

Vetor e Matriz

- 1 Estruturas de Dados
 - Vetor ou Array unidimensional
 - Strings
 - Matriz ou Array multidimensional

Estruturas de Dados

Estruturas de Dados x Variável “simples”

- Variáveis “simples”:
 - ▶ Uso muito limitado da memória do computador;
 - ▶ Cada variável armazena somente um valor por vez.
- Estruturas de dados:
 - ▶ Organizar melhor os dados na memória (sequencial);
 - ▶ Permitir algoritmos mais eficientes quando exploram esta organização (fácil/previsível acesso);
 - ▶ Vetor ou array unidimensional;
 - ▶ Matriz ou array multidimensional;

- 1 Estruturas de Dados
 - Vetor ou Array unidimensional
 - Strings
 - Matriz ou Array multidimensional

Vetor ou Array unidimensional

Declaração

- 1 variável → 1 tipo → vários conteúdos
- Declaração:
 - ▶ TIPO VARIÁVEL[i];
 - ▶ i é uma constante inteira que indica quantidade de posições;

```
1 int produtos [5];  
2 float precos [3];  
3 char palavra [50];  
4  
5 //variaveis como indices  
6 int i=5;  
7 int x[i];
```

```
int a[5];
```

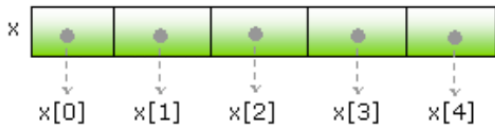


Vetor ou Array unidimensional

Acesso aos elementos

- CADA POSIÇÃO: 0 \rightarrow n-1

```
int x[5];
```



Valores na memória

Programa em execução (processo) x memória

- Toda vez que o programa entra em execução, seus dados são salvos (alocados) na memória RAM
- Área denominada de **stack**
- Alocação e desalocação: automática (SO - sistema operacional)
- Tamanho: limitado pelo SO
 - ▶ Linux: 8192 kB (`ulimit -s`)

Valores na memória

Alocação de variáveis

- Cada tipo ocupa uma quantidade distinta
- Alocação estática (tamanho e quantidade definido antes da execução)
- Cada variável possui um **endereço na memória**
 - ▶ Byte menos significativo - início da alocação
 - ▶ Alocação contínua

Valores na memória

Variáveis x Endereços

- Cada variável possui um endereço na memória
- Endereço = byte menos significativo (início da alocação)

```
int x = 9;
```

$$9_{10} = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1001_b$$

```
int: ocupa 32 bits (4 bytes)
```

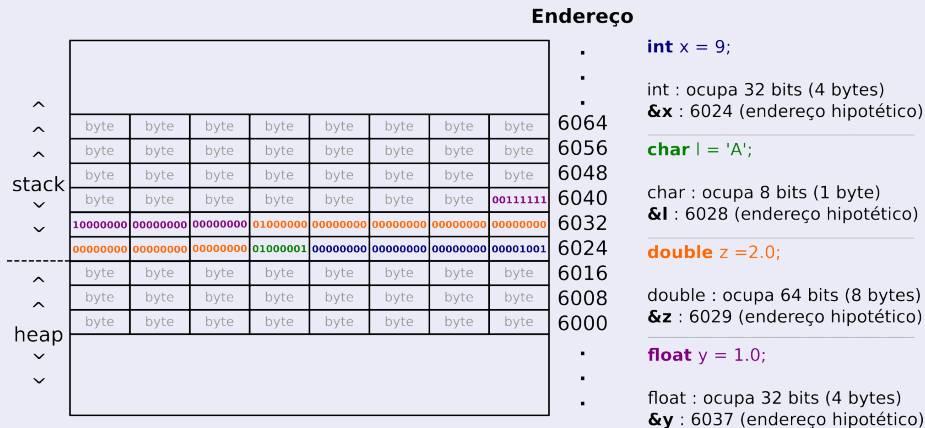
```
&x : 6024 (endereço hipotético)
```

Memória	00000000	00000000	00000000	00001001
Endereço	6027	6026	6025	6024

Valores na memória

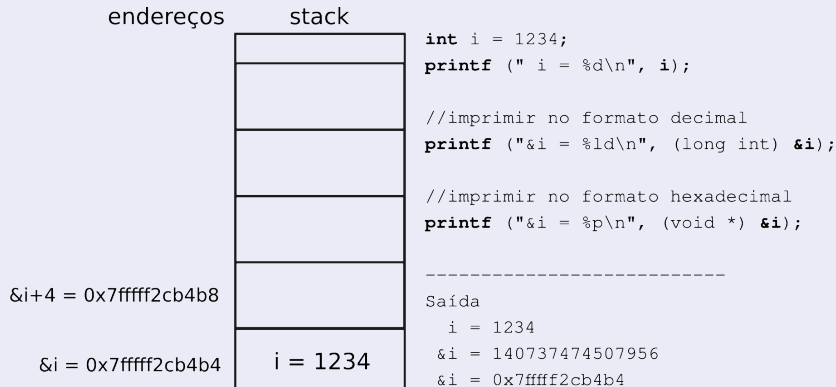
Variáveis x Endereços

- Alocação contínua



Valores na memória

Variáveis x Endereços



Valores na memória

Vetor x Endereços

- Posições de um vetor são alocados de forma contígua
- Cada posição tem um endereço
- Cada posição é calculada a partir do endereço inicial
- Endereço inicial, é apontado pelo identificador (nome) do vetor

```
int V[3];
```

V	V[0]	V[1]	V[2]
	??	??	??
V	&V[0]	&V[1]	&V[2]
1000	1004	1008	1012

Valores na memória

Vetor x Endereços

```
int V[3];
```

V	V[0]	V[1]	V[2]
---	------	------	------

??	??	??
----	----	----

V	&V[0]	&V[1]	&V[2]
---	-------	-------	-------

	V+0	V+1	V+2
--	-----	-----	-----

1000	1004	1008	1012
------	------	------	------

Valores na memória

Vetor x Endereços

```
int v[2];
v[0] = 3;
v[1] = 7;

printf (" endereco de v %ld\n", (long int)v);

printf (" v[0] = %d\n", v[0]);
printf ("&v[0] = %ld\n", (long int) &v[0]);
printf ("&v[0] = %p\n", (void *) &v[0]);

printf (" v[1] = %d\n", v[1]);
printf ("&v[1] = %ld\n", (long int) &v[1]);
printf ("&v[1] = %p\n", (void *) &v[1]);
```

Saída

endereco de v 140727096456912

v[0] = 3
&v[0] = 140727096456912
&v[0] = 0x7ffd949836d0

v[1] = 7
&v[1] = 140727096456916
&v[1] = 0x7ffd949836d4

endereços

&v[1] = 0x7ffd949836d4

v = &v[0] = 0x7ffd949836d0

stack

v[1] = 7
v[0] = 3

Vetor ou Array unidimensional

Inicialização na declaração

- Os arrays podem ser inicializados quando são definidos;
- Valores devem estar entre chaves ({ e }) e são separados por vírgula (,)

```
1 | int a[6]={10,20,40,45,30,12};
```

a	10	20	40	45	30	12
	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

Vetor ou Array unidimensional

Inicialização na declaração

```
1 float dinheiro[3] = {23.4, 123.0, 55.0};
2 char letras[4] = {'a', 'b', 'c', 'd'};
3
4 //o tamanho do array pode ser omitido quando inicializados
5 int peso[] = { 153, 135, 170 }; //compilador aloca
6
7
```


Vetor ou Array unidimensional

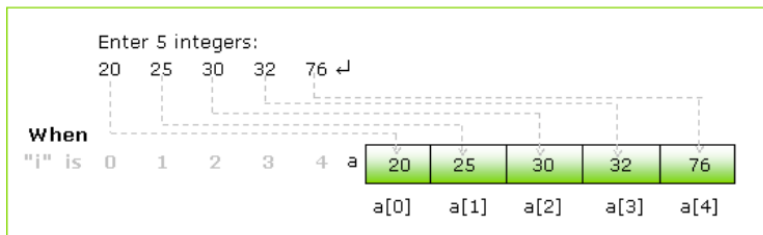
Inicialização na declaração - Erros

```
1 //variaveis com indice
2 int a=5;
3 int y[a] = { 1, 4, 6, 99, 2}; //erro de compilação
4
5 //so podem ser inicializados integralmente na declaracao
6 int erro[5];
7 erro = { 2, 4, 6, 8, 10 }; //erro
8
9 int b[]; //erro
10
```

Vetor ou Array unidimensional

Inicialização pela entrada padrão - scanf

```
1 printf("Enter 5 integers:\n");
2 for(i=0;i<5;i++)
3     scanf("%d",&a[i]);
```



Vetor ou Array unidimensional

Inicialização pela entrada padrão - scanf

```
1  float dinheiro[100]; //0 -> 99
2  char letras[4]; //0 -> 3
3
4  int i=0;
5  while(i<100) {
6      scanf("%f", &dinheiro[i]);
7      i++;
8  }
9
10 for(i=0; i<4; i++) {
11     scanf(" %c", &letras[i]);
12 }
13
```

Vetor ou Array unidimensional

Manipulando vetor pelo índice

```
1  int i, x, sala, total[5];
2
3  //atribuindo valores pelos indices
4  total[2] = 1;
5  total[3] = 2;
6
7  //acessando valor pelo indice
8  x = total[3];
9
10 //acessando e atribuindo valores pelos indices
11 i = 4;
12 total[i] = total[i-1] + total[i-2];
13 total[4]++;
14
15 printf("%d\n", total[4]); //Saída????!!!!
16
17 float area;
18 float tamanho[42];
19
20 //entrada de valor pelo indice
21 scanf("%f", &tamanho[41]);
```

Vetor ou Array unidimensional

Igualar vetores

```
1  int vetorA[10], vetorB[10];
2  int indice;
3
4  // inicializando o vetor A
5  for (indice = 0; indice < 10; indice++) {
6      scanf("%d", &vetorA[indice]);
7  }
8
9  // copiar o conteudo do vetor A para o vetor B
10 vetorB = vetorA;    //????
11
12
```

Vetor ou Array unidimensional

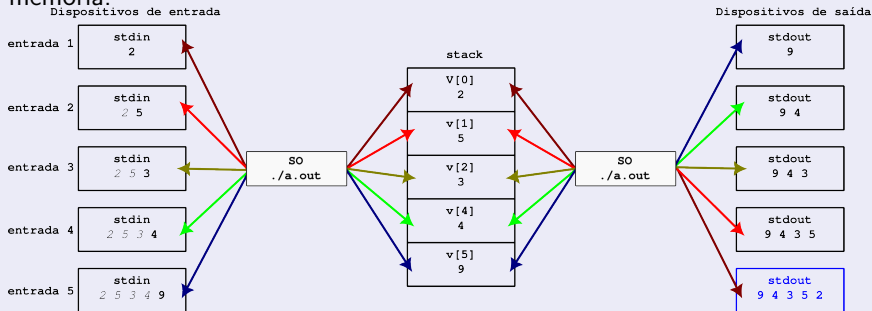
Igualar vetores

```
1  int vetorA[10], vetorB[10];
2  int indice;
3
4  // inicializando o vetor A
5  for (indice = 0; indice < 10; indice++) {
6      scanf("%d", &vetorA[indice]);
7  }
8
9  // copiar o conteudo do vetor A para o vetor B
10 vetorB = vetorA; // ERRADO!
11
12 // copiar o conteudo do vetor B para o vetor A
13 for (indice = 0; indice < 10; indice++) {
14     vetorB[indice] = vetorA[indice];
15 }
16
```

Vetor ou Array unidimensional

Exemplo de quando usar: necessidade de guardar os elementos

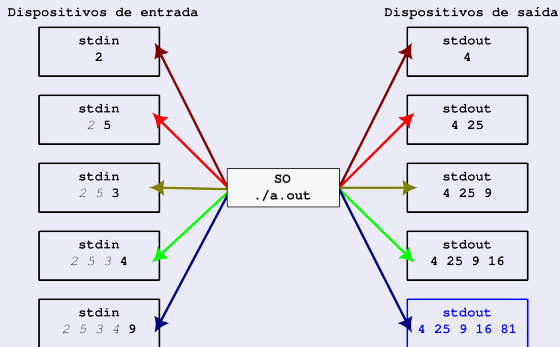
- Ler uma certa quantidade de valores inteiros e os imprimir na ordem inversa da leitura;
- Isto é, se os dados de entrada forem: 2, 5, 3, 4, 9, queremos imprimir na saída: 9, 4, 3, 5, 2;
- Este tipo de problema é impossível de ser resolvido com o uso de apenas uma variável pois, quando se lê o segundo número, já se perdeu o primeiro da memória.



Vetor ou Array unidimensional

Exemplo de quando não usar: processar em tempo de execução

- Ler uma certa quantidade de valores inteiros e os imprimir o quadrado dos valores;
- Isto é, se os dados de entrada forem: 2, 5, 3, 4, 9, queremos imprimir na saída: 4, 25, 9, 16, 81;
- Este tipo de problema é recomendado resolver logo após a leitura.



Vetor em Registros

```
1 typedef struct cadastroProduto Produto;
2 struct cadastroProduto {
3     int codigo;
4     float preco;
5     struct data entrada;
6     int tamanho_qtde[6]; //pp p m g gg xg
7 };
8
```

```
1 Produto camiseta;
2 scanf("%d %f %d/%d/%d", &camiseta.codigo,
3     &camiseta.preco,
4     &camiseta.entrada.dia,
5     &camiseta.entrada.mes,
6     &camiseta.entrada.ano);
7 for(int i=0; i<6; i++){
8     scanf("%d", &camiseta.tamanho_qtde[i]);
9 }
10
```

Vetor de Registros

```
1  Produto camisetas[10];
2  for(int j=0; j<10; j++){
3      scanf("%d %f %d/%d/%d", &camisetas[j].codigo,
4          &camisetas[j].preco,
5          &camisetas[j].entrada.dia,
6          &camisetas[j].entrada.mes,
7          &camisetas[j].entrada.ano);
8
9      for(int i=0; i<6; i++){
10         scanf("%d", &camisetas[j].tamanho_qtde[i]);
11     }
12 }
```

Vetor de Registros em Registros

```
1  typedef struct cor Cor;
2  typedef struct cadastroProduto Produto;
3
4  struct cor {
5      int codigo;
6      int qtde;
7  };
8
9  struct cadastroProduto {
10     int codigo;
11     float preco;
12     struct data entrada;
13     int tamanho_qtde[6];
14     Cor cores[10];
15 };
16
```

Vetor de Registros em Registros

```
1  Produto camisetas[10];
2  for(int j=0; j<10; j++){
3      scanf("%d %f %d/%d/%d", &camisetas[j].codigo,
4          &camisetas[j].preco,
5          &camisetas[j].entrada.dia,
6          &camisetas[j].entrada.mes,
7          &camisetas[j].entrada.ano);
8
9      for(int i=0; i<6; i++){
10         scanf("%d", &camisetas[j].tamanho_qtde[i]);
11     }
12
13     for(int i=0; i<10; i++){
14         scanf("%d %d", &camisetas[j].cores[i].codigo,
15             &camisetas[j].cores[i].qtde);
16     }
17 }
```

Vamos praticar

Ler um número N , seguido de N valores inteiros e os imprimir esses N inteiros na ordem inversa da leitura.

Vamos praticar

Ler um número N, seguido de N valores inteiros e os imprimir esses N inteiros na ordem inversa da leitura.

```
1  #include <stdio.h>
2  int main()
3  {
4      int i, n;
5      scanf("%d", &n);
6      int valor[n];
7      for(i=0; i<n; i++)
8      {
9          scanf("%d", &valor[i]);
10     }
11
12     for(i=n-1; i>=0; i--)
13     {
14         printf("%d ", valor[i]);
15     }
16
17     printf("\n");
18     return 0;
19 }
```

Vamos praticar

Ler um número N , seguido de um conjunto de N valores inteiros, seguido de outro conjunto de N inteiros e imprimir o produto dos elementos com o mesmo índice

Vamos praticar

Ler um número N, seguido de um conjunto de N valores inteiros, seguido de outro conjunto de N inteiros e imprimir o produto dos elementos com o mesmo índice

```
1  #include <stdio.h>
2  int main() {
3      int n;
4      scanf("%d", &n);
5      int v[n], i, x;
6      for(i=0; i<n; i++) {
7          scanf("%d", &v[i]);
8      }
9
10     for(i=0; i<n; i++) {
11         scanf("%d", &x);
12         printf("%d ", v[i]*x);
13     }
14
15     printf("\n");
16     return 0;
17 }
```


Vamos praticar

Inicializar dois vetores com 10 inteiros cada na seguinte ordem $\{2, 4, 1, 3, 5, 7, 6, 9, 8, 0\}$ e $\{5, 4, 1, 9, 7, 6, 45, 4, 12, 3\}$, e imprimir o produto escalar

v:	2	6	1	0	3	3
	×	×	×	×	×	×
w:	1	0	2	4	3	5
	=	=	=	=	=	=
	2	0	2	0	9	15

Os números obtidos a partir das multiplicações para todos os i fixos devem ser somados: $2 + 0 + 2 + 0 + 9 + 15 = 28$. Logo, 28 é o produto escalar de v e w .

Vamos praticar

Inicializar dois vetores com 10 inteiros cada na seguinte ordem {2, 4, 1, 3, 5, 7, 6, 9, 8, 0} e {5, 4, 1, 9, 7, 6, 45, 4, 12, 3}, e imprimir o produto escalar

```
1  #include <stdio.h>
2  int main()
3  {
4      int v[10] = {2, 4, 1, 3, 5, 7, 6, 9, 8, 0};
5      int z[] = {5, 4, 1, 9, 7, 6, 45, 4, 12, 3};
6      int i, prod=0;
7
8      for(i=0; i<10; i++)
9      {
10         prod = prod + v[i]*z[i];
11     }
12     printf("%d\n", prod);
13
14     return 0;
15 }
16
```

Vetor ou Array unidimensional

Busca: procura x em um vetor v com n posições

```
1 //funciona
2 int achou = 0, k = 0;
3 while (k < n && achou == 0) { //testa condicao
4     if (v[k] == x) achou = 1; //testa condicao (mais uma)
5     else k++;
6 }
7
```

Vetor ou Array unidimensional

Busca: procura x em um vetor v com n posições - soluções mais elegantes

```
1  int k = 0;
2  // unifica as condicoes
3  while (k < n && v[k] != x) k++;
4
```

```
1  int k = 0;
2  // problema?!
3  while (v[k] != x && k < n) k++;
4
```

```
1  int v[6] = {1, 3, 2, 5, 6}; //uma posicao a mais
2  int k = 0, x = 2;
3
4  v[5] = x; // sentinela
5  while (v[k] != x) k++;
6
7  //encontrou?!?!
8
```

Vamos praticar

Ler um vetor de 20 inteiros e em seguida um inteiro x . Imprimir “pertence” se x é um elemento do vetor e “não pertence” em caso contrário.

Vamos praticar

Ler um vetor de 20 inteiros e em seguida um inteiro x. Imprimir “pertence” se x é um elemento do vetor e “não pertence” em caso contrário.

```
1  #include <stdio.h>
2  int main()
3  {
4      int valor[10], i, x;
5
6      for(i=0; i<10; i++)
7          scanf("%d", &valor[i]);
8
9      scanf("%d", &x);
10     for(i=0; i<10 && valor[i]!=x; i++);
11
12     if(i<10) printf("pertence\n");
13     else printf("nao pertence\n");
14
15     return 0;
16 }
17
```

Vetor ou Array unidimensional

Inicializar dois vetores com 5 inteiros: $\{1, 2, 3, 4, 5\}$ e $\{1, 10, 100, 1000, 10000\}$. Imprimir o produto de todos os elementos entre sim.

- 1 Percorrer o primeiro vetor
- 2 Para cada elemento m do primeiro vetor
- 3 Percorrer o segundo vetor
- 4 Multiplicando m com cada elemento do segundo vetor

Vetor ou Array unidimensional

Inicializar dois vetores com 5 inteiros: {1, 2, 3, 4, 5} e {1, 10, 100, 1000, 10000}. Imprimir o produto de todos os elementos entre sim.

```
1  int v1[5] = {1, 2, 3, 4, 5};
2  int v2[5] = {1, 10, 100, 1000, 10000};
3
4  for(int i=0; i < 5; i++)
5  {
6      for(int j=0; j < 5 ; j++)
7      {
8          printf("%d ",v1[i] * v2[j]);
9      }
10     printf("\n");
11 }
12
13
```


Roteiro

- 1 Estruturas de Dados
 - Vetor ou Array unidimensional
 - **Strings**
 - Matriz ou Array multidimensional

String (sequência de caracteres)

Definição

Strings são arrays (vetores) de caracteres (arrays com elementos do tipo char) que DEVEM terminar com `'\0'`.

Inicialização na declaração

```
1     int i;  
2     char x[6]="maria"; //string  
3
```

- O compilador automaticamente coloca o `'\0'`;
- Lembre-se: sempre adicionar 1 no tamanho do array;

String (sequência de caracteres)

```
1 char nome1 [6]="maria"; //string
2 char nome2 [6]={ 'm', 'a', 'r', 'i', 'a', '\0' }; //string
3 char nome3 [5]={ 'm', 'a', 'r', 'i', 'a' }; //vetor de caracteres
4
5 //especificador de formato: %s
6 printf("%s, %s, %s\n", nome1, nome2, nome3); //maria maria maria
```

- Vantagem da string: fim bem definido

String (sequência de caracteres)

Leitura de strings x leitura de array de caracteres

```
1 char nome1[100];
2 char nome2[100];
3 char c;
4
5 //Lendo uma palavra em uma string
6 scanf("%s", nome1); //sem & : vetor = endereco da 1a posicao
7                       //lê 1(uma) seq. de caracteres (palavra)
8                       //ate encontrar ' ', \t, \n
9                       //scanf coloca \0 no final
10
11 //Lendo uma palavra em um vetor de caracteres
12 scanf(" %c", &c); //por que o ' ' antes do %c?
13 for(int i=0; i<100 && c!=' ' && c!='\n' && c!='\t'; i++){
14     nome2[i] = c;
15     scanf("%c", &c); //por que não tem ' ' antes do %c?
16 }
17
18 //especificador de formato: %s
19 printf("%s e %s\n", nome1, nome2);
```

String (sequência de caracteres)

Lendo uma quantidade limitada de caracteres

```
1 char nome1[100];
2 char nome2[100];
3
4 scanf("%s", nome1); //sem & : vetor = endereço da 1a posição
5                     //lendo 1 sequência de caracteres (palavra)
6                     //até encontrar ' ', \t, \n
7
8 scanf("%99s", nome2); //le no máximo 99 caracteres seguidos
9                       //ou até encontrar ' ', \t, \n
10
11 //especificador de formato: %s
12 printf("%s e %s\n", nome1, nome2);
```

- Dadas as entradas:

João

José Paulo

Qual a saída? 1 ou 2?

String (sequência de caracteres)

Lendo uma quantidade limitada de caracteres

```
1 char nome1[100];
2 char nome2[100];
3
4 scanf("%s", nome1); //sem & : vetor = endereço da 1a posição
5                     //lendo 1 sequência de caracteres (palavra)
6                     //até encontrar ' ', \t, \n
7
8 scanf("%99s", nome2); //le no máximo 99 caracteres seguidos
9                     //ou até encontrar ' ', \t, \n
10
11 //especificador de formato: %s
12 printf("%s e %s\n", nome1, nome2);
```

- Dadas as entradas:

- ▶ João

- ▶ José Paulo

- Qual a saída? 1 ou 2?

String (sequência de caracteres)

Lendo uma quantidade limitada de caracteres

```
1 char nome1[100];
2 char nome2[100];
3
4 scanf("%s", nome1); //sem & : vetor = endereço da 1a posição
5                     //lendo 1 sequência de caracteres (palavra)
6                     //até encontrar ' ', \t, \n
7
8 scanf("%99s", nome2); //le no máximo 99 caracteres seguidos
9                     //ou até encontrar ' ', \t, \n
10
11 //especificador de formato: %s
12 printf("%s e %s\n", nome1, nome2);
```

- Dadas as entradas:

- ▶ João
- ▶ José Paulo

Qual a saída? 1 ou 2?

String (sequência de caracteres)

Lendo uma quantidade limitada de caracteres

```
1 char nome1[100];
2 char nome2[100];
3
4 scanf("%s", nome1); //sem & : vetor = endereço da 1a posição
5                     //lendo 1 sequência de caracteres (palavra)
6                     //até encontrar ' ', \t, \n
7
8 scanf("%99s", nome2); //le no máximo 99 caracteres seguidos
9                     //ou até encontrar ' ', \t, \n
10
11 //especificador de formato: %s
12 printf("%s e %s\n", nome1, nome2);
```

- Dadas as entradas:
 - ▶ João
 - ▶ José Paulo
- Qual a saída? 1 ou 2?

1 João e José

2 João e José

String (sequência de caracteres)

Lendo uma quantidade limitada de caracteres

```
1 char nome1[100];
2 char nome2[100];
3
4 scanf("%s", nome1); //sem & : vetor = endereço da 1a posição
5                       //lendo 1 sequência de caracteres (palavra)
6                       //até encontrar ' ', \t, \n
7
8 scanf("%99s", nome2); //le no máximo 99 caracteres seguidos
9                       //ou até encontrar ' ', \t, \n
10
11 //especificador de formato: %s
12 printf("%s e %s\n", nome1, nome2);
```

- Dadas as entradas:
 - ▶ João
 - ▶ José Paulo
- Qual a saída? 1 ou 2?
 - ① João e José

String (sequência de caracteres)

Lendo uma quantidade limitada de caracteres

```
1 char nome1[100];
2 char nome2[100];
3
4 scanf("%s", nome1); //sem & : vetor = endereco da 1a posicao
5                     //lendo 1 sequencia de caracteres (palavra)
6                     //ate encontrar ' ', \t, \n
7
8 scanf("%99s", nome2); //le no maximo 99 caracteres seguidos
9                       //ou ate encontrar ' ', \t, \n
10
11 //especificador de formato: %s
12 printf("%s e %s\n", nome1, nome2);
```

- Dadas as entradas:
 - ▶ João
 - ▶ José Paulo
- Qual a saída? 1 ou 2?
 - 1 João e José
 - 2 João e José

String (sequência de caracteres)

Lendo frases

```
1 char nome1[100];  
2  
3 //leia tudo menos a quebra de linha  
4 //leia tudo até a quebra de linha  
5 scanf("%99[^\n]", nome1);  
6  
7 printf("%s\n", nome1);
```

Exemplo:

Entradas

Jose da Silva

Saída:

Jose da Silva

String (sequência de caracteres)

Lendo palavras e frases

```
1 char nome1[100], nome2[100], nome3[100];
2
3 scanf("%99s", nome1); //lê no maximo 99 caracteres seguidos
4 scanf("%99s", nome2); //ou até encontrar ' ', \t, \n
5
6 //lê o restante da entrada anterior inclusive \n
7 scanf("%99[^\n]", nome3); //leia tudo até a quebra de linha
8
9 printf("1 %s\n", nome1);
10 printf("2 %s\n", nome2);
11 printf("3 %s\n", nome3);
```

Exemplo: qual a saída correta?

Entradas	Saída 1:	Saída 2:	Saída 3:	Saída 4:
Joao	1 Joao	1 Joao	1 Joao	1 Joao
Maria Luisa	2 Maria Luisa	2 Maria Luisa	2 Maria	2 Maria
Jose da Silva	3 Jose da Silva	3 Jose	3 Luisa	3 Luisa Jose da Silva

String (sequência de caracteres)

Lendo palavras e frases

```
1 char nome1[100], nome2[100], nome3[100], nome4[100];
2
3 scanf("%99s", nome1); //le no maximo 99 caracteres seguidos
4 scanf("%99s", nome2); //ou ate encontrar ' ', \t, \n
5
6 // "consumir" o \n anterior: utilize um espaço
7 //      |
8 scanf(" %99[^\n]", nome3);
9 scanf(" %99[^\n]", nome4);
10
11 printf("1 %s\n", nome1);
12 printf("2 %s\n", nome2);
13 printf("3 %s\n", nome3);
14 printf("4 %s\n", nome4);
```

Exemplo:

Entradas

Joao
Maria
Jose da Silva
Antonio Souza

Saída:

1 Joao
2 Maria
3 Jose da Silva
4 Antonio Souza

String (sequência de caracteres)

Lendo várias strings seguidas

```
1 char nome1[100], nome2[100], c;
2 int i;
3
4 for(i=0;i<4;i++) {
5     scanf("%99s", nome1); //sequencia de palavras
6     printf("%s\n", nome1);
7 }
8
9 // "pular" separadores(espacos, tab, \n) anteriores
10 for(i=0;i<4;i++) {
11     //     |
12     scanf(" %99[^\n]", nome2); //sequencia de frases
13     printf("%s\n", nome2);
14 }
15
16 for(i=0;i<4;i++) {
17     //     |
18     scanf(" %c", &c);
19     printf("%c\n", c);
20 }
```

String (sequência de caracteres)

Percorrendo a string

```
1 char nome1[100];
2 scanf("%s", nome1); //sem & : vetor = endereco da 1a posicao
3
4 i = 0;
5 while(nome1[i] != '\0')
6 {
7     //especificador %c
8     printf("%c", nome1[i]);
9     i++;
10 }
11 printf("\n");
```

String (sequência de caracteres)

Vamos praticar

- Faça um algoritmo que leia e imprima uma palavra (string sem espaços) e depois uma frase (com espaços).
- Considere 100 o tamanho máximo da frase;
- Considere 15 o tamanho máximo da palavra;
- Um exemplo de execução:

Entrada:

teste

Ola mundo

Saída:

teste

Ola mundo

String (sequência de caracteres)

Vamos praticar

```
1 #include <stdio.h>
2
3 int main() {
4     char palavra[16], frase[101];
5     scanf("%15s", nome1); //sem & : vetor = endereco da 1a
6     //      |
7     //      |
8     scanf(" %100[^\n]", frase);
9
10    printf("%s\n", palavra);
11    printf("%s\n", frase);
12
13    return 0;
14 }
```

String (sequência de caracteres)

Vamos praticar

- Faça um algoritmo que leia e imprima o comprimento de uma string.
- Considere 15 o tamanho máximo da palavra;
- Por exemplo, o comprimento do string “alo” é 3;
- Um exemplo de execução:

Entrada:

Dostoevsky

Saída:

10

String (sequência de caracteres)

Vamos praticar

```
1 #include <stdio.h>
2
3 int main() {
4     char nome1[16];
5     scanf("%15s", nome1); //sem & : vetor = endereco da 1a
6                             posicao
7
8     int i = 0;
9     while(s[i] != '\0')
10         i++;
11
12     printf("%d\n", i);
13
14     return 0;
15 }
```

String (sequência de caracteres)

Vamos praticar

- Implemente um algoritmo que compare duas string;
- Imprima:
 - ▶ 0 se forem iguais;
 - ▶ Diferença entre o primeiro caractere distinto entre si:
 - ★ Um número negativo se S1 for menor que S2 (“S1 vem antes de S2 no dicionário” - tabela ASCII; exemplo: “alo” e “ola”, ‘a’ - ‘o’ = -14
 - ★ Um número positivo se S1 for maior que S2 (“S1 vem depois de S2 no dicionário” - tabela ASCII; exemplo: “string” e “compare”, ‘s’-‘c’ = 16
- Considere 15 o tamanho máximo da palavra;
- Um exemplo de execução:

Entrada:

ele

ela

Saída:

4

String (sequência de caracteres)

Vamos praticar

```
1  #include <stdio.h>
2
3  int main() {
4      char s1[16], s2[16];
5      scanf("%s %s", s1, s2);
6      int i=0;
7      while(s1[i]!='\0' && s2[i]!='\0' && s1[i]==s2[i])
8          i++;
9
10     int diferenca = s1[i] - s2[i];
11     printf("%d\n", diferenca);
12     return 0;
13 }
14
```

String (sequência de caracteres)

Vamos praticar

- Implemente um algoritmo que copie uma string em outra;
- Leia uma frase e imprima a cópia;
- Considere 50 o tamanho máximo da frase;
- Um exemplo de execução:

Entrada:

Ola mundo

Saída:

Ola mundo

String (sequência de caracteres)

Vamos praticar

```
1 #include <stdio.h>
2
3
4 int main() {
5     char s[51], sCopia[51];
6     scanf("%50[^\n]", s);
7     int i;
8     for (i = 0; s[i] != 0; i++)
9         sCopia[i] = s[i];
10
11     sCopia[i] = '\0';
12
13     //for (i = 0; (sCopia[i]=s[i]) != 0; i++);
14     printf("%s\n", sCopia);
15
16     return 0;
17 }
```

man string

String (sequência de caracteres)

Vamos praticar

- Faça um algoritmo que leia um número N que indica quantas frases serão lidas. Nas N linhas seguintes leia uma frase e imprima na tela seu comprimento.
- Considere 50 o tamanho máximo da frase;

- Entrada:

3

Teste de string 1

Teste de string

Teste

Saída:

17

15

5

String (sequência de caracteres)

Vamos praticar

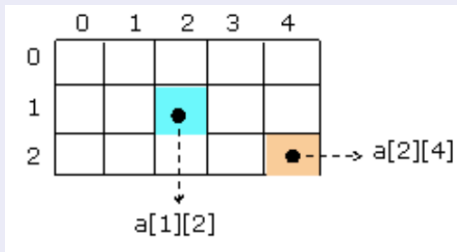
```
1  int n;  
2  scanf("%d", &n);  
3  
4  char s[51];  
5  
6  int i, j;  
7  for (i = 0; i<n; i++)  
8  {  
9      scanf(" %50[^\n]", s);  
10     for (j=0; s[j]!='\0'; j++);  
11     printf("%d\n", j);  
12 }  
13  
14
```

- 1 Estruturas de Dados
 - Vetor ou Array unidimensional
 - Strings
 - Matriz ou Array multidimensional

Matriz ou Array multidimensional

Definição e Declaração

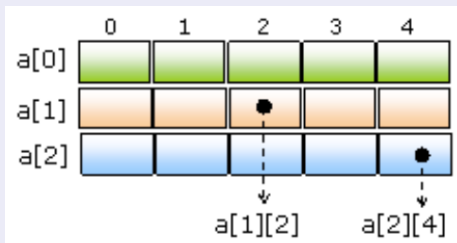
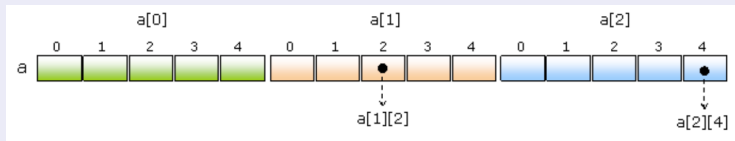
- Assim como os vetores, as matrizes são arrays;
- Os vetores são estruturas unidimensionais enquanto que as matrizes são bidimensionais;
- O acesso às posições de memória de um vetor são feitas com base em duas informações: o nome da variável e o deslocamento;
- Para se acessar os elementos de uma matriz, precisa-se do nome da variável, do deslocamento lateral e do deslocamento vertical.



Matriz ou Array multidimensional

Declaração

```
1  int a[3][5];  
2
```



Matriz ou Array multidimensional

Inicialização na declaração e por atribuição

```
1  int a  
   [3][5]={{10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};  
2  int a  
   [][5]={{10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};  
3  int a[][5]={10,6,7,12,11,23,32,14,52,22,33,17,18,54,28};  
4  a[0][0] = 3;  
5  a[1][4] = 12;  
6  int b[2][2] = {}; //??  
7
```

	0	1	2	3	4
0	10	6	7	12	11
1	23	32	14	52	22
2	33	17	18	54	28

Matriz ou Array multidimensional

Inicialização pela entrada padrão

```
1 int a[3][5];
2
3 int col;
4 //linha 0
5 for(col=0; col<5; col++)
6 {
7     scanf("%d", &a[0][col]);
8 }
9
10 //linha 1
11 for(col=0; col<5; col++)
12 {
13     scanf("%d", &a[1][col]);
14 }
15
16 //linha 2
17 for(col=0; col<5; col++)
18 {
19     scanf("%d", &a[2][col]);
20 }
```

Matriz ou Array multidimensional

Inicialização pela entrada padrão

```
1 int a[3][5];
2
3 int lin, col;
4 //para cada linha
5 for(lin=0; lin<3; lin++)
6 {
7     //e cada coluna
8     for(col=0; col<5; col++)
9     {
10         scanf("%d", &a[lin][col]);
11     }
12 }
```

Matriz ou Array multidimensional

Impressão: percorrendo a matriz

```
1 int a[3][5]={10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};
2
3 int col;
4 //imprimindo a primeira linha
5 for(col=0; col<5; col++)
6     printf("%5d", a[0][col]);
7
8 printf("\n");
9
10 //imprimindo a segunda linha
11 for(col=0; col<5; col++)
12     printf("%5d", a[1][col]);
13
14 printf("\n");
15
16 //imprimindo a terceira linha
17 for(col=0; col<5; col++)
18     printf("%5d", a[2][col]);
19
20 printf("\n");
```


Matriz ou Array multidimensional

Impressão: percorrendo a matriz

```
1 int a[3][5]={10,6,7,12,11},{23,32,14,52,22},{33,17,18,54,28}};
2
3 int lin, col;
4 for(lin=0; lin<3; lin++)
5 {
6     for(col=0; col<5; col++)
7     {
8         printf("%5d", a[lin][col]);
9     }
10    printf("\n");
11 }
```

Saída:

```
10 6 7 12 11
23 32 14 52 22
33 17 18 54 28
```

Matriz ou Array multidimensional

Faça um algoritmo que leia duas matrizes A e B, com tamanho 3x5, e imprima a soma A+B

$$A + B = \begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{bmatrix}.$$

Matriz ou Array multidimensional

```
1 int a[3][5];
2
3 int lin, col;
4 for(lin=0; lin<3; lin++)
5     for(col=0; col<4; col++)
6         scanf("%d", &a[lin][col]);
7
8 for(lin=0; lin<3; lin++)
9     for(col=0; col<4; col++)
10        scanf("%d", &b[lin][col]);
11
12 for(lin=0; lin<3; lin++)
13 {
14     for(col=0; col<4; col++)
15     {
16         printf("%d ", a[lin][col] + b[lin][col]);
17     }
18     printf("\n");
19 }
```

Matriz ou Array multidimensional

Dada a matriz abaixo e imprimir sua transposta

- Matriz:

```
12 45 67 88  
32 56 44 23  
78 56 33 89
```

- Transposta:

```
12 32 78  
45 56 56  
67 44 33  
88 23 89
```

Matriz ou Array multidimensional

Dada a matriz abaixo e imprimir sua transposta

```
1 int v[3][4] = { {12, 45, 67, 88},
2                 {32, 56, 44, 23},
3                 {78, 56, 33, 89} };
4 int col, lin;
5
6 for(col=0; col<4; col++)
7 {
8     for(lin=0; lin<3; lin++)
9     {
10        printf("%d ", v[lin][col]);
11    }
12    printf("\n");
13 }
```

Matriz ou Array multidimensional

Encontrar e imprimir o menor elemento de uma matriz

```
-4 4 18  
-2 0 4  
-9 -2 13
```

Matriz ou Array multidimensional

Encontrar e imprimir o menor elemento de uma matriz

```
1 int v[][3] = {{-4, 4, 18}, {-2, 0, 4}, {-9, -2, 13}};
2 int menor = v[0][0];
3 int i, j;
4 for(i=0; i<3; i++)
5 {
6     for(j=0; j<3; j++)
7     {
8         if(v[i][j] < menor)
9             menor = v[i][j];
10    }
11 }
12 printf("%d\n", menor);
```

Matriz ou Array multidimensional

Encontrar e imprimir o menor elemento de uma matriz

```
1 int v[][3] = {{-4, 4, 18}, {-2, 0, 4}, {-9, -2, 13}};
2 int im=0, jm=0;
3 int i, j;
4 for(i=0; i<3; i++)
5 {
6     for(j=0; j<3; j++)
7     {
8         if(v[i][j] < v[im][jm])
9             im=i;
10            jm=j;
11    }
12 }
13 printf("%d\n", v[im][jm]);
```


Matriz ou Array multidimensional

Dada uma matriz de dimensões 4x4, vamos gerar uma segunda matriz a partir da primeira onde cada elemento é a média da soma dele com três de seus vizinhos na matriz original

- Exemplo, matriz 4x4:

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

- Primeiro elemento é a média da sub-matriz: $(4+6+9+0)/4 = 4.75$

```
4 6
9 0
```

- Segundo elemento é a média da sub-matriz: $(2+1+0+2)/4 = 1.25$

```
2 1
0 2
```

- E assim por diante. No final, queremos produzir a seguinte matriz:

```
4.75 1.25
4.50 4.75
```

- Considere matrizes $m \times n$, sendo m e n pares.

Matriz ou Array multidimensional

Dada uma matriz de dimensões 4x4, vamos gerar uma segunda matriz a partir da primeira onde cada elemento é a média da soma dele com três de seus vizinhos na matriz original

```
1 int v[4][4] = { {4, 6, 2, 1},
2                 {9, 0, 0, 2},
3                 {8, 7, 3, 9},
4                 {1, 2, 3, 4} };
5 int i, j;
6
7 for(i=0; i<4; i+=2)
8 {
9     for(j=0; j<4; j+=2)
10    {
11        int s = v[i][j]    + v[i][j+1] +
12                v[i+1][j] + v[i+1][j+1];
13        printf("%.2f ", s/4.0);
14        printf("\n");
15    }
16 }
```

Matriz ou Array multidimensional

Faça um algoritmo que leia duas matrizes A (3x2) e B (2x3), e imprima a multiplicação AxB

$$A \cdot B = \begin{bmatrix} 2 & 3 \\ 0 & 1 \\ -1 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \\ -2 & 0 & 4 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 + 3(-2) & 2 \cdot 2 + 3 \cdot 0 & 2 \cdot 3 + 3 \cdot 4 \\ 0 \cdot 1 + 1(-2) & 0 \cdot 2 + 1 \cdot 0 & 0 \cdot 3 + 1 \cdot 4 \\ -1 \cdot 1 + 4(-2) & -1 \cdot 2 + 4 \cdot 0 & -1 \cdot 3 + 4 \cdot 4 \end{bmatrix} = \begin{bmatrix} -4 & 4 & 18 \\ -2 & 0 & 4 \\ -9 & -2 & 13 \end{bmatrix}$$

Matriz ou Array multidimensional

Faça um algoritmo que leia duas matrizes A (3x2) e B (2x3), e imprima a multiplicação AxB

■ 1ª linha e 1ª coluna

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 1 \cdot (-1) + 2 \cdot 4 & \\ & \end{bmatrix} \quad C_{11}$$

■ 1ª linha e 2ª coluna

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 1 \cdot (-1) + 2 \cdot 4 & 1 \cdot 3 + 2 \cdot 2 \\ & \end{bmatrix} \quad C_{12}$$

Matriz ou Array multidimensional

Faça um algoritmo que leia duas matrizes A (3x2) e B (2x3), e imprima a multiplicação $A \times B$

- Para cada linha de A
 - ▶ Para cada coluna de B
 - ★ Para cada coluna de A e linha de B
 - Multiplicar os elemento de A e B
 - Somar todas essas multiplicações
 - ★ Imprimir a soma das multiplicações

Matriz ou Array multidimensional

```
1 int A[3][2], B[2][3];
2 int lA, cB, i;
3 int soma_mult;
4
5 for(lA=0; lA<3; lA++)
6     for(i=0; i<2; i++)
7         scanf("%d", &A[lA][i]);
8
9 for(i=0; i<2; i++)
10    for(cB=0; cB<3; cB++)
11        scanf("%d", &B[i][cB]);
12
13 for(lA=0; lA<3; lA++) {
14     for(cB=0; cB<3; cB++) {
15         soma_mult = 0;
16         for(i=0; i<2; i++)
17             soma_mult = soma_mult + A[lA][i] * B[i][cB];
18
19         printf("%d ", soma_mult);
20     }
21     printf("\n");
22 }
```

Matriz ou Array multidimensional

Verifica se uma matriz tem elementos repetidos

```
-4 4 18  
-2 0 4  
-9 -2 13
```

Matriz ou Array multidimensional

```
1 int v[][3] = {{-4, 4, 18}, {-2, 0, 4}, {-9, -2, 13}};
2 int repetiu=0, i, j, p, q;
3
4 for(i=0; i<3 && !repetiu; i++)
5 {
6     for(j=0; j<3 && !repetiu; j++)
7     {
8         for(p=0; p<3 && !repetiu; p++)
9         {
10            for(q=0; q<3 && !repetiu; q++)
11            {
12                if((v[p][q]==v[i][j]) && ((p!=i) || (q!=j)))
13                    repetiu=1;
14            }
15        }
16    }
17 }
```


Matriz ou Array multidimensional

Dada a matriz abaixo e imprimir conforme exemplo

- Matriz:

12 45 67 88

32 56 44 23

78 56 33 89

- Saída:

88 67 45 12

23 44 56 32

89 33 56 78

78 56 33 89

32 56 44 23

12 45 67 88

88 23 89

67 44 33

45 56 56

12 32 78

Matriz x String

Array de Strings

```
1
2 int main()
3 {
4     char nomes[5][20] = {
5         "Jose Silva",
6         "Maria Silva",
7         "Antonio dos Santos",
8         "Pedro dos Santos",
9         "Joao da Silva"};
10    int i;
11
12    for(i = 0; i < 5; i += 1)
13        printf("%s\n", nomes[i]);
14 }
15
```

Jose Silva
Maria Silva
Antonio dos Santos
Pedro dos Santos
Joao da Silva

Matriz x String

Array de Strings

```
1
2 int main()
3 {
4     char nomes [5][20];
5     int i;
6     for(i=0;i<5;i++) {
7         // |
8         scanf("%99[^\n]", nomes[i]);
9     }
10
11     for(i = 0; i < 5; i += 1) {
12         printf("%s\n", nomes[i]);
13     }
14 }
15
```

Jose Silva

Maria Silva

Antonio dos Santos

Pedro dos Santos

Joao da Silva

Matriz x String

Vamos praticar

- Faça um algoritmo que leia um número N que indica quantas frases serão lidas. Nas N linhas seguintes leia uma frase em uma matriz de strings S. Após ler todas as frases, imprima as frases na ordem inversa a lida.
- Considere 50 o tamanho máximo da frase;

- Entrada:

3

Teste de string 1

Teste de string 2

Teste de string 3

Saída:

Teste de string 3

Teste de string 2

Teste de string 1

String (sequência de caracteres)

Vamos praticar

```
1 int n;  
2 scanf("%d", &n);  
3  
4 char s[n][51];  
5  
6 int i;  
7 for (i = 0; i<n; i++)  
8 {  
9     scanf(" %50[^\n]", s[i]);  
10 }  
11 for (i = n-1; i>=0; i--)  
12 {  
13     printf("%s\n", s[i]);  
14 }
```

Vamos praticar

- 1 Crie uma estrutura para armazenar dados de uma pessoa seguintes campos: nome (máx. 50 caracteres) e idade;
- 2 Crie uma estrutura para armazenar dados de alunos com os seguintes campos: dados pessoais e registro acadêmico;
- 3 Crie um programa que:
 - ▶ Declare um vetor para armazenar 3 alunos;
 - ▶ Leia os dados (nome, idade e registro acadêmico) de 3 alunos;
 - ▶ Leia um número x;
 - ▶ Imprima os nomes dos alunos com idade menor que x.

```
1 //Declaracao do tipo
2 struct nome_estrutura {
3     tipo_campo1 nomeCampo1;
4     tipo_campo2 nomeCampo2;
5     ...
6 };
7
8 //Declaracao das variaveis
9 struct nome_estrutura var1, var2;
10
11 //Acessar os campos '.' (ponto)
12 scanf("%d", &var1.campo1);
13 printf("%d\n", var1.campo1);
```

```
1 struct pessoa {
2     char nome[51];
3     int idade;
4 };
5 struct ficha_cadastral {
6     struct pessoa info;
7     int ra;
8 };
9
10 int main() {
11     struct ficha_cadastral alunos[3];
12     int i, m;
13     for(i=0; i<3; i++) {
14         scanf("%[^\n]", alunos[i].info.nome);
15         scanf("%d %d", &alunos[i].info.idade, &alunos[i].ra);
16     }
17
18     scanf("%d", &m);
19     for(i=0; i<3; i++) {
20         if(alunos[i].info.idade < m) {
21             printf("%s\n", alunos[i].info.nome);
22         }
23     }
24     return 0;
25 }
```

Vamos praticar

- 1 Faça uma estrutura para armazenar os seus pokemons com os campos: nome, tipo e nível;
- 2 Implemente um programa que armazene os seus pokemons e depois imprima o pokemon com o maior nível e o pokemon com o menor nível.
- 3 Faça o seu programa ler da entrada padrão os pokemons, seguindo o seguinte formato de entrada:

- ▶ Um número N, indicando a quantidade de pokemons a serem lidos, seguido de N linhas com os dados dos pokemons.

4

Pikachu eletrico 3

Bulbasaur grama 4

Catterpie verme 10

Charmander fogo 6

- ▶ Após a leitura deve-se imprimir os pokemons de maior e menor nível, segue como deve ser a saída:

Catterpie verme 10

Pikachu eletrico 3


```
1 struct pokemon {
2     char nome[50], tipo[50];
3     int nivel;
4 };
5
6 int main() {
7     int n, i, ma=0, me=0;
8     scanf("%d", &n);
9     struct pokemon pks[n];
10
11    for(i=0; i<n; i++) {
12        scanf("%s %s %d", pks[i].nome,
13              pks[i].tipo,
14              &pks[i].nivel);
15
16        if(pks[i].nivel > pks[ma].nivel) {
17            ma=i;
18        }
19        if(pks[i].nivel < pks[me].nivel) {
20            me=i;
21        }
22    }
23
24    printf("%s %s %d\n", pks[ma].nome, pks[ma].tipo, pks[ma].
25           nivel);
26    printf("%s %s %d\n", pks[me].nome, pks[me].tipo, pks[me].
27           nivel);
```

Vamos praticar

- 1 Crie uma estrutura para armazenar uma frase e seu autor. Faça um algoritmo que leia um número N que indica quantas frases serão lidas. Nas N linhas seguintes leia uma frase. E mais N linhas lendo o autor de cada frase. Após ler todas as frases, imprima na ordem inversa as frases e seus respectivos autores, conforme exemplo abaixo.

Entrada:

3

Teste de string 1

Teste de string 2

Teste de string 3

Autor 1

Autor 2

Autor 3

Saída:

Teste de string 3 - Autor 3

Teste de string 2 - Autor 2

Teste de string 1 - Autor 1

Vamos praticar

```
1 struct frases {
2     char frase[100];
3     char autor[50];
4 };
5
6 int main() {
7     int n, i;
8     scanf("%d", &n);
9     struct frases f[n];
10
11     for(i=0; i<n; i++) {
12         scanf(" %[^\n]", f[i].frase);
13     }
14
15     for(i=0; i<n; i++) {
16         scanf(" %[^\n]", f[i].autor);
17     }
18
19     for(i=n-1; i>=0; i--) {
20         printf("%s - %s\n", f[i].frase, f[i].autor);
21     }
22
23     return 0;
24 }
```