

Algoritmos e Estruturas de Dados 1

Segundo Trabalho Prático

Entrega Sugerida: 01/10/2020

ENPE/2020 - DC-UFSCar

1 Introdução

No segundo trabalho prático TP02 será solicitada a entrega de um programa que solucione o problema apresentado na próxima seção. O arquivo entregue deve seguir os itens abaixo:

- O TP02 deverá ser feito individualmente e plágio não será tolerado;
- O cabeçalho de cada função pedida deve seguir o modelo indicado na sua descrição;
- 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 TP02 deve ser entregue no run codes (<https://run.codes>) em um arquivo contendo código em linguagem C e com um cabeçalho com as informações do estudante (nome, curso, RA);
- Cada estudante deve se cadastrar no run codes (<https://run.codes>) informando Nome Completo, escolhendo “UFSCar - Universidade Federal de São Carlos” no campo Universidade e colocando seu RA no campo Número da Matrícula. Depois de cadastrado, basta logar no run codes e se matricular na disciplina “1001502 - Algoritmos e Estruturas de Dados 1”, Professores: Marcela Xavier Ribeiro, Turma: 1-2020-prof. Marcela. Usando o código da Matrícula: SCL5

2 Problema de Josephus

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 à 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 trabalho. O vídeo “The Josephus Problem - Numberphile” é bastante interessante e explora mais a fundo este problema - <https://goo.gl/MVNcB6>.

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.

3 Tarefas

Crie um arquivo trabalho2.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.

3.1 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:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct pessoa Pessoa;
5 struct pessoa {
6     int posicao ;
7     Pessoa * prox;
8 };
9
10 Pessoa * inicio;
11 inicio = NULL ;
12
13 void insereLista(int pos);
14 void removeLista(Pessoa *p);
15 void imprimeLista();
16 int 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, desde que tudo desse modelo continue existindo no trabalho 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.

3.1.1 Função: `insereLista(int pos)`

A inserção nessa lista deve acontecer através da função `insereLista(int pos)`, em que `pos` é a posição do elemento a ser inserido. Como trabalharemos com uma lista circular, o último elemento da lista deve sempre apontar para o primeiro elemento dela.

3.1.2 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`.

3.1.3 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*.

3.1.4 Função: `main()`

A função `main` deve ler o numero de execuções do problema de Josephus e posteriormente os inteiros `m`, representando o número de pessoas, e `p`, representando o tamanho do passo, e devolver o resultado de cada execução. Para isso, deve-se usar a seguinte implementação do programa `main`.

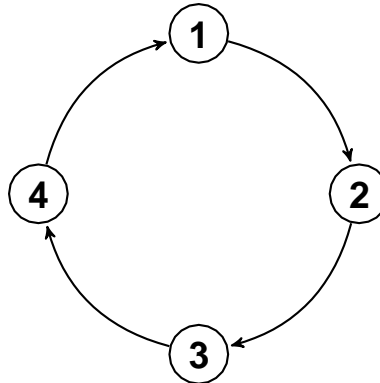
```
1 int main (){
2     int nroexecs;
3
4     scanf("%d", &nroexecs);
5     int *n = malloc(nroexecs * sizeof(int));
6     int *m = malloc(nroexecs * sizeof(int));
7
8     for(int i = 0; i < nroexecs; i++){
9         scanf("%d", &n[i]);
10        scanf("%d", &m[i]);
11    }
12
13    for(int i = 0; i < nroexecs; i++){
14        printf("Usando n=%d, m=%d, resultado=%d\n", n[i], m[
15        i], resolveJosephus(n[i], m[i]));
16    }
17    return 0;
18 }
```

3.1.5 Função `resolveJosephus(int n, int m)`

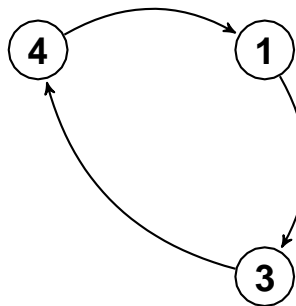
Suponha que n pessoas estão dispostas em um círculo e que são numeradas de 1 a n no sentido horário. Comece pela pessoa de número 1 e elimine a

($m+1$)-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 círculo 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, já que 3 sucede o último eliminado e $m = 1$ posições depois de 3 está o 4.



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

A função `int resolveJosephus(int n, int m)` 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 deve devolver a posição do sobrevivente.

4 Exemplo de caso de testes

Entrada:

```
1 3
2 10
3 1
4 10
5 2
6 10
7 3
```

Saída:

```
1 Usando n=10 , m=1 , resultado=5
2 Usando n=10 , m=2 , resultado=10
3 Usando n=10 , m=3 , resultado=6
```

5 Referências interessantes

Josephus Problem - Wikipedia: https://en.wikipedia.org/wiki/Josephus_problem

The Josephus Problem - Numberphile (vídeo) <https://goo.gl/MVNcB6>

Site Projeto de Algoritmos: <https://www.ime.usp.br/~pf/algoritmos/>