Projeto 1, Programação II

```
Digite uma frase: BOA CAÇA E SEMPRE ALERTA

Frase_original = BOA CAÇA E SEMPRE ALERTA

Frase invertida (repetitiva) = ATRELA ERPMES E AÇAC AOB

Frase invertida (recursiva) = ATRELA ERPMES E AÇAC AOB
```

Categoria 1 “Caranguejo”

```
C A A O O O R S A
 H M R S C R I T

Mensagem codificada: CAAOOORSA HMRSCRIT
```

Categoria 2 “Metades”

```
Palavra codificada: 119215989011
```

Categoria 3 “Data”

Trabalho realizado por:

. Gonçalo Miguel Rainho Taborda, nº14065, LEIC
. Henrique Lima Domingos, nº14064, LEIC
. Manuel Correia da Silva Roque, nº14061, LEIC

Professora:

. Valéria Magalhães Pequeno

Datas:

. Relatório-> 07/04/2024
. Código-> 02/04/2024

2- Introdução:

- ➔ No presente relatório, iremos discutir o desenvolvimento do nosso projeto e os aspetos abordados durante o decorrer do mesmo.

- ➔ Inicialmente, adotamos uma abordagem metodológica Top-Down para a elaboração do código. Essa metodologia envolveu a identificação e organização dos requisitos do projeto em tópicos antes de iniciar a implementação do código propriamente dito. Esse procedimento visou garantir que a lógica do código fosse concebida de forma precisa e eficiente. O código foi desenvolvido utilizando a aplicação Visual Studio Code (VSC) e foi implementado na linguagem de programação Python. A escrita do código foi guiada pelas diretrizes estabelecidas no enunciado fornecido pela professora. Dentro das opções disponíveis para cada categoria, optamos por realizar a do "caranguejo", das "metades" e da "data", respetivamente.

- ➔ O desenvolvimento do código foi realizado de acordo com as melhores práticas de programação, buscando sempre a correção e melhoria da lógica implementada.

3- Estrutura de dados

1ª Categoria -> "Caranguejo"

Neste código, foram implementadas duas principais estruturas de dados: strings e funções.

As strings representam sequências de caracteres e são utilizadas para armazenar as frases inseridas pelo utilizador, bem como as frases invertidas resultantes das operações de inversão.

A função "cifra_caranguejo_repetitivo" e "cifra_caranguejo_recursivo" foram desenvolvidas para inverter uma string. Essas funções desempenham um papel fundamental na execução da operação de inversão das frases fornecidas pelo utilizador. Tanto o método repetitivo quanto o recursivo são empregues para processar a string e gerar a sua inversão.

Portanto, no contexto deste código, as estruturas de dados principais são as strings, responsáveis por armazenar as entradas e saídas, e as funções, fundamentais para implementar a lógica de inversão das strings.

2ª categoria -> "Metades"

A função cifra_metades_repetitiva é responsável por decifrar uma mensagem manipulando as suas metades alternadamente, enquanto a função cifra_metades_recursiva faz o mesmo, mas de forma recursiva. A string de entrada fornecida para as funções é armazenada em variáveis como string e frase. As strings são manipuladas para remover espaços em branco e armazenar apenas os caracteres relevantes. Isso é feito através do método split() para dividir a string em palavras e, em seguida, join() para juntá-las novamente em uma string sem espaços. Já as variáveis são utilizadas para armazenar informações temporárias durante a execução do programa. Por exemplo, as variáveis metades, metade1 e metade2 são usadas para armazenar partes específicas da string processada. Outras variáveis, como i e y, são usadas para controlar o fluxo de execução em funções recursivas.

A função cifra_metades_repetitiva itera sobre a string de entrada, removendo os espaços em branco e armazenando o resultado na variável frase. Esta função, separa os caracteres em posições pares e ímpares da string frase, armazenando-os nas variáveis metade1 e metade2, respetivamente. Os caracteres são concatenados em uma nova string metade, alternando entre as partes pares e ímpares. Por fim, imprime as letras que estão nas posições pares e ímpares, respetivamente.

3ª Categoria -> “Data”

O código fornecido consiste em duas funções que codificam uma palavra de acordo com uma tabela de correspondência entre letras e números.

A primeira função, denominada "data_repetitiva", itera sobre cada letra da palavra fornecida, verificando se a letra da mesma, está presente na tabela de correspondência e, se estiver, substitui a letra pelo número correspondente. Caso contrário, se a letra não estiver na tabela, adiciona um marcador de erro à palavra codificada.

A segunda função, chamada "data_recursiva", realiza o mesmo processo de codificação, porém de maneira recursiva, dividindo a palavra em partes menores e codificando-as até que a palavra inteira seja codificada. Após definir as funções, um exemplo de palavra é fornecido ("ALERTA"), e a palavra é codificada utilizando a função "data_repetitiva". O resultado da codificação é então exibido na tela.

Curiosidade: O código é útil para converter palavras em uma forma codificada, facilitando o processamento ou transmissão de dados de forma mais compacta e padronizada.

4 – Algoritmos

1ª Categoria -> “Caranguejo”

Função para inverter uma string de forma recursiva

Função inverter_recursivamente(string):

Se o comprimento da string for 0:

Retornar a própria string

Senão:

Retornar a concatenação da função inverter_recursivamente(chamando a função com a string omitindo o primeiro caractere) e o primeiro caractere da string

Função para inverter uma string de forma repetitiva

Função inverter_repetitivamente(string):

Inicia a variável string_invertida com uma string nula.

Para cada caractere c na string:

Adiciona o caractere c no início da string_invertida

Retorna a string_invertida

2ª Categoria -> “Metades”

Função para dividir a frase em 2 partes e juntar depois de forma repetitiva

Função cifra_metades_repetitiva(recebe string):

// Remove espaços em branco para codificar a frase

frase = Juntar todas as palavras de string sem espaços

// Divide a frase em duas partes

metade1 = Pegar todas as letras nas posições pares de frase

metade2 = Pegar todas as letras nas posições ímpares de frase

// Junta as duas metades em uma única string

metade = Juntar as metades metade1 e metade2 com um espaço entre elas

Retornar metade

2ª Categoria -> “Metades” (continuação)

Função para dividir a frase em 2 partes e juntar depois de forma recursiva

```
Função cifra_metades_recursiva(recebe string, i, y):
    Se i for igual a 0 então
        // Remove espaços em branco para decodificar a frase
        string = Juntar todas as palavras de string sem espaços
    Fim se

    Se i for maior ou igual ao tamanho da string então
        // Indica que é necessário imprimir um espaço
        Imprimir um espaço
        i = 1

        // Verifica se é necessário parar a recursão
        Se y for igual a 1 então
            Retorna uma string vazia
        Fim se
        // Reinicia o contador y para indicar a próxima etapa
        y = 1
    Fim se

    // Imprime o caractere na posição i da string
    Imprimir o caractere na posição i da string

    // Chama recursivamente a função para o próximo caractere
    Retorna cifra_metades_recursiva(string, i + 2, y)
```

3ª Categoria -> “Datas”

Tabela de conversão onde cada letra vai corresponder a um conjunto de dígitos

Função para codificar de acordo com uma tabela de conversão de forma repetitiva

Função cifra_data_repetitiva(palavra):

palavra_codificada = ""

Para cada letra em palavra convertida para maiúscula:

Se a letra estiver na tabela então

Adiciona à palavra_codificada o valor correspondente na tabela para essa letra

Senão:

Adiciona à palavra_codificada um marcador de erro (!)

Retorna palavra_codificada

Função para codificar de acordo com uma tabela de conversão de forma recursiva

Função cifra_data_recursiva(palavra):

Se a palavra for vazia, retorna uma string vazia
Se o tamanho da palavra for igual a 0 então
Retorna uma string vazia

Se a palavra tiver apenas uma letra, retorna a codificação dela
Senão, se o tamanho da palavra for igual a 1 então
Retorna o valor correspondente da tabela para essa única letra, ou ! se não estiver na tabela

Senão:
# Codifica a primeira letra e chama a função recursivamente para o restante da palavra
primeira_letra = primeira letra da palavra
resto_da_palavra = palavra sem a primeira letra

# Codifica a primeira letra e adiciona à codificação do restante da palavra
código_letra = valor correspondente da tabela para primeira_letra, ou ! se não estiver na tabela
Retorna código_letra concatenado com chamada recursiva de cifra_data_recursiva(resto_da_palavra)

“Menu principal”

Enquanto verdadeiro:

Pedir ao usuário para digitar uma frase

Mostrar um menu com três opções:

- Para ler a frase de trás para frente.
- Para dividir as letras da frase em duas partes.
- Para codificar a frase.

Peça ao usuário para escolher uma das opções.

Se a escolha for 1, leia a frase de trás para frente e mostre o resultado.

Se a escolha for 2, divida as letras da frase em duas partes e mostre o resultado.

Se a escolha for 3, codifique a frase e mostre o resultado.

Se a escolha não for válida, mostre uma mensagem de erro.

Pergunte ao usuário se ele deseja continuar.

Se a resposta for não, torne a condição falsa, definindo o fim do loop.

5- Conclusão

➔ Estamos confiantes de que alcançamos os objetivos estabelecidos, conforme as diretrizes definidas. Durante o desenvolvimento do projeto, confrontamo-nos com uma série de desafios distintos, tais como:

➔ Na categoria 1, por exemplo, implementamos com sucesso um algoritmo para inverter frases e escrevê-las de trás para frente, utilizando funções recursivas e repetitivas.

➔ Na categoria 2, o desafio consistia em dividir uma frase ao meio, com as letras pares na linha superior e as ímpares na inferior. Aqui, a aplicação de estruturas de repetição foi fundamental para manipular eficientemente os caracteres da frase.

➔ Na categoria 3, fomos desafiados a criar uma tabela capaz de decodificar datas e gerar frases específicas através de um código. Este desafio exigiu uma compreensão profunda da lógica por trás das operações de busca e manipulação de dados, bem como habilidades avançadas de programação em Python.

➔ O nosso balanço relativamente a este projeto é positivo pois conseguimos gerir bem o tempo que tínhamos para realizar as tarefas que nos foram propostas, com o máximo de eficácia e de concentração possível.

➔ Em resumo, este relatório proporcionou uma visão detalhada do processo de desenvolvimento do projeto, desde a análise inicial dos requisitos até a implementação das funcionalidades específicas de cada categoria. Esperemos que este documento não apenas forneça uma compreensão clara do trabalho realizado, mas também destaque o papel crucial da metodologia adotada, das escolhas feitas durante a realização do código e as principais funcionalidades implementadas no mesmo.