Estrutura de Dados 1

Strings

Prof. Lucas Boaventura
lucas.boaventura@unb.br

Introdução

- Aprendemos a representação de dados em vetores, muito útil para diversos conceitos de programação
- Caracteres são um tipo de dados básico
- Uma palavra é composta por múltiplos caracteres, por isso é natural que se utilize vetores de caracteres para representar palavras

Strings

- Strings são caracteres adjacentes na memória do computador
- Textos e palavras são amplamente utilizados em programas de computadores: de editores de texto a mensagens de erro em programas que executam apenas em terminal

Strings

- Na linguagem de programação C, declara-se uma string como um vetor de caracteres
  - char str[10];
- Além disso, a string é composta pelas letras e sempre é terminada pelo caracter ‘\0’
  - str[0] = ‘O’;
  - str[1] = ‘i’;
  - str[2] = ‘\0’;

Strings

- Neste caso, str irá conter a palavra “Oi” e o caracter ‘\0’ terminando a string
- Também podemos inicializar as strings com a notação tradicional de vetores:
  - char str[10] = { ‘O’, ‘i’, ‘\0’ };
- No entanto, a linguagem oferece suporte com aspas duplas:
  - char str[10] = “Oi”; //O ‘\0’ é incluído!

Strings

```c
#include <stdio.h>

int main()
{
    char str1[10];

    str1[0] = 'O';
    str1[1] = 'i';
    str1[2] = '\0';

    char str2[10] = { 'O', 'i', '\0' };
    char str3[10] = "Oi";
}
```

Strings

- Cuidado! Aspas simples são utilizadas para caracteres e aspas duplas são para strings!
  - ‘A’ significa um caractere: A
  - “A” significa a strings A, um caractere ‘A’ e um ‘\0’
- O valor ‘\0’ (primeiro elemento da tabela ASCII) ao final é muito importante. Todas funções de strings baseiam-se no fato de que elas terminam com ele.

Strings

- Atenção! A atribuição com aspas duplas para string apenas pode ser utilizada na **inicialização**

```
int main()
{
    char str1[15];

    str1 = "Minha string";

    return 0;
}
```

- error: assignme

Strings

- Assim como vetores e matrizes, não se pode fazer atribuições de strings:

```
int main()
{
    char str1[15] = "Minha string";
    char str2[15];

    str2 = str1; //ERRO de compilacao
    return 0;
}
```

- error: ass ype

Strings

- Pode-se manipular elementos da string como vetores tradicionais:
  - char str[10] = “Ola”;
- str contém a string “Ola”
  - str[1] = ‘p’;
- Agora, str contém a string “Opa”

Imprimindo e Lendo

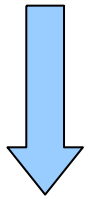
- O formato de dados “%s” pode ser utilizado nas funções:
  - printf(“%s\n”, str);
- O código imprime a string str na tela (e começa uma linha nova)

Imprimindo e Lendo

```c
int main()
{
    char str[] = "Ola, mundo!";

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

Imprimindo e Lendo

Assim como vetores, é possível omitir o tamanho da string quando ela é inicializada



```c
int main()
{
    char str[] = "Ola, mundo!";

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

Imprimindo e Lendo

- A função scanf também pode ser utilizada para ler strings do teclado:
  - scanf(“%s”, str);
- A função scanf para de ler se encontrar um retorno de linha (\n) ou se encontrar um espaço em branco!

Exercício

- Faça um programa que leia uma string do teclado e imprima a string na tela

Exercício

- Faça um programa que leia uma string do teclado e imprima a string na tela

```c
#include <stdio.h>

int main()
{
    char str[20];

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

Imprimindo e Lendo

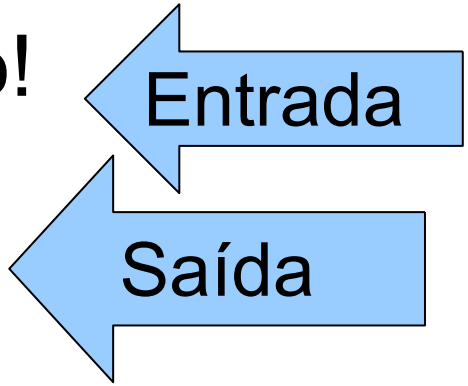
- A função scanf para quando um espaço “ ” é encontrado...
- Mas e se a string do usuário tiver um espaço?
- O restante dos dados serão lidos no próximo scanf

Imprimindo e Lendo

```c
int main()
{
    char str[20];

    scanf("%s", str);
    printf("(%s)\n", str);

    scanf("%s", str);
    printf("(%s)\n", str);

    return 0;
}
```

Execução:

ola mundo!
(ola)
(mundo!)



Usuário digitou apenas uma linha

Programa imprimiu 2 linhas

Exercício

- Faça um programa que leia uma string do teclado e imprima o último caractere dela
- **Dica**: para isso, vamos tratar as strings como vetores!

- Faça um programa que leia uma string do teclado e imprima o último caractere dela

```c
#include <stdio.h>

int main()
{
    char str[20];
    int i;

    scanf("%s", str);
    i = 0;
    while(i < 20 && str[i] != '\0')
    {
        i++;
    }

    printf("%c\n", str[i-1]);
    return 0;
}
```

Imprimindo e Lendo

- Para incluir a leitura de espaços em strings, podemos utilizar um modificador na leitura
  - scanf(“%[^\n]”,str);
- [ ] indicam o uso de uma expressão regular
- ^ significa “não” lógico
- \n é o retorno de linha
- [^\n] significa não é um retorno de linha

Imprimindo e Lendo

- Historicamente, existia a função “gets” para ler a string do teclado. Ela foi depreciada e é marcada como perigosa
- O scanf que vimos, também tem o mesmo problema...
- Escrever fora do vetor, erro out of bounds:
  - char str[10];
  - str[20] = ‘B’;

Imprimindo e Lendo

- O problema do scanf e do gets é que eles não sabem até onde podem escrever!
  - char str[10];
  - scanf(“%s”, str);
- E se o usuário digitar 20 caracteres? Um usuário mal intencionado pode causar muito problema com isso...
- Buffer overflow

Imprimindo e Lendo

- Para ler uma string com segurança, preferimos usar a função fgets
- Faz parte de algumas manipulações de funções de arquivos
- Podemos ler da entrada padrão explicitamente:
  - fgets(str, 20, stdin);

Imprimindo e Lendo

- A função não é equivalente a scanf:
- Ela coloca o ‘\n’ na string quando lê do teclado
- Ela coloca o ‘\0’ no final

Imprimindo e Lendo

- A função não é equivalente a scanf:
- Ela coloca o ‘\n’ na string quando lê do teclado
- Ela coloca o ‘\0’ no final

```
int main()
{
    char str[10];

    fgets(str, 10, stdin);
    printf("%s\n", str);
    return 0;
}
```

```
[user@station codigo]$ ./programa
Alo!
Alo!

[user@station codigo]$
```

Imprimindo e Lendo

```c
#include <stdio.h>

int main()
{
    char str[10];

    fgets(str, 10, stdin);
    printf("%s\n", str);
    fgets(str, 10, stdin);
    printf("%s\n", str);
    return 0;
}
```

- Ler uma string menor que a digitada, irá deixar dados no buffer de leitura!

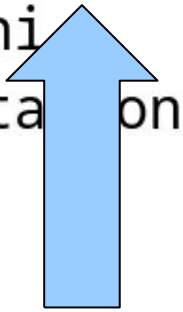
```
[user@station codigo]$ ./programa
123456789abcdefghijklmn
123456789
abcdefghi
[user@station codigo]$
```

Imprimindo e Lendo

```c
#include <stdio.h>

int main()
{
    char str[10];

    fgets(str, 10, stdin);
    printf("%s\n", str);
    fgets(str, 10, stdin);
    printf("%s\n", str);
    return 0;
}
```

- Ler uma string menor que a digitada, irá deixar dados no buffer de leitura!

```
[user@station codigo]$ ./programa
123456789abcdefghijklmn
123456789
abcdefghi
[user@sta  on codigo]$
```



Leu “apenas” 9!
\0 é o 10º

Imprimindo e Lendo

- Apesar de perigoso, na nossa matéria (e em muitas olimpíadas de programação) os dados de entradas são garantidos que possuem um tamanho limitado
- Em ambientes controlados, podemos utilizar a função

Exercícios

- Faça um programa que leia uma string do teclado e imprima o tamanho dela

Dever de Casa

- Aprender como ler uma string até receber um EOF
- (Se ainda não aprenderam)

Leitura de arquivos como entrada no Windows:

- gcc -Wall -o a.exe arquivo.c
- Get-Content .\entrada.txt -Raw | .\a.exe

Leitura de arquivos como entrada no Linux:

- gcc -Wall -o prog arquivo.c
- ./prog < entrada.txt

Dúvidas?

- lucas.boaventura@unb.br