Ramo Estudantil IEEE - UEL

ANDRÉ AKIRA MURAOKA (andre.akira.muraokaa@uel.br)
IGOR SHOITI TUTIDA SUGUIEDA
(igor.tutidasuguieda@uel.br)
JULIA CRISTINA PEREIRA PROENÇA (julia.cristina1@uel.br)
RAFAEL LANÇONI SANTOS(rafael.lanconi@uel.br)

RELATÓRIO FINAL:

Projeto Cifra de César

Londrina
2023

Ramo Estudantil IEEE - UEL

ANDRÉ AKIRA MURAOKA
IGOR SHOITI TUTIDA SUGUIEDA
JULIA CRISTINA PEREIRA PROENÇA
RAFAEL LANÇONI SANTOS

RELATÓRIO FINAL:

Projeto 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

MURAOKA, ANDRÉ AKIRA. SUGUIEDA, IGOR SHOITI TUTIDA. PROENÇA, JULIA CRISTINA PEREIRA PROENÇA. SANTOS, RAFAEL LANÇONI. **Relatório Final**: Projeto Cifra de César. 2023. 18 folhas. Relatório apresentado ao Ramo Estudantil IEEE da Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

O projeto Cifra de César consiste em recriar o sistema de criptografia utilizado pelo general Júlio César para comunicar-se com seu exército. A cifra consiste basicamente em um método de substituição de uma letra por outra, a fim de criptografar uma mensagem e somente ser descriptografada por quem possuir a chave para decodificá-la.

Palavras-chave: Cifra. César. Criptografia.

SUMÁRIO

Sumário

1. Uma breve introdução ao projeto da Cifra de César

1.1 O que é a Cifra de César:

A Cifra de César consiste em um método de criptografia básica utilizado por Júlio César para comunicar-se com seu exército. Trata-se de uma cifra de substituição em que cada letra da mensagem original é deslocada algumas posições no alfabeto e substituída por outra a fim de ocultar uma mensagem, sendo somente descriptografada por aqueles que possuírem a chave.

1.2 Ferramentas utilizadas:

Para a realização do projeto baseado na Cifra de César, foram utilizadas algumas ferramentas auxiliares como a linguagem de programação C com o auxílio do programa Visual Studio Code juntamente com o Mingw (Minimalist GNU for Windows).

2. FUNDAMENTAÇÃO TEÓRICA

2.1 O que é criptografia?

A criptografia é um dos métodos utilizados para garantir a segurança de dados. Nela, dados em formato legível são criptografados com base em uma chave, a fim de ocultar o conteúdo da mensagem para terceiros, sendo somente legíveis após a descriptografia com a chave correta.

Um detalhe interessante é que quanto mais complexa for a chave criptográfica, mais segura ela é, já que torna-se menos suscetível a decodificação por terceiros por “ataques de força bruta”, ou seja, ataques de tentativa e erro.

2.1.1 Quais são os métodos de criptografia mais comuns?

Os dois métodos mais comuns de criptografia são: chaves de criptografia simétrica e chaves de criptografia assimétrica. A seguir, uma breve explicação sobre elas.

Chave de criptografia simétrica: também conhecida por criptografia de chave privada. Nela, há somente uma chave tanto para a criptografia quanto para a descriptografia.

Chave de criptografia assimétrica: esta utiliza duas chaves, uma para criptografia (chave privada) e outra para descriptografia (chave pública).

2.2 O que é uma Cifra e por quê de César

Inicialmente, vale-se uma breve explicação sobre o que é uma cifra. Cifra é basicamente uma técnica usada para esconder uma mensagem por meio da mistura ou substituição das letras da mensagem original.

A cifra de César recebeu esse nome, devido ao grande uso desse método (criptografia por cifras) utilizado pelo general romano Júlio César para comunicar-se com seu exército. Como dito anteriormente, tratava-se de uma cifra de substituição monoalfabética em que cada letra era substituída por outra, três posições à frente no alfabeto.

3. METODOLOGIA

Um dos grandes auxiliares na construção do algoritmo da Cifra de César foi a tabela ASCII.

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

Fonte:https://carlacastanho.github.io/Material-de-APC/assets/images/Strings/ASCII.png

3.1 Programando o Menu

Inicialmente, e como uma das bases do programa, foi configurado um menu para transitar entre as opções de criptografia, descriptografia e finalização do programa. Ela funciona por meio de um “switch case”, que fará o usuário ir para um dos submenus da opção escolhida.

```
while(1){
    printf("\n============ Menu ============");
    printf("\n1 - Criptografar mensagem");
    printf("\n2 - Descriptografar mensagem");
    printf("\n3 - Finalizar programa");
    printf("\n==============================");
    printf("\nSelecione uma opção: ");
    scanf(" %d", &OpcaoMenu);
```

Fonte: o próprio autor

3.2 Programando o algoritmo de criptografia

Após selecionar a opção de criptografia. Somos levados a outro menu, nele, selecionamos se queremos inserir um texto por arquivo ou digitando-o diretamente do teclado.

Caso seja selecionado a opção 1, “ler texto do teclado” o código será continuado nos solicitando um texto e uma chave para codificação. E após isso, ainda pede um nome para gerar o arquivo de destino com o texto criptografado.

```
if(OpcaoSubmenu==1){
    printf("\nDigite um texto: ");
    scanf(" %[^\n]", Frase);
    printf("Digite o valor da chave: ");
    scanf(" %d", &Chave);
    normString(Frase);
    Criptografar(Frase, Chave);

    printf("Digite o nome do arquivo de destino: ");
    scanf(" %[^\n]", nomeDestino);

    arqDestino = fopen(nomeDestino, "w");
    if (arqDestino == NULL) {
        printf("Erro ao abrir o arquivo de destino.\n");
        exit(1);
    }

    fputs(Frase, arqDestino);
    fclose(arqDestino);
```

Fonte: o próprio autor

Caso selecione a opção 2, “Ler arquivo”. O algoritmo solicita o nome do arquivo de origem, o qual se quer criptografar a mensagem e, caso haja algum problema, ele aponta erro.

```
}else if(OpcaoSubmenu==2){
    printf("Digite o nome do arquivo de origem: ");
    scanf(" %[^\n]", nomeOrigem);

    arqOrigem = fopen(nomeOrigem, "r");
    if (arqOrigem == NULL) {
        printf("Erro ao abrir o arquivo de origem.\n");
        exit(1);
    }

    printf("Digite o nome do arquivo de destino: ");
    scanf(" %[^\n]", nomeDestino);

    arqDestino = fopen(nomeDestino, "w");
    if (arqDestino == NULL) {
        printf("Erro ao abrir o arquivo de destino.\n");
        exit(1);
    }

    printf("Digite o valor da chave: ");
    scanf(" %d", &Chave);

    while (fgets(Frase, sizeof(Frase), arqOrigem) != NULL) {
        normString(Frase);
        Criptografar(Frase, Chave);
        fputs(Frase, arqDestino);
    }

    fclose(arqOrigem);
    fclose(arqDestino);
}else{
    printf("\nValor inválido!\n");
    exit(1);
}
break;
```

Fonte: o próprio autor

Agora, em relação ao principal código do algoritmo de criptografia.

Com base no texto inserido, uma variável chamada “comprimento” pega o

tamanho do texto, que é um vetor, para que ele delimite o início e o fim do arquivo (quebra de linha) para criptografia.

Então, baseado na tabela ASCII foi construído um algoritmo que pegava o valor de cada letra da mensagem, comparava com a tabela, e a partir disso com o auxílio da chave recebida, alterava o valor da letra, deslocando-a no alfabeto.

```
void Criptografar(char *Frase, int Chave){
    int iniMaiusculas=64, iniMinusculas=96, fimMaiusculas=90, fimMinusculas=122;
    int Comprimento = strlen(Frase);
    char Letra;

    for(int i=0; i<Comprimento; i++){
        Letra = Frase[i];
        if(Letra>iniMaiusculas&&Letra<=fimMaiusculas){
            if((Letra + Chave)<=fimMaiusculas){
                Letra += Chave;
            }else{
                Letra = (Letra + Chave)%fimMaiusculas + iniMaiusculas;
            }
        }else if(Letra>iniMinusculas&&Letra<=fimMinusculas){
            if((Letra + Chave)<=fimMinusculas){
                Letra += Chave;
            }else{
                Letra = (Letra + Chave)%fimMinusculas + iniMinusculas;
            }
        }
        Frase[i] = Letra;
    }
}
```

Fonte: o próprio autor

3.3 Programando o algoritmo de descriptografia

Selecionado a opção de descriptografia, o algoritmo se assemelha levemente ao seu anterior. Porém, nele tem-se os comandos referentes a descriptografia de textos. Então, temos as opções de digitar um texto criptografado ou entrar com um arquivo cujo o texto esteja criptografado. E, novamente, temos a saída no formato de arquivo .txt.

```
case 2:
    printf("\n1 - Ler texto do teclado");
    printf("\n2 - Ler arquivo");
    printf("\n-----------------------------");
    printf("\nSelecione uma opção: ");
    scanf(" %d", &OpcaoSubmenu);

    if(OpcaoSubmenu==1){
        scanf(" %[^\n]", Frase);
        printf("Digite o valor da chave: ");
        scanf(" %d", &Chave);

        Descriptografar(Frase, Chave);

        printf("Digite o nome do arquivo de destino: ");
        scanf(" %[^\n]", nomeDestino);

        arqDestino = fopen(nomeDestino, "w");
        if (arqDestino == NULL) {
            printf("Erro ao abrir o arquivo de destino.\n");
            exit(1);
        }

        fputs(Frase, arqDestino);
        fclose(arqDestino);
```

Fonte: o próprio autor

```
}else if(OpcaoSubmenu==2){
    printf("Digite o nome do arquivo de origem: ");
    scanf(" %[^\n]", nomeOrigem);

    arqOrigem = fopen(nomeOrigem, "r");
    if (arqOrigem == NULL) {
        printf("Erro ao abrir o arquivo de origem.\n");
        exit(1);
    }

    printf("Digite o nome do arquivo de destino: ");
    scanf(" %[^\n]", nomeDestino);

    arqDestino = fopen(nomeDestino, "w");
    if (arqDestino == NULL) {
        printf("Erro ao abrir o arquivo de destino.\n");
        exit(1);
    }

    printf("Digite o valor da chave: ");
    scanf(" %d", &Chave);

    while (fgets(Frase, sizeof(Frase), arqOrigem) != NULL) {
        Descriptografar(Frase, Chave);
        fputs(Frase, arqDestino);
    }

    fclose(arqOrigem);
    fclose(arqDestino);
}else{
    printf("\nValor inválido!\n");
    exit(1);
}
break;
```

Fonte: o próprio autor

Em relação ao código de descriptografia, temos “um processo inverso” ao de criptografia. As bases são as mesmas: uma chave, um texto e o auxílio da tabela ASCII. Novamente, pegamos o tamanho do texto com a variável “comprimento” para determinar onde se inicia e onde termina o texto. Contudo, dessa vez ao invés de

andarmos para frente no alfabeto, andamos “para trás” a fim de obter o texto original.

```
void Descriptografar(char *Frase, int Chave){
    int iniMaiusculas=65, iniMinusculas=97, fimMaiusculas=91, fimMinusculas=123;
    int Comprimento = strlen(Frase);
    char Letra;

    for(int i=0; i<Comprimento; i++){
        Letra = Frase[i];
        if(Letra>=iniMaiusculas&&Letra<fimMaiusculas){
            if((Letra - Chave)>=iniMaiusculas){
                Letra -= Chave;
            }else{
                Letra = (Letra - Chave - iniMaiusculas)%iniMaiusculas + fimMaiusculas;
            }
        }else if(Letra>=iniMinusculas&&Letra<fimMinusculas){
            if((Letra - Chave)>=iniMinusculas){
                Letra -= Chave;
            }else{
                Letra = (Letra - Chave -iniMinusculas)%iniMinusculas + fimMinusculas;
            }
        }
        Frase[i] = Letra;
    }
}
```

Fonte: o próprio autor

3.4 Programando o algoritmo de encerramento do programa

Como o programa está em um loop while infinito, para finalizá-lo foi criado um comando que encerra seu funcionamento.

```
                case 3:
                    printf("\nEncerrando programa...\n");
                    exit(1);
                    break;

                default:
                    printf("\nValor inválido!\n");
                    exit(1);
            }
        }
    }
```

Fonte: o próprio autor

3.5 Implementação da função extra

A implementação da função extra consiste na substituição de caracteres especiais em caracteres normais, tanto na codificação quanto na decodificação com o uso da tabela ASCII.

```
void normCaracter(char *VetChar, int ini){
    int SemNorm = 0;
    char ident = VetChar[ini+1];
    if(ident>=-128&&ident<=-123){/*Multichar -61(-128 a -123) abrange (Ä,Å,Á,Â,À,Ã)*/
        VetChar[ini]='A';/*Altera para caracter válido 'A'*/
    }else if(ident>=-96&&ident<=-92){/*Multichar -61(-96 a -92) abrange (ä,á,â,à,ã)*/
        VetChar[ini]='a';/*Altera para caracter válido 'a'*/
    }else if(ident>=-120&&ident<=-117){/*Multichar -61(-120 a -117) abrange (É,Ê,Ë,È)*/
        VetChar[ini]='E';/*Altera para caracter válido 'E'*/
    }else if(ident>=-88&&ident<=-85){/*Multichar -61(-88 a -85) abrange (é,ê,ë,è)*/
        VetChar[ini]='e';/*Altera para caracter válido 'e'*/
    }else if(ident>=-116&&ident<=-113){/*Multichar -61(-116 a -113) abrange (Í,Î,Ï,Ì)*/
        VetChar[ini]='I';/*Altera para caracter válido 'I'*/
    }else if(ident>=-84&&ident<=-81){/*Multichar -61(-84 a -81) abrange (í,î,ï,ì)*/
        VetChar[ini]='i';/*Altera para caracter válido 'i'*/
    }else if(ident>=-110&&ident<=-106){/*Multichar -61(-110 a -106) abrange (Ö,Ó,Ô,Ò,Õ)*/
        VetChar[ini]='O';/*Altera para caracter válido 'O'*/
    }else if(ident>=-78&&ident<=-74){/*Multichar -61(-78 a -74) abrange (ö,ó,ô,ò,õ)*/
        VetChar[ini]='o';/*Altera para caracter válido 'o'*/
    }else if(ident>=-102&&ident<=-100){/*Multichar -61(-102 a -100) abrange (Ü,Ú,Û)*/
        VetChar[ini]='U';/*Altera para caracter válido 'U'*/
    }else if(ident>=-71&&ident<=-68){/*Multichar -61(-71 a -68) abrange (ü,ú,û,ù)*/
        VetChar[ini]='u';/*Altera para caracter válido 'u'*/
    }else if(ident==-121){/*Multichar -61-121 é 'Ç'*/
        VetChar[ini]='C';/*Altera para caracter válido 'C'*/
    }else if(ident==-89){/*Multichar -61-89 é 'ç'*/
        VetChar[ini]='c';/*Altera para caracter válido 'c'*/
    }else{
        SemNorm=1;
    }

    if(SemNorm==0){/*caso ocorra uma alteração, o multichar vira char único, restando espaços na string*/
        for(int j=ini+1; j< strlen(VetChar); j++){/*Da posição posterior a mudança até o fim da string um char é adiantado*/
            VetChar[j]=VetChar[j+1];
        }
    }
}

void normString(char *VetChar){
    for(int i=0; i< strlen(VetChar); i++){/*Percorre uma string inteira procurando os caracteres inválidos e fazendo mudanças*/
        if(VetChar[i]==-61){
            normCaracter(VetChar, i);
        }
    }
}
```

4. RESULTADOS E DISCUSSÕES

Após a realização de alguns testes de criptografia e descriptografia das mensagens, foi possível notar o correto funcionamento do algoritmo. De forma a concluir que o projeto é funcional.

Foi possível concluir também que a cifra de césar não é um dos métodos mais eficazes na ocultação de uma mensagem, uma vez que comparado a outros, ela é mais simples na decodificação, podendo ceder a um ataque de força bruta.

5. CONCLUSÕES

Depois de realizar mais alguns testes nos sistemas de criptografia e descriptografia, foi possível concluir que eram funcionais. Além disso, foram adicionadas funções extras, como: a troca de um caractere especial por outro válido na criptografia e descriptografia.

Com isso, mais uma vez constatou-se que o algoritmo foi realizado de forma correta, finalizando o projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

O que é criptografia de dados? Disponível em:<**https://www.kaspersky.com.br/resource-center/definitions/encryption>.** Acesso em 18 de setembro de 2023.

Cifra de César Disponível em: <http://www.bosontreinamentos.com.br/seguranca/criptografia-cifra-de-cesar/> . Acesso em 18 de setembro de 2023.