PseudoCódigo e Explicação

O programa lê um arquivo chamado test.txt (também há o test2.txt caso queira outro caso de teste) que contém um dicionário de palavras e frases criptografadas. O objetivo é decifrar essas frases encontrando um mapeamento consistente de letras entre as palavras criptografadas e as palavras do dicionário.

Para cada frase criptografada, o programa:

1. **Divide a frase em palavras**.
2. **Inicia um mapeamento vazio de caracteres**.
3. **Tenta mapear cada palavra criptografada para uma palavra do dicionário** de mesmo tamanho, garantindo que o mapeamento de letras seja consistente em toda a frase.

- Utiliza uma função recursiva para explorar todas as possíveis combinações de mapeamento.
- Verifica se o mapeamento atual é consistente e não viola a regra de substituição um-para-um.

4. **Se um mapeamento válido for encontrado**:

- Decifra a frase aplicando o mapeamento às palavras criptografadas.
- Exibe a frase decifrada.

5. **Se não for possível encontrar um mapeamento consistente**:

- Substitui cada letra das palavras por *.
- Exibe a frase resultante indicando que a decifração falhou.

O programa busca manter a consistência do mapeamento de caracteres em toda a frase, assegurando que cada letra criptografada corresponda sempre à mesma letra decifrada e que nenhuma letra decifrada seja resultado de mais de uma letra criptografada.

Quando uma palavra decifrada não está no dicionário, o algoritmo considera que o mapeamento de letras utilizado para decifrá-la é incorreto. Nesse caso, ele descarta esse mapeamento e continua a procurar outras possíveis correspondências entre as palavras criptografadas e as palavras do dicionário. O

objetivo é encontrar um mapeamento que seja consistente para toda a frase e que resulte em palavras válidas do dicionário.

Se o algoritmo não conseguir encontrar tal mapeamento após explorar todas as possibilidades, significa que a frase não pode ser decifrada com o dicionário disponível. Nesse cenário, todas as letras serão substituídas por '*' ao imprimir a frase, indicando que não foi possível realizar a decifração.

Portanto, se for decifrado e chegar a uma palavra que não existe no dicionário, o algoritmo tratará isso como um caminho inválido e buscará outras opções até encontrar uma solução válida ou concluir que a decifração não é possível.

Pseudocódigo

```
FUNÇÃO decrypt_phrase(palavras, índice, mapeamento):
    SE índice == comprimento(palavras):
        RETORNAR mapeamento


    palavra = palavras[índice]
    PARA cada palavra_dicionario EM dicionario ONDE comprimento(palavra_dicionario) ==
comprimento(palavra):
        mapeamento_estendido = cópia(mapeamento)
        SE is_mapping_consistent(palavra, palavra_dicionario, mapeamento_estendido):
            resultado = decrypt_phrase(palavras, índice + 1, mapeamento_estendido)
            SE resultado NÃO É nulo:
                RETORNAR resultado
    RETORNAR nulo


FUNÇÃO is_mapping_consistent(palavra_criptografada, palavra_dicionario, mapeamento):
    PARA cada par (c_enc, c_dict) EM zip(palavra_criptografada, palavra_dicionario):
        SE c_enc ESTÁ EM mapeamento:
            SE mapeamento[c_enc] ≠ c_dict:
                RETORNAR FALSO
        SENÃO:
            SE c_dict ESTÁ EM valores(mapeamento):
                RETORNAR FALSO
```

```
            mapeamento[c_enc] = c_dict
    RETORNAR VERDADEIRO


ABRIR arquivo "test.txt" PARA leitura
num_palavras = converter primeira linha do arquivo para inteiro
dicionario = lista vazia
PARA i DE 1 ATÉ num_palavras:
    adicionar próxima linha do arquivo (removendo espaços em branco) à lista dicionario
frases = lista vazia
PARA cada linha_restante NO arquivo:
    adicionar linha_restante (removendo espaços em branco) à lista frases
FECHAR arquivo


PARA cada frase EM frases:
    palavras = dividir frase por espaços
    mapeamento = dicionário vazio
    resultado = decrypt_phrase(palavras, 0, mapeamento)
    SE resultado NÃO É nulo:
        frase_decifrada = lista vazia
        PARA cada palavra EM palavras:
        palavra_decifrada = juntar resultado.get(c, '*') PARA cada c EM palavra
        adicionar palavra_decifrada à frase_decifrada
        imprimir frase_decifrada juntando palavras com espaços
    SENÃO:
    frase_decifrada = lista de '*' multiplicado pelo comprimento de cada palavra EM palavras
    imprimir frase_decifrada juntando palavras com espaços
```