Vetores, Funções e Strings

Prof. Denio Duarte
Prof. Geomar Schreiner

Sumário

- Vetores
- Strings
- Funções

Sumário

- **Vetores**
- Strings
- Funções

Vetores

- Um vetor é uma área de alocação contígua de memória
  - Ou seja, quando declaramos um vetor o programa vai alocar uma área contígua de memória baseada no tamanho do vetor
- Em C o vetor possui apenas um tipo de dado, todos os elementos ou posições do vetor armazenam o mesmo tipo de dado

- Como declarar um vetor:

```
3   int main(){
4
5     int vetor[10];
6     float vet[5000];
7
8     return 0;
9   }
```

Vetores

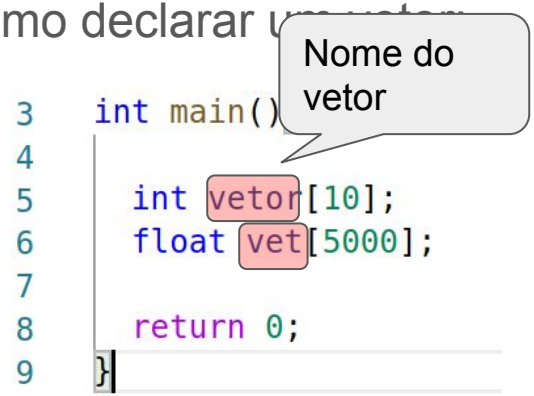
- Um vetor é uma área de alocação contígua de memória
  - Ou seja, quando declaramos um vetor o programa vai alocar uma área contígua de memória baseada no tamanho do vetor
- Em C o vetor possui apenas um tipo de dado, todos os elementos ou posições do vetor armazenam o mesmo tipo de dado

- Como declarar um vetor:

Vetores

- Um vetor é uma área de alocação contígua de memória
  - Ou seja, quando declaramos um vetor o programa vai alocar uma área contígua de memória baseada no tamanho do vetor
- Em C o vetor possui apenas um tipo de dado, todos os elementos ou posições do vetor armazenam o mesmo tipo de dado

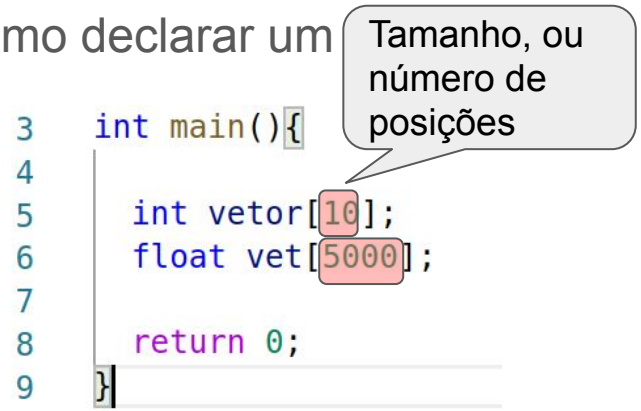
- Como declarar um vetor:



Vetores

- Um vetor é uma área de alocação contígua de memória
  - Ou seja, quando declaramos um vetor o programa vai alocar uma área contígua de memória baseada no tamanho do vetor
- Em C o vetor possui apenas um tipo de dado, todos os elementos ou posições do vetor armazenam o mesmo tipo de dado

- Como declarar um



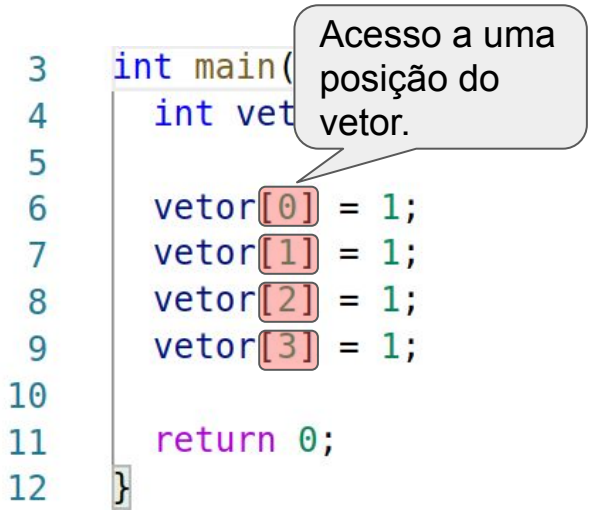
Vetores

- Em C a contagem do índice inicia de 0
  - O primeiro elemento do vetor é sempre o elemento na posição 0
- Acesso é feito da mesma forma que em Python

```
3   int main(){
4     int vetor[4];
5
6     vetor[0] = 1;
7     vetor[1] = 1;
8     vetor[2] = 1;
9     vetor[3] = 1;
10
11    return 0;
12  }
```

Vetores

- Em C a contagem do índice inicia de 0
  - O primeiro elemento do vetor é sempre o elemento na posição 0
- Acesso é feito da mesma forma que em Python



Vetores

- O C não valida se a posição acessada de um vetor é válida, porém não há garantias nos dados fora do intervalo declarado
  - Você pode acessar a posição 11 de um vetor de tamanho 10, mas nesta área haverá lixo.
- Um vetor pode ser inicializado das seguintes formas

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

Sumário

- Vetores
- **Strings**
- Funções

Strings

- C não possui um tipo string explícito.
  - Ous seja, não existe a palavra reservada “string” ou “String”
- Para declarar uma string em C, é feita uma cadeia de caracteres ou vetor de caracteres
  - char string[20];

Strings

- Apesar de ser declarado como um vetor de caracteres para imprimir uma string basta utilizar %s

```c
#include <stdio.h>

int main(){
    char string[20];
    scanf("%s", string);

    printf ("Você digitou: %s\n", string);
    return (0);
}
```

Strings

- Apesar de ser declarado como um vetor de caracteres para imprimir uma string basta utilizar %s

```c
#include <stdio.h>

int main(){
    char string[20];
    scanf("%s", string);

    printf ("Você digitou: %s\n", string);
    return (0);
}
```

Estou limitando o tamanho a 20

Strings

- Apesar de ser declarado como um vetor de caracteres para imprimir uma string basta utilizar %s

```c
#include <stdio.h>

int main(){
    char string[20];
    scanf("%s", string);

    printf ("Você digitou: %s\n", string);
    return (0);
}
```

Apenas vai ler até o primeiro “ ” (espaço)

Strings

- Para manipular strings em C utilizamos a biblioteca “string.h”
  - Ela apresenta uma série de funções úteis que permitem manipular as strings
  - Para utliizá-la basta adicionar
    - #include <string.h>
  - Funções úteis
    - **strcpy(destino, origem)**: Copia uma string **origem** para uma string **destino**
    - **strcmp** : Compara duas strings, retorna 0 se forem iguais, -1 se a primeira vier antes da segunda e 1 se a segunda vier antes da primeira
    - strlen(st): retorna o tamanho da string st.

Vetores

1. Considerando o vetor = {1,2,3,3,4,6,6,7,5,7,1,3}
    a. Faça um código que imprima o vetor na tela
2. Crie um programa que lê 5 valores inteiros e, em seguida, mostre na tela os valores lidos.
3. Baseado no vetor lido na questão anterior, crie um código que imprima o maior e o menor valor contidos no vetor.
4. Crie um programa que inverte o valor de uma string
5. Crie um programa que calcula o tamanho de uma string (sem usar strlen)

Sumário

- Vetores
- Strings
- **Funções**

Funções

- Uma função é um conjunto de comandos (bloco de comandos) associado a um nome
  - O uso deste nome é uma chamada da função
- Chama de função
  - Quando o programa realiza a chamada de uma função, esses dados são empilhados na memória, o bloco é executado, após seu término o programa continua a execução da próxima instrução
    - Quando o bloco de execução de uma função termina é chamado de retorno da função.

Funções

- A chamada de uma função, geralmente, passa informações (argumentos) para o processamento da função
  - Lista pode ser vazia
  - Lista aparece entre ( ) junto ao nome da função

função Python

```
1   def addition( a , b ):
2       return a + b
```

função C

```
1   int addition (int a, int b){
2       return a + b;
3   }
```

Funções

- Uma função pode retornar resultados ao programa que a chamou
  - O tipo de retorno é definido na definição da função

função Python

```python
def addition( a , b ):
    return a + b
```

função C

```c
int addition (int a, int b){
    return a + b;
}
```

Funções

- Uma função pode retornar resultados ao programa que a chamou
  - O tipo de retorno é definido na definição da função
  - Uma função void não retorna nada

função Python

```
1  def funcTeste( a , b ):
2      print("Recebeu A: %s e B: %s"% (a,b));
```

função C

```
1  void funcTeste (int a, int b){
2      printf("Recebeu A: %d e B: %d", a,b);
3  }
```

Funções

- Definição de função
  - Sintaxe

```
1   tipo_retorno nomeFuncao (tipo paramentro, tipo parametro2){
2       //bloco
3   }
```

Qualquer tipo que quiser: int, float, double, char, etc.
Se **declarar** um **tipo** precisa de **retorno**
Se colocar **void** não **retorna nada**

```
1   int nomeFuncao (tipo paramentro, tipo parametro2){
2       //bloco
3       return valore;
4   }
```

Funções

- Definição de função
  - Sintaxe

```
1   tipo_retorno nomeFuncao (tipo paramentro, tipo parametro2){
2       //bloco
3   }
```

Pode colocar o nome que quiser, mas ele não pode conter caracteres especiais.

Funções

- Definição de função
  - Sintaxe

```
1   tipo_retorno nomeFuncao (tipo paramentro, tipo parametro2){
2       //bloco
3   }
```

Aceita quantos parâmetros forem necessários. Sempre seguindo o padrão: **tipo nomeVar**
Pode deixar vazio se não quiser parâmetros.

```
1   void nomeFuncao (){
2       //bloco
3       printf("função sem parametro");
4       return; //opcional
5   }
```

Funções

- Exemplo Programa com função
  - Lê um valor inteiro e o eleva ao quadrado

```c
#include <stdio.h>

int elevaAoQuadrado (int val){
    return val*val;
}

int main(){
    int a, resultado;

    scanf("%d", &a); //lê valor interiro

    resultado = elevaAoQuadrado(a); //chamda de função

    printf("%d ^ 2 = %d\n", a,resultado);

    return (0);
}
```

Funções

- Exemplo Programa com função
  - Lê valores inteiros maiores que 0 e diz se é par ou ímpar, se ler um valor negativo ou zero o programa para.

```c
#include <stdio.h>

void parOuImpar  (int x){
    if ((x % 2) == 0){
        printf("Par!\n");
    } else {
        printf("Impar!\n");
    }
}

int main(){
    int a, resultado;

    scanf("%d", &a); //lê valor interiro
    while (a > 0) {
        parOuImpar(a);
        scanf("%d", &a); //lê valor interiro
    }

    return (0);
}
```

Funções Exercícios

1. Faça uma função que lê dois inteiros e subtrai
2. Crie uma função em linguagem C que receba 2 números e retorne o maior valor.
3. Crie uma função em linguagem C que receba 3 números e retorne o maior valor, utilizando uma chamada para função anterior.
4. Crie um aplicativo de conversão entre as temperaturas Celsius e Farenheit.
    a. Primeiro o usuário deve escolher se vai entrar com a temperatura em Célsius ou Farenheit, depois a conversão escolhida é realizada.
    b. Se C é a temperatura em Celsius e F em Farenheit, as fórmulas de conversão são:
        i. C= 5.(F-32)/9
        ii. F= (9.C/5) + 32