabulados.

MENU

- 1 - **INSERIR**
- 2 - **ORDENAR**
- 3 - **LSTAR**
- 4 - **PROCURAR**
- 5 - **SAIR**

OPCAO:

```
prog vetor43
    int L, c,tam, flag;
    real sal[15], auxn;
    string nomes[15], auxc,op, nomep;
    flag <-0;
faca
{imprima "\n\n\n";
imprima "\n MENU\n";
imprima "\n1 - INSERIR";
imprima "\n2 - ORDENAR";
imprima "\n3 - LSTAR";
imprima "\n4 - PROCURAR";
imprima "\n5 - SAIR";
imprima "\nOPCAO:";
leia op;
se(op=="1")
{flag<-1;
para( L<- 0; L <= 14 ; L++)
{
    imprima "\nDigite ", L+1, " nome : ";
    leia nomes[L] ;
    enquanto(strtam(nomes[L] ) > 30)
    { imprima "\nNome com ate 30 caracteres. Digite ", L+1, " nome: ";
        leia nomes[L] ;
        tam <- strtam(nomes[L]);
        se(tam < 30)
        { para(c<-0;c<=30 - tam; c++)
            {nomes[L]<-strconcat(nomes[L], "b");}
        imprima "\n salario (maior do que 0.): ";
        leia sal[L] ;
        enquanto(sal[L] < 0.)
        {imprima "\nNAO EXISTE SALARIO NEGATIVO. Digite salario (maior
do que 0.): ";
        leia sal[L]; }
    }
}
senao
{ se(op=="2")
{ se(flag==0)
```

```

{imprima "\n NAO TEM DADOS CADASTRADOS";}
senao
{
para( L<- 0; L <= 13 ; L++)
{
para( c<- L + 1; c <= 14 ; c++)
{
se ( nomes[L] > nomes[c] )
{
auxc <- nomes[L];
nomes[L] <- nomes[c];
nomes[c] <- auxc;
auxn <- sal[L];
sal[L] <- sal[c];
sal[c] <- auxn;
} } } } }
senao
{ se(op=="3")
{ se(flag==0)
{imprima "\n NAO TEM DADOS CADASTRADOS";}
senao
{
imprima "\n\n\n";
imprima "\n NOME\t\t\t\t\t\tSALARIO\n ";
para( L<- 0; L <= 14 ; L++)
{
imprima "\n",L +1," - ",nomes[L],"\t", formatar(sal[L],2);
}
}
}
senao
{se(op=="4")
{ se(flag==0)
{imprima "\n NAO TEM DADOS CADASTRADOS";}
senao
{ imprima "\nDigite nome para procura: ";
leia nomep;
enquanto(strtam(nomep)>30)
{imprima "\nNome com ate 30 caracteres. Digite nome: ";
leia nomep;}
tam <-strtam(nomep);
se(tam <30)
{para(c<-0;c<=30 - tam; c++)
{nomep<-strconcat(nomep, " ");}}
c<-0;
enquanto(nomep< >nomes[c] && c < 14)
{c++;}
se(nomep==nomes[c])
{imprima "\nnome: ",nomes[c];
}
}
}
}

```

```

imprima "\nsalario R$ ",formatar(sal[c],2);}
senao
{imprima "\nNOME NAO ENCONTRADO";}
}
}
senao
{se(op=="5")
{saia;}
senao
{imprima "\nOPCAO NAO DISPONIVEL";
} } } }
}
enquanto(op< >"5")
imprima "\n";
fimprog

```

algoritmo 396

Criar um algoritmo que entre com nome e a matrícula para as disciplinas de Programação I e Sistemas de Informação. Cada disciplina tem 100 vagas. Após a entrada de dados, aparecerá o menu a seguir:

LISTA

- 1 – Todos de Programação I
 - 2 – Todos de Sistemas de Informação
 - 3 – Todos que fazem as duas
 - 4 – Sair
- Opcão:

1. Ao se digitar 1, sairão todos os nomes e respectivas matrículas.
2. Ao se digitar 2, sairão todos os nomes e respectivas matrículas.
3. Ao se digitar 3, sairão todos os nomes e respectivas matrículas. A PROCURA DEVERÁ SER FEITA PELA MATRÍCULA.
4. Ao se digitar 4, aparecerá a mensagem: FECHANDO.
5. Qualquer outro número, aparecerá a mensagem: OPÇÃO INVÁLIDA.

```

prog vetor44
string nomesi[100], nomeprog[100];
int vetsi[100] ,vetprog[100], I, J, OP,  C;
imprima "\nEnter com os dados dos alunos de Programacao I";
para( I <- 0; I <= 99; I++)
{ imprima "\n NOME: ", I + 1, ": ";
  leia nomeprog[I] ;
  imprima "\nMATRICULA: ";
  leia vetprog[I] ;
}
imprima "\n\nEnter com os dados dos alunos de Sistema de Informacao";
para( I <- 0; I <= 99; I++)

```

```

{ imprima "\n NOME: ", I, ": ";
leia nomesi [I] ;
imprima "\nMATRICULA: ";
leia vetsi[I] ;
}
faca
{
    imprima "\n\nLISTA";
    imprima "\n1 - LISTA TODOS DE PROGRAMACAO I";
    imprima "\n2 - LISTA TODOS DE SISTEMAS DE INFORMACAO";
    imprima "\n3 - LISTA TODOS QUE FAZEM AS DUAS ";
    imprima "\n4 - SAIR ";
    imprima "\nOPCAO: ";
    leia OP;
    se ( OP == 1)
    {
        imprima "\n\n RELACAO DOS ALUNOS DE PROGRAMACAO I\n";
        para( I <- 0; I <= 99; I++)
        {   imprima "\n",vetprog[ I] , "-", nomeprog[I] ; }
    }
    senao
    { se ( OP == 2)
        { imprima "\n\n RELACAO DOS ALUNOS DE SISTEMA DE INFORMACAO \n";
        para( I <- 0; I <= 99; I++)
        {   imprima "\n",vetsi[ I] , "-", nomesi[I] ; }
    }
    senao
    { se ( OP == 3)
        { imprima "\n\n RELAÇO DOS ALUNOS QUE CURSAM SI / PROG. I \n";
        para( I <- 0; I <= 99; I++)
        {   J <- 0;
            enquanto( J < 99 && vetprog[I] < > vetsi[J] )
            { J++; }
            se( vetprog[I] == vetsi[J] )
            { imprima "\n",vetsi[I] , "-", nomesi[I] ; }
        }
    }
    senao
    { se ( OP == 4)
        { imprima "\nFECHANDO";}
        senao
        { imprima "\nOpcao nao disponivel";}
    } } }
    imprima "\n\n";
}
enquanto( OP < > 4)
imprima "\n";
fimprog

```

algoritmo 397

Uma costureira tem dez freguesas fixas. Ela gostaria de fazer um algoritmo que funcionasse de acordo com o menu a seguir:

↳ A costureira só faz vestidos (simples, passeio ou longo), calça, saia, blusa, conjunto e blazer.

Atelier Maravilha

- 1 – Cadastrar as freguesas
- 2 – Cadastrar preços das costuras
- 3 - Calcular e imprimir o total que será pago por cada freguesa
- 4 - Listar dados de uma cliente
- 5 - Sair do programa

```
prog vetor45
int L, flag, flag1, op, ped,n ;
string resp, nome[10], tel[10];
real preco[8], soma,valor, pedido[10];
para( L<- 0; L < 10 ; L++)
{ pedido[L] <- 0.;}
flag <-0;
flag1 <-0;
faca
{
imprima "\n\n\n";
imprima "\n Atelier Maravilha \n";
imprima "\n1 - Cadastrar as freguesas ";
imprima "\n2 - Cadastrar precos das costuras ";
imprima "\n3- Calcular e imprimir o total que ser pago por cada freguesa ";
imprima "\n4- Lista dados de uma cliente ";
imprima "\n5 - Sai do programa ";
imprima "\nOPCAO:";

leia op;
se( op == 1)
{ flag<-1;
  para( L<- 0; L < 10 ; L++)
  { imprima "\n nome " , L + 1, ": "; leia nome[L] ;
    imprima "\n telefone " , L + 1, ": "; leia tel[L] ;
  }
}
senao
{ se(op == 2)
  {imprima "\n1 Ves-s 2 Ves-p 3 Ves-L 4 Conjunto 5 Blazer 6 Saia 7 Calca 8 blusa";
    para( L<- 0; L < 8; L++)
    { imprima "\n preco do ", L + 1, ": "; leia preco[L];}
  }
}
```

```

flag1 <-1 ;
}
senao
{se(op == 3)
{ se(flag1 == 0)
{imprima "\n Nao tem precos cadastrados";}
senao
{soma <- 0. ;
imprima "\n1 Ves-s 2 Ves-p 3 Ves-L 4 Conjunto 5 Blazer 6 Saia 7 Calca 8
blusa 0 para acabar";
leia ped;
enquanto( ped < > 0)
{enquanto( ped < 1 || ped >8)
{imprima "\n1 Ves-s 2 Ves-p 3 Ves-L 4 Conjunto 5 Blazer 6 Saia 7 Calca 8
blusa r";
leia ped; }
soma <- soma + preco[ped -1];
imprima "\n1 Ves-s 2 Ves-p 3 Ves-L 4 Conjunto 5 Blazer 6 Saia 7 Calca 8
blusa 0 para acabar";
leia ped; }
imprima "\nTotal: ", soma;
imprima "\n Cliente cadastrada(S/N)? "; leia resp;
se( resp == "S" || resp =="s")
{para( L<-0; L < 10 ; L++)
{imprima "\n ", L + 1, ": ", nome[L] ; }
imprima "\n Numero da cliente: "; leia n;
enquanto( n< 1 || n>10)
{imprima "\nNumero de 1 - 10: "; leia n; }
pedido[n-1] <- soma;
}
}
}
senao
{ se(op==4)
{
para( L<-0; L < 10 ; L++)
{imprima "\n ", L + 1, ": ", nome[L] ; }
imprima "\n Numero da cliente: ";
leia n;
enquanto( n< 1 || n>10)
{imprima "\nNumero de 1 - 10: "; leia n; }
imprima "\nSaldo R$: ", pedido[n -1];
imprima "\n Fazer pagamento(S / N)? ";
leia resp;
se( resp == "S" || resp =="s")
{imprima "\n Valor do pagamento: ";
leia valor;
pedido[n-1] <- pedido[n-1]- valor; }
}
senao
{ se(op==5)

```

```

    {imprima "\nSaindo do algoritmo"; }
senao
{imprima "\nOpcao nao disponivel"; }
}
}
}
}
}
enquanto(op< >5)
imprima "\n";
fimprog

```

algoritmo 398

A Fábrica de Queijo Rio Novense deseja elaborar um algoritmo para controlar o estoque e as vendas. Inicialmente, deverão ser lidos e armazenados em vetores: o código, a quantidade disponível em estoque e o preço de venda dos produtos. O término do cadastramento é determinado quando se digita –1 para o código do produto. Sabe-se que a Fábrica de Queijo Rio Novense trabalha com no máximo 50 produtos diferentes.

A segunda fase do algoritmo é a venda. Deverá ser lido o código do produto a ser vendido e a quantidade requerida. Se o código do produto estiver cadastrado, a venda poderá ser realizada; caso contrário, a mensagem Produto Não-Cadastrado deverá ser exibida no monitor. Caso o produto esteja disponível, a venda só poderá ser realizada se a quantidade disponível no estoque for suficiente para atender ao pedido. Nesse caso, você deverá abater do estoque a quantidade vendida. Se o estoque não for suficiente para atender ao pedido, a mensagem Estoque Insuficiente deverá ser exibida no monitor. O final das vendas será detectado quando o código do produto for igual a zero.

No final, deverá aparecer uma listagem no vídeo contendo o total vendido no dia e a relação de todos os produtos do estoque, com suas respectivas quantidades, em ordem decrescente de quantidades disponíveis.

```

prog vetor46
int k,codi,cod[50],quant[50],codigo,t,i,quantidade,aux;
real preco[50], venda, geral, auxp;
k<-0;
geral <- 0.;
imprima "\nDigite o codigo do produto ou 0 para acabar: ";
leia codi;
enquanto(k <50 && codi < > 0)
{ cod[k] <- codi;
  imprima "\nDigite a quantidade do produto: ";
  leia quant[k];
  imprima "\nDigite o preco do produto: ";
  leia preco[k];
  k++;
  imprima "\nDigite o codigo do produto ou 0 para acabar: ";

```

```

senao
{ imprima "\nCentral completa"; }
}
senao
{ se(op==2)
{ i <- 0;
imprima "\nDigite o nome: ";
leia nnome;
enquanto(nnome < > nome[i] && i < k -1)
{ i++; }
se(nnome == nome[i])
{
    imprima "\nDigite o novo telefone: ";
    leia ntel;
    tel[i] <- ntel;
}
senao
{ imprima "\nNome nao cadastrado"; }
}
senao
{ se(op == 3)
{ k--;
i <- 0;
imprima "\nDigite o nome: ";
leia nnome;
enquanto(nnome < > nome[i] && i < k )
{ i++; }
se(nnome == nome[i])
{ nome[i] <- "Vazio"; tel[i] <- 0;}
senao
{imprima "\nNome nao cadastrado"; }
}
senao
{ se(op==4)
{ para(i <- 0; i <= k -2; i++)
{ para(cont <- i +1 ; cont <= k -1; cont++)
{ se(nome[i] > nome[cont])
{ auxnome <- nome[i]; nome[i] <- nome[cont];
nome[cont] <- auxnome;
auxtel <- tel[i]; tel[i]<-tel[cont]; tel[cont] <-
auxtel;
}
}
}
imprima "\nRelacao dos telefones";
para(i <- 0; i < 1000 ; i++)
{ se(nome[i] < > "Vazio")
{

```

```

        imprima "\nNome: ", nome[i];
        imprima "\nTel: ", tel[i];
        imprima "\n";
    }
}

senao
{ se(op == 5)
{ i <- 0;
    imprima "\nDigite o nome: ";
    leia nnome;
    enquanto(nnome < > nome[i] && i < k)
    { i++;}
    se(nnome == nome[i])
    { imprima "\nNome: ", nnome; imprima "\nTel: ", tel[i]; }
    senao
    {imprima "\nNome nao cadastrado"; }
}
senao
{ se ( op == 6)
{ imprima "\nSaindo";}
senao
{ imprima "\nOpcao nao disponivel";}
} } } }

imprima "\n\n";
}
enquanto( op < > 6)
imprima "\n";
fimprog

```

algoritmo 400

Um hotel-fazenda gostaria de fazer um algoritmo que pudesse controlar os seguintes dados dos 50 quartos:

- número de leitos por quarto;
- preço;
- situação: alugado, livre ou reservado;
- aluguel do quarto com data de entrada e de saída e número de diárias;
- despesas dentro do hotel;
- valor a ser pago;
- impressão de todos os quartos com a situação de cada um;
- impressão dos quartos livres.

Criar um algoritmo que funcione de acordo com o menu a seguir:

Hotel-Fazenda Sucesso

- 1. Cadastra quartos*
- 2. Lista todos os quartos*
- 3. Lista quartos ocupados*
- 4. Aluguel/reserva quarto*
- 5. Entra despesas extras*
- 6. Calcula despesa do quarto*
- 7. Sai*

```
prog vetor48
int leitos[50], nd[50], i, k, chave, quarto, op;
string sit[50], din[50], dout[50], nome[50], tel[50], resp, resp1, respo,
nomep;
real precos[50], despesas[50], valor, total;
chave <- 0;
faca
{
    imprima "\n\n\nHotel Fazenda Sucesso \n";
    imprima "\n1. Cadastra quartos ";
    imprima "\n2. Lista todos os quartos ";
    imprima "\n3. Lista quartos desocupados ";
    imprima "\n4. Aluguel / reserva quarto ";
    imprima "\n5. Entra despesas extras ";
    imprima "\n6. Calcula despesa do quarto ";
    imprima "\n7. Sai";
    imprima "\nOpcão: ";
    leia op;
    imprima "\n";
    se(op==1)
    { se( chave ==1)
        { imprima "\nAtenção. Dados ja cadastrados"; }
        senao
        { para( i <- 0; i < 50; i++)
            # use este trecho se vc desejar ver o funcionamento
            { sit[i] <-"L"; nd[i] <- 0; despesas[i] <- 0. ;
             din[i] <-"XXXX"; dout[i] <-"XXXX"; nome[i] <- "";
             tel[i] <- "";
             precos[i] <- 30.; leitos[i] <-5; }

            # o trecho abaixo e que seria o correto
/*{ imprima "\nquantidade de leitos para o quarto ", i + 1, ": ";
             leia leitos[i];
             imprima "\npreco do quarto: ";
```

```

leia precos[i];
sit[i] <- "L";
nd[i] <- 0;
despesas[i] <- 0.;
din[i] <- "XXXX";
dout[i] <- "XXXX";
nome[i] <- "";
tel[i] <- "";
} */

chave <- 1;
}

}

senao
{ se(op==2)
{ se( chave == 0)
{ imprima "\nEscolha a opcao 1"; }
senao
{ para( i <- 0; i < 50; i++)
{ imprima "\nnumero do quarto: ", i +1, "\n";
imprima "\nsituacao quarto: ", sit[i];
se(sit[i]== "A" || sit[i] == "R")
{ imprima "\n nome: ", nome[i]; imprima "\ntelefone: ", tel[i];}
imprima "\nquantidade de leitos: ", leitos[i];
imprima "\npreco do quarto: ", precos[i];
imprima "\ndespesas: ", despesas[i];
imprima "\ndata de entrada: ", din[i];
imprima "\ndata de saida: ", dout[i];
imprima "\nnumero de dias: ", nd[i];
imprima "\npressione enter para continuar: ";
leia respo;
}
}
}
senao
{ se(op==3)
{ se( chave ==0)
{ imprima "\nEscolha a opcao 1"; }
senao
{ para( i <- 0; i < 50; i++)
{ se( sit[i] == "A")
{ imprima "\nnumero do quarto: ", i +1, "\n";
imprima "\ndespesas: ", despesas[i];
imprima "\ndata de entrada: ", din[i];
imprima "\ndata de saida: ", dout[i];
imprima "\nnumero de dias: ", nd[i];
imprima "\npressione enter para continuar: ";
leia resp;
}
}
}
}

```



```

{ imprima "\nQuarto Invalido. Entre novamente: ";
  leia quarto; }
enquanto(sit[quarto-1] == "A")
{ imprima "\nQuarto ocupado. Digite novamente: ";
  leia quarto; }
se( resp == "A" || resp == "a")
{ sit[quarto-1] <- "A"; despesas [quarto -1] <- 0.;}
senao
{ sit[ quarto -1] <- "R"; }
imprima "\nDigite nome: "; leia nome[quarto -1];
imprima "\nDigite telefone para contato: "; leia
tel[quarto-1];
imprima "\ndata de entrada: "; leia din[quarto -1];
imprima "\ndata de saida: "; leia dout[quarto -1];
imprima "\nnumero de dias: ";leia nd[quarto-1];
}
}
senao
{ se(op==5)
{ se( chave ==0)
{ imprima "\nEscolha a opcao 1"; }
senao
{ imprima "\nentre com numero do quarto: ";
  leia quarto;
enquanto( quarto < 1|| quarto > 50)
{ imprima "\nentre com numero do quarto: ";
  leia quarto; }
imprima "\ndespesas: ";
leia valor;
despesas[quarto-1] <- despesas[quarto-1]+ valor;
}
}
senao
{ se(op==6)
{ se( chave ==0)
{ imprima "\nEscolha a opcao 1"; }
senao
{ imprima "\nentre com numero do quarto: ";
  leia quarto;
enquanto( quarto < 1|| quarto > 50)
{ imprima "\nentre com numero do quarto: ";
  leia quarto; }
total <- despesas[quarto-1] + precos[quarto-1];
imprima "\ndespesas: R$ ", total;
sit[quarto-1] <- "L";
despesas[quarto-1] <- 0. ;
din[quarto-1] <- "XXXX";
dout[quarto-1] <-"XXXX";
}
}

```

```

        nd[quarto-1] <- 0;
        nome[quarto-1] <- "";
    }
}
senao
{ se(op==7)
{ imprima "\nSaindo ";
senao
{imprima "\nOpcao nao disponivel";}
} } } } } }
}
enquanto( op< >7)
imprima "\n\n";
fimprog

```

algoritmo 401

Criar um algoritmo que funcione de acordo com o menu a seguir, sabendo-se que poderão ser cadastradas até 50 pessoas.

```

*****
*      MENU      *
*****
1 - cadastra dados do cliente
2 - cadastra milhagem do cliente
3 - lista milhagem do cliente
4 - imprime os nomes que têm maior e menor milhagem
5 - imprime os nomes e as milhagens
6 - SAIR
OPCAO:

```

OBSERVAÇÃO

*A maioria das soluções foi feita com **ses** aninhados para que você pudesse testar no interpretador, mas muitas delas teriam melhor visualização se fossem feitas com escolha. Terminaremos essa lista com um exercício com as duas soluções.*

```

prog vetor49
# nao roda na versao 2 do UAL, troque escolha por ses aninhados para
ver a execucao
int op,c,d, posmenor, posmaior;
string dados1[50], dados2[50], dados3[50], nome;
real milha[50]. milhas;
para( c<-0;c < 5; c++)
{milha[c]<-0.;}
c<-0;
faca

```

```

{ imprima "\n\n\n*****";
imprima "\n      *      MENU      *";
imprima "\n      *****";
imprima "\n 1 - cadastra dados do cliente  ";
imprima "\n 2 - cadastrada milhagem do cliente ";
imprima "\n 3 - lista milhagem do cliente  ";
imprima "\n 4 - imprime os nomes que tem maior e menor milhagem ";
imprima "\n 5 - imprime os nomes e as milhagens ";
imprima "\n 6 - SAIR      ";
imprima "\nOPCAO:";

leia op;
escolha(op)
{
caso 1:
se(c<50)
{
    imprima "\n", c+1,"- nome: ";leia dados1[c];
    imprima "\nendereço: "; leia dados2[c];
    imprima "\ntelefone: "; leia dados3[c];
    c++;
}
senao
{ imprima "\narquivo cheio";}
pare;
caso 2:
imprima "\nNome para procura: ";leia nome;
d<-0;
enquanto( d< c - 1 && nome < > dados1[d] )
{ d++; }
se(nome==dados1[d])
{ imprima "\ndigite milhagem de ", dados1[d] ,": ";leia milhas;
  milha[d] <-milha[d] +milhas;
}
senao
{ imprima "\nNome não encontrado";}
pare;
caso 3:
imprima "\nNome para procura: ";leia nome;
d<-0;
enquanto(d< c - 1 && nome < > dados1[d] )
{ d++; }
se(nome==dados1[d])
{ imprima "\nmilhagem de ",dados1[d] ,": ",milha[d] ;
}
senao
{ imprima "\nnão encontrado";}
pare;
caso 4:
d<-1;
posmenor<-0;
posmaior<-0;

```

```

enquanto(d <= c)
{ se(milha[d] > milha[posmaior])
{ posmaior<-d;}
senao
{ se(milha[d] < milha[posmenor])
{ posmenor<-d; }
}
d++;
}
imprima "\nDados da pessoa de maior milhagem";
imprima "\nNome: ",dados1[posmaior];
imprima "\nEndereco: ",dados2[posmaior];
imprima "\nTelefone: ",dados3[posmaior];
imprima "\nMilhagem: ",milha[posmaior];
imprima "\n\n";
imprima "\nDados da pessoa de menor milhagem";
imprima "\nNome: ",dados1[posmenor];
imprima "\nEndereco: ",dados2[posmenor];
imprima "\nTelefone: ",dados3[posmenor];
imprima "\nMilhagem: ",milha[posmenor];
pare;
caso 5:
imprima "\nListagem";
para( d<-0;d< c; d++)
{ imprima "\n", d,"-", dados1[d] ,": ", milha[d] ;}
pare;
caso 6:
imprima "\nBOA VIAGEM";
pare;
senao
imprima "\nOpcao inexistente ";
}
}
enquanto( op < > 6)
imprima "\n";
fimprog

prog vetor49
int op,c,d, posmenor, posmaior;
string dados1[50], dados2[50], dados3[50], nome;
real milha[50], milhas;
para( c<-0;c < 5; c++)
{ milha[c]<-0.;}
c<-0;
faca
{ imprima "\n\n\n*****";
imprima "\n      *      MENU      *";
imprima "\n      *****";
imprima "\n 1 - cadastra dados do cliente ";
imprima "\n 2 - cadastra milhagem do cliente ";

```

```

imprima "\n 3 - lista milhagem do cliente    ";
imprima "\n 4 - imprime os nomes que tem maior e menor milhagem ";
imprima "\n 5 - imprime os nomes e as milhagens ";
imprima "\n 6 - SAIR ";
imprima "\nOPCAO:";

leia op;
se(op==1)
{ se(c<=49)
{
    imprima "\n", c+ 1,"- nome: ";leia dados1[c];
    imprima "\nendereco: "; leia dados2[c];
    imprima "\ntelfone: "; leia dados3[c];
    c++;
}
senao
{ imprima "\narquivo cheio";}
}
senao
{se( op ==2)
{ imprima "\n nome para procura: ";leia nome;
d<-0;
enquanto( d< c-1  && nome < > dados1[d] )
{ d++; }
se(nome==dados1[d] )
{ imprima "\ndigite milhagem de ", dados1[d] ,": ";leia milhas;
milha[d] <-milha[d] +milhas;
}
senao
{ imprima "\nnome nao encontrado";}
}
senao
{ se( op == 3)
{ imprima "\n nome para procura: ";leia nome;
d<-0;
enquanto( d< c-1  && nome < > dados1[d] )
{ d++; }
se(nome==dados1[d] )
{ imprima "\nmilhagem de ",dados1[d] ,": ",milha[d] ; }
senao
{ imprima "\nnao encontrado";}
}
senao
{ se( op == 4)
{ d <-1;
posmenor<-0;
posmaior<-0;
enquanto(d < c)
{ se(milha[d] > milha[posmaior])
{ posmaior <- d;}
senao
{ imprima "\nmaior milhagem: ",posmaior;
imprima "\nmenor milhagem: ",posmenor;
}
}
}
}

```

```

{ se(milha[d] < milha[posmenor])
  { posmenor <- d; }
}
d++;
}
imprima "\nDados da pessoa de maior milhagem";
imprima "\nNome: ",dados1[posmaior];
imprima "\nEndereco: ",dados2[posmaior];
imprima "\nTelefone: ",dados3[posmaior];
imprima "\nMilhagem: ",milha[posmaior];
imprima "\n\n";
imprima "\nDados da pessoa de menor milhagem";
imprima "\nNome: ",dados1[posmenor];
imprima "\nEndereco: ",dados2[posmenor];
imprima "\nTelefone: ",dados3[posmenor];
imprima "\nMilhagem: ",milha[posmenor];
}
senao
{ se( op == 5)
  { imprima "\nListagem";
    para( d<-0;d< c; d++)
    { imprima "\n", d+1,"-", dados1[d] ,": ", milha[d] ;}
  }
  senao
  { se( op == 6)
    { imprima "\nBOA VIAGEM";}
    senao
    { imprima "\nOpcao inexistente";}
  } } } } }
}
enquanto( op < > 6)
imprima "\n";
fimprog

```

CONCEITOS DE MATRIZES

Nossa experiência mostra que este é um dos momentos críticos no aprendizado de algoritmos e vamos tentar ajudá-lo, recordando alguns conceitos matemáticos. Uma matriz, na verdade, é uma tabela contendo elementos que servirão de base para vários cálculos em muitas ciências tais como Estatística, Economia, Física, Ciência da Computação etc.

Observando os exemplos a seguir, perceberemos que os elementos estão dispostos em linhas e colunas:

$$A \begin{bmatrix} 2 & 3 \\ 6 & 8 \end{bmatrix}$$

$$D \begin{bmatrix} 3 & 12 \\ 14 & 8 \\ 15 & 17 \end{bmatrix}$$

$$X \begin{bmatrix} 2 & 4 & 5 \\ 6 & 9 & 10 \\ 3 & 5 & 8 \end{bmatrix}$$

332 | A 2 linhas e 2 colunas D 3 linhas e 2 colunas X 3 linhas e 3 colunas