Universidade do Minho

Escola de Engenharia

Unidade Curricular:

Laboratórios de Informática III

Ano Lectivo de 2014/2015

Relatório de Desenvolvimento de Projecto

GestHiper - C

70719 Diogo Constâncio,

70513 Pedro Araújo

Índice

Índice de Figuras

Fotos

FIGURA 1 – DIOGO CONSTÂNCIO

FIGURA 2 - PEDRO ARAÚJO

Resumo

Este relatório complementa o Projecto Imperativo no âmbito da U.C. Laboratórios de Informática III, na qual se pôs em prática conhecimentos de modularidade, encapsulamento e desenvolvimento de estruturas de dados eficientes.

Conta ainda com conhecimentos e metodologias adquiridas nas U.C. Algoritmos e Complexidade, Arquitectura de Computadores, sob a forma de algoritmos optimizados, boas práticas de programação e escolha de estruturas de dados.

O projecto foi particionado em 4 fases sucessivas:

- Planeamento e consequente desenvolvimento das estruturas de dados;
- Implementação dos módulos da biblioteca *GestHiper*;
- Escrita da interface com o utilizador em linha-de-comandos;
- Optimização das estruturas de dados;

Como desenvolvido, o projecto apresenta 3 secções:

Main: Interface em linha-de-comandos fornecida ao utilizador que executa operações sobre a biblioteca *GestHiper*, gere o *input*/*output* do utilizador, estando também responsável pelo tratamento destes.

GestHiper: Biblioteca que compõe os módulos de Catálogo de Produtos/Clientes, Contabilidade e Compras.

Estruturas Básicas: Bibliotecas de várias estruturas de dados que complementam os módulos da *GestHiper*, estruturas como *MinHeap*, *BST* e *HashTable*.

Este relatório trata principalmente a biblioteca *GestHiper*, e os módulos que a compõem.

Introdução

Estrutura da Aplicação

De acordo com a prática de modularidade utilizada no desenvolvimento do projecto, verifica-se que a biblioteca *GestHiper* foi construída de forma a que seja facilmente portável ao longo de sistemas operativos diferentes dada a atenção a métodos exclusivamente *standards*, e mesmo input/output diferentes, como por exemplo leitura de dados incremental, manual, ou apresentação de resultados/dados sob a forma de uma *GUI*.

O *Main* trata-se de um exemplo de utilização da biblioteca por parte de um utilizador em linha-de-comandos, sendo assim o único módulo a interagir directamente com o utilizador, pela *prompt*, ou em ficheiros.

Como tal este é responsável por garantir a integridade do *input* que chega à biblioteca *GestHiper*, assim como apresentar o output desta de forma rápida e intuitiva.

Contém ainda funções que auxiliam a comunicação com o utilizador, como métodos para exibir resultados de forma tabulada, com suporte a paginação, e número de argumentos variável.

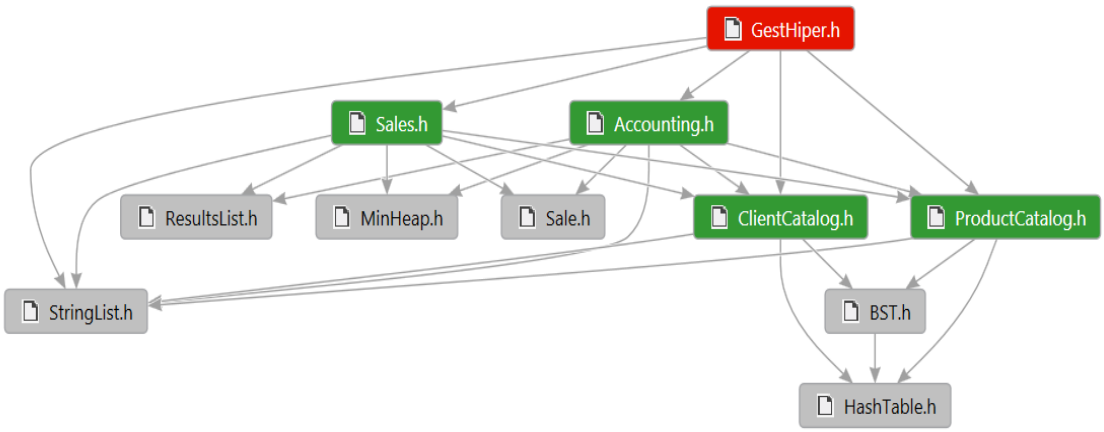

FIGURA 3 – DIAGRAMA DE MAKEFILE

GestHiper

Nesta secção são tratados cada um dos módulos pertencentes à biblioteca *GestHiper*¸ sendo eles **Catálogo** de Produtos/Clientes, **Contabilidade** e **Compras**.

É apresentada uma breve descrição dos encargos de cada um dos módulos, tratando de seguida a estrutura de dados utilizada no módulo, assim como as decisões que motivaram a sua escolha.

As estruturas de dados destes módulos foram desenvolvidas com atenção à resposta eficiente às perguntas que são colocadas no enunciado, pelo que a complexidade encontra-se no próprio desenho da estrutura, e não nos métodos que trabalham sobre estas, que na grande maioria dos casos se tratam de simples acessos constantes/travessias em *arrays.*

Como tal a API será mencionada brevemente apenas na descrição das estruturas, respectivamente a cada módulo, com ênfase nas funções de maior importância, inserção e ordenação (onde for aplicável).

Realça-se também a preocupação com a boa gestão de memória dada a magnitude dos dados que são aqui tratados, pelo que todas as estruturas de dados utilizadas têm um desenho dinâmico, alocando-se o espaço necessário à medida que é inserida nova informação nestas, contando ainda com função de libertação de memória para cada uma das estruturas.

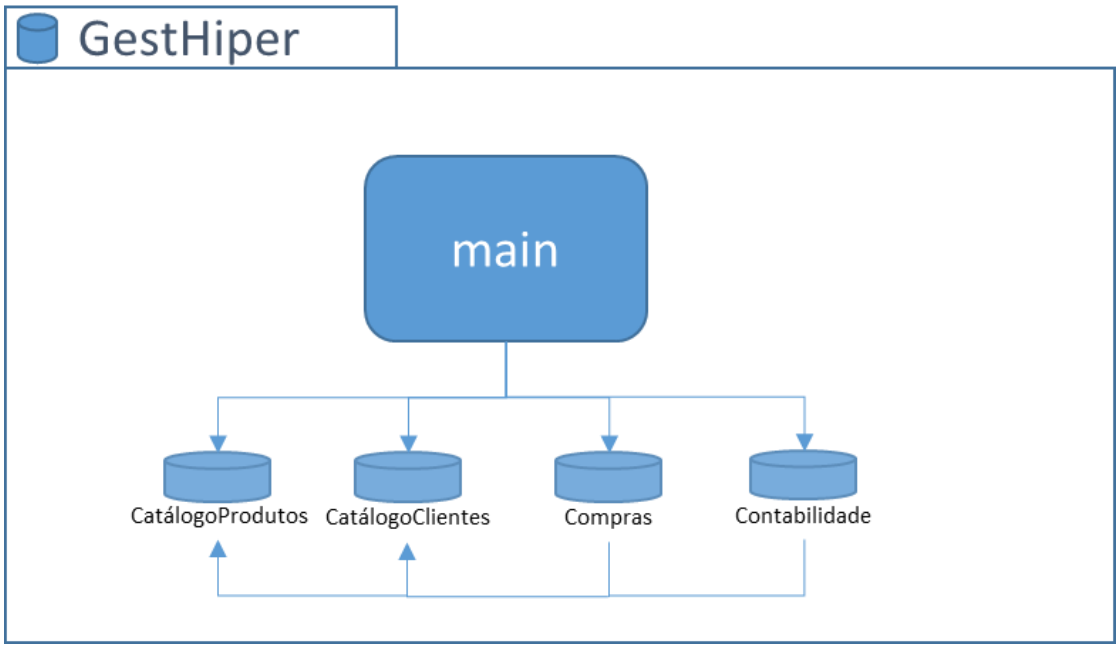

FIGURA 4 – DIAGRAMA GESTHIPER

Catálogo de Produtos/Clientes

São tratados aqui tanto o Catálogo de Produtos, como o Catálogo de Clientes, uma vez que são em tudo bastante semelhantes, diferenciando-se principalmente por terem comprimentos de código diferentes.

Estes módulos fazem a gestão das listas de Clientes/Produtos válidos, assim como as perguntas sob essas listas, que não incluem questões relativas a Compras ou Contabilidades, pelo que são totalmente independentes dos restantes módulos da biblioteca.

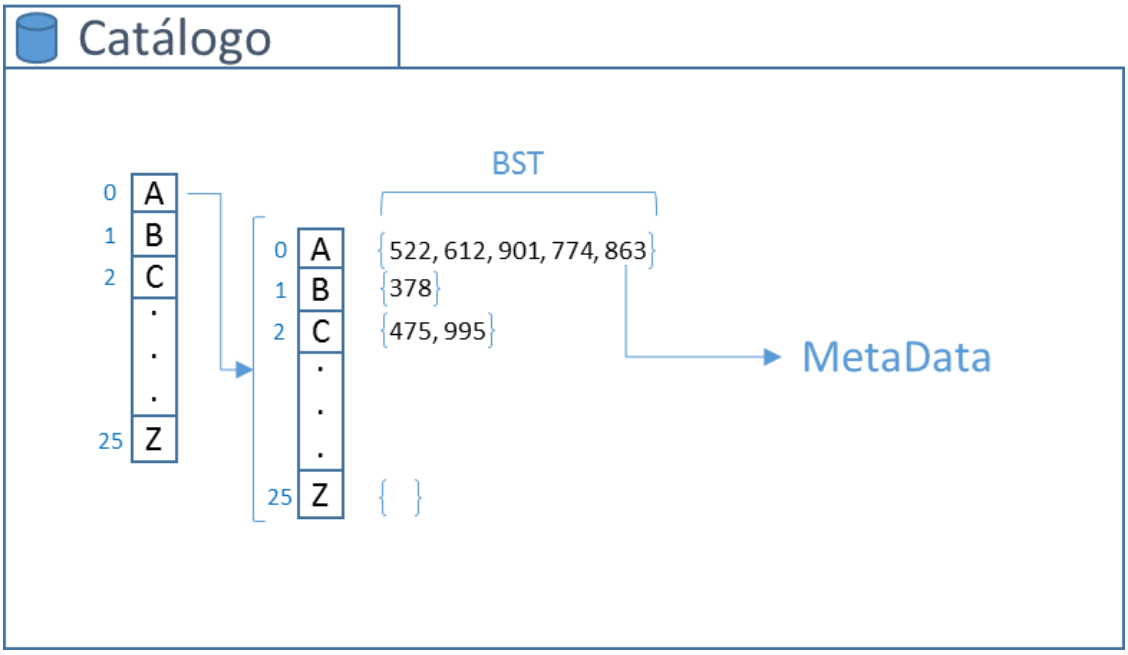

FIGURA 5 – ESTRUTURA DE CATÁLOGOS

Estrutura de Dados

O formato utilizado pelos catálogos de produtos/clientes trata-se de uma matriz de 26x26, os prefixos, que representam os dois caracteres que precedem os dígitos dos códigos no catálogo, a que cada elemento corresponde uma *BST* cujas chaves são os respectivos dígitos numéricos dos produtos/clientes, são no entanto tratados como valores inteiros neste ponto.

Como tal a inserção de um código nesta estrutura traduz-se em 2 simples operações, começando por extrair os caracteres nas primeiras duas posições do código, estes são utilizados para aceder aos índices da matriz, neste ponto temos já a *BST* cujos elementos têm como chave o número que sucede os caracteres, assim sendo basta inserir o novo código na árvore, sem nenhuma ordem particular em mente, uma vez que a única travessia ordenada pedida baseia-se em obter códigos com base no primeiro caracter, que é conseguido especificando apenas o primeiro índice-caracter da matriz.

Um elemento obtido da BST contém ainda um campo de *metadata* opcional, de modo a tornar o módulo mais versátil e aberto a outros tipos de utilizações. Neste caso a aplicação GestHiper opta por utilizar este campo como uma *HashTable*, que por sua vez contém índices que mais tarde se relacionam com as estruturas de Contabilidade e Compras.

A estruturação dos catálogos desta forma reflecte-se em ganhos de performance na aplicação, expandindo até os catálogos de forma a que possam conter adicionalmente outras informações que serão úteis para estabelecer relacionamentos com outras estruturas, ou qualquer outra informação que o utilizador da biblioteca pretenda associar aos elementos.

Contabilidade

Responsável pela resposta a perguntas quantitativas sobre Produtos em relação a Clientes, e vice-versa. A implementação de *Contabilidade* conseguida aqui consiste em estruturas que seguram a estatística recolhida do ficheiro de compras, e a sua subsequente API.

Não têm portanto o relacionamento entre clientes e produtos que se observa no módulo *Compras*, ou seja, uma entrada nestas estruturas é 100% independente de todas as outras.

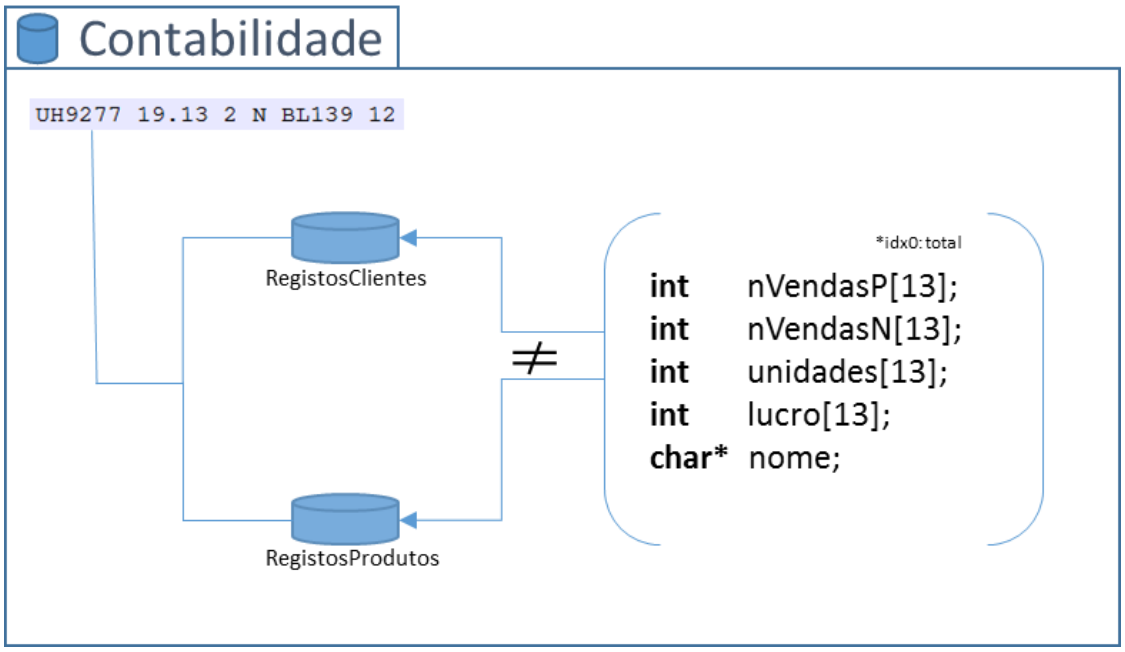

FIGURA 6 – ESTRUTURA DE CONTABILIDADE

Estrutura de Dados

A estrutura de dados usada para este módulo é composta por dois simples *arrays* de registos, um para os Clientes, um para os Produtos, que contém a informação necessária à resposta eficiente as *queries* pedidas, contendo informação sobre o número de vendas em ambos os modos (P/N), unidades vendidas e lucro, separados por mês (estando no índice 0 o somatório dos respectivos campos).

Por forma a manter *performance* à medida que o número de registos aumenta, este módulo depende da *metadata* nos catálogos para garantir acessos directos aos registos, desta forma à medida que o número de registos aumenta o custo de aceder a um certo registo mantém-se constante.

A inserção de um elemento nesta estrutura inicia-se lendo uma linha do ficheiro de compras, estando definida como *Compra*, identifica-se o Produto na *Compra* e caso ainda não exista no *array* de entradas, é criado. De seguida os valores desse Produto, para o mês especificado na *Compra*, são incrementados com o valor desta. O processo é simétrico para com os Clientes.

De modo a responder a questões relativas à ordenação entre produtos (mais comprados, etc.) os elementos do *array* são ordenados de forma crescente, recorrendo a uma *minHeap*, permitindo responder a certas *queries* com custo linear incorrendo apenas num custo inicial um pouco mais elevado.

Este processo de ordenação faz uso da funcionalidade de *metadata* opcional disponível nos catálogos. Consiste em ordenar as entradas com critério baseado no somatório das unidades desse Produto/Cliente. O novo índice ordenado desta entrada é associado ao respectivo Produto/Cliente nos catálogos. Isto trata-se de uma operação relativamente intensiva, pelo que a ordenação só é executada terminada uma ronda de inserções.

Esta ordenação não é no entanto obrigatória, servindo apenas para dispor as entradas de forma arbitrária. Se um utilizador da biblioteca estiver numa situação em que já tem o código de Produto/Cliente, ou seja, não está no seu interesse descobrir 'os mais/menos comprados', tem a mesma facilidade de acesso independentemente da estrutura estar, ou não, ordenada.

Compras

Este módulo trata de estabelecer a tabela relacional entre Produtos e Clientes, e vice-versa, relativamente a compras.

Embora o seu desenho seja bastante parecido com o da Contabilidade, este difere na natureza dos dados apontados por um Cliente/Produto, e na capacidade de relacionar directamente Produtos/Clientes com as compras que dizem respeito a estes.

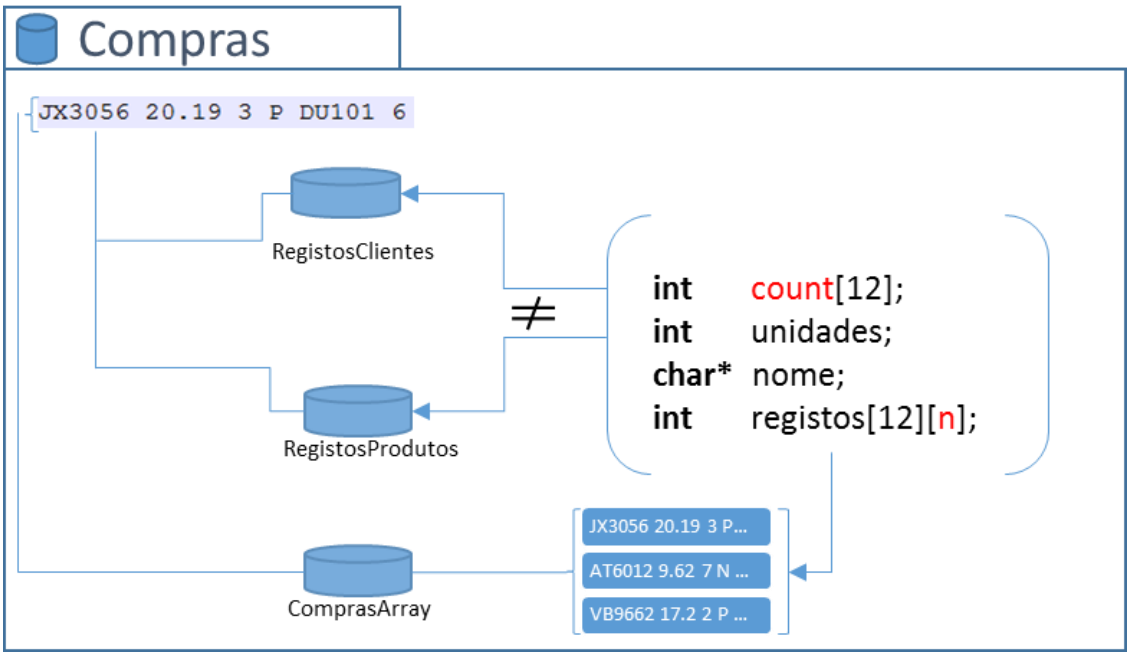

FIGURA 7 – ESTRUTURA DE COMPRAS

Estrutura de Dados

A estrutura de dados usada é muito semelhante à utilizada no módulo de Contabilidade, diferindo no conteúdo de cada entrada, sendo também ordenado da mesma forma que os registos na Contabilidade, de forma a responder a *queries* com as mesmas características.

Cada entrada nos *arrays* tem a si associado a contagem total de unidades vendidas/compradas, seguindo-se do código do elemento correspondente a esta entrada, e finalmente um *array* que ordena, por mês, a contagem de compras associadas a este elemento.

As entradas neste módulo contém ainda 12 *arrays* de registos, em que cada entrada aponta para uma compra realizada no respectivo mês. Foi feita a decisão de manter intactos os dados pertinentes a cada venda por forma a permitir extensibilidade apesar de um maior uso de memoria, permitindo assim, se necessário, adaptar o módulo a novas necessidades.

A inserção segue a mesma linha que a estrutura de Contabilidade, 'incrementando' os valores de uma entrada com a Compra a ser processada actualmente. Este processo trata de incrementar as unidades desta entrada com as unidades da Compra, seguindo-se do registo da Compra no *array* geral de *Compras*, e a consequente associação desta Compra, já colocada na estrutura *Compras*, ao seu Produto e Cliente respectivo.

A ordenação executada nestas estruturas é 100% idêntica à processada na estrutura Contabilidade, sendo que a variável ordenante trata-se do somatório de unidades ao longo do ano, aqui disponível sob a forma de uma única variável-total.

Profiling

Para efeitos de comparação e como forma simplística de avaliação de *performance*, apresenta-se de seguida uma tabela com tempos de execução para cada *query*. Os seguintes factores devem ser considerados, relativamente aos tempos apresentados, método de análise do tempo de execução e configuração do computador utilizado. De se notar, no entanto, que apesar destas considerações, os tempos apresentados são sujeitos a circunstâncias inalteráveis das situações em que os testes foram realizados: como outros processos em execução, *caching* do disco, etc…

O método empregue para levantar estes tempos faz uso da biblioteca *time.h*. Por motivos de portabilidade de código foi utilizada a função *clock()* face a *time()*, uma vez que a última difere no valor de retorno em plataformas diferentes.

A função *clock()* devolve a contagem de ciclos, os tempos são obtidos tendo em conta a constante *CLOCKS_PER_SEC*, declarada na biblioteca *time.h*, uma vez que esta terá em conta a plataforma utilizada, garantindo a portabilidade do código.

Infelizmente e apesar desta atenção, a precisão dos tempos calculados só abrange os milissegundos (sob a forma de número inteiro), sendo que se uma operação analisada não demorar mais que um milissegundo, o tempo devolvido é 0, pelo que a única certeza neste caso é *<1ms*.

O computador-portátil utilizado para calcular estes tempos possui as seguintes características relevantes:

Processador: Intel Core i5-4210U CPU @ 1.7GHz
Memória: 8GB DDR3L Single Channel @ 1600MHz
Disco: 500GB SSHD

Query #	Compras 500k	Compras 1M	Compras 3M
1	12.72 s	22.15 s	56.27 s
2	7 ms	8 ms	9 ms
3	<1 ms	<1 ms	<1 ms
4	28 ms	17 ms	<1 ms
5	<1 ms	<1 ms	1 ms
6	2 ms	3 ms	3 ms
7	7 ms	6 ms	6 ms
8	<1 ms	<1 ms	<1 ms
9	<1 ms	<1 ms	1 ms
10	6 ms	11 ms	11 ms
11	9 ms	9 ms	9 ms
12	30 ms	66 ms	106 ms
13	<1 ms	<1 ms	5 ms
14	14 ms	4 ms	<1 ms

Conclusões

Terminado o desenvolvimento à data de entrega, podemos fazer uma apreciação bastante positiva do projecto concluído, tratando-se de uma fiel representação das aptidões dos membros de equipa.

Como referido em partes anteriores do relatório, algumas optimizações possíveis não foram implementadas por forma a permitir uma maior extensibilidade. Mantemos que esta foi uma decisão acertada dadas as características do *hardware* moderno, que nos permitem ser um pouco menos conservativos com uso de memória ou tempo de CPU por forma a facilitar modificações/reutilização futura.

Todos os pontos de avaliação foram considerados e atingidos, pelo que o produto final se revela como uma aplicação robusta e bastante eficiente, como evidenciado pelos tempos de execução (tendo em conta os factores mencionados no *Profiling)*.