



INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Charqueadas

EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Strings em C

Programação Estruturada

Prof. André del Mestre

Principais Conceitos

Strings

Definição

- Vetor de caracteres
- Particularidades
 - Marcação ‘\0’ é o último caracter da string e marca seu final
 - Um vetor de caracteres com tamanho 10 pode armazenar 9 caracteres, pois um deles é o ‘\0’.
 - As posições após continuam ‘\0’ reservadas para o vetor
- Exemplo:

```
char str[10]={'I', 'F', 'S', 'u', 'l', '\0'};
```

I	F	S	u	l	\0				
0	1	2	3	4	5	6	7	8	9
Região com valores válidos						Lixo de memória			

Strings

Declaração e Inicialização

- Inicialização caracter por caracter
 - Similar a inicialização de vetores de outros tipos
- Inicialização por uma string utilizando aspas duplas

- Exemplo:

```
// char *dia, *mes, *ano, *data;  
// str1 esta declarada mas nao foi inicializada  
char str1[30];  
// inicializacao de strings por aspas duplas  
// um caracter se inicializa com apostrofo/aspas simples  
char ch='A', str2[30]="@lo Mund0!";  
// inicializa de strings caracter por caracter com apostrofo/aspas simples  
// eh similar a inicializacao de outros tipos de vetores  
char str3[10]={'I', 'F', 'S', 'u', 'l', '\0'};  
int vet[30]={0,1,2,3,4,5};
```

Strings

Leitura e Escrita

- Leitura de caracter do usuário
 - Função `scanf()` - possibilita ler mais de uma entrada por vez
 - Utilizar `%c` para um caracter
 - Funções `getc()` e `getchar()` - lê um caracter por vez
 - Funções `getch()` e `getche()`
 - Exclusivas do windows (incluir biblioteca `conio.h`)

- Exemplo:

```
char str[30];
```

```
printf("Digite três caracteres");
```

```
str[0]=getchar();
```

```
str[1]=getc(stdin);
```

```
scanf("%c", &str[2]);
```

Strings

Leitura e Escrita

- Leitura de caracter do usuário
 - Bug na leitura de um caracter
 - Funções `setbuf()` ou `fflush()` **não** funcionam em todos os casos
 - Sugestão de contorno
 - Coloque um espaço no início da string do `scanf()`
- Leitura de string
 - Função `gets()` - lê uma string incluindo espaços
 - Função `scanf()` - lê uma string até encontrar um espaço ou quebra de linha (enter)
- Exemplo:

```
char str[30], ch;
printf("Digite um caracter"); // gambiarra para ler um caracter (espaco antes do porcento)
scanf(" %c", &ch);

printf("Digite uma string sem espacos");
scanf("%s", str);           // nao tem & porque a variavel str eh um vetor
printf("Digite uma string com espacos");
gets(str);
```

Strings

Leitura e Escrita

- Escrita de caracter vs string
 - Na função **printf()** utilizando as seguintes marcações na string de saída
 - **%c** para escrever uma variável do tipo char
 - **%s** para escrever um vetor de caracteres

- Exemplo:

```
char str[30], ch;  
printf("Digite um caracter");  
scanf(" %c", &ch);  
printf("Digite uma string");  
gets(str);
```

```
printf("Voce digitou o caracter: %c \n", ch);  
printf("Voce digitou a string: %s \n", str);
```

Strings

String versus character

- Character está entre apóstrofo/aspas simples
 - `'A'` - character A
 - `' '` - character espaço em branco
 - `'\n'` - character de quebra de linha (representado por dois símbolos, mas ainda é character)
 - `'%%'` - character % (representado por dois símbolos, mas ainda é character)
- String está entre aspas duplas
 - Sempre há o marcador `'\0'` ao final de qualquer string
 - `"Programacao"` - String contendo onze caracteres + `'\0'`
 - `"A"` - String contendo um caractere + `'\0'`
 - `"\n"` - String contendo um caractere + `'\0'`
- Conclusão `'A'` é diferente de `"A"`

Manipulando Strings

Manipulando Strings

Biblioteca `string.h`

- String são vetores, portanto
 - A manipulação de vetores é realizada por meio de funções
 - Toda a abordagem de ponteiros e funções vista anteriormente se aplica integralmente em strings
 - No caso particular das strings, há uma biblioteca pronta chamada **`string.h`**
- **Certo**
 - Atribuição de UM caracter na string
`str[3]='w';`
 - Duas strings iniciam no mesmo endereço de memória
`str1 = str2;`
 - O endereço inicial da string é incrementado em 1 posição
`str++;`
- **Errado**
 - Atribuição de “IFSul” à string str
`str = "IFSul"`
 - Cópia de string
`str1 = str2;`
 - Concatenação strings
`str1 = str1 + str2;`

Manipulando Strings

Biblioteca string.h

- Funções principais de string.h
- `int strlen(char *str);`
 - Retorna um inteiro com o tamanho da string `str`
- `void strcpy(char *destino, char *origem);`
 - Copia a string origem na string destino
 - String destino é sobrescrita
 - String origem não é alterada
 - String origem pode ser uma constante ("exemplo")
- `void strcat(char *destino, char *origem);`
 - Concatena strings destino e origem
 - Resultado está na string destino que é sobrescrita
 - String origem não é alterada e pode ser uma constante
- `int strcmp(char *str1, char *str2);`
 - Retorna 0 se o conteúdo de `str1` e `str2` são iguais
 - Qualquer valor diferente de 0 indica strings diferentes

```
int main () {
    char str1[30] = "Programacao", str2[20];

    printf("Tamanho de str1 = %i \n", strlen(str1));

    strcpy(str2, "Estruturada"); //str2="Programacao"

    if(strcmp(str1, str2) == 0){
        printf("strings str1 e str2 sao iguais \n");
    }else{
        printf("strings str1 e str2 sao diferentes \n");
    }

    strcat(str1, " ");
    strcat(str1, str2); //str1="Programacao Estruturada"

    printf("%s \n", str1);
    return 0;
}
```

Manipulando Strings

Biblioteca string.h

- **Funções principais de string.h**

- `int strlen(char *str);`
 - Retorna um inteiro com o tamanho da string str
- `void strcpy(char *destino, char *origem);`
 - Copia a string origem na string destino
 - String destino é sobrescrita
 - String origem não é alterada
 - String origem pode ser uma constante ("exemplo")
- `void strcat(char *destino, char *origem);`
 - Concatena strings destino e origem
 - Resultado está na string destino que é sobrescrita
 - String origem não é alterada e pode ser uma constante
- `int strcmp(char *str1, char *str2);`
 - Retorna 0 se o conteúdo de str1 e str2 são iguais
 - Qualquer valor diferente de 0 indica strings diferentes

```
int main (){
    char str1[30] = "Programacao", str2[20];

    printf("Tamanho de str1 = %i \n", strlen(str1));

    strcpy(str2, "Estruturada"); //str2="Programacao"

    if(strcmp(str1, str2) == 0){
        printf("strings str1 e str2 sao iguais \n");
    }else{
        printf("strings str1 e str2 sao diferentes \n");
    }

    strcat(str1, " ");
    strcat(str1, str2); //str1="Programacao Estruturada"

    printf("%s \n", str1);
    return 0;
}
```

Manipulando Strings

Criando suas próprias funções

- **Exemplo 1 - Letra maiúscula**

- Transforma uma letra minúscula em maiúscula
- Qualquer caracter **!=** de letra minúscula não é alterado
- Solução utiliza conceitos da **Tabela ASCII**
- Uso exige **passagem por valor**
 - Manipula apenas caracteres simples, sem string

```
ANTES da funcao: ch = x
DEPOIS da funcao: ch = X
```

```
char letraMaiuscula(char letra){
    if(letra >= 'a' && letra <= 'z'){
        letra = letra - 32;
    }
    return letra;
}

int main(){
    char ch='x';
    printf("ANTES da funcao: ch = %c\n", ch);

    ch=letraMaiuscula(ch);

    printf("DEPOIS da funcao: ch = %c\n", ch);
}
```

Manipulando Strings

Criando suas próprias funções

- **Exemplo 2 - Troca valores**

- É a mesma função vista em ponteiros
 - Usa `char` ao invés de `int`
- Uso exige **passagem por referência**
 - Ao passar `char` como parâmetro, utilize `&`

```
ANTES: ch1 = X, ch2 = Y
DEPOIS: ch1 = Y, ch2 = X
```

```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

int main(){
    char ch1='X', ch2='Y';
    printf("ANTES: ch1 = %c, ch2 = %c\n", ch1, ch2);

    trocaChar(&ch1, &ch2);

    printf("DEPOIS: ch1 = %c, ch2 = %c\n", ch1, ch2);
}
```

Manipulando Strings

Criando suas próprias funções

- **Exemplo 3 - Insere caracter na string**
 - pos é a posição da str onde ch será inserida
 - Use **passagem por valor**
 - Para passar o caracter ch como parâmetro da função `insereChar()`
 - Use **passagem por referência**
 - Para passar a string str como parâmetro da função `insereChar()`
 - Para passar o caracter ch como parâmetro de função `trocaChar()`
 - Para passar UM caracter da string str como parâmetro de função `trocaChar()`

Inserte - na pos 2 da str IFSul
Resultado: IF-Sul

```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

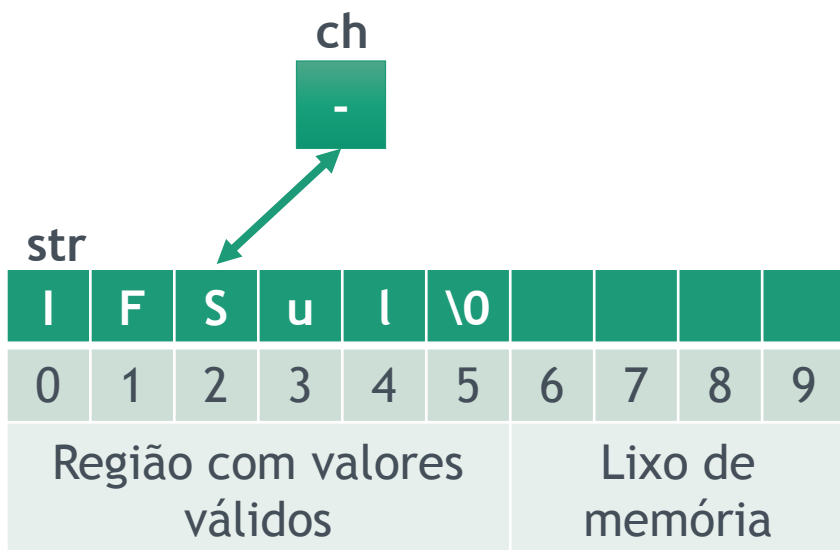
    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 3 - Insere caracter na string



```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

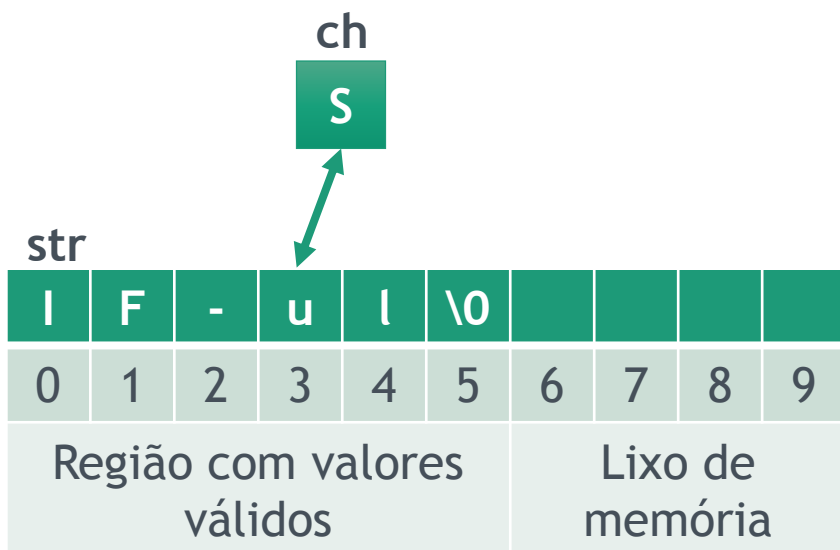
    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```


Manipulando Strings

Criando suas próprias funções

- Exemplo 3 - Insere caracter na string



```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

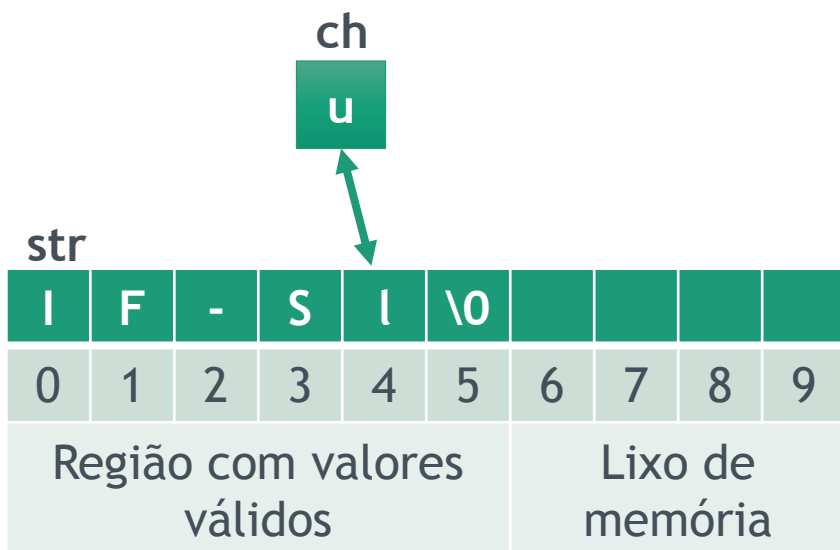
    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 3 - Insere caracter na string



```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

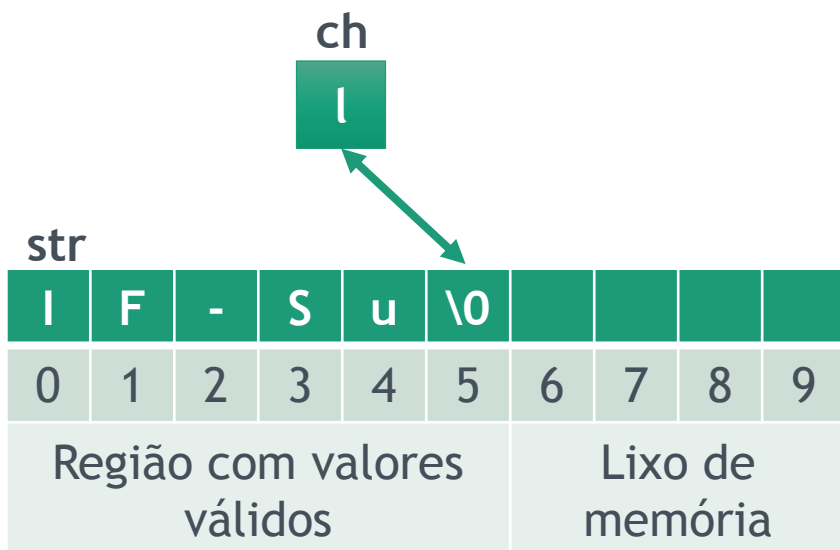
    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 3 - Insere caracter na string



```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

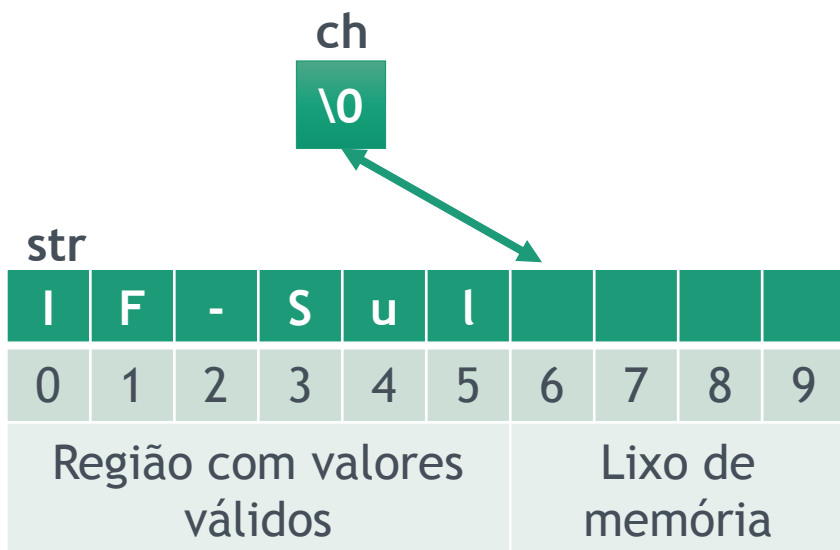
    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 3 - Insere caracter na string



```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1= *ptr2;
    *ptr2= temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

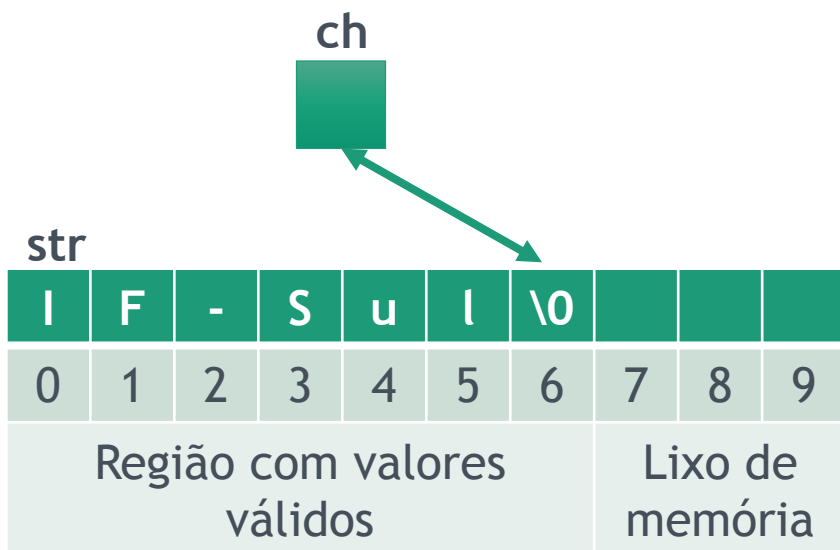
    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 3 - Insere caracter na string



```
void trocaChar(char *ptr1, char *ptr2){
    char temp;
    temp = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = temp;
}

void insereChar(char *str, char ch, int pos){
    int i;
    for(i=pos; i<=strlen(str); i++){
        trocaChar(str+i, &ch);
    }
}

int main(){
    char str[10] = "IFSul", ch='-';
    printf("Insere %c na pos 2 de %s\n", ch, str);

    insereChar(str, ch, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- **Exemplo 4 - Insere palavra na string**

- pos é a posição da str onde palavra será inserida
- Use **passagem por valor**
 - Para passar UM caracter da string palavra como parâmetro da função `insereChar()`
- Use **passagem por referência**
 - Para passar as strings str e palavra como parâmetro da função `inserePalavra()`
 - Para passar a string str como parâmetro da função `insereChar()`

```
Inserir - na pos 2 da str IFSul
Resultado: IF - Sul
```

```
void inserePalavra(char *str, char *palavra, int pos){
    int i;
    for(i=0; i<strlen(palavra); i++){
        insereChar(str, *(palavra+i), pos+i);
    }
}

int main(){
    char str[10] = "IFSul", palavra[10]=" - ";
    printf("Insere %s na pos 2 da str %s\n", palavra, str);

    inserePalavra(str, palavra, 2);

    printf("Resultado: %s\n", str);
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 4 - Insere palavra na string



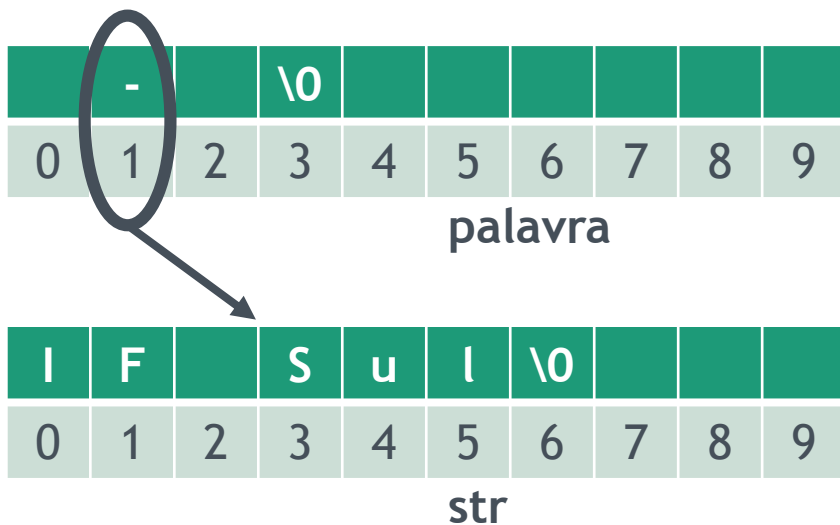
```
void inserePalavra(char *str, char *palavra, int pos){  
    int i;  
    for(i=0; i<strlen(palavra); i++){  
        insereChar(str, *(palavra+i), pos+i);  
    }  
}
```

```
int main(){  
    char str[10] = "IFSul", palavra[10]=" - ";  
    printf("Insere %s na pos 2 da str %s\n", palavra, str);  
  
    inserePalavra(str, palavra, 2);  
  
    printf("Resultado: %s\n", str);  
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 4 - Insere palavra na string



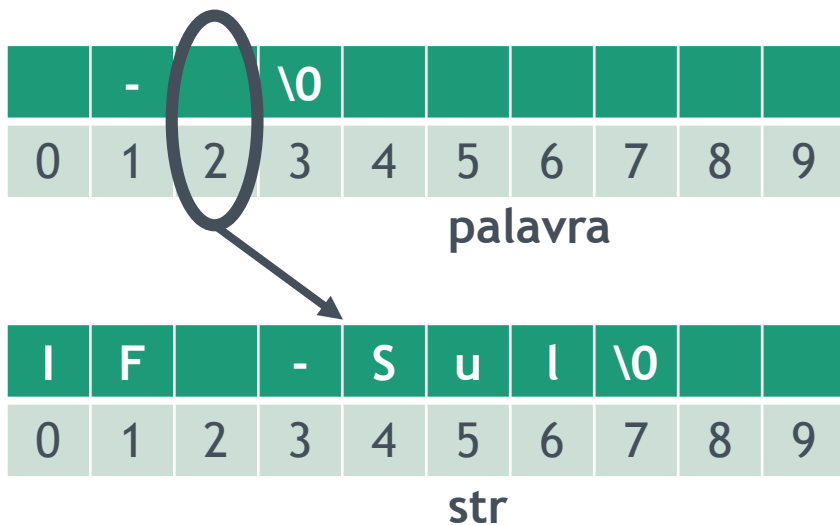
```
void inserePalavra(char *str, char *palavra, int pos){  
    int i;  
    for(i=0; i<strlen(palavra); i++){  
        insereChar(str, *(palavra+i), pos+i);  
    }  
}
```

```
int main(){  
    char str[10] = "IFSul", palavra[10]=" - ";  
    printf("Insere %s na pos 2 da str %s\n", palavra, str);  
  
    inserePalavra(str, palavra, 2);  
  
    printf("Resultado: %s\n", str);  
}
```


Manipulando Strings

Criando suas próprias funções

- Exemplo 4 - Insere palavra na string



```
void inserePalavra(char *str, char *palavra, int pos){  
    int i;  
    for(i=0; i<strlen(palavra); i++){  
        insereChar(str, *(palavra+i), pos+i);  
    }  
}
```

```
int main(){  
    char str[10] = "IFSul", palavra[10]=" - ";  
    printf("Insere %s na pos 2 da str %s\n", palavra, str);  
  
    inserePalavra(str, palavra, 2);  
  
    printf("Resultado: %s\n", str);  
}
```

Manipulando Strings

Criando suas próprias funções

- Exemplo 4 - Insere palavra na string

	-		\0						
0	1	2	3	4	5	6	7	8	9

palavra

I	F		-		S	u	l	\0	
0	1	2	3	4	5	6	7	8	9

str

```
void inserePalavra(char *str, char *palavra, int pos){  
    int i;  
    for(i=0; i<strlen(palavra); i++){  
        insereChar(str, *(palavra+i), pos+i);  
    }  
}
```

```
int main(){  
    char str[10] = "IFSul", palavra[10]=" - ";  
    printf("Insere %s na pos 2 da str %s\n", palavra, str);  
  
    inserePalavra(str, palavra, 2);  
  
    printf("Resultado: %s\n", str);  
}
```

Mais funções de strings nos códigos em anexo

MUITO
OBRIGADO

Prof. André del Mestre

www.ifsul.edu.br
almmartins@charqueadas.ifsul.edu.br