Trabalho prático

Documentação

Luis Antonio Duarte Sousa

- Introdução -

A proposta de atividade consiste na construção de um código em C++ que receba dados e manipule-os seguindo as seguintes instruções:

Considere uma cidade, representada como uma matriz. Na cidade há restaurantes e ruas.As ruas são representadas por coordenadas (x, y) indicando se a coordenada é pavimentada (1) ou não (0). Considere que se duas coordenadas adjacentes forem pavimentadas, então o trecho entre essas coordenadas é pavimentado. O arquivo “ruas.txt” fornece a informação sobre os trechos pavimentados e não-pavimentados (x, y,pavimentado). Todos os restaurantes são acessíveis através das ruas, ou seja, há sempre
um caminho possível entre um local pavimentado e um restaurante. Mais ainda, há sempre apenas um caminho possível.
Os restaurantes podem ser baratos/caros, e cada restaurante possui um entregador. O entregador realiza entregas com sua moto, e a velocidade da moto é especificada. O arquivo “restaurantes.txt” possui essa informação (x, y, nome, custo, velocidade).

A unidade de medida de distância é chamada “zambs”. A distância entre duas coordenadas adjacentes na matriz corresponde a um “zambs”. As velocidades das motos são dadas em zambs/minuto.

O input do código deverá ser as coordenadas de sua casa (pavimentada), sua preferência de restaurantes e seu limite de tempo desejado.

1. Calcule a distância em “zambs” de sua casa a todos os restaurantes.
2. Leia (do teclado) a preferência de custo (caro ou barato), e retorne um arranjo com os restaurantes em ordem de rapidez de entrega.
3. Leia (do teclado) a preferência de custo (caro ou barato) e o tempo máximo que você quer esperar (minutos), e retorne um arranjo com os restaurantes que respeitem a restrição de tempo de espera.

- Proposta de solução -

Qualquer erro de sintaxe nesse arquivo possivelmente é devido a formatação (O código está funcionando)

Como solução, a princípio foi preciso definir as bibliotecas que precisarei durante construção do código:

```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>
#include <string.h>
```

Em seguida a construção das tipagens de dados e métodos:

```
struct Ruas {
    bool paviment;

    void pavimento(int a);
};

void Ruas::pavimento(int a){
    if(a == 1){
        this->paviment = true;
    }
    else{
        this->paviment = false;
    }
}
```

Tipo **Ruas** é definido por um valor booleano (caracteriza a pavimentação) que é definido pelo método **pavimento** .

```

struct Coordenada{
    int x;
    int y;


};
```

O tipo **Coordenada** é definido por um inteiros x e y .

```
struct Restaurantes{
    int x;
    int y;
    char nome[100];
    char custo[100];
    int velocidade;


    int distancia(int x, int y, Ruas city[23][39]);

};
```

O tipo **Restaurantes** é definido por valores inteiros x, y e velocidade, strings nome e custo.

Além disso existe um método **distancia** que será explicado abaixo:

```
int Restaurantes::distancia(int x, int y, Ruas city[23][39]){
    int zambs = 0,contador = 0, marcador = 0;
    Coordenada ruaatual, novarua;
    x--;
    y--;

    bool verificados[23][39];
    for(int i = 0; i < 23; i++){
        for(int j = 0; j < 39; j++){
            verificados[i][j] = false;
        }
    }

    int vizvert[] = {0, 0, -1, 1};
    int vizhor[] = {-1, 1, 0, 0};

    Coordenada fila[23 * 39];
    int frente = 0;
    int tras = 1;

    verificados[y][x] = true;
    fila[0].x = x;
    fila[0].y = y;

    if(x == this->x && y == this->y){
        return 0;
    }
```

```
 while (frente != tras){


       ruaatual = fila[frente];

       int atualX = ruaatual.x;
       int atualY = ruaatual.y;


       for (int i = 0; i < 4; i++){
           novarua.x = atualX + vizhor[i]; // coluna
           novarua.y = atualY + vizvert[i]; // linha




           if (novarua.x < 39 && novarua.x >= 0 && novarua.y < 23 && novarua.y >= 0){ //
se está dentro da cidade

               if (city[novarua.y][novarua.x].paviment &&
!verificados[novarua.y][novarua.x]){ // se é pavimentada e não foi verificada


                   verificados[novarua.y][novarua.x] = true;
                   fila[tras].x = novarua.x;
                   fila[tras].y = novarua.y;
                   tras++;
                   contador++;


                   if (novarua.y == this->y && novarua.x == this->x){
                       return zambs + 1;
                   }
               }
           }
       }

       if(frente == marcador){
           zambs++;
           marcador = frente + contador;
           contador = 0;

       }

       frente++;
   }
}
```

Acredito que o real desafio dessa atividade se baseia nesse método uma vez que tive que estudar um algoritmo (BFS) para tentar adaptar o código para qualquer mapeamento de cidade que eu recebesse.
Esse método tem como lógica base receber as coordenadas inteiras de sua casa e a cidade em tipagem Ruas e em formato matriz.
Criar uma matriz **verificados** com mesma proporção da cidade onde será armazenado ,como valores verdadeiro e falso,coordenadas já visitadas.
Criar uma **fila/pilha** no formato vetor que armazenará novas possibilidades de caminhos a serem exploradas. Essa fila tem um elemento(coordenada) que está na **frente** e será o próximo a ser analisado.Além disso possui o elemento **tras** que se expandirá enquanto houver coordenadas não exploradas (pavimentadas) ou o restaurante corrente (**this**) não encontrado.
A contagem de zambs acontecerá sempre que uma expansão determinada pelo marcador acontecer para que não sejam contados unidades a mais nem a menos.
E caso a coordenada da sua casa seja a do restaurante o retorno está previsto para 0.

Em seguida a estrutura principal (int main()):

```
int x,y,z;
    int atualx, atualy;
    int limite, preferencia;
    char nome[100], custo[100];
    Restaurantes lista[23];

    Ruas cidade[23][39];
    Restaurantes Rest[23];
```

Definições de variáveis importantes.

```
FILE* ruastxt = fopen("ruas.txt", "r");

    if(ruastxt == NULL){
        printf("Problema no arquivo ruas.txt");
    }

    while(!feof(ruastxt)){
        fscanf(ruastxt, "%i", &x);
        fscanf(ruastxt, "%i", &y);
        fscanf(ruastxt, "%i", &z);
        cidade[y-1][x-1].pavimento(z);
    }


    fclose(ruastxt);
```

Leitura e armazenamento das informações fornecidas pelo arquivo **ruas.txt .**
(A redução em uma unidade é devido ao fato de que matriz/vetores começam no 0).

```
FILE* restaurantestxt = fopen("restaurantes.txt", "r");

    if(restaurantestxt== NULL){
        printf("Problema no arquivo restaurantes.txt");
    }

    for(int k = 0;!feof(restaurantestxt); k++){

        fscanf(restaurantestxt, "%i", &x);
        fscanf(restaurantestxt,"%i", &y);
        fscanf(restaurantestxt, "%s", &Rest[k].nome);
        fscanf(restaurantestxt,"%s", &Rest[k].custo);
        fscanf(restaurantestxt,"%i", &z);
        Rest[k].x =  x-1; Rest[k].y = y-1; //y é linha
        Rest[k].velocidade = z;
    }

    fclose(restaurantestxt);
```

Leitura e armazenamento das informações fornecidas pelo arquivo **restaurantes.txt .**

```
printf("Digite a cordenada 0 < X < 40 e 0 < Y < 24 de sua casa ->\n");

    scanf("%i", &atualx);
    scanf("%i", &atualy);

    if(cidade[atualy-1][atualx-1].paviment == false){
        while(cidade[atualy-1][atualx-1].paviment == false ){

            printf("Essa coordenada nao esta pavimentada ou esta fora
da cidade\n");
            printf("digite outra ->\n");
            scanf("%i %i", &atualx, &atualy);
        }
    }

    printf("Prefere restaurantes caros ou baratos ? \ndigite 1 para
caros e 0 para baratos -> \n");

    scanf("%i", &preferencia);
    while(preferencia != 0 && preferencia != 1){
        printf("escolha uma opcao valida\n");
        scanf("%i", &preferencia);
    }
```

Input (entrada) da **coordenada** de sua casa e da **preferência** de restaurantes.

```
int k = 0;
    for (int i = 0; i < 23; i++){
        if (preferencia == 1){
            if(strcmp(Rest[i].custo, "Caro") == 0){
                lista[k] = Rest[i];
                k++;
            }
        }
         if (preferencia == 0){
            if(strcmp(Rest[i].custo, "Barato") == 0){
                lista[k] = Rest[i];
                k++;
            }
        }
```

```
    }
```

Preenchimento do vetor **lista** apenas com os restaurantes que respeitam a preferência do usuário e contagem do número de elementos **k(não indefinidos)** que esse vetor tem.

```
for(int inicio = 0 ;inicio < k ;inicio++){ //organizador de menor para maior
        for(int verificador = inicio + 1; verificador < k;verificador++){
            if(((lista[verificador].distancia(atualx,atualy,cidade)/1.0) /
lista[verificador].velocidade) <
((lista[inicio].distancia(atualx,atualy,cidade)/1.0) /
lista[inicio].velocidade)){
                Restaurantes memory = lista[inicio];
                lista[inicio] = lista[verificador];
                lista[verificador] = memory;
            }
        }
    }
```

Organiza a lista separada no passo anterior,utilizando a **ordenação por seleção**, organizando do restaurante com menor tempo para o com maior tempo.
Tempo(rapidez de entrega) é definido como a distância dividida pela velocidade de entrega.

```
printf("Qual o maximo de tempo que pretende esperar ?\n");
    scanf("%i", &limite);

    printf("Segue a lista ordenada dos restaurantes com menor tempo de
entrega ao maior ->\n\n");

    for(int i = 0; i<k;i++){
        printf("%s -- %f min\n\n", lista[i].nome,
((lista[i].distancia(atualx,atualy,cidade)/1.0) / lista[i].velocidade));
    }

    printf("Segue os restaurantes que respeitam seu limite e
preferencia:\n\n");
    for(int i = 0; i<k;i++){
        if((lista[i].distancia(atualx,atualy,cidade) / lista[i].velocidade)
<= limite){
            printf("%s\n", lista[i].nome);

        }
    }
```

Entrada do **limite** de tempo desejado e impressão da questão número 2 e 3 respectivamente.Assim, finalizando o código.

- Resultados -

Aqui como exemplo mostrarei alguns resultados para entradas padrões de coordenada, preferência e limite.

input - Coordenada (1,1), caro, 10
output <

Digite a cordenada 0 < X < 40 e 0 < Y < 24 de sua casa ->
1
1
Prefere restaurantes caros ou baratos ?
digite 1 para caros e 0 para baratos ->
1
Qual o maximo de tempo que pretende esperar ?
10
Segue a lista ordenada dos restaurantes com menor tempo de entrega ao maior ->

Cantina_da_Carol -- 0.000000 min
Churrasco_de_gato -- 6.500000 min
Taste_Vin -- 7.600000 min
Espeto_do_Chico -- 9.500000 min
Voador -- 9.800000 min
Las_pombas -- 10.500000 min
Xucesso_da_Cida -- 15.600000 min
Caro_e_ruim -- 30.000000 min
Comida_de_buteco -- 58.000000 min
Moto_velha -- 63.000000 min
Macarrao_na_chapa -- 109.000000 min

Segue os restaurantes que respeitam seu limite e preferência:

Cantina_da_Carol
Churrasco_de_gato
Taste_Vin
Espeto_do_Chico
Voador
Las_pombas

>

Nesse exemplo como o Restaurante “Cantina da carol” fica exatamente na mesma coordenada selecionada a distância é 0 e logo o tempo também será.

input - Coordenada(10,1), barato, 15
output <
Cordenada não pavimentada ou fora dos limites da cidade

loop de entradas…
>

Nesse exemplo colocamos uma coordenada não pavimentada que retorna um aviso e um loop para selecionar novamente a coordenada até que seja válida.

input - Coordenada(1,10),barato, 12
output <

Digite a cordenada 0 < X < 40 e 0 < Y < 24 de sua casa ->
1
10
Prefere restaurantes caros ou baratos ?
digite 1 para caros e 0 para baratos ->
0
Qual o maximo de tempo que pretende esperar ?
12
Segue a lista ordenada dos restaurantes com menor tempo de entrega ao maior ->

Vila_Matriz -- 3.000000 min
Burger_queen -- 8.000000 min
Espolex -- 11.000000 min
Emporio_zambs -- 12.800000 min
Cometa -- 15.000000 min
Ta_danado -- 15.666667 min
Dogao_da_esquina -- 20.000000 min
Quibao -- 30.000000 min
Chega_frio -- 30.500000 min
Xapurex -- 57.000000 min
Sai_de_baixo -- 109.000000 min
Xulambs -- 218.000000 min

Segue os restaurantes que respeitam seu limite e preferencia:

Vila_Matriz
Burger_queen
Espolex
Emporio_zambs

>