PONTEIROS

Professor: Francisco Dantas Nobre Neto
E-mail: dantas.nobre@ifpb.edu.br

Agenda

- Ponteiros
  - Ponteiros e Vetores;
  - Ponteiros e String;
  - Ponteiros e Matrizes.

Ponteiros e Vetores

- O que vai ser impresso no código abaixo? Como interpretar o resultado?

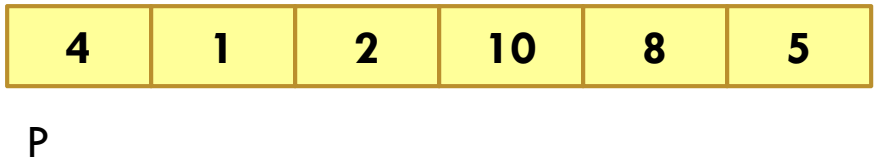
O que vai ser impresso???

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  int a[5] = {1, 8, 3, 4, 5};
  char b[4] = "abd";

  printf("Posicao %x --- %x\n",   a,   (a+1));
  printf("Posicao %x --- %x\n",   b,   (b+1));

  return 0;
}
```

Ponteiros e Vetores

- Os vetores são elementos contíguos em memória:
  - Estão em sequência lineares, um após o outro.
- Na linguagem C, quando declaramos um vetor e queremos passá-lo como parâmetro, o compilador passa o endereço inicial do vetor:
  - Os arrays são passados por referência em C;
  - O endereço do primeiro elemento é que é passado.

```
int p[6] = {4, 1, 2, 10, 8, 5};
```



Ponteiros e Vetores

- Para que seja possível passarmos um vetor de inteiros para uma função C, devemos ter um parâmetro *int* * para apontar ao endereço de inteiros;
- Exemplo:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  int a[5] = {1, 8, 3, 4, 5};
  inc_array(a, 5);
  return 0;
}
```

```
void inc_array(int *vetor, int n) {
  int i;
  for(i = 0; i < n; i++)
    vetor[i]++;
}
```

Ponteiros e Strings

- Strings são cadeias de caracteres;
- Na memória, cada caractere está em um espaço específico;
- Em C, as Strings possuem caracteres de terminação, representado pelo valor '\0';
- Para declaração de uma string, deve-se usar a forma geral abaixo:
  - *char nome_string[tamanho].*

Ponteiros e Strings

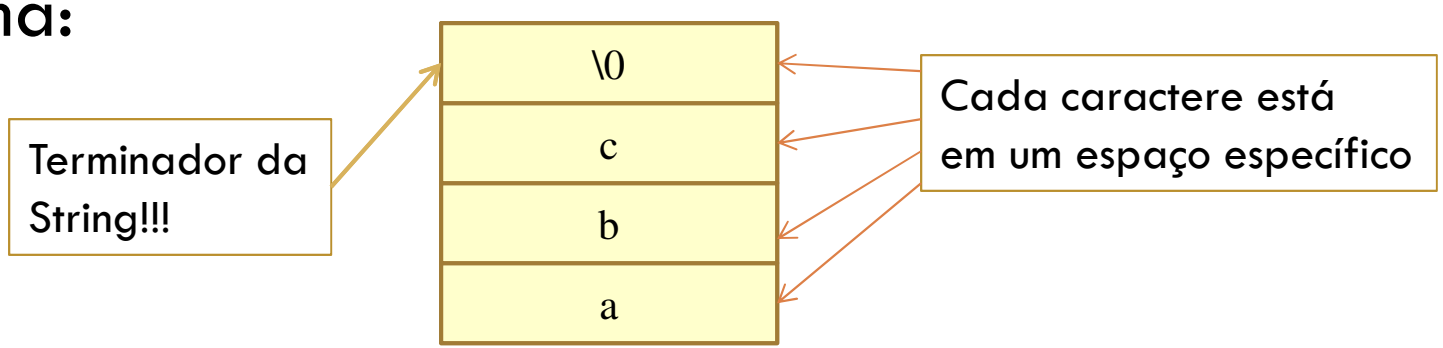
- Equivalência do código abaixo.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  char a[4] = {'a', 'b', 'c', '\0'};

  return 0;
}
```

⟺

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  char a[4] = "abc";

  return 0;
}
```

- A string "abc" estará na memória da seguinte forma:



Ponteiros e Strings

- Para percorrer uma String, podemos utilizar o seguinte código:

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  exibe_string("Teste");
}
```

```
void exibe_string(char *string) {
  while(*string)
      putchar(*string++);
}
```

- Códigos a serem evitados com string:
  - Não use o comando abaixo para copiar uma string na outra.

```
int main(int argc, char *argv[]) {
  char a[5]  = "abc", b[5];
  b = a;  // b vai apontar para a posição inicial do vetor a
  b[0] = 'm'; // essa alteração vai ser refletida no vetor a
  printf("%s", a);
}
```

Ponteiros e Strings

- Códigos a serem evitados com string:
  - Deixar ao menos um caractere reservado para o '\0'.

```
int main(int argc, char *argv[]) {
  char a[4]  = "abc";
  printf("%s", a); // O que vai ser impresso?
}
```

\0
c
b
a

```
int main(int argc, char *argv[]) {
  char a[4]  = "abcd";
  printf("%s", a); // O que vai ser impresso?
}
```

\0
^
e
d
c
b
a

Ponteiros e Strings

- Em C, existe uma biblioteca de funções para facilitar a manipulação de strings:
  - #include <string.h>.
- As principais funções de *string.h*:
  - *strcpy(string_destino, string_origem)*:
    - Copia o conteúdo da string de origem na string de destino.

```
int main(int argc, char *argv[]) {
  char a[4]  = “abc”;
  char b[4];
  strcpy(b, a); // Copia o conteúdo de a para b
}
```

Ponteiros e Strings

- As principais funções de *string.h*:
  - *strcat(string_destino, string_origem)*:
    - Concatena a string de origem com a string de destino.

```
int main(int argc, char *argv[]) {
  char a[10]  = "abc";
  char b[3] = "de";
  strcat(a, b); // Concatena o conteúdo de b com o de a
}
```

  - *int strlen(string)*:
    - Retorna o tamanho da string.

```
int main(int argc, char *argv[]) {
  char a[10]  = "abc";
  strlen(a); // Retorna o valor 3, tamanho da string armazenada em a
}
```

Ponteiros e Strings

- As principais funções de *string.h*:
  - *int strcmp(string1, string2)*:
    - Compara o conteúdo das duas strings:
      - Se forem iguais, o valor 0 (zero) é retornado;
      - Se a string1 for maior que a string2, retorna-se o valor 1;
      - Se a string2 for maior que a string1, retorna-se o valor -1;
      - Existe diferença entre letras minúsculas e maiúsculas.

```
int main(int argc, char *argv[]) {
  strcmp("ab", "ab");    // É retornado o valor 0
  strcmp("ab", "abd");  // É retornado o valor -1
  strcmp("abd", "ab");  // É retornado o valor 1
  strcmp("ab", "AB");   // Faça você mesmo
 strcmp("AB", "ab");   // Faça você mesmo
}
```

Ponteiros e Strings

- 1) Faça uma função que receba duas *strings*: a primeira uma palavra qualquer; a segunda uma *string* em branco do tamanho da primeira. A função deverá fazer a inversão da primeira *string*, e salvar na segunda *string*. Por exemplo, a *string* LINGUAGEM_C será a primeira; enquanto C_MEGAUGNIL é a inversão dela, após chamada à função.

Ponteiros e Strings

- 2) Faça uma função que receba dois parâmetros: uma *string*; e um vetor de inteiros. A função deverá armazenar, na primeira posição do vetor de inteiros, o quantitativo de letras maiúsculas. Na segunda posição, o quantitativo de letras minúsculas e, na terceira, o quantitativo de espaços em branco.

Ponteiros e Matrizes

- Na linguagem C, é possível criarmos vetores bidimensionais, que são as matrizes;
- Podemos iniciar uma matriz de duas formas:

```
int main(int argc, char *argv[]) {
  int x[2][3] = {{1, 2, 3}, {5, 8, 9}};
}
```

OU

```
int main(int argc, char *argv[]) {
  int x[2][3] = {1, 2, 3, 5, 8, 9};
}
```

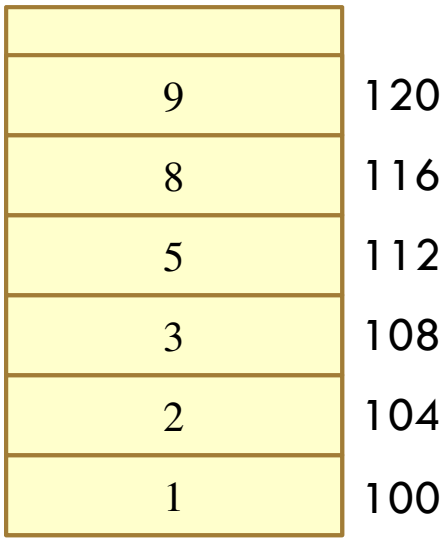
Podemos omitir a linha

```
int main(int argc, char *argv[]) {
  int x[ ][3] = {1, 2, 3, 5, 8, 9};
}
```

Ponteiros e Matrizes

- Em memória, teremos os números dispostos da seguinte forma:

```
int main(int argc, char *argv[]) {
  int x[ ][3] = {1, 2, 3, 5, 8, 9};
}
```



Ponteiros e Matrizes

- Em C, a indexação dos elementos começa em 0 (zero):
  - O elemento da primeira linha e da primeira coluna é acessado por *x[0][0]*;
  - A variável **x** representa um ponteiro para o primeiro "vetor-linha":
    - x[0] – primeiro "vetor-linha";
    - x[1] – segundo "vetor-linha";
    - x[2] – terceiro "vetor-linha".

Ponteiros e Matrizes

- Para passarmos uma matriz para uma função, o parâmetro da função deve receber um ponteiro para vetores;
- Sintaticamente, o protótipo da função pode ser de duas formas:

```
void func(…, int (*matriz)[3], …){
   matriz[0][0]= 5;
}
```

OU

```
void func(…, int matriz[ ][3], …){
   matriz[0][0]= 5;
}
```

Dentro da função, o acesso aos elementos da matriz é feita com indexação dupla!!!