sqtpm
[ggm]

ajuda
voltar

Trabalho: 26-hashing

Data de abertura: 16/06/2025 17:00:00
Data limite para envio: 06/07/2025 23:59:00 (aberto)

Número de arquivos a enviar: entre 1 e 3

Número máximo de envios: 30
Número de envios feitos por ggm: 0

Casos-de-teste abertos: casos-de-teste.tgz

Recursos: cpu 2 s, memória 32768 kb, stack 8192 kb

Enviar:

Linguagem: C Arquivos: Browse... No files selected. Enviar

Hashing de cadeias

Uma aplicação precisa manter um glossário que registra a ordem em que um conjunto de cadeias foram adicionadas a ele.

Implemente uma tabela de hashing para armazenar o glossário dessa aplicação. A tabela deve armazenar pares formados por uma cadeia e um timestamp. Um timestamp é um número natural único e seqüencial, a partir de 0 e sem reuso que indica o momento que a cadeia foi inserida na tabela.

As colisões devem ser resolvidas usando sondagem com incremento por hashing dupla.

Escolha uma função de transformação de cadeias de caracteres em naturais. Para este trabalho não importa a forma como você faça o mapeamento de cadeias para naturais, mas algumas funções são muito piores que outras em termos de espalhamento e podem fazer com que nem todos os casos-de-teste sejam bem-sucedidos.

Um exemplo de uma função que funciona bem na prática é a função djb2:

```
unsigned long djb2(unsigned char *str) {
  unsigned long hash = 5381;
  int c;

  while ((c = *str++))
    hash = ((hash << 5) + hash) ^ c; // hash * 33 XOR c

  return hash;
}
```

A djb2 faz um mapeamento bem espalhado de strings em unsigned long. A saída da djb2 será a entrada para a função de hashing.

Entrada

Cada linha da entrada para o programa é um comando. Os comandos podem ser

- `c n`

  Seu programa deve criar uma tabela vazia para armazenar n pares. O tamanho da tabela deve ser tal que a tabela não fique mais que 70% ocupada quando tiver n pares. Se já houver uma tabela, ela deve ser descartada e deve dar lugar à nova tabela vazia e o timestamp deve ser reiniciado com zero.

- `i cadeia`

  Seu programa deve inserir a cadeia na tabela com o timestamp corrente. Cada cadeia deve ser inserida uma única vez na tabela. Se houver um comando de inserção para uma cadeia que já está na tabela, a tabela deve permanecer como estiver.

  Depois de uma inserção o timestamp deve ser incrementado uma unidade.

- `r cadeia`

  Seu programa deve remover a cadeia da tabela.

- `b cadeia`

  Seu programa deve imprimir uma mensagem indicando se ela está ou não na tabela, no formato

exemplificado abaixo.

- `f`

  Seu programa deve terminar. Qualquer memória alocada dinamicamente deve ser liberada antes do término.

Nos comandos, o caractere é separado do próximo parâmetro por um único espaço.

As cadeias são formadas por pelo menos 1 e por até 250 caracteres do alfabeto ASCII entre 33 e 126 inclusive. Observe que uma cadeia pode começar ou terminar com um ou mais espaços e nesses casos os espaços fazem parte da cadeia.

Saída

A saída do programa é formada pela saída dos comandos b.

Exemplos

Entrada

`c 32`
`i It doesnt matter`
`i the way that you take it,`
`i It doesnt matter`
`i the way that you make it,`
`i Love only matters,`
`i its never a crime,`
`i And if you think you can choose,`
`i you won't find.`
`b it doesnt matter`
`b It doesnt matter`
`b the way that you make it,`
`r the way that you make it,`
`b the way that you make it,`
`r the way that you make it,`
`f`

Saída

`[it doesnt matter] nao esta na tabela`
`[It doesnt matter] esta na tabela, timestamp 0`
`[the way that you make it,] esta na tabela, timestamp 2`
`[the way that you make it,] nao esta na tabela`

Entrada

`c 16`
`i oi`
`i Oi`
`i oi`
`i  oi`
`i oi`
`b oi`
`b  oi`
`b   oi`
`b oi`
`f`

Saída

`[oi] esta na tabela, timestamp 0`
`[ oi] esta na tabela, timestamp 2`
`[  oi] nao esta na tabela`
`[oi ] esta na tabela, timestamp 3`

Requisitos adicionais

- Não pode haver qualquer variável global. Uma variável é global se estiver declarada fora de alguma função (variáveis declaradas dentro da main não são globais, são locais à função main).

Sobre organização do código e comentários

- Faça um programa organizado, bem indentado e que seja fácil de ler.
- Adicione comentários que vão ser úteis para entender o programa se você for relê-lo daqui a alguns anos: comentar cada linha vai ser redundante; documentar blocos de código e a estratégia usada na solução vai ser muito útil.
