Linguagem C: Variáveis Indexadas - Vetores e Strings

Prof. Paulo R. S. L. Coelho
paulo@facom.ufu.br

Faculdade de Computação
Universidade Federal de Uberlândia

Organização

1. Vetores
   - Introdução
   - Exemplo - variáveis escalares
   - Exemplo - variáveis indexadas
   - Exercícios
2. Strings
   - Introdução
   - Leitura e escrita de strings
   - Funções de manipulação de strings
   - Exercícios

Organização

Introdução

- As variáveis usadas até agora são chamadas **variáveis escalares**.
- Além destas, a linguagem C disponibiliza **variáveis estruturadas**.
- Entre elas, temos as **variáveis estruturadas homogêneas** ou **variáveis indexadas**, que contemplam os **vetores** e as **matrizes**.

Definição

- Um **vetor** é uma variável indexada unidimensional, formada por uma sequência de variáveis.
- Todas essas variáveis são do mesmo tipo, ocupam regiões adjacentes de memória e são acessadas pelo mesmo identificador.
- Desta forma, o que as distingue é um índice, que referencia sua localização dentro dessa estrutura.

Vetores em C

- Os índices utilizados na linguagem C para identificar posições de um vetor começam sempre em 0 (zero) e vão até o tamanho do vetor menos uma unidade;
- Na linguagem C, um vetor é declarado da seguinte maneira:

```
tipo variavel[tamanho];
```

Ex:

```
int vetor[10]; //vetor de tamanho 10
```

Este vetor é acessado da através dos índices de 0 a 9:

```
vetor[4] = 88; //quinta posicao recebe 88
```

- Um vetor pode ser inicializado na sua declaração:

```
int vetor[5] = {0, 0, 0, 0, 0};
```

Exemplo - variáveis escalares

O programa a seguir contabiliza o voto de 6 candidatos:

```
#include <stdio.h>
int main() {
  int n, cand0, cand1, cand2, cand3, cand4, cand5, nulos, voto, i;
  printf("entre numero de votos\n");
  scanf("%d", &n);
  cand0 = cand1 = cand2 = cand3 = cand4 = cand5 = nulos = 0;
  for (i = 1; i <= n; i++) {
    printf("\t\tVoto: "); scanf("%d", &voto);
    switch(voto) {
      case 0: cand0++; break; case 1: cand1++; break;
      case 2: cand2++; break; case 3: cand3++; break;
      case 4: cand4++; break; case 5: cand5++; break;
      default: nulos++;
    }
  }
  printf("\n\nResultados:\n\n");
  printf("cand0: %d\ncand1: %d\ncand2: %d\n", cand0, cand1, cand2);
  printf("cand3: %d\ncand4: %d\ncand4: %d\n", cand3, cand4, cand5);
  printf("nulos: %d", nulos);
  return 0;
}
```

Exemplo - variáveis indexadas

Agora, o mesmo exemplo é implementado usando variaveis indexadas:

```
#include <stdio.h>
int main() {
  int n, cand[6], nulos, voto, i;
  printf("entre numero de votos\n"); scanf("%d", &n);
  for (i = 0; i < 6; i++) {
    cand[i] = 0;
  }
  nulos = 0;
  for (i = 1; i <= n; i++) {
    printf("\t\tVoto: "); scanf("%d", &voto);
    if (voto >= 0 && voto <= 5) {
      cand[voto]++;
    } else {
      nulos++;
    }
  }
  printf("\n\nResultados:\n\n");
  for (i = 0; i < 6; i++) {
    printf("cand[%d]: %d votos\n", i, cand[i]);
  }
  printf("nulos: %d votos", nulos);
  return 0;
}
```

Exercícios

1. Escrever um programa para ler um vetor de números inteiros de tamanho 20 e imprimir o maior número, o menor e a média dos números.
2. Escreva um programa para:
   - ler dois vetores A e B de números reais, com 10 elementos cada;
   - formar e escrever dois outros vetores C e D de 10 números reais, tais que:
     C[i] = max(A[i], B[i]) e D[i] = média(A[i], B[i]), para
     $0 <= i < 10$

Respostas I

1.
```c
#include <stdio.h>

int main(int argc, char** argv) {
    int vetor[20], i, maior, menor;
    float media;
    for(i = 0; i < 20; i++) {
        printf("Entre elemento %d: ", i);
        scanf("%d", &vetor[i]);
    }
    media = 0;
    maior = menor = vetor[0];
    for(i = 1; i < 20; i++) {
        if(vetor[i] > maior)
            maior = vetor[i];
        if(vetor[i] < menor)
            menor = vetor[i];
        media += vetor[i];
    }
    media = media/20.0;
    printf("o maior eh %d, o menor eh %d e a media eh %.2f\n",
                maior, menor, media);
    return 0;
}
```

Respostas II

2.
```
#include <stdio.h>
int main() {
    float a[10], b[10], c[10], d[10];
    int i;
    for(i = 0; i < 10; i++) {
        printf("Entre elemento a[%d]: ", i);
        scanf("%f", &a[i]);
        printf("Entre elemento b[%d]: ", i);
        scanf("%f", &b[i]);
    }
    for(i = 0; i < 10; i++) {
        c[i] = a[i] > b[i] ? a[i] : b[i];
        d[i] = (a[i] + b[i]) / 2;
    }
    printf("  %s   |   %s   |   %s   |   %s   \n",
           "A", "B", "C", "D");
    for(i = 0; i < 10; i++) {
        printf("%5.2f | %5.2f | %5.2f | %5.2f \n",
               a[i], b[i], c[i], d[i]);
    }
    return 0;
}
```

Organização

Introdução I

- Vetores ou variáveis indexadas de caracteres são denominadas **cadeia de caracteres** ou ***strings***.
- Esses vetores podem ser manipulados de maneira diferenciada em relação aos vetores numéricos pela maioria das linguagens de programação (inclusive C).
- Uma cadeia de caracteres é formada pela sequência de caracteres mais o caractere '`\0`', que é o **finalizador** da cadeia.
- A declaração é feita de forma semelhante:

```
char cad[30];
```

Introdução II

- A declaração com inicialização pode ser feita da seguinte maneira:

```
char cad[ ] = {'A', 'u', 'l', 'a', '\0'};
char cad[5] = "Aula";
char cad[5] = {'A', 'u', 'l', 'a', '\0'};
```

- Além disso, isso é permitido:

```
cad[3] = 'a';
```

mas isso não é:

```
cad = "xyrz";
```

Leitura e escrita de strings I

- O formato `"%s"` pode ser usado nas funções `printf` e `scanf` para fazer leitura e escrita de variáveis do tipo cadeia de caracteres (strings).
- No `printf`, os caracteres da variável são escritos até que o '`\0`' seja encontrado (ele não é escrito).
- No `scanf`, esse formato faz inicialmente a função procurar no *buffer* do teclado o primeiro caractere diferente de espaço e do *enter*. Quando encontra, ela passa esses caracteres para variável até encontrar o próximo espaço ou *enter* e, no lugar desses, ela coloca um '`\0`'.

Leitura e escrita de strings II

- Ex.:

```
char cad[10];

scanf("%s", cad); // repare que aqui nao é usado o '&'
printf("a string lida foi %s", cad);
```

- A biblioteca `stdlib.h` fornece a função `gets` para leitura de cadeia de caracteres contendo espaço;
- Antes da leitura de uma string, recomenda-se esvaziar o *buffer* do teclado com o comando: `fflush(stdin)`.

Funções de manipulação de strings

A biblioteca `string.h` possui várias funções para manipulação de strings. Entre elas temos:

strlen(s1) : retorna o número de caracteres da string, sem contabilizar o '`\0`'.

strcat(s1, s2) : concatena as duas strings recebidas como argumento e coloca o resultado na string passada como primeiro argumento.

strcmp(s1, s2) : compara lexicograficamente duas strings passadas como parâmetros. Seu retorno é menor, igual ou maior que zero, dependendo de a primeira string ser menor, igual ou maior que a segunda string.

strcpy(s1, s2) : recebe duas strings como parâmetros, copiando o conteúdo da segunda para a primeira.

Exercícios

1. Faça um programa que lê três strings (podem conter espaços em branco) e as imprime em ordena lexicograficamente.
2. Faça um programa que leia o nome e o sobrenome de 5 pessoas e concatene-os em um só string, imprimindo essa string concatenada, bem como o seu tamanho.

Respostas I

1.
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char cad1[30], cad2[30], cad3[30];
    printf("entre 3 strings:\n");
    gets(cad1);
    gets(cad2);
    gets(cad3);
    if (strcmp(cad1, cad2) < 0) {
        if(strcmp(cad1, cad3) < 0) {
            printf("1-%s\n", cad1);
            if (strcmp(cad2, cad3) < 0) {
                printf("2-%s\n", cad2);
                printf("3-%s\n", cad3);
            }
            else { //cad3 < cad2
                printf("2-%s\n", cad3);
                printf("3-%s\n", cad2);
            }
        } else { //cad3 < cad1
            printf("1-%s\n", cad3);
```

Respostas II

```
            printf("2-%s\n", cad1);
            printf("3-%s\n", cad2);
        }
    } else { // cad2 < cad1
        if(strcmp(cad2, cad3) < 0) {
            printf("1-%s\n", cad2);
            if (strcmp(cad1, cad3) < 0) {
                printf("2-%s\n", cad1);
                printf("3-%s\n", cad3);
            }
            else { //cad3 < cad2
                printf("2-%s\n", cad3);
                printf("3-%s\n", cad1);
            }
        } else { //cad3 < cad2
            printf("1-%s\n", cad3);
            printf("2-%s\n", cad2);
            printf("3-%s\n", cad1);
        }
    }
    return 0;
}
```

Respostas III

2.
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char nome[60], sobrenome[30];
    int i;
    for (i = 0; i < 5; i++) {
        printf("entre nome: ");
        scanf("%s", nome);
        printf("entre sobrenome: ");
        scanf("%s", sobrenome);
        strcat(nome, " ");
        strcat(nome, sobrenome);
        printf("\nNome Completo: %s\n",nome);
        printf("Tamanho do Nome Completo: %d\n", strlen(nome));
    }
    return 0;
}
```