Relatório

Algoritmos e Estruturas de Dados II

Aluno/os: Pedro Vieira Simões e João da Costa Apresentação

Professor/es: Alberto Simões e Óscar Ribeiro

Licenciatura em Engenharia de Sistemas Informáticos

Barcelos, maio, 2021

Resumo

Relatório de um programa feito em C, cujo objetivo é armazenar, alterar e organizar informações acerca de um inventário da LEGO, no qual há peças e conjuntos formados por essas peças.

Palavras-Chave:

Programa, C, Linux, LEGO, Inventário, Parts, Parts_Sets, Sets, Peças, Conjuntos, ID, Estrutura, Apontadores.

Informação do programa e estruturas usadas

Este projeto tem como objeto armazenar, alterar e organizar informações acerca de um inventário da LEGO, no qual há peças e conjuntos formados por essas peças.

No enunciado estava presente uma explicação do conteúdo escrito em cada file e a relação entre as mesmas, tinha também presente diferentes pontos, cada um referindo-se a uma funcionalidade diferente que o programa deveria ter, tal como está explicito nas imagens abaixo:

FILES:

1. `sets.tsv`: (set_num, name, year, theme)
   Contém os conjuntos oficiais da LEGO, indicando o seu código, o nome pelo qual o conjunto é comercializado, o ano em que foi colocado no mercado, e o tema em que se insere (duplo, technic, etc).
2. `parts.tsv` (part_num, name, class, stock)
   Contém os dados das diversas peças existentes, indicando a sua identificação, o seu nome, a classe (tipo de peça) e a quantidade atualmente em stock.
3. `parts_sets.tsv` (set_num, quantity, part_num)
   Contém os dados necessários para a definição da constituição de cada conjunto oficial da LEGO, detalhando o conjunto, a quantidade de cada peça que é utilizada, e a identificação da peça.

FUNCIONALIDADES:

Pretende-se com este conjunto de dados calcular:

1. Quais os conjuntos de determinado tema (ordenados pelo ano);
2. As peças de determinado tipo em determinado conjunto;
3. Quais as peças necessárias para construir um dado conjunto, indicando os dados de cada peça e respetiva quantidade;
4. O total de peças em stock;
5. O total de peças incluídas num determinado conjunto;
6. A peça que é utilizada em mais conjuntos diferentes, independentemente da quantidade em cada um deles;
7. A lista dos conjuntos que se conseguem construir com o stock existente.

Para além dos dados lidos a partir dos ficheiros providenciados, devem existir opções para:

- Alterar o número de peças em stock;
- A adição de stock com base no identificador de um conjunto (adicionar as peças que esse conjunto tem ao stock total de peças).
- Remover todas as peças de determinada classe (tipo de peça);
- Remover todos os sets de determinado tema

Para tal foi utilizado 3 listas duplamente ligadas diferentes, cada uma guardava informação de cada file, respetivamente, tal como é apresentado na imagem seguinte:

```
typedef struct _parts
{
    char *part_num, *name, *class;
    int stock;
    struct _parts *next, *previous;
} Parts;

typedef struct _parts_sets
{
    char *set_num, *part_num;
    int quantity;
    struct _parts_sets *next, *previous;
} Parts_Sets;

typedef struct _sets
{
    char *set_num, *name, *theme;
    int year;
    struct _sets *next, *previous;
} Sets;
```

A lista Parts contém o ID da peça, nome, classe e o seu stock, juntamente com os devidos apontadores para montar a lista.

A lista Sets contém o ID do conjunto de peças (LEGO completo), nome, tema e o seu ano de lançamento, juntamente com os devidos apontadores para montar a lista.

A lista Parts_Sets é o que faz haver ligação entre as 2 listas anteriormente referidas, contém o ID do conjunto de peças e o ID da devida peça pertencente a esse conjunto, bem como a quantidade dessas peças que o conjunto necessita, juntamente com os devidos apontadores para montar a lista.

Funcionalidades do programa

O programa apresenta todas as funcionalidades pedidas no enunciado, tendo esta funções para:

- Apresentar um menu e o seu texto;
- Remover listas inteiras do tipo Parts, Parts_Sets e Sets;
- Remover nodos específicos, tendo em conta a sua classe ou tema;
- Listar as listas por completo;
- Ler todos os dados em files e armazenar em listas do tipo Parts, Parts_Sets e Sets;
- Inserção de dados à cabeça em qualquer lista do tipo Parts, Parts_Sets e Sets;
- Inserção de dados por ordem de ano em qualquer lista do tipo Sets e por ordem part_num em qualquer lista do tipo Parts;
- Obtenção de IDs, classes e temas de diferentes peças ou conjuntos a partir do input do utilizador e verificação da existência das mesmas nas listas;
- Listar uma lista Sets por ordem de ano, de um certo tema;
- Mostrar todas as peças de uma determinada classe num certo conjunto;
- Mostrar as peças necessárias para construir um certo conjunto;
- Mostrar o número de peças necessárias num conjunto;
- Mostrar o número total de peças existente no stock;
- Alterar o número de peças no stock;
- Adicionar um conjunto ao stock (adicionar 1 peça de cada as que são precisas para montar esse conjunto);
- Mostrar a peça mais utilizada em conjuntos diferentes;
- Mostrar todos os conjuntos possíveis de serem construídos a partir do inventário de peças atuais;

Coisas a melhorar no programa

Apesar de muito trabalho e esforço da parte do nosso grupo ao ter feito o programa, notamos que havia alguns parâmetros a melhorar, tais como:

- ➔ Utilizar uma estrutura de dados mais eficaz para a pesquisa de informação, exemplo: Binary Trees. A razão de não termos aplicado esta estrutura foi por termos começado cedo o projeto e já tínhamos as listas feitas e o grupo decidiu continuar com as mesmas, todavia, foi perdido imensa eficiência em algumas funções pelo mesmo;

- ➔ Melhor aproveitamento do apontador *previous, houve casos em que o apontador previous era bastante útil mas optamos pela abordagem de uma variável auxiliar que guardava os dados atuais;

- ➔ Função do ponto 6) do enunciado demora imenso tempo até dar um resultado (por volta de 5 minutos), devido, mais uma vez à possível má escolha da estrutura de dados;

- ➔ Guardar informação em files para mais tarde carregar alterações previamente feitas, a mesma não foi feita devido a não termos tido tempo para tal, o que fez com que o grupo se focasse mais na parte principal do trabalho;

Informação extra acerca do programa

O grupo achou este trabalho muito interessante e entusiasmante, revelando também imensos progressos no que toca a estruturas dinâmicas, nomeadamente listas ligadas e adquiriu um melhor conhecimento a nível algorítmico de como enfrentar certos problemas e enunciados em programação. Mesmo assim, o grupo apresentou alguma dificuldade em fazer a função dos pontos 6 e 7, sendo que ambas funcionam, todavia a função 6 é pouco eficaz, apesar de apresentar o resultado correto

O programa foi feito numa Virtual Machine de Linux Ubuntu 64-bit, no IDE Visual Studio Code com o compilador gcc, tendo assim auxílio de diversas funcionalidades na linha de comandos como o gdb.