sqtpm
[256352]

voltar

Trabalho: 07-dinamico
Linguagens: C
Data de abertura: 2019/09/26 10:00:00
Data limite para envio: 2019/10/02 23:59:59 (encerrado)

Último envio: 29% em 2019/10/02 23:00:56
Envios: 15

Vetor dinâmico

Vamos dizer que um conjunto de registros está em *seqüência* se a ordem relativa entre eles é importante. Por exemplo, pessoas em uma fila de atendimento formam uma seqüência.

Neste trabalho você deve implementar um vetor dinâmico para armazenar uma seqüência de números inteiros. O vetor dinâmico deve permitir inserção no início e no final da seqüência, remoção do primeiro e do último elementos e recuperação do primeiro e do último elementos.

A seqüência deve ser implementada de forma circular, isto é, o primeiro elemento dela deve poder estar em qualquer posição do vetor. Dessa forma, qualquer uma dessas operações acima poderá ser realizada em tempo constante, não é necessário fazer deslocamento dos dados armazenados no vetor para a esquerda ou para a direita.

Por exemplo, a seqüência circular de chaves inteiras [2,9,5,7] pode estar armazenada de várias formas em um vetor de tamanho 8:

```
2 9 5 7 _ _ _ _
_ _ 2 9 5 7 _ _
_ _ _ _ 2 9 5 7
5 7 _ _ _ _ 2 9
9 5 7 _ _ _ _ 2
```

Em todas elas, o primeiro elemento da seqüência é o 2 e o último é o 7. Observe que os números não estão ordenados, mas a ordem relativa entre eles deve ser preservada. Eles podem representar o número da conta das tais pessoas na fila de atendimento, por exemplo.

Se um vetor circular contém a seqüência

```
2 9 5 7 _ _ _ _
```

e o número 6 é adicionado ao início da seqüência, então o vetor fica assim:

```
2 9 5 7 _ _ _ 6
```

sqtpm
[256352]

voltar

Depois se o número 8 é adicionado ao início da seqüência, o vetor deve ficar assim:

```
2 9 5 7 _ _ 8 6
```

Se o número 7 é removido do fim da seqüência, o vetor deve ficar assim:

```
2 9 5 _ _ _ 8 6
```

E se o número 1 é adicionado ao fim da seqüência, o vetor deve ficar assim:

```
2 9 5 1 _ _ 8 6
```

A política de redimensionamento do vetor deve ser dobrar quando estiver cheio e reduzir à metade quando estiver 1/4 ocupado. Dessa forma, durante o processamento, o número de posições vazias do vetor não deveria exceder 3n, onde n é o número de posições ocupadas.

Obviamente, quando o vetor for redimensionado, o conteúdo da seqüência e a ordem relativa entre os elementos da seqüência devem ser preservados.

Entrada

A entrada é composta por uma sucessão de comandos, um por linha. Os possíveis comandos estão descritos abaixo.

- `insert-first inteiro`

  Insere um inteiro no início da seqüência.

- `remove-first`

  Remove o inteiro no início da seqüência. Se a seqüência estiver vazia, não faz nada.

- `print-first`

  Imprime o inteiro no início da seqüência. Se a seqüência estiver vazia, não faz nada.

- `insert-last inteiro`

  Insere um inteiro no fim da seqüência.

- `remove-last`

  Remove o inteiro no fim da seqüência. Se a seqüência estiver vazia, não faz nada.

- `print-last`

  Imprime o inteiro no fim da seqüência. Se a seqüência estiver vazia, não faz nada.

- `is-empty`

  Imprime "yes" se a seqüência estiver vazia e "no" se não estiver.

- `exit`

  Termina o programa.

Nenhuma função deve mover os elementos do vetor para a direita ou para a esquerda.

Saída

A saída deve conter as linhas geradas pelos comandos print-first, print-last e is-empty.

Exemplo

Entrada:

```
is-empty
insert-last 101
insert-last 51
insert-last 13
is-empty
print-last
print-first
remove-first
remove-last
print-first
print-last
exit
```

Saída:

```
yes
no
13
101
51
51
```

Requisitos adicionais:

- Cada funcionalidade deve ser implementada por uma função separada.
- A leitura da entrada deve ser feita na função main, que chama as funções para cada funcionalidade.
- Antes de terminar o programa deve liberar a memória ocupada pelo vetor dinâmico.

Sugestões:

- Comece fazendo uma função main que lê a entrada e apenas imprime uma mensagem para cada comando. Depois implemente as funcionalidades para cada operação no vetor dinâmico.

para cada operação no vetor dinâmico.

sqtpm
[256352]

voltar

- Uma boa forma de organizar programas desse tipo é usando uma struct para representar a estrutura de dados. Nesse struct ficam todos os dados que compõem o vetor dinâmico, como o array, os índices para o início e fim, o tamanho e outros que forem necessários.
- É interessante organizar o programa em três arquivos, um .h com as declarações de tipos e funções que manipulam o vetor dinâmico, um .c com as implementações das funções do vetor dinâmico e um outro .c com a função main e outras funções não relacionadas com o vetor dinâmico.
