sqtpm
[jhdapr]

voltar

**Trabalho:** 21-sequencia
Data de abertura: 23/09/2021 14:00:00
Data limite para envio: 30/09/2021 16:00:00 (aberto)
Número máximo de envios: 31
Casos-de-teste abertos: casos-de-teste.tgz
Arquivos-fonte: include.tgz
Arquivos a enviar: entre 1 e 3
Enviar arquivos com nomes: darray.c

**Enviar:**

Linguagem: C
Arquivos: Escolher arquivos Nenhum arquivo selecionado Enviar

---

# Seqüência circular em vetor dinâmico

Vamos dizer que um conjunto de dados forma uma *seqüência* se eles são consecutivos e se a ordem relativa entre eles é importante. Estar em seqüência não quer dizer que seja ordenado. Quer dizer apenas que existe o primeiro, o segundo, o terceiro etc. mas não quer dizer que o primeiro é menor que o segundo, que o segundo é menor que o terceiro etc.

Uma forma de armazenar uma seqüência é colocá-la consecutivamente em um vetor de forma "circular", isto é, o primeiro elemento da seqüência pode estar em qualquer posição.

Por exemplo, se a seqüência de inteiros [2,9,5,7] estiver armazenada de forma circular em um vetor de tamanho 8, ela pode estar em uma das configurações abaixo:

```
2 9 5 7 _ _ _ _
_ 2 9 5 7 _ _ _
_ _ 2 9 5 7 _ _
_ _ _ 2 9 5 7 _
_ _ _ _ 2 9 5 7
7 _ _ _ _ 2 9 5
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, mas a posição dos elementos dentro do vetor é diferente. Nas configurações, o sublinhado representa inteiros que fazem parte do vetor mas não fazem parte da seqüência.

Se um vetor de tamanho 8 contém a seqüência circular [2,9,5,7] na configuração

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

e o número 6 é adicionado ao início da seqüência, então o

# sqtpm
[jhdapr]

voltar

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

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

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
```

Se soubermos quais são os índices do início e o tamanho da seqüência (ou equivalentemente soubermos os índices do início e do fim da seqüência) então as operações de inserção e remoção nas extremidades da seqüência podem ser realizadas facilmente.

Por exemplo, supondo que a seqüência abaixo esteja armazenada em um vetor A de tamanho 10, com início na posição 8 e tamanho 5.

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

O primeiro elemento está em A[8] e o último está em A[(8+5-1)%10] = A[2].

Como outro exemplo, supondo que a seqüência abaixo esteja armazenada em um vetor V de tamanho 13, com início na posição 3 e tamanho 1.

```
_ _ _ 7 _ _ _ _ _ _ _ _ _
```

O primeiro elemento está em V[3] e o último está em V[(3+1-1)%13] = V[3].

Neste trabalho você deve implementar funções para manipular uma seqüência de floats, que devem permitir recuperar o primeiro e o último float, inserir no início e no final da seqüência e remover o primeiro e a último float.

A seqüência deve estar armazenada de forma circular em um vetor alocado dinamicamente. O vetor deve ser redimensionado e aumentar quando estiver cheio. Ele deve diminuir quando estiver com muitas posições não-ocupadas por elementos da seqüência.

O vetor deve começar com tamanho 64. A política de redimensionamento do vetor deve ser dobrar quando estiver cheio e reduzir à metade quando estiver 1/4 ocupado. O vetor nunca deve ficar com menos que 64

sqtpm
[jhdapr]

voltar

elementos. Dessa forma, durante o processamento, o número de posições vazias do vetor não deve exceder 3n, onde n é o número de posições ocupadas pelos elementos da seqüência. Antes de terminar o programa deve liberar a memória ocupada pelo vetor alocado dinamicamente.

## Entrada

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

- `inject x`

  Insere o fracionário x no início da seqüência. x tem no máximo 6 dígitos decimais.

- `eject`

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

- `push x`

  Insere o fracionário x no fim da seqüência.

- `pop`

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

- `print-first`

  Imprime o fracionário no início da seqüência usando conversão %g, como uilustrado abaixo. Se a seqüência estiver vazia, não faz nada.

- `print-last`

  Imprime o fracionário no fim da seqüência usando conversão %g, como uilustrado abaixo. Se a seqüência estiver vazia, não faz nada.

- `is-empty`

  Imprime empty se a seqüência estiver vazia e non-empty se não estiver.

- `exit`

  Termina o programa.

## Saída

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

sqtpm
[jhdapr]

voltar

## Exemplo

**Entrada**

```
push 3.14159
inject 2.71828
inject 1.4142
push 1.732
push 1.618034
is-empty
print-first
print-last
eject
pop
print-first
print-last
is-empty
pop
pop
eject
eject
is-empty
exit
```

**Saída**

```
non-empty
first: 1.4142
last: 1.61803
first: 2.71828
last: 1.732
non-empty
empty
```

## Envio

O programa deve ser composto por um arquivo main.c e por um par de arquivos darray.h e darray.c. O darray.h tem as definições do array e das funções que manipulam o array. O darray.c deve ter as definições das funções.

Você deve enviar ao sqtpm **apenas o arquivo darray.c que você implementar**. Os arquivos main.c e darray.h **estão prontos** e podem ser baixados pelo sqtpm. O darray.h descreve o comportamento esperado das funções. Eles serão usados automaticamente pelo sqtpm nesta forma.

Seu arquivo darray.c pode ter outras funções se você julgar conveniente. Elas serão visíveis apenas dentro do próprio arquivo darray.c. No arquivo main.c apenas as que estão declaradas em darray.h são visíveis.

Antes de sair programando o seu darray.c, use alguns minutos para entender o main.c e o darray.h.

Para compilar o programa no seu computador, compile primeiro o darray.c:

sqtpm
[jhdapr]

voltar

```
gcc -c darray.c
```

Depois compile tudo junto:

```
gcc main.c darray.o -o darray
```

Se você usa um IDE, vai precisar criar um projeto e incluir os três arquivos.

## Nota

Essa estrutura de dados seqüência com as operações push(), pop(), inject() e eject() também é chamada de double ended queue ou deque.

## 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.

---