Tópicos de Programação

Segundo Exercício-Programa (Segunda Versão)

Entrega: 23/01/2017 até 23:55

Verão 2017 - IME-USP

1 Introdução

Conta-se uma história sobre o fim de uma guerra hebraica, na qual o historiador Josephus e mais 40 homens foram cercados pelo exército romano. Eles decidiram não se entregar com vida, mas sua religião condena o suicídio. Por isso, fizeram um pacto no qual os 41 homens fariam um círculo e, começando pelo homem na primeira posição, um homem de cada vez mataria aquele imediatamente à sua esquerda, sendo que o próximo homem a matar é o que estava a esquerda do último a morrer. Nesse contexto, Josephus precisou encontrar em qual lugar no círculo deveria se posicionar para que fosse o último homem que restasse. Essa história dá origem ao **Problema de Josephus** que será o exercício a ser resolvido nesse EP.

Para resolver o problema, utilizaremos o conceito de uma lista ligada circular, por isso será necessário implementar algumas funções que auxiliem a criação e manipulação de listas encadeadas, além da resolução do problema em si.

2 Tarefas

Crie um arquivo ep2.c contendo um cabeçalho com as informações do estudante a implementação da lista ligada circular, a resolução do problema de Josephus e um main que simule e resolva o problema.

2.1 Implementação de Lista Ligada

Implemente as funções de **inserção**, **remoção** e **impressão** de elementos em uma lista ligada circular. Os nós da lista devem seguir a struct **pessoa**, a lista deve ser declarada como **inicio** e a assinatura das funções **devem** ser como mostra o seguinte código:

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct pessoa Pessoa;
struct pessoa {
    int posicao;
    Pessoa *prox;
};
```

```
9
10 Pessoa *inicio;
11 inicio = NULL;
12
13 void insereLista (int x);
14 void removeLista (Pessoa *p);
15 void imprimeLista ();
16 void resolveJosephus(int n, int m);
```

É importante seguir o modelo dado acima para possibilitar a correção. Você pode adicionar o que for necessário, como bibliotecas e alterar a ordem de funções, por exemplo, desde que o que existe nesse modelo continue existindo no ep enviado.

Trabalharemos com uma **lista encadeada circular** chamada *inicio* que é declarada na linha 10 do modelo, Note que *inicio* é uma variável global, logo você pode acessá-la dentro das funções sem a necessidade de passá-la como argumento. Essa lista conterá elementos que representam a posição dos homens no círculo.

2.1.1 Função: insereLista(int x)

A inserção nessa lista deve acontecer através da função *insereLista(int x)*, em que x é o elemento a ser inserido. O elemento deve ser inserido no **final** da lista e, como trabalharemos com uma lista circular, o último elemento da lista deve sempre apontar para o primeiro elemento dela.

2.1.2 Função: removeLista(Pessoa *p)

Quando um homem morrer, ele deve ser retirado da lista, para isso utilizaremos a função *void removeLista(Pessoa *p)* que deve remover o elemento **p→prox**, ou seja, o elemento imediatamente depois de **p**. Perceba também, que a lista não pode conter elementos iguais uma vez que cada pessoa recebe um número diferente de acordo com sua posição.

2.1.3 Função: imprimeLista()

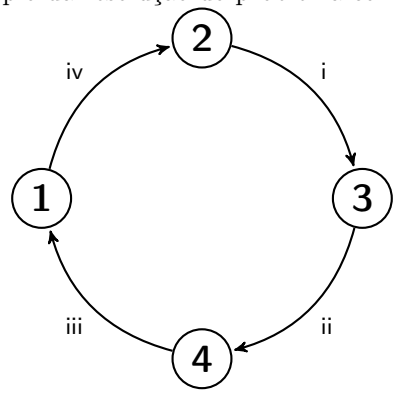
Essa função não recebe nenhum parâmetro e deve **imprimir todos** os elementos presentes na lista naquele momento, começando por aquele apontado por *inicio*.

2.2 Solução do Problema de Josephus*

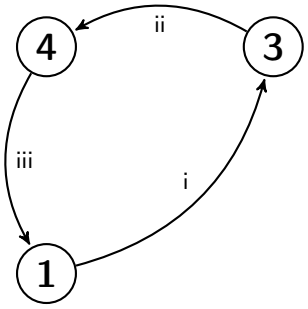
Suponha que n pessoas estão dispostas em um círculo e que são numeradas de 1 a n no sentido horário. Comece pela pessoa de número 1 e elimine a m-ésima pessoa. Em seguida, contando a partir da pessoa p que sucede a última eliminada, elimine a pessoa que está m posições depois de p. Repita este processo enquanto o circulo tiver duas ou mais pessoas. Note que, como estamos falando de um círculo, ao andar m posições a partir de uma pessoa, podemos acabar chegando em pessoas que estão antes dela, ou até mesmo a ela própria. Como a

pessoa nunca pode matar a si própria, neste caso ela deve matar quem estiver imediatamente à frente dela na lista.

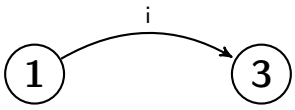
Segue um exemplo da resolução do problema com $n = 4$ e $m = 1$:



A primeira pessoa a ser eliminada é a que se encontra na posição 2.



A próxima pessoa eliminada se encontra na posição 4.



A última pessoa eliminada se encontra na posição 3 e a sobrevivente é que se encontra na posição 1.

Sua função deve receber como parâmetros os inteiros n, que representa o número de pessoas no círculo, e m, que representa o tamanho do passo (número de pessoas puladas no círculo) para a remoção do próximo. Ela também deve **imprimir**, separado por espaços, o número de cada homem que é morto, e, no final imprimir o sobrevivente.

* Enunciado adaptado do livro Projeto de Algoritmos

3 Informações importantes

O programa entregue deve ser seguir os itens abaixo:

- O EP deverá ser feito individualmente e plágio não será tolerado;
- O cabeçalho de cada função pedida deve seguir o modelo indicado na sua descrição para fins de correção;
- Você pode adicionar mais bibliotecas caso necessite;
- Outras funções auxiliares podem ser criadas para facilitar o desenvolvimento, desde que as funções pedidas no enunciado estejam presentes e se comportem de acordo com o que foi pedido;
- O EP Deve ser entregue no PACA em um .zip contendo o arquivo **ep2.c**;
- Compile o seu EP usando o compilador GCC;
- Utilize as flags de compilação: -Wall -ansi -pedantic -O2
- Bom desenvolvimento e divirta-se!

4 Referências interessantes

- Josephus Problem - Wikipedia `https://en.wikipedia.org/wiki/Josephus_problem`
- Vídeo interessante sobre o problema: The Josephus Problem - Numberphile (YouTube) `https://goo.gl/MVNcB6`
- Site Projeto de Algoritmos `https://www.ime.usp.br/~pf/algoritmos/`