SME0230 - Introdução à Programação de Computadores
Primeiro semestre de 2018

Professora: Marina Andretta (andretta@icmc.usp.br)
Estagiário PAE: Petterson Pramiu (ppramiu@usp.br)
Monitores: Victor Forbes (victor.forbes@usp.br),
Hugo Cesar de Lima Vasques (hugocesar@usp.br)

Exercício 22 - Conway's Game of Life

1 Descrição

Nesse exercício você deverá implementar um simulador do Jogo da Vida de Conway. As regras do jogo podem ser lidas com detalhe em: `https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#Rules`.

O universo do jogo original é um *grid* bidimensional quadriculado infinito, mas para o nosso exercício o universo estará limitado a um tabuleiro B de N linhas e M colunas, ou seja, possuindo apenas $N \times M$ células quadradas. **Considere que as células fora do tabuleiro estão mortas.**

Seu programa deverá:

1. Ler a "semente" (tabuleiro B inicial, geração 0) a partir de uma imagem (de nome $filename_{in}$) do tipo *Portable BitMap* em ASCII.
2. Imprimir todas as gerações g até a geração G conforme o formato especificado na seção 3.
3. Gerar uma imagem (de nome $filename_{out}$) também do tipo *Portable BitMap* em ASCII.

Seu código deverá possuir uma `struct Board` que guardará as informações do tabuleiro. Toda alocação de vetores ou matrizes deverá ser feita dinamicamente com o uso de funções como: `malloc()`, `calloc()` ou `realloc()`. Não esqueça de usar a função `free()` para desalocar a memória!

Você deverá implementar, **pelo menos**, as seguintes funções:

- `Board *read_board(char *filename_in)`: Função que lê o tabuleiro B do arquivo $filename_{in}$.
- `void write_board(Board *b, char *filename_out)`: Função que escreve o tabuleiro B no arquivo $filename_{out}$.
- `void print_board(Board *b)`: Função que imprime o tabuleiro B conforme o formato especificado na seção 3.
- `void step(Board *b)`: Função que realiza "um passo no tempo", produzindo a geração seguinte.

OBS.: Você não é obrigado a seguir a risca **os cabeçalhos** das funções definidas acima.

2 Entrada

Na primeira linha haverá o nome da imagem de entrada $filename_{in}$.

Na segunda linha haverá o nome da imagem de saída $filename_{out}$.

Na terceira linha haverá um único inteiro G representando o número da última geração que deverá ser impressa na tela.

O arquivo $filename_{in}$ será uma imagem do tipo *Portable BitMap* em ASCII como descrito em: `https://en.wikipedia.org/wiki/Netpbm_format#PBM_example`. Para facilitar a leitura, não haverá comentários (precedidos por '`#`') na imagem de entrada. Veja o exemplo na seção 4 para esclarecimento.

Atenção: No formato *Portable BitMap* o número de colunas M ($1 \leq M \leq 100$) é dado antes do número de linhas N ($1 \leq N \leq 100$).

3 Saída

Seu programa deverá usar a função `print_board(Board *b)` para imprimir cada geração g ($0 \leq g \leq G$). Essa função deverá imprimir, na primeira linha, o número da geração g e o número de células vivas a da seguinte forma:

```
Generation: g / Alive: a
```

Em seguida, sua função deverá imprimir uma matriz de caracteres $N \times M$ representando o tabuleiro B. Imprima o caractere '`#`' para as células vivas e o caractere '`.`' para as células mortas.

Deve haver um pulo de linha entre a impressão de cada geração g.

4 Exemplo

Entrada

```
beacon.pbm
beacon_out.pbm
3
```

Saída

```
Generation: 0 / Alive: 8
......
.##...
.##...
...##.
...##.
......

Generation: 1 / Alive: 6
......
.##...
.#....
....#.
...##.
......

Generation: 2 / Alive: 8
......
.##...
.##...
...##.
...##.
......

Generation: 3 / Alive: 6
......
.##...
.#....
....#.
...##.
......
```

beacon.pbm

```
P1
6 6
0 0 0 0 0 0
0 1 1 0 0 0
0 1 1 0 0 0
0 0 0 1 1 0
0 0 0 1 1 0
0 0 0 0 0 0
```

beacon_out.pbm

```
P1
6 6
0 0 0 0 0 0
0 1 1 0 0 0
0 1 0 0 0 0
0 0 0 0 1 0
0 0 0 1 1 0
0 0 0 0 0 0
```

5 Observações

- **Visualização do Jogo da Vida**: Se você quiser visualizar a evolução das gerações em seu terminal, use a função `usleep(100 * 1000)` da biblioteca `unistd.h` após cada chamada da função `print_board(Board *b)`.
- **Limites da entrada**: As indicações "$(1 \leq N, M \leq 100)$" na descrição da Entrada servem apenas para indicar quais valores essas variáveis podem assumir. Isso significa que, para esse exercício, haverá apenas casos de teste com N e M entre 1 e 100.
- **Formato da saída**: Se atente para o formato da saída! O Run Codes só considerá correta a saída do seu programa se estiver **idêntica** à saída esperada. Não se esqueça de imprimir um `\n` no final!
- **Forma de entrega**: Os exercícios deverão ser entregues pelo Run Codes (`https://run.codes`). Código de matrícula da disciplina: **XHK1**
- **Plágio**: Esse é um exercício individual. Códigos iguais receberão nota 0!
- **Nota do Run Codes**: Essa nota corresponde à quantidade de casos de teste que seu programa foi capaz de responder corretamente, e não à sua nota final nestes exercícios!
- **Notas**: As notas serão postadas na página da disciplina:

  `conteudo.icmc.usp.br/pessoas/andretta/ensino/sme0230-1-18.html`