Trabalho de Programação 3

Processador Intel (80x86)

1. Descrição Geral

Neste trabalho você deverá desenvolver um programa capaz de ler um arquivo em formato texto (arquivo de entrada) e uma frase a ser criptografada. Então, seu programa deverá criptografar a frase, conforme algoritmo definido nessa especificação, usando os dados lidos do arquivo e, ao final, escrever um arquivo com o resultado criptografado (arquivo de saída).

O arquivo de entrada, cujo nome será fornecido através do console (teclado e tela), possui o texto de referência para a criptografia da frase. No arquivo, estarão disponíveis bytes (valores entre 00_8 e FF_8). O nome fornecido para o arquivo de entrada nunca terá a extensão. Entretanto, o programa deve acrescentar a extensão ".TXT" para lê-lo. Por exemplo, se o nome fornecido for "fulano", o arquivo a ser lido será "fulano.txt".

A frase a ser criptografada também será fornecida através do console. Devem ser criptografados, apenas, os caracteres ASCII visíveis. Ou seja, deve-se criptografar apenas os caracteres codificados entre 21_8 ("!") e $7E_8$ ("~"). Portanto, eventuais espaços em branco no meio da frase a ser criptografada serão ignorados.

As informações criptografadas deverão ser escritas em um arquivo de saída. O nome do arquivo de saída deve ser o mesmo do arquivo de entrada, mas com a extensão "**KRP**". Por exemplo, se o nome fornecido para o arquivo de entrada for "fulano", o arquivo a ser gerado na saída deverá ser "fulano.krp".

Seu programa deverá ser desenvolvido em linguagem simbólica de montagem do processador 8086 da Intel e executado no ambiente DosBox. Para montagem de seu programa fonte será usado o montador MASM 6.11.

2. Algoritmo de Criptografia

O algoritmo de criptografia a ser empregado é chamado de estenografia. A técnica admite uma série de variantes. Nesse trabalho será usada a variante onde uma frase (a ser criptografada) estará escondida dentro de um texto previamente escolhido (arquivo de entrada).

Para cada caractere da frase a ser criptografada, seu programa deverá procurar, no arquivo de entrada, um byte igual. Nessa comparação, deve-se considerar que caracteres maiúsculos são o mesmo que caracteres minúsculos.

Ao encontrar esse byte no arquivo de entrada, seu programa deverá calcular a posição em que foi encontrado, e salvar essa posição. Considera-se que o primeiro byte do arquivo está na posição 0 (zero). Essa posição deve ser calculada com um número de 16 bits, sem sinal. Portanto, essa posição será um número entre 0000_{10} e 65535_{10}. Ainda, como consequência, serão considerados no processo de estenografia apenas os primeiros 65536 bytes do arquivo de entrada.

O byte da posição 0 (zero) do arquivo não deverá ser utilizado na busca pelos caracteres da frase.

Caracteres iguais não poderão ser codificados com posições iguais. Por exemplo, se na frase houver duas letras iguais, elas não podem usar o mesmo byte do arquivo para criptografá-la. Dessa forma, não poderão aparecer, na lista de posições, dois valores iguais.

Os valores obtidos pelo algoritmo deverão ser escritos no arquivo de saída. Cada valor vai ocupar dois bytes desse arquivo, escritos em formato Little-Endian. Ao final da lista de valores, o programa deve escrever um valor 0000_{16}, de forma que seja possível identificar o final da lista de posições.

A frase a ser criptografada terá, no máximo, 100 caracteres ASCII. O arquivo de entrada terá tamanho máximo de 65536 bytes (64kBytes).

Ao final da execução do processamento, o programa deve informar na tela um resumo do processamento realizado. Devem ser informados, também, eventuais situações de erro no processamento.

Se o programa completar o processamento sem erros, deve ser informado:

- Tamanho do arquivo de entrada (em bytes);
- Tamanho da frase (em bytes);
- Nome do arquivo de saída gerado;
- Resultado: “Processamento realizado sem erro”.

As situações de erro que devem ser verificadas pelo programa e informadas na tela são as seguintes:

- Erros na leitura do arquivo de entrada;
- Arquivo de entrada muito grande (excedeu o tamanho máximo);
- Erros na frase:
  - tamanho da frase maior do que o limite estabelecido;
  - frase vazia (sem qualquer caractere);
  - existência de caracteres que não podem ser processados.
- Não foi possível encontrar um dos símbolos da frase, no arquivo de entrada fornecido;
- Erro na criação do arquivo de saída.

3. Exemplo de aplicação do programa

Considere que o arquivo de entrada contém as informações abaixo, onde estão indicadas as letras do arquivo assim como a posição onde se encontram (número em hexa-decimal):

Posição no arquivo	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	000A
Conteúdo do arquivo	m	a	q	u	i	n	a		d	e	

Posição no arquivo	000B	000C	000D	000E	000F	0010	0011	0012	0013	0014	0015
Conteúdo do arquivo	a	l	a	n		t	u	r	i	n	g

Pretende-se criptografar a seguinte frase: “**Arq intel**”. Assim, o arquivo de saída, com a frase criptografada, será:

Frase	A	r	q	i	n	t	e	l
Conteúdo do arquivo KRP	0001	0012	0002	0004	0005	0010	0009	000C

As letras do arquivo de entrada usadas na criptografia estão “marcadas” em laranja na figura que mostra o arquivo de entrada.

Observar que devem ser criptografadas apenas os caracteres entre 21_8 (“!”) e $7E_8$ (“~”), devendo-se ignorar os outros caracteres, o que inclui ignorar qualquer tipo de espaço em branco.

Finalmente, verifica-se que a solução apresentada não é a única. Dependendo do algoritmo empregado, pode-se obter soluções diferentes. Abaixo está apresentada uma solução alternativa, onde estão assinaladas as diferenças em relação à solução anterior.

Frase	A	r	q	i	n	t	e	l
Conteúdo do arquivo KRP	000D	0012	0002	0013	0014	0010	0009	000C

4. Entregáveis: o que deve ser entregue?

Deverá ser entregue, via Moodle da disciplina, **APENAS** o arquivo fonte com a solução do problema apresentado, escrito *na linguagem simbólica de montagem* dos processadores 80X86 da Intel (arquivo .ASM). Além disso, esse programa fonte deverá conter comentários descritivos da implementação.

Para a correção, o programa será montado usando o montador **MASM 6.11** no ambiente **DosBox 0.74** e executado com diferentes arquivos e frases de entrada. A nota final do trabalho será proporcional às funcionalidades que forem atendidas pelo programa.

O trabalho deverá ser entregue até a data prevista, conforme programado no MOODLE. **Não será aceita a entrega de trabalhos após a data estabelecida**.

5. Observações

Recomenda-se a troca de ideias entre os alunos. Entretanto, a identificação de cópias de trabalhos acarretará na aplicação do Código Disciplinar Discente e a tomada das medidas cabíveis para essa situação (**tanto o trabalho original quanto os copiados receberão nota zero**).

O professor da disciplina reserva-se o direito, caso necessário, de solicitar uma demonstração do programa, onde o aluno será arguido sobre o trabalho como um todo. Nesse caso, a nota final do trabalho levará em consideração o resultado da demonstração.