

# Controle de Registros

Projeto parte 1

**Marcos Silvano Almeida** 

Departamento de Computação UTFPR Campo Mourão

### Projeto parte 1: Controle de Registros

#### Objetivo:

 Implementar app gerenciamento de registros de dados em memória, simulando um sistema de armazenamento simples

#### Características

- Trabalho em <u>dupla</u> OU <u>sozinho</u>.
- Interface do usuário (UI) em modo texto com menus de navegação
- Escolher um tema para os dados: filmes, livros, séries, jogos, esportes, etc
- Operações: criar, editar, obter, listar e remover registro + ferramentas
  - Confirmação em cada operação
- Dados são organizados em registros ("string montada por strings") de tamanho fixo, armazenados sequencialmente como texto em uma única string longa.
  - Implementar várias funcionalidades de manipulação de strings e vetores
- Cada registro será composto por um número fixo de campos. Exemplo:
  - REGISTRO DE FILME: título + ano + diretor



### Projeto parte 1: Controle de Registros

Considerando o exemplo anterior:

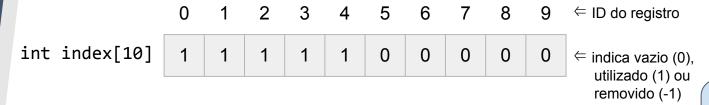
• Exemplo de registro de filme (posições vazias são preenchidas com '\0')
"Big Trouble Little China\0\0\0\0\0\0\0\0\0\0\0\0\0\0'

Para simplificar a visualização, representaremos '\0' com 0 nos slides: "Big Trouble Little China000000019860John Carpenter0000000"



#### Estrutura Geral

O vetor **index** funciona como uma tabela. A posição é o ID do registro. O conteúdo, indica se o registro em **data** está utilizado ou livre



O vetor de chars **data** está representado em linhas para facilitar a visualização. Na prática, os registros ficam um após o outro.

char data[570]

 $10 \times 57 = 570$ 

Tamanho de **data**: tamanho do registro X

tamanho de index



#### Estrutura Geral

- O programa utiliza dois vetores:
  - char data[n x tamanho\_registro]: vetor de caracteres para armazenar registros de texto. Cada posição pode armazenar 1 char.
    - Cada registro é formado por uma <u>sequência de caracteres</u> de tamanho fixo
    - Cada registro possui um <u>número fixo de campos</u>
  - int index[n]: vetor que agirá como tabela de registros, mapeamento posições livres (0) e ocupadas (1) em data.
    - 0 posição livre: nunca utilizada anteriormente
    - 1 posição ocupada: existe um registro nesta posição
    - -1 registro na lixeira: um registro foi removido nesta posição.

Poderá ser restaurado caso não tenha sido sobrescrito.

**OBS:** Durante a adição de um novo registro, o programa deve tentar encontrar posições livres nunca usadas. Caso não haja, utilizará a primeira posição marcada com -1 (lixeira).



Protótipo de Interface

+

Simulação de Funcionamento



### Menu Principal

\_\_\_\_\_

GERENCIAMENTO DE FILMES

\_\_\_\_\_

#### MENU PRINCIPAL

- 1 Adicionar filme
- 2 Consultar filmes
- 3 Remover filme
- 4 Alterar filme

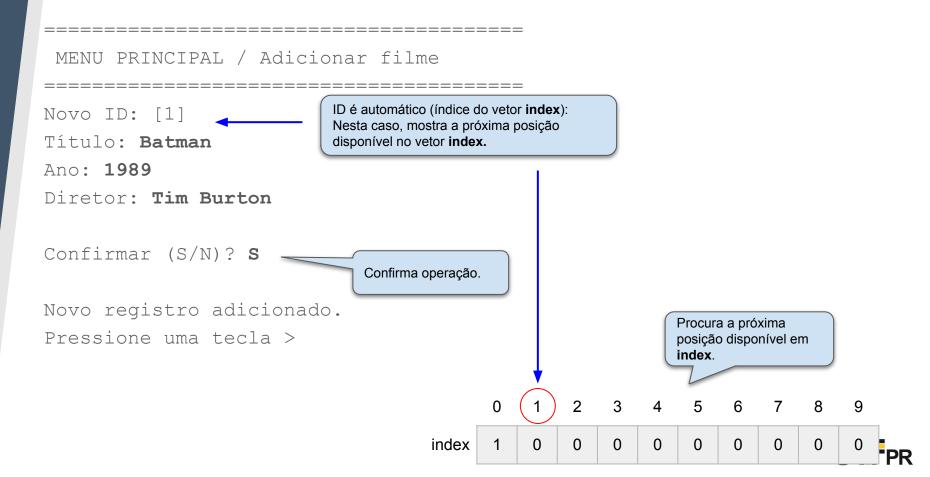
Entre com uma opção: 1

- 5 Lixeira & Restaurar
- 6 Mapa de registros
- 7 Desfragmentar Dados
- 0 Sair

Validar opção (garantir que é válida)

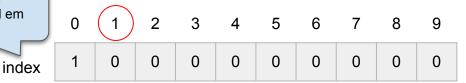


### 1 - Adicionar filme



#### 1 - Adicionar filme: como funciona

Procura a próxima posição disponível em **index**.



Concatenar as strings dos campos em um único vetor de chars de 57 posições

Novo registro a ser armazenado:

char registro[57] ⇒ Batman000000000000000000000019890Tim Burton00000000000

Copia o vetor de char **registro** para a posição correta em **data** 

data



### 2 - Consultar filme

```
MENU PRINCIPAL / Consultar filme
                                                       A busca poderá ser aplicada ao
                                                        registro inteiro, visto que ele é um
                                                        vetor de chars.
Busca: man
Filmes encontrados com 'man':
         Título
                                                      Diretor
ID
                                            Ano
01
         Batman
                                            1989
                                                      Tim Burton
03
         Watchman
                                             2009
                                                       Zack Snyder
```

Pressione uma tecla >



### 2 - Consultar todos os filmes

Para listar todos os registros, defina algum caractere especial, como asterisco \*. Uma alternativa, é adicionar a opção "Listar Todos" ao menu principal.

MENU PRINCIPAL / Consultine

\_\_\_\_\_

Busca: \*

Listando todos os registros:

ID	Título	Ano	Diretor
00	They Live	1988	John Carpenter
01	Batman	1989	Tim Burton
02	Aliens	1986	James Cameron
03	Watchman	2009	Zack Snyder
04	Big Trouble Little China	1986	John Carpenter

Pressione uma tecla >



#### 3 - Remover filme

O processo para remover um filme existente utiliza o processo de:

- Consultar filme

index

MENU PRINCIPAL / Editar filme

\_\_\_\_\_

Busca: alien

Filmes encontrados com 'alien':

ID Título Ano Diretor

02 Aliens 1986 James Camperon

ID a remover: 2

Confirmar (S/N)? **S**Registro na lixeira.

A posição removida em **index** é marcada como "lixeira" (-1). Em **data**, os dados do registro são mantidos, caso seja necessário restaurá-lo.

5

0

9

0

0

data

#### 4 - Alterar filme

\_\_\_\_\_

MENU PRINCIPAL / Editar filme

\_\_\_\_\_

Busca: man

Filmes encontrados com 'man':

ID Título Ano Diretor

01 Batman 1989 Tim Burton

03 Watchman 2009 Zack Snyder

O processo para alterar um filme existente utiliza os processo de:

- Consultar filme
- Remover filme (antigo)
- Adicionar filme (novo)

Informe o ID: 1

Título anterior: Batman => Novo: Batman 2

Ano anterior: **1989** => Novo: **1992** 

Diretor anterior: Tim Burton => Novo: Tim Burton

Confirmar (S/N)? S

Registro anterior removido. Novo registro adicionado.



### 5 - Lixeira & Restaurar

```
MENU PRINCIPAL / Lixeira & Restaurar
Filmes encontrados na lixeira:
                                                            Exibe os registros marcados com -1.
                                                            Caso restaurado, retorna para 1.
ID
         Título
                       Ano
                                  Diretor
02
         Aliens
                       1986
                                 James Cameron
0.3
         Watchmen
                       1986
                                  Zack Snider o
                                                                     5
                                                                                     9
                                        index
                                                         -1
                                                                     0
                                                                             0
                                                                                 0
                                                                                     0
Selecione o ID a restaurar: 2
```

Confirmar (S/N)? **S**Registro restaurado.

data

### 6 - Mapa de Registros

```
MENU PRINCIPAL / Mapa de Registros

-----
Vetor INDEX: 4 / 10 índices

[1, 1, -1, 1, 1, 0, 0, 0, 0, 0]
```

Exibir as duas estruturas de dados do programa: **index** e **data** 

Vetor DATA: 570 caracteres



# 7 - Desfragmentar Dados

Desfragmentar:

Mover as posições ocupadas para a esquerda, sobre as posições livres (em **index** e **data**).

```
MENU PRINCIPAL / Desfragmentar Dados
                Vamos considerar um
Vetor INDEX: 4 / 10 indices
                armazenamento com
                mais "buracos"
[1, 1, -1, 1, 0, 0, 0, 1, 0, 0]
Vetor DATA: 570 caracteres
Big Trouble Little China00000019860John Carpenter0000000
```



# 7 - Desfragmentar Dados (continuação)

Confirmar (S/N)? **S** 

```
Desfragmentação em progresso... concluída.
                            Todas as posições ocupadas
                            foram movidas para a esquerda
Vetor INDEX: 4 / 10 indices
                            em index e data. As posições
                            livres estão agora todas à direita.
[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
Vetor DATA: 570 caracteres
They Live000000000000000000000019880John Carpenter0000000
Big Trouble Little China00000019860John Carpenter0000000
```



Trabalhando com Strings



### Operações com string

- Vários exercícios e exemplos são importantes para a realização do projeto:
  - int stringLength(char str[]): calcula tamanho da string
  - int stringEquals(char str1[], char str2[]): verifica igualdade de strings
  - o int stringContains(char dest[], char src[]): verifica se encontra src está em dest
  - void stringToUpper(char str[]): converte todas as letras para maiúsculas
  - void stringTrim(char str[]): remove espaços antes e depois da string
  - void stringConcat(char dest[], char src[]): acrescenta string src ao final de dest
  - Outras...
- OBS: <u>não é permitido</u> utilizar a biblioteca string.h ou outra biblioteca que efetue operações sobre strings (exceto printf()).



# Lendo strings com scanf()

- Existem várias funções na biblioteca de C padrão que podem ser usadas para leitura de strings
  - scanf(), getLine(), fgets() e gets() (depreciada)
  - Scanf() da <stdio.h> é bastante versátil, pois permite <u>expressões</u> de leitura.
    - Podemos filtrar quais caracteres devem ser lidos
    - Podemos limitar a quantidade caracteres a serem lidos
- Alguns problemas ao ler texto (strings):
  - Digitar um texto mais longo que a string que o receberá
  - Por padrão, scanf termina ao encontrar o primeiro espaço ou quebra de linha (\n)
    - O restante (incluindo o \n), fica no buffer de entrada do teclado e será lido automaticamente no próximo scanf(), inutilizando os próximos scanf()
- Solução:
  - Usar expressão que limita a quantidade de caracteres lidos no scanf()
  - Limpar o <u>buffer de entrada</u> após cada scanf()



# Lendo strings com scanf()

 Solução: Usar expressão que limita a quantidade de caracteres lidos e sempre limpar o <u>buffer de entrada</u> após cada scanf().

```
void clearBuffer() {
  scanf("%*[^\n]"); // faz a leitura de vários caracteres (*) e encerra no \n ([^\n])
  scanf("%*c"); // lê o próximo caractere: neste caso, o '\n'
char s[10];
scanf(" %9[^\n]", s); // lê 9 caracteres do teclado e concatena \0 ao final
clearBuffer();  // limpa o restante do buffer de entrada, se houver
int num;
scanf(" %d", &num);
clearBuffer();  // limpa o restante do buffer de entrada, se houver
```



Trabalhando com cores no terminal



#### Biblioteca "screen.h"

- Biblioteca "screen.h" permitirá criar uma interface mais interessante
- Para utilizar a biblioteca deve-se:
  - Incluir os arquivos na pasta de seu projeto: screen.h + screen.c
  - Incluir a biblioteca em seu codigo: #include "screen.h"

```
void main() {
    setColor(COLOR FG BLUE); // cor AZUL para caracteres
   printf("\nUSANDO CORES NO TERMINAL\n");
    setColor(COLOR BG WHITE); // cor BRANCO para fundo
    setColor(COLOR FG BLACK); // cor PRETO para caracteres
   printf("OBS: nem todos os terminais oferecem suporte\n\n");
    resetColor();
```

A definição de cores funciona na maioria dos terminais para **Linux**, além de terminais web, como **repl.it** 



#### Biblioteca "screen.h"

Funções importantes da biblioteca screen.h

```
// define a posição do cursos na tela
void setCursor(int row, int col);
// restaura configuração de cores do terminal
void resetColor();
// define cor no terminal
// FG = foreground = cor do caractere
     BG = background = cor do fundo do caractere
void setColor(int color);
// limpa a tela com a cor de fundo atual
void clearScreen();
// sorteira cor para fundo (BG) ou caracatere (FG)
void randomBGColor();
void randomFGColor();
```



Avaliação do Projeto



### Critérios de Avaliação do Projeto

- Qualidade/eficiência das soluções utilizadas
  - Algoritmos implementados
- Organização do código
  - Legibilidade do código (identação, identificadores, ...)
  - Documentação do código (comentários)
- Estrutura do código
  - Modularização em funções para organização e evitar duplicações
- Acabamento da interface em texto
- Divisão das atividades na dupla
  - Divisão balanceada
- Conhecimento sobre o programa implementado
  - Apresentação síncrona com questionamentos do professor
  - Cada aluno é responsável por sua parte



### Questões sobre Avaliação

- Cópias: Qualquer tipo de cópia (trabalhos de colegas, internet, etc) anulará imediatamente o trabalho
  - Seja por porções de código ou pelo trabalho completo
  - Projeto deve ser de autoria exclusiva dos integrantes da equipe
  - Cada aluno deve necessariamente conhecer o seu código
    - Estruturas utilizadas
    - Algoritmos implementados

#### Entrega:

- ZIP com código fonte pelo Moodle
- Apresentação síncrona + Responder questionamentos do professor

