Seja um vetor inteiro de 9 elemen
a) preencher por leitura o vetor (**apen**
b) imprimir os elementos do vetor em

```
//Testa vetores
#include <stdio.h>
#include <stdlib.h>
#define ELEM 9
int main ( )
{
 int vet[ELEM]; // será lido: não precisa inicializar
 int i; // controle do for e indice
 int val, cont; // valor a ser lido e contador de ocorrências
 system("color f1");     // altera cores da tela de execução
 printf("Preenchimento do vetor: (apenas valores inteiros e maior que 0)\n\n");
 for (i = 0; i< ELEM; i++)
 {
    printf("   elemento[%d]= ",i + 1);
    scanf("%d",&vet[i]);
    while (vet[i] < 1) // se valor inválido, entra no laço
    {
        printf("*** Valor inválido!");
        scanf("%d",&vet[i]);
    }
 }
 printf("\n\n Conteúdo do vetor:\n");
 for (i=0; i< ELEM; i++)
     printf("%d   ", vet[i]); // imprime valor seguido de 2 espaços em branco
 printf("\n");
```

```
Preenchimento do vetor: (apenas valores inteiros e main

   elemento[1]= 9
   elemento[2]= 8
   elemento[3]= 7
   elemento[4]= 6
   elemento[5]= 7
   elemento[6]= 8
   elemento[7]= 5
   elemento[8]= 4
   elemento[9]= 3

Conteudo do vetor:
9  8  7  6  7  8  5  4  3
```


c) **liberar a primeira posição do vetor**, deslocando todos os 8 primeiros valores para a posição seguinte (**o último valor será perdido nesse processo**);
d) **ler um novo valor** a ser armazenado na primeira posição **e escrever** o vetor modificado (as posições do vetor DEVEM ESTAR efetivamente alteradas!!!), **ocupando 4 espaços** com cada conteúdo;

```
 for (i= ELEM - 1; i > 0 ; i--) //percorre vetor, do final para o início
      vet[i] = vet[i-1]; // substitui conteúdo pelo da posição anterior
 do
 {
    printf("\n*** Informe um valor inteiro e positivo: ");
    scanf("%d",&vet[0]); // lê conteúdo no 1o elemento do vetor
 } while (vet[i] < 1);
 printf("\n\n Conteúdo atual do vetor:\n");
 for (i=0; i < ELEM; i++)
     printf("%4d",vet[i]); // imprime valor ocupando 4 posições da tela
 printf("\n");
```

2 brancos interm.

4 posições

```
Conteudo do vetor:
9  8  7  6  7  8  5  4  3

*** Informe um valor inteiro e positivo: 12

Conteudo atual do vetor:
 12   9   8   7   6   7   8   5   4
```


e) **somar** o **índice posicional** de cada elemento ao conteúdo do mesmo (alterando o calor armazenado no vetor) **e escrever o vetor modificado**;
f) **ler** um valor e **imprimir** o **número de ocorrências** desse valor no vetor.

```
 for (i=0; i < ELEM ; i++) //percorre vetor
     vet[i] = vet[i] + i + 1; // incrementa a posição (índice + 1) ao conteúdo
 printf("\n\n Adiciona posicao ao vetor:\n");
 for (i=0; i < ELEM; i++)
     printf("%4d",vet[i]); // imprime valor ocupando 4 posições da tela
 // obtêm valor e conta quantas vezes ocorre no vetor:
 cont = 0; //zera contador
 printf("\nInforme valor a ser procurado no vetor:");
 scanf("%i", &val);
 for (i=0; i < ELEM ; i++) //percorre vetor
     if (vet[i] == val)
         cont++;
 printf("\nO numero %d aparece %d vezes no vetor. \n\n", val, cont);
 system("pause");
 return 0;
}
```

```
Conteudo atual do vetor:
 12   9   8   7   6   7   8   5   4

Adiciona posicao ao vetor:
 13  11  11  11  11  13  15  13  13
Informe valor a ser procurado no vetor:13

O numero 13 aparece 4 vezes no vetor.

Pressione qualquer tecla para continuar. . .
```


Escreva um programa que:
1) lê um texto contendo até MAXTXT caracteres (consistindo);
2) lê 1 caractere e informa a última posição do texto onde este caractere ocorre ou que não existe tal caractere no texto;

```
//Localiza caractere e palavras em texto:
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#define TXT 40
#define PAL 10
int main()
{
  char texto[TXT+1], palavra[PAL+1];
  char pal_aux[PAL+1]; //solução usando palavra auxiliar
  char caractere;
  int cont_palavra, ind_t, ind_p, existe, lim;
  system("color f1");
  printf("Digite o texto onde ocorrera a pesquisa:\n");
  gets(texto); // \n é substituído por \0
  printf("\nDigite o caractere a ser localizado:\n");
  caractere = getchar();
  ind_t = strlen(texto) - 1; // posiciona no último caractere do texto
  while (ind_t >= 0 && texto[ind_t] != caractere)
        ind_t--; // percorre até encontrar caractere ou chegar ao início do texto
  if (ind_t >= 0) // encerrou while porque encontrou
      printf("\%c aparece a última vez na posição %d.\n",caractere,ind_t + 1);
  else
      printf("\nO caractere %c nao foi localizado no texto\n",caractere);
```

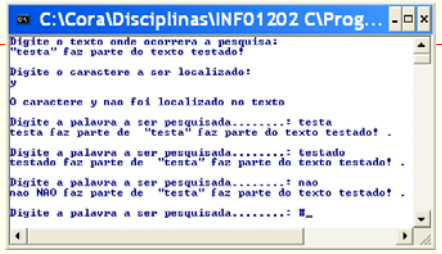

3) lê uma palavra de até MAXPAL caracteres e, enquanto a palavra digitada não iniciar pelo caractere #, verifica se a seqüência de caracteres desta palavra existe no texto lido.

Solução 1: comparando e contando caracteres iguais:


```
.....
do // laço para leitura de palavras a serem pesquisadas
  {
   ind_t=0; // para texto
   printf("\nDigite a palavra a ser pesquisada.....: ");
   fflush(stdin); // limpa conteúdo do buffer do teclado
   gets(palavra);
   if (palavra[0] != '#') // sinalizador de fim de pesquisa
   {  existe = 0; // considera como não existindo
 /* enquanto o nro de caracteres a serem analisados for maior o igual ao
    tamanho da palavra procurada  e não tiver encontrado igual*/
      while (ind_t + strlen(palavra) <= strlen(texto) && existe == 0)
       {
          ind_p = 0; // para a palavra, para comparar caractere a caractere
          while ((toupper(texto[ind_t+ind_p]) == toupper(palavra[ind_p]))
                       &&  (ind_p < strlen(palavra))) //se existir caract suficientes
                ind_p++; // percorre enquanto for igual
          if (ind_p == strlen(palavra)) // se chegou ao final da palavra, existe
          {
             existe = 1;
             printf(" %s aparece, iniciando no caractere %d\n",palavra,ind_t+1);
          }
          ind_t++; // incrementa posição no texto
       }
      if ( !(existe) ) // se existe == 0
         printf("\nTexto nao foi encontrado.\n");
   }
  } while (palavra[0]!='#');
system("PAUSE");
return 0;
} //fim do programa
```

Outra solução, montando palavras e comparando com a palavra lida:

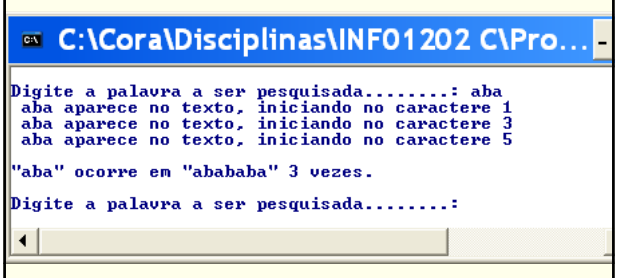

```
...
 do // laço para leitura de palavras a serem pesquisadas
  {
    ind_t=0; // para texto
    printf("\nDigite a palavra a ser pesquisada.....: ");
    fflush(stdin); // limpa conteúdo do buffer do teclado
    gets(palavra);
    if (palavra[0] != '#') // sinalizador de fim de pesquisa
    {  existe = 0; // considera como não existindo
       ind_t = 0; // posiciona ni início do texto
       lim = strlen(palavra); // nro de caracteres da palavra, sem sinalizador
       // repete laço, enquanto strings diferentes e não chegou ao final do texto:
       do
        {
          for (ind_p = 0; ind_p < lim; ind_p++)
             pal_aux[ind_p] = texto[ind_t + ind_p]; /* monta palavra com lim caracteres,
                                                       a partir da posição ind_t do texto */
          pal_aux[ind_p] = '\0';  // inclui sinalizador de final de string
          ind_t ++; // avança 1 caractere no texto, para extrair próxima palavra
        } while (strcmp(pal_aux,palavra) != 0 &&  ind_t <= strlen(texto) - lim);
       if (strcmp(pal_aux,palavra) == 0) // palavras iguais
            printf("%s faz parte de  %s .\n",palavra, texto);
       else
            printf("%s NAO faz parte de  %s .\n",palavra, texto);
    }
 } while (palavra[0] != '#');
 system("pause");
 return 0;
} // fim do programa
```



E se quiser contar o número de vezes que a palavra aparece no texto?


```
printf("\nDigite a palavra a ser pesquisada.....: ");
gets(palavra);   // lê 1ª palavra
while (palavra[0] != '#')
    {
      printf("\nDigite a palavra a ser pesquisada.....: ");
      gets(palavra);
      cont_palavras=0;
      ind_t=0; // para texto
      ind_p=0; // para palavra
/* enquanto nro de caracteres a serem analisados for maior o igual ao tamanho
   da palavra procurada */
      while ((strlen(texto))>=(strlen(palavra) + ind_t))
        {
          ind_p = 0;
          while ((toupper(texto[ind_t + ind_p]) == toupper(palavra[ind_p]))
                 && (ind_p < strlen(palavra)))
                ind_p++;
          if (ind_p == strlen(palavra))
              cont_palavras++;
          ind_t++;
        }
      if (cont_palavras > 0)
       printf("\n\"%s\" ocorre em \"%s\" %d vezes.\n", palavra, texto, cont_palavras);
      else
          printf("\nPalavra nao foi encontrada.\n");
      printf("\nDigite a palavra a ser pesquisada.....: ");
      gets(palavra);  // lê próxima
}
```

Como também mostrar a posição inicial da palavra no texto em cada vez que aparece?



Declaração de Variável Inicializada

tipo nome variável = valor ;

Ex:

int v = 3; → **Idêntico a** → **int v ;** **{mais adiante no programa }** **v = 3;**

```
//testando inicializacao de constante
#define LIMITE 30  // LIMITE não pode ser alterado no programa
int main ( )
{
  int v = 3;  // v é variável -  pode ter o valor alterado
  ...
  v = 7;
  ...
}
```

Arranjos

- *1 dimensão*
- *várias dimensões*

Arranjo 2 dimensões (Matriz)

Ex: Cinco alunos de uma turma;
Ler e armazenar as 3 notas de cada aluno.
notas

Notas

alunos	1	2	3
1	7.5	8.2	9.1
2			
3			
4			
5			

Arranjo 2 dimensões (Matriz)

C - declaração:

```
tipo nome_arranjo[nro_elem_1ª_dimo] [nro_elem_2ª_dim];
```

- Número de elementos da dimensão:
  inteiro ou representar um valor inteiro
- tipo: *int*, *float*, *double* (entre outros).

Ex:

```
float notas[5][3]; // matriz de 15 elementos reais
int x[10][25]; // matriz de 250 elementos inteiros
```


Arranjo 2 dimensões (Matriz)

C - utilização:

```
nome_arranjo [índice1ªdim] [índice2ªdim]
```

Ex:

```
int main ( ) // refere todas as dimensões
{
  float nota [5] [3];
  scanf( "%f", &nota [0] [2]); // 1º aluno, 3ª nota dele

  nota [1] [2] = 7.5 ;

  nota [2] [1] = nota [1] [1] + 2 ;

  if (nota [0] [1] > 6.0)
     printf ("Aprovado");
} ...
```


EX: Ler e armazenar as 3 notas de cada um dos 10 alunos de uma turma.
Calcular e informar a média da 1ª nota
Quantos alunos tem a 1ª nota superior a esta média ?

```
//Processa notas de uma turma
#include <stdio.h>
int main ( )
{
float notas [ 10] [ 3] ;  // matriz para as notas
int a, n;   // indices para as 2 dimensões
float soma, media; // soma e media de notas
int cont; // contador de alunos
for (a = 0; a< 10; a++)          // para cada aluno
     for (n = 0; n < 3; n++)          // obtem as 3 notas do aluno
          scanf ("%f", &notas[ a] [ n]);
// Calculo da media da primeira nota (2º índice = zero)
soma = 0;
for (a = 0; a< 10 ; a++)
          soma = soma + notas [a] [0];
media = soma / 10.
printf("Media da primeira nota: %6.2f", media);

// Conta quantos alunos tem 1a nota (indice zero) > media da primeira nota
cont = 0;
for (a = 0; a < 10; a++)
    if (notas [a] [ 0] > media)
          cont++;
printf("Numero de alunos ...: %d", cont);
return 0;
}
```

Substituir literal 10
por
#define NUMAL 10

Ex: Matriz inteira **m** (10, 3), já preenchida.
Imprimir em forma de matriz (10 espaços) !

```
...
for (linha = 0; linha < 10; linha++)
   {
   for (coluna = 0; coluna < 3; coluna++)
         printf( "%10d", m[linha] [coluna]);
   printf("\n\n");
   }
...
```

Imprimir coluna 3.

```
...
for (linha = 0; linha < 10; linha++)
     printf("%10d\n", m[linha] [2]);
printf("\n\n");
...
```

Somar 2 em cada elemento da linha 7.

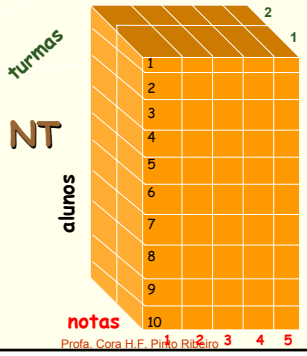
```
...
for (coluna = 0; coluna < 3; coluna++)

     m [6] [coluna] = m [6] [coluna] + 2;
...
```

m

	0	1	2
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

Ex: Duas turmas
10 alunos em cada turma
5 notas para cada aluno
Ler e armazenar as notas de cada aluno.

Matriz NT [2] [10] [5]



Arranjo 3 dimensões – Linguagem C

Declaração:

```
float nt [ 2]  [10]  [5];
```

1ª dimensão
2ª dimensão
3ª dimensão

Arranjo 3 dimensões – Linguagem C

Utilização

- um índice para cada dimensão;
- respeitando a ORDEM da declaração.

Ex:

turmas alunos notas

```
float nt [ 2]  [10]  [5];
```

referência

```
nt [turma] [aluno] [nota]
```