

Tecnologia em Análise e Desenvolvimento de Sistemas - TADS

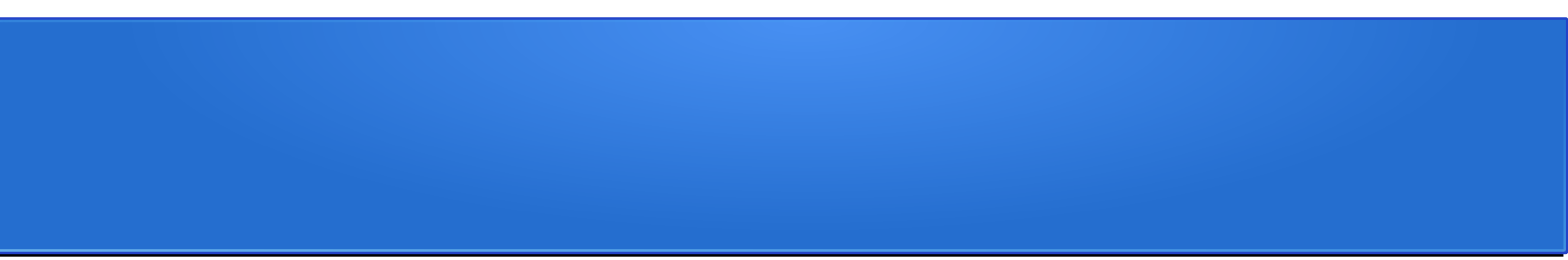
Estrutura de Dados

TADS / IFRS - 2024-1

Prof. Luciano Vargas Gonçalves

E-mail: luciano.goncalves@riogrande.ifrs.edu.br





Aula 2 – Funções, Vetores e Strings

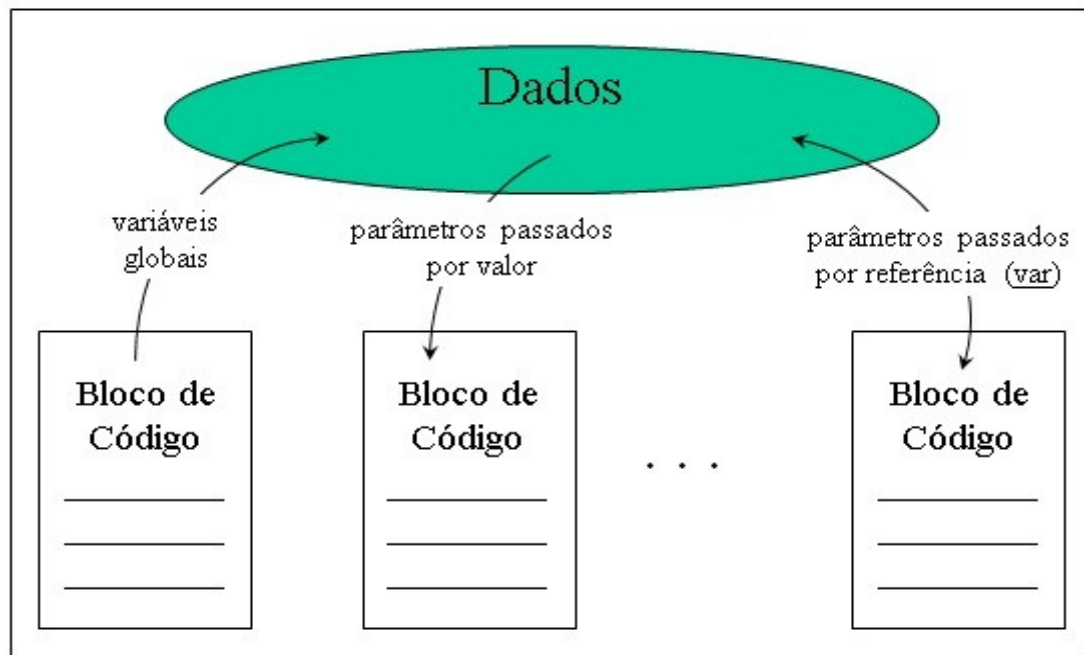
Programação estruturada - Funções

- **Programação estruturada**

- Partindo-se de um dado problema, para qual se deseja encontrar um programa de solução, deve-se procurar subdividi-lo em problemas menores (subproblemas) e consequentemente de solução mais simples (estratégia - dividir para conquistar).
- Subproblemas terão solução imediata (término) ou para os quais não for possível encontrar uma solução direta devem ser novamente subdivididos. Assim, o processo é repetido até que se consiga encontrar um subprograma para solucionar cada um dos subproblemas subjacentes.
- Então, o programa solução do problema original será composto pela justaposição dos subprogramas usados para solucionar cada um dos subproblemas em que o problema original foi decomposto.
- Subproblemas podem ser definidos e implementados com o uso de funções (tem retorno), ou procedimentos (sem retorno);

Programação estruturada - Funções

- Compartilhamento dos dados entre as partes do sistema;
 - Bloco de código ou função. Dados em memória



Função Main e Variáveis Locais

- Main ()
 - Primeira função a ser localizada na execução de um programa
 - Função principal

```
int main (){  
    //variáveis locais, acesso apenas no Main  
    int ano, dia, mes;  
  
    dia = 15;  
    mes = 3;  
    ano = 2024;  
  
    printf("\n Data de Hoje. %d/%d/%d é uma Sexta Feira \n",dia,mes,ano);  
}
```

Variáveis de escopo Local

Variáveis e Constates Globais

- Podemos declarar variáveis globais e constantes
 - Podem ser acessadas em qualquer parte do código;

```
//definicao de uma constante global
#define anoAtual 2024

//declaracao de uma variavel global
int anoNascimento = 1990;
```

Realizado fora das funções, normalmente no início do programa

```
int main (){
    //variáveis locais, acesso apenas no Main
    int ano, dia, mes;
    dia = 15; mes = 3; ano = 2024;

    printf("\n Data de Hoje. %d/%d/%d é uma Sexta Feira \n",dia,mes,ano);
    printf("\n Ano passado dia %d/%d/%d era uma Quarta Feira \n",dia,mes,anoAtual);
    printf("\n Quem nasceu no dia  %d/%d/%d era uma Quinta Feira \n",dia,mes,anoNascimento);
}
```

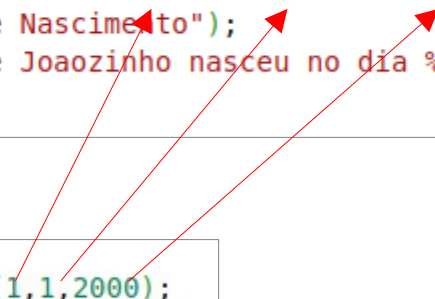
Funções – Passagem por Valor

- Passagem por valor é a troca de informações entre as partes do sistema (funções)
 - Os dados são copiados para novas variáveis dentro da função que os recebe. Duplicidade de informação.

```
//função passagem por valor
void imprimeCertidaoNascimento(int dia, int mes, int ano){
    printf("\n\n Certidão de Nascimento");
    printf("\n Certifico que Joaozinho nasceu no dia %d/%d/%d\n",dia,mes,ano);
}
```

Chamada da Função

```
imprimeCertidaoNascimento(1,1,2000);
```



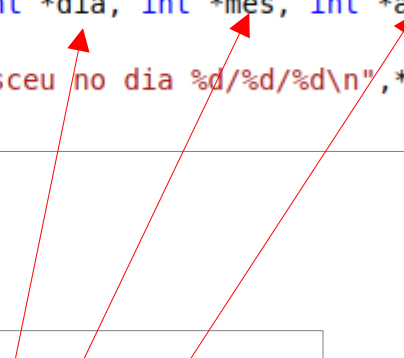
Funções – Passagem por Referência

- Passagem por referência;
 - São passados ponteiros, endereços das informações na memória. Não há duplicidade de informação.

```
//função passagem por referencia
void imprimeCertidaoNascimentoReferencia(int *dia, int *mes, int *ano){
    printf("\n\n Certidão de Nascimento");
    printf("\n Certifico que Mariazinha nasceu no dia %d/%d/%d\n",*dia,*mes,*ano);
}
```

Chamada da Função

```
imprimeCertidaoNascimentoReferencia(&dia,&mes,&ano);
```



Funções - Passagem por Valor

- Troca de valores entre duas variáveis x,y, usando função;

```
void trocal (int a, int b) {  
    int temp;  
    temp = a;  
    a = b;  
    b = temp;  
}
```

```
int main()  
{  
    printf("\n Aula de Ponteiros!! \n");  
    int x = 10, y = 20;  
  
    //passagem por valor  
    printf("\n Troca 1 - Valor de X=%d e Y=%d",x,y);  
    trocal(x,y);  
    printf("\n Troca 1 - Valor de X=%d e Y=%d",x,y);  
}
```

Valor X=10 e Y=20

Valor X=10 e Y=20

Troca não realizada

Passagem por Valor

- Troca de valores entre duas variáveis x,y;

```
void trocal (int a, int b) {  
    int temp;  
    temp = a;  
    a = b;  
    b = temp;  
}
```

```
int main()  
{  
    printf("\n Aula de Ponteiros!! \n");  
    int x = 10, y = 20;  
  
    //passagem por valor  
    printf("\n Troca 1 - Valor de X=%d e Y=%d",x,y);  
    trocal(x,y);  
    printf("\n Troca 1 - Valor de X=%d e Y=%d",x,y);  
}
```

Valor X=10 e Y=20

Valor X=10 e Y=20

Endereço	Valor	Variável
e943ab	10	X
e943ac		
e943ad	20	y
e943ae		
e943af	10	a
e943b0	20	b
e943b1		
e943b2		

Troca não realizada

Passagem por Valor

- Troca de valores entre duas variáveis x,y;

```
void trocal (int a, int b) {  
    int temp;  
    temp = a;  
    a = b;  
    b = temp;  
}
```

Troca na função,
duplicidade de
informação

```
int main()  
{  
    printf("\n Aula de Ponteiros!! \n");  
    int x = 10, y = 20;  
  
    //passagem por valor  
    printf("\n Troca 1 - Valor de X=%d e Y=%d",x,y);  
    trocal(x,y);  
    printf("\n Troca 1 - Valor de X=%d e Y=%d",x,y);  
}
```

Valor X=10 e Y=20

Valor X=10 e Y=20

Endereço	Valor	Variável
e943ab	10	x
e943ac		
e943ad	20	y
e943ae		
e943af	20	a
e943b0	10	b
e943b1		
e943b2		

Troca não realizada no main

Ponteiros e Passagem por Referência

- Troca de valores entre duas variáveis x,y através de ponteiros;

```
void troca2 (int *pa, int *pb) {  
    int temp;  
    temp = *pa;  
    *pa = *pb;  
    *pb = temp;  
}
```

Ponteiros *pa e *pb

Passagem por Referência

```
int main()  
{  
    printf("\n Aula de Ponteiros!! \n");  
    int x, y;  
    //passagem por referencia  
    x = 10, y = 20;  
    printf("\n Troca 2 - Valor de X=%d e Y=%d",x,y);  
    troca2(&x,&y);  
    printf("\n Troca 2 - Valor de X=%d e Y=%d",x,y);  
}
```

Valor X=10 e Y=20

Valor X=20 e Y=10

Troca realizada

Ponteiros e Passagem por Referência

- Troca de valores entre duas variáveis x,y;

```
void troca2 (int *pa, int *pb) {  
    int temp;  
    temp = *pa;  
    *pa = *pb;  
    *pb = temp;  
}
```

Uso de ponteiros na
troca de informação

```
int main()  
{  
    printf("\n Aula de Ponteiros!! \n");  
    int x, y;  
    //passagem por referencia  
    x = 10, y = 20;  
    printf("\n Troca 2 - Valor de X=%d e Y=%d",x,y);  
    troca2(&x,&y);  
    printf("\n Troca 2 - Valor de X=%d e Y=%d",x,y);  
}
```

Endereço	Valor	Variável
e943ab	10	X
e943ac		
e943ad	20	y
e943ae		
e943af	e943ab	pa
e943b0	e943ad	pb
e943b1		
e943b2		

Ponteiros e Passagem por Referência

- Troca de valores entre duas variáveis x,y;

```
void troca2 (int *pa, int *pb) {  
    int temp;  
    temp = *pa;  
    *pa = *pb;  
    *pb = temp;  
}
```

```
int main()  
{  
    printf("\n Aula de Ponteiros!! \n");  
    int x, y;  
    //passagem por referencia  
    x = 10, y = 20;  
    printf("\n Troca 2 - Valor de X=%d e Y=%d",x,y);  
    troca2(&x,&y);  
    printf("\n Troca 2 - Valor de X=%d e Y=%d",x,y);  
}
```

Aula de Ponteiros!!

Troca 2 - Valor de X=10 e Y=20
Troca 2 - Valor de X=20 e Y=10

Saída terminal

Endereço	Valor	Variável
e943ab	20	X
e943ac		
e943ad	10	y
e943ae		
e943af	e943ab	pa
e943b0	e943ad	pb
e943b1		
e943b2		

Exercícios

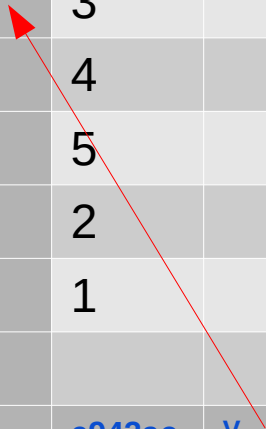
- 1) Crie um procedimento para receber dois valores inteiros (A e B) e uma referência para inteiro (*pc). O ponteiro *pc retornará o resultado da soma de A e B.
- 2) Crie um procedimento para receber duas referências para inteiros (*pa,*pb), some o valor *pa com valor de *pb e atribua o resultado a variável A que deu origem a *pa;
- 3) Crie um procedimento para receber três referências para inteiros (*pa, *pb, *ps), some o valor de *pa com valor de *pb e atribua o resultado a variável referenciada por *ps;
- 4) Crie uma função para receber duas referências para inteiros (*pa, *pb), some o valor de *pa com valor de *pb e atribua a variável S, retorne a referenciada de S;

Vetores e Strings

- Vetores e Strings são ponteiros em C:
 - Um vetor é uma porção de memória contígua de mesmo tipo e o endereço do primeiro elemento é o mesmo do vetor; Os demais elementos estão equidistantes em função do seu tamanho (memória ocupada);
 - Exemplo de um vetor de 5 elementos inteiros
 - Declaração de um vetor de 5 elementos

```
int v[5] = {3,4,5,2,1};
```

Endereço	Valor	Variável
e943ab		
e943ac	3	
e943b0	4	
e943b4	5	
e943b8	2	
e943bc	1	
e943bf		
e943c0	e943ac	v



Execução

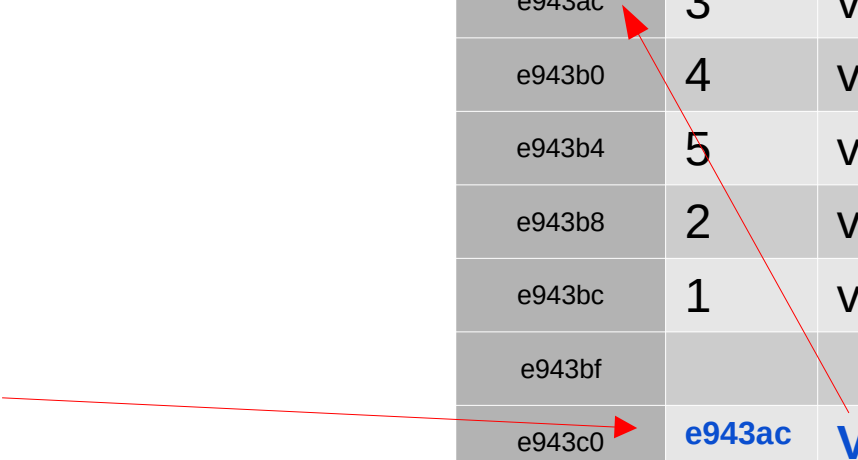
Vetores são Ponteiros

Declaração e passagem de um vetor de inteiros V

```
int main(){  
    int v[5] = {3,4,5,2,1};  
  
    //passagem de um vetor para função  
    imprimeVetor(v,5);  
    imprimeVetorInvertido(v,5);  
}
```

V é um ponteiro para várias (5) posições de memórias do tipo inteiro.

Endereço	Valor	Variável
e943ab		
e943ac	3	v[0]
e943b0	4	v[1]
e943b4	5	v[2]
e943b8	2	v[3]
e943bc	1	v[4]
e943bf		
e943c0	e943ac	V
e943c1		



Vetores são Ponteiros

O vetor pode ser operado como vetor normal com uso de “ [] ”

```
void imprimeVetor(int vt[], int n){  
    printf ("\n Função imprime vetor");  
    printf("\n V = [");  
    int i;  
    for(i=0; i<n-1; i++){  
        printf(" %d,",vt[i]);  
    }  
    printf(" %d ] \n",vt[i]);  
}
```

V é um ponteiro para várias (5) posições de memórias do tipo inteiro.

Endereço	Valor	Variável
e943ab		
e943ac	3	v[0]
e943b0	4	v[1]
e943b4	5	v[2]
e943b8	2	v[3]
e943bc	1	v[4]
e943bf		
e943c0	e943ac	V
e943c1	e943ac	Vt

Vetores são Ponteiros

O vetor pode ser operado como PONTEIRO de vetor normal, com uso de “*(vt +i) = Endereço + Incremento”

```
// função com ponteiro de vetor
void imprimeVetorPonteiro(int *vt, int n){
    printf ("\n Função imprime vetor Ponteiro");
    printf ("\n V = [");
    int i;
    for(i=0; i<n-1; i++){
        printf(" %d,", *(vt+i));
    }
    printf(" %d ] \n", *(vt+i));
}
```

*(vt +i) = Endereço + Incremento”

Endereço	Valor	Variável
e943ab		
e943ac	3	v[0]
e943b0	4	v[1]
e943b4	5	v[2]
e943b8	2	v[3]
e943bc	1	v[4]
e943bf		
e943c0	e943ac	V
e943c1	e943ac	Vt

Vetores são Ponteiros

Os Resultados são os mesmos


```
Função imprime vetor  
V = [ 3, 4, 5, 2, 1 ]
```

```
Função imprime vetor Invertido  
V = [ 1, 2, 5, 4, 3 ]
```

```
Função imprime vetor Ponteiro  
V = [ 3, 4, 5, 2, 1 ]
```

```
Função imprime vetor Invertido Ponteiro  
V = [ 1, 2, 5, 4, 3 ]
```

Endereço	Valor	Variável
e943ab		
e943ac	3	v[0]
e943b0	4	v[1]
e943b4	5	v[2]
e943b8	2	v[3]
e943bc	1	v[4]
e943bf		
e943c0	e943ac	V
e943c1	e943ac	Vt



Vetores

```
void imprimeVetor(int vt[], int n){  
    for (int i=0;i<n;i++){  
        printf("\n v[%d] = %d ",i,vt[i]);  
        printf("\n");  
    }  
}
```

Função sem ponteiro

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

Função Com ponteiro

ponteiro + incremento

```
> void somaDez(int *p, int n){ ...
```

```
int main()
```

```
{  
    int v [5] = {3,4,5,2,1};  
    int n = 5;  
  
    imprimeVetor (v,n); //passagem por referencia  
    imprimeVetorPonteiro (v,n); //passagem por referencia
```

v[0] = 3
v[1] = 4
v[2] = 5
v[3] = 2
v[4] = 1

v[0] = 3
v[1] = 4
v[2] = 5
v[3] = 2
v[4] = 1

Execução

Endereço	Valor	Variável
e943ab		
e943ac	3	v[0]
e943b0	4	v[1]
e943b4	5	v[2]
e943b8	2	v[3]
e943bc	1	v[4]
e943bf		
e943c0	e943ac	v
e943c1	e943ac	p

Vetores

```
void imprimeVetor(int vt[], int n){...
```

```
void imprimeVetorPonteiro(int *p, int n){...
```

```
void somaDez(int *p, int n){  
    for (int i=0;i<n;i++){  
        *(p+i)+=10;  
    }  
}
```

```
int main()  
{  
    int v [5] = {3,4,5,2,1};  
    int n = 5;  
  
    imprimeVetor (v,n); //passagem por referencia  
    imprimeVetorPonteiro (v,n); //passagem por referencia  
    somaDez(v,n);  
    imprimeVetor (v,n); //passagem por referencia
```

Função acrescenta 10 a cada elemento V

v[0] = 3
v[1] = 4
v[2] = 5
v[3] = 2
v[4] = 1

v[0] = 3
v[1] = 4
v[2] = 5
v[3] = 2
v[4] = 1

v[0] = 13
v[1] = 14
v[2] = 15
v[3] = 12
v[4] = 11

Execução

Exercícios de Vetores

- Exercícios

- 1) Escreva uma função **menorVetor** que receba um vetor inteiro **$v[0..n-1]$** , a quantidade de elementos do vetor(**n**), o endereço de uma variável inteira **menor** (passagem por referência), e deposite nesta variável o valor do menor elemento do vetor.

```
void menorVetor(int *vt, int n, int *menor){
```

- 2) Escreva uma função **maiorVetor** que receba um vetor inteiro **$v[0..n-1]$** , a quantidade de elementos do vetor(**n**), o endereço de uma variável inteira **maior**(passagem por referência), e deposite nesta variável o valor do **maior** elemento do vetor.

```
void maiorVetor(int *vt, int n, int *maior){
```

- 3) Escreva uma função **menorMaiorVetor** que receba um vetor inteiro **$v[0..n-1]$** , a quantidade de elementos do vetor(**n**), e os endereços de duas variável inteira **menor e maior** (passagem por referência), e deposite nestas variáveis o valor do menor e maior elemento do vetor. Utilize as funções **menorVetor e maiorVetor implementadas antes**;

```
void menorMaiorVetor(int *vt, int n, int *menor, int *maior){
```

Exercícios de Vetores

- Exercícios

4) Escreva uma função **menorVetor** que receba um vetor inteiro **$v[0..n-1]$** , e a quantidade de elementos do vetor(n), a função retorna o endereço do menor elemento do vetor.

```
int* menorVetorPonteiro(int *vt, int n);
```

5) Escreva uma função **maiorVetor** que receba um vetor inteiro **$v[0..n-1]$** , e a quantidade de elementos do vetor(n), a função retorna o endereço do maior elemento do vetor.

```
int* maiorVetorPonteiro(int *vt, int n);
```

6) Escreva uma função **menorMaiorVetor** que receba um vetor inteiro **$v[0..n-1]$** , a quantidade de elementos do vetor(n), e o endereço de duas variável inteira **menor** e **maior** (passagem por referência), a função retorna os valores do menor e maior elemento do vetor (uso de um vetor).. Utilize as funções **menorVetor** e **maiorVetor** implementadas no passo anterior;

```
int* menorMaiorVetorPonteiro(int *vt, int n);
```


Strings são vetores de Caracteres

```
int main(){  
    //string = Vetor de char  
    char vc [] = "Estrutura de Dados";  
    imprimeString(vc);  
  
    char *pvc = "Estrutura de Dados";  
    imprimeString(pvc);  
}
```

Vc = Vetor de Char;

PVc = ponteiro Vetor de Char;
vc [] = *pvc

Endereço	Valor	Variável
e943ab	E	
e943ac	s	
e943b0	t	
e943b4	r	
e943b8	u	
e943bc	t	
e943bf	u	
e943c0	r	
e943c1	a	
	e943ab	V

Strings são vetores de Caracteres

```
void imprimeString(char *s){  
    printf("\n");  
    for(int i=0; i<strlen(s); i++){  
        printf(" %c",*(s+i));  
    }  
    printf("\n\n");  
}
```

Função imprimeString, recebe ponteiro *s

```
int main(){  
    //string = Vetor de char  
    char vc [] = "Estrutura de Dados";  
    imprimeString(vc);  
  
    char *pvc = "Estrutura de Dados";  
    imprimeString(pvc);  
}
```

Vetores e Strings

```
void imprimeString(char *s){
    printf("\n");
    for(int i=0; i<strlen(s); i++){
        printf(" %c",*(s+i));
    }
    printf("\n\n");
}

void imprimeCodigoAsc(char *s){
    printf("\n\n ");
    for(int i=0; i<strlen(s); i++){
        printf(" %c = %d\n ",*(s+i),*(s+i));
    }
    printf("\n\n");
}

int main(){
    //string = Vetor de char
    char vc [] = "Estrutura de Dados";
    imprimeString(vc);

    char *pvc = "Estrutura de Dados";
    imprimeString(pvc);
    imprimeCodigoAsc(pvc);
}
```

Saídas do programa

S[0] = E
S[1] = s
S[2] = t
S[3] = r
S[4] = u
S[5] = t
S[6] = u
S[7] = r
S[8] = a
S[9] =
S[10] = d
S[11] = e
S[12] =
S[13] = D
S[14] = a
S[15] = d
S[16] = o
S[17] = s
S[18] =

E = 69
s = 115
t = 116
r = 114
u = 117
t = 116
u = 117
r = 114
a = 97
= 32
d = 100
e = 101
= 32
D = 68
a = 97
d = 100
o = 111
s = 115

imprimeString()

imprimeCodigoAsc()

Exercícios de Vetor de Char

- Exercício 1

- Escreva um procedimento para receber uma String(vetor de char), e imprimir na ordem inversa os elementos. `void imprimeStringInvertida(char *s);`

- Escreva um procedimento para receber uma String(vetor de char), e um ponteiro para armazenar todas as vogais presentes na String. `void retornaVogaisString(char *s, char *vg);`

- Escreva um procedimento para receber uma String(vetor de char), e um ponteiro para armazenar todas as consoantes presentes na String.

`void retornaConsoantesString(char *s, char *cs);`

- Escreva uma função para receber uma String(vetor de char), e retorne a quantidade de vogais.

`int retornaQtdVogaisString(char *s);`

- Escreva uma função para receber uma String(vetor de char), e retorne a quantidade de consoantes na String.

`int retornaQtdConsoantesString(char *s);`

-
- Crie um programa para utilizar as funções acima.