Jogo da Vida

PUC-SP Ciência da Computação

Grupo: ALGG

Inicialmente, é relevante proporcionar uma compreensão do que constitui o jogo da vida. Desenvolvido pelo renomado cientista John Conway, esse jogo é fundamentado em um autômato celular, sendo um modelo que representa as dinâmicas e transformações em grupos de organismos vivos.

Na importância desse jogo, está a premissa de que um organismo vivo depende da presença de outros seres vivos para sua subsistência e reprodução. Entretanto, uma densidade populacional excessiva pode resultar na morte do organismo devido à escassez de recursos alimentares.

Quanto às regras fundamentais do jogo da vida, podemos destacar:

- O nascimento de um organismo ocorre em uma célula desocupada quando existem exatamente três seres vivos vizinhos.

- Um organismo que possui dois ou três vizinhos vivos sobrevive para a próxima geração, refletindo a importância da interação em grupo.

- Organismos com quatro ou mais vizinhos enfrentam a morte, uma vez que a superpopulação resulta em escassez de alimentos, comprometendo sua sobrevivência.

- A solidão também é prejudicial, levando à morte de um organismo vivo que possui zero ou apenas um vizinho, destacando a necessidade de interações sociais para a preservação da vida.

Mediante a breve explicação, cabe agora mostrar o layout desse jogo com as breves explicações:

```
Escolha o tamanho do mundo (entre 10 e 60): 10

Escolha uma opcao do menu:
MENU DE OPCOES
1. Apresentar o mapa
2. Limpar o mapa
3. Incluir/excluir celulas vivas
4. Mostrar/esconder os mortos-vizinhos
5. Iniciar o processo
6. Gravar uma geracao inicial
7. Recuperar uma geracao inicial cadastrada
8. Limpar o cadastro de geracoes iniciais cadastradas
9. Regras de evolucao das celulas
0. Sair

Resposta:
```

O menu do jogo apresenta nove opções, e antes de decidir qual delas escolher, é necessário fornecer primeiro o tamanho do mundo. Essa etapa inicial é crucial para configurar adequadamente o ambiente do jogo, garantindo uma experiência personalizada e alinhada às preferências do jogador.

1. **Apresentar o Mapa**

Na primeira opção, o usuário terá a oportunidade de visualizar seu mundo, acompanhando as atualizações realizadas no ambiente de jogo. Essa funcionalidade proporciona uma visão detalhada e imersiva das mudanças ocorridas, permitindo ao jogador acompanhar de perto a evolução e dinâmica do seu universo virtual.

```
Resposta: 1

    00 01 02 03 04 05 06 07 08 09

  00 .  .  .  .  .  .  .  .  .  .
  01 .  .  .  .  .  .  .  .  .  .
  02 .  .  .  .  .  .  .  .  .  .
  03 .  .  .  .  .  .  .  .  .  .
  04 .  .  .  .  .  .  .  .  .  .
  05 .  .  .  .  .  .  .  .  .  .
  06 .  .  .  .  .  .  .  .  .  .
  07 .  .  .  .  .  .  .  .  .  .
  08 .  .  .  .  .  .  .  .  .  .
  09 .  .  .  .  .  .  .  .  .  .

0 celula(s) vivas
Geracao atual: 0

Lista da(s) 0 celula(s) viva(s): A lista esta' vazia!

Lista da(s) 0 celula(s) vizinha(s) morta(s): A lista esta' vazia!

LEGENDA:
 '.' - Celula morta
 'O' - Celula viva
 '+' - Celula vizinha morta
```

Nela pode-se observar que há informações para indicar quantas células vivas existem, quantas vizinhas mortas e além disso, a legenda indicando o que cada caractere corresponde.

2. **Limpar o Mapa**

Na segunda opção, o usuário terá a possibilidade de resetar seu mundo, proporcionando uma ferramenta prática para reiniciar o jogo conforme sua vontade.

Antes:

```
    00 01 02 03 04 05 06 07 08 09

  00 .  .  .  .  .  .  .  .  .  .
  01 .  .  .  .  .  .  .  .  .  .
  02 .  .  .  .  .  .  .  .  .  .
  03 .  O  O  O  .  .  .  .  .  .
  04 .  .  .  .  .  .  .  .  .  .
  05 .  .  .  .  .  .  .  .  .  .
  06 .  .  .  .  .  .  .  .  .  .
  07 .  .  .  .  .  .  .  .  .  .
  08 .  .  .  .  .  .  .  .  .  .
  09 .  .  .  .  .  .  .  .  .  .

3 celula(s) vivas
Geracao atual: 1

Lista da(s) 3 celula(s) viva(s): [lin, col][3, 3] [3, 2] [3, 1]
Lista da(s) 12 celula(s) vizinha(s) morta(s): [lin, col][4, 0] [3, 0] [2, 0] [4, 1] [2, 1] [4, 4] [4, 3] [4, 2] [3, 4] [2, 4] [2, 3] [2, 2]
LEGENDA:
 '.' - Celula morta
 'O' - Celula viva
 '+' - Celula vizinha morta
```

Depois:

```
Resposta: 2

    00 01 02 03 04 05 06 07 08 09

  00 .  .  .  .  .  .  .  .  .  .
  01 .  .  .  .  .  .  .  .  .  .
  02 .  .  .  .  .  .  .  .  .  .
  03 .  .  .  .  .  .  .  .  .  .
  04 .  .  .  .  .  .  .  .  .  .
  05 .  .  .  .  .  .  .  .  .  .
  06 .  .  .  .  .  .  .  .  .  .
  07 .  .  .  .  .  .  .  .  .  .
  08 .  .  .  .  .  .  .  .  .  .
  09 .  .  .  .  .  .  .  .  .  .

0 celula(s) vivas
Geracao atual: 0

Lista da(s) 0 celula(s) viva(s): A lista esta' vazia!

Lista da(s) 0 celula(s) vizinha(s) morta(s): A lista esta' vazia!
```

3. Incluir/excluir células vivas

A opção de incluir/excluir células permitirá ao usuário fornecer coordenadas para criar suas células. Caso deseje remover, basta inserir novamente as coordenadas da célula viva; uma opção será exibida, indicando '1' para sim e '0' para não. Se desejar sair, insira o valor -1 na linha e na coluna.

```
Digite a linha: 1

Digite a coluna: 1

    00 01 02 03 04 05 06 07 08 09

  00 .  .  .  .  .  .  .  .  .  .
  01 .  O  .  .  .  .  .  .  .  .
  02 .  .  .  .  .  .  .  .  .  .
  03 .  .  .  .  .  .  .  .  .  .
  04 .  .  .  .  .  .  .  .  .  .
  05 .  .  .  .  .  .  .  .  .  .
  06 .  .  .  .  .  .  .  .  .  .
  07 .  .  .  .  .  .  .  .  .  .
  08 .  .  .  .  .  .  .  .  .  .
  09 .  .  .  .  .  .  .  .  .  .

1 celula(s) vivas
Geracao atual: 1

Lista da(s) 1 celula(s) viva(s): [lin, col][1, 1]
Lista da(s) 8 celula(s) vizinha(s) morta(s): [lin, col][2, 2] [2, 1
LEGENDA:
 '.' - Celula morta
 'O' - Celula viva
 '+' - Celula vizinha morta


Digite as coordenadas (linha e coluna)
Digite -1 em ambas para sair
```

4. **Mostrar/Esconder os mortos-vizinhos**

Essa opção, quando ativada, resultará na marcação das células vizinhas das células vivas com o símbolo "+".

```
    00 01 02 03 04 05 06 07 08 09

  00 +  +  +  +  +  .  .  .  .  .
  01 +  O  O  O  +  .  .  .  .  .
  02 +  +  +  +  +  .  .  .  .  .
  03 .  .  .  .  .  .  .  .  .  .
  04 .  .  .  .  .  .  .  .  .  .
  05 .  .  .  .  .  .  .  .  .  .
  06 .  .  .  .  .  .  .  .  .  .
  07 .  .  .  .  .  .  .  .  .  .
  08 .  .  .  .  .  .  .  .  .  .
  09 .  .  .  .  .  .  .  .  .  .

3 celula(s) vivas
Geracao atual: 1
```

5. **Iniciar Processo**

A opção "Iniciar Processo" permitirá ao usuário avançar nas gerações de acordo com as regras especificadas sobre a permanência, nascimento ou morte de células. O usuário deverá inserir a quantidade desejada de gerações que deverão suceder e o intervalo de tempo entre a

criação de novas gerações. Se o intervalo for definido como zero, o avanço para a próxima geração será realizado manualmente, utilizando a tecla “Enter”.

Geração 1:

```
Digite a coluna: 6

    00 01 02 03 04 05 06 07 08 09

  00 .  .  .  .  .  .  .  .  .  .
  01 .  .  .  .  .  .  .  .  .  .
  02 .  .  .  .  .  .  .  .  .  .
  03 .  +  +  +  +  +  +  +  .  .
  04 .  +  O  O  O  O  O  +  .  .
  05 .  +  +  +  +  +  +  +  .  .
  06 .  .  .  .  .  .  .  .  .  .
  07 .  .  .  .  .  .  .  .  .  .
  08 .  +  +  +  +  +  +  +  .  .
  09 .  +  O  O  O  O  O  +  .  .

10 celula(s) vivas
Geracao atual: 1
```

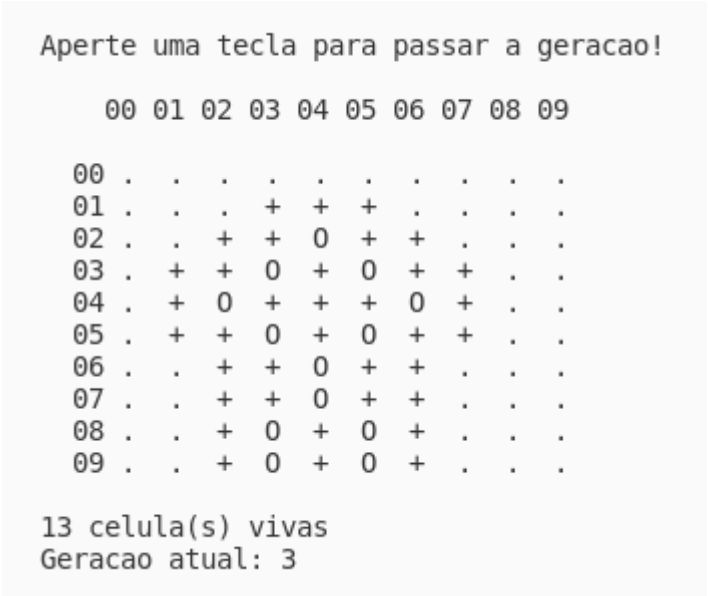
Geração 2:

```
Aperte uma tecla para passar a geracao!
    00 01 02 03 04 05 06 07 08 09

  00 .  .  .  .  .  .  .  .  .  .
  01 .  .  .  .  .  .  .  .  .  .
  02 .  .  +  +  +  +  +  .  .  .
  03 .  .  +  O  O  O  +  .  .  .
  04 .  .  +  O  O  O  +  .  .  .
  05 .  .  +  O  O  O  +  .  .  .
  06 .  .  +  +  +  +  +  .  .  .
  07 .  .  +  +  +  +  +  .  .  .
  08 .  .  +  O  O  O  +  .  .  .
  09 .  .  +  O  O  O  +  .  .  .

15 celula(s) vivas
Geracao atual: 2
```

Geração 3:



6, 7, 8. Gravação, Recuperação, Limpeza

Nessas opções, o usuário tem a oportunidade de preservar múltiplos estados de seu jogo, permitindo a recuperação de diversas situações ou fases específicas do seu mundo celular. Essas funcionalidades proporcionam ao usuário flexibilidade e controle sobre o histórico do seu jogo da vida.

6. **Gravar uma geração inicial**

No momento em que o usuário decidir guardar uma configuração de mundo celular, para dar continuidade depois, ele pode apertar a tecla 6 e gravar a configuração atual. O mundo fica salvo mesmo após o jogo ser fechado. É possível salvar diversas configurações de mundo.

7. **Recuperar uma geração inicial cadastrada**

Após salvar a configuração de mundo desejada, é possível acessá-la apertando a tecla 7. O usuário pode pressionar sucessivamente a tecla 7 para navegar entre as gerações, e parar quando encontrar a desejada.

8. **Limpar o cadastro de gerações iniciais**

A opção 8 apaga todas as gerações iniciais cadastradas, sem a possibilidade de recuperação.

9. **Regras de evolução das células**

A opção 9 apresenta as regras sob quais o mundo funciona no processo geracional. Quais células vivem, quais morrem, e quais nascem.

Dados Técnicos:

1. **Model**

“int tam” – tamanho do mundo (da matriz) digitado pelo usuário .
“int sair = -1” - variável para sair do loop do jogo.
“int x, y” – coordenadas globais.
“int ligarMortas = 0” – 1 para mostrar as vizinhas-mortas, e 0 para esconder as vizinhas-mortas.
“int vizinhas = 0” – quantidade de vizinhas que uma célula tem.
“int atraso” – intervalo de tempo, em segundos, entre gerações selecionada pelo usuário.
“int geracaoAtual” – contagem de gerações.
“int totalvivas = 0” – inicia o número total de células vivas com 0.
“totalmortas = 0” – inicia o número total de vizinhas mortas como 0.
“totalvivasprox = 0” – inicia o número total de células que estarão vivas na próxima geração como 0.
“int qtdgeracoes” – quantidade de gerações iniciais cadastradas.
“int ultimasalva = -1” – índice da última geração recuperada (de 0 a 49).

2. **View**

“void flush_in()” – limpa o buffer.
“int ViewmenuPrincipal()” – apresenta o menu principal na tela e capta a opção escolhida.
“void apresentaMensagem(char mens[100])” – apresenta uma mensagem a ser definida.
“void mostraMundo()” - mostra o mundo.
“int eliminarVida_V()” – elimina uma vida.
“int qtdGeracoes()” – obtém o valor de gerações seguidas que o usuário quer ver.
“void delay_V()” – pergunta a quantidade de delay de tempo em segundos entre cada geração.
“void printarLista(TipoCel *px, int total)” – mostra uma lista.
“void printarRegrasEvolucao()” – mostra as regras básicas de evolução das células.

3. **Controller**

“void menuPrincipal()” – função das opções do menu principal.
“void tamanhoMundo()” – pede o tamanho do mundo e o cria.
“void criarVida_C()” – cria ou elimina uma vida.
“void vizinhasMortas()” – mostra e esconde as vizinhas mortas.
“void obliterarMundo(char matriz[MaxTam][MaxTam])” – limpa o mapa e destrói todas as células vivas.
“int contarVidas()” – conta quantas células estão vivas.
“void proximaGeracao()” – faz a análise de quais células vão permanecer vivas ou de quais vão criar vida na próxima geração e atualiza a matriz atual com a nova geração.
“void delay_C()” – intervalo de tempo utilizado entre as gerações fornecido pelo usuário.

“void iniciaListas()” – inicia as listas vazias.
“void carregaVivas(int ii, int jj)” – carrega na lista de vivas.
“void carregaVMortas(int ii, int jj)” – carrega na lista de vizinhas mortas.
“void carregaVivasProx(int ii, int jj)” – carrega na lista de próximas vivas.
“void LimpaLista(TipoCel *aux, int total)” – limpa uma lista.
“void eliminarVidaLista(int ii, int jj)” – elimina uma célula da lista de vivas.
“void eliminar VMortoLista(int ii, int jj)” – elimina uma célula da lista de mortas.
“void verificaMorta(int ii, int jj)” – verifica se uma célula já está na lista de mortas.
“void atualizaVizinhasMortas()” – vê se as vizinhas de uma célula viva estão mortas, e se sim, as carrega na lista de mortas.
“void atualizaVivasProx()” – gera e atualiza a lista das células que estarão vivas na próxima geração.
“void gravaCelulas()” – salva todas as células vivas, uma por uma.
“void carregaConf()” – carrega a configuração de listas salvas.
“void recuperarCelulas()” – recupera as células da geração inicial desejada.
“void vivasMatriz()” – carrega as células na lista de vivas para a matriz.
“void limpaGeracoes()” – limpa as gerações iniciais salvas.
“void deletaConf()” – remove o arquivo de configurações iniciais para que não seja recuperado.

Estruturas utilizadas no programa

1. Uma matriz bidimensional m de caracteres: char m[MaxTam][MaxTam]

2. Um tipo de estrutura chamado cel, composto por 2 variáveis inteiras, que correspondem às coordenadas de cada célula, e por um ponteiro chamado next, que será usado para guardar o endereço da célula seguinte na lista ligada. O tipo dessa estrutura é TipoCel. Todas as células nas listas são do tipo TipoCel, e também os ponteiros para cada lista.

```
typedef struct cel{
 int lin, col; //as 2 coordenadas da celula
 struct cel *next; //ponteiro (endereco) para a proxima celula
} TipoCel;
```

3. Um tipo de estrutura chamado c, composto apenas por números inteiros, que correspondem as coordenadas das células. Essa estrutura é do tipo Cel, e é usada para criar um array de coordenadas, separadamente das listas ligadas.

```
typedef struct c{
 int lin, col;
} Cel;
```

4. O tipo de estrutura chamado lista, que tem uma variável inteira, que tem a finalidade de ser um contador de células vivas, e um array L[400] do tipo Cel. Essa estrutura é usada para guardar uma configuração de matriz atual, chamada Lvivas, e seu tipo é TipoLista.

```
typedef struct lista{
```

```
 int cont; //conta quantas celulas vivas tem
 Cel L[400]; //tamanho maximo: 400 celulas vivas
} TipoLista;
```

5. A estrutura chamada arquivo, com uma variável ListasConf[50], em que há uma variável TotalListas do tipo TipoLista. É aqui que serão guardadas as configurações de mundo que o usuário desejar salvar.

```
struct arquivo{
TipoLista TotalListas; //listas totais
} ListasConf[50]; //numero maximo de "listas das listas" e' 50
```

Observações a respeito do programa

Uma função interessante é a proximaGeracao, responsável pela sucessão das gerações de células no mundo. Em suma, ela é responsável por realizar a integração entre as listas e a função de geração de próximas células vivas, realizando o ciclo entre as gerações.

Primeiro, a matriz pela qual o usuário pode ver o seu mundo é limpa, para abrir espaço para a próxima geração. Em seguida, é gerada a lista de células vivas próximas a partir da lista de células vivas atuais por uma função chamada atualizaVivasProx, nela, é realizada uma verificação de quais células devem nascer, sobreviver e morrer, e é gerada a lista.

Depois, como a lista da próxima geração já foi gerada, a lista de células vivas atuais e de vizinhas mortas são limpas e os valores dos totais de células vivas atuais e de vizinhas mortas são zerados. Após isso, o apontador da lista de células vivas atuais passa a apontar para lista que era de próximas vivas, o que faz com que a lista de próximas vivas se torne a nova lista de vivas atuais.

Agora, com a lista de vivas atuais atualizada, o total de células vivas passa a receber o valor do total de células próximas vivas, e as células vivas são passadas para a matriz para que o usuário possa vê-las em seu mundo. Então, o valor do total de células próximas vivas deve ser reestabelecido para zero, e o ponteiro que apontava para a lista das células próximas vivas agora deve apontar para NULL, para que a próxima geração possa ser gerada na próxima iteração sem problemas.

Por fim, é chamada a função atualizaVizinhasMortas, que realiza a verificação de quais são as vizinhas mortas e cria uma lista com elas, é chamada a função vizinhasMortas, que apenas coloca ou esconde os marcadores que indicam quais células são vizinhas mortas, e um contador marca em qual geração aquele mundo está.