ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Em C, não existe tipo string nativo
- Strings são vetores de char
- Terminam com o caractere especial: '\0' (null terminator)

- Exemplo: `char nome[] = "IFPE";`

- Armazanamento em memória: `'I','F','P','E','\0'`

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Declaração de Strings:

```
char s1[] = "IFPE";
char s2[10] = "IFPE";
char s3[] = {'I','F','P','E','\0'};
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Declaração de Strings:

```
char s1[] = "IFPE";
char s2[10] = "IFPE";
char s3[] = {'I','F','P','E','\0'};
```

- Ponto importante:
  - Strings em C sempre terminam com o caractere nulo '\0', que também ocupa espaço no vetor.
  - Assim, em um vetor char s[10], é possível armazenar no máximo 9 caracteres úteis, reservando a última posição para o '\0'.

Strings em C

- Saída de strings:

Imprime até encontrar '\0'

```c
char nome[] = "IFPE";

printf("%s\n", nome);
```

Imprime a string e adiciona \n automaticamente

```c
char nome[] = "IFPE";
puts(nome);
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Entrada de Strings:

```
char nome[5];
scanf("%s", nome);
```

não usa &

- Limitação do scanf: lê só até o primeiro espaço

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Entrada de Strings:

```
char nome[5];
scanf("%s", nome);
```

- Limitação do scanf: lê só até o primeiro espaço

O que acontece se o usuário digitar um nome com 5 ou mais caracteres?

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Limitando o número de caracteres lidos em um scanf:

```
char nome[5];
scanf("%4s", nome);
```

- Lê no máximo 4 caracteres da entrada
- Adiciona automaticamente o '\0' ao final

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Buffer Overflow: Ocorre quando se ultrapassa o limite de uma área de memória (buffer).
- Em C, não há proteção automática contra esse problema.
- Pode causar:
  - corrupção de memória
  - comportamento inesperado
  - falhas (crash)
- Exemplo: char nome[4]; para armazenar “IFPE”;
  - Memória esperada [I][F][P][E]
  - Armazenamento real [I][F][P][E][\0]

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Buffer Overflow: Ocorre quando se ultrapassa o limite de uma área de memória (buffer).
- Em C, não há proteção automática contra esse problema.
- Pode causar:
  - corrupção de memória
  - comportamento inesperado
  - falhas (crash)
- Exemplo: char nome[4]; para armazenar “IFPE”;
  - Memória esperada [I][F][P][E]
  - Armazenamento real [I][F][P][E][\0] ✖ ultrapassa limite

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Entrada de strings com espaços e limite do tamanho da entrada:

```c
char nome[20];

fgets(nome, sizeof(nome), stdin);

printf("%s", nome);
```

- Mais seguro

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Ao usar o fgets para armazenar IFPE o \n (Enter) digitado pelo usuário faz parte da string.
- Assim, você pode ter uma quebra de linha “extra” ou comportamento inesperado em comparações.
- Solução:

```
nome[strcspn(nome, "\n")] = '\0';
```

- strcspn(string, "\n") retorna a posição do primeiro \n na string

Strings em C

INSTITUTO FEDERAL
Pernambuco

- O bug do ENTER “fantasma” é um problema clássico quando se mistura scanf com fgets:

```c
int idade;
char nome[50];

printf("Idade: ");
scanf("%d", &idade);

printf("Nome: ");
fgets(nome, sizeof(nome), stdin);

printf("Nome digitado: %s", nome);
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- O bug do ENTER “fantasma” é um problema clássico quando se mistura scanf com fgets:

```c
int idade;
char nome[50];

printf("Idade: ");
scanf("%d", &idade);

printf("Nome: ");
fgets(nome, sizeof(nome), stdin);

printf("Nome digitado: %s", nome);
```

- O programa pode pular a leitura:

```
Idade: 18
Nome: Nome digitado:
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Solução simples:

```c
int idade;
char nome[50];

printf("Idade: ");
scanf("%d", &idade);

getchar();

printf("Nome: ");
fgets(nome, sizeof(nome), stdin);

printf("Nome digitado: %s", nome);
```

limpa o buffer consumindo o \n

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A biblioteca padrão da linguagem C que fornece funções para manipulação de strings e blocos de memória é a <string.h>.

- Incluindo a biblioteca: `#include <string.h>`

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A função strlen calcula o tamanho de uma string, ou seja, a quantidade de caracteres antes do caractere nulo \0.
- Assinatura: size_t strlen(const char *str);
  - Recebe um ponteiro para char (string)
  - Retorna um valor do tipo size_t (inteiro sem sinal)

- Exemplo:

```
char nome[] = "IFPE";
strlen(nome);
printf("%zu\n", strlen(nome));
```

4

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A função strcpy copia o conteúdo da string origem para a string destino, incluindo o \0.
- Assinatura: char *strcpy(char *destino, const char *origem);
  - destino: onde será copiada a string
  - origem: string a ser copiada
  - Retorna: ponteiro para destino
  - Exemplo:

```
char origem[] = "IFPE";
char destino[10];

strcpy(destino, origem);

printf("%s\n", destino);
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A função strcat concatena (junta) duas strings.
- Assinatura: char *strcat(char *destino, const char *origem);
  - destino: string que será modificada
  - origem: string que será anexada
  - Retorna: ponteiro para destino

- Exemplo:

```c
char origem[] = "Belo Jardim";
char destino[] = "IFPE - ";

strcat(destino, origem);

printf("%s\n", destino);
```

```
IFPE - Belo Jardim
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A função strcmp é usada para comparar duas strings.
- Assinatura: int strcmp(const char *s1, const char *s2);
  - s1 e s2: strings a serem comparadas
  - Retorna um inteiro
- A comparação é lexicográfica (igual à ordem de dicionário), baseada nos valores ASCII.
- Valores de retorno:
  - 0 → strings iguais
  - < 0 → s1 é menor que s2
  - > 0 → s1 é maior que s2

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A função strcmp é usada para comparar duas strings.
- Exemplo:

```
printf("%d\n", strcmp("abc", "abc"));
printf("%d\n", strcmp("abc", "abe"));
printf("%d\n", strcmp("ba", "aa"));
```

```
0
-1
1
```

Strings em C

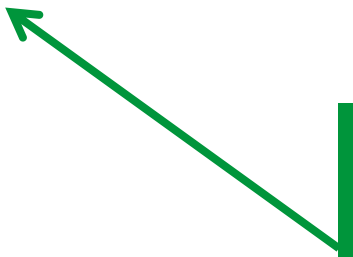
INSTITUTO FEDERAL
Pernambuco

- A função strcmp é usada para comparar duas strings.
- Exemplo:

```
printf("%d\n", strcmp("abc", "abc"));
printf("%d\n", strcmp("abc", "abe"));
printf("%d\n", strcmp("ba", "aa"));
```

```
0
-1
1
```

- Erro comum em comparação de strings: if (s1 == s2)



Isso compara endereços, não conteúdo.

Strings em C

INSTITUTO FEDERAL
Pernambuco

- A função strcmp é usada para comparar duas strings.
- Exemplo:

```
printf("%d\n", strcmp("abc", "abc"));
printf("%d\n", strcmp("abc", "abe"));
printf("%d\n", strcmp("ba", "aa"));
```

```
0
-1
1
```

- Erro comum em comparação de strings: if (s1 == s2)
- Forma correta de comparar strings: if (strcmp(s1, s2) == 0)

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Percorrendo strings:

```
for(int i = 0; nome[i] != '\0'; i++) {
    printf("%c\n", nome[i]);
}
```

```
I
F
P
E
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

- Percorrendo strings:

```c
char s[20];
fgets(s, sizeof(s), stdin);

for (int i = 0; s[i] != '\0' && s[i] != '\n'; i++) {
    printf("%c\n", s[i]);
}
```

Strings em C

INSTITUTO FEDERAL
Pernambuco

1. Crie um programa que leia uma string e um caractere e em seguida informe quantas vezes esse caractere aparece.
2. Crie um prorgama que leia uma string e crie outra sem espaços.
3. Crie um prorgama que leia um nome completo e mostre a quantidade de caracteres, o nome em maiúsculo e se contém a letra ‘a’ ou ‘A’.

Dúvidas

INSTITUTO FEDERAL
Pernambuco

ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite