Trabalho Prático 14

Pensando com portais

Algoritmos e Estruturas de Dados I

Prazo: **17** de **novembro** de 2017

1 Introdução

Depois do sucesso do muito esperado Meia-Vida 3, a empresa Torneira lançou seu novo sucesso, Portais 4, disponibilizado na nuvem de jogos Vapor. Nele, um personagem anda por um cenário onde portais podem aparecer em uma posição a qualquer momento, mas desaparecem depois de um tempo pré-definido. O jogador vence se, durante todo o tempo de jogo ele não cair em um portal.

Zezinho começou a jogar semana passada e apaixonou-se pelo jogo. Só que ele não é muito bom e quer poder acumular maior pontuação que seu irmão, Joãozinho. Para isso, ele deseja que você crie um simulador para que ele possa treinar de maneira mais rápida.

2 Tarefa

Você deverá criar um pequeno simulador do jogo, onde dados o tempo de jogo, posições e tempo dos portais e movimentos do jogador, definir se o jogador ganha ou perde o jogo.

3 Entrada

A primeira linha da entrada contém dois números inteiros, `T` e `P`, que representam o tempo de jogo que o personagem deve evitar os portais e o número de portais que aparecerão. As próximas `P` linhas são formadas de 4 números inteiros, $\texttt{T}_\texttt{i}$, $\texttt{T}_\texttt{f}$, $\texttt{X}_\texttt{p}$, $\texttt{Y}_\texttt{p}$, representando, respectivamente, o tempo de início (iniciando em 0), o tempo de fim e a posição onde onde cada portal irá aparecer. O jogador sempre começa na posição $\texttt{X}_\texttt{j}$, $\texttt{Y}_\texttt{j} = 0, 0$. As próximas `T` linhas representam os movimentos do jogador, podendo ser, `C`, `B`, `E` ou `D`, representando, respectivamente, os movimentos para cima (soma 1 à posição $\texttt{Y}_\texttt{j}$), baixo (subtrai 1 da posição $\texttt{Y}_\texttt{j}$), esquerda (subtrai 1 da posição $\texttt{X}_\texttt{j}$) e direita (soma 1 à posição $\texttt{X}_\texttt{j}$).

4 Saída

Para cada tempo T_j ($0 \leqslant T_j \leqslant T$, onde em $T_j = 0$ o jogador está na posição inicial), o programa deverá exibir `"Tempo` T_j `(`X_j`,` Y_j`): J"` (sem aspas), onde T_j é o tempo atual do jogo, X_j é a posição X do jogador, Y_j é a posição Y do jogador e `J` é um caractere `"S"` (sem aspas) caso ele esteja vencendo (não caiu em nenhum portal) ou `"N"` (sem aspas) caso ele perca (caiu em um portal). No caso de derrota, você também terá de finalizar a leitura e ignorar os próximos passos. Um jogador perde se em um dado tempo T_k existe um portal tal qual eles estão na mesma posição ($X_p = X_j$ e $Y_p = Y_j$) e o portal existe ($T_i \leqslant T_k \leqslant T_f$). Na dúvida, siga o exemplo abaixo.

5 Exemplo

Entrada

```
10 5
4 7 1 1
8 9 -2 0
4 6 1 -2
5 6 1 -1
3 8 1 -1
B
B
C
C
B
D
C
E
B
B
```

Saída

```
Tempo 0 (0, 0): S
Tempo 1 (0, -1): S
Tempo 2 (0, -2): S
Tempo 3 (0, -1): S
Tempo 4 (0, 0): S
Tempo 5 (0, -1): S
Tempo 6 (1, -1): N
```

Outros exemplos estão disponíveis na página deste trabalho no Moodle.

6 Avisos

Avisos mandatórios para o envio do trabalho:

- O código deve ser escrito em linguagem `C`
- As entradas que serão utilizadas para teste não conterão erros, então não será necessário testar a validade das mesmas
- Não utilize chamadas da função `system` (por exemplo, `system("pause")`) pois essas podem variar de acordo com o sistema operacional e os programas instalados da máquina onde o programa está rodando
- Deixe seu código bem comentado para facilitar a correção
- Não utilize espaços ou caracteres especiais nos nomes dos arquivos. Utilize apenas os caracteres de `A` a `Z` (tanto maiúsculas como minúsculas) sem acento, os números de `0` a `9` e os caracteres `-` (hífen), _ (*underscore*) e `.` (ponto final)
- Utilize a extensão `.c` para arquivos de código e `.h` para arquivos de cabeçalho, quando aplicar
- Se for submeter os arquivos via upload, não envie um arquivo comprimido (por exemplo, `.zip`, `.rar`, etc.). Utilize os diversos campos da aba "Submissão", um para cada arquivo
- Envie apenas os arquivos `.c` e `.h`
- Não copie o trabalho de algum colega ou baixe da internet. Lembre-se que o prejudicado será você pois o aprendizado obtido nessa disciplina será utilizado durante diversas outras etapas do seu curso

7 Dicas

Dicas importantes para o desenvolvimento deste trabalho:

- Utilize a função `scanf` para ler as entradas do usuário
- Utilize a função `printf` para imprimir os resultados das operações
- Cuidado com o fim de jogo, que pode acontecer a qualquer momento, mesmo tendo mais movimentos para serem lidos
- Crie uma `struct` para armazenar os portais
- Não utilize uma matriz para guardar o jogador e portais, já que o primeiro pode movimentar-se livremente e o segundo pode ser criado em qualquer lugar

- Guarde os portais em um arranjo
- Utilize arranjos de caracteres (*strings*) para ler os movimentos, pois o `scanf` com `%c` também lê espaços em branco e quebras de linha!
- Você deve utilizar *strings* de tamanho no mínimo 2 para conseguir guardar 1 caractere (devido ao caractere `'\0'` ao final)
- Note que `scanf("%s", movimento)` lê um arranjo de caracteres (*string*) para a variável `movimento`
- Você pode acessar um caractere de uma *string* através dos colchetes, assim como é feito em arranjos. Por exemplo, na *string* `"abcdef"`, o índice 0 é a letra `a`, o índice 1 é a letra `b` e assim sucessivamente
- Qualquer semelhança com a realidade é mera coincidência!
- Qualquer dúvida que tiver, utilize o fórum de dúvidas no Moodle. Inicie o assunto do tópico com a tag `[TP14]`
- Caso prefira, participe das monitorias toda quarta das 17h às 18h na sala 2012

8 Checklist

Checklist não exaustiva de passos até a entrega do trabalho:

1. Pesquise o funcionamento das funções citadas acima para facilitar o uso
2. Implemente e compile o programa
3. Teste para o exemplo dado acima. Compare as saídas para garantir o funcionamento correto
4. Faça o mesmo do item anterior para os exemplos disponibilizados na página deste trabalho no Moodle
5. Envie o trabalho pelo Moodle, onde ele será testado automaticamente para todos os casos disponíveis
6. Caso algum teste dê errado, volte ao passo 2

Bom trabalho e divirta-se!