Notas Emoji

Tutorial · CC3036 · 2021

Diogo Peralta Cordeiro

Separar representação e significado

Um emoji não corresponde necessariamente a um byte, a um ponto de código Unicode ou a uma célula visível. A tarefa fica mais simples quando estas noções são separadas. Lê os bytes, descodifica os valores escalares Unicode, converte os símbolos permitidos para ASCII e só depois interpreta nomes e notas.

O cabeçalho `NOME  NOTA` e a frase final são dados literais definidos pelo enunciado. Mantêm-se iguais nas duas versões linguísticas do exercício.

Descodificar UTF-8 explicitamente

Usa `unsigned char` para os bytes de entrada. Um `char` com sinal pode transformar um byte superior a 127 num valor negativo e comprometer os testes sobre os bits.

Byte inicial	Bytes seguintes	Bits iniciais do valor
`00–7F`	0	O próprio byte
`C2–DF`	1	Os 5 bits inferiores
`E0–EF`	2	Os 4 bits inferiores
`F0–F4`	3	Os 3 bits inferiores

Cada byte de continuação tem de satisfazer `(byte & 0xC0) == 0x80`. Acrescenta os seus seis bits úteis:

```
cp = (cp << 6) | (byte & 0x3F);
```

Os valores mínimos de sequências de dois, três e quatro bytes são `0x80`, `0x800` e `0x10000`. Rejeita valores inferiores a esses mínimos, superiores a `0x10FFFF` ou dentro do intervalo de substitutos `0xD800–0xDFFF`. Estas verificações excluem codificações demasiado longas e valores escalares inválidos. Uma sequência que termina antes de todos os bytes de continuação também é inválida.

Por exemplo, `F0 9F 87 B3` representa U+1F1F3, o indicador regional N. Depois de obter este inteiro, o desenho apresentado no ecrã deixa de ser relevante.

Converter símbolos e consumir os sufixos

Os indicadores regionais correspondem diretamente ao alfabeto:

```
letter = 'A' + cp - 0x1F1E6;
```

Trata os cinco símbolos especiais de letras com um pequeno `switch`. Depois de um desses símbolos, consome U+FE0F se estiver presente. Neste problema, os indicadores regionais não admitem esse seletor.

Para os algarismos, aceita o algarismo ASCII isolado, seguido de U+20E3, ou seguido de U+FE0F e U+20E3. Um seletor depois de um algarismo exige o ponto de código de tecla a seguir. Um seletor ou uma marca de tecla isolados não têm significado próprio.

Um invariante útil é: **cada conversão bem-sucedida consome um símbolo permitido completo e produz exatamente um carácter ASCII**. Os seletores e as marcas de tecla não produzem caracteres adicionais. A solução descodifica primeiro a linha para pontos de código e percorre depois esse vetor, consultando os elementos seguintes quando necessário.

Interpretar a pauta descodificada

Remove CR apenas quando antecede um terminador LF. A última linha pode não ter terminador, mas um CR isolado não é um terminador de linha. Descodifica a linha e remove depois os espaços ASCII permitidos no final.

Os nomes podem ter espaços simples, enquanto o separador tem pelo menos dois. Assim, a primeira sequência de dois ou mais espaços identifica inequivocamente a fronteira entre nome e nota. Separar por todos os espaços dividiria incorretamente um nome como `ANA MARIA`.

A primeira linha não vazia tem de conter os dois campos do cabeçalho. Nas linhas dos alunos, valida o comprimento e o alfabeto do nome, o tamanho do separador e o intervalo da nota. Converte a nota para inteiro para que `07` seja escrito como `7`.

Regista se já começaram as linhas vazias após a pauta. A partir daí, uma nova linha não vazia é inválida: não podem existir linhas vazias entre alunos. Antes do cabeçalho, essas linhas são permitidas. A implementação valida antes de escrever e envia os diagnósticos para a saída de erro. O juiz promete dados válidos, pelo que esses diagnósticos não fazem parte da resposta exigida.

Guardar os registos e alinhar a tabela

A largura final depende do maior nome, que pode surgir apenas na última linha. Guarda cada nome e nota enquanto atualizas $W = \max(4, \text{comprimentos de todos os nomes descodificados})$.

Os nomes de saída são ASCII; por isso, `strlen` mede corretamente o seu número de caracteres. Medir os bytes dos emoji antes da conversão daria uma largura incorreta.

Para `ANA` e `LUIS`, $W = 4$. ANA recebe um espaço de preenchimento seguido dos dois espaços de separação:

```
NOME  NOTA
ANA   20
LUIS  7
Cumprimentos algorítmicos!
```

Em C, o alinhamento à esquerda de `%-*s` fornece o preenchimento:

```c
printf("%-*s  NOTA\n", (int)width, "NOME");
printf("%-*s  %d\n", (int)width, name, grade);
```

Não acrescentes espaços depois da nota. Escreve LF no fim de todas as linhas, incluindo a frase final.

Complexidade

Sejam B o número de bytes de entrada, O o número de bytes de saída, N o número de alunos e L o comprimento máximo de um nome. A descodificação, a conversão e a interpretação fazem um número limitado de passagens por cada linha. O tempo total é $O(B + O)$ e o espaço é $O(NL)$, além de uma linha de entrada com tamanho limitado. A solução fornecida suporta 10000 alunos e nomes de 80 caracteres através de buffers fixos.

Não é necessário inverter a semente 42. Esta afeta as escolhas do conversor e o preenchimento, cuja interpretação já está definida no enunciado. O gerador de testes calcula a saída esperada independentemente, a partir dos registos de alunos em texto simples.

Compilar e verificar

```
cc -std=c17 -O2 -Wall -Wextra -Werror -pedantic \
  solution.c -o notas-emoji
./notas-emoji < samples/input.txt > actual.txt
diff -u samples/output.txt actual.txt
python3 test.py
```

Para gerar um caso maior:

```
python3 generate.py --seed 42 --count 10000 \
  --varied --output generated
./notas-emoji < generated/input.txt > actual.txt
diff -u generated/output.txt actual.txt
```

Verifica todas as letras, as duas formas de tecla, os seletores opcionais, as notas 0 e 20, os nomes compostos e de 80 caracteres, CRLF, a ausência de terminador final e as linhas vazias nas margens. UTF-8 inválido e seletores em posições não permitidas são bons testes defensivos. Para a pauta de vinte alunos, compara com `samples/classroom-output.txt`.