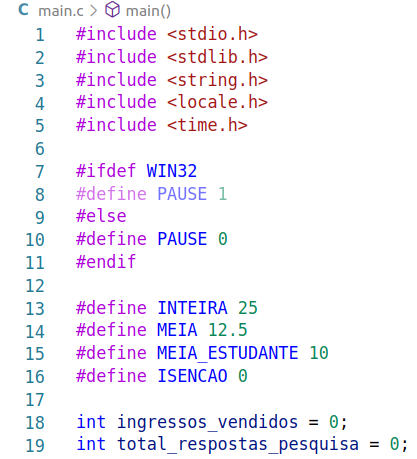
**1 DOCUMENTAÇÃO DO CÓDIGO DO SISTEMA DESENVOLVIDO**

Desenvolvemos um sistema na linguagem de programação C que será utilizado nos terminais via linha de comando. O sistema tem todas as funcionalidades que foram definidas nos requisitos funcionais anteriormente, todas as funcionalidades foram separadas em rotinas e sub-rotinas que organizam o código.

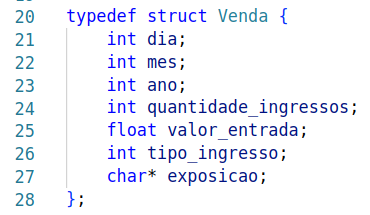
As funcionalidades disponíveis são: cadastro de venda de ingressos em arquivo .csv, consulta de preços dependendo do tipo de entrada, busca de ingresso pelo ID (número para identificação que é gerado pelo sistema), listagem das vendas de uma data específica, e geração de relatórios de vendas mensais em formato .md.

Segue a documentação do programa:

**1.1 Bibliotecas, constantes e structs**

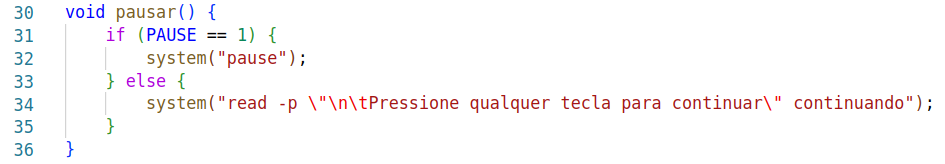


Foi necessário importar as bibliotecas de entrada e saída, a biblioteca padrão e a biblioteca relacionada a cadeias de caracteres, assim como a biblioteca responsável por adicionar caracteres especiais ao programa e a biblioteca relacionada a data e hora. A constante PAUSE se refere ao tipo de comando que será usado para pausar a execução do programa, visto que essa instrução é diferente dependendo do sistema operacional. Em seguida, temos a definição dos preços de cada tipo de entrada, assim como uma variável global que armazena a quantidade de ingressos vendidos e outra que armazena o total de respostas na pesquisa de satisfação.

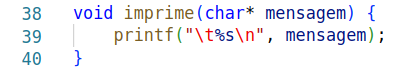


O struct chamado Venda representa uma venda a ser cadastrada no arquivo .csv, as informações necessárias são: data da venda (dia, mês e ano todos como inteiro), inteiro que representa a quantidade de ingressos do mesmo tipo a serem vendidos, o valor de cada entrada como variável real, um inteiro que representa o tipo dos ingressos e uma string para armazenar o nome da exposição.

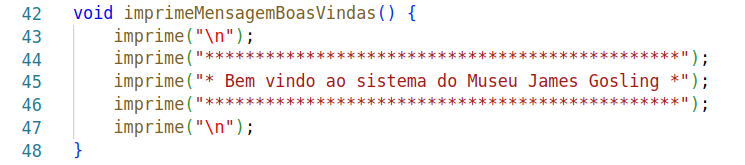
**1.2 Sub-rotinas auxiliares**



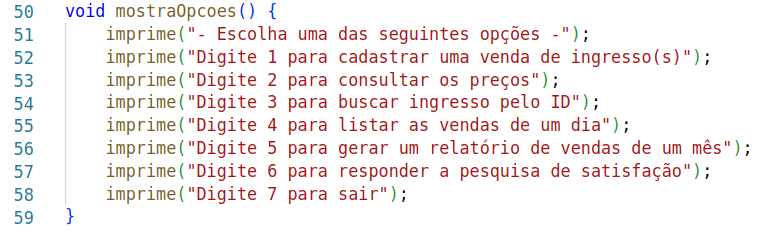
A função pausar realiza o comando de pause referente ao sistema operacional atual, a constante PAUSE foi criada no início do arquivo e tem valor 1 quando o sistema operacional é Windows e 0 quando o sistema operacional é Unix, chamando o comando correto do sistema operacional. Essa sub-rotina é usada para permitir que o usuário veja as informações mostradas no terminal antes de continuar a execução do programa.



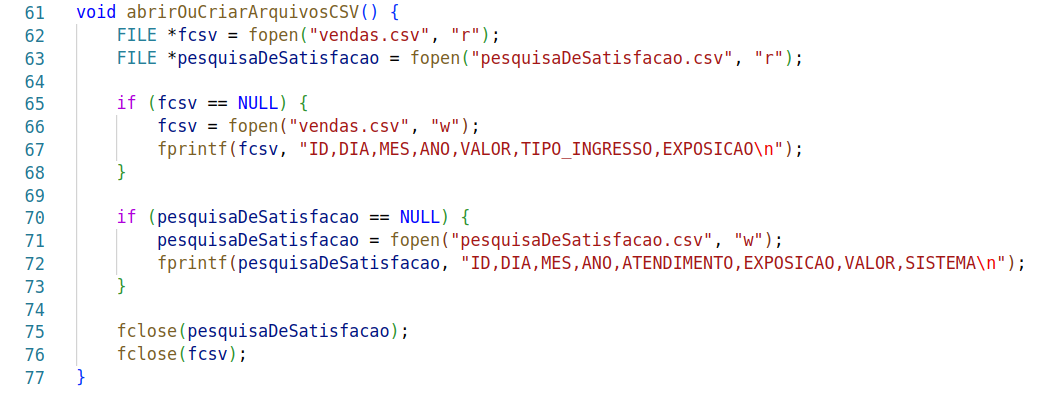
A rotina imprime é usada para mostrar um texto no console com tabulação e quebra de linha, facilitando a visualização por parte dos usuários.



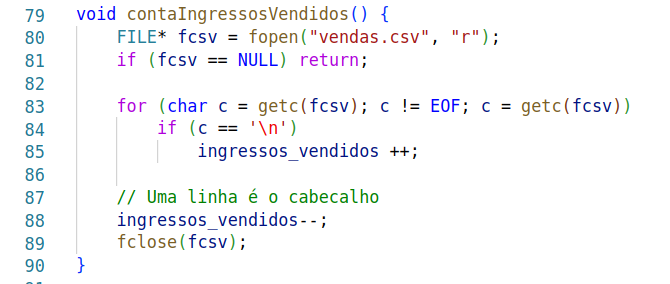
Rotina que mostra no console uma mensagem de boas-vindas, sendo executada sempre que o sistema é inicializado.



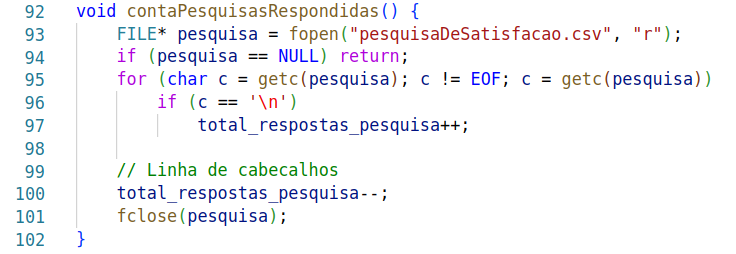
A função mostra opções informa ao usuário as funcionalidades disponíveis e qual é o número correspondente a cada uma.



Rotina que verifica se os arquivos “vendas.csv” e “pesquisaDeSatisfacao.csv” existem, tentando abri-lo no modo de leitura. Caso a abertura resulte em nulo, o que significa que o arquivo não existe, o arquivo é aberto no modo de escrita, fazendo com que este seja criado. Após sua criação, colocamos o cabeçalho contendo os campos que serão armazenados com o cadastro das vendas.

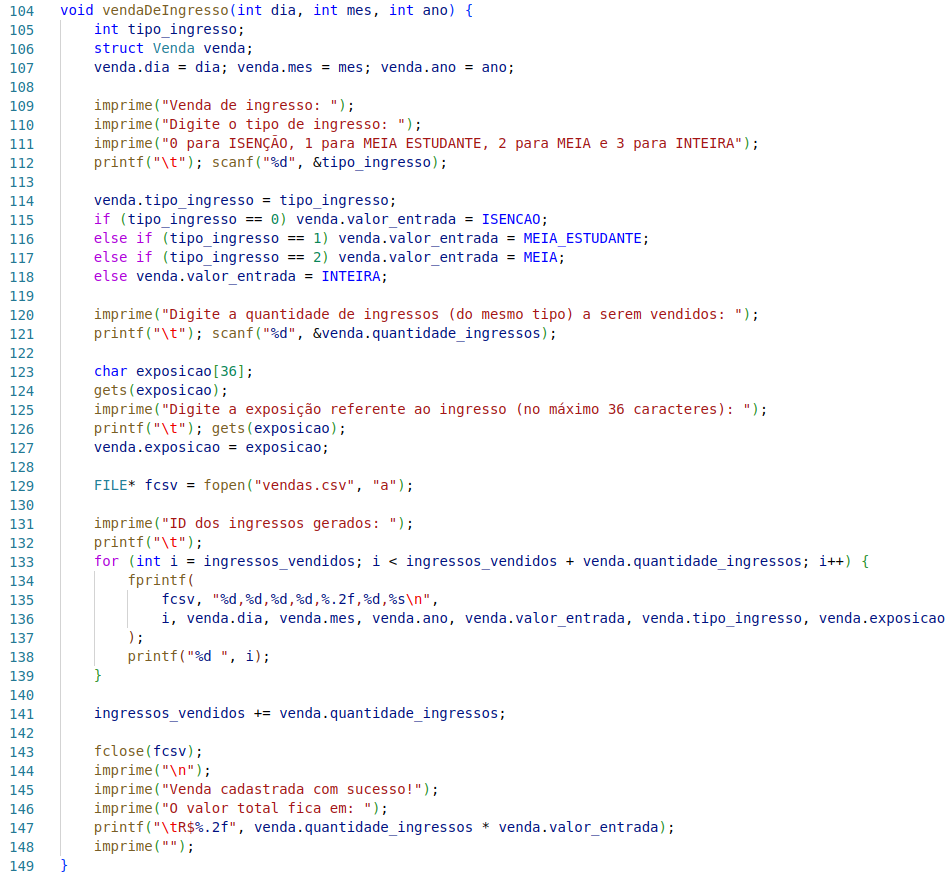


Esta função conta o número de registros existentes no arquivo “vendas.csv” para descobrir quantos ingressos já foram vendidos até o momento. Essa informação é utilizada posteriormente no cadastro de venda para gerar um número de identificação (ID) único e sequencial para cada ingresso vendido. O primeiro ingresso vendido terá o ID 0, o segundo terá ID 1 e assim por diante. Após reiniciar o sistema, a contagem começará a partir do número 2, visto que os números 0 e 1 já foram usados como ID anteriormente.



Esta função tem a mesma função da anterior, porém referente às respostas da pesquisa de satisfação, gerando um ID a ser utilizado para identificar unicamente cada resposta e quantas pesquisas foram respondidas.

**1.3 Venda de ingresso**



A função de venda de ingressos recebe a data da venda como parâmetro, sendo dividido em dia, mês e ano. Para começar, a rotina captura o tipo de ingresso na forma de número inteiro, sendo 0 para entrada isenta, 1 para meia de estudante, 2 para meia e 3 para inteira. Com o tipo de ingresso informado, o programa já calcula o valor do ingresso baseado nas constantes definidas no começo do arquivo.

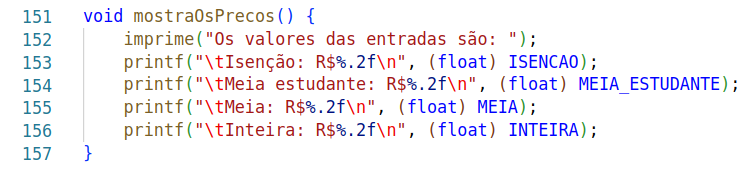
Depois, o programa pergunta a quantidade de ingressos do mesmo tipo a serem vendidos, é possível vender vários ingressos de um mesmo tipo ao mesmo tempo, porém caso seja necessário venda uma entrada inteira e uma meia entrada, por exemplo, será necessário chamar esta função duas vezes.

Em seguida é questionado qual será a exposição que o visitante quer ver, sendo armazenada no formato de texto e deve ser limitada em 36 caracteres no máximo.

Com os dados coletados, o arquivo “vendas.csv” é aberto no modo de adição, onde podemos adicionar informações ao final do arquivo. Realizamos um laço *for* que vai do número de ingressos vendidos até o número de ingressos vendidos mais a quantidade de ingressos da venda atual, a cada iteração do laço escrevemos no arquivo “vendas.csv” uma nova linha contendo as informações de cada um dos ingressos que estão sendo vendidos e mostra na tela o ID dos ingressos gerados.

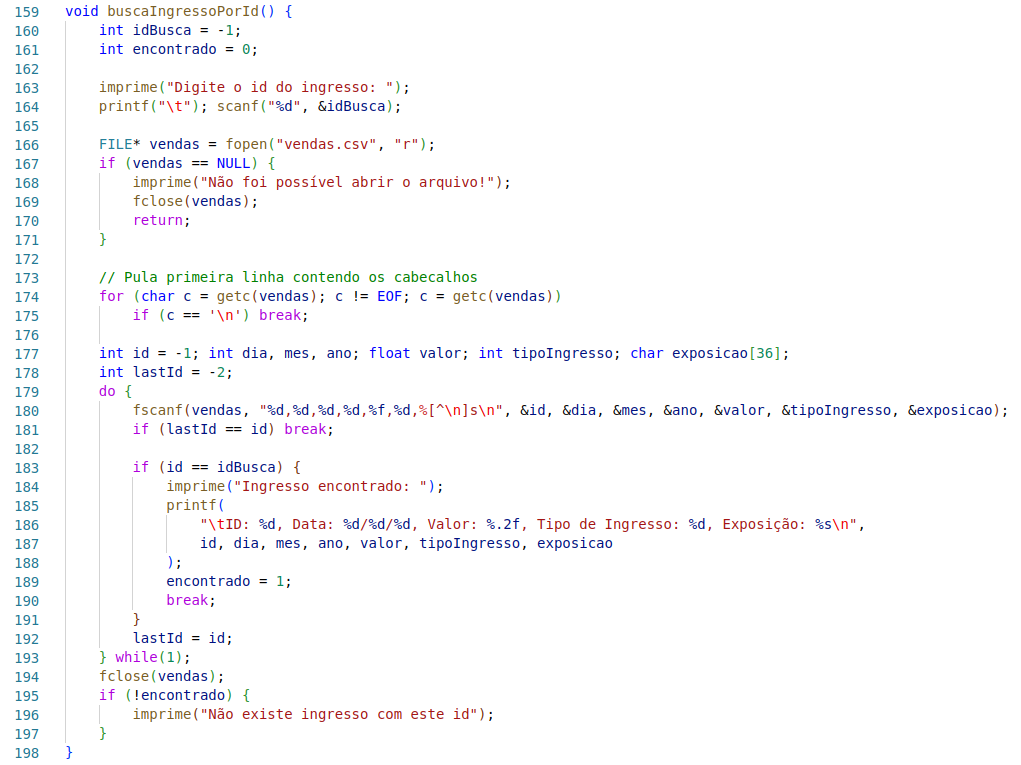
Após isso, a variável contendo a quantidade de ingressos vendidos no total é atualizada e valor total da venda é mostrado no terminal.

**1.4 Consulta de preços**



A função mostra os preços informa ao usuário o valor da entrada de cada um dos tipos de entrada, sendo 0 para isenção, 10 reais para meia de estudante, R$12,50 para meia entrada e 25 reais para entrada inteira. Os valores são provenientes das constantes definidas no topo do arquivo.

**1.5 Busca de ingresso por id**

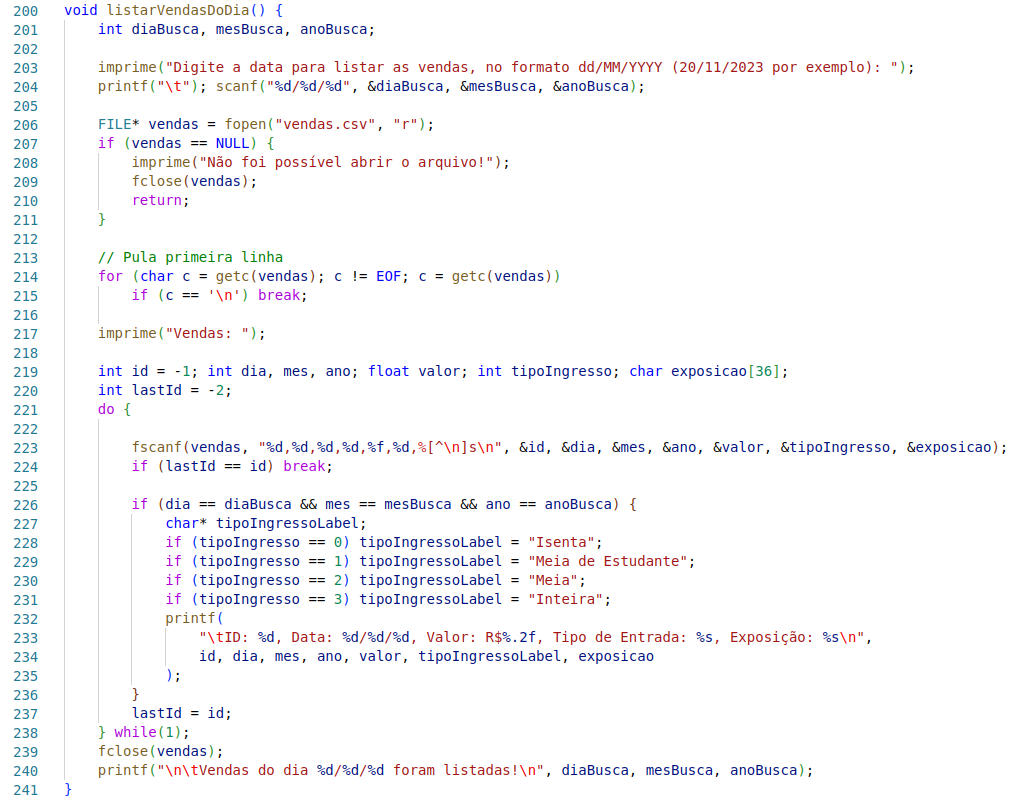


A função de busca de ingresso por ID inicia perguntando ao usuário qual o ID do ingresso a ser buscado, assim o arquivo “vendas.csv” é aberto no modo de leitura, sendo que há uma verificação de se foi possível abri-lo.

Em seguida, iniciamos um laço de repetição *while* para ler o conteúdo do arquivo e procurar o registro contendo o mesmo ID daquele que foi informado, caso o ingresso seja encontrado, as informações do ingresso são mostradas na tela e o laço de repetição é interrompido forçadamente.

Caso o ingresso com o ID informado não for encontrado, o laço de repetição também será interrompido forçadamente e uma mensagem de erro será mostrada na tela. A verificação se o arquivo chegou ao fim consiste através da variável inteira *lastId*, caso o ID do ingresso lido pelo comando *fscanf* for igual ao último ID encontrado, isso significa que o arquivo chegou ao fim. Em um arquivo contendo os ingressos com IDs 1, 2 e 3, o valor do ID lido a cada iteração será 1, 2, 3, 3, 3 e assim por diante, sendo que o 3 se repetirá infinitamente, por isso quebramos o laço de iteração forçadamente assim que a primeira repetição acontecer.

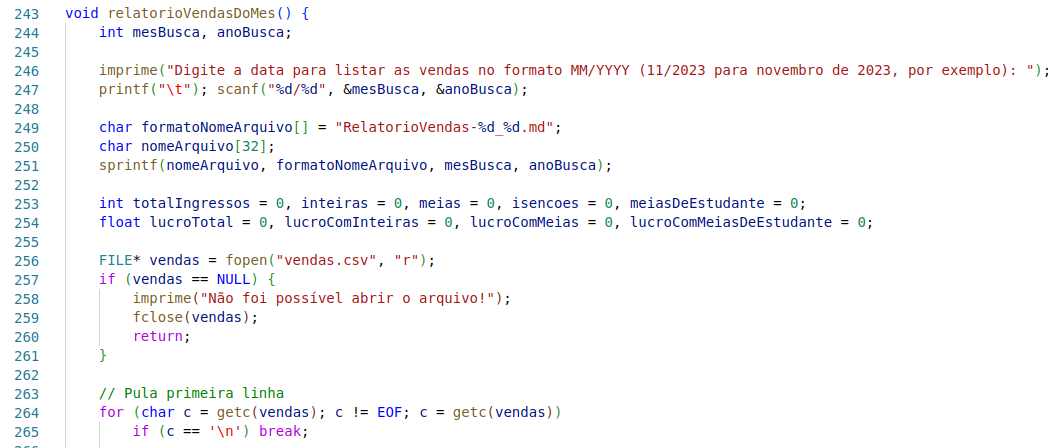
**1.6 Listagem de vendas do dia**



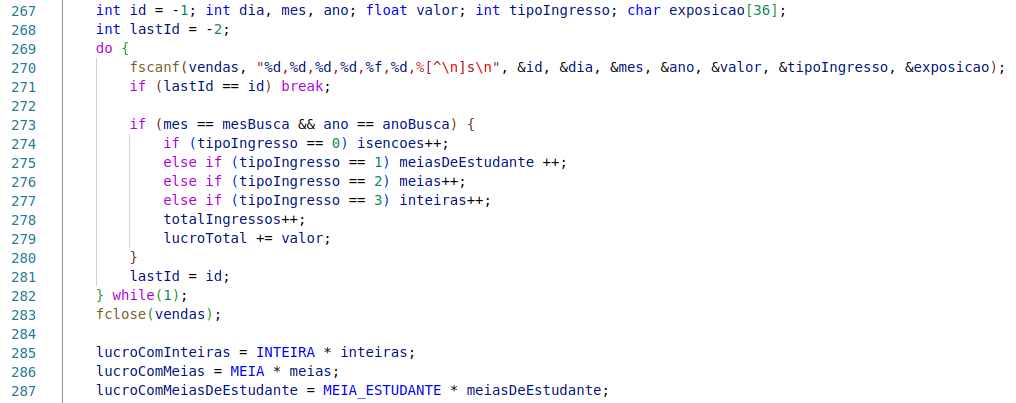
A função de listagem inicia perguntando ao usuário qual é a data da busca dos ingressos, sendo necessário passar as informações no formato dd/MM/YYYY, como em 24/09/2023. A primeira linha do arquivo é pulada, pois contém o cabeçalho com o nome das colunas. Em seguida, abrimos o arquivo “vendas.csv” no modo leitura e realizamos um laço *while* do mesmo modo que utilizamos na função de busca por ID, tendo a mesma verificação se o final do arquivo foi alcançado.

No laço de repetição, os dados do ingresso são armazenados na memória e caso a data seja a igual àquela que foi informada, as informações da venda são mostradas na tela. Assim que o arquivo chega ao fim, o laço é interrompido e uma mensagem é mostrada ao usuário informando que a rotina foi finalizada.

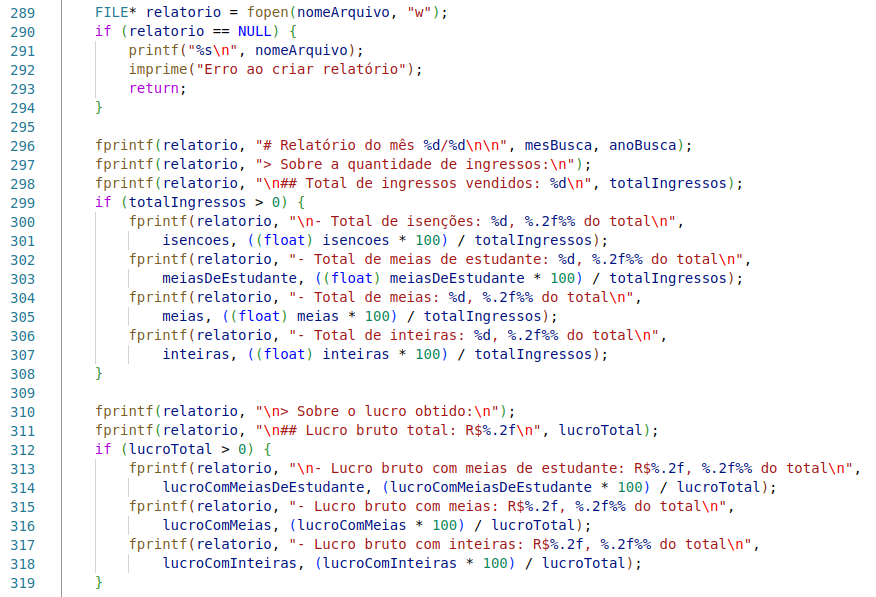
**1.7 Relatório de vendas por mês**



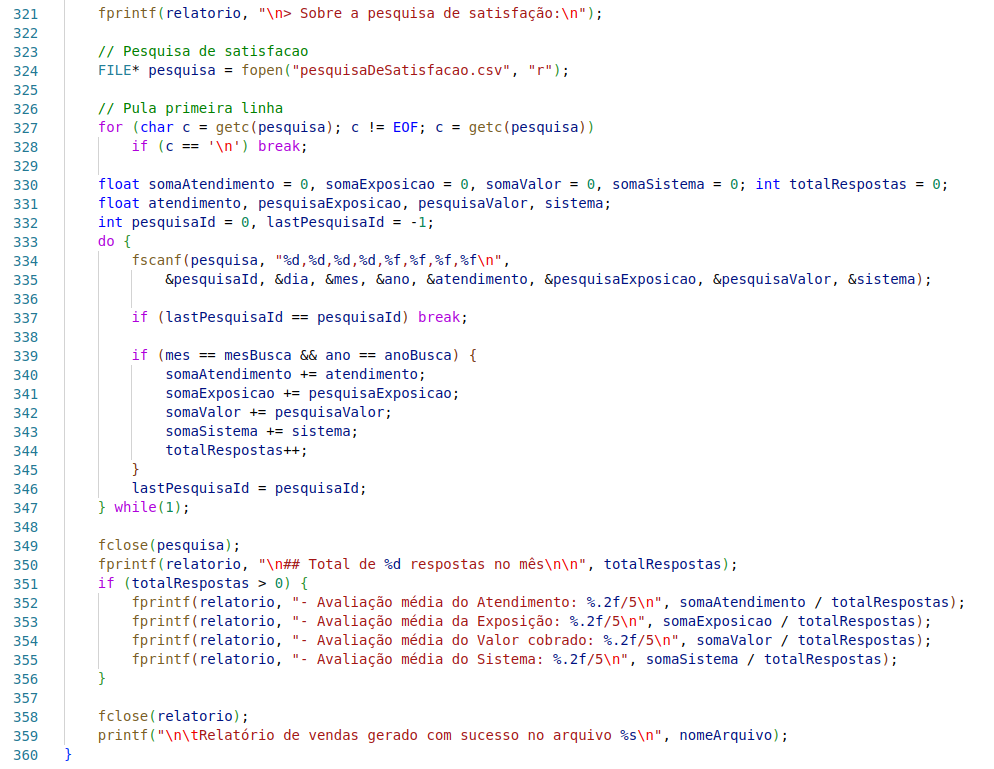
Esta primeira parte da função pergunta ao usuário o mês e o ano referentes às vendas de ingresso, cria o nome do arquivo que será gerado com o relatório, usando como base o mês e o ano. Logo em seguida, inicializamos as variáveis que serão usadas para contar as estatísticas e abrimos o arquivo, verificando se a abertura foi bem-sucedida e pulando a primeira linha com o cabeçalho.



Em seguida, realizamos o mesmo laço de repetição, com a mesma interrupção forçada com o comando *break* que utilizamos nas outras rotinas. Aqui estamos contando quantos ingressos de cada tipo foram vendidos no mês, assim atualizando as variáveis de contagem e as variáveis que armazenam o lucro bruto obtido com cada tipo de ingresso. O lucro obtido é calculado multiplicando o total de ingressos de cada tipo com o valor do tipo de entrada.



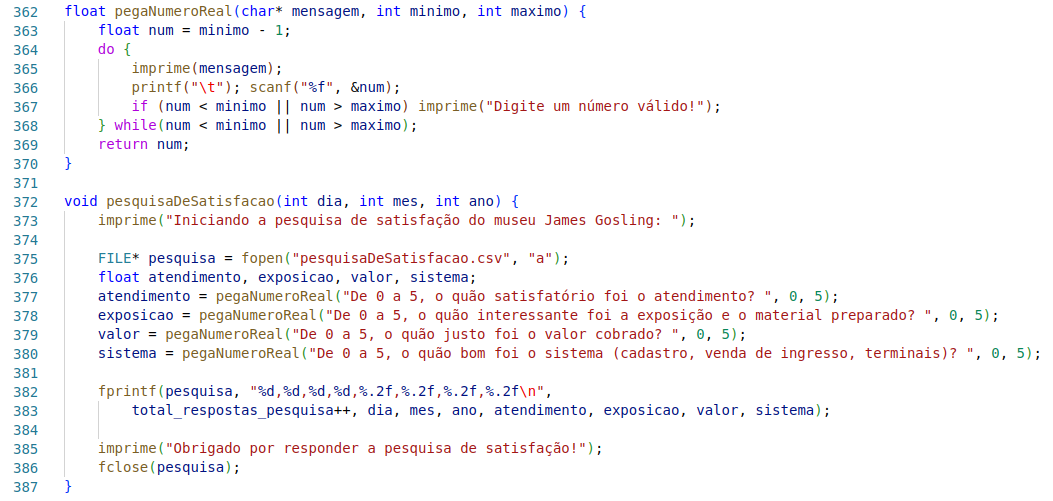
Na parte final da rotina, criamos o arquivo do relatório, o abrindo no modo de escrita e verificando se foi possível, caso contrário mostrando uma mensagem de erro. Iniciamos com um cabeçalho no relatório, informando o mês a qual se refere. Em seguida, colocamos informações sobre o total de ingressos vendidos, caso o número de ingressos vendidos tenha sido maior que zero, informamos o total de ingressos e a porcentagem com relação ao total de cada tipo de entrada (isenção, meia de estudante, meia e inteira). Por último, mostramos informações sobre o lucro bruto obtido no mês e qual foi o lucro obtido e o percentual por cada tipo de entrada.



Em seguida, incluímos informações sobre as pesquisas de satisfação respondidas no período referente à busca do usuário, realizamos uma média da avaliação de cada critério estabelecido: atendimentos, exposição, valor cobrado e sistema. A média é adicionada ao relatório e informamos quantas respostas tivemos no mês.

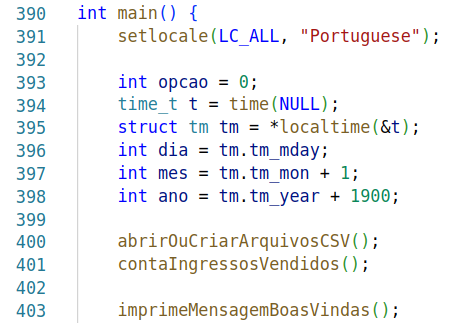
Finalizamos informando o usuário que o relatório foi gerado, e mostrando o nome do arquivo criado. O arquivo é criado no formato *.md*, também chamado de *MarkDown*.

**1.8 Responder pesquisa de satisfação**

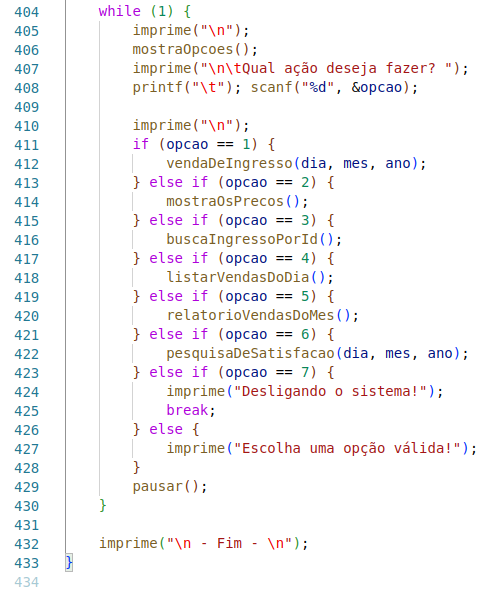


A rotina de responder pesquisa de satisfação recebe a data atual como parâmetro e usa de uma sub-rotina chamada pega número real para solicitar que o usuário avalie alguns pontos do museu. São avaliados o atendimento, exposição, valor cobrado e sistema, cada resposta tem um ID único e é armazenada no arquivo “pesquisaDeSatisfacao.csv”, que foi preparado anteriormente na rotina de abrir ou criar arquivos csv.

**1.9 Método principal**



O método principal começa abrindo ou criando o arquivo “vendas.csv” através da sub-rotina citada anteriormente, em seguida atualizando a contagem de ingressos vendidos para gerar os IDs dos ingressos, e por fim coletando a data atual, através da biblioteca “time.h”, pois essas informações serão usadas posteriormente na hora de cadastrar a venda dos ingressos.

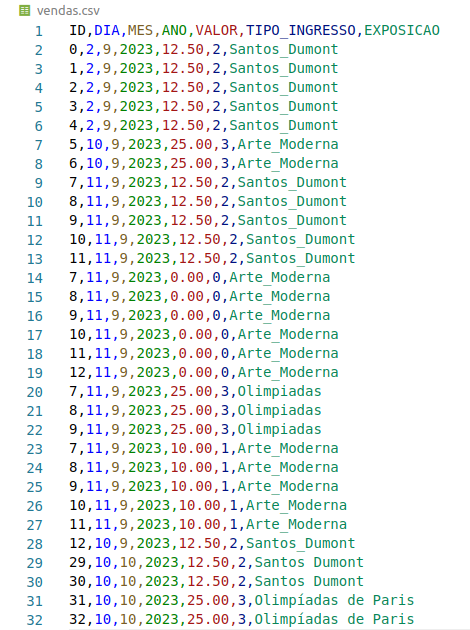


No corpo do método principal temos um laço de repetição sem fim, que continuamente mostra as opções disponíveis ao usuário e pergunta qual ação deverá ser executada. Caso uma opção inválida seja escolhida, uma mensagem de erro é mostrada. Ao fim de cada iteração, executamos um pause, para permitir que o usuário visualize com calma o resultado da operação anterior.

A opção 1 realiza uma venda de ingresso, a opção 2 mostra os preços das entradas, a opção 3 realiza uma busca de ingresso por ID, a opção 4 lista as vendas de determinado dia, a opção 5 gera um relatório com as vendas de um determinado mês, a opção 6 é utilizada para responder a pesquisa de satisfação e, por fim, a opção 7 interrompe a execução do sistema, desligando-o.

**1.10 Formato dos arquivos**

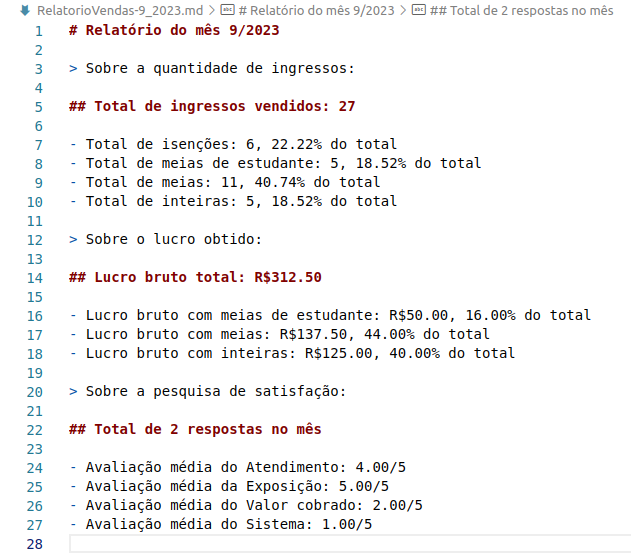
O arquivo “vendas.csv”, que é usado para armazenar os dados das vendas é do formato .*csv*, que significa valores separados por vírgula (do inglês, *comma separated values*). A estrutura do arquivo contém uma linha de cabeçalho, contendo as colunas ID, dia, mês, ano, valor, tipo de ingresso e exposição, em seguida há linhas com informações, de acordo com a seguinte imagem:



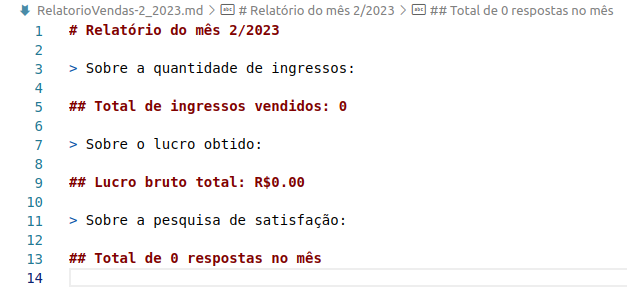
O arquivo “pesquisaDeSatisfacao.csv” armazena as respostas da pesquisa de satisfação e segue o mesmo padrão do arquivo de “vendas.csv”.



Enquanto isso, o arquivo de relatório de vendas é do formato *.md*, chamado de arquivo *MarkDown*,tipo de arquivo que permite algumas formatações adicionais, como inclusão de títulos, subtítulos, imagens e listas*.* Seguindo o seguinte padrão:



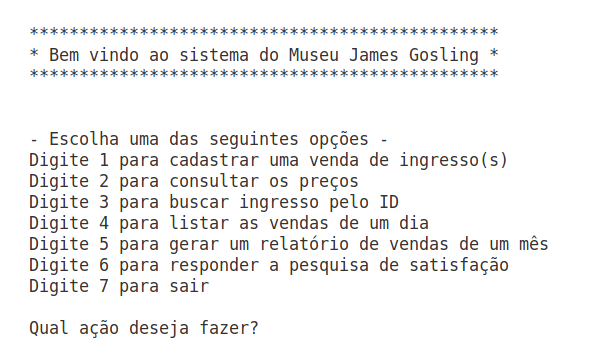
Caso o usuário tente gerar um relatório em um mês em que não houve vendas nem respostas na pesquisa de satisfação, o resultado será o seguinte:



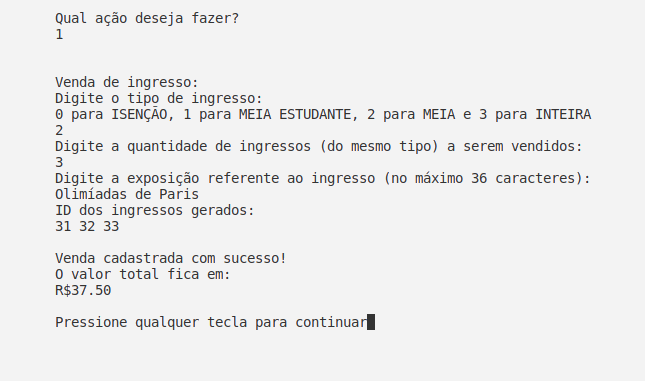
Não são mostradas as porcentagens nem médias, pois neste caso, haveria uma divisão por zero, resultando em um valor inválido.

**1.11 Execução do Sistema**

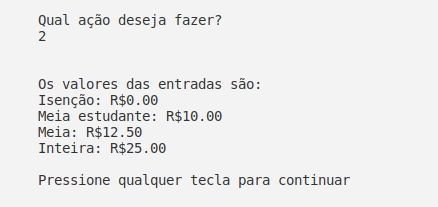
Segue um exemplo da execução de cada função do sistema, começando com a parte inicial:



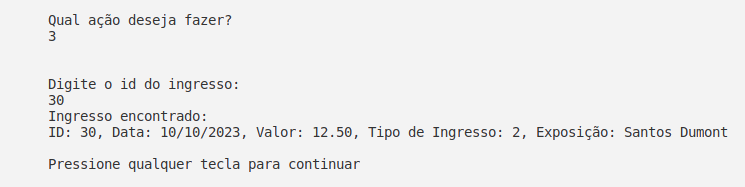
Em seguida, a execução do cadastro da venda de ingresso:



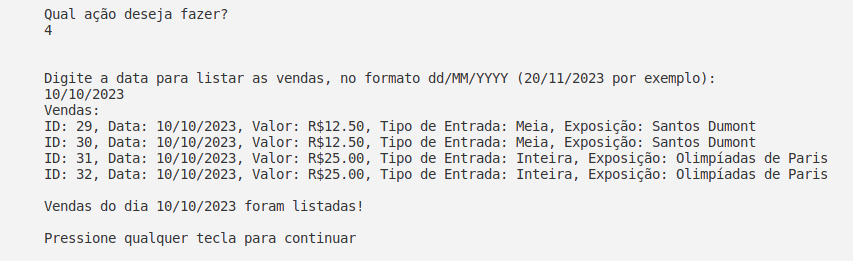
Da consulta de preços:



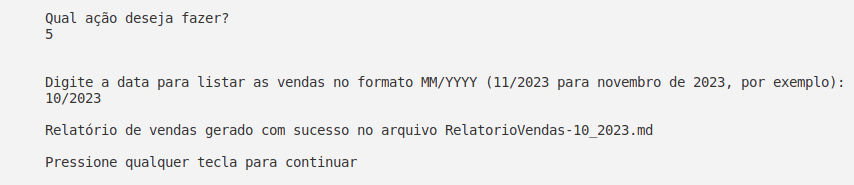
Busca de ingresso por id:



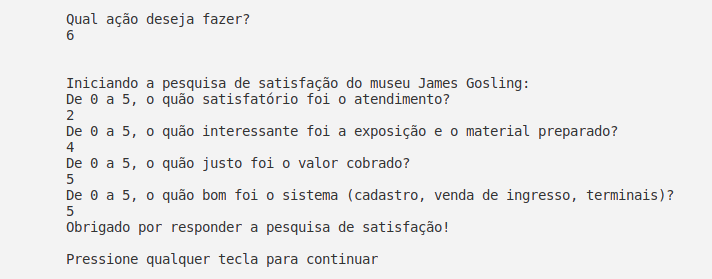
Listagem de ingressos por dia:



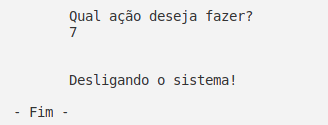
Geração de relatório de vendas mensal:



Pesquisa de satisfação:

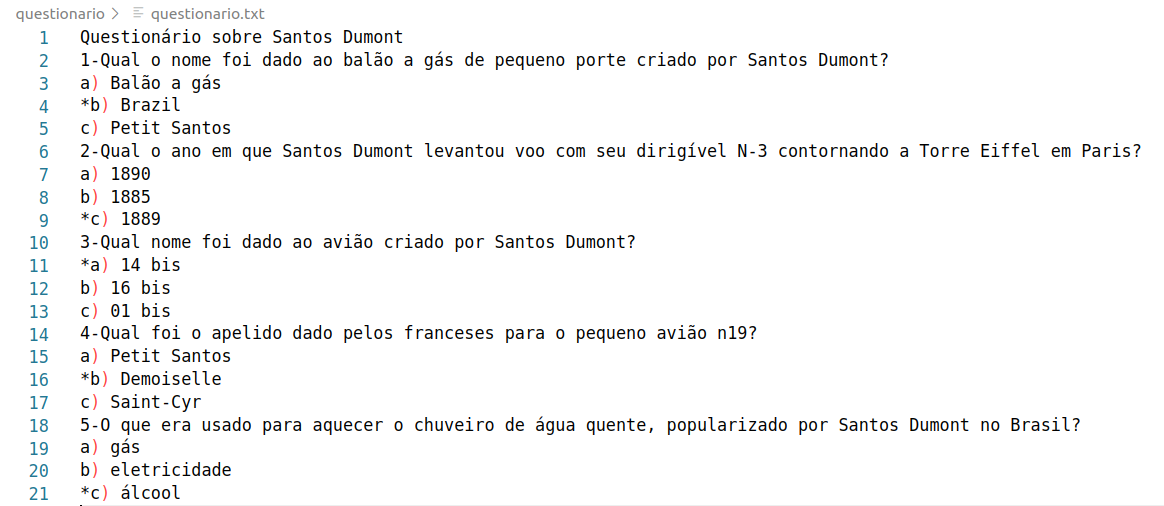


Desligamento do sistema:

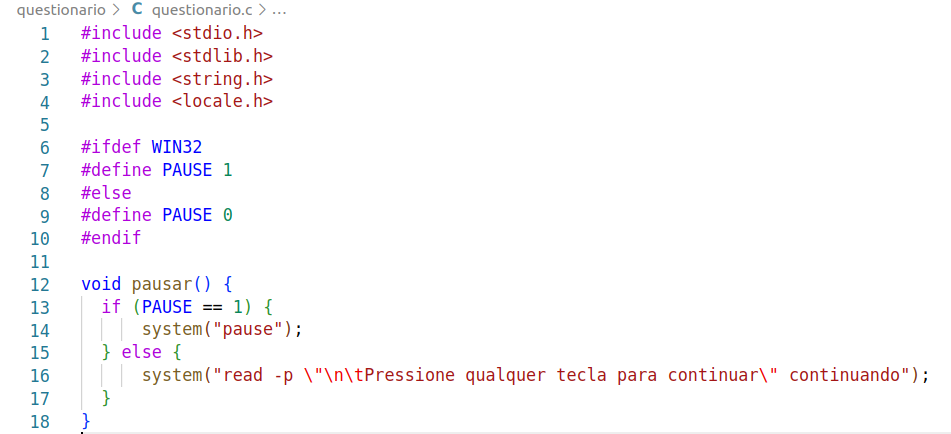


**4.12 Questionários**

De acordo com as especificações e requisitos, os visitantes devem poder responder questionários sobre as exposições através de terminais que ficarão nas salas de cada tema. Cada questionário deverá ser armazenado em um arquivo.txt, chamado “questionario.txt”, esse arquivo conterá todas as questões, assim como as alternativas, que no caso devem ser três: a, b e c. A alternativa correta deverá ser marcada com um asterisco antes da letra da alternativa, de acordo com o seguinte exemplo:



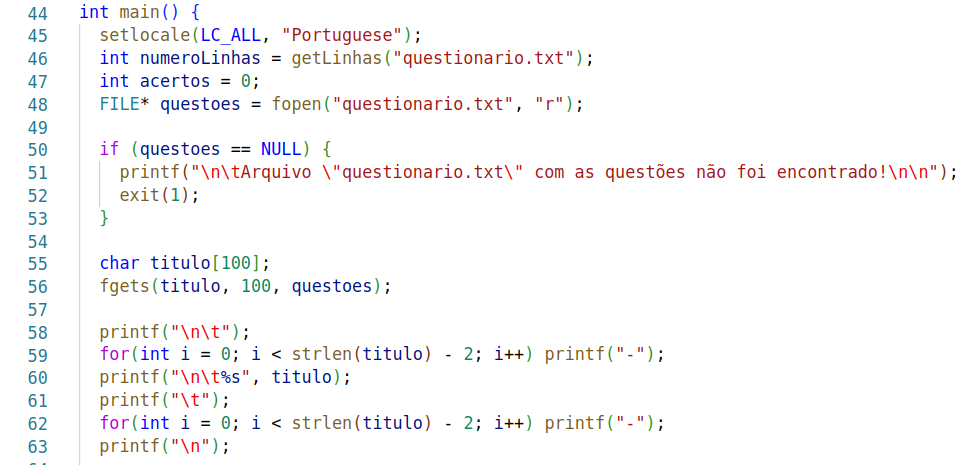
O código para os questionários é o seguinte:



Começamos importando as mesmas bibliotecas que utilizamos no sistema principal anteriormente e implementamos a mesma função de pausar que também já foi citada anteriormente.



Aqui temos algumas funções auxiliares, a função getLinhas pega o número de linhas de um arquivo qualquer, contando o número de caracteres “\n”, que indicam quebra de linha, essa função é usada para verificar quantas questões temos no questionário. A função insereLinhaCSV serve para inserir um novo registro de uma resposta do usuário no arquivo que armazena as respostas dos visitantes, armazenando qual foi a questão respondida, qual alternativa o usuário escolheu e se a questão foi acertada.

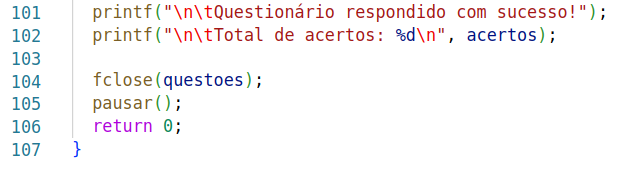


A parte inicial do método principal habilita o uso de caracteres com acento através da função setlocale, em seguida o arquivo do questionário é aberto, verificando se ele existe, interrompendo a execução caso não esteja presente. Em seguida, buscamos o título do questionário e o mostramos, rodeado de linhas com hifens para efeito de estilo.

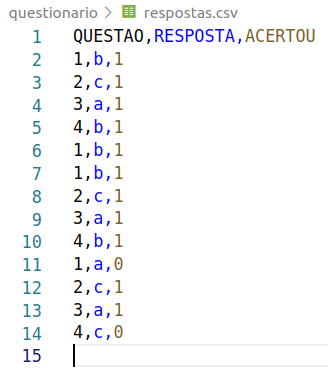


Nessa parte do método principal, temos um laço de repetição que ocorre a cada questão que há no questionário, o número de linhas do arquivo é dividido por 4, já que cada questão contém uma linha com um enunciado mais três linhas com as alternativas. A cada questão, o enunciado é lido e mostrado no console, então as alternativas são lidas do arquivo, verificando qual contém um asterisco no início, indicando que é a resposta correta.

O asterisco da resposta correta não é mostrado ao usuário, que é perguntado qual a sua resposta, digitará em letras minúsculas e será informado se acertou ou não, em seguida um novo registro será inserido no arquivo que armazena as respostas dos visitantes.



O final do método principal informa o total de acertos do visitante e informa que o questionário chegou ao fim, assim finalizando a execução do programa. O arquivo gerado com as respostas do usuário segue o seguinte padrão:



Segue um exemplo da execução do programa:

