

CPAS'2010@CIC

FINAL

29 de Abril 2010
Colégio Internato dos Carvalhos

Exercício 1

Implemente um programa que permita a introdução de uma lista de **N** ($1 \leq N \leq 20$) datas no formato **DD MM AAAA** (**DD** - dia, **MM** - mês e **AAAA** - ano) e apresente as datas introduzidas ordenadas em ordem cronológica crescente.

Exemplo 1

Entrada:

```
6
25 06 1965
16 06 1965
13 12 1941
21 04 1965
06 10 1989
01 10 1973
```

Saída:

```
13 12 1941
21 04 1965
16 06 1965
25 06 1965
01 10 1973
06 10 1989
```

Exercício 2

Designa-se por **matriz triangular superior**, uma matriz quadrada em que os elementos localizados abaixo da diagonal principal são nulos (Zero).

Implemente um programa que leia uma matriz **A**, de **M** x **N** ($1 \leq M, N \leq 20$), constituída por valores inteiros ($-100 \leq A_{ij} \leq 100$) e que determine se a mesma é uma matriz triangular superior. Caso a matriz:

- Não seja quadrada ($M \neq N$), o programa emite a mensagem **A MATRIZ NAO E QUADRADA** e termina a execução.
- Seja triangular superior, o programa emite a mensagem **A MATRIZ E TRIANGULAR SUPERIOR**.
- Não seja triangular superior, o programa emite a mensagem **A MATRIZ NAO E TRIANGULAR SUPERIOR**.

Nota: Todas as mensagens são escritas em maiúsculas sem acentos.

Exemplo 1

Entrada:

```
3 3
1 7 -2
0 2 3
0 0 4
```

Saída:

```
A MATRIZ E TRIANGULAR SUPERIOR
```

QUESTÕES SOBRE O FUNCIONAMENTO DO SISTEMA

INPUT/OUTPUT (ENTRADA/SAÍDA)

- ✓ Em todos os problemas o input é feito pelo standard input e o output é feito pelo standard output. Portanto, os vossos programas podem e devem usar as funções "normais" de escrita leitura, como sejam o scanf e printf (em C) ou o read/readln e write/writeln (em Pascal).
- ✓ Os inputs/outputs devem ser exactos, isto é, os dados de entrada e saída devem ser EXACTAMENTE IGUAIS (no conteúdo, na formatação, na sequência, no respeito pelas maiúsculas e minúsculas, no espaçamento, nas mudanças de linha, etc.) aos apresentados no enunciado.
- ✓ O standard input pode ser pensado como uma stream equivalente a qualquer ficheiro. Portanto, quando acaba, tem um end-of-file.
- Em C, quando o final de input é atingido, um fgets retorna NULL, um scanf retorna zero argumentos lidos e um feof(stdin) retorna true. Em FP, quando o final de input é atingido, EOF retorna true. Para simular um end-of-file se estiver a escrever via teclado, pressione Ctrl+D em Linux e Ctrl+Z em Windows.
- ✓ A última linha do output deve incluir mudança de linha (writeln em Pascal ou '\n' em C).

Importante:

- Em C/C++:
- ✓ Não utilizar fflush(stdin), pois a sua utilização torna todas as submissões erradas.
 - ✓ Deve ter o cuidado de colocar a instrução return 0 na função int main () (o que é interpretado pelo Sistema de Submissão como fim do programa).
 - ✓ São permitidas apenas as bibliotecas: **stdio**, **strings**, **math** e **stdlib**.
- Em FP:
- ✓ Não é permitida a inclusão de qualquer biblioteca externa.

SITUAÇÕES OMISSAS OU NÃO REGULAMENTADAS SERÃO DECIDIDAS PELO JÚRI DO CONCURSO. EM TODOS OS CASOS, AS EQUIPAS COMPROMETEM-SE A RESPEITAR A DECISÃO SOBERANA DO JÚRI, NÃO HAVENDO DIREITO A RECURSO.

REGULAMENTO

INÍCIO DE SESSÃO E SOFTWARE

- ✓ Deve efectuar o início de sessão com o utilizador **cpas** e password **cpas123**
- ✓ Todo o software necessário ao desenvolvimento da prova foi previamente instalado, estando os respectivos atalhos disponíveis no ambiente de trabalho.

ADVERTÊNCIAS

- ✓ Não podem consultar quaisquer materiais de apoio, para além dos que forem fornecidos pela organização.
- ✓ Não é permitida a utilização de dispositivos de memória secundária (PenDisks, etc.).
- ✓ Os elementos da equipa podem comunicar apenas entre si, com os cuidados devidos de forma a não perturbarem as restantes equipas.
- ✓ Devem desligar o telemóvel ou outros aparelhos de comunicação.
- ✓ Não podem usar a Internet/Intranet para comunicar, consultar informação ou partilhar ficheiros.
- ✓ Só podem abandonar a sala com a autorização do membro da organização presente na sala.
- ✓ Não podem instalar, eliminar ou alterar o software e a configuração dos postos.

No caso do não cumprimento das disposições anteriores a equipa é automaticamente desclassificada.

RECOMENDAÇÕES

- ✓ Por prevenção, devem guardar as resoluções com alguma regularidade no disco local, numa pasta criada para o efeito, com o nome cpas1 para o exercício 1, cpas2 para o exercício 2, etc.:

DÚVIDAS

- ✓ Em caso de dúvida, devem dirigir-se ao membro da organização presente na sala.

ATRIBUIÇÃO DA CLASSIFICAÇÃO

- ✓ 100 pontos por cada problema resolvido com sucesso (pressupõe que o programa ultrapassou com sucesso a bateria de testes efectuada automaticamente pelo Sistema de Submissão).
- ✓ Cada problema pode ser submetido 5 vezes, no máximo (esgotadas as 5 tentativas a submissão desse problema fica bloqueada).
- ✓ A classificação final é determinada pela soma dos pontos obtidos nas submissões efectuadas. Em caso de empate, aplicam-se os seguintes critérios:
 1. Tempo gasto pela equipa (em minutos) desde a hora de início do concurso até ao instante em que o último problema é submetido com sucesso.
 2. Após a aplicação do critério anterior, se o empate persistir, o júri decide a classificação em função da qualidade global das soluções apresentadas.

Exercício 3

Implemente um programa que leia uma palavra em minúsculas e indique quais as teclas de um teclado de telemóvel que o utilizador teria que pressionar para escrever a palavra pretendida. O programa deve ainda indicar o número de vezes que cada tecla terá de ser pressionada.

1	2 abc	3 def
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz

Notas:

- Considere que a palavra contém apenas as letras: a b c d e f g h i j k l m n o p q r s t u v w x y z
- A palavra terá no máximo 100 letras.

Exemplo 1

Para a palavra **dia** o utilizador terá que pressionar a tecla 3 uma vez (**T3#1**), a tecla 4 três vezes (**T4#3**) e a tecla 2 uma vez (**T2#1**).

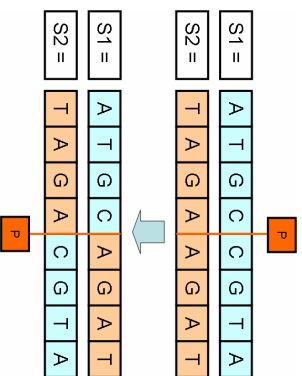
Entrada:
dia

Saída:

T3#1
T4#3
T2#1

Exercício 4

Considere o exemplo abaixo, em que a duas *strings* **S1** e **S2**, com o mesmo número de caracteres, é aplicada uma transformação (troca de conteúdo), no ponto **P**.



Implemente um programa que leia as duas *strings* (**S1** e **S2**), o ponto **P**, efectue a troca de conteúdos (conforme exemplificado) e apresente as *strings* obtidas.

Notas:

- As *strings* lidas terão, no máximo, 200 caracteres.
- Considere o ponto de troca **P** um número inteiro no intervalo **[0,201]**, independente do comprimento das *strings* lidas.

Exemplo 1

Entrada:

```
ATGCCGTA
TAGAAGAT
4
```

Saída:

```
ATGCAGAT
TAGACGTA
```

Exemplo 1

Entrada:

```
2 2 2
\X
X
```

Saída:

```
\ \
\ \
\ \
\ \
```

Exemplo 2

Entrada:

```
2 1 2
X
```

Saída:

```
\ /
/ \
```

Exemplo 3

Entrada:

```
4 4 -2
\ \ /
\ \ /
\ \ /
\ \ /
```

Saída:

```
\X
X
```

palavras começadas por 'a' aparecem antes das começadas por 'b', organizam-se os arranjos para que os começados por 1 apareçam antes dos começados por 2 e os começados por 1, e a seguir 1, apareçam antes dos começados por 1, e a seguir 2.

Implemente um programa que, com base num valor **S** fornecido, apresente o *enésimo* arranjo de blocos (na ordem indicada) que perfaça a soma pretendida.

Nota: Não são fornecidos números de ordem superiores ao número de formas de arranjar os blocos.

Exemplo 1

Para **S=4**, o arranjo com o número de ordem **5**, é **2 1 1**.

Entrada:

4 5

Saída:

2 1 1

Exercício 10

Implemente um programa que com base numa imagem ASCII fornecida como entrada e um factor de ampliação, produza uma nova imagem. As imagens são constituídas pelos caracteres 'X', '/', '\', ' ' e espaços. Cada um destes caracteres pode ser ampliado. Exemplo para o dobro e para o triplo:

```
Normal 2 (dobro) 3 (triplo)
X              \ /      \ /
X              // \     // \
X              X      X
\              \     \
\              \     \
/              /     /
/              /     /
```

Uma ampliação positiva, por exemplo 5, significa que um carácter se transforma num de 5x5. Uma ampliação negativa (redução), por exemplo -3, significa que cada bloco de 3x3 se transforma num carácter.

A entrada do programa consiste numa primeira linha em que são fornecidos 3 inteiros referindo-se, respectivamente, às **C** colunas de largura da imagem, às **L** linhas de altura da imagem e, por último, ao factor **F** de ampliação. Seguem-se as **L** linhas de altura da imagem, cada uma com **C** caracteres. Quer a imagem original, quer a imagem resultante não terão mais do que 500 linhasx500 colunas.

Exercício 5

Considere que se pretende encriptar uma linha de texto, escrita em maiúsculas, com, no máximo, 200 caracteres. O método de encriptação é o seguinte:

1. Retiram-se os espaços.
2. Divide-se a linha de texto em blocos de **5** letras.
3. A **1ª** letra de cada bloco é trocada pela seguinte, a **2ª** por duas à frente, a **3ª** por três à frente e, assim, sucessivamente, até à **5ª** letra.

Nota: Considere que a linha de texto contém apenas as letras: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Exemplo 1:

Para a frase lida:

ALEXANDRA VIU AS UVAS

Retiram-se os espaços:

ALEXANDRAVIUASUVAS

Divide-se o texto em blocos de 5 letras (os espaços em branco são apresentados apenas para facilitar a compreensão):

ALEXA NDRAV IUASU VAS

Após a troca:

BNHBFOPUEAJWDWZWCV

Entrada:

ALEXANDRA VIU AS UVAS

Saída:

BNHBFOPUEAJWDWZWCV

Exercício 6

Implemente um programa que leia **N** frases ($1 \leq N \leq 100$) e que mostre no ecrã as frases com palavras duplicadas.

Notas:

- O número de caracteres de cada frase é inferior ou igual a 200;
- O número de frases é inferior ou igual a 100.
- Uma palavra é uma sequência de letras constituída exclusivamente por caracteres alfabéticos (excluem-se os acentuados e os específicos da língua portuguesa).

Exemplo 1

Entrada:

3

Quem casa quer casa!

Concurso de programacao

Dia de sol, dia alegre

Saída:

Quem casa quer casa!

Dia de sol, dia alegre

Exercício 7

Um jogo de palavras cruzadas pode ser representado por uma matriz de **M** linhas por **N** colunas (**M**, **N** pertencentes ao intervalo [1,201]).

Cada elemento da matriz corresponde a uma quadrícula, em que **0** (zero) indica uma quadrícula branca (início de palavra) e **-1** indica uma quadrícula preta.

Implemente um programa que leia as dimensões da matriz **M** e **N**, os respectivos elementos e indique na matriz as posições que são início de palavras horizontais ou verticais, substituindo os zeros pelo número de ordem da palavra numerado da esquerda para a direita e de cima para baixo, considerando que uma palavra tem pelo menos duas letras.

Exemplo:

0	-1	0	-1	-1	0	-1	0
0	0	0	0	-1	0	0	0
0	0	-1	-1	0	0	-1	0
-1	0	0	0	0	-1	0	0
0	0	-1	0	0	0	-1	-1

1	-1	2	-1	-1	3	-1	4
5	6	0	0	-1	7	0	0
8	0	-1	-1	9	0	-1	0
-1	10	0	11	0	-1	12	0
13	0	-1	14	0	0	-1	-1

Exemplo 1

Entrada:

```
5 8
0 -1 0 -1 -1 0 -1 0
0 0 0 0 -1 0 0 0
0 0 -1 -1 0 0 -1 0
-1 0 0 0 0 -1 0 0
0 0 -1 0 0 0 -1 -1
```

Saída:

```
1 -1 2 -1 -1 3 -1 4
5 6 0 0 -1 7 0 0
8 0 -1 -1 9 0 -1 0
-1 10 0 11 0 -1 12 0
13 0 -1 14 0 0 -1 -1
```

Exercício 8

Defina-se uma *palavra dígito* como uma palavra que após remover parte das letras que a constituem ou não remover nenhuma, forma uma das palavras que identificam os algoritmos: **UM**, **DOIS**, **TRES**, **QUATRO**, **CINCO**, **SEIS**, **SETE**, **OTTO** ou **NOVE**.

Por exemplo:

- **PERFUME** e **DEPOIS** são palavras dígitos porque contêm, respectivamente, **UM** e **DOIS**.
- **MUNDO** não é palavra dígito porque embora contenha as letras **U** e **M** não estão pela ordem correcta.

Implemente um programa que leia palavras em maiúsculas (no máximo 15 letras) e determine se cada palavra lida é uma palavra dígito. A introdução da lista de palavras (no máximo 100) termina com #.

Se a palavra não for dígito, a saída deve ser **NAO E PALAVRA DIGITO** (sem acentos). Caso a palavra seja dígito, a saída deve ser o número que contém, sobre a forma de numeral (1,2,3,...). Não serão introduzidas palavras que contenham mais do que um dígito.

Exemplo 1

Entrada:
PERFUME
DEPOIS
MUNDO
#

Saída:

```
1
2
NAO E PALAVRA DIGITO
```

Exercício 9

Considere um conjunto de blocos de brincar com os números de **1 a 9**. Supondo que existem sempre blocos em número suficiente para realizar qualquer arranjo admissível, podem-se organizar os blocos de **N** maneiras diferentes para que a sua soma atinja um dado valor **S** (número inteiro, entre 1 e 15).

Por exemplo, para **S=4**, há 8 formas diferentes de organizar os blocos:

```
1 1 1 1
1 1 2
1 2 1
1 3
2 1 1
2 2
3 1
4
```

Estas formas foram organizadas pela ordem do dicionário. Como num dicionário, onde as palavras começam por 'a' aparecem antes das começadas por 'b' e as