Centro Federal de Educação Tecnológica de Minas Gerais

Turma: Engenharia da Computação **Prof.:** Bruno

Nome: Rúbia Alice Moreira de Souza

Trabalho Prático 4 - Ponteiros e Alocação Dinâmica

Belo Horizonte

2022

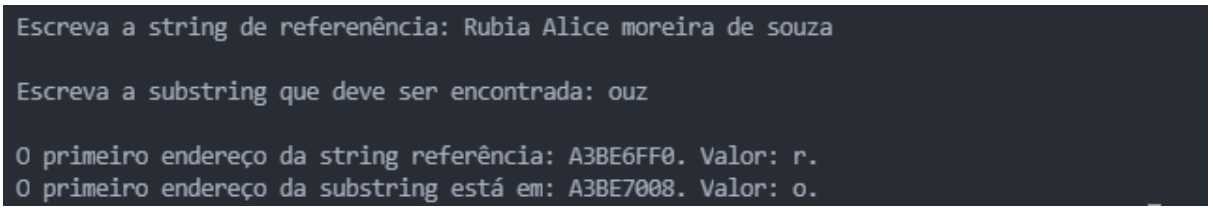
Sumário

1. Introdução

Este relatório explica a execução do programa do Trabalho Prático 4 do Curso de Engenharia da computação do CEFET-MG e o código fonte do projeto está disponível no meu github: Rubia-Souza, mas apenas será tornado público após a data de entrega do trabalho.

2. Questão 1

2.1. Valor de substring encontrado

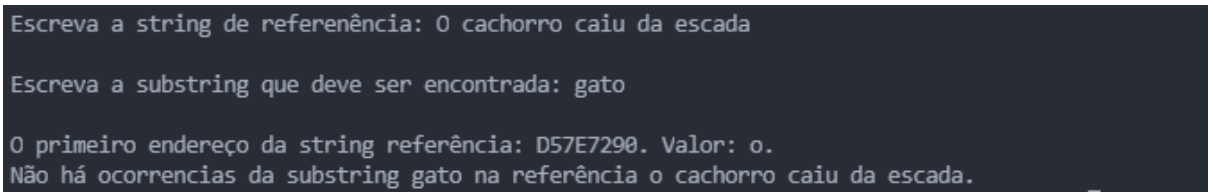


Inicialmente, o programa pede ao usuário que informe qual a string em que será realizada a busca pela substring e depois o padrão de substring que deve ser verificado.

Após isso, ele informa o endereço em hexadecimal e o valor da primeira posição do vetor de referência.

Caso encontre o valor da substring, ele informa o endereço de onde está a primeira letra do padrão na string de referência e o conteúdo do ponteiro.

2.2. Valor de substring não encontrado



Quando o valor não é encontrado, o sistema trata o retorno nulo da função informando ao usuário que a substring não está na string de referência fornecida.

3. Questão 2

Nesta questão, um vetor de unsigned int é alocado de acordo com o tamanho informado no início da execução pelo usuário.

A partir disso, esse vetor é preenchido com valores inteiros entre 1 e 1000 de forma aleatória, mas garantindo que não há nenhuma repetição de números. Esses valores são exibidos a seguir:

```
Digite o tamanho do vetor de inteiros positivos: 20


---- Vetor de valores:
[0] Valor: 884
[1] Valor: 330
[2] Valor: 405
[3] Valor: 140
[4] Valor: 534
[5] Valor: 784
[6] Valor: 728
[7] Valor: 850
[8] Valor: 361
[9] Valor: 227
[10] Valor: 574
[11] Valor: 705
[12] Valor: 529
[13] Valor: 604
[14] Valor: 955
[15] Valor: 493
[16] Valor: 172
[17] Valor: 740
[18] Valor: 439
[19] Valor: 813
```

Depois, um novo print é feito sem alterar o conteúdo ou ordem do vetor para informar quais os valores e a sua respectiva posição de memória em hexadecimal.

```
---- Vetor de endereços:
[0] Endereço: 948B1740 - Valor: 884.
[1] Endereço: 948B1744 - Valor: 330.
[2] Endereço: 948B1748 - Valor: 405.
[3] Endereço: 948B174C - Valor: 140.
[4] Endereço: 948B1750 - Valor: 534.
[5] Endereço: 948B1754 - Valor: 784.
[6] Endereço: 948B1758 - Valor: 728.
[7] Endereço: 948B175C - Valor: 850.
[8] Endereço: 948B1760 - Valor: 361.
[9] Endereço: 948B1764 - Valor: 227.
[10] Endereço: 948B1768 - Valor: 574.
[11] Endereço: 948B176C - Valor: 705.
[12] Endereço: 948B1770 - Valor: 529.
[13] Endereço: 948B1774 - Valor: 604.
[14] Endereço: 948B1778 - Valor: 955.
[15] Endereço: 948B177C - Valor: 493.
[16] Endereço: 948B1780 - Valor: 172.
[17] Endereço: 948B1784 - Valor: 740.
[18] Endereço: 948B1788 - Valor: 439.
[19] Endereço: 948B178C - Valor: 813.
```

Então, é criado um ponteiro de ponteiros para fazer a ordenação dos endereços sem trocar o conteúdo entre eles. A ordenação é feita segundo o algoritmo de insertion sort e tem o seguinte resultado:

```
---- Vetor de endereços:
[0] Endereço: 948B1778 - Valor: 955.
[1] Endereço: 948B1740 - Valor: 884.
[2] Endereço: 948B175C - Valor: 850.
[3] Endereço: 948B178C - Valor: 813.
[4] Endereço: 948B1754 - Valor: 784.
[5] Endereço: 948B1784 - Valor: 740.
[6] Endereço: 948B1758 - Valor: 728.
[7] Endereço: 948B176C - Valor: 705.
[8] Endereço: 948B1774 - Valor: 604.
[9] Endereço: 948B1768 - Valor: 574.
[10] Endereço: 948B1750 - Valor: 534.
[11] Endereço: 948B1770 - Valor: 529.
[12] Endereço: 948B177C - Valor: 493.
[13] Endereço: 948B1788 - Valor: 439.
[14] Endereço: 948B1748 - Valor: 405.
[15] Endereço: 948B1760 - Valor: 361.
[16] Endereço: 948B1744 - Valor: 330.
[17] Endereço: 948B1764 - Valor: 227.
[18] Endereço: 948B1780 - Valor: 172.
[19] Endereço: 948B174C - Valor: 140.
```

4. Questão 3

No início da execução de programa 3, são instanciados 2 arrays de uma estrutura chamada vetor. A struct foi criada a fim de conseguir manter um controle mais fácil sobre o tamanho dos arrays, uma vez que não é possível obter o tamanho de um array criado dinamicamente em C sem ter definido o seu tamanho máximo anteriormente. Depois, ambos são preenchidos com valores inteiros positivos aleatórios entre 1 e 50.

```
---- Valores vetor Q:
[0] Valor: 22.
[1] Valor: 38.
[2] Valor: 43.
[3] Valor: 11.
[4] Valor: 50.
[5] Valor: 39.
[6] Valor: 46.
[7] Valor: 14.
[8] Valor: 28.
[9] Valor: 19.
[10] Valor: 16.
[11] Valor: 9.
```

```
---- Valores vetor R:
[0] Valor: 11.
[1] Valor: 21.
[2] Valor: 30.
[3] Valor: 21.
[4] Valor: 29.
[5] Valor: 36.
[6] Valor: 16.
[7] Valor: 48.
[8] Valor: 13.
[9] Valor: 31.
[10] Valor: 17.
[11] Valor: 20.
[12] Valor: 18.
[13] Valor: 23.
[14] Valor: 44.
[15] Valor: 4.
[16] Valor: 32.
[17] Valor: 46.
[18] Valor: 5.
[19] Valor: 7.
```

Então, é chamada a função interception() para buscar os valores comuns entre os dois vetores. Nela, um novo vetor é criado e o valor do seu array interno é redimensionado a cada novo número inserido.

Com base nos vetores Q e R anteriores, o resultado para o vetor W foi o seguinte:

```
---- Valores vetor W:
[0] Valor: 11.
[1] Valor: 46.
[2] Valor: 16.
```