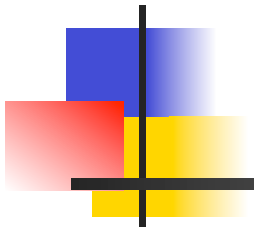
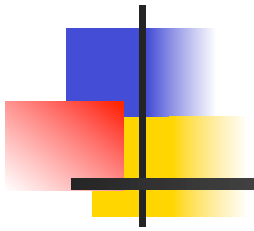
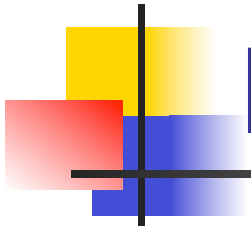
Organização de Arquivos
e
Acesso a Arquivos



Cristina D. A. Ciferri
Thiago A. S. Pardo
Leandro C. Cintra
M.C.F. de Oliveira
Moacir Ponti Jr.



Organização de Arquivos

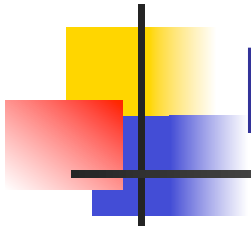


Por que Organizar Arquivos?

- Considere o seguinte *stream* (fluxo) de bytes
  - AmesJohn123 MapleStillwaterOK74075MasonAlan90 EastgateAdaOK74820

- Perguntas:
  - quais dados são armazenados?
  - onde começa e onde termina cada dado?
  - como recuperar um dado específico?

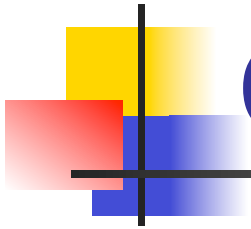
Necessidade de organização adequada de dados em arquivos!



Formas de Organização

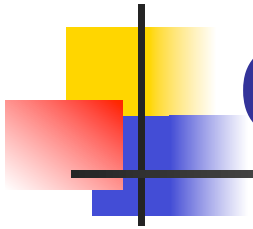
- Organização em campos
  - menor unidade lógica de informação em um arquivo
- Organização em registros
  - conjunto de campos agrupados, os quais estão logicamente associados a uma mesma entidade

Os conceitos de campo e de registro correspondem a ferramentas conceituais, que não necessariamente existem no sentido físico



Métodos para Organização em Campos

- Forçar todos os campos para um tamanho (comprimento) fixo
- Começar cada campo com um indicador de tamanho (comprimento)
- Colocar delimitadores entre campos
- Usar expressões (*tags*) "keyword=value" para identificar cada campo e seu conteúdo



Campos com Tamanho Fixo

- Cada campo ocupa no arquivo um tamanho fixo, pré-determinado
- O fato do tamanho ser conhecido garante que é possível recuperar cada campo

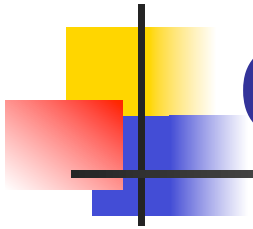
Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Campos com Tamanho Fixo

```
char last[10];
char first[10];
char city[15];
char state[2];
char zip[9];
```

ou

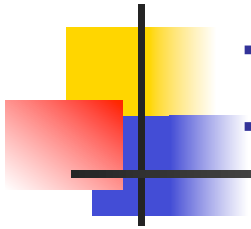
```
struct {
        char last[10];
        char first[10];
        char city[15];
        char state[2];
        char zip[9];
} set_of_fields;
```



Campos com Tamanho Fixo

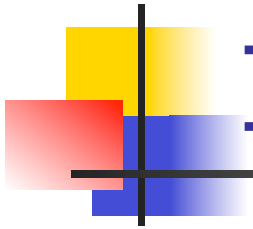
- Desvantagens
  - espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (desperdício de espaço de armazenamento)
  - solução inapropriada quando se tem uma grande quantidade de dados com tamanho variável
- Vantagens
  - facilidade na pesquisa
  - indicado para situações nas quais o comprimento dos campos é fixo ou apresenta pouca variação

Campos com Indicador de Tamanho



- Tamanho de cada campo
  - armazenado imediatamente antes do dado
  - não contabilizado no tamanho do campo

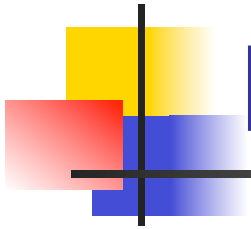
05Maria05Rua 10312310São Carlos
04João05Rua A0325509Rio Claro
05Pedro06Rua 10025610Rib. Preto



Campos com Indicador de Tamanho

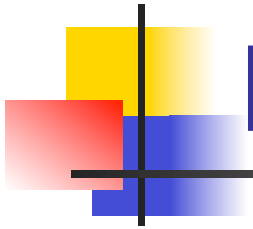
- Vantagem
  - economia de espaço de armazenamento
- Desvantagem
  - dificuldade na pesquisa

Campos Separados por Delimitadores



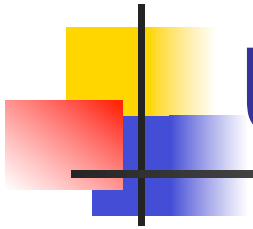
- Caractere(s) especial(ais) que não fazem parte do domínio do dado
  - escolhido(s) para ser(em) inserido(s) ao final de cada campo
- Ex.: para o campo *nome*
  - pode-se utilizar /, tab, #, etc ...
  - não pode-se usar espaços em branco ...

```
Maria|Rua 1|123|São Carlos|
João|Rua A|255|Rio Claro|
Pedro|Rua 10|56|Rib. Preto|
```



Campos Separados por Delimitadores

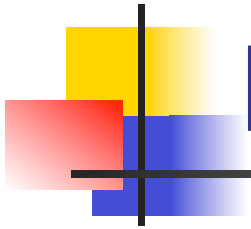
- Vantagem
  - economia de espaço de armazenamento
- Desvantagens
  - dificuldade na pesquisa
  - necessidade de escolha de um delimitador que não pertence ao domínio dos dados



Uso de *Tags*

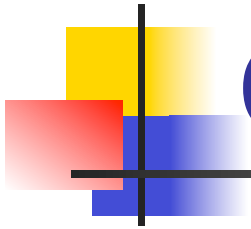
- Expressão "keyword=value"
  - colocada antes do campo
  - possui semântica que explica o significado do campo
- Geralmente usada em conjunto com outro método, como delimitadores

```
Nome=Maria|Endereço=Rua 1|Número=123|Cidade=São Carlos|
Nome=João|Endereço=Rua A|Número=255|Cidade=Rio Claro|
Nome=Pedro|Endereço=Rua 10|Número=56|Cidade=Rib. Preto|
```



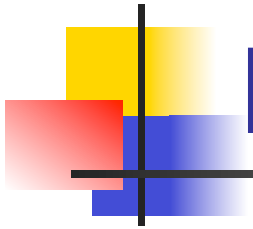
Uso de *Tags* Expressão "keyword=value"

- Vantagens
  - campo fornece informação semântica sobre si
  - fica mais fácil identificar o conteúdo do arquivo
  - fica mais fácil identificar campos "vazios"
    - *keyword* não aparece
- Desvantagem
  - as *keywords* podem ocupar uma porção significativa do arquivo (desperdício de espaço de armazenamento)



Métodos para Organização em Registros

- Forçar todos os registros para um tamanho fixo
- Forçar todos os registros para conterem um número fixo de campos
- Começar cada registro com um indicador de tamanho
- Usar índice para manter informação de endereçamento
- Colocar delimitadores entre registros



Registros de Tamanho Fixo

- Todos os registros têm o mesmo número de bytes
- Um dos métodos mais comuns de organização de arquivos
- Pode-se ter:
  - registros de tamanho fixo com campos de tamanho fixo
  - registros de tamanho fixo com campos de tamanho variável

Registros de Tamanho Fixo

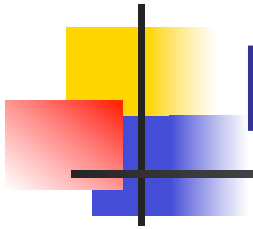
Registro de tamanho fixo e campos de tamanho fixo:

Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Registro de tamanho fixo e campos de tamanho variável:

Maria|Rua 1|123|São Carlos| ← Espaço vazio →|
João|Rua A|255|Rio Claro| ← Espaço vazio →|
Pedro|Rua 10|56|Rib. Preto| ← Espaço vazio →|

Registros com Número Fixo de Campos

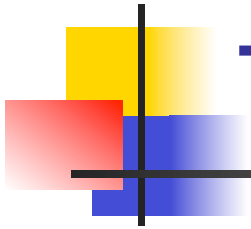


- Cada registro contém um número fixo de campos
  - o tamanho do registro, em bytes, é variável
- Necessidade de ser usado em conjunto com outro método, como delimitadores

Registro com número fixo de campos:

Maria|Rua 1|123|São Carlos|João|Rua A|255|Rio Claro|Pedro|Rua 10|56|Rib. Preto|

Indicador de Tamanho para Registros

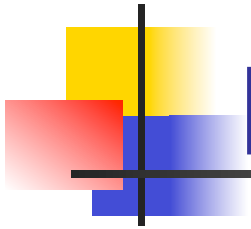


- O indicador que precede o registro fornece o seu tamanho total, em bytes
- Os campos são separados internamente por delimitadores.

Registro iniciados por indicador de tamanho:

28Maria|Rua 1|123|São Carlos|25João|Rua A|255|Rio Claro|27Pedro|Rua 10|56|Rib. Preto|

Método muito utilizado para manipular registros de tamanho variável

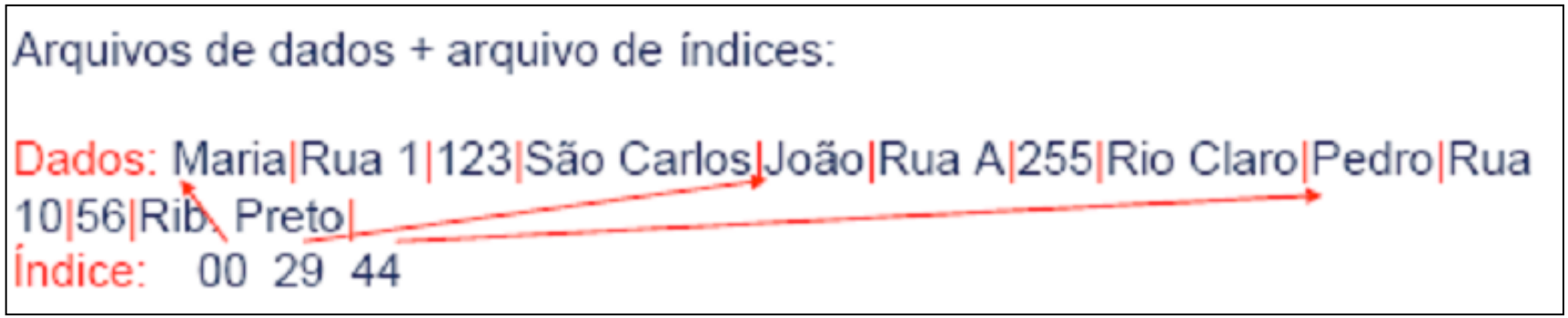


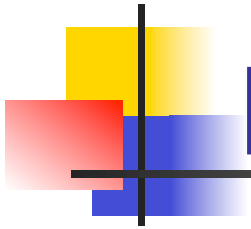
Usar um Índice para Endereçamento

- Arquivo secundário que mantém informações sobre o endereço do primeiro *byte* de cada registro
  - indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo
- *Byte offset*
  - permite encontrar o começo de cada registro
  - permite calcular o tamanho dos registros

Usar um Índice para Endereçamento

- Necessidade de ser usado em conjunto com outro método, como delimitadores

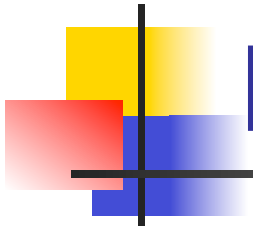




Usar um Índice para Endereçamento

- Vantagem
  - flexibilidade
- Desvantagens
  - necessidade de acessar dois arquivos diferentes
  - manutenção da consistência do índice

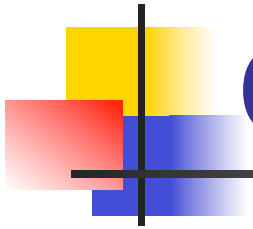
Registros Separados por Delimitadores



- Separar os registros com delimitadores análogos aos de fim de campo
  - o delimitador de campos é mantido, sendo que o método combina os dois delimitadores

Registro delimitado por marcador (#):

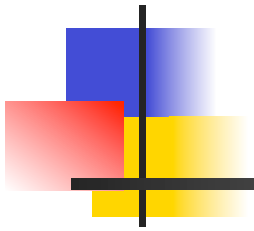
Maria|Rua 1|123|São Carlos|#João|Rua A|255|Rio Claro|#Pedro|Rua 10|56|Rib. Preto|



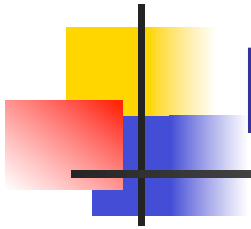
Observações

- Nenhum dos métodos descritos é apropriado para todas as situações
- Escolha do método depende
  - da natureza dos dados
  - para o que eles serão usados

Ver programas em C no livro texto
que ilustram a criação de arquivos
com as diferentes organizações estudadas!



Acesso a Arquivos

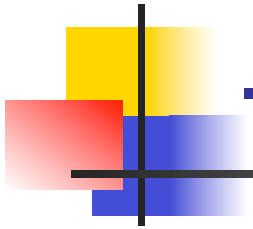


Formas de Acesso

- Busca sequencial
  - lê o arquivo registro a registro
- Acesso direto
  - realiza um *seek* direto para o início do registro desejado e lê o registro imediatamente

Desempenho da pesquisa (busca) em disco:
número de acessos a disco

Custa da Busca Sequencial - Pior Caso -

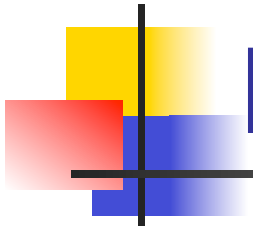


- Custo sem considerar páginas de disco

$C_{busca\_sequencial} = n$
onde n é o tamanho do arquivo

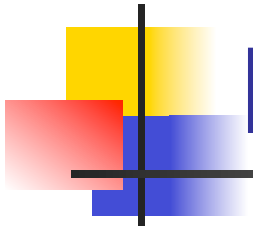
- Custo considerando páginas de disco

$C_{busca\_sequencial} = b$
onde b é o número de páginas de disco que contêm os registros



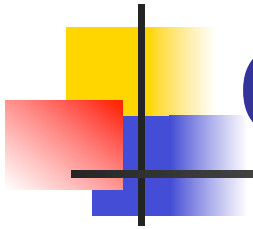
Busca Sequencial

- Vantagens
  - fácil de programar
  - requer estruturas de arquivos simples
  - pode ser aplicada a qualquer arquivo
- Desvantagem
  - pode ser ineficiente



Busca Sequencial

- Exemplos de utilidade
  - busca por uma cadeia específica em um arquivo ASCII
  - busca em arquivos com poucos registros
    - até 10 registros
  - busca em arquivos pouco pesquisados
    - arquivos armazenados em armazenamento terciário

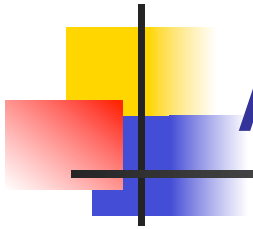


Custo do Acesso Direto

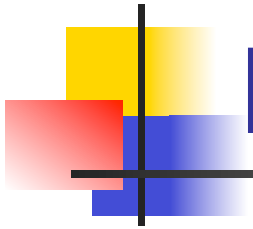
- Um único acesso traz o registro, independentemente do tamanho do arquivo

Cacesso_direto = 1

Formas de Prover o Acesso Direto



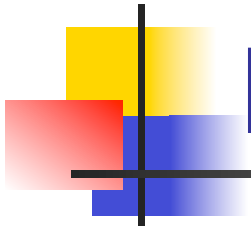
- Uso do RRN (*relative record number*)
  - para registros de tamanho fixo
- Uso de um arquivo índice separado
  - obrigatório para registros de tamanho variável
  - também pode ser utilizado para registros de tamanho fixo



RRN e *Byte Offset*

- RRN
  - fornece a posição relativa do registro dentro do arquivo
  - permite calcular o byte offset
- *Byte Offset*
  - permite encontrar a posição de início do registro

byte offset = RRN x tamanho do registro

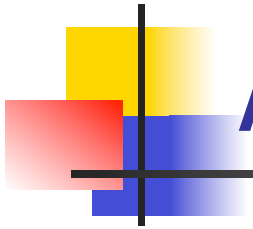


Exemplo

- Campos
  - char nome[15];
  - char rua[10];
  - char numero[5];
  - char cidade[10];
- Características
  - registros de tamanho fixo
  - tamanho: 40

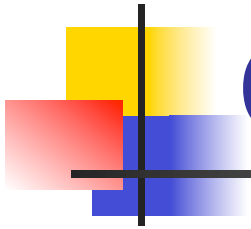
					RRN	byte offset
Maria	Rua 1	123	São Carlos	←	0	0
João	Rua A	255	Rio Claro	←	1	40
Pedro	Rua 10	56	Rib. Preto	←	2	80

Organização de Arquivos e Acesso a Arquivos



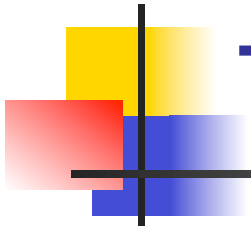
- Organização de arquivos
  - registros de tamanho fixo
  - registros de tamanho variável
- Acesso a arquivos
  - busca sequencial
  - acesso direto

Como Escolher a Organização de um Arquivo



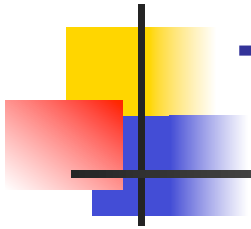
- Análises
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?
  - registros têm tamanho fixo ou variável?
  - como separar os registros?
  - como identificar o espaço utilizado e o "lixo"?
- Escolha
  - depende, entre outras coisas, do que dados vão ser armazenados no arquivo

Escolha: Registros de Tamanho Variável

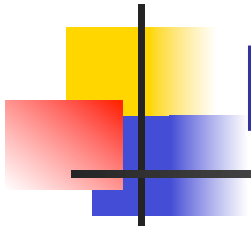


- Principal resultado da análise
  - registros devem ter tamanhos muito diferentes
- Problemas
  - Como acessar esses registros diretamente?
  - Como manter o acesso direto a esses registros eficientemente?
  - Como superar as limitações das linguagens de programação?

Escolha: Registros de Tamanho Fixo



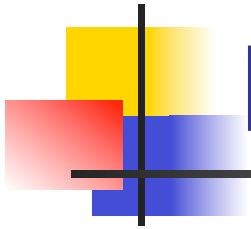
- Vantagem
  - RRN pode ser usado para prover acesso direto aos registros
- Problema
  - tamanho dos registros
    - dependente do tamanho dos campos



Tamanho dos Registros: Exemplo com Decisão Fácil

- Arquivo de controle de vendas
  - número do comprador: 8 bytes
  - data no formato DDMMAA: 6 bytes
  - número do item: 4 bytes
  - quantidade vendida: 4 bytes
  - valor da venda: 8 bytes
- Campos de tamanho fixo: 30 bytes

Tamanho dos Registros: Exemplo com Decisão Fácil

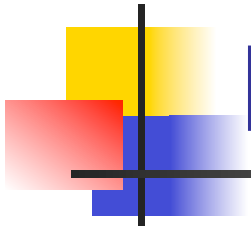


- Registros de 30 bytes
  - páginas de disco de 4KB (4.096 bytes)
  - número de registros por página: 136,53
- Registros de 32 bytes
  - páginas de disco de 4KB (4.096 bytes)
  - número de registros por página: 128

escolhido

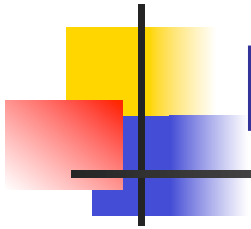
tamanho do registro deve se encaixar no tamanho da página de disco

Tamanho dos Registros: Exemplo com Decisão Difícil



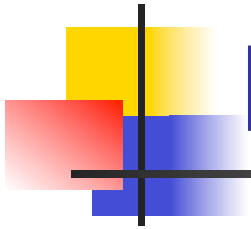
- Campos com tamanho muito variável
  - nome
  - endereço
- Abordagens simplistas para o tamanho do registro
  1. soma do tamanho máximo de cada campo
  2. soma do tamanho mínimo de cada campo

Tamanho dos Registros: Exemplo com Decisão Difícil



- Todos os campos de tamanho fixo
  - simplicidade
  - *porém* problemas das opções 1 e 2
- Todos os campos de tamanho variável
  - pode-se aplicar o efeito do tamanho médio
    - *nomes mais longos em geral não aparecem no mesmo registros que os endereços mais longos*
  - *porém* mais dificuldade na pesquisa por campos que não sofrem muita variação

Tamanho dos Registros: Exemplo com Decisão Difícil



- Decisão interessante
  - campos cujos dados possuem pouca variabilidade: campos de tamanho fixo
  - campos cujos dados possuem grande variabilidade: campos de tamanho variável

usar uma organização em campos adequada às características dos dados armazenados no arquivo