Terceira Atividade Prática

Estrutura de Dados 1

Prof. Paulo Henrique Ribeiro Gabriel

Palíndromos

Um palíndromo é "uma palavra ou frase que tem a mesma sequência de letras em qualquer ordem de leitura, seja da esquerda para a direita, seja da direita para a esquerda"[1]. Como exemplos, podemos considerar as palavras "ovo" e "sopapos". É possível estender esse conceito também para frases: nesse caso, devemos desconsiderar alguns espaços, além de sinais de pontuação, acentos e outros símbolos. Um exemplo de frase palíndroma é "Socorram-me, subi no ônibus em Marrocos".

Neste trabalho, você deverá criar um programa em C que recebe como entrada uma *string* de tamanho arbitrário e verifica se ela é um palíndromo. Para isso, você pode, primeiro, pegar cada caractere da *string*, inserir em uma fila e, também, colocar esse mesmo caractere em uma pilha. Feito isso, é necessário retirar um elemento da fila (ou seja, o primeiro caractere da fila) e o caractere do topo da pilha, comparando-os para ver se são iguais. Se forem iguais, continue esse processo de desenfileiramento, desempilhamento e comparação até que ambas as estruturas de dados estejam vazias. Caso ocorra uma não correspondência, significa que a *string* não é um palíndromo.

Tarefa

Escreva as seguintes declarações e implementações:

- Dois tipos abstratos de dados: um para sua pilha e outro para sua fila.
- Uma função `push` que insere um caractere na pilha.
- Uma função `enqueue` que insere um caractere na fila.
- Uma função `pop` que desempilha e retorna o caractere no topo da pilha.
- Uma função `dequeue` que desenfileira e retorna o primeiro caractere na variável da fila.

Obviamente, você deverá criar outras funções auxiliares, tanto para tratamento de *strings* quando para verificações adicionais (por exemplo, para descobrir se a pilha ou a fila estão vazias). Você pode usar tanto estruturas sequenciais (*arrays*) quando encadeadas. Em ambos os casos, é obrigatório o uso de alocação dinâmica.

Formato de entrada

A entrada será composta por uma *string* de caracteres ASCII minúsculos, sem acentos, hifens, pontuação ou qualquer outro símbolo: apenas letras e espaços. Cada arquivo de entrada será composto por apenas uma **string**. Alguns exemplos de entrada são mostrados a seguir:

```
arara
```

```
ufu
```

```
socorram me subi no onibus em marrocos
```

```
was it a car or a cat i saw
```

Observe que espaços não fazem parte do palíndromo; portanto, você deve tratá-los de alguma maneira. É possível, por exemplo, ler a *string*, remover todos os espaços (criando uma nova *string*) e, só então, inserir os caracteres nas estruturas de dados. Alternativamente, você pode inserir os caracteres nas estruturas diretamente, ignorando os espaços. Caso tenha alguma ideia diferente dessas, sinta-se à vontade para implementá-la; tudo, no entanto, deve estar bem organizado em funções e comentado.

Formato de saída

A saída será composta por apenas um valor inteiro: 1, caso a *string* seja um palíndromo, ou 0 caso contrário. O inteiro deve ser seguido de uma quebra de linha (`\n`).

Por exemplo, se a *string* de entrada for `socorram me subi no onibus em marrocos`, a saída deverá ser:

```
1
```

Já no caso da frase `o rato roeu a roupa do rei de roma` a saída esperada é:

```
0
```

Observações

1. O trabalho deve ser feito individualmente e em Linguagem C.
2. O código-fonte de ver submetido, obrigatoriamente, ao sistema run.codes. Não serão aceitos trabalhos encaminhados por outros meios.
3. Todas as submissões são checadas para evitar plágio; portanto, evite problemas e implemente o seu próprio código.
4. Comente o seu código com uma explicação rápida do que cada função ou trecho importante de código faz (ou deveria fazer). Os comentários e a modularização do código serão checados e valem nota.

5. Todas as funções devem ser implementadas em um único arquivo `.c`; porém, deve-se ficar atendo ao bom uso de TAD.

6. Entradas e saídas devem obrigatoriamente ser lidas e escritas a partir dos dispositivos padrão, ou seja, use as funções `scanf` (ou similares, como `gets`, entre outras) e `printf`.

7. Lembre-se de respeitar estritamente o formado de entrada e saída. Uma quebra de linha a mais ou a menos resultará em erro no caso de teste.

8. Todas as operações de leitura e escrita de valores devem ser feitas na função `main()` ou em funções próprias para esse fim. Lembre-se: TAD deve ser genérico, ou seja, suas funções devem evitar a escrita de mensagens.

9. Use alocação dinâmica de memória. As *strings* não possuirão tamanho fixo, cabe ao seu programa verificar isso.

Avaliação

A avaliação considerará os seguintes critérios:

(N_1) Funcionamento do programa (avaliado pelo run.codes): 0 a 10.

(N_2) Bom uso de TAD (estruturas e funções bem organizadas, transparentes, que executam funções bem definidas): 0 a 10.

(N_3) Bom uso de funções em geral (funções bem estruturas, uso correto de alocação dinâmica, tratamento adequado de erros): 0 a 10.

(N_4) Clareza do código (bom uso de comentários, nomes adequados de funções, endentação correta): 0 a 10.

Nota final:

$$\frac{N_1 + \frac{N_2+N_3+N_4}{3}}{2}$$

Casos de plágio serão tratados diretamente com os responsáveis.

Referências

1 CARNEIRO, R. O que é um palíndromo? *Super Interessante*, 2018. Disponível em: <https://www.dicio.com.br/lista-palindromos/>.