Universidade Federal de Mato Grosso do Sul

Faculdade de Computação (FACOM)

Relatório do Trabalho Prático 2

Show do Milhão

Algoritmos e Programação 2

Prof. Dr. Anderson Bessa da Costa

Autores:
Isabele Maria Firmino
Joao Pedro Santos de Brito

Campo Grande, 04 de Novembro de 2025

1 Introdução

O trabalho descreve a implementação do jogo "Show do Milhão", desenvolvido como avaliação prática para a disciplina de Algoritmos e Programação 2. O objetivo foi aplicar os conceitos fundamentais da linguagem C, com ênfase no gerenciamento de memória, manipulação de arquivos binários e modularização de código.

O programa simula o jogo de perguntas e respostas, carregando um banco de 70 perguntas de um arquivo binário, gerenciando uma partida de 16 rodadas com dificuldade progressiva e implementando as quatro ajudas clássicas (Pular, Plateia, Universitários e Cartas).

2 Decisões de Implementação

O projeto foi estruturado seguindo as boas práticas de programação, como a verificação de erros e a correta liberação de recursos alocados.

2.1 Conceitos Aplicados das Aulas

Para a construção do projeto, foram utilizados diversos conceitos abordados em aula:

- **Variáveis Compostas (Structs):** A ‘struct pergunta‘ foi utilizada exatamente como sugerido.
- **Ponteiros e Alocação Dinâmica:** A alocação dinâmica com ‘malloc‘ foi usada para carregar todas as 70 perguntas na memória principal. O ponteiro liberado com ‘free‘ ao final do programa.
- **Passagem por Referência (Ponteiros):** Todas as funções de ajuda recebem ponteiros para os contadores (ex: ‘int *pulos_restantes‘), permitindo que a função modifique a variável original na ‘main‘.
- **Arquivos Binários:** O núcleo do programa utiliza ‘fopen‘ no modo leitura binária (‘"rb"‘) e ‘fread‘ para ler os dados do arquivo ‘perguntas.dat‘ diretamente para a ‘struct‘ alocada.
- **Arrays e Lógica de Seleção:** Vetores foram usados para armazenar os níveis de perguntas (‘faceis‘, ‘medias‘, etc.) e para a lógica de probabilidade, enquanto a função ‘rand()‘ foi usada para a seleção aleatória.

2.2 Estrutura de Dados e Carregamento

Seguindo a especificação de que as perguntas estavam ordenadas por nível no arquivo, a estratégia de carregamento foi:

1. Alocar dinamicamente um vetor principal (‘todasPerguntas‘) para 70 ‘structs‘.
2. Abrir ‘perguntas.dat‘ em modo ‘"rb"‘ e verificar se era ‘NULL‘ .
3. Usar ‘fread‘ dentro de um laço ‘while‘ para ler as 70 perguntas para o vetor alocado.
4. Fechar o arquivo com ‘fclose‘.

5. Usar ‘memcpy‘ para copiar os blocos de perguntas do vetor principal para quatro vetores menores (ex: ‘faceis[20]‘, ‘medias[20]‘), facilitando a seleção aleatória durante o jogo.

6. Liberar a memória do vetor principal cno final da execução.

2.3 Lógica Principal do Jogo

O fluxo do jogo é controlado por dois laços ‘while‘ principais:

- **Laço Externo:** Controla a rodada atual (‘while (pergunta_atual <= 16 ...)‘). A seleção da pergunta aleatória (ex: ‘rand() % 20‘) é feita no início deste laço.
- **Laço Interno:** Controla a interação com a pergunta atual (‘while (!pergunta_respondida ...)‘). Este laço permite que o jogador use ajudas (como Plateia ou Cartas) sem que a pergunta seja trocada, pois a mesma é exibida repetidamente até que o jogador responda, pare ou pule.

2.4 Implementação das Funções de Ajuda

As quatro ajudas foram realizadas em funções separadas:

- **Pular Pergunta:** A função ‘executar_pulos‘ decrementa o contador (via ponteiro) e retorna ‘true‘. Na ‘main‘, este retorno ‘true‘ atribui ‘pergunta_respondida = true‘, quebrando o laço interno e forçando o laço externo a selecionar uma nova pergunta do mesmo nivel.
- **Plateia e Universitários:** Ambas as funções (‘executar_plateia‘ e ‘executar_universitarios‘) implementam a lógica de probabilidade através de um modelo de "limite acumulado". As probabilidades (ex: 40/20/20/20) são somadas (ex: 40, 60, 80, 100), e um ‘rand() % 100‘ é comparado com esses limites para determinar o voto.
- **Ajuda das Cartas:** Esta função manipula um vetor ‘bool visiveis[4]‘ que é passado como parâmetro. A função identifica as alternativas incorretas que ainda estão visíveis, as embaralha e oculta um número fixo delas, alterando o vetor ‘visiveis‘. A função ‘exibir_interface‘ foi modificada para ler este vetor e exibir ‘[Alternativa Excluida]‘ quando o valor é ‘false‘.

3 Funcionalidades Implementadas

Todas as funcionalidades descritas no manualforam implementadas. O programa carrega as perguntas, gerencia os 16 turnos, aplica a pontuação correta (incluindo o zerar do prêmio em caso de erro) e permite o uso das quatro ajudas com seus respectivos contadores e lógicas de probabilidade. A verificação para impedir que o jogador escolha uma alternativa já excluída também foi implementada.

4 Problemas Enfrentados

O principal desafio foi ajustar a lógica do "Pulo Falso", onde tentar pular com 0 pulos restantes fazia o jogo selecionar uma nova pergunta. Isso foi corrigido movendo a lógica de seleção da pergunta para *antes* do laço interno de ajuda, garantindo que a mesma pergunta seja reexibida caso a ajuda falhe.

Também houve uma dificuldade na compreensão da lógica de probabilidades acumuladas. A implementação da 'Plateia' e 'Universitários' exigia mapear um número aleatório (‘rand() % 100‘) para probabilidades distintas (ex: 40%, 20%, etc.). A técnica de somar as probabilidades (ex: 20, 40, 80, 100) para criar 'tetos' de comparação para o ‘if/else if‘ foi um conceito que acabou sendo um pouco complexo.

5 Ambiente de Desenvolvimento

O trabalho foi desenvolvido e compilado pelo VSCODE, num SO Windows e utilizando versionamento Git entre a dupla.