

Algoritmos e Estruturas de Dados II

Aula 16 – Arquivos - Introdução

Prof. Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

2025

O que é um arquivo?

O que é um arquivo?

- Do ponto de vista **físico**:

O que é um arquivo?

- Do ponto de vista **físico**:
 - Uma sequência de bytes armazenados em algum lugar (disco, SSD, pen-drive, fita etc.)

O que é um arquivo?

- Do ponto de vista **físico**:
 - Uma sequência de bytes armazenados em algum lugar (disco, SSD, pen-drive, fita etc.)
 - Pode ser visto como um espaço de endereçamento (em disco, SSD etc.)

O que é um arquivo?

- Do ponto de vista **físico**:
 - Uma sequência de bytes armazenados em algum lugar (disco, SSD, pen-drive, fita etc.)
 - Pode ser visto como um espaço de endereçamento (em disco, SSD etc.)
- Do ponto de vista **lógico**:

O que é um arquivo?

- Do ponto de vista **físico**:
 - Uma sequência de bytes armazenados em algum lugar (disco, SSD, pen-drive, fita etc.)
 - Pode ser visto como um espaço de endereçamento (em disco, SSD etc.)
- Do ponto de vista **lógico**:
 - Fluxo de bytes (*stream*) – ex: sequência digitada no teclado, o que está sendo “impresso” na tela etc.

Tipos de arquivos^a:

^aHá vários outros:

<https://docs.fileformat.com/>

Tipos de arquivos^a:

- Texto (.txt, .doc, ...):
- Imagem (.tiff, .jpeg, .gif, ...)
- Planilha (.xls, .csv, ...)
- Páginas Web (.asp, .htm, .html, .mhtml, .xhtml, ...)
- Executável (.exe)
- ...

^aHá vários outros:

<https://docs.fileformat.com/>

Arquivos

Tipos de arquivos^a:

- Texto (.txt, .doc, ...):
- Imagem (.tiff, .jpeg, .gif, ...)
- Planilha (.xls, .csv, ...)
- Páginas Web (.asp, .htm, .html, .mhtml, .xhtml, ...)
- Executável (.exe)
- ...

O que define um tipo de arquivo?

^aHá vários outros:

<https://docs.fileformat.com/>

Arquivos

Tipos de arquivos^a:

- Texto (.txt, .doc, ...):
- Imagem (.tiff, .jpeg, .gif, ...)
- Planilha (.xls, .csv, ...)
- Páginas Web (.asp, .htm, .html, .mhtml, .xhtml, ...)
- Executável (.exe)
- ...

O que define um tipo de arquivo?

A definição de seu formato / conteúdo.

^aHá vários outros:

<https://docs.fileformat.com/>

Arquivos

Exemplo: Arquivo texto (.txt)

Arquivos

Exemplo: Arquivo texto (.txt)

- Texto simples (sequências de caracteres) estruturado em linhas

Arquivos

Exemplo: Arquivo texto (.txt)

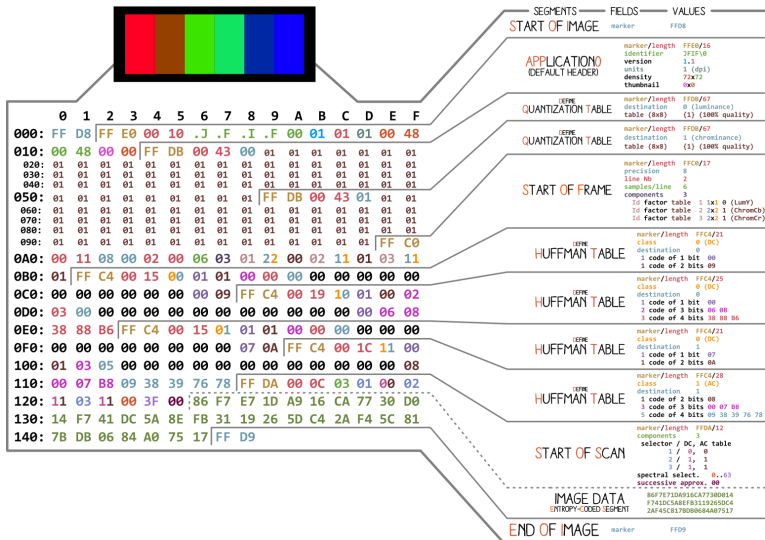
- Texto simples (sequências de caracteres) estruturado em linhas
- Pode ser aberto (lido/editado) em qualquer programa editor de texto

Exemplo.txt

```
Prezados(as) alunos(as)  
Não esqueçam de estudar para as  
provas, fazer os trabalhos e beber  
água.  
Atenciosamente,  
Digi
```

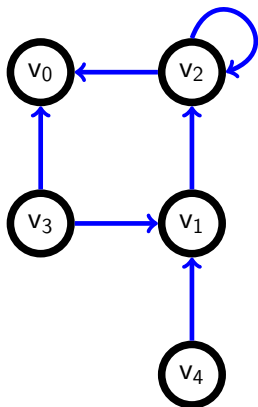
Arquivos

Exemplo: Arquivo JPG



Salvando Arquivos

Como poderíamos salvar nossos grafos?



Imprimindo grafo (vértices: 5; arestas: 6).

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	1	0	0
2	1	0	1	0	0
3	1	1	0	0	0
4	0	1	0	0	0

Salvando Arquivos

Arquivo Texto (57 bytes):

5

0 0 0 0 0

0 0 1 0 0

1 0 1 0 0

1 1 0 0 0

0 1 0 0 0

Salvando Arquivos

Arquivo Texto (57 bytes):

```
5
0 0 0 0 0
0 0 1 0 0
1 0 1 0 0
1 1 0 0 0
0 1 0 0 0
```

Arquivo Binário - salvando cada informação em um byte (26 bytes).

```
00000101 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001 00000000 00000000 00000001
00000000 00000001 00000000 00000000 00000001 00000001
00000000 00000000 00000000 00000000 00000001 00000000
00000000 00000000
```

Salvando Arquivos

Arquivo Texto (57 bytes):

```
5
0 0 0 0 0
0 0 1 0 0
1 0 1 0 0
1 1 0 0 0
0 1 0 0 0
```

Arquivo Binário - salvando cada informação em um byte (26 bytes).

```
00000101 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001 00000000 00000000 00000001
00000000 00000001 00000000 00000000 00000001 00000001
00000000 00000000 00000000 00000000 00000001 00000000
00000000 00000000
```

Arquivo Binário (5 bytes):

```
00000101 00000001 00101001 10000100 00000000
```

FILE* fopen(char* filename, char* mode)

- **fopen**: função para abrir um arquivo
 - *retorno*: endereço de memória de um 'arquivo' (FILE*)
 - *filename*: nome do arquivo
 - *mode*: forma de abertura do arquivo (para leitura "r", escrita "w", acréscimo "a" ...)

Arquivos - Comandos em C

FILE* fopen(char* filename, char* mode)

- **fopen**: função para abrir um arquivo
 - *retorno*: endereço de memória de um 'arquivo' (FILE*)
 - *filename*: nome do arquivo
 - *mode*: forma de abertura do arquivo (para leitura "r", escrita "w", acréscimo "a" ...)

int fclose(FILE* file)

- **fclose**: função para fechar um arquivo
 - *retorno*: valor inteiro igual a zero se conseguiu fechar o arquivo ou *EOF*, caso contrário
 - *file*: ponteiro/referência para um arquivo (FILE*)

Arquivos - Comandos em C

int feof(FILE* file)

- feof: função que verifica fim de arquivo (*end-of-file*)
 - *retorno*: 0 (zero) se o fim de arquivo não foi encontrado (se a respectiva *flag* não estiver ativa); e não-zero (tipicamente 1) caso contrário.
 - *file*: ponteiro/referência para um arquivo (FILE*)

Arquivos - Comandos em C

int feof(FILE* file)

- feof: função que verifica fim de arquivo (*end-of-file*)
 - *retorno*: 0 (zero) se o fim de arquivo não foi encontrado (se a respectiva *flag* não estiver ativa); e não-zero (tipicamente 1) caso contrário.
 - *file*: ponteiro/referência para um arquivo (FILE*)

int fprintf(FILE* file, char* format, ...)

- fprintf: envia saída formatada para um 'arquivo'
 - *retorno*: número de caracteres escritos, ou valor negativo se falhou
 - *file*: ponteiro/referência para um arquivo (FILE*)
 - *format*: texto a ser escrito e máscara/string de controle
 - *parâmetros adicionais*: conteúdos/variáveis cujo conteúdo será escrito

int fscanf(FILE* file, char* format, ...)

- **fscanf**: função para ler conteúdo formatado de um 'arquivo'
 - *retorno*: valor inteiro igual ao número de elementos que foram lidos
 - *file*: ponteiro/referência para um arquivo (FILE*)
 - *format*: máscara indicando o que será lido
 - *parâmetros adicionais*: endereços de memória que receberão os conteúdos lidos

size_t fwrite(void* ptr, size_t size, size_t nmemb, FILE* file)

- **fwrite**: escreve no arquivo os dados apontados pelo endereço passado como parâmetro
 - *retorno*: número de elementos escritos no arquivo
 - *ptr*: referência/endereço de memória do conteúdo a ser escrito
 - *size*: tamanho em bytes de cada elemento a serem escritos
 - *nmemb*: número de elementos a serem escritos
 - *file*: ponteiro/referência para um arquivo (FILE*)

Arquivos - Comandos em C

int fgetc(FILE* file)

- **fgetc**: função que retorna o próximo caractere do 'arquivo' e avança uma posição nele
 - *retorno*: o caractere atual do arquivo ou *EOF*, caso contrário
 - *file*: ponteiro/referência para um arquivo (FILE*)

Arquivos - Comandos em C

int fgetc(FILE* file)

- **fgetc**: função que retorna o próximo caractere do 'arquivo' e avança uma posição nele
 - *retorno*: o caractere atual do arquivo ou *EOF*, caso contrário
 - *file*: ponteiro/referência para um arquivo (FILE*)

char* fgets(char* str, int n, FILE* file)

- **fgets**: função para retornar uma string a partir do 'arquivo'
 - *retorno*: a string que foi colocada em *str ou NULL em caso de falha
 - *n*: número máximo de caracteres a serem lidos
 - *file*: ponteiro/referência para um arquivo (FILE*)

Arquivos - Características Desejadas

Algumas das principais características desejáveis em arquivos:

Arquivos - Características Desejadas

Algumas das principais características desejáveis em arquivos:

- **Persistência:** conteúdo precisa ser mantido mesmo quando o computador é desligado

Arquivos - Características Desejadas

Algumas das principais características desejáveis em arquivos:

- **Persistência:** conteúdo precisa ser mantido mesmo quando o computador é desligado
- **Concorrência:** múltiplos processos devem poder acessar o mesmo arquivo simultaneamente

Arquivos - Características Desejadas

Algumas das principais características desejáveis em arquivos:

- **Persistência:** conteúdo precisa ser mantido mesmo quando o computador é desligado
- **Concorrência:** múltiplos processos devem poder acessar o mesmo arquivo simultaneamente
- **Alta capacidade de armazenamento:** possibilidade de se armazenar uma grande quantidade de informação, maior do que a capacidade da memória principal

Estrutura Interna e Alocação

- Muitos arquivos de dados são compostos por uma lista de dados, cada um podendo conter vários campos
 - Por exemplo, linhas de uma tabela, seja de uma planilha, de um arquivo .csv, de uma tabela de banco de dados, ...
 - Cada dado normalmente tem uma chave, pela qual é possível realizar operações de busca e ordenação
- Chamaremos cada dado (contendo a chave e demais campos) de **registro**

Estrutura Interna e Alocação

- Muitos arquivos de dados são compostos por uma lista de dados, cada um podendo conter vários campos
 - Por exemplo, linhas de uma tabela, seja de uma planilha, de um arquivo .csv, de uma tabela de banco de dados, ...
 - Cada dado normalmente tem uma chave, pela qual é possível realizar operações de busca e ordenação
- Chamaremos cada dado (contendo a chave e demais campos) de **registro**
- Como separar os campos? Como separar os registros?

Estrutura Interna e Alocação

NUSP	Nome	Ano de Ingresso	e-mail
3140792	Luciano Antonio Digiampietri	2008	digiampietri@usp.br
12345678	Norton Trevisan Roman	2010	norton@usp.br
7654321	Daniel de Angelis Cordeiro	2015	daniel.cordeiro@usp.br

...

Estrutura Interna e Alocação

NUSP	Nome	Ano de Ingresso	e-mail
3140792	Luciano Antonio Digiampietri	2008	digiampietri@usp.br
12345678	Norton Trevisan Roman	2010	norton@usp.br
7654321	Daniel de Angelis Cordeiro	2015	daniel.cordeiro@usp.br

...

Como gravar/organizar esse conjunto de dados em disco?

Estrutura Interna e Alocação

01001100
01110101
01100011
01101001
01100001
01101110
01101111
00100000
01000001
01101110
01110100
01101111
01101110
01101001
01101111
00100000
01000100
01101001
01100111
01101001
01100001
01101101
01110000
01101001
01100101
01110100
01110010
01101001
00100000
00100000
00100000
00100000
00100000
00100000
00100000
01001100
01110101
01100011
01101001
01100001
01101110

3140792 -
Luciano
Antonio
Digiampietri
- 2008 -
digiampietri
@usp.br
12345678 -
Norton
Trevisan
Roman -
2010 -
norton
@usp.br
7654321 -
Daniel de
Angelis
Cordeiro -
2015 -
daniel.corde
iro @usp.br

...

...

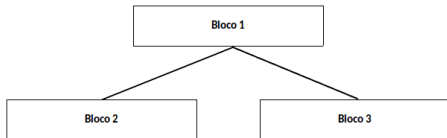
...

...

Bloco 1

Bloco 2

Bloco 3



Sequência de Bytes

Sequência de Registros

Conjunto de Blocos

Blocos Organizados como uma Árvore

Organização dos Dados

Em arquivos de dados, como organizar:

Organização dos Dados

Em arquivos de dados, como organizar:

- Os **campos** dentro de cada registro?

Organização dos Dados

Em arquivos de dados, como organizar:

- Os **campos** dentro de cada registro?
- Os **registros** dentro do arquivo?

Organização dos Campos

Os campos podem ser organizados utilizando:

- Comprimento fixo
- Indicador de comprimento
- Delimitadores
- Marcadores (*tags*)

Campos com Comprimento Fixo

Cada campo ocupa no arquivo um tamanho fixo.
Por exemplo: número USP (10 chars), nome (30 chars),
ano de ingresso (4 chars)

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

Campos com Comprimento Fixo

Cada campo ocupa no arquivo um tamanho fixo.
Por exemplo: número USP (10 chars), nome (30 chars),
ano de ingresso (4 chars)

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

- Com base nessa informação, como eu carrego os campos?

Campos com Comprimento Fixo

Cada campo ocupa no arquivo um tamanho fixo.
Por exemplo: número USP (10 chars), nome (30 chars),
ano de ingresso (4 chars)

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

- Com base nessa informação, como eu carrego os campos?
 - Leio exatamente a quantidade de bytes/caracteres que eu preciso.

Campos com Comprimento Fixo

Cada campo ocupa no arquivo um tamanho fixo.
Por exemplo: número USP (10 chars), nome (30 chars),
ano de ingresso (4 chars)

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

- E se eu quiser acessar o terceiro campo da i -ésima linha?

Campos com Comprimento Fixo

Cada campo ocupa no arquivo um tamanho fixo.
Por exemplo: número USP (10 chars), nome (30 chars),
ano de ingresso (4 chars)

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

- E se eu quiser acessar o terceiro campo da i -ésima linha?
 - Faço a conta para acessar apenas os bytes desejados.
Cada linha tem 45 bytes ($10+30+4+1$), incluindo o *fim de linha*.

Campos com Comprimento Fixo

Cada campo ocupa no arquivo um tamanho fixo.
Por exemplo: número USP (10 chars), nome (30 chars),
ano de ingresso (4 chars)

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

- E se eu quiser acessar o terceiro campo da i -ésima linha?
 - Faço a conta para acessar apenas os bytes desejados.
Cada linha tem 45 bytes ($10+30+4+1$), incluindo o *fim de linha*.
 - $(i-1)*45 + 10 + 30$: para $i=2$, há 85 bytes antes, quero os bytes 86, 87, 88 e 89

Campos com Comprimento Fixo

É como se nosso arquivo (conjunto de registros) fosse um arranjo de nossos registros¹

```
typedef struct
    char nusp[10];
    char nome[30];
    char anoDeIngresso[4];
    PROF;
```

¹Colocamos a quebra de linha entre registros para facilitar a visualização, mas do ponto de vista do código isso não seria necessário.

Uma das vantagens: Podemos salvar e ler um registro muito facilmente

Uma das vantagens: Podemos salvar e ler um registro muito facilmente

```
fwrite(prof, sizeof(PROF), 1, arq);  
fwrite("\n", 1, 1, arq);
```

```
fgets(buffer, sizeof(PROF)+1, arq);  
imprimirProfessor((PROF*)buffer);
```

Sendo **arq** do tipo FILE*, **prof** do tipo PROF* e **buffer** do tipo char*.

Vejam o código completo em:

<http://www.each.usp.br/digiampietri/ACH2024/arquivosTamanhoFixo2.c>

Desvantagens dessa abordagem:

- **Desperdício** de espaço: os campos são dimensionados pelo tamanho máximo 'possível' para aquele campo.
 - Solução **adequada** quando o comprimento dos campos é realmente fixo ou possui pouca variação
 - Solução **inapropriada** quando os dados possuem tamanhos variáveis

Campos com Indicador de Comprimento

O **tamanho** de cada campo é armazenado imediatamente antes do dado.

- Se armazenarmos de forma binária, com um único byte podemos representar tamanhos inferiores a 256.

7314079228Luciano Antonio Digiampietri42008

81234567821Norton Trevisan Roman42010

7765432126Daniel de Angelis Cordeiro42015

Campos com Indicador de Comprimento

O **tamanho** de cada campo é armazenado imediatamente antes do dado.

- Se armazenarmos de forma binária, com um único byte podemos representar tamanhos inferiores a 256.

7314079228Luciano Antonio Digiampietri42008

81234567821Norton Trevisan Roman42010

7765432126Daniel de Angelis Cordeiro42015

Desvantagens:

Campos com Indicador de Comprimento

O **tamanho** de cada campo é armazenado imediatamente antes do dado.

- Se armazenarmos de forma binária, com um único byte podemos representar tamanhos inferiores a 256.

7314079228Luciano Antonio Digiampietri42008

81234567821Norton Trevisan Roman42010

7765432126Daniel de Angelis Cordeiro42015

Desvantagens:

- Há um gasto extra por campo (pode ou não valer a pena, dependendo da aplicação)

Campos com Indicador de Comprimento

O **tamanho** de cada campo é armazenado imediatamente antes do dado.

- Se armazenarmos de forma binária, com um único byte podemos representar tamanhos inferiores a 256.

7314079228Luciano Antonio Digiampietri42008

81234567821Norton Trevisan Roman42010

7765432126Daniel de Angelis Cordeiro42015

Desvantagens:

- Há um gasto extra por campo (pode ou não valer a pena, dependendo da aplicação)
- É custoso acessar um campo específico (não é possível fazer uma conta 'fácil' para isso)

Campos Separados por Delimitadores

Caracteres especiais são utilizados para indicar o fim de cada campo (por exemplo |, ,, ;, #, %, \$, ...

- Exemplo: arquivos .csv

```
3140792|Luciano Antonio Digiampietri|2008|  
12345678|Norton Trevisan Roman|2010|  
7654321|Daniel de Angelis Cordeiro|2015|
```

Campos Separados por Delimitadores

Caracteres especiais são utilizados para indicar o fim de cada campo (por exemplo |, ,, ;, #, %, \$, ...

- Exemplo: arquivos .csv

```
3140792|Luciano Antonio Digiampietri|2008|  
12345678|Norton Trevisan Roman|2010|  
7654321|Daniel de Angelis Cordeiro|2015|
```

Desvantagens:

- Há um gasto extra por campo (de um caractere)
- É custoso acessar um campo específico (não é possível fazer uma conta para isso)

Uso de um Marcador (*chave=valor*)

Vantagem: o campo fornece *informação semântica* sobre si.

- Fica mais fácil identificar o conteúdo do arquivo e campos ausentes não precisam ser representados.
- Exemplo: arquivos .xml

```
NUSP=3140792|NOME=Luciano Antonio Digiampietri|ANO=2008|
```

```
NUSP=12345678|NOME=Norton Trevisan Roman|ANO=2010|
```

```
NUSP=7654321|NOME=Daniel de Angelis Cordeiro|ANO=2015|
```

Uso de um Marcador (*chave=valor*)

Vantagem: o campo fornece *informação semântica* sobre si.

- Fica mais fácil identificar o conteúdo do arquivo e campos ausentes não precisam ser representados.
- Exemplo: arquivos .xml

```
NUSP=3140792|NOME=Luciano Antonio Digiampietri|ANO=2008|  
NUSP=12345678|NOME=Norton Trevisan Roman|ANO=2010|  
NUSP=7654321|NOME=Daniel de Angelis Cordeiro|ANO=2015|
```

Desvantagem:

- As marcações podem ocupar grande parte do arquivo (inclusive mais do que os dados propriamente ditos)

Organização dos Registros

Os **registros** podem ser organizados utilizando:

- Comprimento fixo
- Número fixo de campos
- Índice
- Delimitadores

Registros de Tamanho Fixo

Seguindo a mesma ideia dos campos de tamanho fixo, porém aqui podemos ter **registros de tamanho fixo** com campos de tamanho fixo ou não.

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

3140792	Luciano Antonio Digiampietri	2008	
12345678	Norton Trevisan Roman	2010	
7654321	Daniel de Angelis Cordeiro	2015	

Registros de Tamanho Fixo

Seguindo a mesma ideia dos campos de tamanho fixo, porém aqui podemos ter **registros de tamanho fixo** com campos de tamanho fixo ou não.

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

3140792	Luciano Antonio Digiampietri	2008	
12345678	Norton Trevisan Roman	2010	
7654321	Daniel de Angelis Cordeiro	2015	

Uma das abordagens mais usadas em arquivos.

Registros de Tamanho Fixo

Seguindo a mesma ideia dos campos de tamanho fixo, porém aqui podemos ter **registros de tamanho fixo** com campos de tamanho fixo ou não.

3140792	Luciano Antonio Digiampietri	2008
12345678	Norton Trevisan Roman	2010
7654321	Daniel de Angelis Cordeiro	2015

3140792	Luciano Antonio Digiampietri	2008	
12345678	Norton Trevisan Roman	2010	
7654321	Daniel de Angelis Cordeiro	2015	

Uma das abordagens mais usadas em arquivos.

- **Custo x Benefício:** desperdício de espaço x facilidade de cálculo da posição de cada registro

Registros com Número Fixo de Campos

O número de campos é definido (campos de tamanho variável).

- O tamanho do registro é **variável**
- Os campos possuem algum separador/delimitador

```
3140792|Luciano Antonio Digiampietri|2008|12345678|Norton Trevisan  
Roman|2010|7654321|Daniel de Angelis Cordeiro|2015|
```

Registros com Número Fixo de Campos

O número de campos é definido (campos de tamanho variável).

- O tamanho do registro é **variável**
- Os campos possuem algum separador/delimitador

```
3140792|Luciano Antonio Digiampietri|2008|12345678|Norton Trevisan  
Roman|2010|7654321|Daniel de Angelis Cordeiro|2015|
```

- **Custo x Benefício:** dificuldade para acessar um registro específico x potencial economia de espaço

Campo Indicador do Tamanho do Registro

O **tamanho total** do registro é armazenado em seu início.

- Campos são separados por delimitadores
- Permite o deslocamento entre registros sem a necessidade de analisar o registro inteiro

```
423140792|Luciano Antonio Digiampietri|2008|3612345678|Norton  
Trevisan Roman|2010|407654321|Daniel de Angelis Cordeiro|2015|
```

Campo Indicador do Tamanho do Registro

O **tamanho total** do registro é armazenado em seu início.

- Campos são separados por delimitadores
- Permite o deslocamento entre registros sem a necessidade de analisar o registro inteiro

423140792|Luciano Antonio Digiampietri|2008|3612345678|Norton
Trevisan Roman|2010|407654321|Daniel de Angelis Cordeiro|2015|

- **Custo x Benefício:** não consigo acessar um registro arbitrário, mas consigo 'pular' rapidamente para o próximo registro x potencial economia de espaço (custo extra potencialmente pequeno)

Uso de um Índice

Um **índice externo** indica o deslocamento de cada registro em relação ao início do arquivo.

- Campos são separados por delimitadores
- O índice também serve para calcular o tamanho do registro

```
3140792|Luciano Antonio Digiampietri|2008|12345678|Norton  
Trevisan Roman|2010|7654321|Daniel de Angelis Cordeiro|2015|
```

Índice: 00 42 78

Uso de um Índice

Um **índice externo** indica o deslocamento de cada registro em relação ao início do arquivo.

- Campos são separados por delimitadores
- O índice também serve para calcular o tamanho do registro

```
3140792|Luciano Antonio Digiampietri|2008|12345678|Norton  
Trevisan Roman|2010|7654321|Daniel de Angelis Cordeiro|2015|
```

Índice: 00 42 78

- **Custo x Benefício:** exige uma estrutura/arquivo extra x permite acessar registros arbitrários de forma relativamente rápida

Uso de Delimitadores de Registros

Os registros são separadores por **delimitadores**, diferentes dos usados para separar campos.

```
3140792|Luciano Antonio Digiampietri|2008|#12345678|Norton  
Trevisan Roman|2010|#7654321|Daniel de Angelis Cordeiro|2015|#
```

Uso de Delimitadores de Registros

Os registros são separadores por **delimitadores**, diferentes dos usados para separar campos.

```
3140792|Luciano Antonio Digiampietri|2008|#12345678|Norton  
Trevisan Roman|2010|#7654321|Daniel de Angelis Cordeiro|2015|#
```

- **Custo x Benefício:** não é fácil navegar entre registros ou campos x é uma das abordagens mais econômicas em termos de espaço (exceto se os dados, naturalmente, tiverem tamanho fixo)

Cabeçalho de Arquivos

- Um formato de arquivo precisa definir como os campos/registros são organizados

Cabeçalho de Arquivos

- Um formato de arquivo precisa definir como os campos/registros são organizados
- O **cabeçalho do arquivo** (descritor na parte inicial do arquivo) pode conter informações (meta-informações) como:

Cabeçalho de Arquivos

- Um formato de arquivo precisa definir como os campos/registros são organizados
- O **cabeçalho do arquivo** (descriptor na parte inicial do arquivo) pode conter informações (meta-informações) como:
 - Descrição dos formatos/tamanhos dos campos de um registro
 - Códigos/marcadores para registros de tamanho variável
 - Data de criação e atualização ...

Cabeçalho de Arquivos

- Um formato de arquivo precisa definir como os campos/registros são organizados
- O **cabeçalho do arquivo** (descriptor na parte inicial do arquivo) pode conter informações (meta-informações) como:
 - Descrição dos formatos/tamanhos dos campos de um registro
 - Códigos/marcadores para registros de tamanho variável
 - Data de criação e atualização ...
- Permite a definição de padrões (pdf, tiff, jpg etc.) e facilita conversão entre padrões

Referência

- Slides baseados no material da profa. Arianne Machado Lima - ACH2024
- Slides da Profa. Graça (ICMC) - [http://wiki.icmc.usp.br/index.php/SCC-203_\(gracan\)](http://wiki.icmc.usp.br/index.php/SCC-203_(gracan)) (Arquivos 8, 9 e 12)
- GOODRICH et al, Data Structures and Algorithms in C++. Ed. John Wiley & Sons, Inc. 2nd ed. 2011. Seção 14.2
- ELMARIS, R.; NAVATHE, S. B. Fundamentals of Database Systems. 4 ed. Ed. Pearson-Addison Wesley. Cap 13 (até a seção 13.7).
- TANEMBAUM, A. S. & BOS, H. Modern Operating Systems. Pearson, 4th ed. 2015

Algoritmos e Estruturas de Dados II

Aula 16 – Arquivos - Introdução

Prof. Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

2025