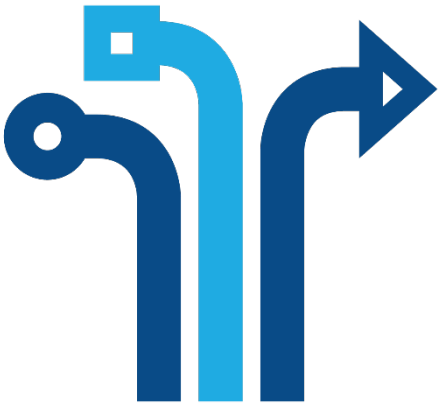
Trabalho Prático – Algoritmos e Estruturas de Dados 2



ESCOLA
SUPERIOR
DE TECNOLOGIA
IPCA

Duarte Ribeiro de Melo, 21149

Abordagem e estrutura de dados escolhida

Depois da leitura do enunciado, e após breve análise aos exercícios pedidos, decidi utilizar listas duplamente ligadas, com uma outra estrutura com dois apontadores – para o primeiro nodo da lista e para o último.

Sempre que um elemento é adicionado na lista, é adicionado ao fim da mesma.

```
 */

/**
 * @brief
 * part_num: Identificação da peça.
 * name: Nome da peça.
 * part_class: Classe da peça.
 * Stock: stock da peça existente.
 */
typedef struct _part
{
    char *part_num;
    char *name;
    char *part_class;
    int stock;
    struct _part *next, *prev;
} Part;

/**
 * @brief Apontadores para início e fim da lista das parts.
 *
 */
typedef struct _partsList
{
    Part *first;
    Part *last;
} PartsList;

```

Funcionalidades do programa

O programa cumpre com praticamente tudo o que é pedido no enunciado, exceto a alínea 7:

- A lista dos conjuntos que se conseguem construir com o stock existente.

Dada alguma falta de tempo, e falta de boa gestão do mesmo, fiz o trabalho um pouco em cima do dia de entrega, o que originou a falta desta alínea (que considero ter capacidade para realizar a mesma).

```
#pragma region Parts

extern PartsList *new_parts_list();
extern Part *new_part();
extern void add_part(PartsList *list, Part *item);
extern void change_part_data(Part *part, char *part_num, char *name, char *part_class, int stock);
extern void print_part_data(Part *part);
extern void print_part_list(PartsList *parts);
extern void change_stock(Part *part, int newStock);
extern Part *find_part_in_list(char *part_num, PartsList *parts);
extern int total_stock(PartsList *parts);
extern void remove_parts_per_class(PartsList *parts, char *part_class);
extern Part *most_used_part(PartsList *parts, PartsSetList *partssetslist);

#pragma endregion

#pragma region PartsSets

extern PartsSetList *new_parts_sets_list();
extern PartsSet *new_parts_set();
extern void add_parts_set(PartsSetList *list, PartsSet *item);
extern void change_parts_set_data(PartsSet *partset, char *set_num, int quantity, char *part_num);
extern void print_parts_set_data(PartsSet *partset);
extern void print_partset_list(PartsSetList *partsset);
extern void parts_in_partsset_per_class(PartsSetList *list, PartsList *parts, char *part_class, char *set_num);
extern void parts_to_build_set(PartsSetList *list, PartsList *parts, char *set_num);
extern void parts_quantity_to_build_set(PartsSetList *list, char *set_num);
extern void add_stock_from_partsset(PartsSetList *list, char *set_num, PartsList *parts);

#pragma endregion

#pragma region Sets

extern SetList *new_sets_list();
extern Set *new_set();
extern void add_set(SetList *list, Set *item);
extern void change_set_data(Set *set, char *set_num, char *name, int year, char *theme);
extern void print_set_data(Set *set);
extern void print_set_list(SetList *sets);
extern void organize_setsList_per_year(SetList *sets);
extern void print_sets_per_theme_year(SetList *sets, char *theme);
extern void remove_sets_per_theme(SetList *sets, char *theme);

#pragma endregion
```

Aspetos a melhorar / dificuldades encontradas

Na minha opinião, a eficiência de algumas funções podia ser melhorada.

Em relação ao *makefile*, tenho ainda alguma dificuldade na criação/lógica do mesmo, e, mais uma vez, por falta de tempo, não o criei.

A organização dos ficheiros podia ser melhor, mas, mais uma vez, sem estar confortável com *makefiles*, os ficheiros terão de estar todos "ao lado" uns dos outros.

Em suma, a eficiência de algumas funções, o makefile, o facto de não ter feito o exercício 7 e a organização dos ficheiros são aspetos a melhorar para futuros trabalhos.

Ferramentas utilizadas

- VS Code;

- Git e GitHub;

- MinGW;

- Draw.io;

Links úteis

- Email: a21149@alunos.ipca.pt

- GitHub: https://github.com/duartemelo/