

Campos e Registros: Aulas Práticas

6897/9895 – Organização e Recuperação de Dados

Profa. Valéria D. Feltrim

UEM – CTC – DIN

Slides preparados com base no Cap. 4 do livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992

Escrita de um arquivo contendo uma sequência de campos delimitados por “|”

Funções stdio.h:

- printf (...)
- fopen (arq, mod)
- fputs (str, arq)
- fclose (arq)

Função stdlib.h

- exit(1)

```
PROGRAM: writstrm
get output file name and open it with the logical name OUTFILE
get LAST name as input
while length(LAST) > 0 do
    get FIRST name, ADDRESS, CITY, STATE and ZIP as input
    write LAST to the file OUTFILE
    write "|" to the file OUTFILE
    write FIRST to the file OUTFILE
    write "|" to the file OUTFILE
    write ADDRESS to the file OUTFILE
    write "|" to the file OUTFILE
    write CITY to the file OUTFILE
    write "|" to the file OUTFILE
    write STATE to the file OUTFILE
    write "|" to the file OUTFILE
    write ZIP to the file OUTFILE
    write "|" to the file OUTFILE
    get LAST name as input
end /* while */
close OUTFILE
end PROGRAM
```

Funções de leitura

■ **scanf (const char * format, ...)**

- Lê dados do stdin (entrada padrão) e os armazena, de acordo com o parâmetro *format*, nas localizações apontadas pelos parâmetros adicionais
- A função vai **ler e ignorar quaisquer caracteres do tipo ‘espaço’** encontrados antes do primeiro caractere ‘não-espaço’
 - São considerados ‘espaço’: espaços em branco, quebras de linha (<Enter>) e tabulações
 - São exceções os formatos **%c** e **%[...]**
- Formato **%s**
 - Lê **qualquer número de caracteres não-espaço**, parando no primeiro caractere ‘espaço’ encontrado. O caractere nulo ('\0') é automaticamente adicionado no final da sequência armazenada

```
char nome[20];  
scanf ("%[^\\n]s", nome);
```

```
char nome[20];  
scanf ("%s", nome);
```

! Não verifica se o tamanho da *string* lida é compatível com a *string* alocada !

Funções de leitura

- A lógica do programa writstrm (slide anterior) não funciona se a leitura dos campos for feita com **scanf()**
 - A chamada **scanf("%s", sobrenome)** vai ignorar caracteres **‘espaço’**, o que inclui o <Enter> (`‘\n’`)
 - A função vai esperar que pelo menos um caracter diferente de <Enter> seja digitado para o **sobrenome**
 - Assim, o comprimento da *string* **sobrenome** nunca será zero

Funções de leitura

■ **fgets (char * str, int size, FILE * stream)**

- Lê caracteres de **stream** e os armazena em **str** até que os **size-1** caracteres sejam lidos ou até que uma quebra de linha ('\n' ou <Enter>) seja encontrada ou o fim do arquivo, o que ocorrer primeiro.

```
char nome[20];  
fgets(nome, 20, stdin);
```

- Um caractere de quebra de linha faz com que a função **fgets** pare de ler, mas é considerado um caractere válido pela função e incluído na **string** copiada para **str**
- Um caractere nulo ('\0') é automaticamente anexado após os caracteres copiados para **str**

! Size deve ser compatível com o tamanho da **string alocada !**

Funções de leitura

- Função que lê uma *string* ***str***, desconsidera o '\n', limpa o ***stdin*** e retorna o tamanho da *string* lida

```
int input(char * str, int size) {  
    int i = 0;  
    char c = getchar();  
    while (c != '\n') {  
        if (size > 0) {  
            str[i] = c;  
            i++;  
            size--;  
        }  
        c = getchar();  
    }  
    str[i] = '\0';  
    return i;  
}
```

Leitura de um arquivo contendo uma sequência de campos delimitados por “|”

Funções stdio.h:

- printf (...)
- fopen (arq, mod)
- fclose (arq)
- fgetc (arq)

Função stdlib.h

- exit(1)

```

PROGRAM: readstrm

get input file name and open it with the logical name INFILE
FIELD_LENGTH := readfield(INFILE, STR, SIZE)
while FIELD_LENGTH > 0 do
    write STR to the screen
    FIELD_LENGTH := readfield(INFILE, STR, SIZE)
end /* while */
close INFILE
end PROGRAM

FUNCTION: readfield (INFILE, STR, SIZE)
initialize I to 0
read a character from INFILE into CH
while CH does not equal EOF and CH does not equal '|' do
    if SIZE > 0 then
        append CH to STR
        increment I
        decrement SIZE
    end /* if */
    read a character from INFILE into CH
end /* while */
append '\0' to STR    /* finalização de string em C*/
return I             /* I armazena o comprimento do campo lido */
end FUNCTION

```

Escrita de um arquivo
contendo registros
de tamanho variável
com indicação de
tamanho no início
do registro e
campos delimitados
por “|”

Funções string.h

- strcat (str1, str2)
- strlen (str)

Funções stdio.h:

- printf (...)
- fopen (arq, mod)
- fclose (arq)
- fwrite (buffer, tam_elem, qtd_elem, arq)

Pseudo writrec

```
PROGRAM: writrec  
  
get output file name and open it with the logical name OUTFILE  
get LAST name as input  
while length(LAST) > 0 do  
    set length of string in BUFFER to zero /* BUFFER[0] = '\0' */  
    concatenate BUFFER + LAST + "|"   
    for each field of record do  
        get FIELD as input  
        concatenate BUFFER + FIELD + "|"   
    compute REC_LENGTH as the length of the string in BUFFER  
    write REC_LENGTH to the file OUTFILE /* with fwrite */  
    write the string in BUFFER to the file OUTFILE  
    get LAST name as input  
  
close OUTFILE  
  
end PROGRAM
```


Leitura de um arquivo contendo registros de tamanho variável com indicação de tamanho no início do registro e campos delimitados por “|”

Funções **stdio.h**:

- fopen (arq, mod)
- fclose (arq)

Função **string.h**:

- strtok(string_de_busca, string_delimitadora) → na 1ª chamada
- strtok(NULL, string_delimitadora) → nas chamadas subsequentes

```

PROGRAM: readrec

get input file name and open it with the logical name INFILE
REC_LENGTH := get_rec(INFILE, BUFFER, SIZE)
while REC_LENGTH > 0 do
    FIELD := strtok (BUFFER, "|")
    while FIELD does not equal NULL do
        print FIELD on the screen
        FIELD := strtok (NULL, "|")
    end /* while */
    REC_LENGTH := get_rec(INFILE, BUFFER, SIZE)
end /* while */

close INFILE
end PROGRAM

FUNCTION: get_rec (INFILE, BUFFER, SIZE)
read REC_LENGTH from INFILE
if EOF was reached then return 0
if REC_LENGTH < SIZE then
    read the record contents (REC_LENGTH bytes) into BUFFER
    append '\0' to BUFFER
    return REC_LENGTH
else return 0
end FUNCTION

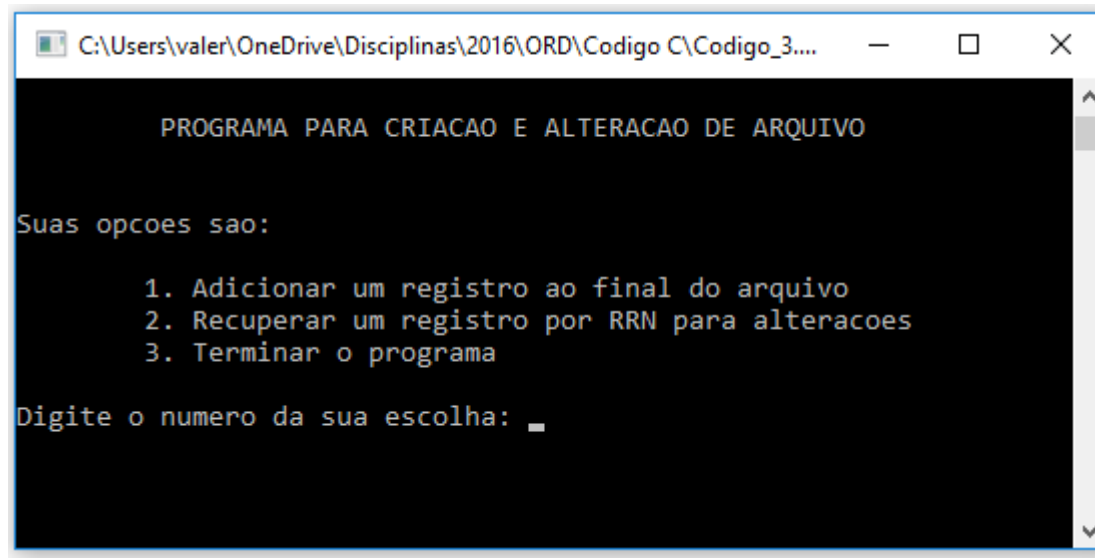
```

Busca sequencial
em um arquivo de
registros de
tamanho variável
no formato
gravado pelo
programa
writerec

Pseudo **finc.c**

```
PROGRAM: find
get input file name and open it with the logical name INFILE
get SEARCH_KEY as input /* SEARCH_KEY is a last name */
set flag MATCHED to false
while not MATCHED and (REC_LENGTH := get_rec(INFILE, BUFFER)) > 0 do
    LAST := strtok(BUFFER, "|")
    if (LAST = SEARCH_KEY) then
        MATCHED := true
    end /* if */
end /* while */
if (MATCHED) then
    print LAST on the screen
    FIELD := strtok(NULL, "|")
    while FIELD does not equal NULL do
        print FIELD on the screen
        FIELD := strtok(NULL, "|")
    end /* while */
end /* if */
close INFILE
end PROGRAM
```

Exercício update.c



```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3....
```

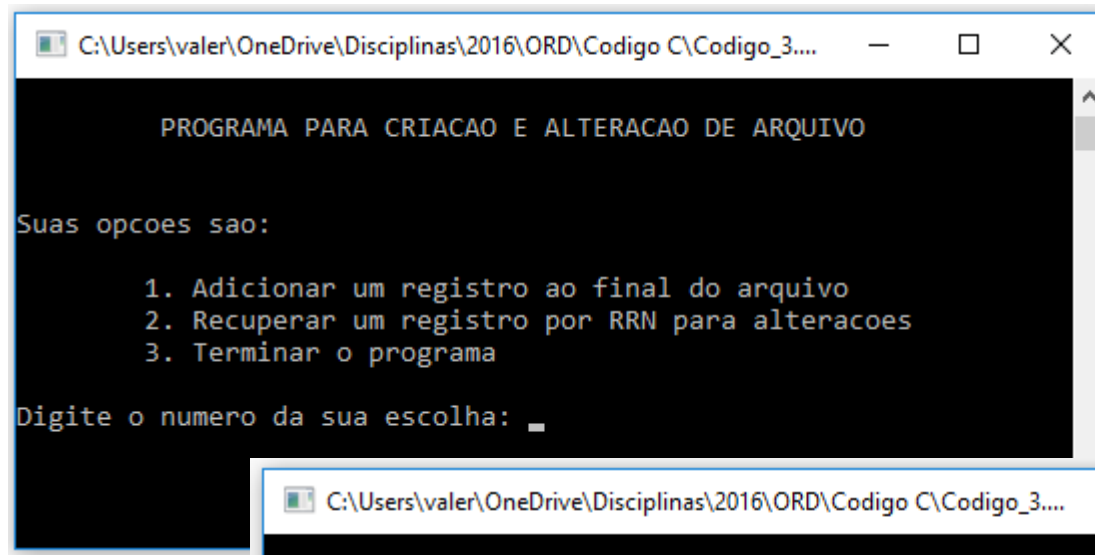
PROGRAMA PARA CRIACAO E ALTERACAO DE ARQUIVO

Suas opcoes sao:

- 1. Adicionar um registro ao final do arquivo
- 2. Recuperar um registro por RRN para alteracoes
- 3. Terminar o programa

Digite o numero da sua escolha: _

Exercício update.c



```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3....
```

```
PROGRAMA PARA CRIACAO E ALTERACAO DE ARQUIVO
```

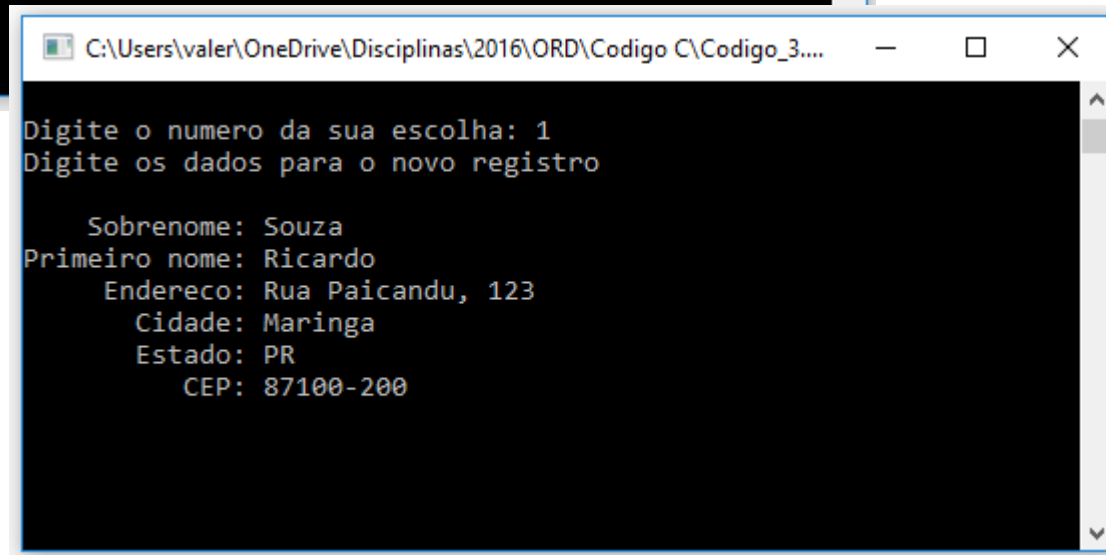
```
Suas opcoes sao:
```

```
1. Adicionar um registro ao final do arquivo
```

```
2. Recuperar um registro por RRN para alteracoes
```

```
3. Terminar o programa
```

```
Digite o numero da sua escolha: _
```



```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3....
```

```
Digite o numero da sua escolha: 1
```

```
Digite os dados para o novo registro
```

```
Sobrenome: Souza
```

```
Primeiro nome: Ricardo
```

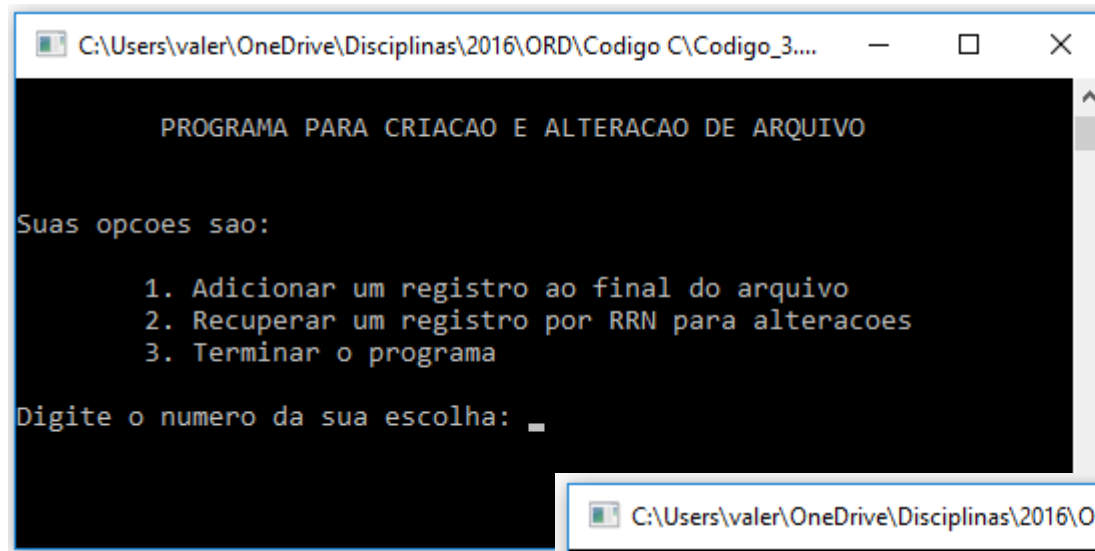
```
Endereco: Rua Paicandu, 123
```

```
Cidade: Maringa
```

```
Estado: PR
```

```
CEP: 87100-200
```

Exercício update.c



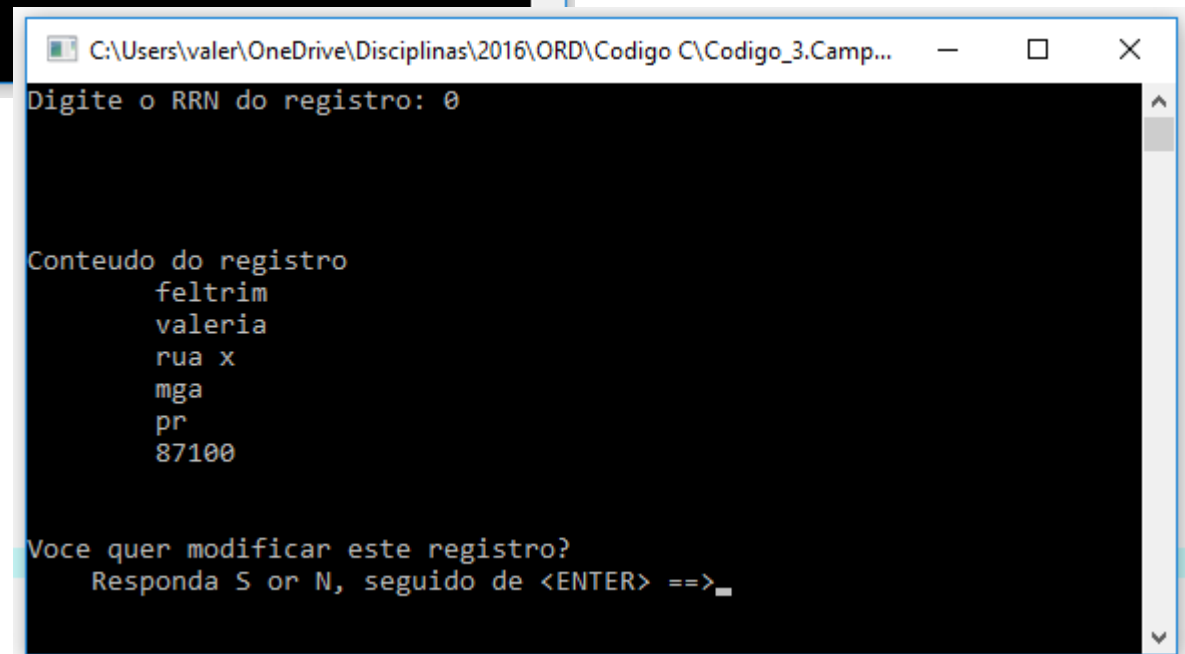
```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3....

PROGRAMA PARA CRIACAO E ALTERACAO DE ARQUIVO

Suas opcoes sao:

    1. Adicionar um registro ao final do arquivo
    2. Recuperar um registro por RRN para alteracoes
    3. Terminar o programa

Digite o numero da sua escolha: _
```



```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3.Camp...

Digite o RRN do registro: 0

Conteudo do registro
    feltrim
    valeria
    rua x
    mga
    pr
    87100

Voce quer modificar este registro?
    Responda S or N, seguido de <ENTER> ==> _
```

Exercício update.c

```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3.Camp...
Digite o RRN do registro: 0

Conteudo do registro
    feltrim
    valeria
    rua x
    mga
    pr
    87100

Voce quer modificar este registro?
Responda S or N, seguido de <ENTER> ==>_
```

```
C:\Users\valer\OneDrive\Disciplinas\2016\ORD\Codigo C\Codigo_3.Camp...

Voce quer modificar este registro?
Responda S or N, seguido de <ENTER> ==>s

Digite os novos dados do registro:

    Sobrenome: _
```

Estrutura e tamanho dos registros

■ Registros de tamanho fixo com campos de tamanho variável

- O registro de tamanho fixo serve como um *container* para campos de tamanho variável
- Diminuição dos problemas de espaço em comparação ao uso de campos de tamanho fixo
 - O tamanho fixo do registro pode ser uma média dos possíveis tamanhos de campo
- É preciso sinalizar de alguma forma onde os dados de cada registro terminam e onde começam as sobras de espaço
 - Qualquer técnica para delimitação de registro vista anteriormente vai funcionar
 - Alternativa → preencher o espaço vazio com algum caracter sinalizador (p.e., '\0' em C)

64 bytes

Silva Alan Rua Tiete 123 Maringa PR 87100
Flores Andre Rua Braga 34 Sarandi PR 87111

Programa que abre ou cria um arquivo de **registros de tamanho fixo**

Os registros podem ser inseridos ou lidos.

Registros a serem lidos são **buscados por RRN**

Use um buffer de tamanho **64+1** → todo reg ocupará 64 bytes, portanto, limpe o buffer antes de cada leitura

Armazene o número total de regs no cabeçalho do arquivo

O cabeçalho deve ser gravado já na criação do arquivo e deve ser lido/atualizado a cada utilização

```
PROGRAMA: update
Leia o nome do arquivo em filename
Abra o arquivo filemane para L/E:
    Se (o arq não existir):
        crie e abra o arquivo para L/E
        faça header.reg_cont = 0 e grave-o no arquivo
    Senão:
        leia o cabeçalho (header) e armazene-o em header.reg_cont
Leia a opção do usuário /*(1)inserir (2)buscar/atualizar (3)sair */
Enquanto (opção < 3) faça
    Caso opção == 1:
        leia os dados do registro /* sobrenome, nome, etc */
        concatene os dados no buffer com os delimitadores
        calcule a posição de gravação //header.reg_cont*64+sizeof(header)
        faça o seek para a posição correta e grave o registro
        incremente header.reg_cont
    Caso opção == 2
        leia o RRN a ser buscado
        se RRN >= header.reg_cont imprima msg de erro e saia do Caso
        calcule a posição de leitura //RRN * 64 + sizeof(header)
        faça o seek, leia o registro para o buffer e mostre na tela
    Leia a opção do usuário
Fim enquanto
Faça seek para o início do arquivo
Grave o cabeçalho
Feche o arquivo
```


Exercício convertetxt.c

- O código ASCII da quebra de linha ('\n') em arquivos texto muda dependendo do S.O.
 - DOS/Windows → par CR-LF (decimal ASCII 13 e 10)
 - Linux → LF (decimal ASCII 10)
- **Faça um programa que converta arquivos texto do Windows para Linux e vice-versa**
 - Leia o nome do arquivo texto a ser convertido e um código que informe o sentido da transformação (Windows → Linux ou Linux → Windows)
 - Abra esse arquivo e grave todo o seu conteúdo em um novo arquivo a ser criado em tempo de execução, modificando o(s) caracter(es) de quebra de linha de acordo com a transformação indicada
 - O char 13 pode ser representado por '\r'
 - Inclua tratamento de erro para os casos do arquivo indicado não existir e do código da transformação ser inválido.