PROJECTO DE LABORATÓRIOS DE INFORMÁTICA II

Versão de 30 de Março

OBJECTIVO

Há vários cenários em que é importante perceber como organizar um conjunto de rectângulos numa dada área. Por exemplo, na indústria de corte (e.g., de couro ou metal) é importante cortar o máximo possível de peças de uma única folha. Neste caso estaríamos interessados em maximizar o número de rectângulos ou em minimizar o desperdício. Outro caso é a organização de uma área de trabalho pela disposição de vários recursos produtivos (e.g., máquinas). Em todos estes casos as várias peças não podem partilhar o mesmo espaço físico e estão também sujeitas a várias restrições de disposição relativa.

O objectivo do projecto é criar uma ferramenta de disposição de peças rectangulares sujeitas a várias restrições que permita ao utilizador colocar as peças na área de trabalho e verificar se todas as restrições são cumpridas. Adicionalmente, deveria ser possível pedir ao sistema para gerar soluções válidas ou mesmo encontrar a melhor solução possível.

Para isso é necessário descrever uma linguagem que represente tanto o problema como possíveis soluções. A representação do problema implica tanto a representação dos objectos que já estão dispostos na área como as restrições entre os vários objectos. Também existe uma linguagem de disposição dos objectos na área de trabalho.

LINGUAGEM

A linguagem de representação do problema utiliza um dado conjunto de palavras-chave:

- **DIM tamx tamy** - Define uma área rectangular que começa nas coordenadas (1, 1) até às coordenadas (tamx, tamy)
- **RECT nome tamx tamy rodar** - Define um rectângulo com um dado nome, dimensões (tamx, tamy) e se este pode ser rodado ou não (usando s ou n)
- **COL nome x y** - Serve para colocar o rectângulo com um dado nome nas coordenadas (x, y) i.e., o canto superior esquerdo do rectângulo fica nessas coordenadas
- **COLR nome x y** - Coloca o rectângulo mas roda-o antes (i.e., faz uma rotação de 90º)
- **DIR nomeA nomeB** - Define a restrição que o rectângulo nomeA está á direita do rectângulo nomeB
- **ESQ nomeA nomeB** - O rectângulo nomeA está à esquerda do rectângulo nomeB
- **CIM nomeA nomeB** - O rectângulo nomeA está por cima do rectângulo nomeB
- **BX nomeA nomeB** - O rectângulo nomeA está por baixo do rectângulo nomeB
- **CLD nomeA nomeB** - Os rectângulos nomeA e nomeB tem um lado colado
- **SEP nomeA nomeB** - Os rectângulos nomeA e nomeB não podem ter um lado colado
- **AREA nome x y tamx tamy** - Define uma área rectangular que começa no ponto com coordenadas (x, y) e acaba no ponto com coordenadas (x + tamx, y + tamy)

- **DENTRO nomeRect nomeArea** - Restringe o rectângulo com nome nomeRect a estar completamente dentro da área com nome nomeArea
- **FORA nomeRect nomeArea** - Restringe o rectângulo com nome nomeRect a estar completamente fora da área com nome nomeArea
- **ESTADO** – Imprime usando a linguagem de saída dada a seguir as posições de todos os rectângulos por ordem alfabética

Existe uma linguagem que deverá ser utilizada pelo programa para indicar a posição em que os rectângulos foram colocados. A linguagem de colocação das peças cria uma linha para cada rectângulo com a seguinte forma:

nomeRect x1 y1 x2 y2

onde nomeRect é o nome do rectângulo, (x1, y1) é o canto superior esquerdo, (x2, y2) é o canto inferior direito.

CLARIFICAÇÕES

Um rectângulo está à direita de outro se o seu lado esquerdo está à direita (ou colado) ao lado direito do outro. Usa-se o mesmo raciocínio para as outras direcções.

As áreas (definidas pelo comando AREA) não têm que ser disjuntas. Isto é, poderíamos ter os dois comandos:

AREA a1 1 2 7 5

AREA a2 5 4 6 6

Que tem em comum a área entre (5, 4) e (8, 7).

Já dois rectângulos não podem ter nenhuma área em comum.

EXEMPLO

Veja o(s) exemplo(s) em http://omega.di.uminho.pt/li2/1011

GESTÃO DE PROJECTO

Pretende-se que os grupos de trabalho façam a gestão do projecto, evidenciando desse modo a sua capacidade de organização. Para o efeito será utilizada uma ferramenta de gestão de projectos que permitirá ao grupo planear o desenvolvimento do projecto, definindo tarefas, fazer a sua atribuição aos elementos da equipa e acompanhar a sua implementação. Mesmo que uma tarefa precise de mais do que um elemento deverão discriminar o tempo gasto nessa tarefa por cada elemento. Deverão também utilizar todas as restantes funcionalidades da ferramenta (e.g. documentação, wiki, etc.). Seguem-se alguns exemplos de tarefas:

- Instalar uma ferramenta utilizada
- Documentar o código
- Criar uma nova função
- Integrar várias funções
- Avaliar o desempenho do código
- Reescrever uma função para melhorar o seu desempenho

- Reescrever uma função para remover *bugs*
- Testar o código
- Gerar testes para automatizar o processo de teste do código

ETAPAS

O projecto está dividido em três etapas. A primeira etapa preocupar-se-á com a leitura do dispositivo padrão de entrada (*stdin*) dos comandos e da escrita no dispositivo de saída (*stdout*) da linguagem de disposição dos rectângulos. Esta etapa assume que primeiro se define a área de trabalho, os rectângulos utilizados e as eventuais áreas e depois se colocam os rectângulos. Ao colocar os rectângulos, deve-se validar se esta colocação é possível (pode não o ser se já foi anteriormente colocado outro rectângulo no mesmo espaço físico). Posteriormente definem-se as restrições devendo ser impresso para cada restrição a palavra **SIM** ou **NAO** caso a restrição seja ou não cumprida.

A segunda etapa difere da primeira porque as restrições podem ser colocadas antes do posicionamento dos rectângulos. Neste caso, ao colocar um rectângulo é necessário verificar todas as restrições que tenham sido colocadas anteriormente. Para além disso nesta etapa deve ser possível desfazer operações efectuadas voltando a um estado anterior do problema.

Na terceira etapa deve ser possível pedir à ferramenta que gere uma solução para o problema. Deveriam ser contemplados vários tipos de problemas tal como soluções que maximizam o espaço entre as várias peças ou casos em que se permitem múltiplas instâncias de certas peças sendo o objectivo maximizar o número de peças colocadas ou minimizar a área sobrante.

MATERIAL A ENTREGAR EM CADA ETAPA

- Documentação gerada automaticamente pelo *Doxygen*
- Relatório de desempenho do grupo na execução das diversas tarefas utilizando funcionalidades da ferramenta de gestão de projecto (descrição das tarefas incluindo tempo total dispendido e tempo por cada pessoa envolvida)
- Código fonte e respectiva *makefile*

CRITÉRIOS

Os critérios seguintes são considerados básicos e devem ser cumpridos:

- O programa tem que compilar e funcionar em ambiente LINUX
- A *makefile* deve compilar com as opções -Wall -Wextra -pedantic -ansi -O2
- O nome do executável tem que ser **rect**
- A compilação não pode ter erros nem *warnings*
- O programa deve ler os comandos da linha de entrada
- O programa deve identificar correctamente os comandos e executá-los
- Os comandos devem funcionar correctamente em situações normais
- O código deve estar correctamente comentado e documentado (i.e., a documentação gerada pelo *Doxygen* deve ser completa, clara e fidedigna)

Adicionalmente, considerar-se-ão os seguintes critérios não obrigatórios:

- Devem ser reportados erros de inconsistência de comandos
- As estruturas de dados deverão ser eficientes tanto em termos de armazenamento como de facilidade de acesso
- A implementação deve ser eficiente
- O código deve estar bem estruturado
- As funções devem ser curtas e usar funções auxiliares que implementem tarefas comuns
- O código deve ser *legível* (não deve ser necessário perder mais do que um segundo para perceber uma linha)
- O nome das funções e variáveis deve ser escolhido de forma a perceber-se a sua função
- Não devem existir variáveis globais

ENTREGA

Deverá ser colocado na opção "Files" do redmine ("Ficheiros" para os alunos que usam a versão portuguesa) um arquivo compactado com o comando tar do qual constem os seguintes ficheiros e pastas:

Identificacao ficheiro com a identificação dos alunos (nome completo e número)

code com o código fonte e respectiva *makefile*

doc com a documentação gerada pelo *doxygen*

A pasta deverá ter o nome PL<nº do turno>g<nº do grupo>-et<nº da etapa>.tar.bz2

Nos casos em que o número do grupo seja só um algarismo este deverá ser precedido de um zero.

Exemplos:

- PL6g02-et1.tar.bz2 grupo 2 do turno 6 a entregar a etapa 1
- PL2g11-et2.tar.bz2 grupo 11 do turno 2 a entregar a etapa 2

Para se usar o comando tar da forma **correcta**

1. Abre-se uma consola no Linux
2. Usando o comando cd vai-se para a directoria que contém as pastas code e doc
3. Escreve-se o comando tar jcf PL2g11-et2.tar.bz2 identificacao code doc

Se não cumprirem alguns destes requisitos, o trabalho não será considerado entregue.