Programação para Engenharia

Ficha de Laboratório Nº 4

Coordenação de Engenharia Informática
Departamento de Engenharias e Tecnologias
Instituto Superior Politécnico de Tecnologias e Ciências

Objectivo

- Implementar a função principal com toda lógica necessária para conclusão do programa.

Introdução

Este laboratório visa auxiliar os estudantes a realizarem o trabalho académico de PPE, que consiste na implementação dum jogo de computador intitulado "Jogo da Vida".

Nesta fase do trabalho, os estudantes devem desenhar a lógica de programação necessária para implementar as funções adicionais.

Esse laboratório assume que o estudante concluiu o laboratório anterior (implementação da biblioteca `jogodavida`).

1. Execução do trabalho (ficheiro de trabalho main.c)

1.1. Inclusão de bibliotecas

Para simplificar o trabalho, deve incluir as seguintes bibliotecas:

```
/* Funções auxiliares - biblioteca do jogo*/
#include "jogodavida.h"

/* Adiciona outras bibliotecas aqui, se necessário*/
#include <dos.h>
```

1.2. Definição do número de gerações para a simulação

```
/* número de gerações para evoluir o mundo */
#define NUM_GERACOES 50
```

1.3. Implementação da função prox_geracao

Esta função deve definir o estado de todas as células na próxima geração e chamar `finalizar_evolucao()` para actualizar o estado actual do mundo para a próxima geração. O seu protótipo é:

```
void prox_geracao(void){
```

```
    // escreva aqui seu código
}
```

1.4. Implementação da função get_prox_estado

Esta função deve retornar o estado da célula em (x, y) na próxima geração, de acordo com as regras do Jogo da Vida de Conway (veja o enunciado principal). O seu protótipo é:

```
int get_prox_estado(int x, int y){

    // escreva aqui seu código

}
```

1.5. Implementação da função num_vizinhos

Esta função deve calcular o número de vizinhos vivos da célula em (x, y). O seu protótipo é:

```
int num_vizinhos(int x, int y){

    // escreva aqui seu código

}
```

1.7. Implementação da função main

Para concluir a sua implementação, escreva o seguinte código:

```
int main(void) {
    int i;

    iniciar_mundo();

    for(i = 0; i < NUM_GERACOES; i++) {
        prox_geracao();
        system("cls");
        imprimir_mundo();
        sleep(2);
    }

    return 0;
}
```


joao.costa@isptec.co.ao (João José da Costa) @ ISPTEC 2021-22

Apêndice – Estrutura do ficheiro main.c

```c
/*
    Name: Jogo da Vida (main.c)
    Copyright: 2021
    Author: Joao José da Costa (joaojdacosta@gmail.com)
    Date: 25/11/21 06:41
    Description:
    Jogo sem jogadores, que simula a geração de células
    num mundo bi-dimensional.
*/

/* Funções auxiliares - biblioteca do jogo*/
#include "jogodavida.h"

/* Adiciona outras bibliotecas aqui, se necessário*/
#include <dos.h>

/* número de gerações para evoluir o mundo */
#define NUM_GERACOES 50


/* funções a implementar */

/*
esta função deve definir o estado de todas as células
na próxima geração e chamar finalizar_evolucao() para
actualizar o estado actual do mundo para a próxima geração. */
void prox_geracao(void);

/* esta função deve retornar o estado da célula em (x, y)
na próxima geração, de acordo com as regras do Jogo da Vida de Conway
(veja o enunciado principal).*/
int get_prox_estado(int x, int y);


/* esta função deve calcular o número de vizinhos vivos
da célula em (x, y)*/
int num_vizinhos(int x, int y);

int main(void) {
    int i;

    iniciar_mundo();

    for(i = 0; i < NUM_GERACOES; i++) {
        prox_geracao();
        system("cls");
        imprimir_mundo();
        sleep(2);
    }

    return 0;
}
```

```
51
52  void prox_geracao(void) {
53
54      /* TODO (Alunos#1#): para cada célula, defina o estado na próxima
55                           geração de acordo com as regras do Jogo da
56                           Vida.
57
58                           Dica: use get_prox_estado (x, y),
59                           get_mundo_largura(), get_mundo_altura(),
60                           set_estado_celula(x, y, estado) e
61                           finalizar_evolucao() */
62
63  }
64
65  int get_prox_estado(int x, int y) {
66
67      /* TODO (Alunos#1#): para a célula especificada, calcula o estado na
68                           próxima geração usando as regras
69
70                           Use num_vizinhos (x, y) para calcular o
71                           número de vizinhos vivos
72
73                           Use get_estado_celula(x, y) e
74                           num_vizinhos(x, y) */
75  }
76
77  int num_vizinhos(int x, int y) {
78      /* TODO (Alunos#1#): para a célula especificada, retorna o número
79                           de vizinhos que estão VIVOS
80
81                           Use get_estado_celula (x, y) */
82  }
```