MAC0110 – Introdução à Computação

BCC IME/USP 2022

Segundo Exercício–Programa – Data de entrega: até 04/06/2022

1 Introdução

Suponha uma cidade fictícia com veículos autônomos. Nessa cidade, há uma área protegida (AP) na qual os veículos não podem entrar. Para monitorar a AP, foram instaladas antenas em diferentes pontos da cidade. Cada uma das antenas possui uma identidade (um número inteiro) e sua posição absoluta (coordenadas (x, y) em metros) e elas transmitem sinais. Cada veículo autônomo é equipado com um dispositivo receptor que é capaz de detectar esses sinais vindos das antenas. Um esquema desse cenário é mostrado na figura 1. Um dispositivo desses é capaz de calcular sua distância H (em metros) até o transmissor da antena e também o ângulo de incidência θ (em graus) do sinal vindo da antena, e esses valores são automaticamente transmitidos para uma central de monitoramento.

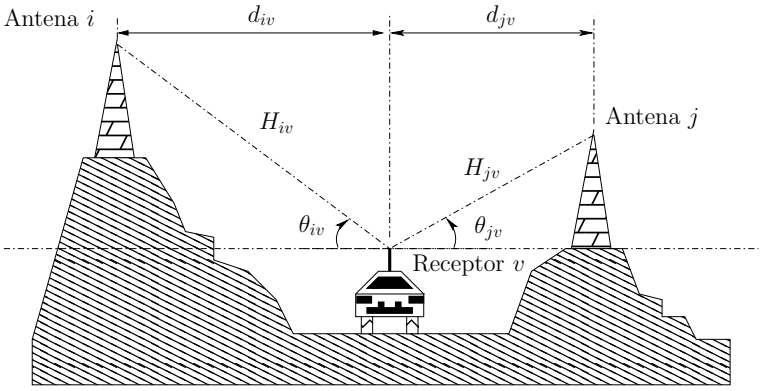

Figura 1: Um veículo (v), duas antenas (i e j), e as distâncias (H_{iv} e H_{jv}) e ângulos (θ_{iv} e θ_{jv}) do veículo v até o transmissor das antenas i e j. As distâncias, no plano, do veículo v até a base das antenas i e j são denotadas por d_{iv} e d_{jv}.

O objetivo deste EP é escrever um programa (em linguagem C) que será usado pela central de monitoramento.

2 Detalhes do sistema de monitoramento

O **sistema de monitoramento**, a ser instalado na central de monitoramento, deverá processar as informações enviadas pelos veículos. Especificamente, ele deve calcular a localização e velocidade dos veículos e, dependendo da situação, emitir alertas correspondentes.

2.1 Localização do veículo

A rede de monitoramento é planejada de forma que pelo menos três sinais de antenas cheguem a qualquer ponto na cidade. Estamos supondo que cada veículo envia, periodicamente, informações

referentes a três antenas, digamos i, j e k, para a central. Descrevemos aqui como a central poderá calcular a posição absoluta do veículo.

Sejam (x_i, y_i) as coordenadas absolutas de uma antena i. Então, a distância d_{iv} da base da antena i ao veículo v (no plano do veículo) é dada por (veja figura 1):

$$d_{iv} = H_{iv} \cos \theta_{iv}. \tag{1}$$

Se as coordenadas absolutas do veículo (x_v, y_v) fossem conhecidas, a distância d_{iv} entre a antena i e o veículo v poderia também ser expressa como:

$$\begin{aligned} d_{iv}^2 &= (x_i - x_v)^2 + (y_i - y_v)^2 \\ &= x_i^2 + x_v^2 + y_i^2 + y_v^2 - 2x_i x_v - 2y_i y_v \end{aligned} \tag{2}$$

Porém, as informações de uma antena permitem calcular apenas a distância do veículo em relação a sua base. Para determinar a posição absoluta (x_v, y_v) de um veículo v, precisamos usar a informação de 3 antenas i, j e k, não colineares, combinadas em um sistema de equações.

Considere então o sistema formado pelas equações de distância d_{iv}^2, d_{jv}^2 e d_{kv}^2. Ao tomarmos a diferença de duas equações, digamos para as antenas i e j, temos que

$$d_{iv}^2 - d_{jv}^2 = x_i^2 - x_j^2 + y_i^2 - y_j^2 - x_v(2x_i - 2x_j) - y_v(2y_i - 2y_j). \tag{3}$$

Por exemplo, para as antenas 1 e 2, teríamos a equação

$$d_{1v}^2 - d_{2v}^2 = x_1^2 - x_2^2 + y_1^2 - y_2^2 - x_v(2x_1 - 2x_2) - y_v(2y_1 - 2y_2).$$

Observe a eliminação dos termos quadráticos em x_v e y_v. Colocando agora a variável x_v em evidência, obtém-se que

$$\begin{aligned} x_v &= \frac{\left(x_i^2 - x_j^2 + y_i^2 - y_j^2 - y_v(2y_i - 2y_j) - d_{iv}^2 + d_{jv}^2\right)}{(2x_i - 2x_j)} \\ &= \underbrace{\frac{\left(x_i^2 - x_j^2 + y_i^2 - y_j^2 - d_{iv}^2 + d_{jv}^2\right)}{2(x_i - x_j)}}_{p_{ij}} + \underbrace{\frac{(y_j - y_i)}{x_i - x_j}}_{q_{ij}} y_v \end{aligned} \tag{4}$$

Ao fazermos o mesmo para outro par de antenas, por exemplo i e k, obtemos os coeficientes p_{ik} e q_{ik}. Então, podemos obter as coordenadas absolutas (x_v, y_v) do veículo resolvendo o sistema

$$\begin{cases} x_v = p_{ij} + q_{ij}\, y_v \\ x_v = p_{ik} + q_{ik}\, y_v \end{cases} \tag{5}$$

Note que, na equação 4, os denominadores de p_{ij} e q_{ij} podem eventualmente ser nulos. Quando as três antenas não são colineares, é sempre possível evitar a ocorrência de denominador nulo na equação 4. Basta selecionarmos como i uma antena cuja coordenada na abscissa seja diferente das coordenadas na abscissa das outras duas antenas.

2.2 Cálculo de velocidade

Suponha que de acordo com o cálculo acima um veículo esteja localizado em (x_0, y_0) num certo instante t qualquer e em (x_1, y_1) no instante $t + \Delta T$. Essa informação é suficiente para o sistema estimar a velocidade e direção do veículo. (Iremos supor que os veículos só fazem trajetórias retilíneas e em velocidade uniforme, para não complicar o EP.)

Supondo que ele percorreu nesse intervalo de tempo (em segundos) uma distância ΔS em uma trajetória retilínea, a velocidade média do veículo pode ser calculada por:

$$V = \frac{\Delta S}{\Delta T}\, m/s \tag{6}$$

Se a velocidade média for nula, então o **veículo está estacionado** e, caso contrário, o **veículo está em movimento**.

2.3 Situação do veículo

A situação de um veículo relaciona-se com a região AP a ser monitorada. Conforme mostrado na figura 2, a AP é uma região circular de raio 200 metros ($r = 200\, m$) com o centro localizado exatamente na origem do plano (coordenadas $(0, 0)$). Portanto, ela pode ser definida pela circunferência dada pela equação:

$$x^2 + y^2 = r^2 \tag{7}$$

- **Zona de alerta**: é a região formada pelos pontos exteriores a AP cuja distância à origem (centro de AP) é menor ou igual a 2000 metros.
- **Invasão:** Dizemos que um veículo **invadiu a AP** se ele se encontra na AP (ou seja, se a distância da localização atual à origem é menor ou igual a 200 metros).
- **Ameaça:** Dizemos que um veículo representa uma **ameaça de invasão** se sua localização atual está na zona de alerta e se, ao prosseguir em movimento retilíneo uniforme, o veículo atingir a AP em no máximo 60 segundos.

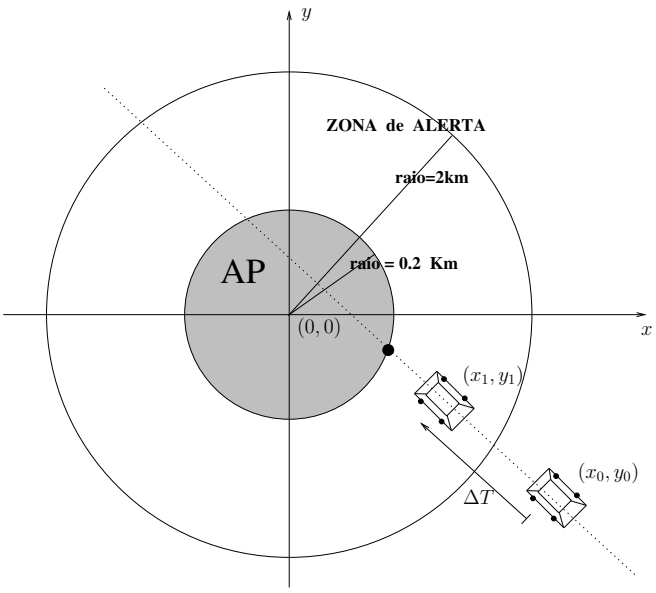

Figura 2: Representação da AP ocupando uma área circular de $raio = 200\, m$ e centro $(0, 0)$ e zona de alerta correspondente ao círculo de raio=2km com centro também na origem. Duas posições (x_0, y_0) e (x_1, y_1) de um veículo, em instantes de tempo separados pelo intervalo de tempo ΔT segundos.

Para determinar se a trajetória de um veículo interceptará a AP, deve-se primeiramente obter a reta definida pelos pontos (x_0, y_0) e (x_1, y_1). A reta que passa por dois pontos distintos (x_0, y_0) e

(x_1, y_1) pode ser dada por uma equação na forma $ax + by + c = 0$. Para determinar os coeficientes a, b e c, devemos considerar dois casos:

- **Caso 1**: $x_0 = x_1$

  Neste caso, o veículo se movimenta numa direção paralela ao eixo das ordenadas, ou seja, $x = x_0 = x_1$. Portanto, $b = 0$. Há vários valores possíveis para a e c.

- **Caso 2**: $x_0 \neq x_1$

  Neste caso, a equação da reta pode ser expressa como

$$y - y_0 = m(x - x_0) \tag{8}$$

  em que m é o coeficiente angular (note que se o veículo estiver em uma trajetória paralela ao eixo das abscissas, então $m = 0$ e, portanto, $a = 0$). da reta dado por $m = (y_1 - y_0)/(x_1 - x_0)$. Os coeficientes a, b e c podem ser determinados diretamente dessa equação.

A reta encontrada intercepta a AP se e somente se o seguinte sistema de equações admite uma solução real.

$$\begin{cases} ax + by + c &= 0 \\ x^2 + y^2 &= r^2 \end{cases} \tag{9}$$

Note, entretanto, que apenas verificar se a reta intercepta ou não a AP não é suficiente para se afirmar que a trajetória irá interceptar a AP. A figura 3 ilustra as três possíveis situações nas quais a reta intercepta dois pontos da circunferência que delimita a AP. Apenas a primeira situação pode representar ameaça de invasão.

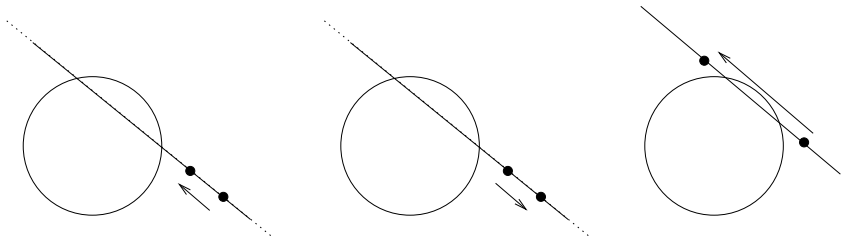

Figura 3: As três possibilidades no caso em que a reta intercepta dois pontos da circunferência da AP.

---

3 O que você deve implementar

Você deverá escrever um programa em linguagem C, que servirá como o sistema de monitoramento. Vamos supor que os sinais transmitidos pelos veículos à central estão em um arquivo, que contém os dados especificados no quadro da figura 4.

O seu EP deverá ler o nome do arquivo que contém os dados a serem processados, e em seguida ler do arquivo o número de casos a serem processados. Veja detalhes de como fazer a leitura de dados de arquivos na seção 4. Para cada caso, deverá executar o processamento descrito nos próximos itens (veja também os exemplos de saída disponibilizados nos anexos).

1. Processamento de cada caso: devem ser lidas a identificação do veículo, as informações referentes à posição prévia, o intervalo de tempo, e as informações referentes à posição atual.

- número de casos a serem analisados
- para cada caso são fornecidos os seguintes dados
  - identificação do veículo (v)
  - sinais relativos à posição prévia (para calcular (x_0, y_0))
    - identificação da primeira antena, coordenadas x e y, comprimento H e ângulo θ
    - identificação da segunda antena, coordenadas x e y, comprimento H e ângulo θ
    - identificação da terceira antena, coordenadas x e y, comprimento H e ângulo θ
  - intervalo ΔT
  - sinais relativos à posição atual (após intervalo de tempo ΔT; para calcular (x_1, y_1))
    - identificação da primeira antena, coordenadas x e y, comprimento H e ângulo θ
    - identificação da segunda antena, coordenadas x e y, comprimento H e ângulo θ
    - identificação da terceira antena, coordenadas x e y, comprimento H e ângulo θ

Figura 4: Conteúdo de um arquivo a ser processado.

Para cada caso, deve ser impressa a identificação do veículo, e para cada instante (prévio e atual), é esperada a impressão (ver exceção no próximo item) da identificação, localização absoluta, distância H, ângulo θ e distância ao veículo (no plano do veículo) de cada antena. Além disso deve ser calculada e impressa a localização absoluta do veículo, caso tal cálculo seja possível.

2. Caso ocorra erro no cálculo de uma das posições do veículo, mensagem adequada sobre a impossibilidade de cálculo da posição e determinação da situação do veículo devem ser impressas, e o restante do procedimento para esse veículo deve ser interrompido. O programa deverá então prosseguir com o tratamento do próximo caso (se houver). Isto significa que, caso o cálculo da posição prévia não seja possível, não se deve imprimir nenhuma informação refrente à posição atual.

3. Calcular e imprimir a distância percorrida e a velocidade média do veículo entre as duas posições (suponha que o veículo sempre realiza um trajeto retilíneo com velocidade uniforme).

4. Imprimir a distância atual do veículo ao centro da AP, o seu estado (em movimento ou estacionado), e a região em que ele está localizado (AP, zona de alerta, ou fora da zona de alerta). Além disso,
   - se o veículo se encontra na AP, o alarme deve ser disparado (imprimir uma mensagem de alarme).
   - se o veículo encontra-se na zona de alerta e se a continuação a partir de (x_1, y_1) da trajetória (retilínea, que parte de (x_0, y_0) e passa por (x_1, y_1)) interceptar a AP, devem ser impressos: a distância da posição atual ao ponto de intersecção na AP, o tempo até o veículo atingir esse ponto na AP e as coordenadas desse ponto.

     Caso seja detectada uma ameaça de invasão (ver seção 2.3, ou seja tempo $\leq$ 60s), o alarme deve ser disparado (imprimir uma mensagem de alarme).

3.1 Mais detalhes sobre a implementação

1. Defina e utilize as seguintes constantes no seu programa:

```
#define PI            3.14159
#define RAIO_AP       200
#define RAIO_ZA       2000      /* zona de alerta */
#define DELTA_ALARME  60
#define EPS_COS       0.000001  /* precisão para cálculo do cosseno */
#define EPS           0.01      /* precisão para valores envolvendo metros */
```

2. Como estamos trabalhando com números reais, podem ocorrer erros devido à precisão dos números. Para comparar se dois números reais são iguais, inclua e use a função

```
int iguais(double x, double y) {
   if(x-y < EPS && y-x < EPS)
     return 1;
   else
     return 0;
}
```

   que devolve 1 se `|x-y| < EPS` e 0 se `|x-y| >= EPS`.

3. Nos exemplos abaixo, mostramos como você pode formatar a impressão de números reais:

```
x = 1.2367;
printf("x = %f\n", x);     /* imprime "x = 1.236700" */
printf("x = %10.3f\n", x); /* Imprime "x =      1.237" */
printf("x = %5.1f\n", x);  /* Imprime "x =   1.2" */
printf("x = %.2f\n", x);   /* Imprime "x = 1.24" */
```

   O primeiro `printf` imprime o valor de `x` com 6 casas depois da "vírgula". O segundo imprime o valor de `x` usando 10 caracteres, incluindo o '.', sendo que 3 deles são depois da vírgula. O terceiro imprime o valor de `x` usando 5 caracteres, incluindo o '.', sendo que 1 deles é depois da vírgula. O quarto imprime o valor de `x` com 2 dígitos depois da vírgula. Note que em todos os casos há arredondamento do dígito menos significativo.

4. Para o cálculo da distância, implemente e utilize obrigatoriamente a função de protótipo

```
double cosseno(double theta, double epsilon);
```

   que recebe um ângulo `theta` em graus e um real `epsilon`, e devolve o cosseno do ângulo `theta` com precisão `epsilon`.

   Para calcular o valor de $cos\, x$ com uma certa precisão ε, utilize a fórmula abaixo e some os termos até que um termo da série seja menor que ε. O primeiro termo menor que ε deve ser somado. **Observe que, nessa série, o ângulo x deve ser em radianos**. O seu programa deve adotar $\varepsilon = 0.000001$, ou seja, este deve ser o valor do parâmetro `epsilon` na chamada da função.

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots + (-1)^k \frac{x^{2k}}{(2k)!} + \cdots \tag{10}$$

5. Para calcular as localização do veículo, implemente e utilize obrigatoriamente a função de protótipo

```
int localiza ( double xi, double yi, double div,
               double xj, double yj, double djv,
               double xk, double yk, double dkv,
               double *xv, double *yv);
```

A função `localiza` devolve em `*xv` e `*yv` as coordenadas absolutas do veículo, calculadas utilizando as coordenadas `(xi,yi)`, `(xj,yj)`, `(xk,yk)` de 3 antenas `i`, `j` e `k` e as distâncias `div`, `djv` e `dkv` do veículo a essas três antenas. Caso existam antenas com uma mesma coordenada nas abscissas, na solução do sistema 5 conforme descrito na seção 2.1), a função deve escolher os pares de forma a evitar divisão por zero.

Se os cálculos para determinar `*xv` e `*yv` não forem possíveis (por exemplo, por conta de divisão por zero), a função deve devolver o valor 0 (zero) e, caso contrário, o valor 1 (um).

6. Para calcular a velocidade média implemente e utilize obrigatoriamente a função de protótipo

   ```
   double velocidade(double x0, double y0, double x1, double y1, double deltaT);
   ```

   que recebe as posições prévia e atual, `(x0,y0)` e `(x1,y1)`, de um veículo e o intervalo de tempo `deltaT` em segundos gasto para percorrer o trajeto entre estas duas localizações. A função deve devolver a velocidade média (em metros por segundo) que deve ser positiva caso o veículo esteja em movimento e nula caso ele esteja estacionado.

7. Para verificar se a trajetória do veículo irá interceptar a AP, implemente e utilize obrigatoriamente a função de protótipo

   ```
   int intercepta(double x0, double y0, double x1, double y1, double *x, double
   *y) ;
   ```

   que recebe as posições prévia `(x0,y0)` e atual `(x1,y1)` do veículo e, caso a continuação da trajetória com início em `(x0,y0)` e passando por `(x1,y1)` intercepte a AP, devolve em `*x` e `*y` as coordenadas do ponto de interseção mais próximas à posição atual do veículo (pode ser as coordenadas da própria posição atual). A função deve devolver valor 1 (um) caso a trajetória intercepte a AP e valor 0 (zero) caso contrário.

8. Caso precise calcular a raiz quadrada de algum número $x \geq 0$, implemente uma função de protótipo

   ```
   double raiz(double x, double epsilon) ;
   ```

   que devolve a raiz quadrada de um número real `x` não-negativo, com precisão `epsilon`.

   Para o cálculo da raiz quadrada, utilize necessariamente a aproximação descrita a seguir. Dado um número real $x \geq 0$, considere a fórmula

   $$\begin{cases} r_0 = x \\ r_i = \dfrac{1}{2}\left(r_{i-1} + \dfrac{x}{r_{i-1}}\right) & i = 1, 2, \cdots \end{cases}$$

   A partir de r_0, obtemos r_1, r_2 e assim por diante. Por exemplo, os termos r_1 e r_2 são calculados da seguinte forma:

$$r_1 = \frac{1}{2}\left(r_0 + \frac{x}{r_0}\right) = \frac{1}{2}\left(x + \frac{x}{x}\right) = \frac{x+1}{2}$$

$$r_2 = \frac{1}{2}\left(r_1 + \frac{x}{r_1}\right) = \frac{1}{2}\left(\frac{x+1}{2} + \frac{2x}{x+1}\right).$$

A raiz quadrada de x, com precisão eps, é o termo r_{n+1} para o menor n que satisfaz $|r_n - r_{n+1}| < eps$. No programa utilize $eps = 10^{-2}$.

Cuidado!!! Não aplique a recorrência para $x = 0$, ou ocorrerá uma divisão por zero!!!

9. Além das funções listadas acima, você pode implementar e usar outras que achar conveniente. **Não é permitida a utilização de funções da biblioteca `math.h`**

4 Manipulação de arquivos

Para usar um arquivo como entrada do seu programa, crie inicialmente uma variável do tipo apontador para FILE. Por exemplo:

```
FILE *arq_entrada;
```

onde `arq_entrada` é o nome da variável. Antes de ler qualquer dado do arquivo, você deve abrí-lo para leitura. Para isso, você deve usar o comando `fopen`. Um exemplo de chamada desta função é

```
arq_entrada = fopen( "entrada.txt", "r" );
```

O primeiro argumento é o nome do arquivo a ser aberto e o segundo argumento `"r"` significa "read", ou seja, que o arquivo será aberto para leitura. O comando `fopen` devolve um apontador que deve ser armazenado na variável do tipo apontador para `FILE` que você criou anteriormente. Se por algum motivo ele não conseguir abrir o arquivo (por exemplo, o arquivo não existe), `fopen` devolve um apontador nulo.

Depois de abrir o arquivo, você pode ler seus dados usando o comando `fscanf`. Ele é similar ao comando `scanf`, com a diferença que o primeiro argumento é o apontador do arquivo. Por exemplo, a chamada

```
fscanf(arq_entrada, "\%d", &num);
```

lê um número inteiro do arquivo e armazena-o na variável `num`.

Ao final da leitura, deve-se sempre fechar o arquivo usando o comando `fclose`.

Veja, no seguinte programa, como utilizar estas funções.

```
/*
 * Este programa lê n e n números inteiros armazenados
 * em um arquivo cujo nome será digitado pelo usuário
 * e imprime os n números na tela.
 */

 #include <stdio.h>
```

```
int main() {
    FILE *arq;
    char filename[256] ;
    int  cont, n, num;

    /* lê o nome do arquivo de entrada */
    printf("Digite o nome do arquivo de entrada: ");
    scanf("%s", filename);

    /* abre o arquivo para leitura */
    arq = fopen(filename, "r");

    /* verifica se ocorreu erro na abertura do arquivo */
    if (arq == NULL) {
        printf("Nao consegui abrir o arquivo %s.\n", filename);
        return 0;
    }

    /* lê a quantidade n de números e, em seguida, os n números */
    fscanf(arq, "%d", &n) ;

    cont = 0 ;
    while (cont < n) {

        /* lê o próximo número */
        fscanf(arq, "%d", &num);

        /* imprime o número lido do arquivo na tela */
        printf("%d\n", num);

        cont++ ;
   }

    /* fecha o arquivo */
    fclose(arq);

    return 0;
}
```

Para testar esse exemplo, crie um arquivo texto contendo os dados a serem lidos e coloque ele na mesma pasta do programa executável.

5 Algumas dicas

- Compile seu programa da seguinte forma:

  ```
  gcc -ansi -Wall -pedantic -o ep2 ep2.c
  ```

- Teste cada uma das funções individualmente, comparando os cálculos realizados por elas com os cálculos feitos à mão ou usando a calculadora ou mesmo as funções da biblioteca `math.h`, antes de testar o programa completo. Ou seja, durante o desenvolvimento do EP, vá testando ele aos poucos.

- Para facilitar os testes, pode ser conveniente separar individualmente os casos no arquivo exemplo de entrada, colocando-os cada um em arquivo de entrada próprio.

- Dado um arquivo de entrada, por exemplo, `entrada.txt`, você pode criar um outro arquivo `input.txt` tendo como conteúdo uma única linha

```
entrada.txt
```

e então, na lina de comandos, executar

```
./ep2 < input.txt > output.txt
```

Ao fazer isso, você não precisará digitar o nome do arquivo de entrada (`entrada.txt`), pois o nome será obtido pelo `ep2` a partir do arquivo `input.txt`. Além disso, a saída do programa (o resultado de todos os `printf`) será escrito no arquivo `output.txt`.