# Breve introdução

Este documento tem como objetivo abordar sobre e mostrar o funcionamento do projeto desenvolvido na disciplina de Algoritmo e lógica de programação.

O projeto tem como finalidade resolver circuitos lógicos em formato de arquivo, para isso o circuito deverá ser adaptado seguindo as regras do programa, pois só assim, o circuito será resolvido corretamente

O programa consiste em ler um arquivo de texto informado pelo usuário na tela de execução, criar as portas lógicas e ler suas entradas e saída, após isso, é gerada a tabela-verdade e para cada caso, o programa mostrará a saída equivalente.

# Como baixar o programa:

Para baixar o programa, primeiramente o usuário deverá possuir um compilador capaz de executar códigos em C, de preferência o Dev C++. Em seguida, deverá entrar no GitHub pelo link: <https://github.com/mvfreitas99/projeto> e baixa-lo.

# Exemplo de circuito lógico:

# 

Figura 1: Circuito lógico

Como podemos ver, é um circuito todo numerado, onde cada porta tem suas entradas e saídas numeradas, em alguns casos, a saída de uma porta é conectada na entrada de outra porta, então podemos dizer “A saída 4 da porta not é uma das entradas da porta XOR”, com isso, o arquivo de texto deve estar no seguinte formato:

# Como elaborar o arquivo de texto

1 – CIRCUIT NOME\_DO\_CIRCUITO

2 – PORTA1 ENTRADA1 ENTRADA2 SAIDA1

3 – PORTA2 SAIDA1 ENTRADA4 SAIDA2

4 – PORTA3 ENTRADA4 ENTRADA5 SAIDA3

5 – INPUT ENTRADA1 NOME\_DA\_ENTRADA

6 – INPUT ENTRADA2 NOME\_DA\_ENTRADA

7 – OUTPUT SAIDAFINAL NOME\_DA\_SAIDA

O mais importante do arquivo de texto é que não seja criado de outra forma, pois poderá ser lido de outra forma e compilar erroneamente, o programa diferencia maiúsculas de minúsculas, para ficar mais fácil de entender, segue o código do arquivo de texto para o circuito da figura 1:

CIRCUIT Expressao logica

NOT 1 4

AND 2 3 8

NOR 4 8 7

AND 1 7 5

AND 2 5 6

INPUT 1 A

INPUT 2 B

INPUT 3 C

OUTPUT 6 S

Quando o programa for executado com esse arquivo, deverá mostrar a seguinte tabela verdade:

Entradas | Saídas

|  |
| --- |
| A | B | C | S |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |

Caso seja executado esse arquivo e aparecer essa tabela, significa que o arquivo de texto está correto e o programa funcionará corretamente.

# Executando o programa:

Quando o arquivo de texto for criado e o programa já estiver no compilador, é hora de executá-lo, para isso, o usuário deverá iniciar a compilação e quando o programa solicitar o diretório de arquivo, deverá ser informada corretamente, segue um exemplo na figura 2:

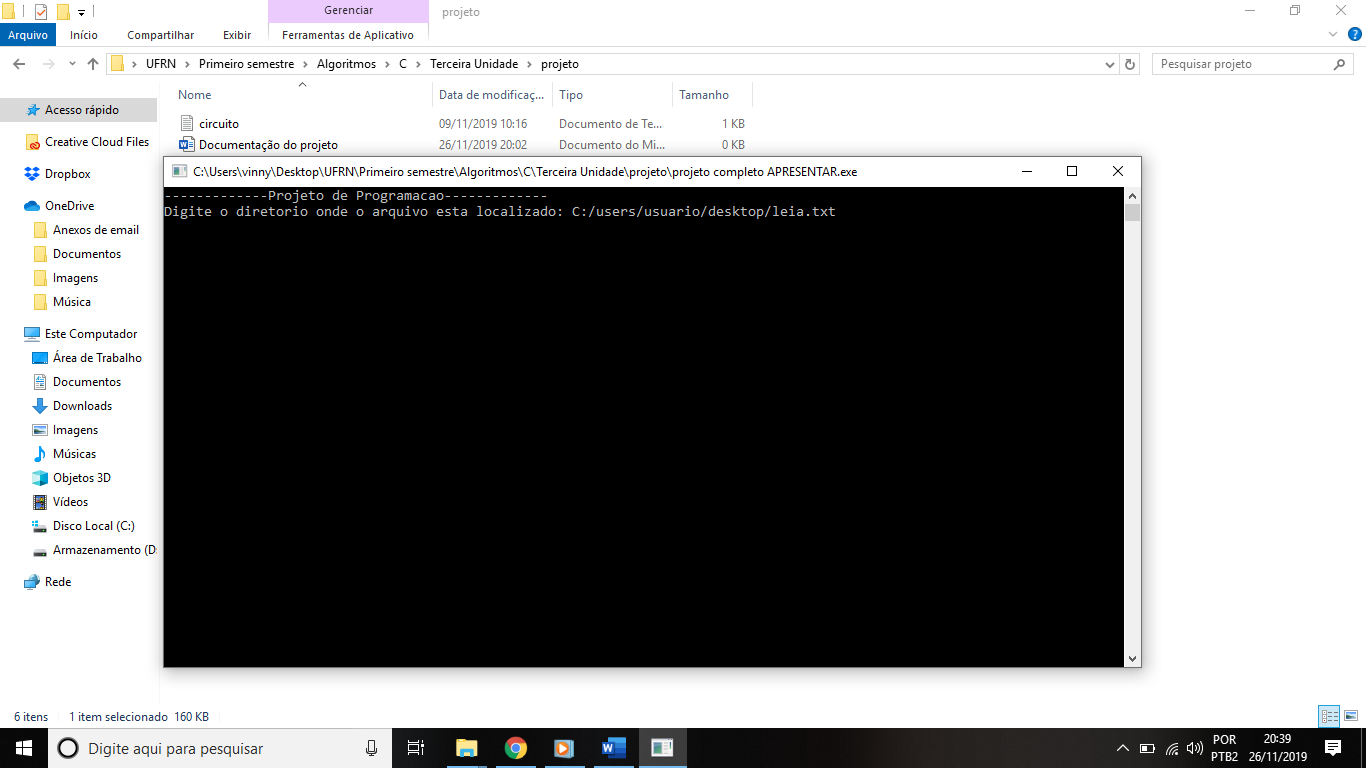


Figura 2: Colocando o diretório do arquivo de texto

Caso o diretório esteja correto, o programa informará a tabela-verdade correspondente ao circuito. Caso algo esteja errado, o programa informará uma mensagem de erro correspondente ao erro cometido pelo usuário.

# Programa funcionando corretamente:

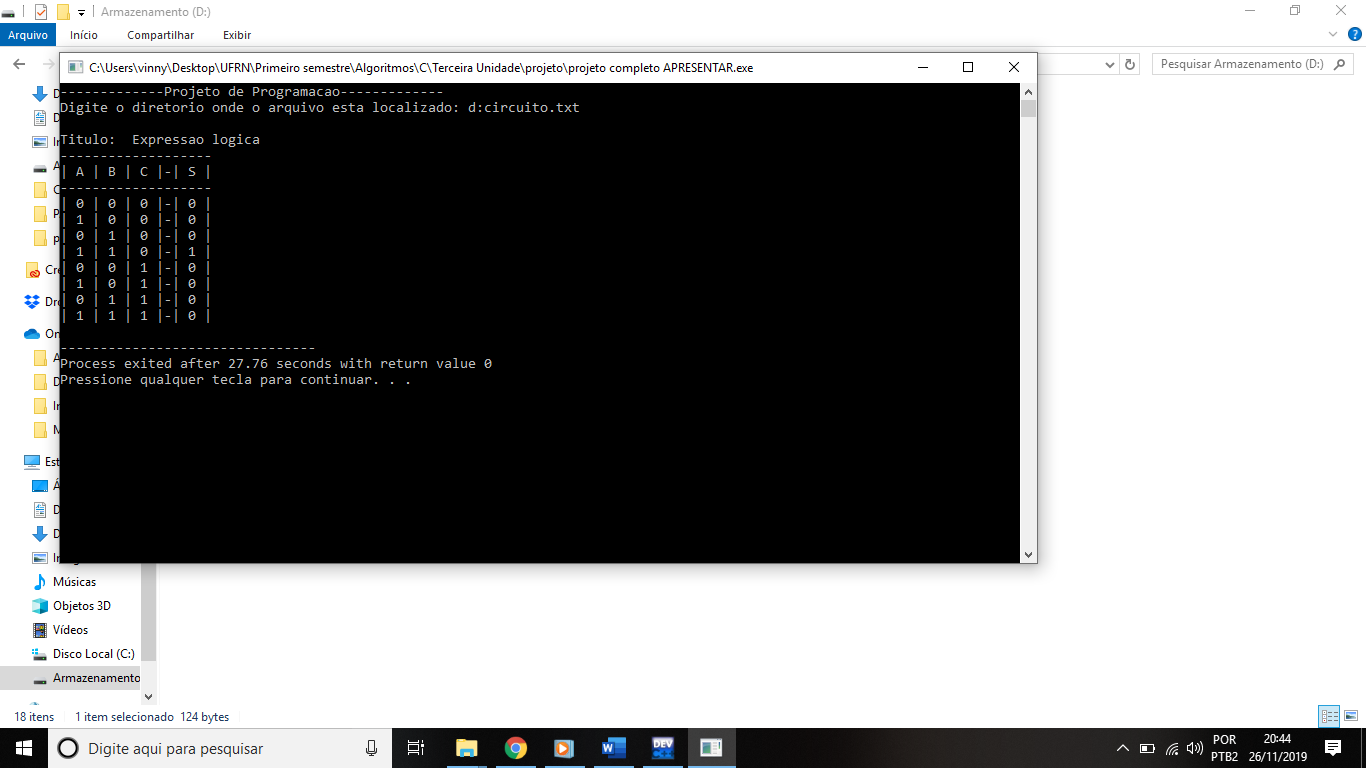


Figura 3: Programa funcionando corretamente

Como podemos ver, para este caso, o programa funcionou corretamente, nota-se que a saída está diferente da ordem teórica, mas o resultado é o mesmo.

# Programa com erros:

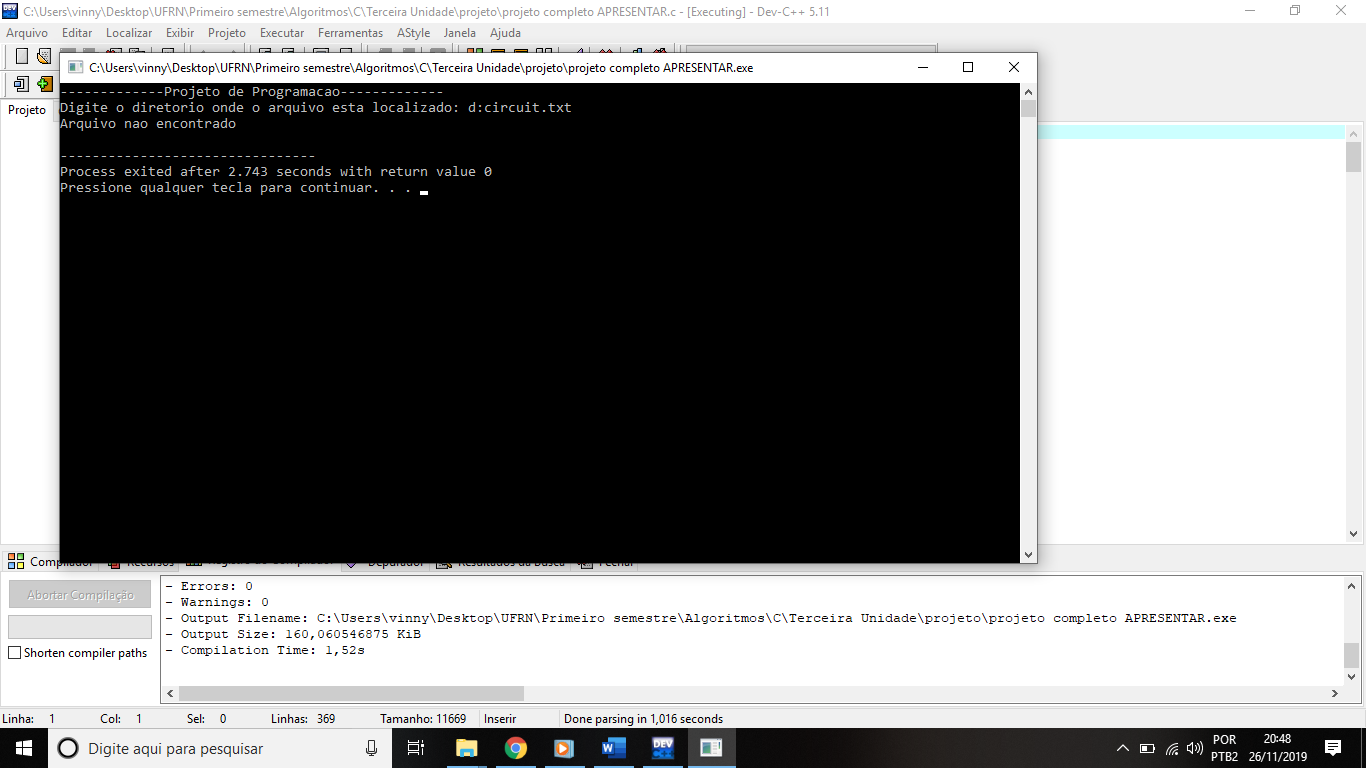


Figura 4: arquivo não encontrado

Como podemos ver o diretório foi informado errado propositalmente a fim de testar a eficiência do programa, e como esperado, retornou a mensagem de que o arquivo circuito não foi encontrado.