Relatório Projecto LI II

Pedro Faria nº60998 Helena Alves nº61000
André Santos nº60994 PL6 Grupo 8

29 de Maio de 2011

Conteúdo

0.1 Introdução

No âmbito da cadeira Laboratórios de Informática II, perante o problema apresentado iremos neste trabalho desenvolver em C um programa que vai organizar um conjunto de rectângulos numa dada área. O oblectivo era 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. O problema foi dividido em três etapas. Na primeira etapa desenvolvemos os comandos base do programa, criação de estrutura de dados e os comandos necessários. Na segunda etapa, mudamos o código para estruturas ligadas, eliminamos as estruturas de comprimento fixo e passamos a comprimento variável, criamos um comando adicional, o comando 'DESF' e colocamos a verificação das restrições antes da colocação dos rectângulos. E, por último, na terceira etapa, os rectângulos são colocados automaticamente atraves do comando 'RSV'.

0.2 Estrutura de dados utilizada

O nosso grupo decidiu optar por ter várias estruturas independentes, pois, de início, os nossos conhecimentos não eram os melhores e, assim era mais fácil iniciar essas mesmas estruturas. Então, criamos estruturas para a etapa 1 de array de comprimento fixo:

- Rect: define caractrísticas do rectângulo;
- ListaRect: lista que guarda os rectângulos definidos;
- RectCol: define características necessárias para o rectângulo ser colocado;
- ListaRectCol: lista que guarda os rectângulos colocados;
- Area: defina nova área;
- ListaArea: lista que guarda todas as áreas;
- RestrDIM: lista que guarda as restrições da área de trabalho;
- Lista: lista qu contem uma lista de rectângulos, uma lista de rectângulos colocados, uma lista de áreas e guarda as dimensões da área de trabalho.

Na segunda e terceira etapa, as nossas estruturas foram todas modificadas e passaram a estar em listas ligadas:

- DIM: define a dimensão da área de trabalho; Aqui usamos uma variável 'u' para verificar se a dimensão já foi utilizada. Esta variável não era necessária, mas decidimos colocá-la pois ia ser útil no desenvolvimento do código.
- LRestr: lista de restrrições; Nesta estrutura, a nossa estratégia para guardar as restrições foi guardá-las como um número, ou seja, cada restrição possui o seu próprio número, por exemplo a restrição 'DIR' é guardada com o número 1. Esta forma de guardar as restrições não era obrigatória mas foi a melhor forma,para nós, que encontramos para guardá-las.
- LRect: lista de rectângulos; Aqui, para verificarmos se o rectângulo estava ou não colocado utilizamos a variável 'c'.
- LArea: lista de áreas;
- Stack: histórico dos últimos comandos que verificaram o estado, o 'COL', o 'COLR' e 'AREA'. Aqui para guardarmos as restrições relativas a esta estrutura, nomeadamente as restrições 'COL', 'COLR' e 'AREA', optamos por guardá-las com números, sendo assim temos: 1=COL, 2=COLR, 3=AREA. Para nós, esta foi a melhor forma de guardarmos as restrições, sendo mais facilitada o seu uso no desenvolvimento do nosso código.

0.3 Estruturas e tipo de algoritmos

No nosso trabalho usamos dois tipos de algoritmos: iterativos e recursivos. Criar funções de forma recursiva é mais fácil, mas por outro lado é mais desvantajoso, pois caso estejamos a trabalhar com um grande número de rectângulos, as funções recursivas vão ser utilizadas várias vezes. Sendo assim, sempre que o nível de dificuldade, entre fazer recursivamente e , não era elevado optamos sempre por fazer iterativamente pois, também é o tipo de algoritmo que sentimos mais á vontade. Caso o nível de dificuldade seja elevado optamos por fazer recursivamente. No nosso código, funções como a inserção de áreas e de rectângulos("adicionaA"e "adicionaR") fizemos de forma recursiva, pois neste caso era a forma mais acessível. Por outro lado, funções como "COL"e "COLR"fizemos de forma iterativa.

0.4 Pontos fracos e fortes do trabalho

Pontos Fracos:

- Duplicação do código Durante todas as etapas, o nosso grupo não se preocupou com a duplicação do código, o mais importante era fazermos todas as funções e não nos preocupavamos em modificar as funções e torná-las da nova maneira que pretendiamos.
- Código pouco estruturado

Pontos Fortes:

- Trabalho estável e completo Resolvemos todas as funções pedidas, sendo ou não eficientes, tudo foi cumprido.
- Defesa contra o uso incorrecto

0.5 Balanço do projecto

O balanço que o grupo faz deste projecto é positivo, sendo que realizamos todas as etapas e entregamos as mesmas atempadamente. Apesar do trablho ter um elevado número de linhas de código, podemos dizer que o objectivo principal em cada uma das etapas foi atingido, na medida em que temos um trabalho nomeadamente estavel e completo. Os elementos do grupo ao terem-se reunido varias vezes empenharam-se para que a realização de todas as funções fossem o mais eficaz possivel, sendo que foram criadas todas as funções mas nem todos são o mais eficintes possivel.

0.6 Conclusão

Após a realização de todas as etapas deste projecto, podemos dizer que ao longo da realização do projecto surgiram algumas dificuldades, tais como, a manipulação das estruturas de dados genéricas, o uso de grandes quantidades de informação e a forma para se estruturar esta informação sem perdas de memória

e o mais eficaz possível.
A realização deste trabalho foi para todos os elementos do grupo, uma grande ajuda no desenvolvimento e compreensão da programação de linguagem C. Relativamente ao projecto, tentámos cumprir todas as etapas da melhor forma possível. Assim, numa forma de auto-avaliação do trabalho, consideramos que este projecto foi efectuado de forma positiva, porém alguns aspectos deste projecto poderiam ter tido um melhor desenvolvimento.