VETORES E STRINGS

Alocação estática de memória

- Ao se declarar uma variável qualquer, o compilador deixa reservado um espaço suficiente na memória para armazená-la

```
int a; // 4 bytes
float x; // 4 bytes
double y; // 8 bytes
char c; // 1 byte

char *c;  // 4 bytes
```

Alocação estática de memória

- Ao fazer a alocação estática, apenas o espaço necessário na memória é reservado
- O conteúdo de cada posição não é alterado, e portanto uma variável apenas declarada pode conter qualquer coisa
- Inicializar as variáveis, atribuindo valores, antes do uso
  - Inclusive vetores, matrizes e strings

Vetores

- Quando se declara um vetor, o valor entre colchetes indica quantas vezes o espaço de memória necessário para o tipo básico será alocado

```
char v[100]; //  100 * sizeof(char) = 100 bytes
int v[100]; //  100 * sizeof(int) = 400 bytes
float vf[200]; // 200 * sizeof(float) = 800 bytes
double z[1000]; // 1000 * sizeof(double) = 8000 bytes
```

Pode ser utilizado para descobrir o tamanho de um vetor?

Vetores

- A referência a uma posição de um vetor indica o cálculo de uma posição de memória a partir do início do vetor

```
float x[1000];
// x[20] está na posição x + 20*sizeof(float)
```

- Na verdade, o símbolo x é um apontador para o início da região de memória reservada para o vetor

Vetores

- C **NÃO AVISA NEM PRODUZ ERRO** QUANDO O LIMITE DE UM VETOR OU MATRIZ FOR EXCEDIDO

```
float x[1000];
y = x[2000]; // não dá erro, mas vai acessar
                    // uma parte inesperada da
memória
```

  -Erro mais comum (runtime):

```
segmentation fault
```

- É responsabilidade do programador verificar os limites, e garantir que não sejam excedidos

Matrizes

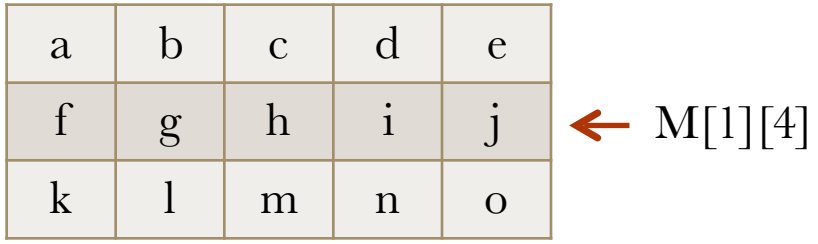
= Vetores de mais de uma dimensão

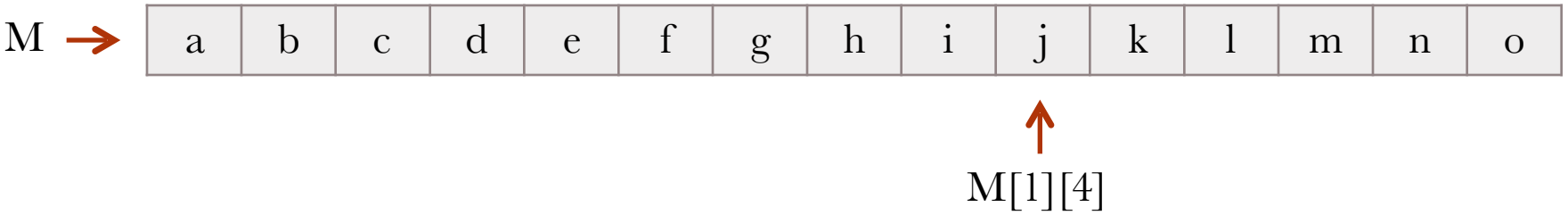
- Na verdade, a alocação na memória é linear
- Muda apenas o cálculo da posição de memória do elemento referenciado

```
int M[5][5];
...
M[0][0] = 15;
M[2][3] = 2; // posicao: M + (2*5 + 3)*sizeof(int)
```

Matrizes

```
char M[3][5];
v = M[1][4];
```





```
char *M;
M = (char *) malloc(3 * 5 * sizeof(char));

// acesso a linha i e coluna j em matriz l x c
// v = M[i * c + j]
v = M[1 * 5 + 4];
```

Strings

- Um string é um vetor do tipo char
- Para manipulação do string, atribuindo e recuperando valores, e realizando operações sobre ele (funções no **string.h**), é importante entender essa característica
- Quando o conteúdo de um string não ocupa todo o espaço disponível, usa-se um caractere ‘\0’ (ou NULL, código ASCII 0) como delimitador

Strings

- Strings constantes aparecem no código entre aspas

```
printf("%s", "Bom dia!");
```

- O delimitador \0 está incluído:

- Tamanho da string:

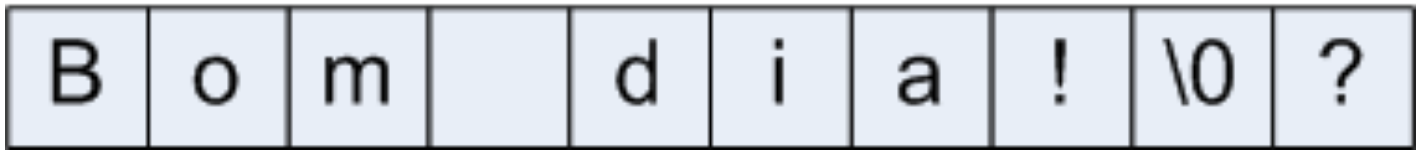
```
size_t strlen(char *str);
```

Strings

- Não é possível fazer atribuições diretas para strings
- Usar a função strcpy ou a função strncpy

```
char *strcpy(char *DST, char *SRC);
```

```
char s[10];
strcpy(s, "Bom dia!");
```



Código!!

Strings

- Tamanho da string (não conta o \0):

```
size_t strlen(char *str);
```

```
char s[10];
int tamanho;

   strcpy(s, "Bom dia!\n");
   tamanho = strlen(s);  // 9

   tamanho = sizeof(s);  // 10 (Não usar!)
```

Código!!

Strings

- Na inicialização, pode-se usar o mesmo recurso disponível para vetores
- `char nome[] = {'A', 'n', 'a', '\0'};`

Ou

- `char nome[] = "Ana";`

Strings

- Como o nome do string representa o endereço onde começa o string, é possível manipulá-lo diretamente
  - Cuidado!
- Exemplo

```
char nome[] = "Alexandre";
printf("%s\n", nome + 3); // imprime "xandre"
```

Strings

- Funções
  - strcat(s1, s2): concatena o s2 no s1 (s1 tem que ter espaço)
  - strcmp(s1, s2): retorna
    - < 0 se s1 é menor que s2,
    - 0 se s1 é igual a s2
    - > 0 se s1 é maior que s2 (ordem alfabética)
    - A comparação entre strings também tem que ser feita caractere a caractere,
    - Não se pode usar s1==s2; isso só compara os endereços

Strings

- Comparação de strings

```
char str1[] = "abcd ";
char str2[] = "abcd ";
int cmp;

   cmp = strcmp(str1, str2);    // retorna 0

   str2[4] = 'e';
   str1[4] = 'a';
   cmp = strcmp(str1, str2);    // retorna -1

   str2[4] = 0;
   cmp = strcmp(str1, str2);    // retorna 1
```

Strings

- Impressão para uma string (sprintf)

```
char str1[10];

   sprintf(str1, “%d“, 33);
```

Strings

- strncat, strncmp, strncpy: idem aos anteriores, mas especifica o número de caracteres que serão considerados

Strings

- Vetores de strings podem ser criados
- Exemplo

```
char DiaSemana[7][14] = {“Domingo”,
  “Segunda”, “Terça”, “Quarta”,
  “Quinta”, “Sexta”, “Sabado”};
...
printf(“%s\n”, DiaSemana[3]);
```