Insper

Mutirão de Programação em C

Aula 2 – Vetores, Matrizes, Strings, Funções

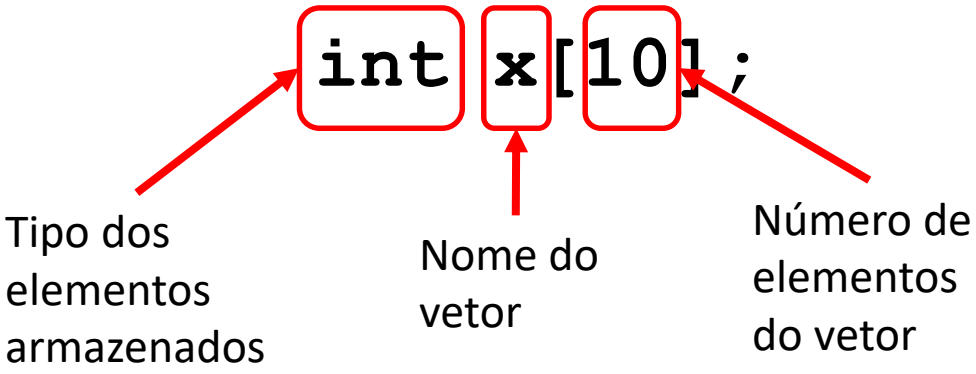
Objetivos

- Vetores e matrizes em C
- Strings
- Funções

Vetores em C

- Em inglês: *array*
  - A palavra *array* pode ser traduzida também como *série* ou *sequência*

- Exemplo de declaração de vetor:



Insper

Vetores em C: exemplos de declaração

```
unsigned long y[32];
char frase[80];
double pos[3];
```

```
float *data[200];
```

Vetor de ponteiros

```
struct color paleta[256];
```

Vetor de structs

```
struct color *paleta_ptr[256];
```

Vetor de ponteiros para structs

(Evite números *hard-coded*!)

- Sempre que você usar uma constante em seu código (como o número de elementos de um vetor) procure definir uma macro com o valor de sua constante:

```
#define MAX_SAMPLES 30
...
float sample[MAX_SAMPLES];
...
for (i = 0; i < MAX_SAMPLES; ++i) {
   ...
}
```

- Exercício: discuta com o seu colega do lado porque isso é uma boa idéia! Encontre pelo menos duas razões. (1 min)

Acessando elementos

```
#define N 10
...
int x[N], y;
...
x[4] = -300;
y = x[7];
```

Os índices devem ser valores inteiros entre 0 e N-1

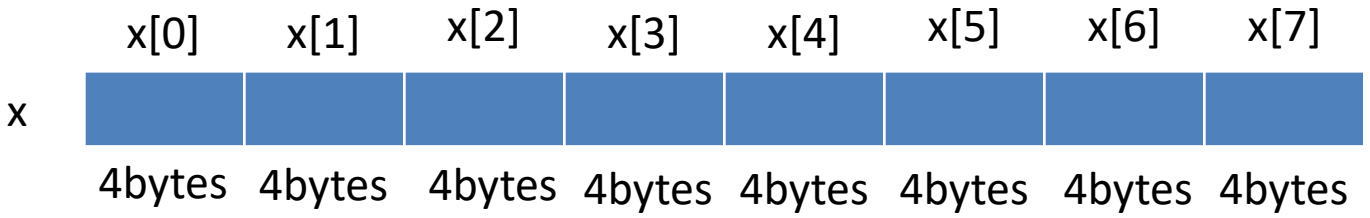
```
x[10] = 4;
x[3.0] = 12;
```

Erro!

(Como os dados se organizam em memória?)

- Os elementos de um vetor estão organizados sequencialmente em memória:

```
int x[8]
```



Insper

Exercícios

1. Declare um vetor apropriado para registrar as idades de todos os seus colegas de sala. Evite desperdício de espaço.
2. Declare um vetor de caracteres para armazenar uma palavra qualquer em português
3. Declare um vetor que servirá de buffer: armazenará até 2 segundos de áudio amostrado a 44100 Hz, 16 bits.

Exercícios

4. Faça um programa que leia um número N menor que 1000 e em seguida leia N números inteiros. Em seguida imprima estes N números em ordem reversa.

5. Faça um programa que leia números inteiros positivos até que um número negativo ou zero apareça. O programa deve então imprimir o quinto maior número. (Desafio)

(Dica: lendo elementos de *stdin* e escrevendo em *stdout)*

```c
#include <stdio.h>

#define N 30

int main(int argc, char *argv[]) {
  int a[N];
  int i, n;

  /* Ler número de elementos a serem lidos.
     Deve ser menor que N. */
  scanf("%d", &n);
  if (n > N) {
    return -1;
  }

  /* Ler n elementos. */
  for (i = 0; i < n; i++) {
    scanf("%d", &a[i]);
  }

  /* Imprime os elementos lidos. */
  for (i = 0; i < n; i++) {
    printf("%d\n", a[i]);
  }

  return 0;
}
```

in.txt

```
5
2 8 4 9 6
```

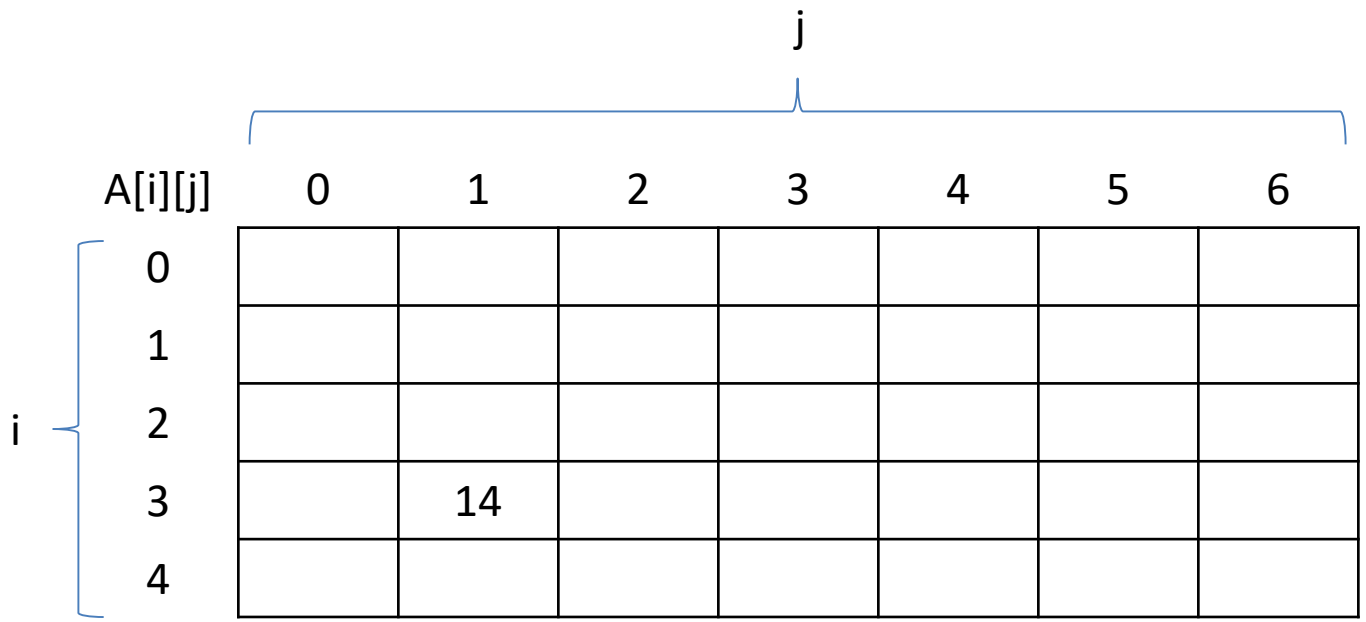
Linha de comando

```
$ gcc -o prog2 prog2.c
$ ./prog2 < in.txt > out.txt
```

```
$ cat out.txt
2
8
4
9
6
```

Matrizes

```
int A[5][7];
A[3][1] = 14;
```



Insper

(Como os dados se organizam em memória?)

- Os elementos da matriz estão armazenados linha-por-linha:

```
int A[2][3];
```

A[0][0]	A[0][1]	A[0][2]	A[1][0]	A[1][1]	A[1][2]

Exercícios

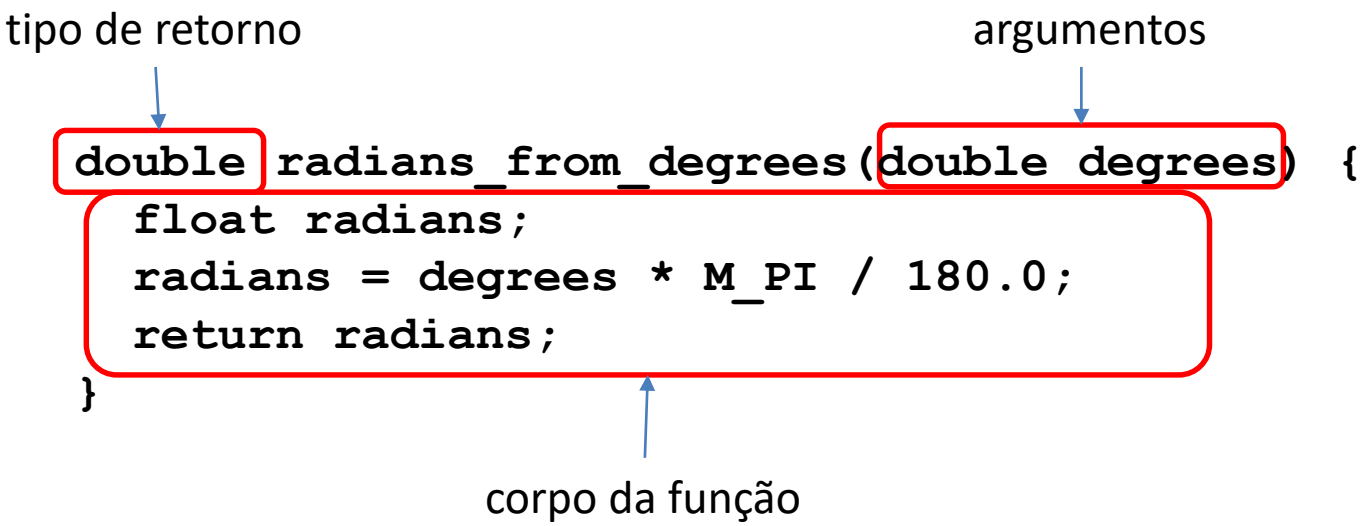
1. Declare uma matriz para armazenar transformações lineares 3D.

2. Declare uma matriz para armazenar um quadro de imagem de resolução 640 linhas x 480 colunas. Cada posição é do tipo **`struct pixel`**

3. Faça um programa que lê uma matriz 3x3 e imprime a sua transposta

Funções

- prog3.c



Insper

Declaração versus definição

- Declaração: apenas o nome, tipo de retorno, e tipos de argumentos são apresentados. Termina em ponto-e-virgula.

- Definição: A função completa

Declaração versus definição

- A definição de uma função que não foi declarada já conta como a declaração da função.
- Uma função só pode ser usada no código após ser declarada (mas não necessariamente definida)
- O código só é transformado em executável se todas as funções usadas tiverem sido definidas

Arquivos *header*

- São coletâneas de declarações de tipos e funções

```
#include <stdio.h>
#include “meu_próprio_arquivo_header.h”
```

Como funcionam as funções?

- Os argumentos são passados por valor
  - Cada argumento torna-se uma variável local dentro da função
  - Os valores passados na chamada da função são como valores iniciais destas variáveis locais

Exemplos de declarações de função

```
/* Recebe dois argumentos double e retorna um double. */
double pow(double base, double exponent);

/* Recebe uma string (um vetor de char) e não devolve nada. */
void log_error(char msg[]);

/* Não recebe nada, retorna um int. */
int read_int(void);

/* Não recebe nem retorna nada, apenas executa algo. */
void say_hi(void);

/* Recebe o número de argumentos de linha de comando, e um vetor
de ponteiros para char (trata-se de um vetor de strings, na
próxima aula entenderemos melhor isso). */
int main(int argc, char *argv[]);
```

Exercícios

1. Faça uma função que recebe dois inteiros e retorna o maior deles.

2. Usando a função do item 1, faça uma função que recebe um vetor de inteiros e retorna o maior deles.

3. Faça uma função que recebe um char c e um int n, e imprime n vezes o caractere na variável c

Exercícios

3. Faça uma função que calcula o maior divisor comum entre dois números do tipo unsigned long, e retorna um unsigned long com este resultado. O algoritmo deve ser como segue:

Algoritmo mdc(a, b):

    se b for zero retorne a

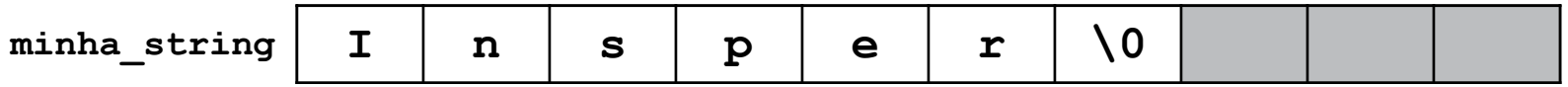
    caso contrario, retorne mdc(b, a%b)

Strings

- Uma string em C é um vetor de char onde um dos caracteres é o valor especial '\0'
- Este caractere marca o final da string.
- Exemplo:

```
char minha_string[10];
scanf("%s", minha_string);
printf("Voce digitou: %s\n", minha_string);
```

Note a ausência de '&' !



Insper

Exemplo

- Faça uma função que copia uma string em outro vetor de char, com os caracteres na ordem reversa

```c
#include <stdio.h>
#include <string.h>

#define NUM_CHARS 10

void reverte_string(char orig[], char dest[]) {
  int i, n;

  n = strlen(orig);

  for (i = 0; i < n; i++) {
    dest[(n - 1) - i] = orig[i];
  }

  dest[n] = '\0';
}

int main(int argc, char *argv[]) {
  char original[NUM_CHARS], reversa[NUM_CHARS];

  scanf("%s", original);
  reverte_string(original, reversa);
  printf("%s\n", reversa);

  return 0;
}
```

Número de caracteres sem contar o '\0'

orig

I	n	s	p	e	r	\0			

dest

r	e	p	s	n	I	\0			

Algumas funções úteis para manipulação de strings

```
/* Retorna o número de caracteres da string. */
size_t strlen(const char *str);

/* Copia a string em src para dest. Retorna dest. O vetor dest
   deve ser grande o suficiente para armazenar strlen(src) + 1
   caracteres. */
char *strcpy(char *dest, const char *src);

/* Compara alfabeticamente duas strings. Retorna negativo se
   str1 vem antes de str2, positivo se str1 vem depois de str2,
   e zero se str1 e str2 são iguais. */
int strcmp(const char *str1, const char *str2);

/* Concatena a string src no final de dest, retorna dest.
   O vetor dest deve ser grande o suficiente para armazenar
   strlen(src) + strlen(dest) + 1. */
char *strcat(char *dest, const char *src);
```

Inicialização de vetores e strings

```
int v[] = {0, 1, 1, 2, 3, 5, 8};

char help_msg[] = "Uso: meu_programa <arquivo.txt>";
```

Exercícios

1. Faça uma função

```
int is_palindrome(char str[]);
```

que recebe uma string e retorna 1 se str é palíndromo ou 0 caso contrário.

2. Faça um programa que imprime os argumentos de linha de comando

Exercícios

2. Faça uma função

```
int strfind(char str[], char token[]);
```

que recebe strings str e token, e procura a primeira ocorrência de token em str. A função retorna o índice em str da ocorrência de token. Se token não aparece em str, retorne -1. Por exemplo:

```
strfind("Engenharia Insper", "Insper") -> retorna 11
strfind("Churrasquinho grego", "bactéria") -> retorna -1 !!!
```

Insper

www.insper.edu.br