UNIVERSIDADE
LUSÓFONA

LINGUAGEM DE PROGRAMAÇÃO I

RELATÓRIO DO MINI PROJECTO

CifreX

2019/2020

INTRODUÇÃO

****

O presente relatório cientifico tem como objetivo descrever a estrutura e o funcionamento do programa desenvolvido na disciplina de Linguagem de programação I, refletindo os conhecimentos adquiridos nas aulas.

Foi utilizado a linguagem de programação C para codificar o programa e para isso foi necessário recorrer a algumas funções standarts das bibliotecas **<stdio.h>**, **<string.h>**, **<stdlib.h>** e **<ctype.h>** da linguagem.

O principal objetivo do programa é implementar alguns algoritmos de encriptação de textos conversões.

ESTRUTURA DO SOFTWARE

****

Fazendo uma analogia, podemos ver o programa como uma caixa que recebe um texto do utilizador e devolve o mesmo texto só que encriptado. Mas o farto de o programa não implementar um único algoritmo de codificação, faz com que a entrada do programa não seja apenas o texto que vai ser encriptado. Por isso o formato do input para que o programa execute corretamente deve ser: **<op> <n> <texto>**; sendo "op" um só carácter (tipo char) que indica qual o algoritmo de encriptação ou num único caso o encerramento do programa, "n" um numero inteiro (tipo int) que indica o grau de encriptação.

O programa geralmente está em "standby" esperando o input do utilizador (usando um ciclo pseudo-infinito). A partir do momento que ele faz a captura do input como uma cadeia de caracteres, usando a função "**fgets**", podemos dividir a execução em duas partes:

(1) Validação e interpretação do input:

Na validação do input, a primeira coisa que o programa faz é remover eventuais espaços vazios antes o input formatado. Para isso é invocada a função "**remove_space**" que recebe como parâmetro o input bruto e percorre até encontrar o primeiro carácter e com um ciclo substitui o input por uma nova cadeia igual ao input anterior mas sem espaços.

De seguida o programa passa para a interpretação do char "op", que como indica o formato de input, deve ser o primeiro carácter do input. Por isso com um "**if**" é verificado se "op" é igual ao carácter 'q', que se for verdade marca o fim seguro do programa (ou seja, dentro do "**if**" é enviado a mensagem de saída e há o retorno do inteiro zero da função main). Também é verificado se o "op" é apenas um carácter, respeitando o formato de input.

Após confirmar que "op" não corresponde à opção de saída do programa, "op" é usado como parâmetro na invocação da função "op_is_valid". A invocação é feita com auxilio de um

“**`if`**”, sendo a função de carácter booleano, ou seja, tem como retorno um inteiro de valor lógico. A função simplesmente compara o “op” que lhe é passado com os caracteres que correspondem a algum algoritmo de encriptação. Se haver correspondência, a função retorna o valor 1, caso contrário imprime a mensagem de erro e retorna 0. na função “main”, o programa só executa o corpo do “**`if`**”, se o op não for válido, executando uma instrução “break”, ou seja, interrompe a iteração do ciclo principal e volta ao “standby”.

Se o “op” for válido, antes de verificar o inteiro “n”, o programa praticamente extrai o “n” do input. Para isso é invocada a função “get_n”, que recebe o input como parâmetro e tendo em conta que no formato de input padrão, o “n” deve estar entre o primeiro e o segundo espaço vazio do input, a função faz algumas manipulações com esta parte do input e retorna um numero que é suposto corresponder ao inteiro “n”. Após este processo, o programa verifica se o “n” respeita as regras de input, dependendo do valor do “op”. Caso contrário é imprimido um o valor do “n”

De seguida o programa passa a validar a ultima parte do input que é o texto a ser encriptado. Para isso é invocado a função “text_is_valid” que recebe três parâmetros, sendo o input bruto, uma cadeia de caracteres onde será guardada texto e um valor inteiro que marca o inicio do texto no input, visto que a natureza do valor do “n” desejado pelo utilizador, faz com que o índice de inicio do texto no input seja variável. A função, também de carácter booleano, praticamente só verifica o tamanho do texto que não deve exceder 166 caracteres retornando 0 ou 1. Semelhante à verificação do “op”, a função é invocada num “**`if`**” que se for verdade, ou seja, texto não for válido, interrompe a iteração e o programa volta ao “standby”.

(2) Encriptação do texto:

Após as validações supracitadas, segue-se à fase de encriptação na qual o programa, mediante um fluxo “switch”, avalia o valor do “op” e computa qual o algoritmo usar associado à uma função. Abaixo passa a ser explicado como cada algoritmo é implementado pelas funções.

- “substituition_cipher” :

Função que implementa o algoritmo de substituição, recebendo como parâmetro o valor de “n” e o texto a ser codificado. Antes de efetuar as substituições é preciso fazer um *shift* do alfabeto e para isso a função divide o alfabeto em duas partes sendo a posição correspondente ao valor absoluto de “n” o centro da divisão. E dependendo do sinal de “n”, a parte “cortada” é “colada” no fim ou no inicio. Para esse processo, são usados dois ciclos e duas cadeias auxiliares e para a colagem recorre-se às funções “**`strcpy`**” e “**`strcat`**” da biblioteca “**`string.h`**” Depois da criação desta cifra, ocorre propriamente a substituição de caracteres mediante um ciclo que faz a correspondência entre as posições dos caracteres do alfabeto e da cifra. E no fim é imprimido o texto encriptado pelo algoritmo.

- “substituition_cipher_reverse” :

Função que permite descodificar um texto encriptado com ao algoritmo de substituição. Como próprio nome indica, é a função inversa à “substituition_cipher” executando processos idênticos à geração da cifra desta, mas a principal diferença reside na substituição dos caracteres. Enquanto que na “substituition_cipher”, é feita a correspondência entre as posições do alfabeto e substituídas pelos caracteres da cifra, nesta

função é feita a correspondência entre as posições da cifra e substitutas pelos caracteres do alfabeto padrão. E no fim da descodificação é imprimido o texto original.

- "transposition_cipher_or_reverse" :

Função que permite codificar e descodificar um texto usando o algoritmo de transposição. Primeira mente ao função avalia o valor de "n" para determinar as dimensões da matriz a ser usada para a disposição previa dos caracteres do texto a ser codificado e cria-la com espaços vazios. De seguida um ciclo preenche a matriz com os caracteres do texto, usa um ultimo ciclo para gerar o código a partir das posições da matriz utilizada. No fim do processo é imprimido o texto codificado. A vantagem da implementação, como já referida, é o facto de ela ser responsável pela codificação e descodificação de textos usando o mesmo algoritmo.

NOTA: Nos algoritmos de substituição foram considerados apenas os caracteres maiúsculos, de modo que, antes da execução das funções "substituition_cipher" e "substituition_cipher_reverse", é invocada a função "to_uppercase" que recebe o texto como parâmetro e usando funções da biblioteca "**`ctype.h`**" transforma os caracteres minúsculos do texto em maiúsculos.

CONCLUSÃO

****

O processo de desenvolvimento deste trabalho, permitiu consolidar conteúdos estudados na disciplina de linguagens de programação I, pois, permitiu aprimorar alguns conceitos como a manipulação funcional de cadeias de caracteres e manipulação de matrizes e permitiu um contacto mais profundo com as bibliotecas de funções da linguagem C .

Observações finais:

A figura usada na capa do relatório é uma máquina de encriptação de nome enigma que foi muito usado no século XX.