

# Atividade Prática: Campos e Registros

Organização e Recuperação de Dados  
Profa. Valéria

UEM – CTC – DIN

# Programa 1

- Implemente o pseudocódigo *escreve\_campos*
- O programa *escreve\_campos* lê dados de pessoas repetidamente até que se digite um sobrenome “vazio”
  - Sobrenome, Nome, Endereço, Cidade, Estado e CEP
- Os dados de cada pessoa devem ser gravados em um arquivo texto como uma sequência de campos de tamanho variável delimitados por ‘|’

```
Silva|Alan|RuaTiete 123|Maringa|PR|87100|  
Flores|Andre|Rua Braga 34|Sarandi|PR|87111|  
...
```

Escrita de um arquivo texto contendo uma sequência de campos delimitados por '|'

Função **built-in**:

- open (nomearq, 'w')

**Métodos** de um objeto f do tipo arquivo texto:

- f.write(string)
- f.close()

PROGRAMA: **escreve\_campos**

receba o *nome do arquivo* a ser criado na string NOME\_ARQ  
abra o arquivo NOME\_ARQ para escrita com o nome lógico SAIDA  
receba o *sobrenome* na string SOBRENOME

**enquanto** SOBRENOME for diferente de string vazia **faça**

    receba *nome, endereço, cidade, estado e cep* nas strings NOME,  
        ENDERECO, CIDADE, ESTADO e CEP, respectivamente

    escreva a string SOBRENOME no arquivo SAIDA

    escreva a string '|' no arquivo SAIDA

    escreva a string NOME no arquivo SAIDA

    escreva a string '|' no arquivo SAIDA

    escreva a string ENDERECO no arquivo SAIDA

    escreva a string '|' no arquivo SAIDA

    escreva a string CIDADE no arquivo SAIDA

    escreva a string '|' no arquivo SAIDA

    escreva a string ESTADO no arquivo SAIDA

    escreva a string '|' no arquivo SAIDA

    escreva a string CEP no arquivo SAIDA

    escreva a string '|' no arquivo SAIDA

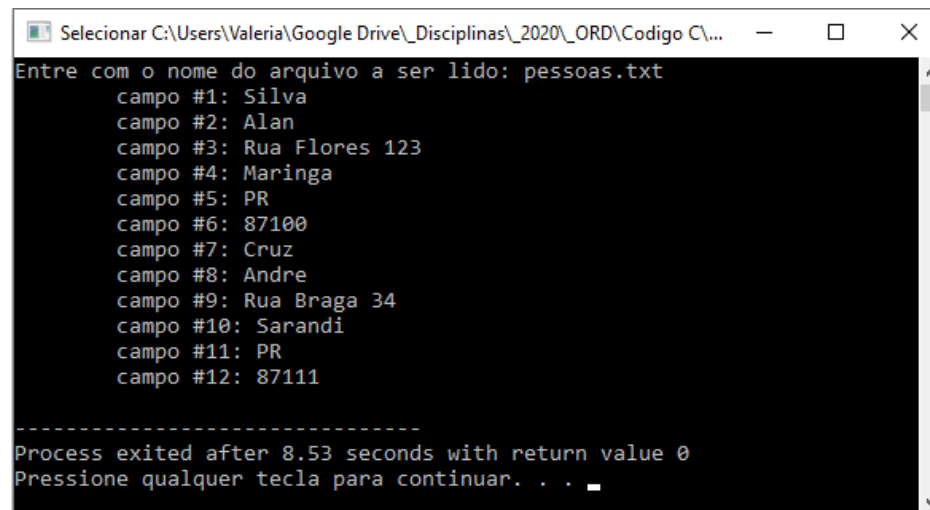
    receba o próximo *sobrenome* na string SOBRENOME

    feche SAIDA

fim PROGRAMA

# Programa 2

- Implemente o pseudocódigo *le\_campos*
- O programa *le\_campos* lê os dados gravados no arquivo texto criado pelo programa *escreve\_campos*
- Os campos devem ser lidos do arquivo um a um e apresentados em tela



```
Selecionar C:\Users\Valeria\Google Drive\Disciplinas\2020\ORD\Codigo C\...
Entre com o nome do arquivo a ser lido: pessoas.txt
campo #1: Silva
campo #2: Alan
campo #3: Rua Flores 123
campo #4: Maringa
campo #5: PR
campo #6: 87100
campo #7: Cruz
campo #8: Andre
campo #9: Rua Braga 34
campo #10: Sarandi
campo #11: PR
campo #12: 87111

-----
Process exited after 8.53 seconds with return value 0
Pressione qualquer tecla para continuar. . . _
```

Leitura de um arquivo texto contendo uma sequência de campos delimitados por '|'

Função **built-in**:

- open (nomearq, 'r')

**Métodos** de um objeto *f* do tipo arquivo texto:

- f.read(qtd\_char)
- f.close()

Comando **try-except** para tratar erros de abertura:

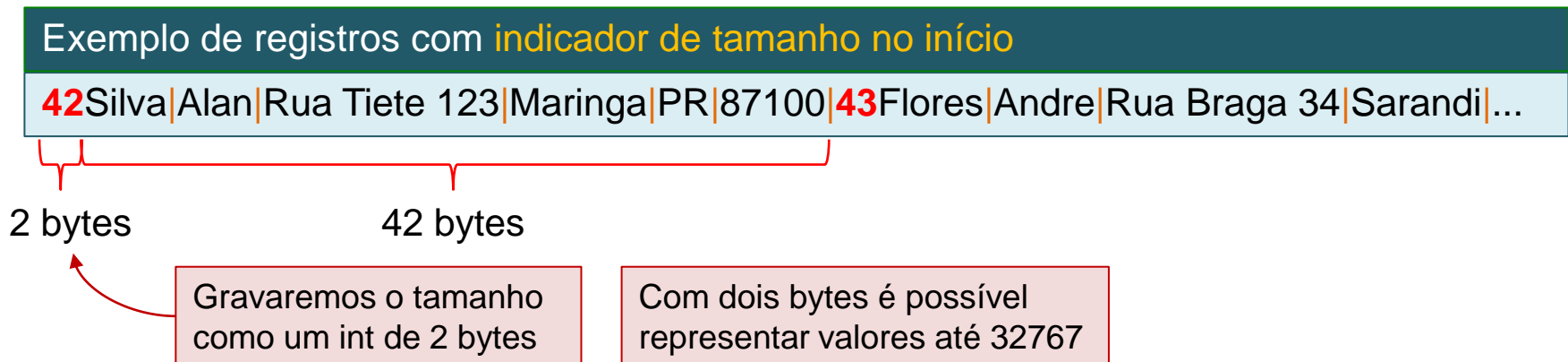
- FileNotFoundError

```
PROGRAMA: le_campos
    receba o nome do arquivo na string NOME_ARQ
    abra o arquivo NOME_ARQ para leitura com o nome lógico ENTRADA
    se o arquivo não foi aberto, termine o programa
    chame a função leia_campo(ENTRADA) e armazene o retorno em CAMPO
    enquanto CAMPO for diferente de string vazia faça
        escreva a string CAMPO na tela
        chame leia_campo(ENTRADA) e armazene o retorno em CAMPO
    feche ENTRADA
fim PROGRAMA

FUNÇÃO: leia_campo(ENTRADA)
    inicialize a string CAMPO como vazia
    leia um caractere de ENTRADA e armazene na string C
    enquanto C diferente de string vazia e C diferente de '|' faça
        concatene a string C na string CAMPO
        leia um caractere de ENTRADA e armazene na string C
    retorne CAMPO
fim FUNÇÃO
```

# Programa 3

- Implemente o pseudocódigo *escreve\_registros*
- Assim como o *escreve\_campos*, o *escreve\_registros* lê dados de pessoas repetidamente até que se digite um sobrenome “vazio”
- Para cada pessoa lida, os dados são gravados em um arquivo binário como um registro de tamanho variável com indicação de tamanho no início do registro e campos delimitados por ‘|’



Escrita de um **arquivo binário** contendo registros de tamanho variável com indicação de tamanho no início do registro

Função **built-in**:

- open (nomearq, mod)

**Métodos** de um objeto f do tipo arquivo binário:

- f.write(var)
- f.close()

**Método** de um objeto s do tipo str:

- s.encode()

**Método** de um objeto i do tipo int:

- s.to\_bytes(numBytes)

## Pseudo escreve\_registros

PROGRAMA: **escreve\_registros**

receba o nome do arquivo na string NOME\_ARQ

abra o arquivo NOME\_ARQ para **escrita binária** com o nome lógico SAIDA

receba o sobrenome na string CAMPO

**enquanto** CAMPO for diferente de string vazia **faça**

    inicialize a string BUFFER como vazia

    concatene as strings CAMPO e '|' em BUFFER

**para** cada campo **faça**

        receba o campo na string CAMPO

        concatene as strings CAMPO e '|' em BUFFER

    converta a string BUFFER para o tipo binário

*# utilize a função encode() da classe str*

    armazene o comprimento string BUFFER no inteiro TAM

    converta TAM para um binário de 2 bytes

*# utilize a função to\_bytes(2) da classe int*

    escreva a variável TAM no arquivo SAIDA

    escreva a variável BUFFER no arquivo SAIDA

    receba o próximo sobrenome na string CAMPO

feche SAIDA

fim PROGRAMA

# Programa 4

- Implemente o pseudocódigo *le\_registros*
- O programa *le\_registros* lê os dados gravados no arquivo criado pelo programa *escreve\_registros*
- Os registros devem ser lidos do arquivo um a um e apresentados em tela

Exemplo de registros com **indicador de tamanho no início**

**42**Silva|Alan|Rua Tiete 123|Maringa|PR|87100|**43**Flores|Andre|Rua Braga 34|Sarandi|...

2 bytes

42 bytes

```
2024/ORD/_Praticas/Pratica 1 - Campos e registros$ python3.12 le_registr
os.py
Digite o nome do arquivo a ser aberto: pessoas.dat

Registro #1 (Tam = 42):
Campo #1: Silva
Campo #2: Alan
Campo #3: Rua Tiete 123
Campo #4: Maringá
Campo #5: PR
Campo #6: 87100

Registro #2 (Tam = 43):
Campo #1: Flores
Campo #2: André
Campo #3: Rua Braga 34
Campo #4: Sarandi
Campo #5: PR
Campo #6: 87111
```



Leitura de um arquivo binário contendo registros de tamanho variável com indicação de tamanho no início do registro

Função **built-in**:

- open (nomearq, mod)

**Métodos** do objeto f do tipo arquivo binário:

- f.read(numbytes)
- f.close()

**Método** do objeto s do tipo str:

- s.split(sep='|')

**Método** da classe int:

- int.from\_bytes(var)

**Método** do objeto b do tipo bytes:

- b.decode()

PROGRAMA: **le\_registros**

```

receba o nome do arquivo na string NOME_ARQ
abra NOME_ARQ para leitura binária com o nome lógico ENTRADA
se o arquivo não foi aberto, termine o programa
chame a função leia_reg(ENTRADA) e armazene o retorno em BUFFER
enquanto BUFFER for diferente de string vazia faça
    converta BUFFER para uma LISTA de campos
        # utilize a função split(sep='|') da classe str
    para cada CAMPO em LISTA faça
        escreva CAMPO na tela
        chame leia_reg(ENTRADA) e armazene o retorno em BUFFER
    feche ENTRADA
fim PROGRAMA

```

FUNÇÃO: **leia\_reg(ENTRADA)**

```

leia 2 bytes do arquivo ENTRADA e armazene em TAM
converta TAM para inteiro
    # utilize a função int.from_bytes(TAM)
se TAM > 0 então
    leia TAM bytes do arquivo ENTRADA e armazene em BUFFER
    converta BUFFER para o tipo string
        # utilize a função decode() da classe bytes
    retorne BUFFER
senão retorne ''
fim FUNÇÃO

```