

# Arquivos

Estruturas de Dados II – Vanessa Braganholo

# Entidades

---

- ▶ Aplicações precisam armazenar dados sobre as mais diversas **entidades**, que podem ser concretas ou abstratas
  - ▶ Funcionário de uma empresa (concreto)
  - ▶ Carros de uma locadora de veículos (concreto)
  - ▶ Contas-corrente dos clientes de um banco (abstrato)
  - ▶ Ligações telefônicas dos clientes de uma empresa de telefonia (abstrato)



# Atributos

---

- ▶ Cada uma dessas entidades pode ser descrita por um conjunto de **atributos**
  - ▶ Funcionário: nome, CPF, data-nascimento, salário
  - ▶ Carro: marca, modelo, ano-fabricação, placa
  - ▶ Conta-Corrente: agência, conta, saldo
  - ▶ Ligações Telefônicas: data, origem, destino, duração
- ▶ Os atributos também podem ser chamados de **campos**



# Registros

---

- ▶ Indivíduos dessas entidades possuem um valor para cada um desses atributos (chamados de **pares atributo-valor**)
- ▶ Um conjunto de pares atributo-valor que identifica um indivíduo de uma entidade é chamado de **registro**



# Exemplos de Registros

---

- ▶ Funcionário:

<nome, João>, <CPF, 012345678-90>, <data-nascimento, 10/04/1980>, <salário, 3000>

- ▶ Carro

<marca, Honda>, <modelo, Fit>, <ano-fabricação, 2010>, <placa, XYZ0123>

- ▶ Conta-Corrente

<agencia, 0123>, <conta, 123456>, <saldo, 2000>

- ▶ Ligação Telefônica

<data, 01/07/2010>, <origem, 21-2598-3311>, <destino, 21-2589-3322>, <duração, 10'36">



# Tabela

---

- ▶ Uma **tabela** é um conjunto ordenado de registros. Uma tabela pode ser armazenada em memória principal ou em memória secundária (disco)
- ▶ Nesse segundo caso, também costuma ser chamada de **arquivo**



# Exemplo: Arquivo de Funcionários

---

Nome	CPF	Data-Nascimento	Salário
João	012345678-90	10/04/1980	3000
Maria	234567890-12	25/07/1978	5000
Lúcia	345678901-23	27/04/1981	1500

**IMPORTANTE:** Todos os registros de uma mesma tabela possuem a mesma estrutura (mesmo conjunto de atributos/campos)

---



# Problema: encontrar registros

---

- ▶ Problema comum de diversas aplicações: encontrar um ou mais registros em uma tabela
  - ▶ Encontrar o empregado de nome Maria
  - ▶ Encontrar todos os empregados que ganham 3000
  - ▶ Encontrar todos os empregados que nasceram em 27/04/1981



# Conceito de Chave

---

- ▶ Dados usados para encontrar um registro: chave
- ▶ Chave: subconjunto de atributos que identifica um determinado registro



# Chave Primária e Secundária

---

- ▶ **Chave primária:** subconjunto de atributos que identifica unicamente um determinado registro. Exemplo: CPF do funcionário ou RG do funcionário
- ▶ Na hipótese de uma chave primária ser formada por uma combinação de campos, essa combinação deve ser mínima (não deve conter campos supérfluos)
- ▶ Eventualmente, podemos encontrar mais de uma combinação mínima de campos que forma uma chave primária
- ▶ **Chave secundária:** subconjunto de atributos que identificam um conjunto de registros de uma tabela.  
Exemplo: Nome do funcionário



# Tabelas

---

- ▶ Aplicações reais lidam com várias tabelas, cada uma delas representando uma entidade
- ▶ **Uma aplicação de controle bancário precisaria de quais tabelas?**



# Aplicação Bancária

---

- ▶ **Uma aplicação de controle bancário precisaria de quais tabelas?**
  - ▶ Cliente
  - ▶ Agência
  - ▶ Conta-Corrente



# Certa redundância é necessária

---

- ▶ Neste caso, é necessário correlacionar os dados, para que seja possível saber que conta pertence a que agência, e que conta pertence a que cliente
- ▶ Para isso, é usual repetir a chave primária da tabela referenciada no outro arquivo
  - ▶ Cliente: CodCliente, Nome, CPF, Endereço
  - ▶ Agência: CodAgencia, NumeroAgencia, Endereco
  - ▶ Conta-Corrente: **CodAgencia, CodCliente**, CodConta, NumeroConta, Saldo



# Certa redundância é necessária

---

- ▶ Neste caso, é necessário correlacionar os dados, para que seja possível saber que conta pertence a que agência, e que conta pertence a que cliente
- ▶ Para isso, é usual repetir a chave primária da tabela referenciada no outro arquivo
  - ▶ Cliente: CodCliente, Nome, CPF, Endereço
  - ▶ Agência: CodAgencia, NumeroAgencia, Endereco
  - ▶ Conta-Corrente: CodAgencia, CodCliente, CodConta, Numero Conta, Saldo

**Quais são as chaves primárias e secundárias deste exemplo?**



# Aplicação Financeira

---

- ▶ Chaves primárias:
  - ▶ Cliente: **CodCliente**
  - ▶ Agência: **CodAgencia**
  - ▶ Conta-Corrente: **CodAgencia** E **CodConta**
- ▶ Chaves primárias alternativas:
  - ▶ Cliente: **CPF**
  - ▶ Agência: **NumeroAgencia**
  - ▶ Conta-Corrente: **NumeroConta**
- ▶ Chaves secundárias:
  - ▶ Cliente: **Nome**
  - ▶ Agência: **Endereço**
  - ▶ Conta-Corrente: **CodAgencia** OU **CodCliente** OU **Saldo**



# Discussão sobre chaves

---

- ▶ Por quê não usar CPF como chave primária de cliente?  
Por quê os atributos artificiais (código)?



# Exercício

---

Deseja-se automatizar uma locadora de automóveis. A locadora possui filiais espalhadas por todo país. Cada filial possui um código que a identifica, um telefone e um endereço. Cada filial da locadora sedia um conjunto de veículos que ela aluga. O veículo é identificado por um número sequencial que o distingue dos demais veículos da filial. Para o veículo é importante saber a placa, data de vencimento do seguro, nome do modelo, número de portas e se possui ar-condicionado ou não. Quando um veículo é alugado é fechado um contrato de aluguel. Cada contrato possui um número identificador, uma data de saída do veículo, uma data de retorno provável, para veículos ainda não retornados, e uma data de retorno efetivo, para veículos já retornados. O contrato é feito para um veículo e um cliente. Para os clientes é preciso armazenar seu nome, CPF, endereço, o telefone, bem como o número e data de expiração de seu cartão de crédito.



# Exercício

---

- ▶ Para o cenário das locadoras, identificar:
  - ▶ Entidades
  - ▶ Atributos
  - ▶ Chaves primárias



# Discussão

---

- ▶ Por que não usar uma única tabela?



# Banco de Dados

---

- ▶ Esse conjunto de arquivos pode ser considerado um banco de Dados?



# Características de um Sistema de Gerência de Banco de Dados

---

- ▶ **Natureza auto-descritiva** do sistema de banco de dados: banco de dados possui um catálogo onde estão descritas as estruturas e tipos de dados de cada tabela e suas restrições – ex. quais são as chaves primárias de cada tabela
- ▶ **Isolamento entre os programas e os dados, e a abstração dos dados**: em programação com arquivos a estrutura dos arquivos é embutida dentro das aplicações. Isso não acontece quando se usa banco de dados.
- ▶ **Suporte para as múltiplas visões dos dados**: usuários diferentes podem ver porções diferentes dos dados
- ▶ **Compartilhamento de dados e processamento de transações multi-usuários**



# Características de um Sistema de Gerência de Banco de Dados

---

- ▶ **Natureza auto-descritiva** do sistema de banco de dados: banco de dados possui um catálogo onde estão descritas as estruturas e tipos de dados de cada tabela e suas restrições – ex. quais são as chaves primárias de cada tabela
- ▶ **Isolamento entre os programas e os dados, e a abstração dos dados**: em programação com arquivos a estrutura dos arquivos é embutida dentro das aplicações. Isso não acontece quando se usa banco de dados.
- ▶ **Suporte para as múltiplas visões dos dados**: usuários diferentes podem ver porções diferentes dos dados
- ▶ **Compartilhamento de dados e processamento de transações multi-usuários**

**NÃO É O NOSSO FOCO  
NESSA DISCIPLINA!**

# Níveis de Organização das tabelas/arquivos

---

- ▶ **Organização Lógica** dos dados: é a visão que o usuário tem dos dados, com base em entidades, seus atributos e seus relacionamentos
- ▶ **Organização Física** dos dados: é a maneira pela qual as informações ficam armazenadas nos dispositivos periféricos (disco, pen-drive, etc.)
  - ▶ Aprenderemos várias alternativas diferentes nessa disciplina



# Operações sobre arquivos

---

- ▶ Programas que lidam com arquivos realizam os seguintes tipos de operações sobre eles:
  - ▶ Criação: alocação e inicialização da área de dados, assim como de seus descritores
  - ▶ Destruição: liberação da área de dados e descritores usados na representação da tabela
  - ▶ Inserção: inclusão de novo registro na tabela
  - ▶ Exclusão: remoção de um registro da tabela
  - ▶ Alteração: modificação dos valores de um ou mais atributos/campos da tabela
  - ▶ Consulta: obtenção dos valores de todos os campos de um registro, dada uma chave de entrada



# Manipulação de Arquivos em C

Fonte:  
Schildt, H. C Completo e Total. Ed. McGraw-Hill

# IDE C: NetBeans

## Download o NetBeans IDE 8.2

8.1 | 8.2 | Desenvolvimento | Arquivo

Endereço de email (opcional):

Inscrever-se na newsletter:

Mensal  Semanal

Permito me contatar neste email

Idioma do IDE:

Português (Brasil) 

Plataforma:

Mac OS X



Nota: Tecnologias em cinza não são suportadas para esta plataforma.

### Distribuições para baixar do NetBeans IDE

Tecnologias suportadas *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	Tudo
① SDK da plataforma NetBeans	●	●				●
① Java SE	●	●				●
① Java FX	●	●				●
① Java EE		●				●
① Java ME						—
① HTML5/JavaScript		●	●	●		●
① PHP			●	●		●
① C/C++					●	●
① Groovy						●
① Java Card(tm) 3 Connected						—
Servidores embutidos						
① GlassFish Server Open Source Edition 4.1.1		●				●
① Apache Tomcat 8.0.27		●				●

**Download** **Download** **Download** **Download** **Download** **Download**

116 MB livre(s) 242 MB livre(s) 142 MB livre(s) 142 MB livre(s) 147 MB livre(s) 277 MB livre(s)

# IDE C: observação importante

---

- ▶ O esqueleto de código que será fornecido para as implementações que vocês farão ao longo da disciplina também funcionam com o compilador **GCC** via linha de comando em qualquer plataforma
- ▶ Usar o makefile que será fornecido junto com o esqueleto do código



# Stream

---

- ▶ O sistema de arquivos de C é projetado para trabalhar com uma ampla variedade de dispositivos, incluindo terminais, acionadores de disco e de fita, impressoras, etc.
- ▶ Cada um desses dispositivos é muito diferente, mas o sistema de arquivos com buffer os transforma em um **dispositivo lógico chamado stream**
- ▶ Todos os streams se comportam de forma semelhante, portanto pode-se usar a mesma função para escrever num arquivo em disco e no console, por exemplo



# Tipos de Stream

---

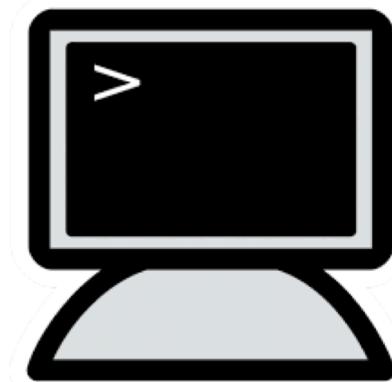
- ▶ Stream de Texto
  - ▶ Sequência de caracteres
- ▶ Stream Binário
  - ▶ Sequência de bytes com uma correspondência 1:1 com aqueles encontrados no dispositivo externo – não ocorre nenhuma tradução de caracteres
  - ▶ O número de bytes lido é sempre o mesmo encontrado no dispositivo externo



# Arquivos

---

- ▶ Em C, um arquivo pode ser qualquer coisa, desde um arquivo em disco até um terminal ou uma impressora



# Abertura de Arquivo

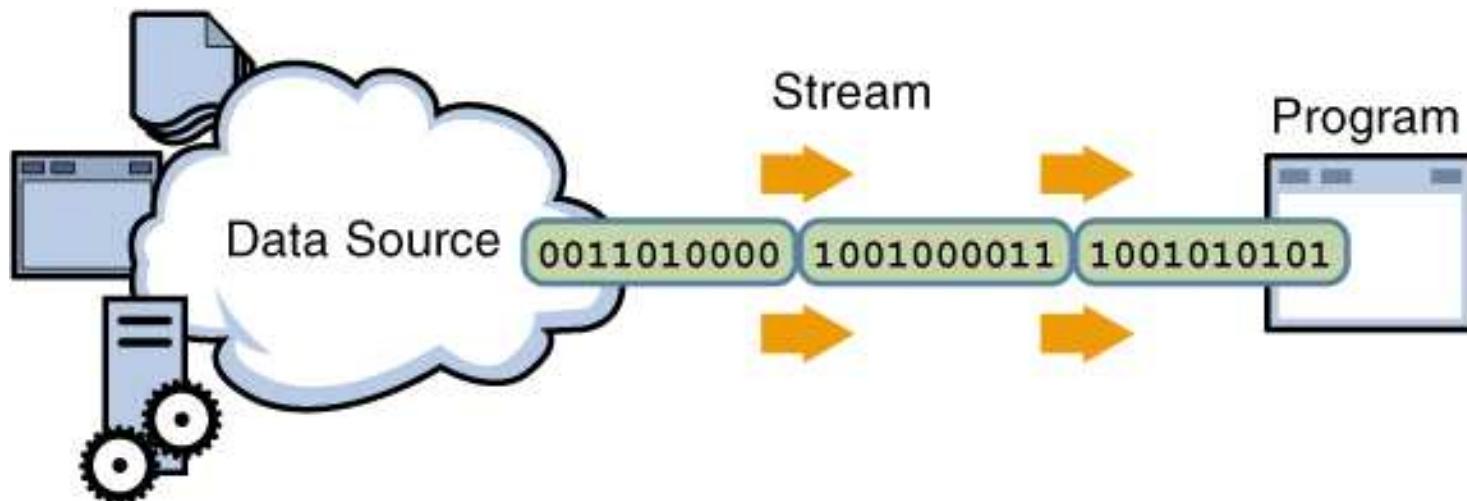
---

- ▶ Deve-se associar um stream a um arquivo e realizar uma operação de **abertura**
- ▶ Após a abertura, informações podem ser trocadas entre o dispositivo e o seu programa



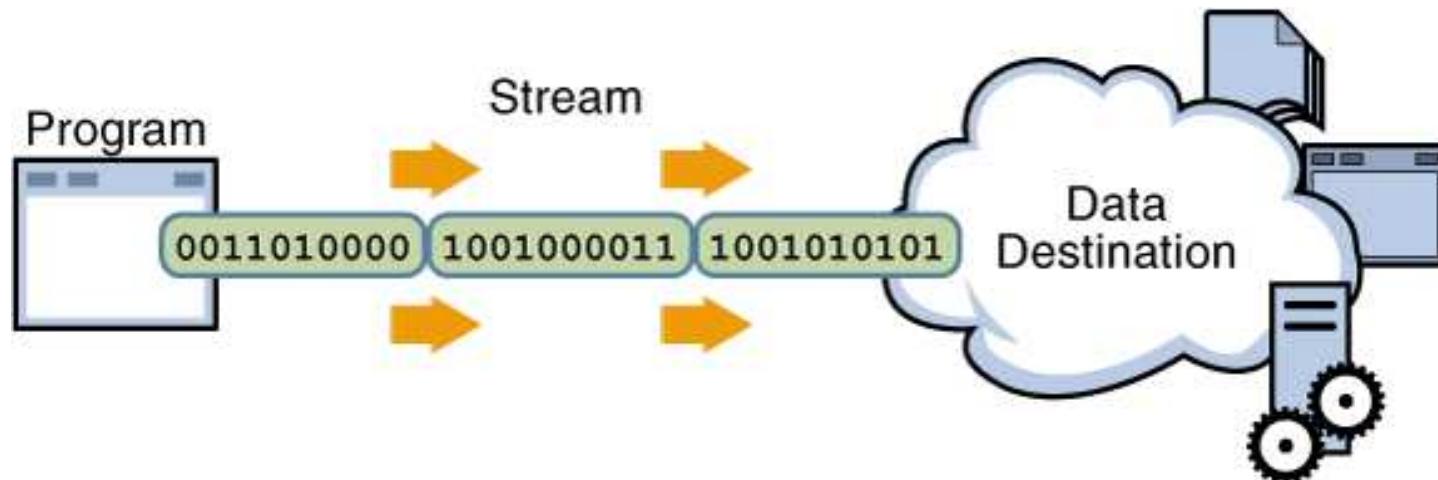
# Leitura de Arquivo

- Um programa utiliza um **Arquivo** associado a um **Stream** para ler dados de um dispositivo, um item de cada vez



# Escrita em Arquivo

- Um programa utiliza um **Arquivo** associado a um **Stream** para escrever dados em um dispositivo, um item de cada vez



Fonte da Figura: <http://download.oracle.com/javase/tutorial/essential/io/>

# Arquivos

---

- ▶ Nem todos os arquivos apresentam os mesmos recursos
- ▶ Exemplo: um arquivo em disco pode suportar acesso aleatório, enquanto um teclado não pode
- ▶ Ponto importante: todos os streams são iguais, mas nem todos os arquivos são iguais
- ▶ Se o arquivo suporta acesso aleatório, **abrir esse arquivo inicializa o indicador de posição (cursor) no arquivo**
- ▶ Quando cada caractere é lido ou escrito no arquivo, o **cursor é incrementado**



# Fechamento de Arquivo

---

- ▶ A operação de fechamento de arquivo **desassocia o arquivo do stream**
- ▶ Se um arquivo aberto para escrita for fechado, o conteúdo de seu stream associado é escrito no arquivo para evitar perda de conteúdo

BUFFER



# File

---

- ▶ Em C, cada stream associado a um arquivo tem uma estrutura de controle de arquivo do tipo **FILE**
- ▶ Essa estrutura é definida no cabeçalho **STDIO.H**, que deve ser incluído em todos os programas que manipulam arquivos



# Na nossa disciplina

---



Nosso foco na  
disciplina:

Arquivo será  
**SEMPRE**  
arquivo em  
disco



# Funções mais comuns

Nome	Função
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
fputc()	Escreve um caractere em um arquivo
fgetc()	Lê um caractere de um arquivo
fseek()	Posiciona o cursor em um byte específico
fprintf()	É para um arquivo o que printf() é para o console
fscanf()	É para um arquivo o que scanf() é para o console
feof()	Devolve <b>verdadeiro</b> se o fim do arquivo for atingido
ferror()	Devolve <b>verdadeiro</b> se ocorreu erro
rewind()	Posiciona o cursor no início do arquivo
remove()	Apaga um arquivo
fflush()	Descarrega um arquivo



# Exemplo

---

```
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char** argv) {
    //declara ponteiro para arquivo
    FILE *in;
    //abre arquivo
    if ((in = fopen("teste.dat", "rb")) == NULL) {
        printf("Erro ao abrir arquivo\n");
        exit(1);
    }
    //processa arquivo
    //...

    //fecha arquivo
    fclose(in);
}
```



# Exemplo

```
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char** argv) {
    //declara ponteiro para arquivo
    FILE *in;
    //abre arquivo
    if ((in = fopen("teste.dat", "rb")) == NULL) {
        printf("Erro ao abrir arquivo\n");
        exit(1);
    }
    //processa arquivo
    //...
    //fecha arquivo
    fclose(in);
}
```

Nome do Arquivo

Modo de Abertura  
(rb = leitura de arquivo  
binário)

# Exemplo

```
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char** argv) {
    //declara ponteiro para arquivo
    FILE *in;
    //abre arquivo
    if ((in = fopen("teste.dat", "rb")) == NULL) {
        printf("Erro ao abrir arquivo\n");
        exit(1);
    }
    //processa arquivo
    //...

    //fecha arquivo
    fclose(in);
}
```

Se houver erro na abertura, **in** ficará com valor **NULL**

# Modos de Abertura de Arquivos

Modo	Significado
r	Abre um arquivo <b>texto</b> para leitura (se não existir, retorna NULL)
w	Abre um arquivo <b>texto</b> para escrita (se já existir, conteúdo é apagado, se não existir, será criado)
a	Abre (se já existir) ou cria um arquivo <b>texto</b> para escrita, preservando o conteúdo já existente
rb	Abre um arquivo <b>binário</b> para leitura (se arquivo não existe, retorna NULL)
wb	Abre um arquivo <b>binário</b> para escrita (se arquivo já existe, conteúdo é apagado, se arquivo não existe, será criado)
ab	Abre um arquivo <b>binário</b> para escrita, preservando o conteúdo já existente (se arquivo não existe, será criado)
r+	Abre um arquivo <b>texto</b> para leitura e escrita (se arquivo não existe, retorna NULL)
w+	Cria e abre um arquivo <b>texto</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
a+	Abre (se já existir) ou cria um arquivo <b>texto</b> para leitura e escrita – cursor é posicionado no final do arquivo
rb+	Abre um arquivo <b>binário</b> para leitura e escrita (se arquivo não existe, retorna NULL)
wb+	Cria um arquivo <b>binário</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
ab+	Abre (se já existir) um arquivo <b>binário</b> para leitura e escrita, preservando o conteúdo já existente (escreve sempre no final do arquivo – append). Se não existir, cria arquivo.



# Modos de Abertura de Arquivos

Modo	Significado
r	Abre um arquivo <b>texto</b> para leitura (se não existir, retorna NULL)
w	Abre um arquivo <b>texto</b> para escrita (se já existir, conteúdo é apagado, se não existir, será criado)
a	Abre (se já existir) ou cria um arquivo <b>texto</b> para escrita, preservando o conteúdo já existente
rb	Abre um arquivo <b>binário</b> para leitura (se arquivo não existe, retorna NULL)
wb	Abre um arquivo <b>binário</b> para escrita (se arquivo já existe, conteúdo é apagado, se arquivo não existe, será criado)
ab	Abre um arquivo <b>binário</b> para escrita, preservando o conteúdo já existente (se arquivo não existe, será criado)
r+	Abre um arquivo <b>texto</b> para leitura e escrita (se arquivo não existe, retorna NULL)
w+	Cria e abre um arquivo <b>texto</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
a+	Abre (se já existir) ou cria um arquivo <b>texto</b> para leitura e escrita, preservando o conteúdo já existente
rb+	Abre um arquivo <b>binário</b> para leitura e escrita (se arquivo não existe, retorna NULL)
wb+	Cria um arquivo <b>binário</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
ab+	Abre (se já existir) um arquivo <b>binário</b> para leitura e escrita, preservando o conteúdo já existente (escreve sempre no final do arquivo – append). Se não existir, cria arquivo.

Arquivo Texto

# Modos de Abertura de Arquivos

Modo	Significado
r	Abre um arquivo <b>texto</b> para leitura (se não existir, retorna NULL)
w	Abre um arquivo <b>texto</b> para escrita (se já existir, conteúdo é apagado, se não existir, será criado)
a	Abre (se já existir) ou cria um arquivo <b>texto</b> para escrita, preservando o conteúdo já existente
rb	Abre um arquivo <b>binário</b> para leitura (se arquivo não existe, retorna NULL)
wb	Abre um arquivo <b>binário</b> para escrita (se arquivo já existe, conteúdo é apagado, se arquivo não existe, será criado)
ab	Abre um arquivo <b>binário</b> para escrita, preservando o conteúdo já existente (se arquivo não existe, será criado)
r+	Abre um arquivo <b>texto</b> para leitura e escrita (se arquivo não existe, retorna NULL)
w+	Cria e abre um arquivo <b>texto</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
a+	Abre (se já existir) ou cria um arquivo <b>texto</b> para leitura e escrita – cursor é posicionado no final do arquivo
rb+	Abre um arquivo <b>binário</b> para leitura e escrita (se arquivo não existe, retorna NULL)
wb+	Cria um arquivo <b>binário</b> para leitura e escrita (se ar
ab+	Abre (se já existir) um arquivo <b>binário</b> para leitura e escrita – cursor é posicionado sempre no final do arquivo – append). Se não existir,

Arquivo Binário

# Modos de Abertura de Arquivos

Modo	Significado
r	Abre um arquivo <b>texto</b> para leitura (se não existir, retorna NULL)
w	Abre um arquivo <b>texto</b> para escrita (se já existir, conteúdo é apagado)
a	Abre (se já existir) ou cria um arquivo <b>texto</b> para escrita – cursor é posicionado no final do arquivo
rb	Abre um arquivo <b>binário</b> para leitura (se arquivo não existe, retorna NULL)
wb	Abre um arquivo <b>binário</b> para escrita (se arquivo já existe, conteúdo é apagado, se arquivo não existe, será criado)
ab	Abre um arquivo <b>binário</b> para escrita, preservando o conteúdo já existente (se arquivo não existe, será criado)
r+	Abre um arquivo <b>texto</b> para leitura e escrita (se arquivo não existe, retorna NULL)
w+	Cria e abre um arquivo <b>texto</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
a+	Abre (se já existir) ou cria um arquivo <b>texto</b> para leitura e escrita – cursor é posicionado no final do arquivo
rb+	Abre um arquivo <b>binário</b> para leitura e escrita (se arquivo não existe, retorna NULL)
wb+	Cria um arquivo <b>binário</b> para leitura e escrita (se arquivo já existe, conteúdo é apagado)
ab+	Abre (se já existir) um arquivo <b>binário</b> para leitura e escrita, preservando o conteúdo já existente (escreve sempre no final do arquivo – append). Se não existir, cria arquivo.

## Leitura e Escrita

# Na nossa disciplina

---



Nosso foco na  
disciplina:

Arquivo será  
**SEMPRE**  
arquivo em  
disco

Arquivo será  
**SEMPRE**  
arquivo  
**BINÁRIO**



# Exemplo

---

- ▶ Gravar registros de funcionários num arquivo



# Struct Funcionário

---

```
typedef struct Funcionario {  
    int cod;  
    char nome[50];  
    char cpf[15];  
    char data_nascimento[11];  
    double salario;  
} Funcionario;
```



# Salva Funcionário

---

```
// Salva no arquivo out, na posição atual do cursor
void salva(Funcionario *func, FILE *out) {
    fwrite(&func->cod, sizeof(int), 1, out);
    //func->nome ao invés de &func->nome,
    //pois string já é ponteiro
    fwrite(func->nome, sizeof(char),
           sizeof(func->nome), out);
    fwrite(func->cpf, sizeof(char),
           sizeof(func->cpf), out);
    fwrite(func->data_nascimento, sizeof(char),
           sizeof(func->data_nascimento), out);
    fwrite(&func->salario, sizeof(double), 1, out);
}
```



# Salva Funcionário

```
// Salva no arquivo out, na posição atu
void salva(Funcionario *func, FILE *out
    fwrite(&func->cod, sizeof(int), 1, out);
    //func->nome ao invés de &func->nome,
    //pois string já é ponteiro
    fwrite(&func->nome, sizeof(char),
        sizeof(func->nome), out);
    fwrite(&func->cpf, sizeof(char),
        sizeof(func->cpf), out);
    fwrite(&func->data_nascimento, sizeof(char),
        sizeof(func->data_nascimento), out);
    fwrite(&func->salario, sizeof(double), 1, out);
}
```

ORDEM  
é importante –  
posteriormente, registro  
deverá ser lido nessa  
mesma ordem

# Salva Funcionário

```
// Salva no arquivo out, na posição atual do cursor
void salva(Funcionario *func, FILE *out) {
    fwrite(&func->cod, sizeof(int), 1, out);
    //func->nome ao invés de &func->nome,
    //pois string já é ponteiro
    fwrite(func->nome, sizeof(char),
           sizeof(func->nome), out);
    fwrite(func->cpf, sizeof(char),
           sizeof(func->cpf), out);
    fwrite(func->data_nascimento, sizeof(char),
           sizeof(func->data_nascimento), out);
    fwrite(&func->salario, sizeof(double), 1, out);
}
```

Dado a ser gravado

Arquivo destino

Tamanho em bytes do  
dado a ser gravado

Número de itens a serem  
gravados

# Ler um registro de funcionário

---

```
// Lê do arquivo in na posição atual do cursor
// Retorna um ponteiro para funcionário lido do arquivo
Funcionario *le(FILE *in) {
    Funcionario *func = (Funcionario *)
                           malloc(sizeof(Funcionario));
    if (0 >= fread(&func->cod, sizeof(int), 1, in)) {
        free(func);
        return NULL;
    }
    fread(func->nome, sizeof(char), sizeof(func->nome), in);
    fread(func->cpf, sizeof(char), sizeof(func->cpf), in);
    fread(func->data_nascimento, sizeof(char),
          sizeof(func->data_nascimento), in);
    fread(&func->salario, sizeof(double), 1, in);
    return func;
}
```



# Ler um registro de funcionário

```
// Lê do arquivo in na posição atual do cursor  
// Retorna um ponteiro para funcionário lido  
Funcionario *le(FILE *in) {  
    Funcionario *func = (Funcionario *)  
        malloc(sizeof(Funcionario));  
    if (0 >= fread(&func->cod, sizeof(int), 1, in)) {  
        free(func);  
        return NULL;  
    }  
    fread(func->nome, sizeof(char), sizeof(func->nome), in);  
    fread(func->cpf, sizeof(char), sizeof(func->cpf), in);  
    fread(func->data_nascimento, sizeof(char),  
          sizeof(func->data_nascimento), in);  
    fread(&func->salario, sizeof(double), 1, in);  
    return func;  
}
```

Leitura na mesma  
ORDEM  
da gravação

Variável para guardar  
dado a ser lido

# Ler um registro de funcionário

```
// Lê do arquivo in na posição atual do cursor
// Retorna um ponteiro para funcionário lido do arquivo
Funcionario *le(FILE *in) {
    Funcionario *func = (Funcionario *)
        malloc(sizeof(Funcionario));
    if (0 >= fread(&func->cod, sizeof(int), 1, in)) {
        free(func);
        return NULL;
    }
    fread(&func->nome, sizeof(char), sizeof(func->nome),
            in);
    fread(&func->sobrenome, sizeof(char), sizeof(func->sobrenome),
            in);
    fread(&func->data_nascimento, sizeof(char),
            sizeof(func->data_nascimento), in);
    fread(&func->salario, sizeof(double), 1, in);
    return func;
}
```

Tamanho em bytes do  
dado a ser lido

Arquivo fonte

Número de itens a serem  
lidos

# Colocando tudo em ação...

---

- ▶ Acompanhem o exemplo do tutorial de manipulação de arquivos em C, disponível no site da disciplina



# Exercício

---

- ▶ Modifique a implementação para lidar com registros de conta-corrente
- ▶ Dois arquivos:
  - ▶ Agência (Cod, Nome, Gerente)
  - ▶ Conta-Corrente (Cod, CodAgencia, Saldo)
- ▶ Usuário deve poder escolher o que quer cadastrar
- ▶ Dados devem ser lidos do teclado
- ▶ Aplicação deve ter opção de Cadastrar, Ler ou Sair

