



DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA


Dicas para o projecto 1






IAED, 2016/2017

Processamento de mensagens

- O programa irá permitir o registo e processamento de mensagens.



Comando	Descrição
 A	Adiciona uma mensagem
 L	Lista todas as mensagens
U	Lista todas as mensagens introduzidas por um dado utilizador.
O	Lista as mensagens mais longa
T	Escreve o utilizador mais activo na escrita de mensagens.
S	Lista todas as mensagens por ordem alfabética.
C	Escreve o número de ocorrências de uma palavra dada.
 X	Termina o programa

Interface

- Vamos supor que queremos fazer um programa com 3 comandos — A, B e X, sendo que X termina o programa

```
int main()
{
    char command;
    while (1) {
        command = getchar(); /* le o comando */
        switch (command) {
            case 'A':
                executa_A(); /* Chama a funcao responsavel pela execucao do comando A */
                break;
            case 'B':
                executa_B(); /* Chama a funcao responsavel pela execucao do comando B */
                getchar(); /* Se em executa_B NAO lemos o \n, temos de o ler aqui. */
                break;
            case 'X':
                executa_X(); /* Faz o que tem de fazer antes de sair */
                return EXIT_SUCCESS; /* Termina o programa com sucesso (STDLIB) */
            default:
                printf("ERRO: Comando desconhecido\n");
        }
    }
    return EXIT_FAILURE; /* se chegou aqui algo correu mal (STDLIB) */
}
```

Exemplo **A** (adicionar, s/output)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

L

X

leitura de um inteiro
entre 0 e 999

```
void LeLinha(char s[])  
{  
    ...  
}
```

A user frase

The diagram consists of three arrows originating from the text 'A user frase'. One arrow points to the first input line 'A 123 Mensagem de teste'. A second arrow points to the first input line 'A 123 Outra mensagem de teste'. A third arrow points to the function signature 'void LeLinha(char s[])' in the code block below.

Exemplo **A** (adicionar, s/output)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

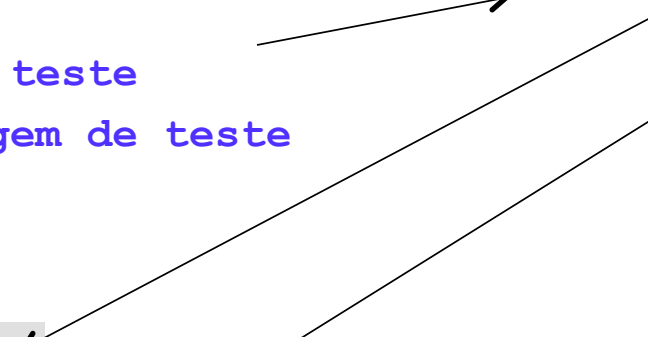
L

X

leitura de um inteiro
entre 0 e 999

```
int LeLinha(char s[])  
{  
    ...  
}
```

A user frase



Exemplo **A** (adicionar, s/output)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

L

X

A user frase



Estrutura?

id

text

Exemplo **A** (adicionar, s/output)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

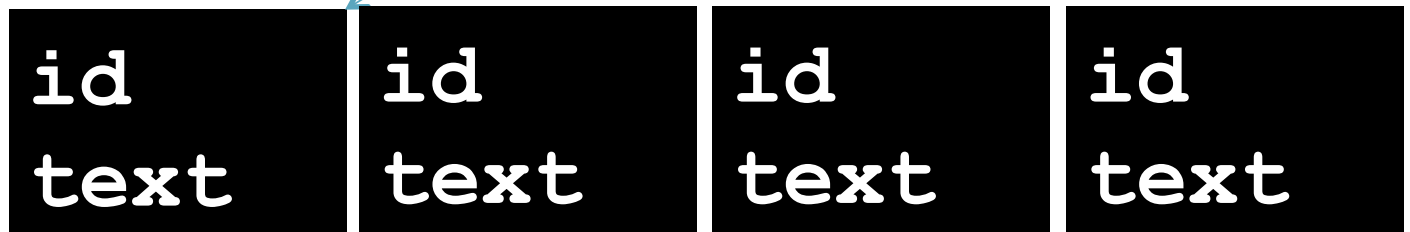
L

X

A user frase

Estrutura?

Vector
de estruturas?
MAX->10000



Count

Exemplo **A** (adicionar, s/output)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

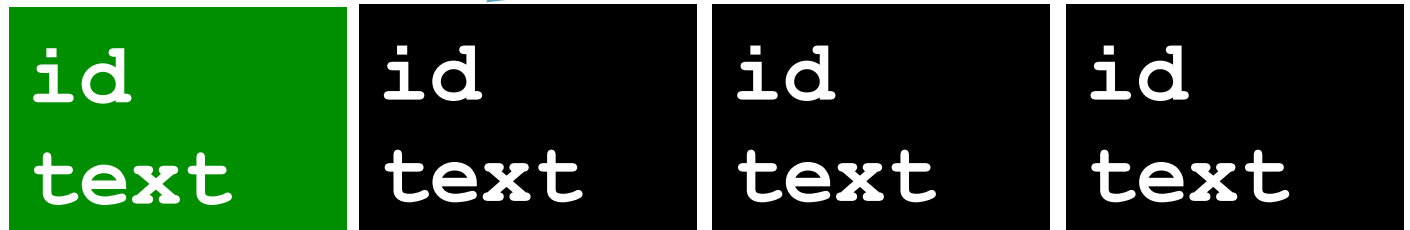
L

X

A user frase

Estrutura?

Vector
de estruturas?
MAX->10000



Count

Exemplo **A** (adicionar, s/output)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

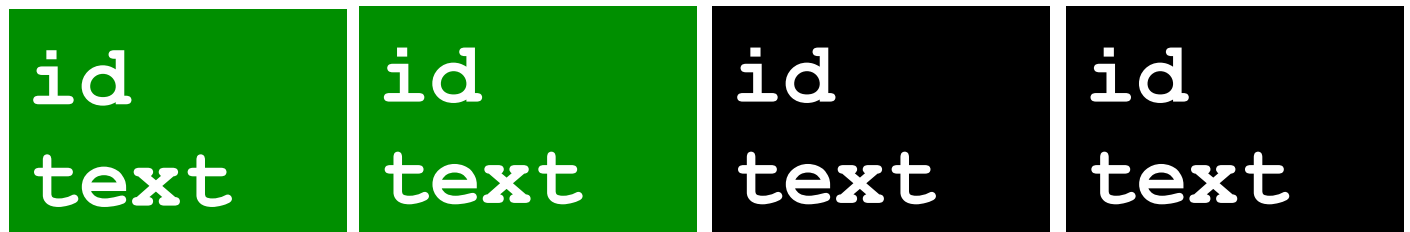
L

X

A user frase

Estrutura?

Vector
de estruturas?
MAX->10000



Count

Exemplo *L* (Lista)

- Input:

A 123 Mensagem de teste

L

A 123 Outra mensagem de teste

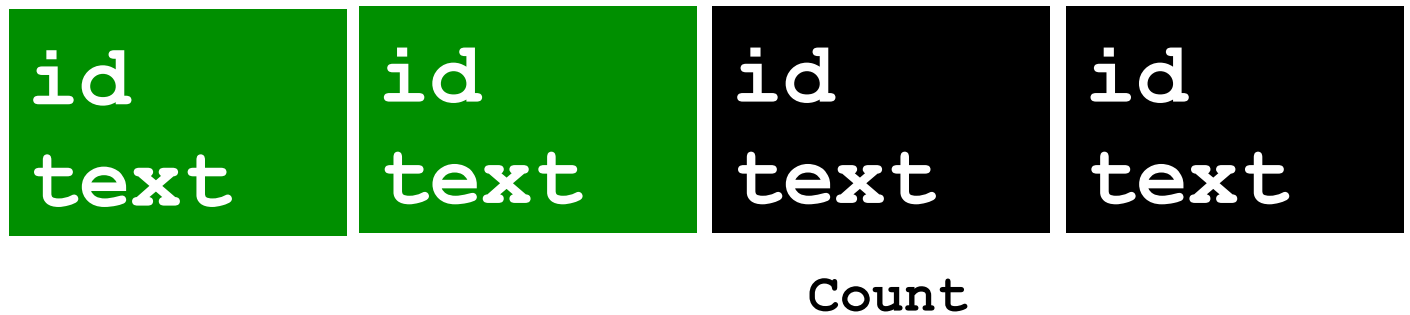
A 567 Ainda outra mensagem de teste

L

X

Output:

```
*TOTAL MESSAGES:1
123:Mensagem de teste
*TOTAL MESSAGES:3
123:Mensagem de teste
123:Outra mensagem de teste
567:Ainda outra mensagem de teste
3
```



Exemplo **x** (Sai)

- Input:

```
A 123 Mensagem de teste
L
A 123 Outra mensagem de teste
A 567 Ainda outra mensagem de teste
L
```

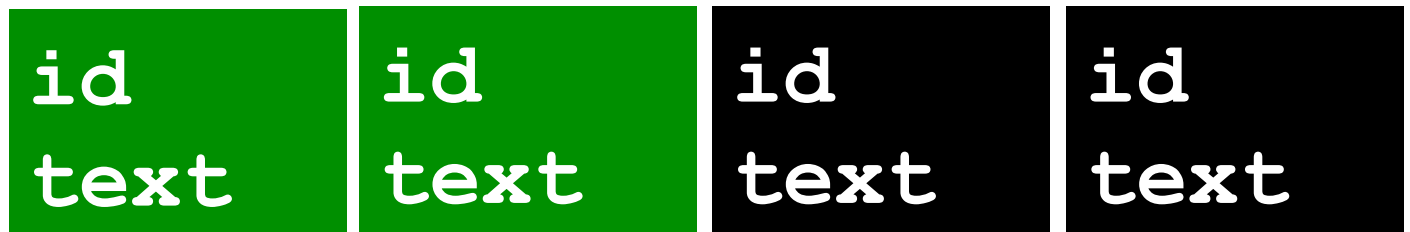
x

Nº de mensagens

Output:

```
*TOTAL MESSAGES:1
123:Mensagem de teste
*TOTAL MESSAGES:3
123:Mensagem de teste
123:Outra mensagem de teste
567:Ainda outra mensagem de teste
```

3



Count

Exemplo *U* (Lista)

- Input:

A 123 Mensagem de teste

A 567 Ainda outra mensagem de teste

U 123

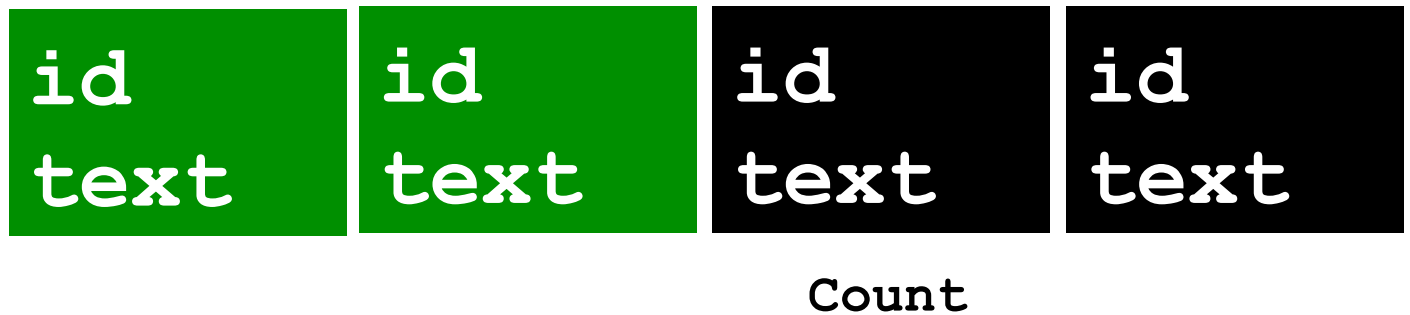
x

Output:

*MESSAGES FROM USER:123

Mensagem de teste

2



Exemplo ○ (Linha mais longa)

- Input:

A 123 Mensagem de teste

A 567 Ainda outra mensagem de teste

○

x

Output:

```
*LONGEST SENTENCE:567:Ainda outra mensagem de teste
2
```

```
*LONGEST SENTENCE:<user>:<frase>
```

id	id	id	id
text	text	text	text

Exemplo ○ (Linha mais longa)

- Input:

A 123 Mensagem de teste

A 567 Ainda outra mensagem de teste

○

x

Output:

```
*LONGEST SENTENCE:567:Ainda outra mensagem de teste
2
```

Nota: caso tenha mais do que uma entrada, deverá listar todas as entradas com igual comprimento, pela ordem que foram introduzidas

id text	id text	id text	id text
------------	------------	------------	------------

Exemplo *T* (user mais activo)

- Input:

A 123 Mensagem de teste

A 123 Outra mensagem de teste

A 567 Ainda outra mensagem de teste

T

X

Output:

```
*MOST ACTIVE USER:123:22  
3
```

```
*MOST ACTIVE USER:<user>:<N_frases>
```

Nota: caso tenha mais do um deverá listar esses elementos por ordem crescente do id...



Esta estrutura de dados é capaz de não chegar... Podem tirar partido do facto dos ids terem índices limitados entre 0 e 999. Mais não digo...

Exemplo *s* (sort)

- Input:

```
A 123 Mensagem de teste
A 123 Outra mensagem de teste
A 567 Ainda outra mensagem de teste
```

S

X

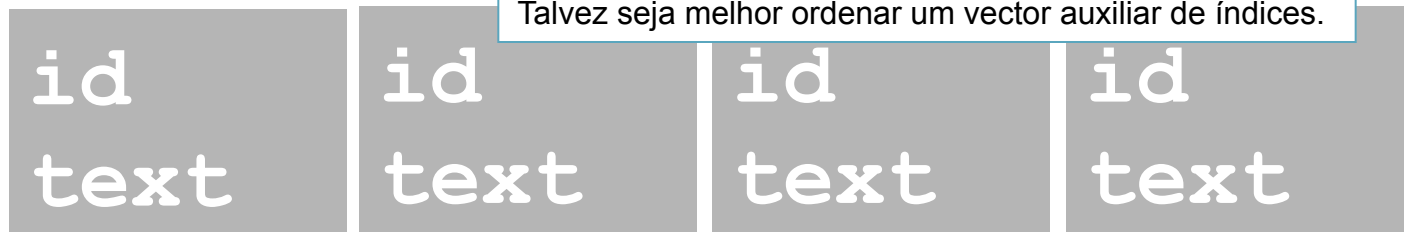
Output:

```
*SORTED MESSAGES:3
567:Ainda outra mensagem de teste
123:Mensagem de teste
123:Outra mensagem de teste
3
```

Sugestão 1: Ver função strcmp (disponível em string.h)...

Sugestão 2: ver enunciado para critérios de desempate...

Sugestão 3: Evitem alterar a vossa estrutura de dados...
Talvez seja melhor ordenar um vector auxiliar de índices.



Exemplo **C** (conta palavras)

- Input:

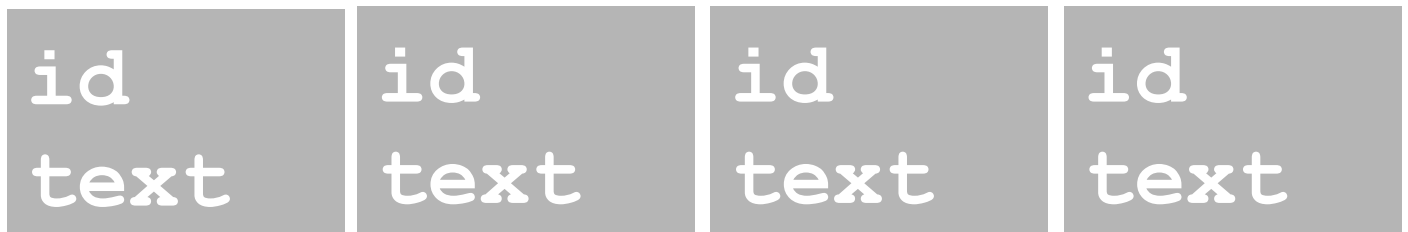
A 123 teste Mensagem de teste
A 123 Outra mensagem de teste teste
A 567 Ainda outra teste mensagem de testeteste

C teste

X

Output:

```
*WORD teste:4  
3
```





Pensar na eficiência da solução escolhida

- A eficiência vai ser avaliada
(componente avaliação automática & discussão)...
- Complexidade do comando A?
- Complexidade do comando C?
- Complexidade do comando S?

Não esquecer!! 😊

- Procure usar abstrações
- Idente e documente convenientemente o seu código.
- Teste o seu código à medida que o escreve.
- Submeta com antecedência.
- Inscrições dos grupos acabam quarta-feira dia 19 de Abril às 13h!!

Contem connosco!!!!



Francisco João Duarte Cordeiro Correia dos Santos
franciscocsantos@tecnico.ulisboa.pt



André Filipe Caldaça da Silva Carvalho
andre.silva.carvalho@tecnico.ulisboa.pt



Andreia Sofia Monteiro Teixeira
sofia.teixeira@tecnico.ulisboa.pt



Diogo Luis Cardoso
diogocardoso@tecnico.ulisboa.pt



Filipe Gouveia
filipe.gouveia@tecnico.ulisboa.pt



Bons códigos!