Vetor e Matriz

Roteiro

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;

Roteiro

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 VARIAVEL[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];
```

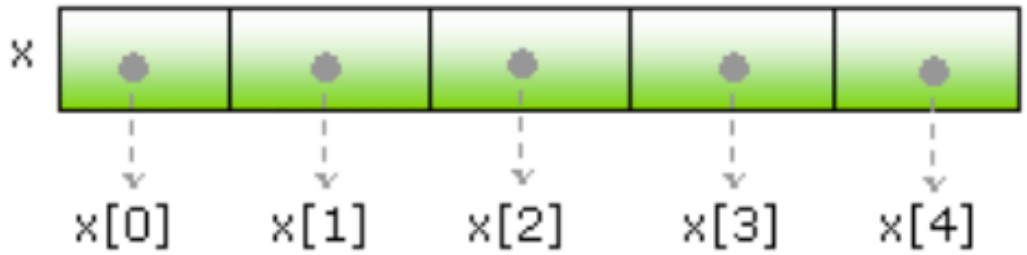
```
int a[5];
```

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;
```

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

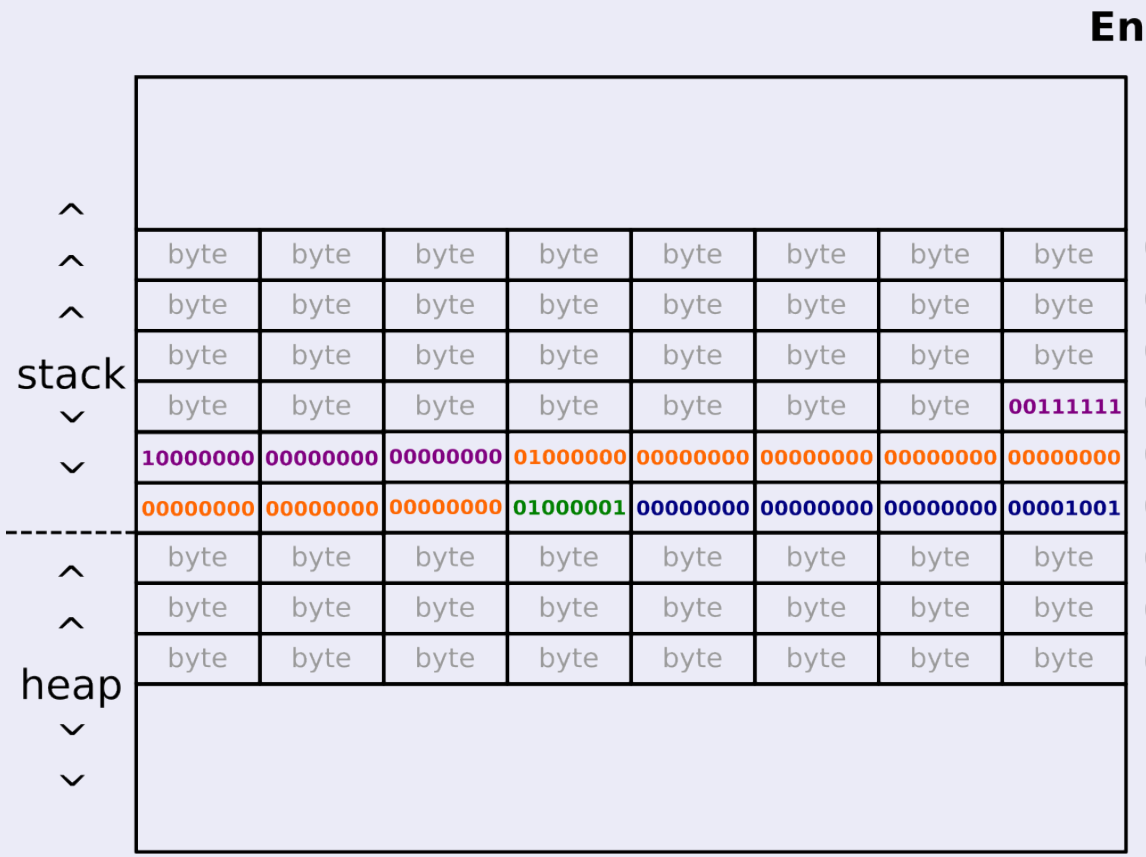
```
int: ocupa 32 bits (4 bytes)
&x : 6024 (endereço hipotético)
```

Memória	00000000	00000000	00000000	00001001
Endereço	6027	6026	6025	6024

Valores na memória

Variáveis x Endereços

- Alocação contínua



int x = 9;

int : ocupa 32 bits (4 bytes)
&x : 6024 (endereço hipotético)

---

char l = 'A';

char : ocupa 8 bits (1 byte)
&l : 6028 (endereço hipotético)

---

double z =2.0;

double : ocupa 64 bits (8 bytes)
&z : 6029 (endereço hipotético)

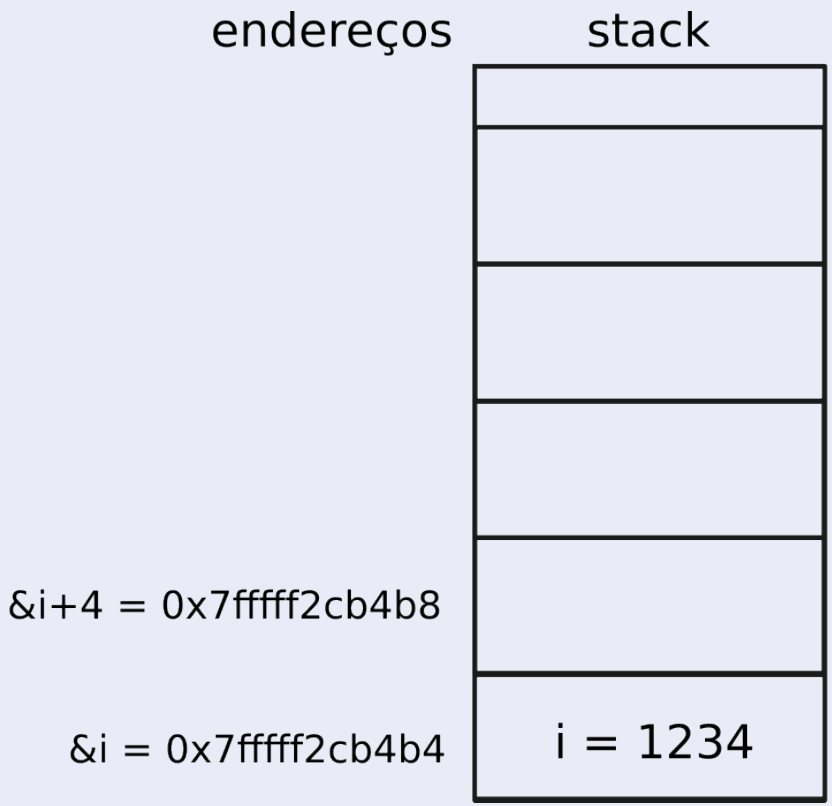
---

float y = 1.0;

float : ocupa 32 bits (4 bytes)
&y : 6037 (endereço hipotético)

Valores na memória

Variáveis x Endereços



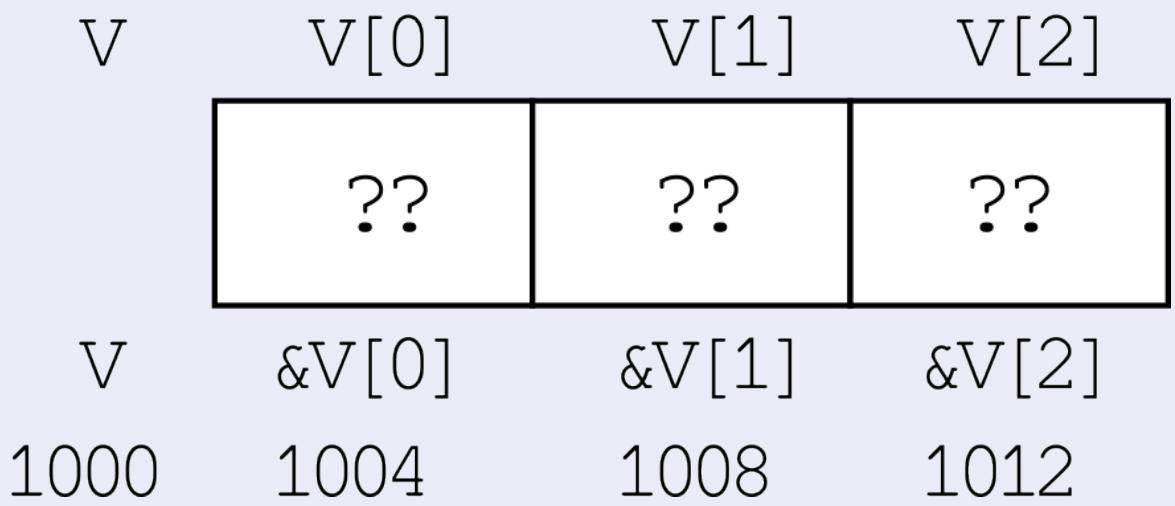

```
int i = 1234;
printf (" i = %d\n", i);

//imprimir no formato decimal
printf ("&i = %ld\n", (long int) &i);

//imprimir no formato hexadecimal
printf ("&i = %p\n", (void *) &i);

---------------------------
Saída
  i = 1234
 &i = 140737474507956
 &i = 0x7fffff2cb4b4
```

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

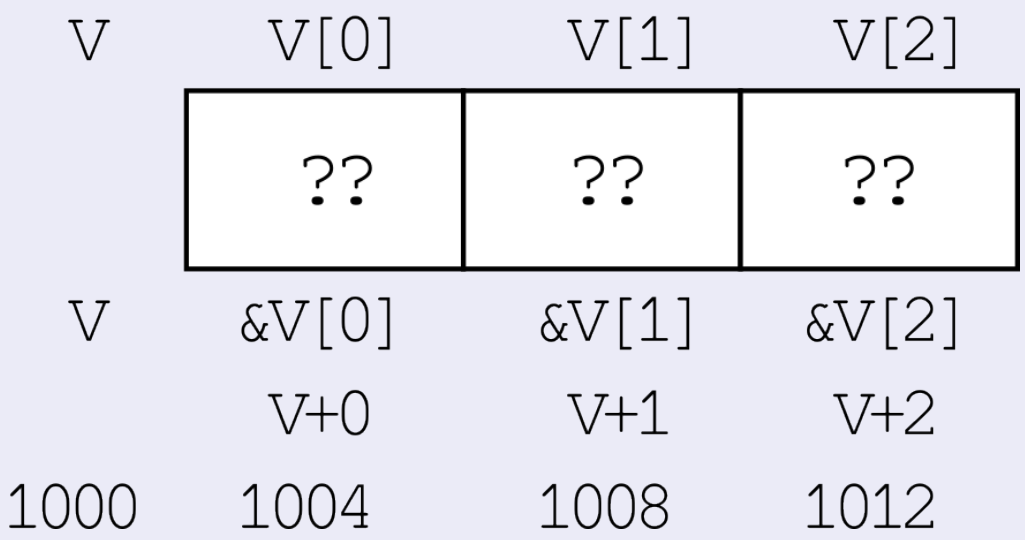
```
int V[3];
```



Valores na memória

Vetor x Endereços

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



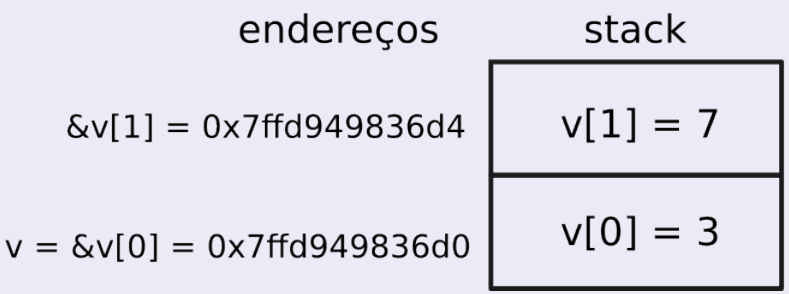
Valores na memória

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
```



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};
```

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

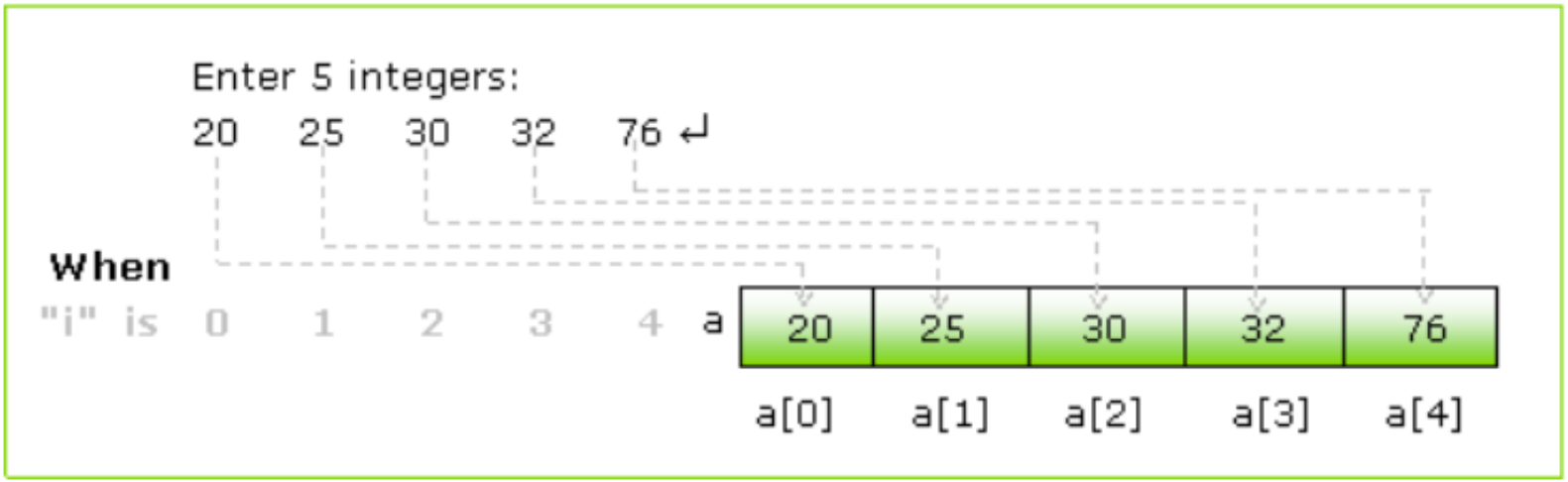
Inicialização na declaração - Erros

```
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

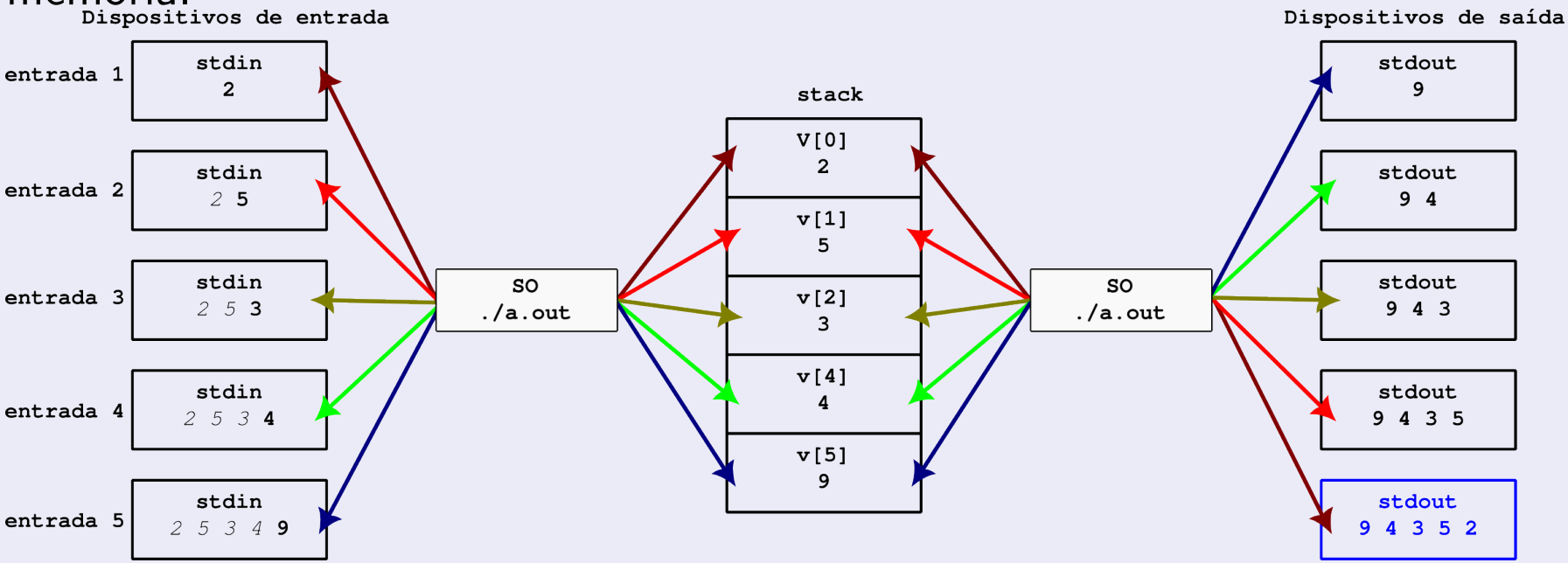
Igualar vetores

```
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

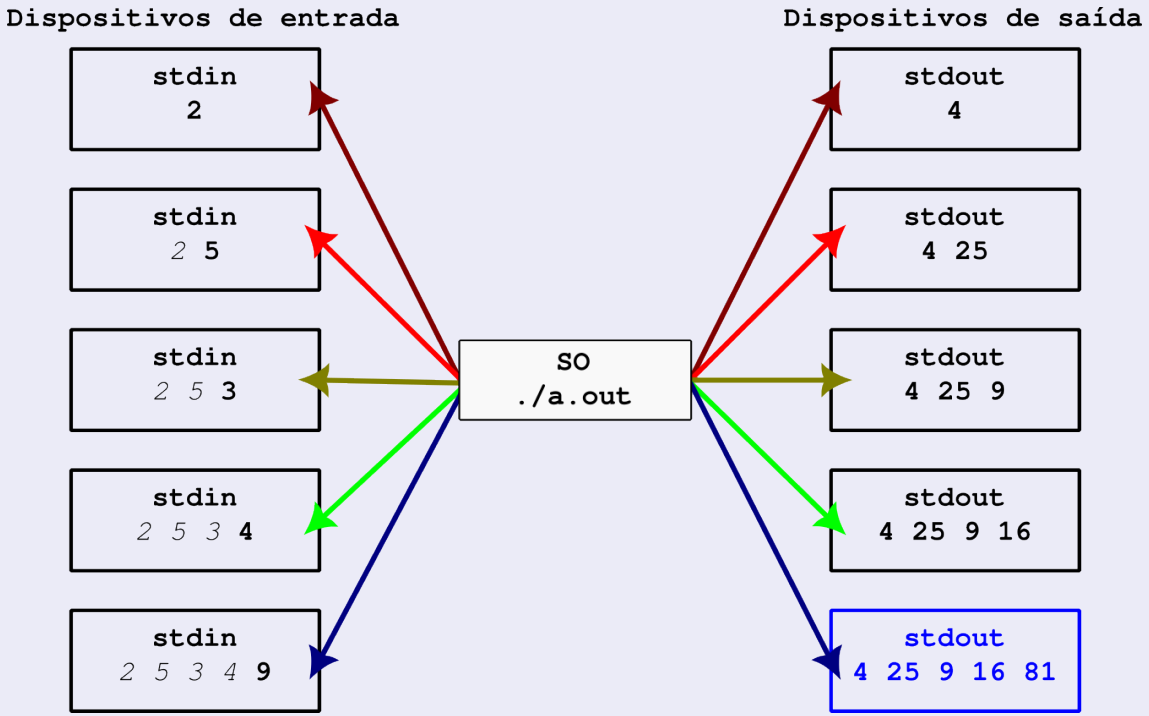
- Ler uma certa quantidade de valores inteiros e os imprimir na ordem inversa da leitura;
- Isto é, se os dados de entrada forem: 2, 5, 3, 4, 9, queremos imprimir na saída: 9, 4, 3, 5, 2;
- Este tipo de problema é impossível de ser resolvido com o uso de apenas uma variável pois, quando se lê o segundo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



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.

```c
#include <stdio.h>
int main()
{
    int i, n;
    scanf("%d", &n);
    int valor[n];
    for(i=0; i<n; i++)
    {
        scanf("%d", &valor[i]);
    }

    for(i=n-1; i>=0; i--)
    {
        printf("%d ", valor[i]);
    }

    printf("\n");
    return 0;
}
```

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

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int v[n], i, x;
    for(i=0; i<n; i++) {
        scanf("%d", &v[i]);
    }

    for(i=0; i<n; i++) {
        scanf("%d", &x);
        printf("%d ", v[i]*x);
    }

    printf("\n");
    return 0;
}
```

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)

```c
char nome1[6]="maria"; //string
char nome2[6]={'m', 'a', 'r', 'i', 'a', '\0'}; //string
char nome3[5]={'m', 'a', 'r', 'i', 'a'}; //vetor de caracteres

//especificador de formato: %s
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 encontar ' ', \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 = endereco da 1a posicao
5                      //lendo 1 sequencia de caracteres (palavra)
6                      //ate encontar ' ', \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);//printf("%.10s e %.20s\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 = endereco da 1a posicao
5                      //lendo 1 sequencia de caracteres (palavra)
6                      //ate encontar ' ', \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);//printf("%.10s e %.20s\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 = endereco da 1a posicao
5                      //lendo 1 sequencia de caracteres (palavra)
6                      //ate encontar ' ', \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);//printf("%.10s e %.20s\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 = endereco da 1a posicao
5                      //lendo 1 sequencia de caracteres (palavra)
6                      //ate encontar ' ', \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);//printf("%.10s e %.20s\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 = endereco da 1a posicao
5                      //lendo 1 sequencia de caracteres (palavra)
6                      //ate encontar ' ', \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);//printf("%.10s e %.20s\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 = endereco da 1a posicao
5                      //lendo 1 sequencia de caracteres (palavra)
6                      //ate encontar ' ', \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);//printf("%.10s e %.20s\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

```c
char nome1[100];

//leia tudo menos a quebra de linha
//leia tudo até a quebra de linha
scanf("%99[^\n]", nome1);

printf("%s\n", nome1);
```

Exemplo:

Entradas	**Saída:**
Jose da Silva	Jose da Silva

String (sequência de caracteres)

Lendo palavras e frases

```c
char nome1[100], nome2[100], nome3[100];

scanf("%99s", nome1); //lê no maximo 99 caracteres seguidos
scanf("%99s", nome2); //ou até encontar ' ', \t, \n

//lê o restante da entrada anterior inclusive \n
scanf("%99[^\n]", nome3); //leia tudo até a quebra de linha

printf("1 %s\n", nome1);
printf("2 %s\n", nome2);
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

```c
char nome1[100], nome2[100], nome3[100], nome4[100];

scanf("%99s", nome1); //le no maximo 99 caracteres seguidos
scanf("%99s", nome2);  //ou ate encontar ' ', \t, \n

//"consumir" o \n anterior: utilize um espaco
//      |
scanf(" %99[^\n]", nome3);
scanf(" %99[^\n]", nome4);

printf("1 %s\n", nome1);
printf("2 %s\n", nome2);
printf("3 %s\n", nome3);
printf("4 %s\n", nome4);
```

Exemplo:

Entradas	**Saída:**
Joao	1 Joao
Maria	2 Maria
Jose da Silva	3 Jose da Silva
Antonio Souza	4 Antonio Souza

String (sequência de caracteres)

Lendo várias strings seguidas

```c
char nome1[100], nome2[100], c;
int i;

for(i=0;i<4;i++) {
    scanf("%99s", nome1); //sequencia de palavras
    printf("%s\n", nome1);
}

//"pular" separadores(espacos, tab, \n) anteriores
for(i=0;i<4;i++) {
    //     |
    scanf(" %99[^\n]", nome2); //sequencia de frases
    printf("%s\n", nome2);
}

for(i=0;i<4;i++) {
    //     |
    scanf(" %c", &c);
    printf("%c\n", c);
}
```

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

```c
#include <stdio.h>

int main() {
    char palavra[16], frase[101];
    scanf("%15s", nome1); //sem & : vetor = endereco da 1a
    posicao
    //      |
    scanf(" %100[^\n]", frase);

    printf("%s\n", palavra);
    printf("%s\n", frase);

    return 0;
}
```

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

```c
#include <stdio.h>

int main() {
    char nome1[16];
    scanf("%15s", nome1); //sem & : vetor = endereco da 1a
    posicao

    int i = 0;
    while(s[i]!= '\0')
        i++;

    printf("%d\n", i);

    return 0;
}
```

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

```c
#include <stdio.h>

int main() {
    char s1[16], s2[16];
    scanf("%s %s", s1, s2);
    int i=0;
    while(s1[i]!='\0' && s2[i]!='\0' && s1[i]==s2[i])
        i++;

    int diferenca = s1[i] - s2[i];
    printf("%d\n", diferenca);
    return 0;
}

```

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

```c
#include <stdio.h>


int main() {
    char s[51], sCopia[51];
    scanf("%50[^\n]", s);
    int i;
    for (i = 0; s1[i] != 0; i++)
        sCopia[i] = s[i];

    sCopia[i] = '\0';

    //for (i = 0; (sCopia[i]=s[i]) != 0; i++);
    printf("%s\n", sCopia);

    return 0;
}
```

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

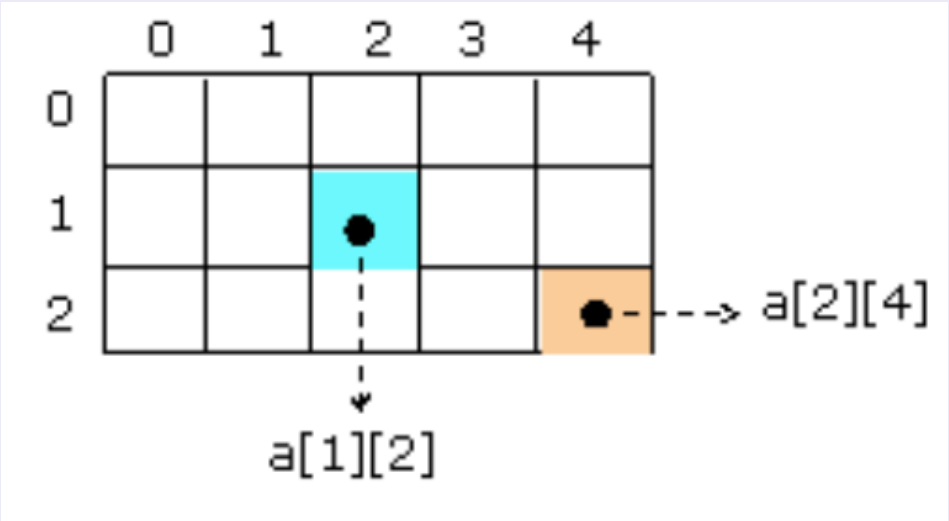
```
1     int n;
2     scanf("%d", &n);
3
4     char s[51];
5
6     int i, j;
7     for (i = 0; i<n; i++)
8     {
9         scanf(" %50[^\n]", s);
10        for (j=0; s[j]!'\=0'; j++);
11        printf("%d\n", j);
12    }
13
14
```

Roteiro

Matriz ou Array multidimensional

Definição e Declaração

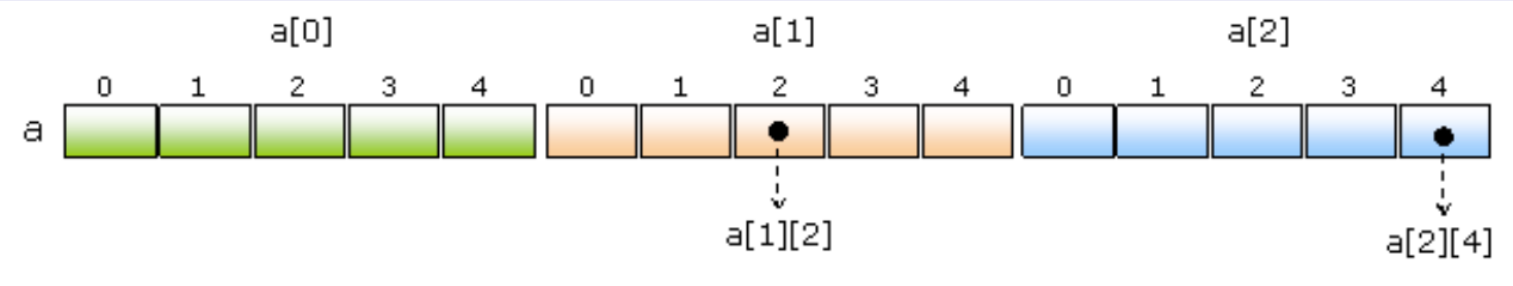
- Assim como os vetores, as matrizes são arrays;
- Os vetores são estruturas unidimensionais enquanto que as matrizes são bidimensionais;
- O acesso às posições de memória de um vetor são feitas com base em duas informações: o nome da variável e o deslocamento;
- Para se acessar os elementos de uma matriz,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

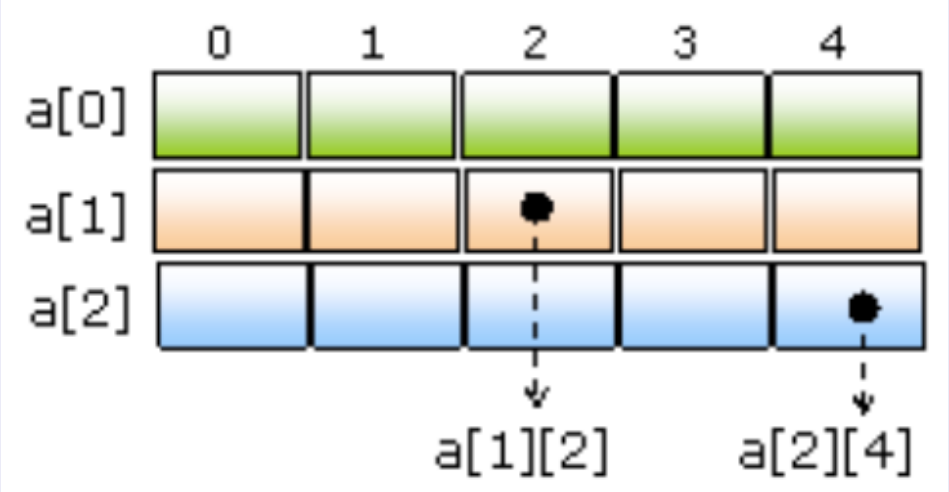


Matriz ou Array multidimensional

Declaração

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

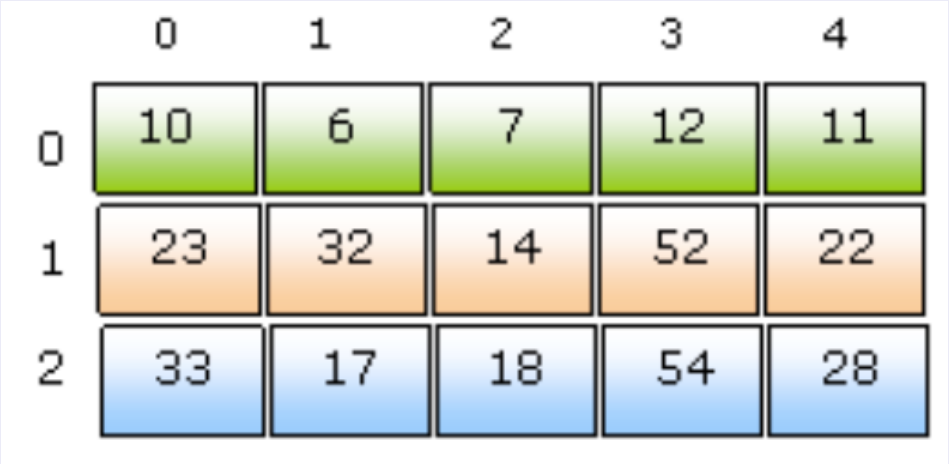




Matriz ou Array multidimensional

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
```



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

```c
int a[3][5];

int lin, col;
for(lin=0; lin<3; lin++)
    for(col=0; col<4; col++)
        scanf("%d", &a[lin][col]);

for(lin=0; lin<3; lin++)
    for(col=0; col<4; col++)
        scanf("%d", &b[lin][col]);

for(lin=0; lin<3; lin++)
{
    for(col=0; col<4; col++)
    {
        printf("%d ", a[lin][col] + b[lin][col]);
    }
    printf("\n");
}
```

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

```
int v[3][4] = { {12, 45, 67, 88},
                {32, 56, 44, 23},
                {78, 56, 33, 89} };
int col, lin;

for(col=0; col<4; col++)
{
    for(lin=0; lin<3; lin++)
    {
        printf("%d ", v[lin][col]);
    }
    printf("\n");
}
```

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

```
int v[][3] = {{-4, 4, 18}, {-2, 0, 4}, {-9, -2, 13}};
int im=0, jm=0;
int i, j;
for(i=0; i<3; i++)
{
    for(j=0; j<3; j++)
    {
        if(v[i][j] < v[im][jm])
            im=i;
            jm=j;
    }
}
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 mxn, 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

```
int v[4][4] = { {4, 6, 2, 1},
                {9, 0, 0, 2},
                {8, 7, 3, 9},
                {1, 2, 3, 4} };
int i, j;

for(i=0; i<4; i+=2)
{
    for(j=0; j<4; j+=2)
    {
        int s = v[i][j]   + v[i][j+1] +
                v[i+1][j] + v[i+1][j+1];
        printf("%.2f ", s/4.0);
        printf("\n");
    }
}
```

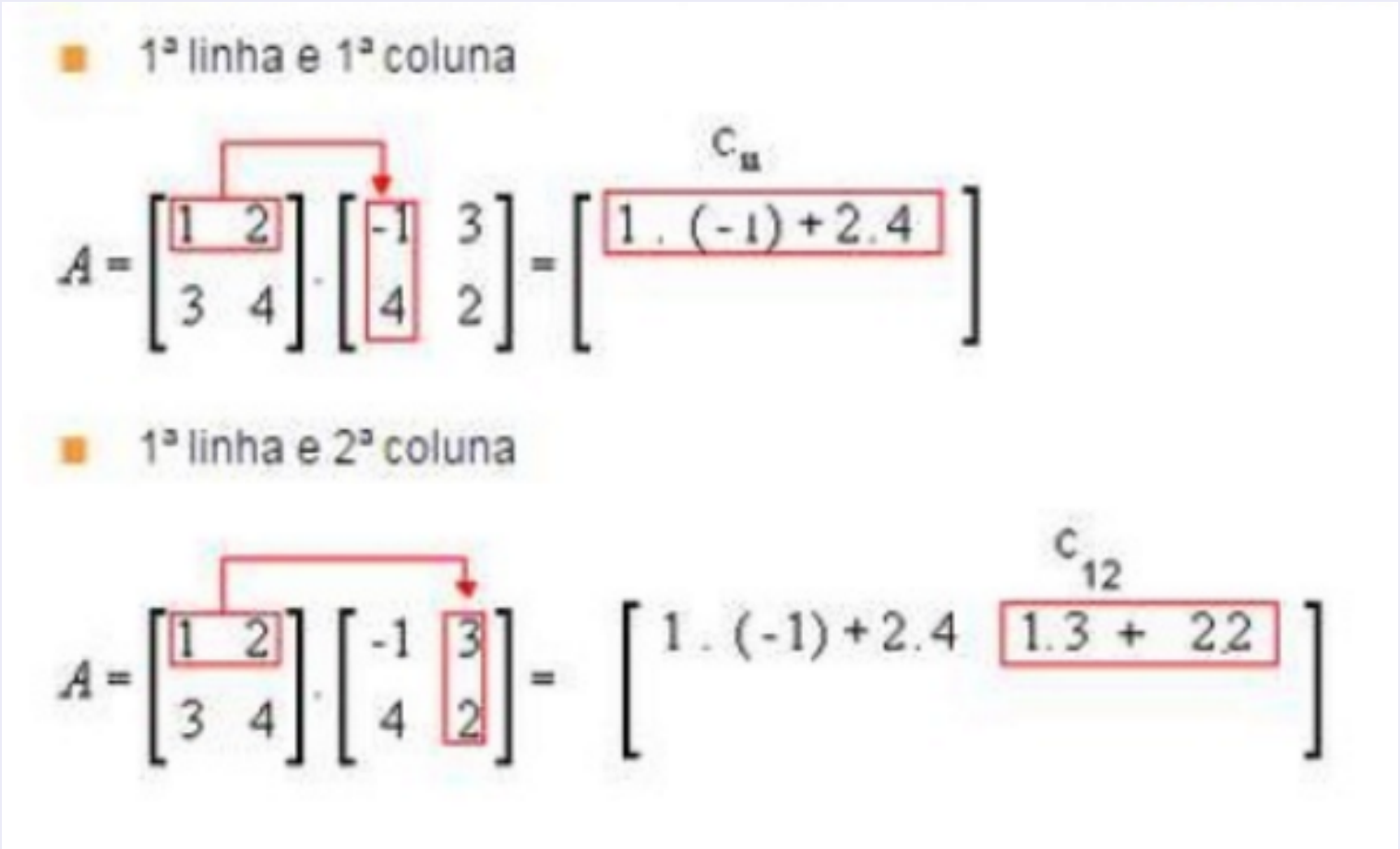
Matriz ou Array multidimensional

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

$$A.B = \begin{bmatrix} 2 & 3 \\ 0 & 1 \\ -1 & 4 \end{bmatrix} . \begin{bmatrix} 1 & 2 & 3 \\ -2 & 0 & 4 \end{bmatrix} = \begin{bmatrix} 2.1+3(-2) & 2.2+3.0 & 2.3+3.4 \\ 0.1+1(-2) & 0.2+1.0 & 0.3+1.4 \\ -1.1+4(-2) & -1.2+4.0 & -1.3+4.4 \end{bmatrix} = \begin{bmatrix} -4 & 4 & 18 \\ -2 & 0 & 4 \\ -9 & -2 & 13 \end{bmatrix}$$

Matriz ou Array multidimensional

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



Matriz ou Array multidimensional

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

- 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

```c
int A[3][2], B[2][3];
int lA, cB, i;
int soma_mult;

for(lA=0; lA<3; lA++)
    for(i=0; i<2; i++)
        scanf("%d", &A[lA][i]);

for(i=0; i<2; i++)
    for(cB=0; cB<3; cB++)
        scanf("%d", &B[i][cB]);

for(lA=0; lA<3; lA++) {
    for(cB=0; cB<3; cB++) {
        soma_mult = 0;
        for(i=0; i<2; i++)
            soma_mult = soma_mult + A[lA][i] * B[i][cB];

        printf("%d ", soma_mult);
    }
    printf("\n");
}
```

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

```c
int n;
scanf("%d", &n);

char s[n][51];

int i;
for (i = 0; i<n; i++)
{
    scanf(" %50[^\n]", s[i]);
}
for (i = n-1; i>=0; i--)
{
    printf("%s\n", s[i]);
}
```