**Relatório Trabalho 01 – Fábrica de Brinquedos**

Este trabalho é referente à disciplina Algoritmos e Estruturas de Dados I sobre os temas Pilha, Fila e Lista e suas diversas implementações além dos conceitos e implementações de TAD (Tipo Abstrato de Dados). A ideia do projeto é recriar uma fábrica de brinquedos onde através de valores conhecidos de quantidade de brinquedos, espaçamento de coleta e capacidade de armazenamento, é possível mimetizar uma esteira de coleta de brinquedos (lista circular), caixas de armazenamento de brinquedos(pilhas) e uma fila com todas as caixas (fila de pilhas).

As implementações foram realizadas de 2 (duas) formas, “Estática Sequencial” e “Dinâmica Encadeada”, apresentando diferenças entre si e suas estruturas de dados, ambas sendo apresentadas e explicadas adiante.

Em ambos os casos, temos a separação em [Main, Lista, Pilha e Fila].c e [Lista, Pilha e Fila].h. Começando pela lista, uma definição de um valor MAX determinado nas especificações do trabalho (para a implementação estática) e a inclusão das bibliotecas necessárias, foram realizadas em lista.h. Structs foram utilizadas para criar a lista circular e facilitar a criação do tipo “brinquedo” onde foram armazenadas as informações como número serial (inteiro) e o nome(um vetor de char).

Os cabeçalhos das funções utilizadas também foram descritos, sendo funções de Criação, Inserção, Remoção e Verificações de Lista Cheia ou Vazia, como determinado nas especificações, para a implementação estática e as mesmas sem a verificação de “cheia” na dinâmica. Passando assim para “lista.c”, inicialmente foram adicionadas as bibliotecas necessárias e a “lista.h” para que se pudesse utilizar as structs necessárias, criando por fim as funções necessárias. Assim temos funções para criar a lista (posicionando os ponteiros ou a posição de vetor), adicionar um brinquedo na lista (preenchendo as posições do vetor ou criando novos nós para encadeamento), remover da lista (onde a informação era repassada e adicionada nas pilhas, sendo substituída por um valor não utilizado no caso estático e liberando o espaço anteriormente ocupado no caso dinâmico) e as verificações de pilha cheia/vazia para se ter um ponto de inicio/parada e determinar possíveis erros.

A estrutura de pilha.h apresenta a inclusão da lista.h, uma vez que o tipo brinquedo será utilizado, pois as pilhas que foram criadas iriam “guardar” os brinquedos. As structs necessárias para pilha foram criadas e os cabeçalhos de funções registrados. Em pilha.c, além das inclusões de bibliotecas necessárias e do pilha.h, as funções de Criar pilha (cria a pilha posicionando as informações de topo e numero de elementos por exemplo), Adicionar e Remover brinquedos (Push e Pop) e Verificação se as pilhas estavam cheias, foram criadas.

Por fim, a fila de pilhas foi adicionada. A inclusão do pilha.h foi necessária, criando a estrutura de fila necessária e os cabeçalhos de funções, no fila.h. Assim como na lista, as funções criadas e implementadas em fila.c foram a de Criação, Inserção, Remoção e Verificação de fila vazia e cheia(esta apenas na versão estática) e de impressão da fila para finalização e apresentação de saída (também apenas na versão estática) A última é uma função extra criada para facilitar a impressão de contagem de quantidade e as informações dos brinquedos separados.

Apesar de estruturas muito parecidas, a implementação dos tipos estático e dinâmico foi diferente. Para a primeira versão, as estruturas de Lista, Fila e Pilha necessitaram de criação de apenas uma struct (duas para lista, onde foi criado a struct para o tipo lista e para o tipo brinquedo), onde eram criadas variáveis (normalmente inteiras) para guardar informações como inicio, fim, tamanho e posição. Vetores de tipos determinados (brinquedo e pilhas) também eram apresentados como forma de armazenar as informações principais. Na versão dinâmica as mesmas variáveis eram criadas, porém como ponteiros, indicando a posição que cada variável representa. Uma struct de “nó” também foi criada para cada tipo, para ser alocada dinamicamente durante a inserção ou liberada durante remoção.

Quando se entra na questão das funções, na versão estática é fácil de verificar que são constituídas de “movimentação” dentro dos vetores para verificar, adicionar e remover as informações, além dos incrementos e decrementos necessários nas contagens. Percebe-se também que nas remoções nesta versão, a posição ficava vazia mas ainda reservada, uma vez que era sequencial e estática. Tais posições eram preenchidas com valores que não seriam utilizados (normalmente -2, etc) para que as posições ficassem marcadas como “vazias”. Na versão dinâmica, cada inserção e remoção, mexia-se com os ponteiros para indicar onde estaria o inicio, fim e próximas posições. A remoção não apresenta tantos problemas, pois era possível salvar a informação e liberar o espaço ocupado pela mesma além de criar apenas o que era necessário.

Como o limite máximo era alto, 1000 (mil) posições, caso poucas sejam utilizadas, seria mais vantajoso utilizar a implementação dinâmica, uma vez que com ela só se aloca espaço necessário para cada utilização, enquanto o que o estático aloca as 1000 posições desde início mesmo que não sejam usadas. O fato ainda de que brinquedos são movidos o tempo todo, ter a possibilidade se “desalocar” um espaço não mais utilizado também é uma vantagem que a implementação dinâmica apresenta. Como as estruturas, variáveis e funções criadas são muito similares, não há grande diferença entre os 2 tipos e apesar de a dinâmica requerer um pouco mais de cuidados e atenção em sua criação, acreditamos que a implementação Dinâmica Encadeada seria mais recomendada para este caso.