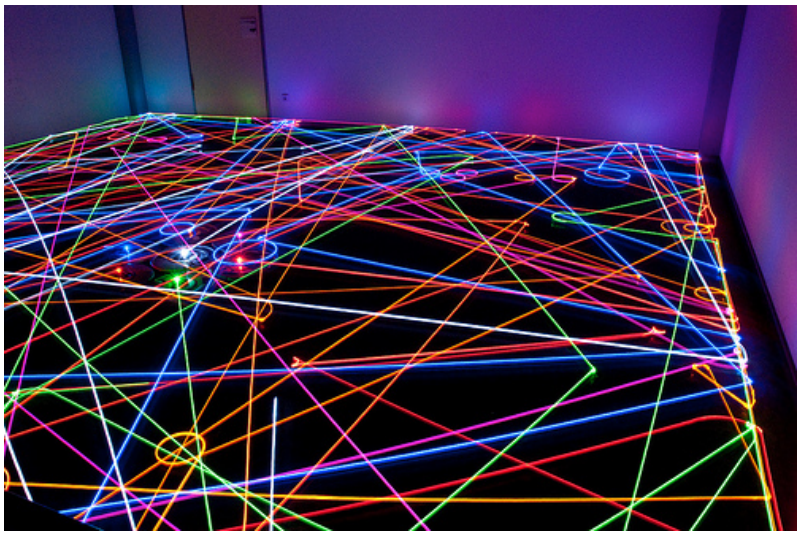
Programação 2014/2015

Mestrado em Engenharia Electrotécnica e de Computadores (MEEC)

Robot de Limpeza - iClean



Projeto de Programação – Entrega Final

1 Introdução

O objectivo deste projeto é desenvolver uma aplicação que simula o funcionamento de um conjunto de robots (frota) de limpeza. Desta forma, os alunos deverão alterar o programa da entrega intermédia adicionando as seguintes funcionalidades:

- Múltiplos robots de limpeza.
- Obstáculos no mapa.
- Leitura de um ficheiro com as posições iniciais de cada robot e com o mapa da sala.
- Adicionar e remover um robot de limpeza.

Este novo projeto tem uma parte da funcionalidade igual ao das entregas intermédias: permitir a visualização gráfica do mapa da divisão (incluindo espaços limpos/ocupados/sujos), simular a limpeza de cada robot, bem como a gravação de um ficheiro que indique o nome do robot e algumas estatísticas da limpeza.

2 Funcionamento

O funcionamento do programa pode ser sumarizado da seguinte forma:

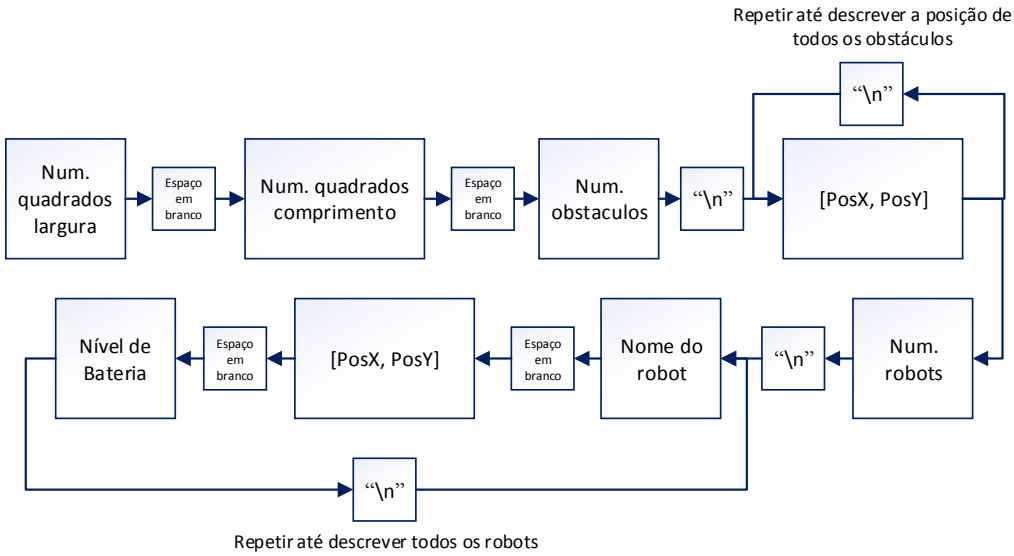
- Leitura do nome do ficheiro com o mapa da divisão, nomes dos robots e a localização inicial de cada robot.
- Os nomes dos robots devem ter apenas duas letras, a primeira é R e a segunda poderá ser qualquer carácter alfabético [a, b, ..., z].
- Cada robot deverá ter associado um nível de bateria, representado por um inteiro com valores entre 0 e 100.
- Simulação da limpeza da divisão:
    - Tal como na entrega intermédia a simulação poderá ficar mais rápida ou mais lenta a pedido do utilizador.
    - Deverá ser possível pausar a simulação, não se movimentando nenhum robot nesse caso.
    - O mapa da divisão deverá ser atualizado para indicar os quadrados ocupados (com obstáculos), já limpos e as novas posições dos robots.
    - Um robot nunca poderá ocupar posições no mapa declaradas como tendo um obstáculo ou como tendo outro robot.
    - Sempre que o robot se movimentar para um novo quadrado, o nível de bateria deverá decrescer um valor aleatório entre 2 a MAX (lido a partir da linha de comando, argumento argv da função main).
- Deverá ser possível gerir uma frota de robots de limpeza:
    - Deverá ser mostrado uma lista de robots em funcionamento no lado direito da aplicação (espaço em branco).
    - Deverá ser possível acrescentar um robot de limpeza à divisão em qualquer altura, passando este a movimentar-se de forma idêntica aos outros.
    - Um robot de limpeza deverá ser removido sempre que o seu nível de bateria for inferior ou igual a zero, desaparecendo do mapa da divisão.
- Gravação do caminho percorrido pelos robots da frota, o número de quadrados percorridos e o número de quadrados ocupados (i.e. tempo demorado). Apenas devem ser escritos no ficheiro os robots ativos, isto é, aqueles que não foram removidos
- O utilizador deve gerir a frota de robots e a própria simulação através do teclado:
    - Q - Sair da aplicação.
    - I – Reiniciar a simulação.
    - P – Pausar a simulação.
    - Setas – Diminuir ou aumentar a velocidade da simulação.
    - A – Adicionar robot.
    - E – Escrever ficheiro com o resultado da simulação.

3 Leitura e Escrita de Ficheiros

No inicio da aplicação deverá ser lido um ficheiro com a seguinte informação:

- Tamanho da divisão em número de quadrados na altura e largura.
- Número de quadrados ocupados.
- Lista com o número de quadrados ocupados (que nenhum dos robots pode atravessar) em coordenadas matriciais (x, y).
- Número de robots presentes.
- Nome de cada robot (duas letras apenas), a sua localização inicial e a seu nível de bateria.

O nome do ficheiro com o mapa da divisão deverá ser perguntado ao utilizador e lido a partir dos argumentos do programa (argv). A sintaxe deste ficheiro está ilustrada aqui:



A pedido do utilizador (opção e) deve ser escrito um ficheiro com algumas estatísticas sobre cada um dos robots da frota:

- Nome do robot.
- Número de quadrados limpos.
- Número de quadrados ocupados.

Este ficheiro tem um nome predefinido intitulado estatisticas.txt. A ordem da escrita no ficheiro passa a ser a ordem pelo qual os robots de limpeza estão presentes na lista dinâmica.

4 Descrição do Funcionamento das Opções

As várias opções definidas na secção 2 devem ser realizadas da seguinte forma:

- Opção I – Quando se reinicializa a aplicação, esta deve começar tal e qual como quando foi inicializada (dados definidos no ficheiro de entrada).
- Opção P – Quando se efetuar a pausa da aplicação, nenhum dos robots se pode movimentar. Quando se voltar a carregar na tecla P a simulação prossegue, isto é, os robots voltam a movimentar-se partindo da posição onde estavam.

- Opção A – Esta opção apenas poderá ser selecionada quando a simulação estiver em pausa. Nesta opção deverá ser acrescentado um novo robot, cujo nome deve ter duas letras. A primeira letra é 'R' e a segunda letra será inserida pelo utilizador a partir do teclado (usem as funções SDL). Para além do nome do robot, o utilizador deve indicar qual é o quadrado em que este deve ser inserido. Um robot depois de inserido deve se movimentar como os restantes efetuando a limpeza dos quadrados por onde passa. O seu nível de bateria deverá ser um número aleatório entre 50 e 100.
- Remoção de robots – Deverá ser removido um robot da lista dinâmica, sempre que este tiver o nível de bateria inferior ou igual a zero. Depois do robot ser removido este também deverá desaparecer do mapa da divisão.

5 Simulação da Limpeza

Em relação à simulação da limpeza, para além das funcionalidades anteriormente implementadas, um robot não pode colidir com outro, não pode ocupar áreas do mapa marcadas como ocupadas (obstáculos). Também se deve ter em consideração o seguinte:

- Um robot apenas consegue detectar obstáculos que se encontrem num quadrado vizinho da sua posição actul.
- Sempre que um robot encontrar um obstáculo ou outro robot devem gerar outra nova posição (aleatoriamente) para onde se devem mover.
- Um robot não conhece as áreas que outros robots limparam, isto é, os robots não comunicam entre si.
- Os robots devem guardar as posições do mapa que já foram limpas de forma a escolher sempre uma posição que ainda não tenham limpo.
- Quando toda a divisão estiver limpa, os robots não se devem movimentar e a simulação acaba.

6 Estruturas de Dados

Em relação aos tipos de dados a serem usados no projeto, as seguintes regras aplicam-se:

- Não podem ser usados vectores nem matrizes de tipos de dados elementares (int, char, float, etc.) para representarem o mapa da divisão ou a frota de robots.
- Devem usar uma lista (escolham o tipo de lista mais adequado) para representarem a frota de robots.
- A lista que representa a frota de robots, deverá estar organizada por ordem alfabética.
- O mapa da divisão pode ser representado por uma matriz (2D) ou vector (1D) de estruturas.
- Devem usar alocação dinâmica de memória para guardar a informação (estruturas) referente à frota de robots e mapa da divisão.

Também é importante salientar que o programa desenvolvido deverá ser estruturado em múltiplos ficheiros que permitam uma organização adequada do código.

7 Aspecto Gráfico

A aplicação iClean deverá ter um aspecto gráfico semelhante ao da entrega intermédia. No entanto, as seguintes modificações são necessárias:

- Cada posição do mapa poderá conter um obstáculo, estar limpa ou suja. Cores diferentes devem ser usadas para indicar o estado de cada quadrado.
- Cada posição do mapa poderá conter apenas um robot. O nome do robot deverá ser sobreposto ao quadrado que ocupa.
- Do lado direito da aplicação, deve ser mostrado uma lista dos robots em funcionamento ordenados por ordem alfabética mais o nível da bateria. Assim, permite-se a visualização da frota de robots (lista ligada).
- Poderão diminuir o tamanho do quadrados de forma a mostrar mapas maiores e mais complexos.

8 Desenvolvimento do Programa iClean

Também é fundamental que os alunos cumpram as regras que se seguem no desenvolvimento do programa iClean. Neste contexto, as recomendações que foram dadas para o projeto intermédio também se aplicam aqui.

8.1 Desenvolvimento Faseado

O desenvolvimento deste projeto deverá ser feito de uma forma faseada, devendo os alunos garantir que todas as funcionalidades codificadas até ao momento estão a funcionar corretamente. Assim sugerem-se os seguintes passos, pela ordem apresentada, para realização do projeto:

- Leitura do ficheiro de entrada com o mapa da divisão e informação com o número de robots.
- Atualização da interface gráfica para mostrar os robots e o mapa da divisão.
- Movimentação dos robots de forma a que estes não colidam com outros e com obstáculos.
- Aumentar e diminuir a velocidade de simulação.
- Paragem e reinicialização da simulação.
- Adição de um robot.
- Remoção de um robot.
- Escrita do ficheiro com o resultado da simulação.

Os alunos deverão garantir a robustez da aplicação verificando todos os casos de erro (por exemplo quando o ficheiro com o mapa da divisão não existir).

8.2 Documentação

O código produzido pelos alunos deverá ser devidamente comentado. Os comentários presentes no código deverão explicitar e explicar o funcionamento da aplicação assim como as decisões tomadas. As seguintes regras devem ser cumpridas:

- O código deve ser comentado sempre que realize alguma operação não óbvia.
- Os comentários devem ser claros, gramaticalmente corretos e usando frases simples.

- A declaração de todas as variáveis e constantes deve ser acompanhada de um comentário com uma breve descrição de para que servem.
- Cada estrutura de controlo (seleção ou repetição) deve ser precedida de um breve comentário explicativo.
- Todos os programas devem ter um comentário inicial que identifique, no mínimo, o trabalho, os seus autores o fim a que se destina e a data de realização.

8.3 Indentação

Um ponto fundamental na organização de escrita de código é a indentação, isto é, organização hierárquica das linhas de código, de acordo com âmbito onde elas se encontram. A indentação deixa o código fonte do programa mais organizado, mais legível, fácil de entender e de modificar sendo uma parte essencial do trabalho.

8.4 Estrutura do Código

Todos os programas que seguem a linguagem C devem possuir a mesma estrutura genérica, composta pelas seguintes secções em ficheiros *.h e *.c:

- Definição de tipos e estruturas de dados complexas (.h)
- Constantes globais e variáveis globais (apenas num caso extremo) (.h)
- Declaração de funções (.h)
- Bloco de comentários (.c)
- Diretivas #include (.c)
- Função main() (.c)
- Definição de funções (.c)

Como regra geral deve considerar que as funções devem caber num único ecrã, isto é, devem ter no máximo cerca de 30 linhas. Além disso, devem organizar o código em vários ficheiros (*.h) e (*.c) de forma a que cada ficheiro tenha um conjunto de funções correlacionadas entre si. Além disso, devem cumprir as seguintes regras:

- Inicialize sempre as variáveis na sua declaração.
- Teste a validade dos parâmetros recebidos por uma função.
- Declare constantes e nunca use números no seu código.
- Evite repetições de código, use funções, ciclos, etc.
- Evite o uso de variáveis globais.
- Não use gotos.
- Escreva código simples e claro que um colega teu possa perceber !

8.5 Compilação

O compilador a usar na execução do projeto é o gcc em ambiente Linux. **Os projetos que não compilem, i.e. que tenham erros de sintaxe, não serão avaliados.** A existência de avisos durante a fase de compilação poderá ser indício da existência de problemas no código. Estes deverão ser eliminados corretamente ou ignorados com cuidado extremo.

8.6 Decisões do Projeto

Como em qualquer projeto de informática, o funcionamento do programa não está totalmente definido no enunciado, existindo algumas ambiguidades e omissões. Para resolver essas omissões os alunos deverão tomar algumas decisões aquando do desenvolvimento do projeto. Estas decisões devem ser fundamentadas, sem nunca ir contra o definido no enunciado.

8.7 Biblioteca SDL

Durante o desenvolvimento deste projeto devera ser usada a biblioteca SDL2. A aplicação deverá ser compilada usando as bibliotecas SDL2, SDL2_image, SDL2_ttf. Mais informação disponível aqui:

- http://wiki.libsdl.org/APIByCategory
- https://wiki.libsdl.org/FrontPage

9 Submissão

Os alunos deverão submeter o código desenvolvido através do sistema FENIX até 22 de Maio. Deverá entregar o código correspondente ao programa desenvolvido bem como ficheiros auxiliares (imagens e fontes) que tenham sido usados. Também devem incluir junto à função main, um conjunto de comandos (usando o gcc) que permitam compilar o projecto.

10 Cópia de Trabalhos

Os trabalhos serão objecto de um sistema de detecção de plágio. Os alunos podem conversar entre si para discutir possíveis soluções para algum problema que tenham mas não podem partilhar código fonte. Nesta entrega intermédia, todo o código deve ser realizado por ambos os membros do grupo de dois elementos.

11 Avaliação da Entrega Final

A avaliação do projeto terá em conta diversos parâmetros:

- Funcionalidades implementadas
- Qualidade do código produzido
- Estruturação da aplicação
- Comentários e legibilidade do código
- Tratamento de erros
- Optimizações efectuadas