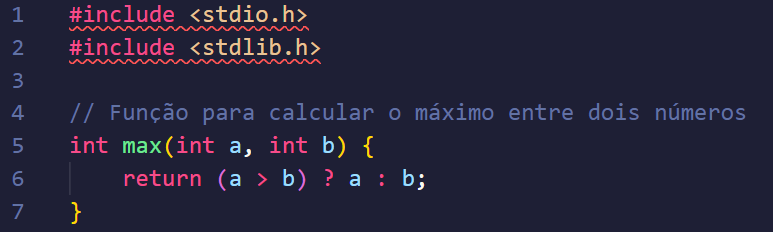
# Teste para desenvolvimento de SW

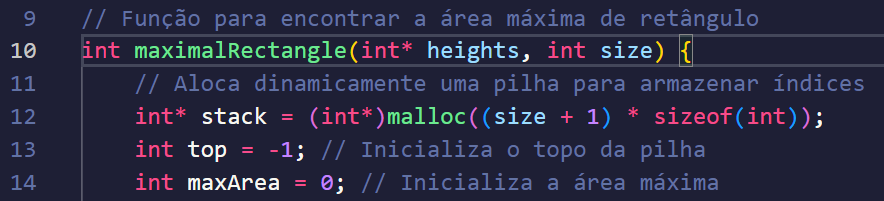
A abordagem utilizada é uma técnica de programação dinâmica para calcular a altura máxima de barras consecutivas em cada linha da matriz. Em seguida, aplica-se o algoritmo de maior retângulo a cada linha para obter a área máxima.

Solução para o problema 1:

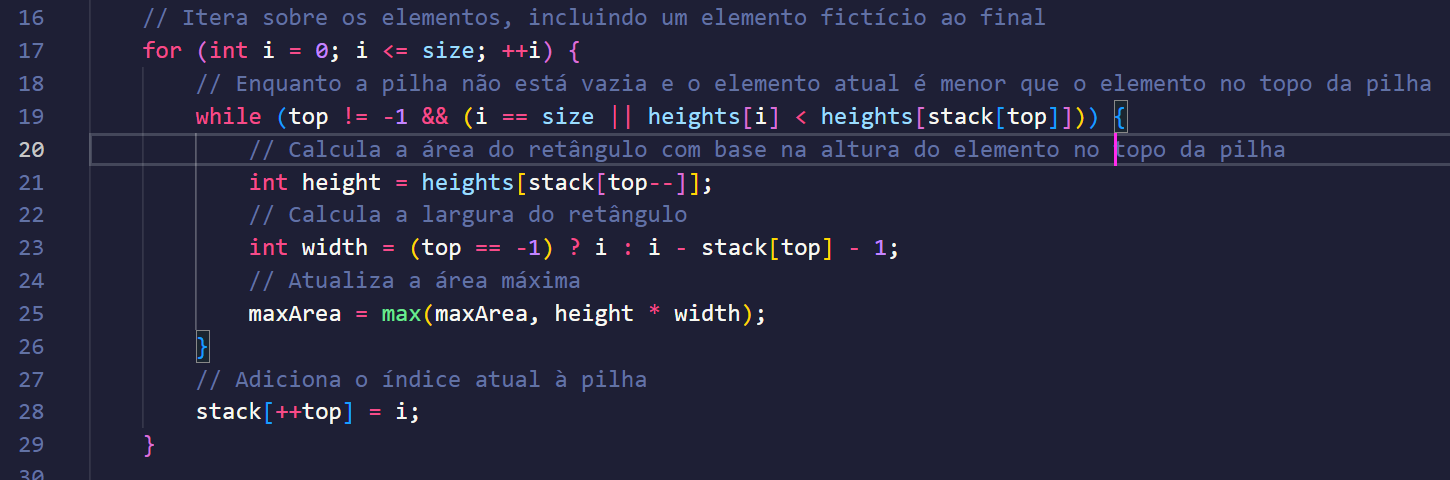
1. O código para o primeiro problema está abaixo:



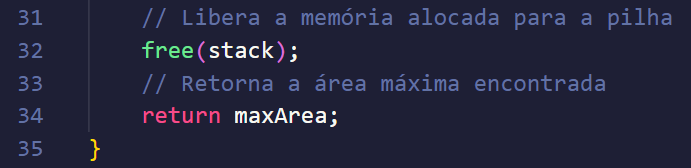
1. Primeiramente são chamadas as bibliotecas que serão utilizadas no programa, em seguida é definida uma função de nome “*max*” para calcular o máximo entre dois números.



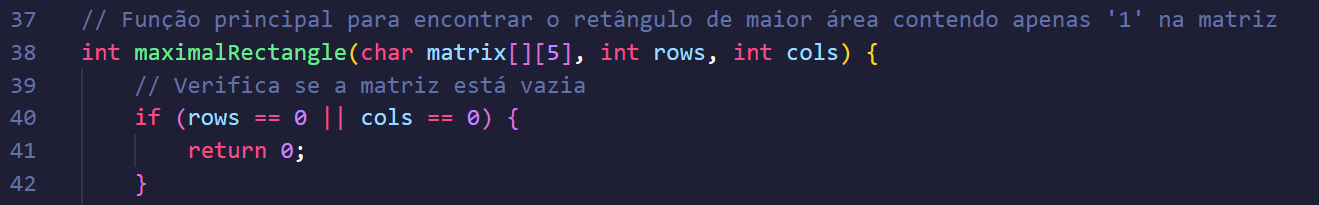
1. Em seguida é criada outra função de nome “*maximalRectangle*” que serve para encontrar a área máxima do retângulo. Nesta função é utilizado o conceito de pilha para armazenar os índices dentro de uma *stack*.



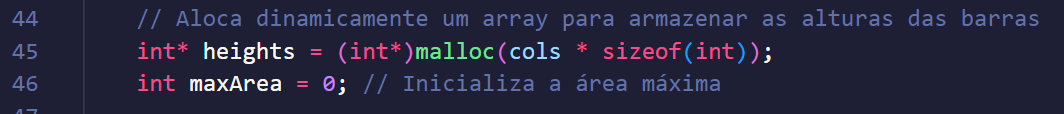
1. Após isso, cria-se um laço de repetição para iterar sobre os elementos, e é aplicada uma lógica para encontrar a maior área do retângulo dentro da matriz fornecida.



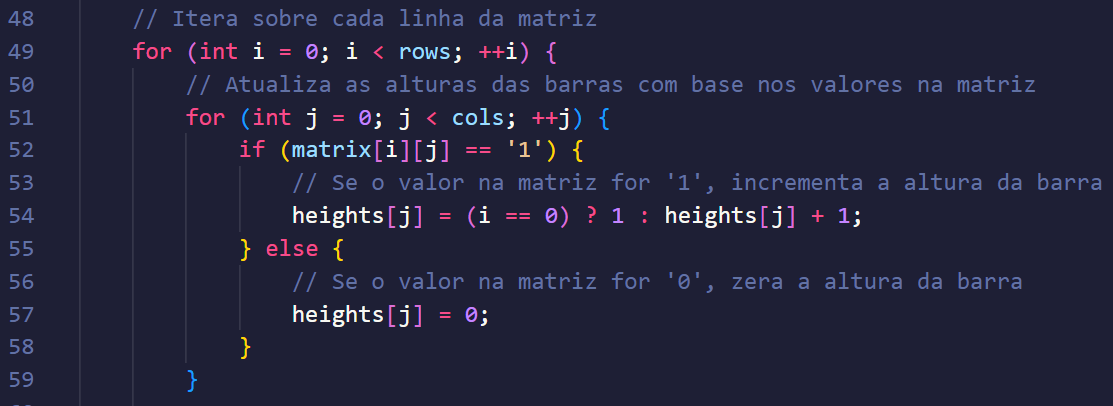
1. Após o fim do laço de repetição, a memória alocada na criação da *stack* é liberada e o maior valor de retângulo é retornado através do *return*.



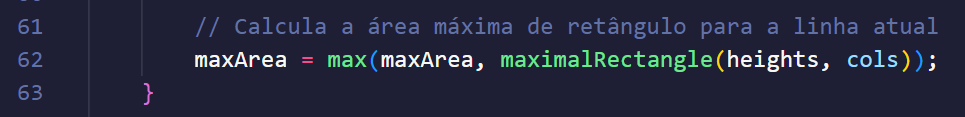
1. Cria-se então mais uma função para encontrar o retângulo de maior área que contém apenas o número ‘1’ na matriz. Logo após, aplica-se um filtro para verificar se a matriz é vazia antes de começar o algoritmo.



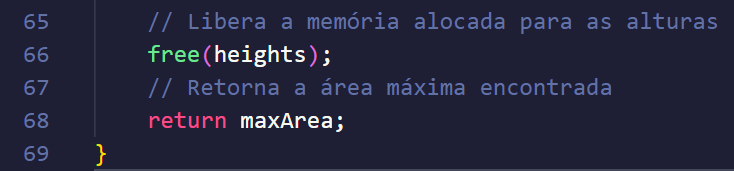
1. Aloca-se novamente na memória um espaço para armazenar o tamanho dos retângulos e é criada uma variável de controle para salvar o maior tamanho encontrado.



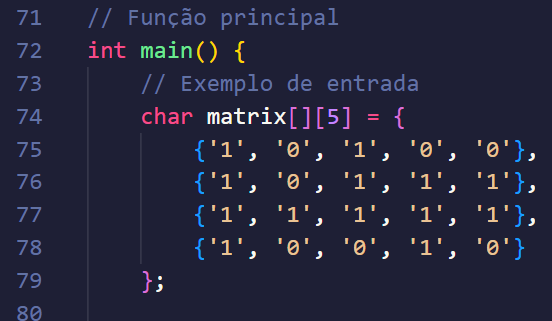
1. É criado um laço de repetição para iterar em cada linha da matriz e é aplicado uma lógica para salvar o retângulo de números ‘1’ de maior tamanho.



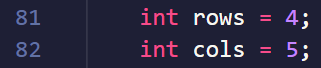
1. Após o final do laço, é chamada a função “*max*” criada na linha 5 para conseguir retornar o maior valor de área, utilizando como parâmetro a maior área de 1 encontrada e uma chamada de função para a que foi criada na linha 10.



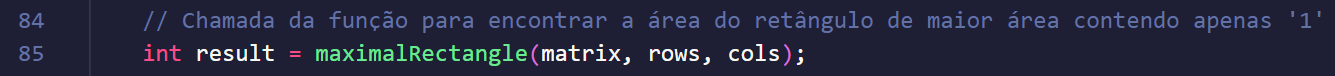
1. Após o final do cálculo de área, a memória alocada no começo da função é liberada e o valor total é retornado.



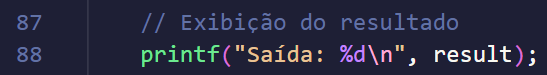
1. Um exemplo da implementação do *main* é como na imagem acima, onde é necessário criar uma matriz de valores para passar para a função.



1. São criadas 2 variáveis para salvar os valores de linhas e colunas.



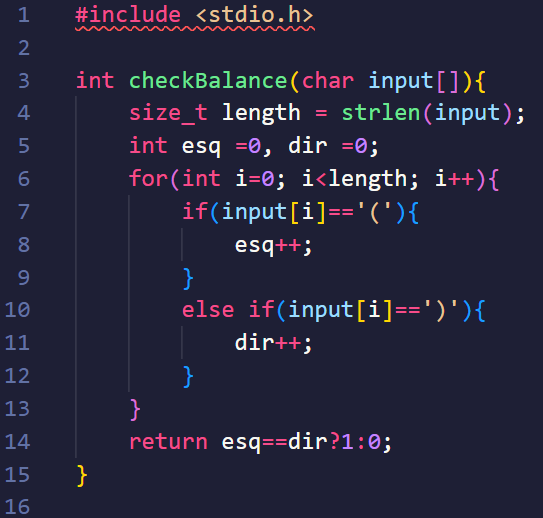
1. Após isso, elas são utilizadas na chamada função para calcular a maior área de retângulo.



1. Ao final é utilizada a função “*printf*” para imprimir a saída na tela e visualizar o resultado.

Solução para o problema 2:

1. O código para o segundo problema está abaixo:



1. Primeiro é declarado o nome da função, que nesse caso foi “*checkBalance*”.Em seguida, é criada uma variável para gravar o tamanho da *string* (texto) de entrada, que neste caso se chama “*length*”.

Abaixo são criadas 2 variáveis de controle do tipo inteiro (números) de nome *esq* e *dir*, representando o colchete esquerdo e direito respectivamente.

Foi criado também um laço de repetição para ler todas as posições possíveis da *string* e verifica-se se algum deles é um dos parênteses. Em caso positivo é adicionado 1 na variável de controle correspondente aos parênteses.

No “*return*” o programa verifica primeiramente se as quantidades de colchetes são iguais (esq==dir). Caso sejam, o algoritmo retorna o número 1 como resposta, caso contrário ele retornará o número 0.

O *main* do código pode ser escrito da forma abaixo para testar o código.

