UNIVERSIDADE FEDERAL DA GRANDE DOURADOS
Faculdade de Ciências Exatas e Tecnologias - FACET

FABRICIO FOGAÇA DA SILVA LEMOS

RELATÓRIO TRABALHO: Trabalho Show do Milhão
Laboratório de Programação II

Dourados - MS
2022

FABRICIO FOGAÇA DA SILVA LEMOS

RELATÓRIO TRABALHO: Trabalho Show do Milhão

Trabalho realizado no curso de Engenharia de computação como requisito de finalização da disciplina de Laboratório de programação 2 pelo docente Anderson Bessa da Costa.

Dourados - MS
2022

SUMÁRIO

INTRODUÇÃO

O presente relatório se põe a relatar a respeito do desenvolvimento da aplicação “Show do milhão” através da linguagem C, sendo a aplicação “Show do milhão” uma cópia do show de mesmo nome popular da TV brasileira adaptado para sua implementação.

O jogo funciona de maneira simples, são quinze perguntas, cinco de cada nível, iniciando no nível mais fácil e ficando mais difícil, ao responder todas as quinze perguntas corretamente, é lhe dada a opção de responder a última pergunta e sair com um milhão de reais, ou perder tudo. Para ajudar nestas questões, menos na pergunta final, há métodos para te ajudar, métodos limitados.

A lógica para a implementação deste programa assim como as dificuldades no meio desse processo será relatada no próximo tópico.

1- IMPLEMENTAÇÃO

Para comentar a implementação do trabalho dividirei em partes, a primeira a qual comentarei especificamente da main, e em seguida das funções. As bibliotecas adicionadas foram a Stdin, Stdlib e Time. Além de utilizado a struct fornecida pelo professor. A Seed do Srand foi chamada no inicio da Main. Assim como o arquivo foi aberto para leitura e verificado se houve ou não erro.

1.1 – MAIN

Primeiramente vou explicar as variáveis que declarei, pois acredito que com elas já tem a explicação bruta do que eu pensei enquanto produzia o programa. Temos três variáveis criadas para loops for (*'n' 'j' 'i'*), isto pois decidi implementar as quinze primeiras perguntas em um for apenas incrementando o número, e então a última pergunta separada, pois ela é diferente o suficiente das outras perguntas para ser uma implementação separada.

A variável *contador* para se manter a conta de em que pergunta se está, no início esta variável também era usada para se manter o controle do numero da pergunta, mas logo percebi que isso não seria possível pois ao pular uma pergunta o numero da pergunta permanece, mas o contador sobe, por isso separei as variáveis em *contador* e *n_pergunta*, ambas iniciando em zero.

Um *vetor de 18 posições*, este vetor armazena o numero de 18 perguntas para depois em um laço for eu verificar se a pergunta que estou gerando não é uma pergunta repetida, caso for eu realizo o sorteio de novo até obter um resultado diferente ao todo. O número 18 se deve ao fato de serem 15 perguntas obrigatórias para se chegar à última perguntas, mais 3 perguntas que são as puláveis, que também não podem ser repetidas.

A variável *sorteio* é a que decidirá qual pergunta será coletada para ser feita, esse numero é multiplicado no *Fseek* e então recuperado no *Fread* e armazenado na variável *perguntas* do tipo *struct pergunta*.

A variável *acerto* define se houve um acerto ou não e caso tenha havido prosseguir o programa, caso não apenas encerra-lo.

Valido é uma variável de controle que criei para contornar problemas, o problema que a variável *Valido* contorna é o de repetir a pergunta caso seja colocada uma opção invalida quanto às dicas, caso você tenha 0 tentativas de uma dica e tente usa-la de novo, a pergunta se repete e você tem que escolher uma opção válida. Isso

não contorna o fato de você colocar uma opção que não esteja entre as do menu; nesse caso eu achei melhor o programa apenas se encerrar.

Temos quatro *contadores* para as dicas, contadores simples que se decrementam enquanto as dicas são impostas, um float para o *valor* total que se está ganhando no jogo e por fim um char que é a *opção* escolhida pela pessoa no menu.

Poucas coisas restaram para serem explicadas na main após a explicação das variáveis, as perguntas são lidas através da variável *perguntas* tipo *struct perguntas*, a pergunta é escolhida através de uma função, e após que a pergunta é lida na tela uma *opção* é retirada do usuário, se essa *opção* é a opção correta vai para a próxima pergunta, aumentando-se o prémio recebido, caso ele erre o programa se encerra.

Os casos de dicas, com exceção da opção ‘1’, chamam sua função, dão a dica, subtrai 1 ao contador da dica e então pedem uma resposta, o caminho a seguir é igual ao se ele tivesse digitado anteriormente a resposta. No caso de ser escolhido a opção ‘1’, que é pular a pergunta, além de diminuir seu próprio contador de dica, ele também diminui 1 ao *i*, dizendo basicamente que deve se realizar um loop a mais e diminui o numero da pergunta, pois o número deve permanecer o mesmo, além de condicionais if depois analisarem se a opção escolhida for ‘1’, caso for não é adicionado nenhum dinheiro para o prémio nem é dado como se a pessoa houvesse errado a questão. Nada acontece, além da pergunta ser pulada e não poder ser repetida.

A última pergunta é feita fora do for, como ela é a última também é o fim do programa, ela segue a mesma lógica anterior, mas agora não há dicas e eu também adicionei algo que acontece no programa, mas não estava especificado no texto, que é a opção de não responder a última pergunta e sair com o seu prémio acumulado, no caso, meio milhão de reais.

1.2 – SORTEADOR

O Sorteador é um switch que calculará os quatro casos onde é necessário retirar uma pergunta, o caso das primeiras cinco, onde se deve retirar das 20 primeiras, as próximas 5 que é entre 21 e 40, as próximas que são entre 41 e 60 e finalmente a pergunta final que é entre 61 e 70.

Para isso é realizado uma seleção aleatória através do Srand no môdulo de quantas perguntas existem no intervalo, somando o valor de início. Por exemplo, caso queremos as perguntas entre a posição 61 e 70, acontece que como pegamos o início do endereço no Fseek isso na verdade seria como 60 e 69, então fazemos rand()%10, para retirarmos um numero aleatório entre 0 e 9 e somamos ao 60 que é o inicio do intervalo. Obtendo assim de maneira aleatória um valor entre 60 e 69.

1.3 – PLATEIA, UNIVERSITÁRIOS E CARTAS

Plateia e Universitário se baseiam na mesma lógica, onde chamo a função probabilidades, com pesos diferentes, em i tantas vezes e imprimo na tela esse valor como se fossem as respostas da plateia ou dos universitários. Antes de eu decidir fazer uma função separada para a probabilidades tudo era feito dentro dessas funções, o que as deixava enorme, separando as deixou mais diminuta e simples.

A Cartas se baseia em gerar um numero de 0 a 3 aleatoriamente e então realizar uma das quatro opções de remoção de cartas, para as cartas não se repetirem as armazeno em um vetor de chars para comparação. A seleção aleatória das cartas é feita por meio da função probabilidades e para dizer quais opções foram descartadas para o usuário eu falo as letras em vez de imprimir de novo só as opções restantes, esta foi uma decisão feita pois achei desnecessário imprimir de novo as perguntas que já estariam em cima.

1.4 – PROBABILIDADES

A função probabilidades é uma função genérica criada para gerar probabilidades baseada em peso, até chegar nela foram várias tentativas e erros, a ideia de se usar intervalos para serem pesos estava desde o início, mas até a realização e implementação do que seria necessária uma função genérica muitas linhas foram escritas e apagadas.

Esta função recebe a struct da pergunta para a leitura da opção correta, e então quatro pesos de probabilidades; O peso da probabilidade correta, um peso b, um peso c e por fim o peso da última probabilidade.

Como a lógica é baseada em intervalos para a utilização dessa função temos que pensar quais intervalos queremos para cada opção, pensaremos em dois casos, o caso da plateia e o caso das cartas.

No caso da plateia a opção correta é 70% de chance de ocorrer, e as outras opções subsequentemente 10% cada. Isso pode ser facilmente pensado em um intervalo de 1 até 10 ou de 1 até 100, utilizaremos até 100. A primeira opção, a correta é até 70, então esse é o número passado como prob_correta, as restantes probabilidades são somando 10 ao intervalo, logo 80, 90 e por fim 100. Sobra apenas gerar um numero aleatoriamente de no máximo valor igual ao ultimo intervalo, no caso 100. Existem 70 chances de cair a probabilidade correta e 10 para cada uma das seguintes. Assim um modo de gerar aleatoriamente uma opção, mas com cada opção tendo um peso diferente.

No caso das cartas ocorre a mesma coisa, mas podemos ver em maior estado o poder de se criar uma função genérica, pois nas cartas nós nunca queremos retirar a opção correta, e as opções seguintes devem ter 1/3 de chance cada de saírem. Então é simples, definimos o caso da probabilidade correta sendo 0, ou seja, nunca vai acontecer pois o menor numero gerado é 1, e as seguinte como sendo intervalos possíveis de se dividir em três, no caso o utilizado foi 4,8 e 12, assim existem opções iguais de se cair cada uma das três opções que não sejam a opção correta, que nunca irá cair, pois no caso das cartas, ela não pode ser eliminada.

2 – AMBIENTE E CONCLUSÃO

O ambiente em que esse trabalho foi realizado é uma máquina rodando a versão mais atualizada do Windows 10, usando-se da IDE Codeblocks e compilador GNU GCC.

Com a realização deste trabalho é possível tirar várias conclusões, a criação de um projeto é uma ótima maneira de abrir os olhos para o poder da programação que as vezes o aluno por estar muito preso a teoria não percebe. Criar o hábito da manipulação de arquivos no aluno somado aos desafios de desenvolver a lógica de todo um jogo fermenta a visão do que é realmente programar, fora dos pequenos programas que fazemos no dia a dia das aulas. Pensamentos desde como deixar algo mais eficiente, como diminuir as linhas, problemas que aparecem e precisam ser debuggados, toda essa experiência só se é possível ao desenvolver um programa que se passa das 200 linhas, e é uma experiência necessária a qualquer estudante.