
UNIPAC

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

4° PERÍODO 2021/02

DISCENTES:

BERNARDO RESENDE ANDRÉS

CLAUDIMAR JOSÉ DA CRUZ

RAFAEL DE SOUZA DAMASCENO

# EDITOR CRIPTOGRAFADO

**cXUZUf[chd]fW4fch]XY**

Atividade da 2°etapa para a
aprovação da disciplina de Estrutura
de dados, ministrada pelo
Prof. Nairon Neri Silva .

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

Barbacena – 2021

## Descrição da Atividade

A atividade tem como objetivo a criação de um editor de texto para comunicação entre agentes de segurança, por meio do qual eles poderão trocar mensagens criptografadas. A criptografia será baseada nos caracteres da tabela ASCII. Todos os agentes utilizarão o mesmo programa.

No editor teremos as opções de criar um arquivo, ler e editar um arquivo e de salvar o arquivo que está sendo editado ou lido. O nome do arquivo também deverá ser criptografado.

O algoritmo de criptografia usado no editor deverá ser criado pelo grupo. Além do código do editor deverá ser entregue também um relatório com os testes realizados, descrição do funcionamento do algoritmo e observações feitas pelo grupo no decorrer da elaboração do trabalho.

## Codificação e decodificação

Tratando do nosso algoritmo de criptografia e descriptografia, usamos alternâncias para os dados de entrada durante o processo de envio das mensagens.

Basicamente, o seu funcionamento começa com o usuário criando um arquivo, o nome do arquivo criptografado é mostrado após sua criação, juntamente com uma mensagem relatando se o processo obteve êxito ou não . Em seguida, na opção de editar o arquivo o usuário digita o nome do arquivo criptografado, após isso ele pode começar a escrever sua mensagem de forma que ele lê, no caso, uma mensagem na língua portuguesa. Após pronta a mensagem, irá começar todo o processo de criptografia da mensagem.

A criptografia se baseia no tamanho da mensagem do usuário. Ou seja, para cada tamanho do texto, haverá diferentes procedimentos tomados pelo código. Para fins de exemplo, supondo uma palavra de 4 caracteres escrita pelo usuário, quando é chamada a função para a codificação, para cada caractere fornecido pela palavra, o código irá “pular” 4 casas a frente para aquele caractere e para completar mais uma camada de segurança ele inverte a palavra já criptografada, sendo assim a sua soma irá sempre se basear no tamanho da mensagem, aplicando uma criptografia diferente para cada codificação e decodificação.

A decodificação consiste nos mesmos passos da codificação, desde a chamada da leitura de descriptografia, o código irá ler o arquivo criptografado, irá receber em uma variável ainda com os caracteres criptografado, e logo em seguida irá fazer uma chamada para a função de descriptografar. Nesse caso, voltando ao exemplo acima, em uma palavra de 4 caracteres, ele irá voltar 4 casas, e também irá reverter a mensagem, voltando assim a mensagem original.

**Descrição do Código**

Bibliotecas usadas, e a criação do ponteiro para o arquivo.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <time.h>

FILE *arquivo;
```

Função responsável por criar o arquivo.

```c
void criar_Arquivo(char nome[12])
{
    arquivo = fopen(nome, "w");

    if(arquivo == NULL)
    {
       printf("\nErro ao criar o arquivo\n");
       return 1;
    }
    fflush(stdin);
    printf("\nArquivo criado com sucesso! \n\nCom o nome de: %s \n\n", nome);
    fclose(arquivo);
}
```

Função responsável por criptografar a mensagem.

```c
void codifica(char s[1000], int tamanho, int Rand)
{
    int i, j;
    for (i= tamanho-1; i>=0 ; i--)
    {
        j = s[i];
        j = j + Rand;
        s[i] = j;
        printf("%c", s[i]);
        fputc(s[i], arquivo);
    }
    printf("\n\n");
}
```

Função responsável por descriptografar a mensagem.

```
40  void descodifica(char s[1000], int tamanho, int rand)
41  {
42      int i, j;
43      for (i=tamanho-1; i>=0; i--)
44      {
45          j = s[i];
46          j = j - rand;
47          s[i] = j;
48          printf("%c", s[i]);
49      }
50      printf("\n\n");
51  }
```

Função responsável por descriptografar a mensagem recebida de outro usuário.

```
53  void descodifica2(char s[1000], int tamanho, int rand)
54  {
55      int i, j;
56      for (i= tamanho-1 ; i>=0; i--)
57      {
58          j = s[i];
59          j = j - rand;
60          s[i] = j;
61          printf("%c", s[i]);
62      }
63      printf("\n\n");
64  }
```

Variáveis usadas na execução do código.

```
80  int main()
81  {
82      //setlocale(LC_ALL, "portuguese");
83      srand(time(NULL));
84
85      int tamanho =0;
86      int opcao =0;
87      int criar = 0;
88      int conteudo =0;
89      char nome_arquivo[12];
90      char arquivo_usuario[12];
91      char texto[1000];
92      char texto_decodificado[1000];
93      char nome_arquivo_recebido[12];
94
```

,

do while com o texto do menu da aplicação.

```
83      printf("\n---------------------------------");
84      printf("\n\tEditor de texto\n");
85
86      do
87      {
88          printf("---------------------------------\n\n");
89          printf("[1] Criar arquivo\n");
90          printf("[2] Editar arquivo\n");
91          printf("[3] Salvar arquivo\n");
92          printf("[4] Ler arquivo criado\n");
93          printf("[5] Traduzir mensagem recebida\n");
94          printf("[6] Sair\n");
95          printf("\nOpcao: ");
96          fflush(stdin);
97          scanf("%d", &opcao);
98          printf("\n---------------------------------\n");
99
```

case 1, onde o usuário cria o arquivo. E seu nome é exibido.

```
100         switch (opcao)
101         {
102
103         case 1:
104             //for(i =0;i< 255;i++){
105                 //printf("%d - %c\n", i,i);
106             //}
107             strcpy(nome_arquivo, "iponkjh.txt");
108             //printf("%s", nome_arquivo);
109             criar_Arquivo(nome_arquivo);
110             criar = 1;
111             opcao = 0;
112
113             fflush(stdin);
114             break;
```

case 2, onde o usuário edita o arquivo, é solicitado o seu nome criptografado, após isso o usuário digita a mensagem para criptografar.

```
116         case 2:
117             if(criar == 1)
118             {
119                 fflush(stdin);
120                 printf("\nExemplo de saida de arquivo: 'arquivo.txt'\n");
121                 printf("\nInforme o nome do arquivo criado acima: ");
122                 fgets(arquivo_usuario, 12, stdin);
123 
124                 if(strcmp(arquivo_usuario, "iponkjh.txt") == 0)
125                 {
126                     printf("\nArquivo encontrado!\n\n");
127                     arquivo = fopen(nome_arquivo, "a");
128 
129                     if(arquivo == NULL)
130                     {
131                         printf("Erro ao abrir arquivo!");
132                         return 1;
133                     }
134                     fflush(stdin);
135                     printf("Digite sua mensagem: ");
136                     fflush(stdin);
137                     fgets(texto, 1000, stdin);
138                     tamanho = strlen(texto) - 1;
139                     printf("\nMensagem criptografada: ");
140                     codifica(texto, tamanho, tamanho);
141                     conteudo = 1;
142 
143                     printf("\nAtencao: Apos edicao salve o arquivo na opcao[3]\n.");
144                 }else
145                 {
146                     printf("\nErro! Nome do arquivo incorreto!\n\n");
147                 }
148                 }else
149                 {
150 
151                     printf("\nVoce nao criou o arquivo\n");
152 
153                 }
154             break;
```

case 3, responsável por salvar o arquivo, caso haja algum erro será exibido uma mensagem.

```
156         case 3:
157             if((criar == 1) && (conteudo == 1))
158             {
159                 fclose(arquivo);
160                 printf("\nArquivo salvo com sucesso!\n\n");
161             }else
162             {
163                printf("\nArquivo nao criado ou arquivo vazio, por isso nao pode ser salvo.\n\n");
164             }
165             fflush(stdin);
166             opcao =0;
167             break;
```

case 4, onde o usuário poderá ler o arquivo criado.

```
168         case 4:
169             strcpy(nome_arquivo, "iponkjh.txt");
170             arquivo = fopen(nome_arquivo, "r");
171 
172             if(arquivo == NULL)
173             {
174                printf("\nErro ao abrir o arquivo\n");
175                 return 1;
176             }
177             printf("\nMensagem descriptografada: ") ;
178 
179 
180            while(feof(arquivo) ==0)
181            {
182                 fgets(texto_decodificado, 1000, arquivo);
183            }
184             tamanho = strlen(texto_decodificado);
185             descodifica(texto_decodificado, tamanho, tamanho);
186 
187             fclose(arquivo);
188             break;
```

case 5, aqui o usuário tem a opção de digitar ou colar uma mensagem recebida para a sua descriptografia.

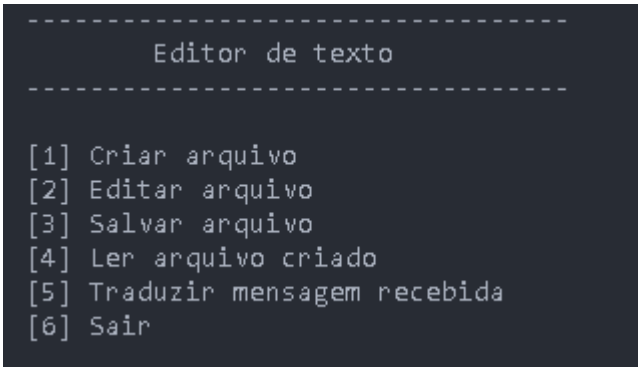
```
190         case 5:
191             fflush(stdin);
192             printf("\nDigite ou cole o texto criptografado: ");
193             gets(nome_arquivo_recebido);
194             tamanho = strlen(nome_arquivo_recebido);
195             printf("\nTexto descodificado: ");
196             descodifica2(nome_arquivo_recebido, tamanho, tamanho);
197             break;
```

case 6, opção para sair do programa

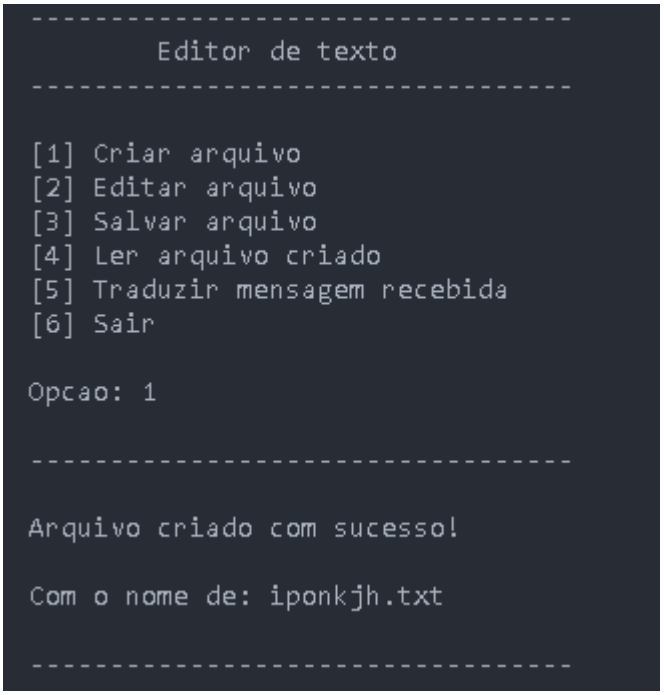
```
199         case 6:
200             printf("\nObrigado! %c %c %c %c %c\n", 1,3,2,3,1 );
201             printf("\n---------------------------------\n");
202             break;
203 
204         default:
205             printf("\nNumero nao reconhecido! = %d\n\n", opcao);
206         }
207     }while(opcao != 6);
208 
209     return 0;
210 }
```

**Tela de Testes**

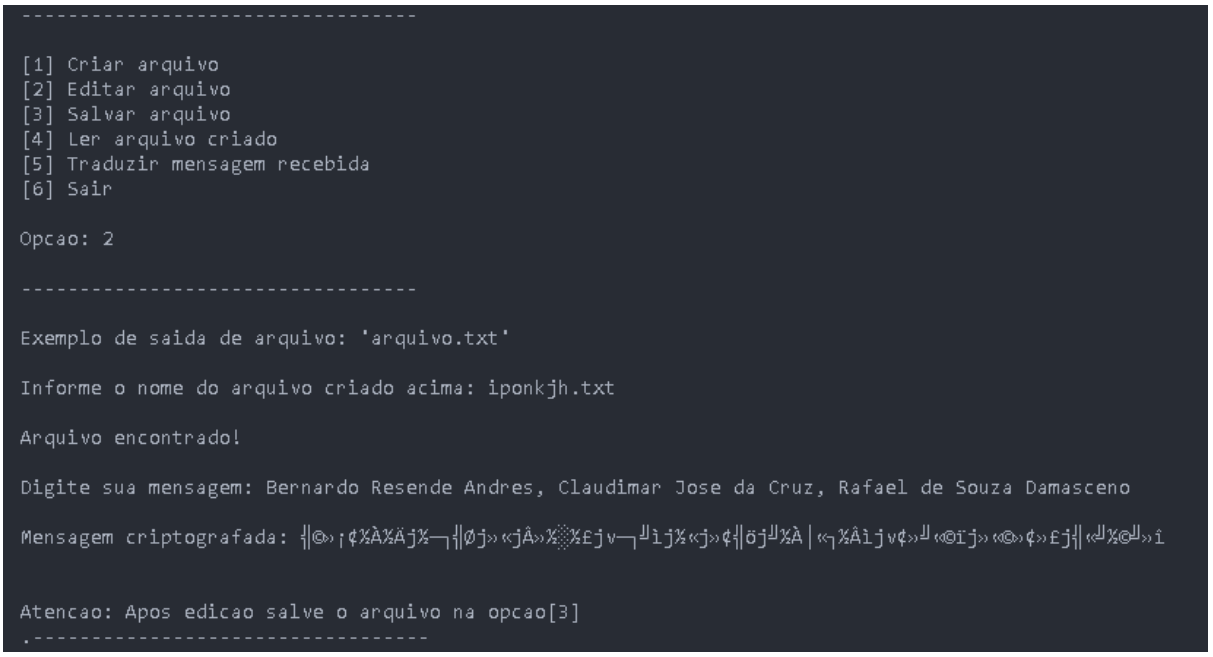
Tela inicial do programa.



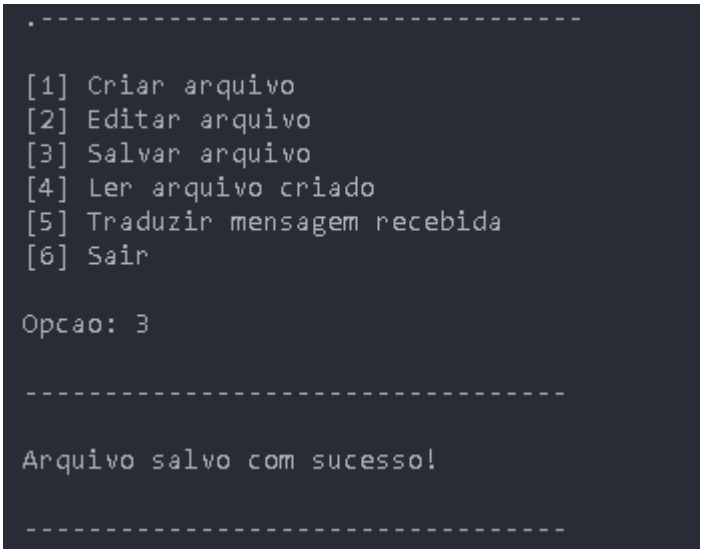

Criação de um arquivo e exibição de seu nome criptografado.



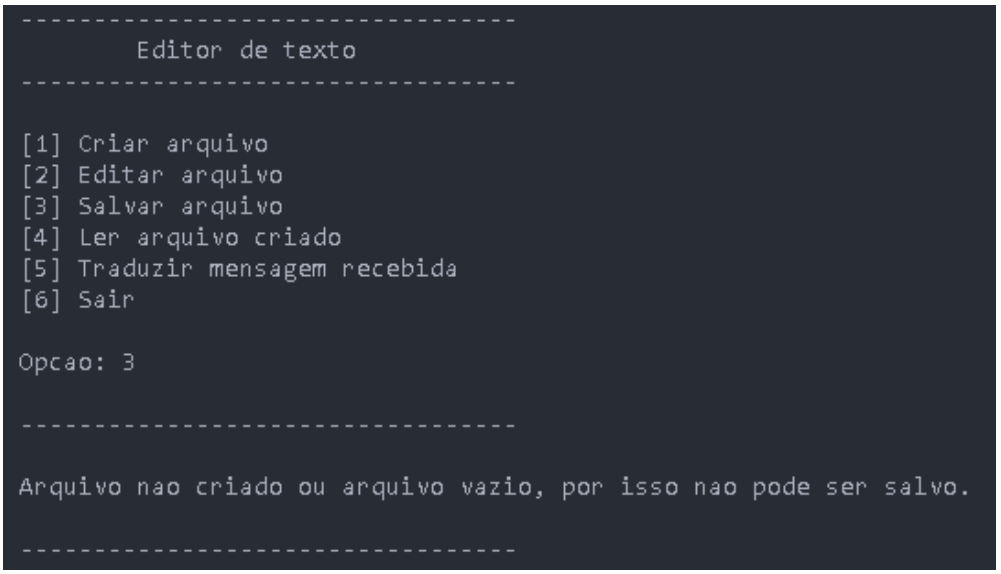
Parte de edição do arquivo, onde o usuário digita o nome do arquivo criptografado, a mensagem que deseja criptografar. A mensagem criptografada é exibida.



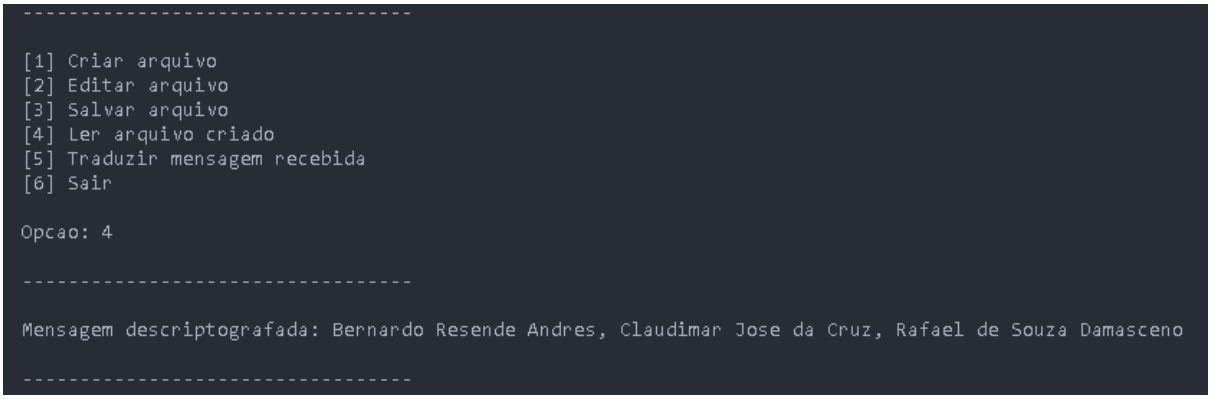

Mensagem exibida após o arquivo ser salvo.



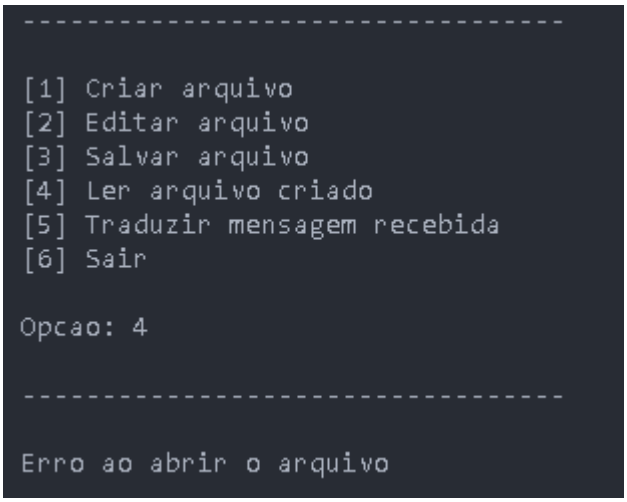
Mensagem exibida explicando porque o arquivo não pode ser salvo, por não ter sido criado ou estar vazio.



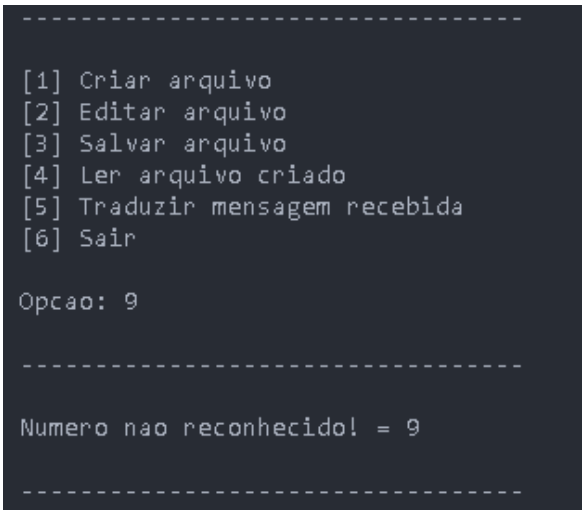

Exibição da mensagem descriptografada de um arquivo criado.



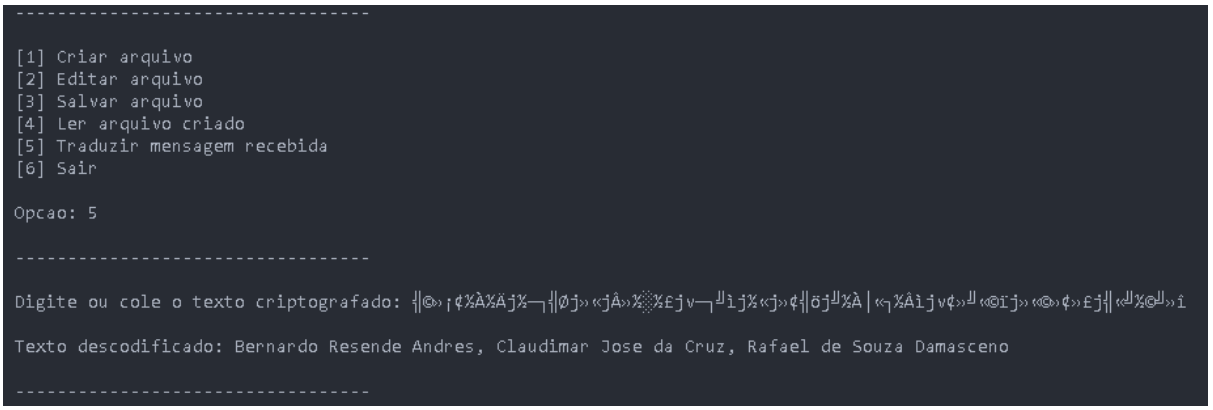

Mensagem de erro ao tentar abrir um arquivo que não existe.



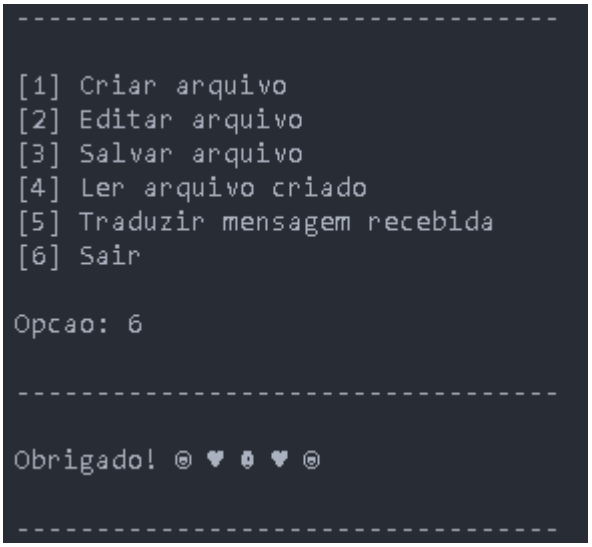
Opção do menu digitada incorretamente.




Exibição de mensagem descriptografada.




Tela de saída do programa.



## Observações

- Notamos uma curiosidade quando imprimimos todos os caracteres da tabela ascii na tela para fazermos os testes no algoritmo de criptografia, sempre que executamos escutamos um sinal sonoro. Após pesquisa vimos que se tratava do caractere Bell (código 7), ele é um caractere de controle que emite um sinal audível, nenhum dado é impresso na tela.
- Podemos observar a presença de emojis, que possibilita ao usuário uma interface mais agradável.
- Estávamos com problemas para pedir os dados ao usuário, diversas vezes o programa pulava as etapas pedidas. Para isso usamos em diversas partes do código a função *fflush(stdin)*, que é usada especificamente para limpar o buffer do teclado.
- O nome do arquivo criptografado ficou sendo como *"iponkjh.txt"*, o qual será exibido ao usuário.
- No princípio estávamos com a ideia de utilizarmos números randômicos para fazer a criptografia e descriptografia, esses números seriam somados ou diminuídos em cada caractere passado para obter novas palavras. Porém depois de alguns testes vimos que não teríamos o controle de uma mensagem recebida de outra pessoa. Com isso passamos a utilizar como parâmetro o tamanho das strings criadas ou recebidas, obtido através da função *strlen()*.
- Após a criptografia da mensagem, ela passaria por mais um camada de segurança, a inversão da própria mensagem. Porém encontramos alguns problemas na hora da visualização do arquivo, após uma análise feita conseguimos reverter o problema com a função *fputc(),* colocada no lugar da função *fprintf().*

## Referências

Caractere bell. Wikipédia. Disponível em: https://pt.wikipedia.org/wiki/Caractere_bell. Acesso em: 29 de novembro de 2021.