SUPERMARKET REPORT

February 10, 2020

Diana Egas
Francisco Gonçalves
Faculdade de Ciências da Universidade do Porto

Contents

0.1 Introdução

Este trabalho consiste na implementação do funcionamento de um supermercado. Após receber input do utilizador para inserir o valor pretendido para quatro variáveis distintas (**afluência, apetência, número de caixas e número de ciclos**), o programa imprime as caixas existentes em cada ciclo assim como algumas estatísticas e ocasionais mensagens caso um cliente seja criado ou removido.
Assim que o limite número de ciclos definido for atingido, é imprimida uma mensagem de aviso de fecho de caixas. Deste modo, deixa de ser permitida a criação de novos clientes e as caixas permanecem em funcionamento até todas estarem vazias. Por último, são imprimidas as estatísticas finais relativas a cada caixa existente.

0.2 Estrutura do Programa

Na elaboração deste trabalho decidimos dividir o nosso código em 5 ficheiros diferentes:

Quatro headers:
-queue.h: contém a estrutura para uma fila e as declarações das funções utilizadas relativamente à sua pesquisa e manutenção.

-minHeap.h: contém a estrutura para uma fila de prioridade e as declarações das funções utilizadas relativamente à sua pesquisa e manutenção.

-customer.h: contém a estrutura para dados de um cliente de supermercado assim como as declarações das funções para a sua criação e pesquisa das variáveis que constituem esta estrutura.

-cashier.h: contém a estrutura para dados de uma caixa de supermercado assim como todas as funções para a sua criação, pesquisa e manutenção entre outras operações fundamentais para o bom funcionamento do sistema de supermercado.

Quatro ficheiros respetivos:
-queue.c: contém funções fundamentais para operações com filas. Com exceção de uma função, as restantes já foram fornecidas inicialmente para a realização deste trabalho, tendo apenas sido modificadas para permitir o seu uso com a estrutura e cliente (CUSTOMER). O método enqueue() recebe como argumentos um void* e um QUEUE*, sendo que a variável genérica do tipo void* é inserida na lista dada através do segundo argumento. Quanto ao dequeue(), recebe um apontador de QUEUE e retorna o primeiro elemento dessa lista que será do tipo CUSTOMER*, o que é possível através do uso de void* que permite que a lista seja usada para uma variável de qualquer tipo.

-minHeap.c: contém funções fundamentais para operações com filas de prioridade. Todas as funções contidas neste ficheiro foram extraidas da página da cadeira de Desenho e Análise de Algoritmos. Contudo, tal como o ficheiro queue.c, também este foi alterado da mesma forma de maneira a poder ser usado para guardar apontadores para a estrutura CUSTOMER. O método compare() é usado integralmente nas restantes funções para organizar a fila de prioridade de acordo com o número de itens de cada cliente assim como o seu tempo de chegada.

customer.c: contém a função para a criação de um novo cliente, sendo-lhe fornecido como argumentos o seu número de itens, que é um número aleatório de acordo com o valor da apetência que é definido através de input do utilizador e o seu tempo de chegada que é um inteiro maior ou igual a 1 mas menor ou igual ao número de ciclos, também definido através de input do utilizador. Deste ficheiro também fazem parte os getters que retornam as duas variáveis desta estrutura: número de itens (**nItems**) e tempo de chegada (**toa**).

cashier.c: contém a função para a criação de uma nova caixa de supermercado, sendo-lhe apenas fornecida um inteiro que coincide ao número da caixa (**number**), sendo que a primeira caixa criada terá o número 1 e a última terá como número o limite definido pelo utilizador quanto ao número de caixas pretendidas. Quando aos restantes parâmetros existentes na estrutura de uma caixa, o tempo de despacho estimado de um cliente (**eta**), o número de clientes atendidos (**nCustomers**), o número de produtos processados (**nProducts**) e o tempo de espera (**waitTime**) são inicializados a zero, sendo incrementados posteriormente sempre que um cliente entra na fila dessa caixa ou quanto um cliente é despachado e sai da caixa. Quanto à velocidade de processamento de produtos (**speed**), é gerado um número aleatório entre 1 e 5 e quanto à variável **heap** é um número aleatório, podendo ser 1 ou 2. Caso heap seja 1, a caixa usa a estrutura QUEUE e cria uma fila normal. Caso contrário, recorre à estrutura HEAPMIN e cria uma fila de prioridade. As restantes funções foram adaptadas do ficheiro supermarket.py previamente disponível e convertidas de Python para a Linguagem C.

Main:
É pedido input para quatro variáveis distintas:
-afluência: percentagem de popularidade do supermercado. Quanto maior maior a probabilidade de clientes serem criados no início de todos os ciclos.
-apetência: probabilidade dos clientes comprarem mais itens. Quanto maior maior será a quantidade de itens de cada cliente.
-número de caixas: número de caixas a ser criadas para atender clientes, podendo ser normais ou caixas de prioridade, sendo que, em circunstâncias normais sáo utilizadas por clientes com 10 ou menos itens.
número de ciclos: número mínimo de ciclos que todas as caixas terão de executar para atender clientes. Após o limite dado ser atingido deixa de ser possível criar novos clientes. Contudo, as caixas permanecem em execução até ficarem vazias, ou seja, não existir nenhum cliente por atender.

Após receber os valores para as quatro variáveis em questão, chama a função runSim(), incluída no ficheiro **cashier.c** que recebe essas variáveis como argumentos e corre o simulador de supermercado, imprimindo as caixas em cada ciclo, as suas estatísticas e, após todas elas esvaziarem as suas filas, imprime as estatísticas finais de cada caixa.

0.3 Implementação de Extensões

No nosso trabalho implementamos listas de prioridade com vetores, uma estrutura de dados mais simples onde a alocação de memória é contínua e portanto implica definir um tamanho máximo da lista, os elementos são de fácil acesso, por índices, no entanto a remoção implica percorrer a lista para selecionar o elemento a eliminar.
A vantagem da implementação de listas com prioridade com listas ligadas reside no facto de por exemplo, na inserção de um elemento, neste caso, um cliente com prioridade mais elevada não ser necessário a mudança de lugar dos outros, apenas mudar o apontador, a cada nó corresponde um valor e um nó para o próximo elemento da lista.

0.4 Conclusão

Este trabalho foi fundamental na aquisição e aperfeiçoamento de conhecimentos e capacidades no que toca a múltiplas operações que envolvem estruturas com diferentes tipos de variáveis assim como a sua utilização em diversas estruturas de dados. Foi também essencial iniciar este trabalho a partir de ficheiros de código e estruturas já existentes em vez de recorrer a uma criação de raíz. Foi também inovador e igualmente importante converter um ficheiro de uma linguagem em que tínhamos poucos conhecimentos (Python) para uma linguagem familiar (C), o que permitiu um aprofundamento de conhecimentos das duas linguagens.