Ramo Estudantil IEEE - UEL

ELIAS JULIÃO DA SILVA NETO (elias.j.s.neto@uel.br)
GUILHERME ALVES DE OLIVEIRA
(guilherme.alves26@uel.br)
LEONARDO FUZIKAWA (leonardo.daiki@uel.br)
PAULO ROGÉRIO (paulo.rogerio.candido@uel.br)

RELATÓRIO FINAL:

CIFRA DE CÉSAR

Londrina
2023

Ramo Estudantil IEEE - UEL

ELIAS JULIÃO DA SILVA NETO
GUILHERME ALVES DE OLIVEIRA
LEONARDO FUZIKAWA
PAULO ROGÉRIO

RELATÓRIO FINAL:

CIFRA DE CÉSAR

Relatório apresentado ao Ramo Estudantil IEEE da Universidade Estadual de Londrina.

Diretor de Projetos: Nathan Andreani Netzel
Gestores de Projetos: Daniel Tresse Dourado, Levi Monteiro dos Santos

Londrina
2023

Contato do Ramo: sb.uel@ieee.org
Institute of Electrical and Electronics Engineers – IEEE
Universidade Estadual de Londrina - UEL • Paraná - Brasil

NETO, Elias Julião da Silva. OLIVEIRA, Guilherme Alves de. FUZIKAWA, Leonardo. ROGÉRIO, Paulo. **Relatório Final**: CIFRA DE CÉSAR. 2023. 19 folhas. Relatório apresentado ao Ramo Estudantil IEEE da Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

O projeto da “Cifra de César”, consiste na construção de um código em linguagem C, utilizando suas bibliotecas e lógica de programação com a finalidade de criptografar e descriptografar mensagens, inseridas por meio de um arquivo ou pela entrada do teclado, com a utilização de uma chave fornecida pelo usuário, salvando os resultados em um arquivo de texto.

Palavras-chave: Cifra de César. Criptografia. Código. Linguagem C. Computação. Programação.

SUMÁRIO

Sumário

1. INTRODUÇÃO

O projeto “Cifra de César” é um dos Projetos Tutoriais vinculados ao processo seletivo do Ramo Estudantil IEEE, sendo considerado um projeto voltado para a área da computação por envolver a criação de um código na Linguagem C de programação.

Neste projeto, foi desenvolvido um programa que possui a capacidade de criptografar ou descriptografar uma mensagem inserida por um usuário, alterando seus caracteres pelo método do deslocamento, de acordo com a senha informada.

O objetivo é utilizar o projeto desenvolvido para o compartilhamento de informações com maior privacidade, criação de enigmas e demonstração de um aspecto da segurança digital, ao passo em que, como desenvolvedores, exploramos, treinamos e aprendemos fatores relevantes relacionados à Linguagem C e programação no geral.

2. RECURSOS UTILIZADOS

2.1 LINGUAGEM C:

A Linguagem C é uma linguagem de programação de propósito geral, isso significa que ela pode ser usada na maioria dos projetos, sua sintaxe é simples, porém potente, além de ser o ponto de partida para muitas outras linguagens, como C++ e Java. Por conta de sua versatilidade e clareza, ela foi a escolhida para a realização desse projeto.

2.2 VISUAL STUDIO CODE:

O Visual Studio Code é um software de edição de códigos fornecido pela Microsoft, disponível para os sistemas operacionais Windows, Linux e Mac. Possui um código-fonte aberto, o que possibilita que todos os usuários contribuam com novas extensões e funcionalidades. Ele foi necessário para a criação, teste e execução do projeto, bem como as ferramentas que disponibiliza.

2.2.1 Extensões:

Foram utilizadas ao todo 6 extensões do Visual Studio Code para que o software interpretasse e analisasse corretamente os dados do códigos, são elas: “C/C++”, “C/C++ Extension Pack”, “CMake Tools”, todas fornecidas pela Microsoft, “C/C++ Compile Run” fornecida por danielpinto8zz6 e “CMake” fornecida por twxs.

3. METODOLOGIA

3.1 Pré-Processamento:

Nessa etapa do processo de programação, declaramos as bibliotecas necessárias para que sejam atribuídos determinados recursos ao programa, garantindo que certos termos e operações sejam realizados. Nos comentários vinculados abaixo são especificadas as funções de cada uma das bibliotecas.

```
#include <stdio.h>  //Entrada e saída padrão.
#include <stdlib.h> //Necessário para função system.
#include <string.h> //Necessário para armazenar um texto em variável.
#include <locale.h> //Para o uso de acentos.
#include <conio.h>  //Necessário para utilizar o getch.

#define MAX_SIZE 100
```

3.2 Estruturas:

Essa etapa é responsável por definir quais os tipos de constantes que podem ser recebidos por determinada variável, para que, em futuras operações, os valores fiquem restritos ao destino correto.

```
enum TipoChar { NUMERO, MAIUSCULO, MINUSCULO, ESPECIAL }; //Especifica o tipo
de caractere.
enum Operacao { DESCRIPTOGRAFAR, CRIPTOGRAFAR };
```

3.3 Protótipos:

Agora, são estabelecidas uma primeira versão das funções que irão desempenhar um papel essencial ao decorrer do código, cada uma com uma determinação específica, denotadas nos comentários.

```
int RetornaTipoChar(unsigned char caracter); //Retorna o tipo do caracter
fornecido.
void CifrarMensagem(char* mensagem, int chave, int criptografar); //Percorre o
vetor de caracteres da mensagem e atualiza os valores dos caracteres.
int ProcessarArquivos(char* nomeArquivoEntrada, char* nomeArquivoSaida, int
chave, int criptografia); //Salva mensagem criptografada em um arquivo.
```

3.4 Função Principal:

Neste momento, se dá início à função “main”, que será o começo do programa de fato, todo a interação com o usuário ocorrerá dentro dela.

Primeiramente, serão definidas algumas variáveis, para melhor aproveitamento do código e armazenamento de informações sobre os arquivos, o menu e os processos que serão necessários para a realização das criptografias e descriptografias.

```
int main() {
    setlocale(LC_ALL, "portuguese"); //para fazer uso de acento e letras
especiais

    char nomeArquivoSaida[MAX_SIZE]; //Declara um variável para armazenar o
nome do arquivo de saída.
    int escolha = 1; //Armazena a opção do menu.
    int operacao = 0; //Armazena inteiro usado como decisão de codificar ou
decodificar.
    int chave = 0; //Armazena o número de casas que todas as letras do
alfabeto devem se deslocar.
```

Então, ajustaremos o programa para definir um número para cada opção de processo que pode ser realizado, habilitando o número 1 para criptografia e 2 para descriptografia. Com a repetição garantida pela função “do... while”, asseguramos que o usuário possa corrigir sua entrada caso insira um valor inválido

```
  do
    {
        system("cls"); //limpar a tela do console no ambiente Windows.

        if (escolha != 1 && escolha != 2) //Condição para imprimir um aviso de
entrada inválida.
            printf("Opção inválida. Por favor, escolha 1 para criptografar ou
2 para descriptografar.\n\n");

        printf("Escolha uma opção:\n");
        printf("1 - Criptografar\n");
        printf("2 - Descriptografar\n");
        scanf("%d", &escolha);

    } while (escolha != 1 && escolha != 2); //Repete o loop enquanto a entrada
não for válida.

    (escolha == 1) ? (operacao = CRIPTOGRAFAR) : (operacao = DESCRIPTOGRAFAR);
//Atribui uma operação a cada escolha.
```

Usaremos os mesmos princípios anteriores, mas agora para definir os números 1 e 2 para as opções “Criptografar/Descriptografar o texto de um arquivo .txt” e “Ler

do teclado e salvar em um arquivo .txt", respectivamente, novamente utilizando um laço de repetição para garantir que o usuário insira uma entrada válida.

```
  system("cls");

    printf("Escolha uma opção:\n");
    printf("1 - %s texto de um arquivo .txt\n", (operacao) ? "Criptografar" :
"Descriptograr");
    printf("2 - Ler do teclado e salvar em um arquivo .txt\n");

    do
    {
        scanf("%d", &escolha);
    } while (escolha != 1 && escolha != 2);
```

Caso o usuário opte por inserir o texto diretamente do terminal de comandos, as seguintes linhas de código serão executadas, elas receberão, em sequência, a chave que será usada para a codificação, a mensagem que se deseja alterar e, por fim, o nome do arquivo que será gerado, como resultado, pela operação.

```
if (escolha == 2) {

        char mensagem[MAX_SIZE]; //Declara variável para armazenar mensagem.

        system("cls");

        printf("%s > Ler do teclado e salvar em um arquivo.\n\n", (operacao) ?
"Criptografar" : "Descriptograr");

        printf("Digite a chave (um número inteiro): ");
        scanf("%d", &chave);
        getchar(); // pegar o caractere de nova linha pendente do scanf
anterior

        system("cls");
        printf("Digite a mensagem:\n");
        fgets(mensagem, sizeof(mensagem), stdin);


        printf("Digite o nome do arquivo de saída: ");
        scanf("%s", nomeArquivoSaida);
```

```
        strcat(nomeArquivoSaida, ".txt"); //função para concatenar string lida
do teclado com o tipo de arquivo.
```

Agora, o arquivo designado será aberto, mas, caso ocorra um erro ao encontrá-lo, isso será comunicado ao usuário, se aberto corretamente, as operações de criptografia serão executadas, o resultado será colocado no arquivo e ele será fechado, encerrando o processo e o programa.

```
FILE* arquivoSaida = fopen(nomeArquivoSaida, "w"); //Abre o arquivo
        if (arquivoSaida == NULL) {
            printf("Não foi possível criar o arquivo de saída.\n");
            return 1;
        }

        CifrarMensagem(mensagem, chave, operacao); // Criptografar a mensagem

        fprintf(arquivoSaida, "%s", mensagem);// grava a mensagem
criptografada no arquivo.txt

        fclose(arquivoSaida); //Fecha o arquivo após salvar a mensagem.
        printf("Texto criptografado e salvo com sucesso no arquivo de
saída.\n");
        return 0;
    }
```

No entanto, caso a opção escolhida seja a análise de um arquivo já existente, a cadeia de comando a ser utilizada, ela fará os mesmos processos anteriores de obtenção e processamento de dados, mas substituirá a entrada de mensagem pela entrada do nome de arquivo. Uma mensagem de erro também será enviada caso haja algum obstáculo com a localização do arquivo. Finalizada essa cadeia, o programa é encerrado. Essa é a última parte da função principal.

```
system("cls");
    printf("%s > Ler do arquivo.\n\n", (operacao) ? "Criptografar" :
"Descriptograr");
    printf("Digite a chave (um número inteiro): ");
    scanf("%d", &chave);

    char nomeArquivoEntrada[MAX_SIZE];
    printf("Digite o nome do arquivo de entrada: ");
    scanf("%s", nomeArquivoEntrada);
```

```
    strcat(nomeArquivoEntrada, ".txt"); //aproveitando a biblioteca string.h
para concatenar o nome com o tipo de arquivo

    printf("Digite o nome do arquivo de saída que será criado: ");
    scanf("%s", nomeArquivoSaida);
    strcat(nomeArquivoSaida, ".txt");

    return ProcessarArquivos(nomeArquivoEntrada, nomeArquivoSaida, chave,
operacao); //Caso não seja possível criar ou abrir arquivo retorna 1.
}
```

3.5 Função de Retorno:

Essa função será responsável por fazer a devolutiva do tipo de caractere que está sendo inserido no texto, ela será útil para que, ao criptografarmos as mensagens, sejam substituídos caracteres da mesma classe e para que se mantenha a organização do código.

```
int RetornaTipoChar(unsigned char caracter) {
    if (caracter >= '0' && caracter <= '9')
        return NUMERO;
    else if (caracter >= 'A' && caracter <= 'Z')
        return MAIUSCULO;
    else if (caracter >= 'a' && caracter <= 'z')
        return MINUSCULO;

    return ESPECIAL;
}
```

3.6 Função de Cifra:

Essa é a função mais importante deste projeto, é nela em que a “Cifra de César” realmente ocorre, aqui serão realizadas as operações necessárias para que a criptografia ou descriptografia ocorra.

O primeiro passo será definir os parâmetros a serem analisados, em seguida será criada uma variável para que se identifique o tamanho da mensagem. Então um laço de repetição “for” será criado.

```
void CifrarMensagem(char* mensagem, int chave, int criptografar) {
    int tamanhoDaMensagem = strlen(mensagem);

    for (int i = 0; i < tamanhoDaMensagem; i++) {
```

No laço de repetição, será identificado o tipo de caractere, a sequência abaixo se refere às letras maiúsculas, que serão substituídas apenas por letras maiúsculas. Ao analisar o tipo de processo selecionado (criptografia ou descriptografia), o programa realizará a operação indicada para tal, baseando-se na chave para regular o deslocamento.

```
        if (RetornaTipoChar(mensagem[i]) == MAIUSCULO) {
            if (criptografar) {
                mensagem[i] = ((mensagem[i] - 'A' + chave) % 26) + 'A';
            }
            else {
                mensagem[i] = ((mensagem[i] - 'A' - chave + 26) % 26) + 'A';
            }
        }
```

As linhas de códigos subsequentes são muito semelhantes, realizarão os mesmos processos, mas para diferentes caracteres, letras minúsculas e números, respectivamente, encerrando-se a função de cifra.

```
else if (RetornaTipoChar(mensagem[i]) == MINUSCULO) {
            if (criptografar) {
                mensagem[i] = ((mensagem[i] - 'a' + chave) % 26) + 'a';
            }
            else {
                mensagem[i] = ((mensagem[i] - 'a' - chave + 26) % 26) + 'a';
            }
        }
        else if (RetornaTipoChar(mensagem[i]) == NUMERO) {
            if (criptografar) {
                mensagem[i] = ((mensagem[i] - '0' + chave) % 10) + '0';
            }
            else {
                mensagem[i] = ((mensagem[i] - '0' - chave) % 10) + '0';
            }
        }
    }
}
```

3.7 Função de Processamento de Arquivos:

Na última função de nosso código, serão definidos os processos necessários para a leitura e edição de um arquivo no formato .txt.

Começaremos definindo parâmetros e identificando os arquivos a serem abertos. O primeiro a ser analisado será o arquivo de entrada, para os casos em que o usuário deseja utilizar o texto de um arquivo previamente preparado. Logo em seguida será identificado o arquivo de saída, ou seja, o que guardará o resultado de nossa cifra.

Em ambos os casos, serão apontadas mensagens de erro caso o arquivo desejado não seja encontrado ou não possa ser aberto.

```
int ProcessarArquivos(char* nomeArquivoEntrada, char* nomeArquivoSaida, int
chave, int criptografia) {
    FILE* arquivoEntrada = fopen(nomeArquivoEntrada, "r");
    if (arquivoEntrada == NULL) {
        printf("Não foi possível abrir o arquivo de entrada.\n");
        return 1;
    }

    FILE* arquivoSaida = fopen(nomeArquivoSaida, "w");
    if (arquivoSaida == NULL) {
        printf("Não foi possível criar o arquivo de saída.\n");
        fclose(arquivoEntrada);
        return 1;
    }
```

Então, armazenamos a mensagem lida temporariamente a mensagem inserida para que seja lida, ao receber a mensagem criptografada, o programa substitui os caracteres, reescrevendo a mensagem na cifra desejada.

Depois disso, os arquivos são fechados e a função é encerrada. Essas são as linhas finais do código como um todo.

```
 char linha[1000]; // Usada para armazenar temporariamente as linhas lidas de
um arquivo
    while (fgets(linha, sizeof(linha), arquivoEntrada)) {
        CifrarMensagem(linha, chave, criptografia);
        fprintf(arquivoSaida, "%s", linha);
    }

    fclose(arquivoEntrada);
    fclose(arquivoSaida);
```

```
    printf("Operação concluída com sucesso.\n");
    return 0;
}
```

4. RESULTADOS E DISCUSSÕES

Ao final da construção do código, foi constatado que as mensagens foram alteradas como o esperado, deslocando seus caracteres de acordo com a chave fornecida nos processos de criptografia, da mesma forma, reorganizando-os na descriptografia. Diversas combinações de chaves e mensagens foram testadas, bem como os diferentes métodos de entradas de dados. Pode-se dizer que o projeto “Cifra de César” funciona da forma como deveria.

As mensagens de erro também foram testadas através de erros propositais, colocando à prova a análise de informações. Em todos os casos testados, nos quais inserimos valores inválidos, bem como arquivos inexistentes, foram apontados erros de processamento. Isso indica que o código foi construído corretamente.

5. CONCLUSÕES

Com o fim dos testes e execuções do programa, averiguamos que é possível criar um código em linguagem C para a cifra de mensagens, confirmando também que a lógica de programação utilizada está correta e que funciona de maneira adequada.

REFERÊNCIAS BIBLIOGRÁFICAS

SOUZA, Fábio. **Domine a linguagem C: Tudo o que você precisa saber em um único lugar**. Disponível em:<https://embarcados.com.br/linguagem-c-guia-completo/>. Acesso em: 22 de setembro de 2023.

Ramo Estudantil IEEE - UEL

Contato do Ramo: sb.uel@ieee.org
Institute of Electrical and Electronics Engineers – IEEE
Universidade Estadual de Londrina - UEL • Paraná - Brasil