

UNIVERSIDADE FEDERAL UBERLÂNDIA
CAMPUS MONTE CARMELO
CURSO BACHARELADO EM SISTEMAS DE
INFORMAÇÃO

RHUAN FLORES CUNHA
FERNANDES CLÉSIO RODRIGUES
DA SILVA JÚNIOR

**2º TRABALHO DE ESTRUTURA DE
DADOS II**

Monte
Carmelo-MG
04/2023

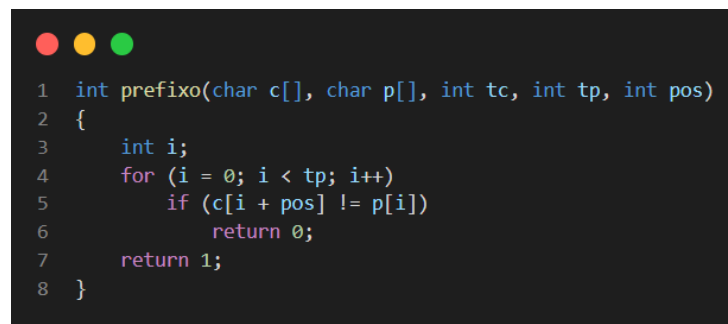
SUMÁRIO

1	DEFINIÇÃO DE ALGORITMO BOYLER-MOORE.....	3
2	RESOLUÇÃO DO PROBLEMA.....	4

1 DEFINIÇÃO DE ALGORITMO BOYLER-MOORE

O algoritmo de Boyler-Moore é utilizado para encontrar as ocorrências de uma palavra em uma string/texto. Dito isso, o algoritmo é dividido em duas funções, uma básica para toda busca em texto chamanda prefixo, onde ela recebe uma palavra ou cadeia de caracteres e a string onde a ela será procurada. Assim, ao passar o tamanho da palavra e o local aonde ela deve conferir, ela retorna 1 se a palavra coincidir com a posição, ou 0 se contrário.

Imagem I – Função Prefixo



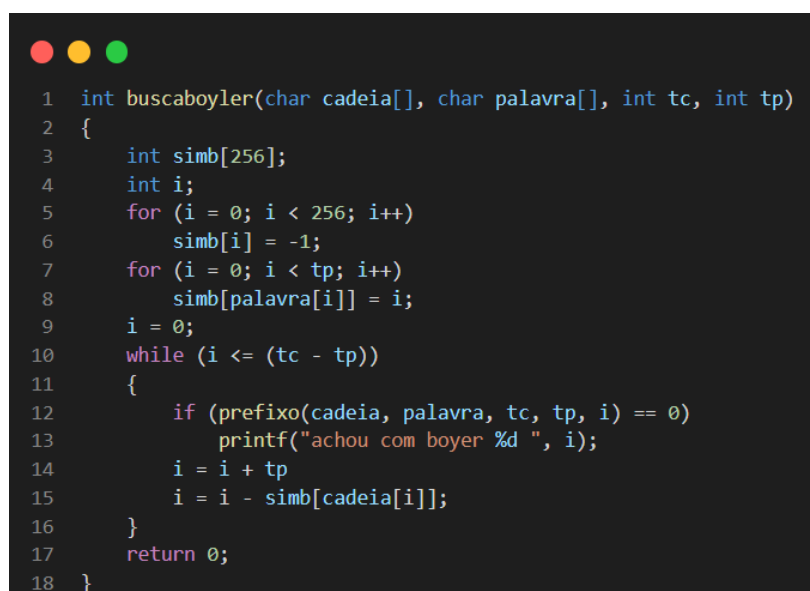
```

1  int prefixo(char c[], char p[], int tc, int tp, int pos)
2  {
3      int i;
4      for (i = 0; i < tp; i++)
5          if (c[i + pos] != p[i])
6              return 0;
7      return 1;
8  }

```

A segunda parte do algoritmo é o que separa ele dos de mais, ele acelera o método comum de busca em string, utilizando informações no pré-processamento para pular seções do texto onde não existe a palavra procurada. Resultando em um desempenho maior que os outros algoritmos conforme o tamanho da cadeia é aumentada.

Imagem II – Função Busca Boyler-Moore



```

1  int buscaboyler(char cadeia[], char palavra[], int tc, int tp)
2  {
3      int simb[256];
4      int i;
5      for (i = 0; i < 256; i++)
6          simb[i] = -1;
7      for (i = 0; i < tp; i++)
8          simb[palavra[i]] = i;
9      i = 0;
10     while (i <= (tc - tp))
11     {
12         if (prefixo(cadeia, palavra, tc, tp, i) == 0)
13             printf("achou com boyer %d ", i);
14         i = i + tp;
15         i = i - simb[cadeia[i]];
16     }
17     return 0;
18 }

```

2 RESOLUÇÃO DO PROBLEMA

O problema consistia em ler algum arquivo de texto especificado pelo usuário, trocar uma palavra por outra (também especificadas pelo usuário) e por fim escrever o texto criado em outro arquivo com o nome que o usuário quiser. Enfim, na primeira tentativa foi utilizado a linguagem C, entretanto ela não possuía alguns métodos e tipos, como o tipo String, que em C é definido por um vetor de Char[]. Assim, como o problema era de um texto onde não é de conhecimento o tamanho da string que ele irá virar, é aconselhável fazer uma TAD que possui dois atributos, um char e um ponteiro para outro TAD desse mesmo tipo, assim fazendo uma lista, com métodos de inserção, remoção e contagem de caracteres. Entretanto, o professor permitiu o uso de C++ para a utilização da classe String, que faz esse trabalho automaticamente.

Dito isso, foi utilizado para substituir essa TAD a classe String no C++ e criada também para auxiliar no código a TAD “noptr”, que possui um int e um ponteiro para outro tipo “noptr”, com métodos de inserção e remoção.

Imagem III – TAD “noptr”

```

1 typedef struct no
2 {
3     int dados;
4     struct no *prox;
5 }noptr;

```

```

1 void insereLista(int valor){
2     noptr *novo;
3     novo = (struct no *) malloc (sizeof(noptr));
4     if(!novo){
5         printf("\nSem Memoria!!!!\n");
6         return;
7     }
8     novo->dados = valor;
9     if(inicio == NULL){
10        novo->prox = NULL;
11    }else{
12        novo->prox = inicio;
13    }
14    inicio = novo;
15 }
16
17 int removelist(){
18     noptr *p;
19     int valor;
20     if(inicio == NULL){
21         return -1;
22     }else{
23         p = inicio;
24         valor = p->dados;
25         inicio = p->prox;
26         free(p);
27     }
28     return valor;
29 }

```

Com as TADs montadas e o algoritmo Boyler-Moore implementado, começamos o desenvolvimento. A solução consiste em instanciar 2 arquivos, abrir o primeiro arquivo, com nome especificado na entrada do usuário, no modo de leitura e ler caractere

por caractere, inserindo eles em uma string. Com a string montada igual ao arquivo recebemos as duas palavras, uma para achar e uma para trocar no texto, e colocamos o algoritmo em ação até que a primeira palavra lida seja vazia. Sempre que a função prefixo retornar 1, o valor da posição da palavra na cadeia é armazenada na TAD “noptr”, assim, após a leitura da cadeia por completo, temos todas as posições da palavra encontrada.

Após isso começamos a escrita do novo arquivo, manipulando a string cópia do arquivo, alteramos a palavra achada pela trocada e após a TAD ficar vazia, começamos a escrita no arquivo. Pedimos o nome do arquivo de salvamento para o usuário e começamos a escrever nele, com a função `putc`, cada caractere da string, no final teremos o arquivo desejado.

Vale ressaltar que todos os arquivos utilizados ficam salvos em “.../output”, existem funções para conferir se o arquivo aberto para leitura é existente e limpar o buffer de entrada das strings (`cin.sync()`). Além disso, as strings são arrumadas por uma função própria, chamada `replace()`, onde é passada como argumentos a posição que deseja ser mudada, o tamanho até onde será retirado e a string que entrará no lugar. Confira o código abaixo.

Imagem IV – Função Principal

```
1  int main(){
2      FILE *arq1, *arq2;
3      string aux, s;
4
5      cout << "Digite o nome do Arquivo (lembre-se do .txt)" << endl;
6      cin >> aux;
7      arq1 = fopen(aux.data(), "r");
8      if(arq1 == NULL){
9          cout << "Arquivo Inexistente" << endl;
10         return 0;
11     }
12
13     int i = 0;
14     char c;
15     while((c=fgetc(arq1)) != EOF){
16         s+=c;
17         i++;
18     }
19     cout << s << endl;
20
21     string palavraTrocar;
22     string palavraAchar;
23     cout << "Digite a palavra a ser encontrada" << endl;
24     cin.sync();
25     getline(cin, palavraAchar);
26     while(!palavraAchar.empty()){
27         cout << "Digite a palavra para colocar no lugar" << endl;
28         cin >> palavraTrocar;
29         cin.sync();
30
31         buscaboyler(s,palavraAchar,s.length(),palavraAchar.length());
32
33         int num = removelistas();
34         while(num != -1){
35             s.replace(num, palavraAchar.length(), palavraTrocar);
36             num = removelistas();
37         }
38
39         cout << s << endl;
40         cout << "Digite a palavra a ser encontrada" << endl;
41         getline(cin, palavraAchar);
42     }
43
44     cout << "Digite o nome do Arquivo a ser Criado" << endl;
45     cin >> aux;
46     arq2 = fopen(aux.data(), "w");
47     for(i = 0; i<s.length();i++){
48         putc(s[i],arq2);
49     }
50 }
```