Instituto Federal de Educação, Ciência e Tecnologia Catarinense - Campus Videira
Bacharelado em Ciência da Computação - 1ª fase
Disciplina: Algoritmos
Professor: Manasses Ribeiro
Aluno: Henrique Pieri de Lima

# RELATÓRIO DO PROJETO FINAL - CAÇA-PALAVRAS

O algoritmo serve para o usuário inserir um caça-palavras em forma de vetor e procurar palavras no mesmo, retornando a sua posição. É utilizado um registro *Roi* que armazena as posições: linha inicial, coluna inicial, linha final e coluna final. O vetor do caça-palavras se chama *grade* e possui um tamanho pré-definido, inserindo seus valores pelo procedimento *carregarCacaPalavra*, que também verifica se o vetor forma uma matriz quadrada (necessário para o funcionamento do algoritmo) através do cálculo da quantidade de linhas e colunas (raiz quadrada do tamanho do vetor).

Após inserido, o caça-palavras é exibido ao usuário em forma de matriz através do procedimento *printVetorEmMatriz*, utilizando a quantidade de linhas/colunas (tamanho da matriz) como limite nas estruturas de repetição *for* e imprimindo os caracteres da posição da grade calculada através da multiplicação do contador de linhas com o tamanho da matriz, somado com o contador de colunas.

Então, o usuário insere a palavra a ser pesquisada na string *palavra*, que também possui um tamanho pré-definido. O *Roi* é inicializado, zerando as posições e é chamada a função *localizadores*, que executa todas as 8 funções para localizar a palavra e retorna a posição que ela foi encontrada, fazendo isso através de *ifs* e verificando se a função executada retorna uma posição válida através da função *palavraEncontrada*, caso não seja encontrada, todas as posições retornadas serão 0.

As 8 funções para localizar a palavra de formas diferentes são:

- Horizontal:
  - Esquerda → Direita (*locEsqDir*)
  - Direita → Esquerda (*locDirEsq*)
- Vertical:
  - Cima → Baixo (*locCimBai*)
  - Baixo → Cima (*locBaiCim*)

- Diagonal:
  - Cima Esquerda → Baixo Direita (*locDiagCEBD*)
  - Cima Direita → Baixo Esquerda (*locDiagCDBE*)
  - Baixo Esquerda → Cima Direita (*locDiagBECD*)
  - Baixo Direita → Cima Esquerda (*locDiagBDCE*)

Elas funcionam de forma semelhante: contêm duas estruturas de repetição *for* (percorrendo as linhas e colunas) e um *while* que utiliza uma variável contadora *cont*. O método para encontrar a palavra é verificar se na posição da *grade* há o mesmo caractere da palavra procurada, caso seja verdadeiro, o *cont* é incrementado e muda a posição dos caracteres que devem ser procurados. Ou seja, caso o caractere da posição X da *grade* for igual ao caractere da posição 0 da *palavra*, o *cont* é incrementado, alterando a posição da *grade* para uma posição Y (que muda de acordo com a forma de pesquisa), e passa ao próximo caractere da palavra (no caso, a posição 1), fazendo isso sucessivamente até encontrar a palavra completa. Quando encontrá-la, o valor do contador será igual ao tamanho da palavra, e então retornará a posição da mesma. No *while* também temos outros parâmetros que verificam se a palavra está dentro dos limites da matriz, podendo diferir de acordo com a forma de pesquisa.

**Exemplo com a função *locEsqDir*:**

```
Roi locEsqDir(Roi posicao, char grade[], char palavra[], int tamM, int tamP) {
    int l, c, cont;

    for (l=0; l<tamM; l++) {
        for (c=0; c<tamM; c++) {
            cont = 0;
            while ((grade[l*tamM+c+cont] == palavra[cont]) && (cont<tamM) && ((c+cont) < tamM)) {
                cont++;
                if (cont == tamP) {
                    posicao.li = l;
                    posicao.ci = c;
                    posicao.lf = l;
                    posicao.cf = c + tamP - 1;

                    return posicao;
                }
            }
        }
    }
    return posicao;
}
```

Em um caça-palavras de vetor com tamanho 100 (o tamanho da matriz sendo de 10 linhas/colunas), a palavra “PAO” está na posição: linha inicial 6, coluna inicial 1, linha final 6, coluna final 3. Então os dois *for* percorrerão toda a grade até chegar na posição inicial da palavra, no *while* ele verifica:

| grade[l*tamG+c+cont] == palavra[cont] | cont < tamG | (c+cont) < tamG |
|---|---|---|
| grade[6*10+1+0] == palavra[0] | 0 < 10 | 1 + 0 < 10 |
| grade[61] == palavra[0] | ✔ | 1 < 10 |
| “P” == “P” | | ✔ |
| ✔ | | |

Então o *cont* é incrementado e repete novamente a estrutura:

| grade[l*tamG+c+cont] == palavra[cont] | cont < tamG | (c+cont) < tamG |
|---|---|---|
| grade[6*10+1+1] == palavra[1] | 1 < 10 | 1 + 1 < 10 |
| grade[62] == palavra[1] | ✔ | 2 < 10 |
| “A” == “A” | | ✔ |
| ✔ | | |

Assim sucessivamente até completar a palavra (caso esteja correto), com o *cont* se igualando ao tamanho da palavra e retornando a posição da mesma para a função *localizaPalavra*, exibindo-a ao usuário. Após pesquisar a palavra desejada, o usuário pode sair da procura de palavras digitando 0 e escolher se quer inserir um novo caça-palavras ou sair do algoritmo.