Prova 2

Algoritmos e Estruturas de Dados I

Professor: Pedro O.S. Vaz de Melo

02 de junho de 2016

Nome: _____

escrevendo o meu nome eu juro que seguirei o código de honra

Código de Honra para este exame:

- Não darei ajuda a outros colegas durante os exames, nem lhes pedirei ajuda;
- não copiarei nem deixarei que um colega copie de mim;
- não usarei no exame elementos de consulta não autorizados.

Sobre a prova: nesta prova você vai implementar o simulador de um jogo semelhante ao *Bomberman* que farão como TP da disciplina. O seu programa deve simular o movimento e ataque aleatório de 20 jogadores em um cenário de tamanho 300×300. Cada jogador carrega uma única bomba que, quando explodir, mata qualquer jogador (inclusive o próprio) que estiver no seu raio de explosão. Assim que for depositada, o jogador é teletransportado para uma coordenada aleatória do cenário e a bomba inicia o seu relógio. Quando o relógio chegar a 0, a bomba explode. Você deve fazer uso das seguintes definições nas questões a seguir:

```
#define TAM 300 //tamanho da tela
#define MAXMOV 10 //tamanho máximo do passo do jogador
#define MAXTIMER 10 //tempo de relógio máximo da bomba
#define MAXRAIO 10 //raio máximo da bomba
#define NPLAYERS 20 //número de jogadores
#define MAXTAMNOME 20 //tamanho máximo do nome do jogador
```

1. (*3 points*) Defina os dois novos tipos de dados descritos abaixo:

a. (*1 pt*) Defina um **novo tipo de dados** para representar uma bomba. Esse tipo de dados deve ser chamado de `Bomb` e deve conter as seguintes informações: coordenadas x e y de aonde a bomba foi depositada, raio de explosão da bomba, relógio da bomba, ou seja, o tempo que levará para a bomba explodir a partir do momento que ela foi depositada.

b. (*2 pts*) Defina um **novo tipo de dados** para representar um jogador. Esse tipo de dados deve ser chamado de `Bomber` e deve conter as seguintes informações: coordenadas x e y do jogador, um indicador se o jogador está vivo ou morto, o nome do jogador, e a bomba que o jogador carrega.

2. (*4 points*) Faça um procedimento para criar uma *string* de tamanho aleatório com letras minúsculas aleatórias (códigos 97 a 122 na tabela ASCII). Este procedimento recebe um ponteiro para caractere e o máximo tamanho da *string*. O protótipo deste procedimento é:

```
void preencheString(char *str, int maxtam)
```

3. (*3 points*) Implemente uma função de nome `preencheBomber` que recebe um `Bomber` como parâmetro **por referência** e preenche os seus seguintes campos com valores aleatórios: x, y, nome e relógio da sua bomba. Indique também que o jogador está vivo.

4. (*3 points*) Implemente uma função de nome `checkKill` que recebe um `Bomber` p como parâmetro por referência e uma `Bomb` b como parâmetro por valor. A sua função deve verificar se o jogador p está no raio de explosão de b. Caso afirmativo, marque p como morto e retorne 1. Caso contrário, retorne 0. A fórmula para calcular a distância entre dois pontos (x_1, y_1) e (x_2, y_2) é: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

5. (*4 points*) Implemente um procedimento para mover n jogadores aleatoriamente pelo cenário. O tamanho do salto do jogador deve ser sorteado aleatoriamente, tanto na direção x quanto na y. Use a definição `MAXMOV` para o tamanho máximo do salto em cada direção. Você também deve definir aleatoriamente o sentido (esquerda ou direta e para cima ou para baixo) do salto em cada uma das direções. Não mova jogadores mortos. O protótipo do seu procedimento é:

```
void moveBombers(Bomber p[], int n);
```

6. (*5 points*) Escreva uma função para verificar a morte de todos os jogadores e retornar o número de jogadores que morreram. Para cada jogador, verifique se ele foi atingido por uma bomba. Lembre-se que a bomba só explode quando o relógio dela chegar a 0. O protótipo da sua função é:

```
int checkAll(Bomber p[], int n) { //n jogadores
```

7. (*3 points*) Escreva um procedimento que recebe um `Bomber` como parâmetro por referência e verifica o status da bomba dele. Caso o relógio da bomba seja 0, você deve depositar a bomba do jogador nas coordenadas do mesmo. Além disso, inicialize o relógio da bomba com um valor aleatório (no máximo `MAXTIMER`) e teletransporte o jogador para coordenadas x e y aleatórias. Se o relógio da bomba for maior que 0, diminua uma unidade dele. O protótipo do seu procedimento é:

```
void checkBomb(Bomber *p);
```

8. (*3 points*) Complete o código abaixo, que simula um jogo.

```
void main() {
  Bomber p[NPLAYERS];
  int alive = NPLAYERS, i;
  for(i=0; i<NPLAYERS; i++)
    preencheBomber(_________________);

  while(alive > 1) {
    moveCreatures(_________________);
    for(i=0; i<NPLAYERS; i++) {
      if(_________________) //jogador vivo
        checkBomb(_________________);
    }
    alive -= checkAll(_________________);
  }

  for(i=0; i<NPLAYERS; i++)
    if(_________________) {
      printf("\nwinner: %s:\n", p[i].nome);
      break;
    }
}
```

Referências:

Função/Operador	Descrição	Biblioteca	Exemplo
`float pow(float b, float e)`	retorna b^e	`math.h`	`pow(2,3)` retorna $2^3 = 8$
`double sqrt(double x)`	retorna $\sqrt{x}$	`math.h`	`sqrt(9)` retorna $\sqrt{9} = 3$
`%`	retorna o resto da divisão	-	`20 % 3` retorna 2