INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

Trabalho Prático Programação 1

Linguagem C: Estruturas de dados dinâmicas

#include<stdio.h>

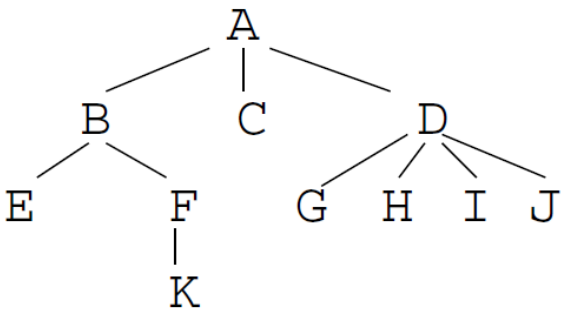
Autores: Luís Pereira nº14868;
Daniel Mendes nº 14871 ;
Marcelo Barbosa nº 14163;

Curso: Informática Médica

Orientador Científico: Prof. João Carlos Silva

Decomposição do Problema

Neste trabalho é pedido para implementar um programa na linguagem C que faça o registo de componentes eletrónicos, bem como sua listagem, pesquisa e remoção, e ainda a construção de um equipamento eletrónico através dos componentes registados em forma de uma árvore. O programa deverá ser capaz de escrever e ler a sua informação em ficheiro de texto.



Implementação

- Estruturas –

  Lista de componentes:

  Estrutura básica de uma lista ligada, com os campos destinados à informação do componente e um apontador para o registo do próximo componentes

```
typedef struct reg {
    int cod;
    char nome[100];
    float preco;
    struct reg *next;
}*COMP;
```

  Árvore do equipamento:

  Estrutura básica de uma árvore genérica, com os apontadores para os registos filhos e irmãos.

  Os campos "*id, des e preco*" estão reservados à informação do componente associado.

  O campo cod é o número do registo da árvore, permitindo assim repetir os componentes.

```
typedef struct registo {
    int cod;
    int id;
    char des[30];
    float preco;
    struct registo * irmaos;
    struct registo * filhos;
} * AG;
```

- Menu – Inserir o número da função que deseja executar

```
**********[SISTEMA REPRESENTAÇÃO COMPONENTES ELETRÓNICOS] **********
*                                                                   *
*              1. Inserir um novo componente                        *
*              2. Listar componentes                                *
*              3. Pesquisar componente por número                   *
*              4. Pesquisar componente por nome                     *
*              5. Alterar componente                                *
*              6. Remover componente                                *
*              7. Definir equipamento eletrónico                    *
*              8. Consultar equipamento eletrónico e seu custo total*
*              9. Guardar componentes em ficheiro                   *
*              0. Sair                                              *
*                                                                   *
*                          $ Informática Médica $                   *
************************** $ IPCA-EST 2017-2018 $ ********************************
---->Introduza a opção:
```

- Inserir Componente – Função que permite inserir um novo componente numa lista ligada. Cada componente deve conter:
  - Código;
  - Designação;
  - Preço;

```
[INSERIR NOVO COMPONENTE]

        NOTA: Por predefinição, o código do componente é gerado automaticamente por ordem crescente

        Código:5
        Introduza o nome do componente:amplificador
        Introduza o preço do componente:10


                *Componente inserido com sucesso*
```

Nota: Para facilitar a fase de testes, e a utilização do programa em geral, o código atribuído aos componentes é gerado automaticamente por ordem crescente através da função *cod_returner*.

Esta função retorna o código do último registo da lista +1.

Se a lista tiver vazia, o código será 1.

Nota: Se um registo for apagado, o número do seu código não voltará a estar disponível.

```
int cod_returner(COMP start) {
    int cod=0;
    COMP ptr=start;
    if(start==NULL) return 1;
    else {
        while(start!=NULL) {
            cod=start->cod;
            start=start->next;
        }
        return cod+1;
    }
}
```

- Listar Componentes – Esta função realiza uma listagem de todos os componentes da lista, bem como a sua informação.

Nota: Foi pedido para realizar a listagem por ordem crescente dos códigos dos componentes, porém, como a inserção dos componentes tem o código gerado automaticamente por ordem crescente, não foi preciso realizar nenhuma função de ordenação.

No final da listagem é dado o número de componentes registados.

```
[COMPONENTES REGISTADOS]

        Código:1
        Nome:BOBINE
        Preço:12,00

        Código:2
        Nome:RESISTENCIA
        Preço:40,00

        Código:3
        Nome:CONDENSADOR
        Preço:30,00

        Código:4
        Nome:DIODO
        Preço:9,00

        Código:5
        Nome:AMPLIFICADOR
        Preço:10,00


Componentes registados: 5


                *Listagem completa*
```

- Pesquisar componente por código – Função que pede ao utilizar para introduzir o um código, se esse código existir, a função imprime no ecrã a informação desse componente.

```
[PROCURAR COMPONENTE PELO NUMERO]

        Insira o número a procurar:3

        Código:3
        Nome:CONDENSADOR
        Preço:30,00


                *COMPONENTE ENCONTRADO COM SUCESSO*
```

- Pesquisar componente por designação – Semelhante à função anterior, é pedido ao utilizar para introduzir um nome de um componente, se o nome existir, a função imprime no ecrã a informação desse componente.

  Nota: Graças ao algoritmo usado para procurar o nome introduzido na lista, é possível o utilizador introduzir apenas uma letra, a função irá imprimir informação de todos os componentes em que o nome começa por essa letra.

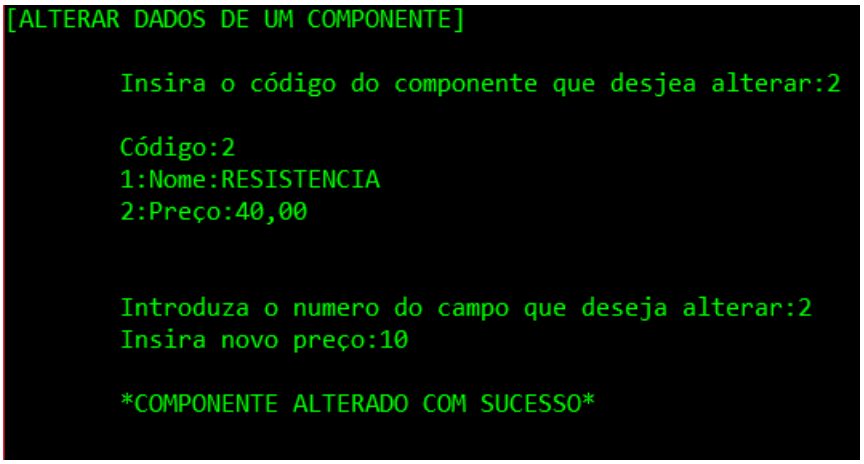
```c
int pesquisa_nome(COMP start, char nome[]) {
    int k=0, cont=0, find=0;
    while(start!=NULL) {
        cont=0;
        for(k=0; k<strlen(nome); k++) {
            if(start->nome[k]==nome[k])cont++;
        }
        if(cont==strlen(nome)) {
            printf("\n\tCódigo:%i\n\tNome:%s\n\tPreço:%.2f\n\n",start->cod, start->nome, start->preco);
            find=1;
        }
        start=start->next;
    }
    return find;
}
```

Exemplo:

```
[PROCURAR COMPONENTE POR NOME]

        Insira o nome a procurar:r

        Código:2
        Nome:RESISTENCIA
        Preço:40,00


        Código:5
        Nome:RETIFICADOR
        Preço:12,00


        Código:6
        Nome:RATO
        Preço:14,00


                *COMPONENTE ENCONTRADO COM SUCESSO*
```

- Alterar componente – Esta função permite alterar os campos "*nome e preço"* do componente cujo código é introduzido.

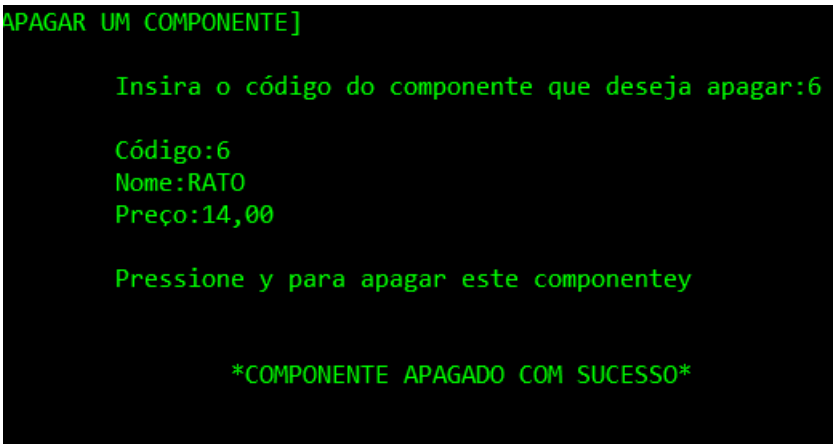
  Após a interseção de um código, se o componente existir, será mostrada a sua informação atual, o utilizador terá depois de introduzir "*1*" ou "*2*", para escolher entre alterar nome ou preço, respetivamente.



Nota: Se um componente já estiver associado à árvore de um equipamento, ele permanecerá na árvore tal como era, a informação só será alterada na lista de componentes.

- Remover componente – Dado um código, se existir, a informação do componente é exibida no ecrã, bem como um pedido para o utilizador confirmar se realmente o quer remover.

  Para remover o utilizar tem de pressionar "*y*".

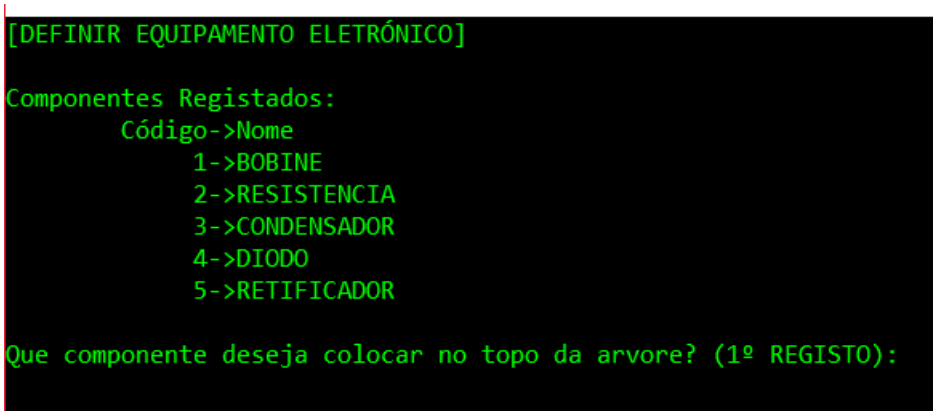


Nota: Se um componente já estiver associado à árvore de um equipamento, ele permanecerá na árvore, apenas será removido da lista de componentes.

- Definir equipamento eletrónico – Esta função permite construir um equipamento eletrónico em forma de árvore por cada sessão. Os elementos da árvore são dados pelos componentes da lista ligada e cabe ao utilizador apenas fazer as ligações entre registos da arvore e componentes que pretende.

    1. Na execução da função, inicialmente, será feita uma listagem dos componentes da lista.

    2. De seguida é feito um pedido ao utilizador para escolher que componente deseja colocar no topo da árvore (1ºregisto).



    3. A partir daí, o utilizador poderá fazer as ligações que quiser introduzindo-as com o seguinte formato:
    (“nº registo pai”->”código componente que deseja atribuir como filho”)

    Exemplo: “1->3” faz a ligação do topo da árvore (registo 1) ao componente com código 3.

    Se o registo pai e o componente existirem, uma mensagem aparecerá com o número do novo registo criado, caso contrário, uma mensagem de erro aparecerá.

    O utilizador pode fazer as ligações que quiser, quando quiser sair apenas precisa de introduzir “*0*”.

Exemplo:

```
[DEFINIR EQUIPAMENTO ELETRÓNICO]

Componentes Registados:
        Código->Nome
             1->BOBINE
             2->RESISTENCIA
             3->CONDENSADOR
             4->DIODO
             5->RETIFICADOR

Que componente deseja colocar no topo da arvore? (1º REGISTO):1

                Topo da árvore  -  1-BOBINE

        Escreva a ligação no formato (nºregisto pai->código componente filho)
                Exemplo (1->5)

        NOTAS:
                Insira 0 para terminar
                Se o registo pai introduzido não existir, nenhuma ligação não feita

Fazer ligações:
1->3
        Registo 2 criado

→(1)    3->2
        Registo ou componente inválido.  A ligação não foi feita
→(2)    1->6
        Registo ou componente inválido.  A ligação não foi feita
2->3
        Registo 3 criado

3->1
        Registo 4 criado

0


                *ÁRVORE DO EQUIPAMENTO CONSTRUÍDA COM SUCESSO*
```

(1) Como podemos ver, quando o utilizador tentou atribuir ao registo 3 o componente 2, a mensagem erro apareceu, pois o registo 3 ainda não tinha sido criado.

(2) Mensagem de erro, pois o componente 6 não existe.

→(1)
→(2)
→(3)

```
    while(scanf("%i->%i", &pai, &filho)==2) {
        ptr1=ptr2;
        find2=0;

        while(ptr1!=NULL) {
            if(ptr1->cod==filho) {
                aux=ptr1;
                find2=1;
            }
            ptr1=ptr1->next;
        }
        if(find2==1&&codx(ptr,pai)==1) {
            ptr=inserirFilho(ptr,pai,0,filho,aux->nome,0,aux->preco,i);
            printf("\tRegisto %i criado\n\n", i);
            i++;
        } else {
            printf("\tRegisto ou componente inválido.  A ligação não foi feita\n");
        }
    }
    return ptr;
}
```

(1) Este ciclo "*while"* permite ao utilizador fazer ligações continuadamente. Quando o utilizador insere 0, o ciclo para, pois a função *scanf* não retorna 2 valores lidos.

(2) Verificar se o componente filho introduzido existe.

(3) A verificação da existência do registo pai é feita pela função "*codx*". Se tanto o componente como o registo existirem, a ligação é feita.

Função "*codx":*

```
int codx(AG ptr, int cod) {
    if(ptr==NULL) return 0;
    else if (ptr->cod==cod) return 1;
    else {
        return(codx(ptr->filhos,cod)||(codx(ptr->irmaos,cod)));
    }
}
```

- Consultar equipamento e seu custo total – Função que lista as ligações da árvore e exibe a soma de todos os preços dos registos.

  Listagem e Custo da árvore construída no exemplo acima:

```
[LIGAÇÕES DA ÁRVORE DO EQUIPAMENTO ELETRÓNICO]

FORMATO:
        (Registo) codigo-nome-preço  ->  codigo-nome-preço (Registo)

(1) 1-BOBINE-12,00  ->  3-CONDENSADOR-30,00 (2)
(2) 3-CONDENSADOR-30,00  ->  3-CONDENSADOR-30,00 (3)
(3) 3-CONDENSADOR-30,00  ->  1-BOBINE-12,00 (4)


        Custo total do equipamento=84,00
```

  Função "*custo total*":

```
float custototal(AG ptr) {
    if(ptr==NULL) return 0;
    return(ptr->preco + custototal(ptr->filhos)+custototal(ptr->irmaos));
}
```

- Guardar e ler de ficheiro – O programa permite salvar os componentes introduzidos em ficheiro de texto, para numa próxima sessão os conseguir ler de novo para a lista.

  Função "*save*":

```
void save(COMP ptr) {
    FILE *file;
    file = fopen("Lista Componentes.txt", "w");
    if(file!=NULL) {
        while(ptr!=NULL) {
            fprintf(file, "%i\n%s\n%f\n",ptr->cod, ptr->nome, ptr->preco);
            ptr=ptr->next;
        }
    }
    fclose(file);
}
```

  Para o programa salvar a informação em ficheiro é necessário introduzir "*9*" no menu antes de fechar a aplicação

```
case 9:
    save(start);
    break;
```

Função “*read_file*:

```
COMP read_file(COMP ptr) {
    FILE *file;
    int cod;
    char nome[100];
    float preco;
    file = fopen("Lista Componentes.txt", "r");
    while (fscanf(file, "%i\n%s\n%f\n", &cod, &nome, &preco)!=EOF) {
        ptr=InsComp(ptr, cod, nome, preco);
    }
    fclose(file);
    return ptr;
}
```

Problemas/Comentários

Todas as funções implementadas que dizem respeito à lista de componentes foram executadas com relativa facilidade, pois tínhamos visto em aula a maior parte das funções pedidas, apenas necessitavam de ser ajustados ao enunciado.

Os problemas apenas começaram a surgir aquando da implementação da construção da árvore genérica, pois, mesmo tendo disponíveis as funções codificadas pelo professor na aula, o tempo de abordagem e discussão sobre o tema foi curto, o que contribuiu para a complexidade do problema.