Disciplina:
Programação Computacional

Prof. Fernando Rodrigues
e-mail: fernandorodrigues@sobral.ufc.br

Aula 10: Programação em C

- Estruturas de dados homogêneas;
- Conceito de estruturas de dados estáticas;
- Vetores e matrizes;
- Manipulação de cadeias de caracteres (strings).

Vetores unidimensionais (Arrays)

- São estruturas de dados homogêneas, ou seja, formam um conjunto ordenado de dados onde todos os elementos são do mesmo tipo.
- São referenciadas por um nome comum, e acessadas através de um índice.
- A estrutura de dados é dita estática quando tem um tamanho definido, onde os dados ficam em posições contíguas na memória . Ex: int vet[3];

Vetores unidimensionais (Arrays)

- Sintaxe:
  - tipo nome_var[tamanho];

```
9          int vet[3];
```

- Onde tamanho precisa ser um valor inteiro ou uma expressão avaliada como um inteiro, podendo conter variáveis e literais.

Ex: #define N 20

int vet[N+5];

- Precisa ser declarado e ter seu tamanho definido, para o compilador alocar espaço para cada célula do vetor na memória.
- Matrizes e ponteiros estão intimamente ligados em C.
- Vet é um ponteiro para o elemento vet[0]:
  - Por conta disso vet e &vet[0] são a mesma coisa, a referência para o endereço de memória do primeiro elemento

Vetores unidimensionais (Arrays)

- Percorrendo e inicializando um vetor

```
9       int vet[3];
10
11      for(int i = 0; i < 3 ; i++)
12              vet[i] = i+1;
```

- Lendo valores em um vetor

```
9       int vet[3];
10
11      for(int i = 0; i < 3 ; i++){
12              printf("Digite o valor: ");
13              scanf("%d",&vet[i]);
14      }
```

- Índice é diferente de posição
  - 0 1 2 3 4 - índices
  - 5 10 12 8 9 - valores
  - 1 2 3 4 5 - posição

Matrizes bidimensionais

- C suporta matrizes multidimensionais.
- A forma mais simples de matriz multidimensional é a bidimensional.
- É uma matriz de matrizes unidimensionais, ou seja, é um vetor de vetores.

Matrizes bidimensionais (Matrizes)

- Sintaxe

<tipo_de_dado> <nome_matriz>[tamanho1] [tamanho2];

```
9       int mat[3][3];
```

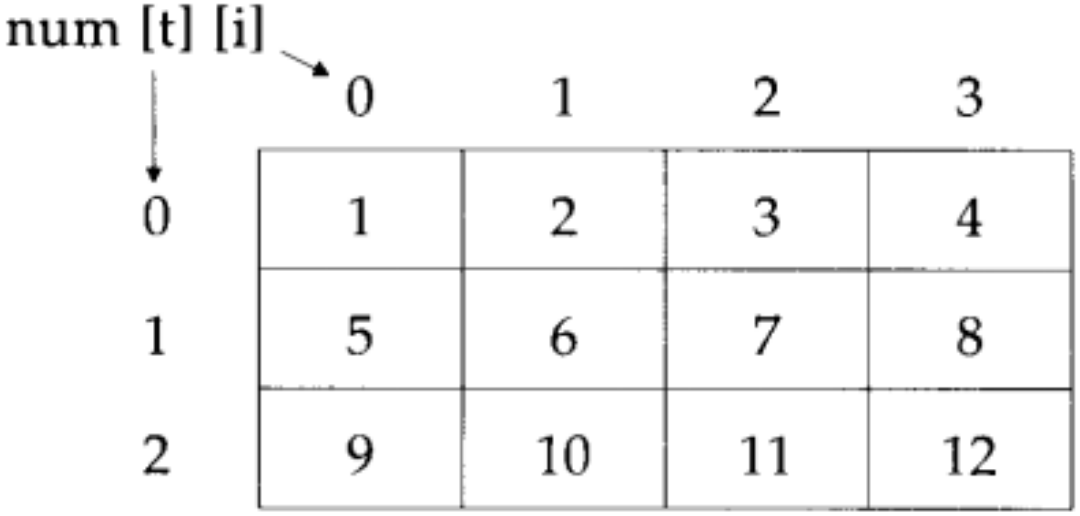
```
#include <stdio.h>

void main(void)
{
  int t, i, num[3][4];

  for(t=0; t<3; ++t)
    for(i=0; i<4; ++i)
      num[t][i] = (t*4)+i+1;

/* agora escreva-os */
  for(t=0; t<3; ++t) {
    for(i=0; i<4; ++i)
      printf("%3d ", num[t][i]);
    printf("\n");
  }
}
```

Matrizes bidimensionais (Matrizes)



Matrizes multidimensionais

- Sintaxe
  - <tipo> <nome>[tamanho1][tamanho2]...[tamanhoN];
  - Ex:
    - int matNum[3][5][2];
    - float matPesos[5][2][4][5];
    - double valores[10][20][30];

Matrizes

- Podemos aplicar o conceito de matrizes a todos os tipos de dados, sendo ele um tipo básico, uma estrutura homogênea ou heterogênea(composta).

- Ex: char vet[] = “teste”;

Inicialização de vetores e matrizes

- int vet[ ] = {1,2,3,4,5,6};
  - vet[6]
- int mat[ ][3] = {1,2,3,4,5,6};
  - Aqui deve ser informado pelo menos o número de colunas
  - mat[2][3]
- char string[ ] = “Mensagem”;
  - string[9];

O tipo **char**

Variáveis ou constantes do tipo **char** são usadas para amazenar caracteres.

Na atribuição de valores a variáveis do tipo **char** os símbolos devem ser escritos entre aspas simples ' '.

Usa-se o código de formatação %c para ler ou exibir valores do tipo **char**.

```
int main()
{
  char c1, c2, c3;

  c1 = 'O';
  c2 = 'b';
  c3 = 'A';
  printf("%c %c %c\n", c1, c2, c3);   // exibe O b A
  printf("%c%c%c\n", c2, c1, c3);     //exibe bOA

  system("PAUSE");
  return(0);
}
```

Funções nativas de entrada e saída

- `getchar()`: lê um caractere até que a tecla <ENTER> seja pressionada. Se mais de um caractere for digitado, apenas o primeiro caractere será considerado e o restante será descartado.

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

int main()
{
  char a;

  printf("Digite um caractere: ");
  a = getchar();          // armazena a entrada até pressionar <ENTER>
  printf("Caractere digitado: %c\n", a);
  system("PAUSE");
  return(0);
}
```

Funções nativas de entrada e saída

- `putchar( )`: Exibe na tela o caractere passado como argumento, a partir da posição atual do cursor.

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

int main()
{
  char a;

  printf("Digite uma letra minuscula: ");
  a = getchar();
  putchar( toupper(a) );
  putchar('\n')
  system("PAUSE");
  return(0);
}
```

A função puts ( )

- É utilizada apenas para exibir mensagens na tela.
- A mensagem a ser exibida deverá ser escrita entre aspas.
- Após a exibição da mensagem, a função puts ( ) muda de linha automaticamente.

```
int main()
{
  puts("Digite sua opcao:");
  puts("[1] Consultar");
  puts("[2] Incluir");
  puts("[3] Atualizar");
  puts("[4] Excluir");
  puts("[5] Encerrar");
  printf("-> ");
  ...
}
```

Trabalhando com vetores (atribuição)

- Temos que tomar cuidado ao trabalhar com vetores na operação de atribuição, pois não pode ser feita atribuição direta, como ilustrado abaixo:

Não se pode fazer atribuição de arrays inteiros, apenas de suas posições individualmente.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int v[5] = {1,2,3,4,5};
05    int v1[5];
06    v1 = v; //ERRADO!
07    int i;
08    for(i=0; i<5; i++)
09       v1[i] = v[i]; //CORRETO
10    system("pause");
11    return 0;
12  }
```

Trabalhando com strings (atribuição)

- Como strings são vetores de caracteres em C, então acontece a mesma coisa em relação a operação de atribuição:

Strings são arrays. Portanto, não se pode fazer atribuição de strings.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    char str1[20] = "Hello World";
05    char str2[20];
06
07    str1 = str2; //ERRADO!
08
09    system("pause");
10    return 0;
11  }
```

Atribuição de strings (correto)

- O correto seria fazer como a seguir:

Exemplo: copiando uma string

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  int main(){
04    int i;
05    char str1[20] = "Hello World";
06    char str2[20];
07    for (i = 0; str1[i]!='\0'; i++)
08       str2[i] = str1[i];
09    str2[i] = '\0';
10    system("pause");
11    return 0;
12  }
```

Lendo uma string do teclado

- Usando a função scanf()
  - Existem várias maneiras de se fazer a leitura de uma sequência de caracteres do teclado. Uma delas é utilizando a já conhecida função scanf() com o formato de dados “%s”:
  - Ex:

```
char str[20];
scanf(“%s”,str);
```

Quando usamos a função **scanf()** para ler uma string, o símbolo de & antes do nome da variável não é utilizado. Os colchetes também não são utilizados, pois queremos ler a string toda e não apenas uma letra.

A função scanf() padrão (com uso da string de formatação "%s") lê apenas strings digitadas sem espaços em branco, ou seja, somente uma palavra.

Lendo uma string do teclado

- Usando a função scanf()
  - Para se fazer a leitura de uma sequência de caracteres (string) do teclado, incluindo os espaços em branco, com a função scanf(), pode-se utilizar a seguinte string de formatação de dados: “%[^\n]s”, que fará a leitura até que uma quebra de linha seja encontrada.
  - Ex:

```
char *a;
scanf("%[^\n]s",a);
```

Limpando o buffer do teclado

- Usando a função setbuf()
  - Às vezes, podem ocorrer erros durante a leitura de caracteres ou strings do teclado. Para resolver esses pequenos erros, podemos limpar o buffer do teclado (entrada-padrão) usando a função setbuf(stdin, NULL) antes de realizar a leitura de caracteres ou strings:
  - Ex:

```
char ch;
setbuf(stdin, NULL);
scanf("%c", &ch);
```

```
char str[10];
setbuf(stdin, NULL);
gets(str);
```

Lendo uma string do teclado

- Usando a função gets()
  - Uma alternativa mais eficiente para a leitura de uma string é a função gets(), a qual faz a leitura do teclado considerando todos os caracteres digitados (incluindo os espaços em branco) até encontrar uma tecla <Enter>:
  - Ex:

```
char str[20];
gets(str);
```

Funções de manipulação de caracteres

```
char a;
```

- `isalpha(a)`: testa se o caractere é uma letra.
- `isdigit(a)`: testa se o caractere é um algarismo.
- `isespace(a)`: testa se é o caractere espaço `' '`.
- `islower(a)`: testa se é uma letra minúscula.
- `isupper(a)`: testa se é uma letra maiúscula.
- `tolower(a)`: converte o caractere para minúscula.
- `toupper(a)`: converte o caractere para maiúscula.

Estas funções estão definidas na biblioteca `ctype.h`.

Funções para manipulação de strings

Nome	Função
strcpy(s1, s2)	Copia s2 em s1.
strcat(s1, s2)	Concatena s2 ao final de s1.
strlen(s1)	Retorna o tamanho de s1.
strcmp(s1, s2)	Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2.
strchr(s1, ch)	Retorna um ponteiro para a primeira ocorrência de ch em s1.
strstr(s1, s2)	Retorna um ponteiro para a primeira ocorrência de s2 em s1.

Essas funções usam o cabeçalho padrão **STRING.H**.

Operações em string: biblioteca string.h

- Tamanho:
  - size_t strlen(const char * str);
    - Retorna o número de caracteres da string passada.
  - Ex:

```
char s1[80], s2[80];

gets(s1);

gets(s2);

printf("comprimentos: %d %d\n", strlen(s1), strlen(s2));
```

- Cópia:
  - char * strcpy(char *, const char *);
    - Copia todo o conteúdo da segunda string para a primeira string.
  - char * strncpy(char *, const char *, size_t);
    - Copia os “n”primeiros caracteres(indicados por size_t) da segunda string para a primeira string.
  - int sprintf(char *, const char *, …);
    - Grava dados formatados em uma string.

Operações em string: biblioteca string.h

- Concatenação:
  - char * strcat(char *, const char *);
    - Concatena o conteúdo da segunda string no final da primeira string.
  - char * strncat(char *, const char *, size_t);
    - Adiciona (concatena) os “n” primeiros caracteres da segunda string no final da primeira string.
- Inversão:
  - char * strrev(char *);
    - Inverte a string passada sobre ela mesma.

Operações em string: biblioteca string.h

- Busca:
  - char *strchr(const char *str, int ch);
    - Retorna um ponteiro para a localização em que o caractere 'ch' aparece pela primeira vez na string apontada por *str, ou NULL se não encontrar.
  - char *strrchr(const char *str, int ch);
    - Faz a mesma coisa da função anterior, mas ao invés de localizar a primeira ocorrência de 'ch', localiza e retorna a última ocorrência.
  - char * strstr(const char * strOrigem, char * strChave);
    - Retorna um ponteiro para a primeira ocorrência da string apontada por strChave na string apontada por strOrigem.

Operações em string: biblioteca string.h

- Comparação:
  - strcmp(s1,s2);
    - Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética).
  - stricmp(s1,s2);
    - Retorna 0 se s1 e s2 são iguais; menor que 0 se s1<s2; maior que 0 se s1>s2 (comparação alfabética). Essa função considera letras maiúsculas ou minúsculas como símbolos iguais.

Exercício 3 – Ling. C (Strings)

1) Faça um programa que leia uma string e a imprima na tela.

2) Faça um programa que leia uma string e imprima as quatro primeiras letras dela.

3) Sem usar a função strlen(), faça um programa que leia uma string e imprima quantos caracteres ela possui.

4) Faça um programa que leia uma string e a imprima de trás para a frente.

5) Faça um programa que leia uma string e a inverta. A string invertida deve ser armazenada na mesma variável. Em seguida, imprima a string invertida.

6) Leia uma string do teclado e conte quantas vogais (a, e, i, o, u) ela possui. Entre com um caractere (vogal ou consoante) e substitua todas as vogais da palavra dada por esse caractere. Ao final, imprima a nova string e o número de vogais que ela possui.

7) Faça um programa que leia uma string e imprima uma mensagem dizendo se ela é um palíndromo ou não. Um palíndromo é uma palavra que tem a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita.

   Exemplos: ovo, arara, rever, asa, osso, ana etc.

8) Leia dois nomes (strings de caracteres), compare se os mesmos são iguais e guarde o resultado em uma variável lógica “saoIguais”. Por fim, teste se “saoIguais” é VERDADEIRO: Se sim, imprima: “Nomes iguais”. Caso contrário, imprima “Nomes diferentes”.

Exercício 3 – Ling. C (Strings)

9) Construa um programa que leia duas strings do teclado. Imprima uma mensagem informando quantas vezes a segunda string lida está contida dentro da primeira.

10) Escreva um programa que leia uma string do teclado e converta todos os seus caracteres em maiúscula. Dica: subtraia 32 dos caracteres cujo código ASCII está entre 97 e 122.

11) Escreva um programa que leia uma string do teclado e converta todos os seus caracteres em minúscula. Dica: some 32 dos caracteres cujo código ASCII está entre 65 e 90.

12) Construa um programa que leia duas strings do teclado. Imprima uma mensagem informando se a segunda string lida está contida dentro da primeira.

13) Escreva um programa que leia o nome e o valor de determinada mercadoria de uma loja. Sabendo que o desconto para pagamento à vista é de 10% sobre o valor total, calcule o valor a ser pago à vista. Escreva o nome da mercadoria, o valor total, o valor do desconto e o valor a ser pago à vista.

14) Escreva um programa que recebe uma string S e dois valores inteiros não negativos i e j. Em seguida, imprima os caracteres contidos no segmento que vai de i a j da string S.

FIM