Linguagem C

Vetores e matrizes como parâmetros de funções

Professora: Emanoeli Madalosso
emanoelim@utfpr.edu.br

Vetores como parâmetros de funções

- Existem duas maneiras de passar um vetor para uma função. Considere uma função que precisa receber um vetor e então imprimi-lo na tela:
  - 1ª forma: void imprime_vetor(int vetor[], int n);
  - 2ª forma: void imprime_vetor(int *vetor, int n);

Vetores como parâmetros de funções

- Exemplo 1:

```
void imprime_vetor(int vetor[], int n) {
    int i;
    for(i = 0; i < n; i++)
        printf("%d\t", vetor[i]);
}
```

Vetores como parâmetros de funções

- Exemplo 2:

```
void imprime_vetor(int *vetor, int n) {
    int i;
    for(i = 0; i < n; i++)
        printf("%d\t", vetor[i]);
}
```

Vetores como parâmetros de funções

- Nas duas formas é necessário passar o tamanho do vetor, pois este não pode ser recuperado dentro da função;
- O tamanho só não é necessário quando for uma string, pois ela possui o ‘\0’ que pode ser usado para encontrar o final da string.

Vetores como parâmetros de funções

- Quando passamos uma variável do tipo inteiro para uma função, por ex., o que ela recebe é uma cópia do valor dessa variável;
- No caso de vetores, ao passar um vetor para uma função, seja usando “vetor[]” ou “*vetor”, a função não recebe uma cópia do vetor inteiro. O que a função recebe é uma cópia do primeiro endereço de memória ocupado pelo vetor;
- Desta maneira, diferente de funções que recebem tipos simples de dados, **qualquer alteração feita nos itens do vetor irá alterar seu conteúdo diretamente no endereço de memória**;
- **Isso significa que quando algum item do vetor for alterado na função, essa alteração irá refletir na main.**

Vetores como parâmetros de funções

- Considere as funções:

```
void imprime_vetor(int *vetor, int n) {
    int i;
    for(i = 0; i < n; i++)
        printf("%d\t", vetor[i]);
}
void altera_item_vetor(int vetor[], int n) {
    vetor[0] = 999;
}
```

Vetores como parâmetros de funções

- Ao chamar na main a função altera_item_vetor e em seguida a função imrpime_vetor:

```
int main(void) {
    int n = 10;
    int vetor[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    altera_item_vetor(vetor, n);
    imprime_vetor(vetor, n);
    return 0;
}
```

Vetores como parâmetros de funções

- Os valores apresentados por imprime_vetor serão:

```
999     2       3       4       5       6       7       8       9       10
Process returned 0 (0x0)   execution time : 0.008 s
Press ENTER to continue.
```

Strings como parâmetros de funções

- Aplicam-se as mesmas considerações dos vetores, exceto que não é necessário passar o tamanho da string.

Strings como parâmetros de funções

- O que o printf irá mostrar?

```
int main(void) {
     char s[] = "hoje está chovendo";
     substitui_espacos(s);
     printf("%s", s);
     return 0;
}
```

```
void substitui_espacos(char s[]) {
     int i = 0;
     while(s[i]) {
     if(s[i] == ' ')
          s[i] = '*';
     i++;
     }
}
```

Matrizes como parâmetros de funções

- Para passar uma matriz como parâmetro de uma função, utilizamos dois pares de colchetes após o identificador do parâmetro;
- Seguindo a mesma ideia da declaração de uma matriz, apenas a dimensão mais à esquerda pode ser omitida, desta forma os dois exemplos a seguir são válidos:

Matrizes como parâmetros de funções

```
void print_matriz(int m[2][2]){
     int i, j;
     for(i = 0; i < 2; i++) {
          for(j = 0; j < 2; j++) {
               printf("%d\t", m[i][j]);
          }
          printf("\n");
     }
}
```

```
void print_matriz(int m[][2]){
     int i, j;
     for(i = 0; i < 2; i++) {
          for(j = 0; j < 2; j++) {
               printf("%d\t", m[i][j]);
          }
          printf("\n");
     }
}
```

Matrizes como parâmetros de funções

- No caso de o número de linhas e número de colunas não ser fixo, podemos criar duas variáveis globais, *l* e *c*, por exemplo:

Matrizes como parâmetros de funções

```
#include <stdio.h>

int l, c;

void print_matriz(int m[l][c]) {
     int i, j;
     for(i = 0; i < l; i++) {
          for(j = 0; j < c; j++) {
               printf("%d\t", m[i][j]);
          }
     printf("\n");
     }
}
```

```
int main(void) {
     int i, j;
     printf("l, c: ");
     scanf("%d, %d", &l, &c);
     int m[l][c];
     for(i = 0; i < l; i++) {
          for(j = 0; j < c; j++) {
               printf("%d, %d: ", i, j);
               scanf("%d", &m[i][j]);
          }
     }
     print_matriz(m);
     return 0;
}
```

Matrizes como parâmetros de funções

- A seguinte forma também é válida:

Matrizes como parâmetros de funções

```
#include <stdio.h>

int l, c;

void print_matriz(int m[][c]) {
     int i, j;
     for(i = 0; i < l; i++) {
          for(j = 0; j < c; j++) {
               printf("%d\t", m[i][j]);
          }
     printf("\n");
     }
}
```

```
int main(void) {
     int i, j;
     printf("l, c: ");
     scanf("%d, %d", &l, &c);
     int m[l][c];
     for(i = 0; i < l; i++) {
          for(j = 0; j < c; j++) {
               printf("%d, %d: ", i, j);
               scanf("%d", &m[i][j]);
          }
     }
     print_matriz(m);
     return 0;
}
```

Vetores/Matrizes como retorno de funções

- Quando passamos um vetor ou matriz para função, a função recebe uma cópia do primeiro endereço ocupado pela estrutura;
- Qualquer alteração em um item da estrutura é feita diretamente no endereço de memória ocupado pelo item;
- Ou seja, essa alteração não é feita sobre uma cópia, mas sim sobre o conteúdo original, refletindo na main;
- Sendo assim não existe a necessidade de retornar o vetor ou a matriz;

Vetores/Matrizes como retorno de funções

- Só fará sentido ter uma função que retorna um vetor ou matriz se for criado um novo vetor ou uma nova matriz dentro da função;
- Isso só poderá ser feito alocando memória dinamicamente (conteúdo de Algoritmos 1).