

Descrição

Dr. Henry Walton é um grande pesquisador e arqueólogo. Sabendo de um naufrágio perdido que ocorreu durante a Segunda Guerra Mundial, ele se interessa muito e vai pessoalmente até o local onde os rumores indicavam. Chegando lá, Henry se encontra com uma caverna cheia d'água e, dentro dela um submarino que carregava as antigas cores alemãs.

O dr. consegue entrar dentro do submarino através de um rombo no casco e lá ele encontra, cheia de água, uma máquina enigma utilizada para criptografar e descriptografar mensagens durante a Segunda Guerra. Ao lado da máquina, inúmeros papéis contendo sequências de letras sem sentido algum.

Henry consegue voltar a superfície com a máquina e os papéis que, ele descobrira. Eles continham mensagens criptografadas. Entretanto, a máquina enigma estava severamente avariada e não poderia ser utilizada para decifrar as mensagens escondidas. Além disso, essa máquina parecia mais antiga e funcionava de uma forma diferente às outras enigmas encontradas durante a guerra. Apesar disso, a máquina ainda entregava as configurações necessárias para decifrar os códigos.

É então que o dr. pede a você que construa um programa capaz de decifrar as mensagens codificadas e descobrir o segredo a muito esquecido. Afinal, o que fazia um submarino perdido dentro daquela caverna, o que eles poderiam ter planejado?

A máquina consiste em três rotores em sequência. Cada rotor é basicamente uma engrenagem e, para cada lado da engrenagem, existem 26 conexões. Toda conexão em um lado da engrenagem conecta a uma outra correspondente do outro lado e não há duas conexões que se juntem. Ou seja, ao conectar cada uma das 26 conexões a um teclado, as conexões do outro lado do rotor serão alguma versão embaralhada das conexões que o teclado envia.

A forma como a máquina funciona é simples. Suponhamos que haja apenas um rotor. Conectamos um teclado a um dos lados do rotor e lâmpadas no outro. Cada lâmpada possui um rótulo que indica uma letra. Quando pressionamos uma letra no teclado, uma conexão se fecha que passa pelo rotor e chega a uma lâmpada correspondente. Como as conexões do rotor são embaralhadas, a lâmpada que acende não é a mesma tecla pressionada. Na verdade, ela representa a versão codificada da letra que foi apertada. Depois de uma tecla pressionada, o rotor gira uma posição e as conexões elétricas com o teclado e lâmpada mudam. Como existem 26 conexões, também existem 26 posições que o rotor pode assumir.

Com 3 rotores a situação é a mesma, mas temos 3 rotores conectados em série. Assim, o sinal que sai do teclado vai para o rotor 1, que vai para o 2 e por fim para o 3, embaralhando cada vez mais a letra. Entretanto, há uma diferença: o segundo rotor só gira uma posição quando o primeiro deu uma volta completa (andou 26 posições), e o terceiro só gira quando o segundo deu uma volta completa. Como se fossem os ponteiros de um relógio.

Por sorte, não é necessário construir fisicamente essa máquina, já que seria lento e custoso. Podemos, ao invés disso, simular ela!

Cada rotor pode ser representado por um vetor, onde o índice é a conexão no lado do teclado e o valor naquele índice representa a conexão com a lâmpada. Imagine o seguinte vetor:

```
int rotor[26] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
                  14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 0 };
```

Aqui estamos considerando que o número 0 representa 'a', 1 'b', e assim por diante. Naquele exemplo com apenas um rotor, teríamos que quando a letra 'a' é pressionada, olhamos `rotor[0]` que é 1 e portanto a lâmpada acesa seria a de rótulo 'b'. Ou seja, a decodificação da letra 'a', nessa configuração, é a letra 'b'. Em seguida rotacionamos o rotor o que é efetivamente mover todos os valores do vetor uma casa para a esquerda. O próximo estado do rotor seria

```
int rotor[26] = { 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
                  15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 0, 1 };
```

Ou seja, se pressionássemos a letra 'a' novamente, obteríamos 'c', já que agora `rotor[0]` é 2.

Com mais rotores, basta sequenciar eles. Com 3 rotores, para codificar a letra 'a', por exemplo, fazemos `rotor3[rotor2[rotor1[0]]]` e vemos o resultado.

Entrada

A entrada será fornecida no seguinte formato

```
Rotores:
<valores rotor 1>
<valores rotor 2>
<valores rotor 3>
```

```
Mensagem:
<mensagem>
```

Os valores dos rotores serão uma sequência de 26 números indicando o que deve ser posto em cada vetor que representa cada rotor.

`<message>` é um texto cifrado de tamanho arbitrário (ler até EOF). Ao invés de ler tudo de uma vez e depois processar, pode-se ficar num ciclo de lê caractere, transforma, imprime, lê, transforma, imprime, etc.

O formato da entrada será exatamente esse. Uma linha escrito "Rotores:", três linhas com os dados dos rotores, uma linha em branco, uma linha escrito "Mensagem:" e qualquer número de linhas com a mensagem.

Atenção: Quando uma letra maiúscula é codificada ela continua maiúscula e o mesmo vale para quando ela é decodificada. Além disso, caracteres que não são do alfabeto não devem ser decodificados. Eles já estão em sua forma final e devem ser impressos desse jeito mesmo.

Saída

A saída deve ser a mensagem decodificada.

Dicas de implementação

Para ignorar uma linha, pode-se usar

```
scanf("%*[^\\r\\n]s"); // Ignora tudo até o final da linha
scanf("%*\\r\\n]s"); // Ignora o pula linha
```

Exemplo 1:

Entrada:

Rotores:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0

Mensagem:

Jaimtyvc hspdpdj ixan.

Zdtjar jeibtk.

Saída:

Mensagem secreta aqui.

Varias linhas.

Exemplo 2:

Entrada:

Rotores:

1	9	16	4	5	7	22	13	6	25	10	2	18	3	11	12	0	19	23	21	15	8	24	20	17	14
18	1	12	7	0	14	23	21	10	16	24	20	8	25	11	2	3	5	19	15	22	9	6	13	17	4
14	24	22	7	3	20	17	18	19	10	0	11	9	21	8	5	23	1	15	16	4	6	25	12	13	2

Mensagem:

Cnxwz, Pxzra!

Saída:

Hello, World!