< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

Vitor Ferreira França

Documentação tp-3

Introdução:

O seguindo trabalho tem como objetivo a utilização da arvore binaria para armazenar e “traduzir” códigos Morse para mensagens em texto e mensagens em texto para códigos Morse. O trabalho é dividido em uma “main” onde sera feita a aplicação desejada, um arquivo “.c” onde o desenvolvimento das funções é feitas, e um arquivo “.h” onde estará a declaração das estruturas abstratas.

Estrutura usadas:

```
typedef struct No {

    char letra;
    char codigo[10];
    struct No *esquerda ,*direita;


}No;

typedef struct letra{

    char codigo[10];
    char caractere;

}letra;
```

-Na estrutura é declarada a letra na qual corresponde o código e o código em si, logo após é declarado dois ponteiros para desenvolvimento da arvore um para a folha da esquerda e outra para folha da direita.

-A segunda estrutura foi usada para criar um vetor para tradução por pesquisa sequencial para o inverso da outra tradução.

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

MakeFile:

Estrutura do Makefile

```
all:funcoes.o
    gcc -g funcoes.o main.c -o main

funcoes.o: lista_de_funcoes.h
    gcc -c -g funcoes.c
```

Funções utilizadas:

```
No *cria_arvore();
void free_arvore(No *raiz);
void traduz_letra(No *raiz,char x[10]);
void print_arvore(No *raiz);
void traduz_Palavras(No *raiz,char *string);
void traduz_pra_morse(letra *vetor,char x);
```

Função cria_arvore

-Esta é a maior função desta trabalho, nela é criado a raiz e todas suas respectivas sub-arvores e folhas, sendo primeiramente todas alocadas dinamicamente.

-A função não recebe nem um parâmetro e retorna o No raiz principal.

```
No  *start=(No*)malloc(sizeof(No))
    ,*A=(No*)malloc(sizeof(No))
    ,*B=(No*)malloc(sizeof(No))
    ,*C=(No*)malloc(sizeof(No))
    ,*D=(No*)malloc(sizeof(No))
    ,*E=(No*)malloc(sizeof(No))
    ,*F=(No*)malloc(sizeof(No))
    ,*G=(No*)malloc(sizeof(No))
    ,*H=(No*)malloc(sizeof(No))
    ,*I=(No*)malloc(sizeof(No))
    ,*J=(No*)malloc(sizeof(No))
    ,*K=(No*)malloc(sizeof(No))
    ,*L=(No*)malloc(sizeof(No))
```

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

```
    ,*M=(No*)malloc(sizeof(No))
    ,*N=(No*)malloc(sizeof(No))
```

<Todas as letras e números ate 9, foram alocadas desta forma>

```
start->esquerda = E;
start->direita = T;
```

<Nesta etapa foi apontado onde a raiz aponta ,neste caso a raiz principal não possui nem um outro conteúdo>

```
E->letra = 'e';
strcpy(E->codigo, ".");
E->esquerda = I;
E->direita = A;


A->letra = 'a';
strcpy(A->codigo, ".-");
A->esquerda = R;
A->direita = W;

R->letra = 'r';
strcpy(R->codigo, ".-.");
R->esquerda = L;
R->direita = NULL;

L->letra='l';
strcpy(L->codigo, ".-..");
L->esquerda=NULL;
L->direita=NULL;
```

<As letras e códigos e suas respectivas folhas da esquerda e direta aqui são declaradas >

Com este formato:

Primeira linha = qual letra correspondente.

Segunda linha =qual o código morse correspondente deste caracter.

Terceira linha =para qual próximo No esta apontando na esquerda.

quarta linha =para qual próximo No esta apontando na direita.

```
return start;
```

<E no final é retornado o No raiz >

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

Função free_arvore:

-Função designada para desalocar a memora das alocações feitas pela “cria_arvore”.

-A função deve receber um ponteiro que aponta para a raiz para assim percorrela.

```
void free_arvore(No *raiz){
        if(raiz){
            free(raiz);
            free_arvore(raiz->esquerda);
            free_arvore(raiz->direita);
        }

}
```

<2-6>primeiramente é checado se a raiz é nula, se não deve-se ocorrer o processo.

<4-5> Chama-se recursivamente para percorrer a raiz.

<3> É liberado a memória.

Função print_arvore:

-Função designada para printar a arvore por completo com todas suas letras e códigos utilizando o caminhamento “pre-ordem”.

-Deve receber o no da raiz principal.

```
void print_arvore(No *raiz){
   if(raiz){
    printf("codigo correspondete [%s] ao caracter:%c\n",raiz-
>codigo,raiz->letra);
    print_arvore(raiz->esquerda);
    print_arvore(raiz->direita);
   }
}
```

<2-7> Primeiramente é chegado se a raiz existe.

<3>Nesta linha é dado o comando para printar o código e a letra correspondente da raiz do momento.

<5-6> Nesta parte a arvore é percorrida por recursão.

Função traduz_letra:

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

-Esta função fica responsável por traduzir uma letra em código morse para uma letra “normal”, percorrendo a arvore e procurando o código desejada.

-A função deve receber a raiz e uma string com o código.

```
void traduz_letra(No *raiz,char x[10]){
    int teste;

    if(raiz){

    teste = strcmp(raiz->codigo,x);
        if(teste == 0){
            printf("%c\n",raiz->letra);
        }

        else{
            for(int i = 0;i<10;i++){
                int teste2;

                if(x[i] == '.'){
                    raiz = raiz->esquerda;

                    teste2 = strcmp(raiz->codigo,x);
                    if(teste2 == 0){
                        printf("%c\n",raiz->letra);
                        break;
                    }

                }

                if(x[i] =='-'){
                    raiz = raiz->direita;

                    teste2 = strcmp(raiz->codigo,x);
                    if(teste2 == 0){
                        printf("%c\n",raiz->letra);
                        break;
                    }
                }
            }
        }

    }
}
```

<1> É declarado oque denotara se o código é igual ao da raiz no momento,

No caso o teste.

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

<3> É visto se a raiz existe, se não a função acabara.

<5> É utilizado a função “strcmp” que compara as duas strings e volta 0 se forem iguais e 1 se não.

<6-8>Se o teste for zero quer dizer que é a letra desejada, então deve-se printar a letra.

<10-36>Se o teste não provar que as strings são iguais deve-se percorrer a arvore ate achar.

<11-21>Nesta parte é colocado um “for” para percorrer a string desde o primeiro caractere até o ultimo que esteja preenchido.

<12>É declarado mais um teste para comparação.

<14> Este if checa se o primeiro caractere é “.” ou “-“,se for “.” devera percorrer a arvore a esquerda.

<18> Aqui é comparado se este a esquerda é a letra do código desejado.

Se sim deve-se imprimir a letra e sair do laço.

<23-30> Nesta linha é checado se o caractere do vetor é “-“ se sim, devera percorrer a direita da raiz, e logo após e comparado para ver se os códigos são iguais, se sim , devera printar a letra e sair do laço.

Função traduz_palavra:

-Esta função deve receber a “string” completa e separar palavra por palavra, para depois separar as letras e passar para o alfabeto usual.

-A função deve receber um ponteiro para “raiz” da arvore e uma “string”.

```
void traduz_Palavras(No *raiz,char *string) {
    char *token;

    // Usando a função strtok() para separar as palavras
    token = strtok(string, " ");

    while (token != NULL) {

        traduz_letra(raiz,token);
        token = strtok(NULL, " ");
    }
}
```

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

<1> É declarado o “token” que sera dividido as sub-strings para serem printadas letras por letras.

<4>Nesta linha é chamada a função “strtok” para token, que divide as em sub-rstrings que são as palavras ou letras e sinais.

<6-10>Este loop chega ate o tooken ser nulo, isso quer dizer que todas as sub-strings acabaram e deve acabar o loop.

<8> É chamado o traduz letra para passar para letras usuais, letra por letra.

Função traduz_pra_morse

-Esta função recebe um ponteiro pra string da estrutura letra e um caractere.

-Ela é responsável por pegar um caractere em alfanumérica e passar para morse.

```
void traduz_pra_morse(letra *vetor,char x){
    for(int i = 0;i < 39;i++){
        if(vetor[i].caractere == x ){
            printf("%s ",vetor[i].codigo);
            break;
        }
    }
}
```

<1-6>Este “for” é responsável por percorrer todo vetor que possou os caracteres e códigos morse , para achar a chave desejada e imprimir o codigo.

<2-5> É declarado um “if“ para quando encontrar o item desejado ,printar o código e sair do laço.

Função main:

1-Primeiramente é declarado um vetor da estrutura letra de 39 itens para guardar todos os caracteres e códigos morse necessário.

```
//criação do array pra pesquisa
    letra caracteres[39];
    caracteres[0].caractere ='e';
    strcpy(caracteres[0].codigo ,".");      //strcpy(A->codigo, ".-");
```

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

```
    caracteres[1].caractere ='i';
    strcpy(caracteres[1].codigo ,"..");

    caracteres[2].caractere ='s';
    strcpy(caracteres[2].codigo ,"...");

    caracteres[3].caractere ='h';
    strcpy(caracteres[3].codigo ,"....");

    caracteres[4].caractere ='5';
    strcpy(caracteres[4].codigo ,".....");
```

2-Nesta etapa é declara e aberto os arquivos “.txt” para serem utilizados depois

```
//-----declarar o primeiro arquivo----------------
    FILE *arquivo;
    arquivo = fopen("morse.txt","r");
    if(arquivo == NULL){
        printf("arquivo não aberto\n");
    }
    //---------------------------

    //-------declarar o segundo arquivo-------
    FILE *arquivo2;
    arquivo2= fopen("alpha.txt","r");
    if(arquivo2== NULL){
        printf("arquivo não aberto \n");
    }
    //------------------------------
```

3-É criada a arvore e um controlador para o usuário decidir o que quer fazer.

```
No *raiz=cria_arvore();
    int controlador ;
```

4- É declarado um laço para fazer a interface e fazer as ordens que forem escolhidas.

As opções de comando são:

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

0-Para acabar o programa.

1-Para traduzir de morse para português.

2-Para printar a arvore por completo.

3-Para ler um “txt” e traduzir para português.

4- Para escrever em português e passar parar morse.

5-Para ler um “txt” em português e passar para morse.

```
while(1){
        printf("_____________________________________________");
        printf("\n");
        printf("OPERACOES :\n");
        printf("\n");
        printf(" 0 - para acabar o programa\n");
        printf(" 1 - para traduzir de morse para portugues \n");
        printf(" 2 - Para printar a arvore por completo \n");
        printf(" 3 - Para ler do arquivo txt em morse e passar para
portugues \n");
        printf(" 4 - Para escrever em portugues e passar para
morse  \n");
        printf(" 5 - Para ler o arquivo em portugues e  passado para
morse  \n");
        printf("_____________________________________________");
        printf("\n");
        scanf("%d",&controlador);
        setbuf(stdin, NULL);
```

<16-17>É lido qual comando sera executado , e logo depois é limpado o buferr.

5-Onde é necessário ler ou escrever string ,como nas linahs <1,12,16,33> é declarado uma string.

-Onde precisa-se dos arquivos com nas linhas <14,38> é lido e passado para as strings assim printando quando chamado ou passado caractere por caractere para tradução.

```
if(controlador == 1){
            char s[1000];
            printf("escreva a frase em codigo morse \n");
            printf("coloque um espaço antes de terminar a mensagem\n");
            fgets(s,1000,stdin);
```

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

```
            printf("\n");
            traduz_Palavras(raiz,s);
            printf("\n");
        }
        if(controlador ==2 ){
            print_arvore(raiz);
        }
        if(controlador == 3){
            char str[1000];
            fgets(str,1000,arquivo);
            traduz_Palavras(raiz,str);
            printf("\n");

        }
        if(controlador == 4){

            char str[500];
            printf("escreva a frase em portugues \n");
            fgets(str,500,stdin);
            printf("\n");
            int i = 0;

            while(str[i]){

                traduz_pra_morse(caracteres,str[i]);

                i++;
            }
            printf("\n");

        }
        if(controlador == 5){
            printf("\n");
            char str[1000];
            fgets(str,1000,arquivo2);

            printf("frase escrita:  %s",str);
            printf("\n");

            int i;
            printf("em morse:  ");
            while(str[i]){

                traduz_pra_morse(caracteres,str[i]);

                i++;
            }
            printf("\n");
```

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7

```
        }
        if(controlador == 0) break;
```

Bibliografia:

-Foram usadas apenas os slides disponibilizados pelo professor.

Conclusão:

- As dificuldades foram resolvidas com pesquisa nos slides ou apenas relações logicas melhores, não teve muitas dificuldades.

- Trabalho divertido de se fazer, que desenvolveu aprendizado por meio de resolução de tarefas.

< > = corresponde a qual linha esta o item sendo descrito .
Exemplo <4-7> = linha 4 a linha 7