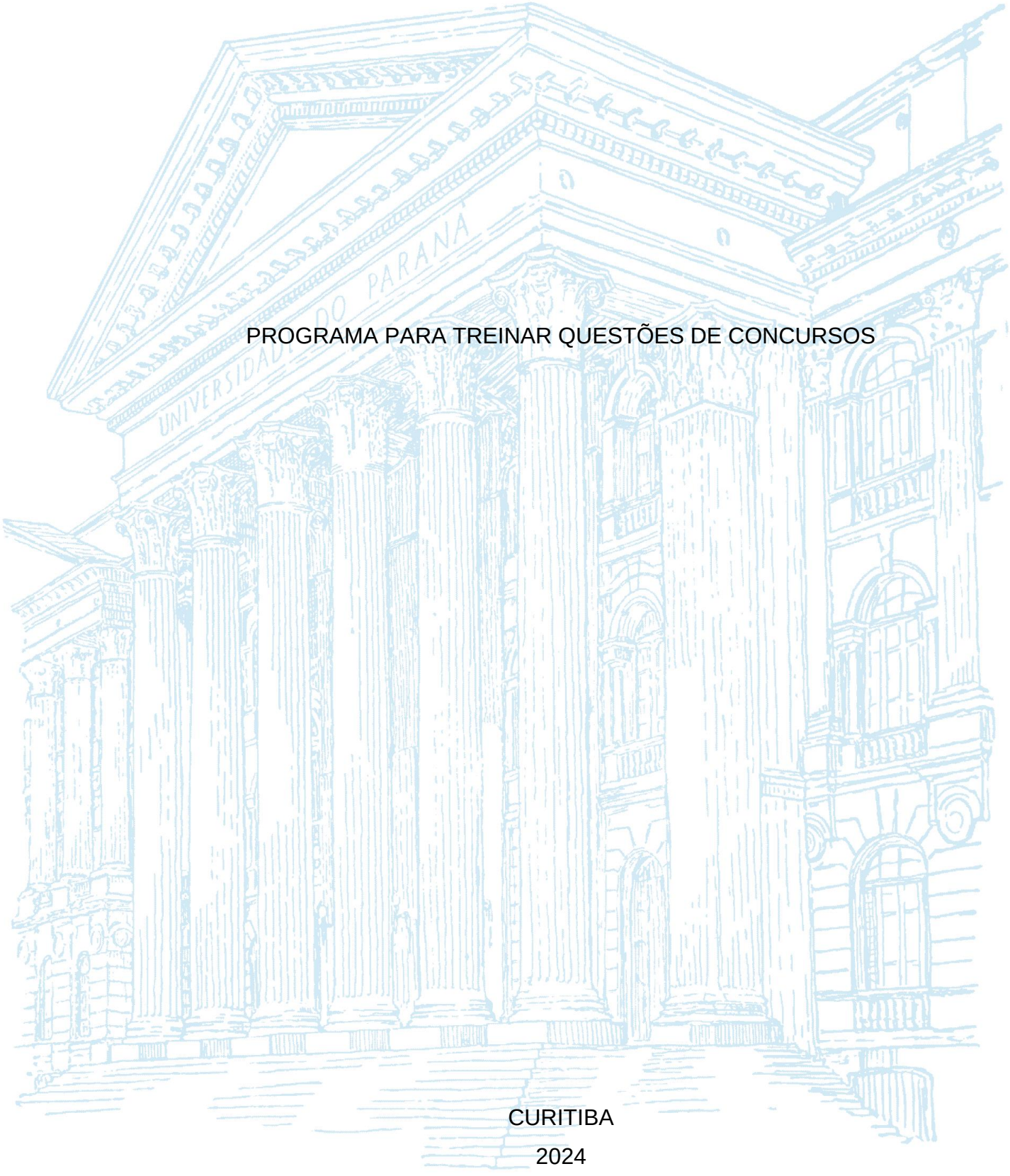
UNIVERSIDADE FEDERAL DO PARANÁ

DIEGO PUJOL ALVARES



PROGRAMA PARA TREINAR QUESTÕES DE CONCURSOS

CURITIBA

2024

Diego Pujol Alvares

PROGRAMA PARA TREINAR QUESTÕES DE CONCURSOS
: UM PROGRAMA NA LINGUAGEM C

Trabalho apresentado na disciplina Linguagem de Programação do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Federal do Paraná, como requisito parcial à obtenção da nota final.

CURITIBA

2024

SUMÁRIO

1. INTRODUÇÃO AO DOCUMENTO 16
2. DESCRIÇÃO GERAL DO SISTEMA 17
3. REQUISITOS DO SISTEMA 17
4. ANÁLISE E DESIGN 18
5. IMPLEMENTAÇÃO 20
6. TESTES 21
7. IMPLANTAÇÃO 21
8. MANUAL DO USUÁRIO 22
9. CONCLUSÕES E CONSIDERAÇÕES FINAIS 23

1. INTRODUÇÃO AO DOCUMENTO

1.1. Tema

O projeto aborda o desenvolvimento de um simulador de provas de concurso público utilizando a linguagem de programação C.

1.2. Objetivo do Projeto

O principal objetivo deste projeto é proporcionar uma ferramenta para candidatos a concursos públicos praticarem questões e avaliarem seu desempenho. O sistema oferece um ambiente simulado de prova, onde os usuários podem responder a questões, registrar suas pontuações e ver rankings de desempenho.

1.3. Delimitação do Problema

Candidatos a concursos públicos muitas vezes enfrentam dificuldades na preparação devido à falta de ferramentas práticas e acessíveis. Este projeto visa suprir essa necessidade, proporcionando um simulador de provas que permite aos usuários praticarem questões de maneira estruturada e monitorarem seu progresso.

1.4. Justificativa da Escolha do Tema

A preparação para concursos públicos é um processo que requer disciplina e prática contínua. Uma ferramenta que simula provas pode ajudar os candidatos a se familiarizarem com o formato das questões e a gerenciarem seu tempo de forma mais eficaz. A escolha do tema se justifica pela necessidade crescente de recursos educacionais que possam melhorar o desempenho dos candidatos.

1.5. Método de Trabalho

O desenvolvimento do projeto seguiu uma abordagem iterativa e incremental, com etapas de análise de requisitos, design arquitetural, implementação, testes e documentação. O software foi desenvolvido em linguagem C, utilizando as melhores práticas de programação e modularização de código.

1.6. Organização do Trabalho

O documento está organizado em capítulos que cobrem introdução, descrição geral do sistema, requisitos, análise e design, implementação, testes, implantação, manual do usuário, e conclusões.

1.7. Glossário

- **Simulador de Provas**: Aplicativo que permite a prática de questões de concurso.
- **Questões**: Perguntas de múltipla escolha ou certo/errado que compõem as provas simuladas.
- **Ranking**: Lista ordenada de usuários com base na pontuação e tempo gasto na prova.

- **Usuário**: Pessoa que utiliza o simulador para responder questões e ver seu desempenho.

2. DESCRIÇÃO GERAL DO SISTEMA

2.1. Descrição do Problema

A preparação para concursos públicos pode ser desafiadora devido à falta de recursos para prática. Este projeto oferece um simulador de provas que permite aos candidatos praticarem questões de diferentes áreas, registrar suas pontuações e ver um ranking de desempenho, ajudando-os a melhorar suas habilidades e a se prepararem de forma mais eficaz.

2.2. Principais Envolvidos e suas Características

2.2.1. USUÁRIOS DO SISTEMA

- **Candidatos**: Pessoas que estão se preparando para concursos públicos. Necessitam de ferramentas para praticar questões e avaliar seu desempenho.
- **Desenvolvedores**: Equipe responsável pelo desenvolvimento, manutenção e melhoria do sistema.

2.3. Regras de Negócio

- O simulador deve permitir que o usuário responda a 30 questões aleatórias em uma prova simulada.
- As questões devem ser distribuídas entre diferentes áreas de conhecimento.
- O tempo máximo para a prova é de 2 horas.
- O ranking deve ser ordenado por pontuação em ordem decrescente e, em caso de empate, pelo menor tempo de resolução.

3. REQUISITOS DO SISTEMA

3.1. Requisitos Funcionais

- **Cadastro de Usuários**: O sistema deve permitir o cadastro de usuários.
- **Seleção de Questões Aleatórias**: O sistema deve selecionar questões aleatórias para cada prova.

- **Registro de Pontuação e Tempo**: O sistema deve registrar a pontuação e o tempo gasto pelo usuário.
- **Exibição de Ranking**: O sistema deve exibir um ranking com as melhores pontuações.
- **Visualização de Gabarito**: O sistema deve permitir a visualização do gabarito ao final da prova.

3.2. Requisitos Não-Funcionais

- **Linguagem de Programação**: O sistema deve ser desenvolvido em linguagem C.
- **Execução em Terminal**: O sistema deve ser executado em um terminal.
- **Armazenamento em Arquivos Texto**: O sistema deve utilizar arquivos texto para armazenar as questões e o ranking.

4. ANÁLISE E DESIGN

4.1. Arquitetura do Sistema

O sistema segue uma arquitetura monolítica, onde um único aplicativo desenvolvido em C é responsável por todas as funcionalidades. A modularização é feita através do uso de funções que se encarregam de diferentes tarefas, como leitura de arquivos, seleção de questões, cálculo de pontuação, exibição de ranking, entre outras.

4.2. Modelo do Domínio

O modelo do domínio inclui as seguintes estruturas de dados:

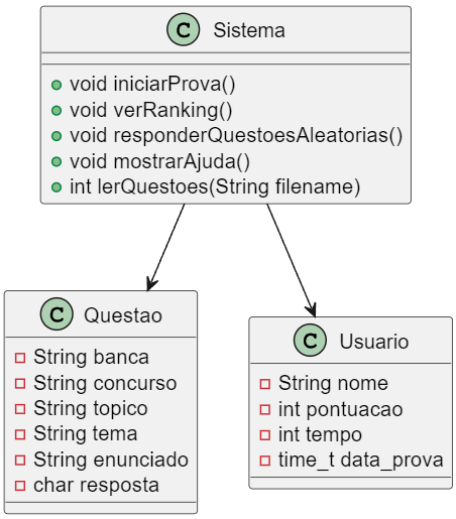
- **Estrutura `Questao`**: Armazena as informações sobre cada questão, incluindo banca, concurso, tópico, tema, enunciado e resposta.
- **Estrutura `Usuario`**: Armazena as informações sobre cada usuário, incluindo nome, pontuação, tempo gasto e data da prova.

4.3. Diagramas de Interação

- **Diagrama de Seqüência**: Representa o fluxo de interação entre o usuário e o sistema durante a realização da prova.
- **Diagrama de Colaboração**: Mostra como as diferentes partes do sistema colaboram para realizar tarefas como seleção de questões, registro de respostas e cálculo de pontuação.

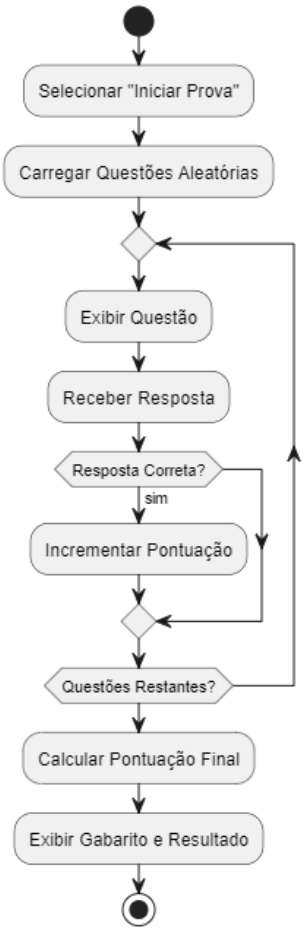
4.4. Diagrama de Classes

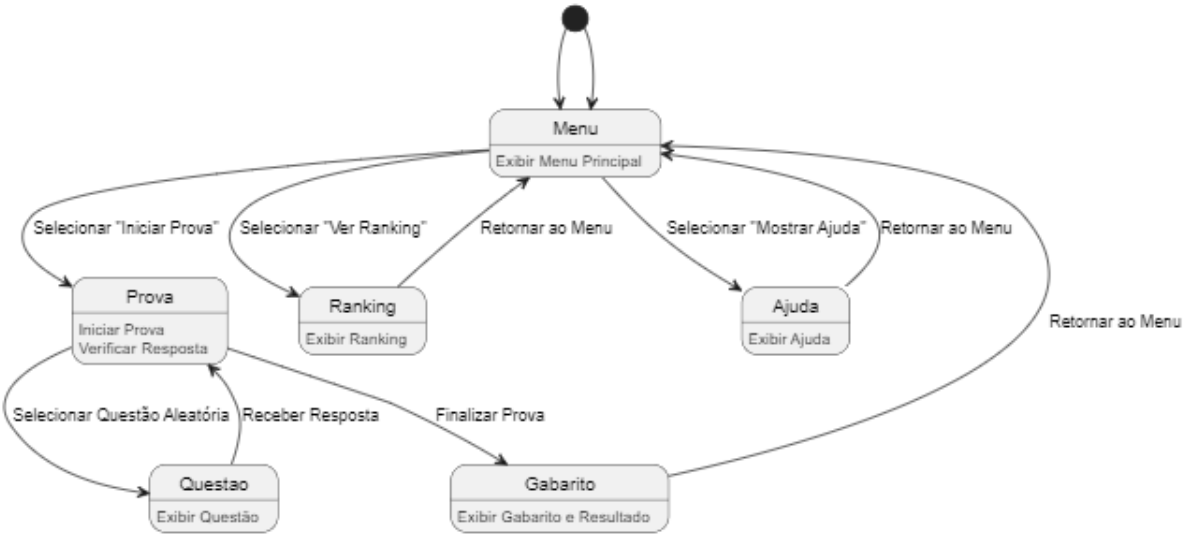
Inclui classes como `Questao` e `Usuario`, com atributos e métodos necessários para gerenciar as informações das provas e dos usuários.



4.5. Diagrama de Atividades e Estado

Representa o fluxo de atividades desde o início da prova até a exibição do resultado final e a atualização do ranking.





5. IMPLEMENTAÇÃO

5.1. Estruturas de Dados

As principais estruturas de dados utilizadas no sistema são as estruturas `Questao` e `Usuario`.

```
typedef struct {
    char banca[50];
    char concurso[100];
    char topico[100];
    char tema[100];
    char enunciado[512];
    char resposta;
} Questao;

typedef struct {
    char nome[50];
    int pontuacao;
    int tempo; // em segundos
    time_t data_prova;
} Usuario;
```

5.2. Funções Principais

O código foi dividido em várias funções para modularizar o comportamento do sistema e promover a reutilização de código. As principais funções incluem:

- `void limparTela()`: Limpa a tela do terminal.
- `void telaInicial()`: Exibe a tela inicial do sistema.
- `void mostrarAjuda()`: Exibe a tela de ajuda com instruções detalhadas.
- `char obterRespostaValida()`: Valida a resposta do usuário.
- `void mostrarTemporizador(int tempoRestante)`: Mostra o tempo restante durante a prova.

- `void mostrarGabarito(Questao *questoes, int *questoesSelecionadas, char *respostasUsuario, int totalQuestoes)`: Mostra o gabarito ao final da prova.
- `void mostrarResultados(Usuario usuario, Questao *questoes, int *questoesSelecionadas, char *respostasUsuario)`: Exibe os resultados da prova.
- `void iniciarProva(Questao *questoes, int totalQuestoes)`: Inicia a prova, selecionando questões aleatórias e registrando as respostas do usuário.
- `void verRanking()`: Exibe o ranking dos usuários.
- `void exibirRankingPorPeriodo()`: Exibe o ranking dos usuários por período.
- `void responderQuestoesAleatorias(Questao *questoes, int totalQuestoes)`: Permite ao usuário responder questões aleatórias.
- `void exibirMenu(Questao *questoes, int totalQuestoes)`: Exibe o menu principal do sistema.
- `int lerQuestoes(const char *filename, Questao *questoes)`: Lê as questões de um arquivo texto.

6. TESTES

6.1. Plano de Testes

- **Testar a Leitura e Escrita de Arquivos**: Verificar se as funções de leitura e escrita de arquivos funcionam corretamente.
- **Verificar a Seleção Aleatória de Questões**: Garantir que as questões são selecionadas aleatoriamente.
- **Validar o Cálculo de Pontuação e Tempo**: Testar se a pontuação e o tempo são calculados corretamente.
- **Testar a Exibição do Ranking**: Verificar se o ranking é exibido corretamente.

6.2. Execução do Plano de Testes

Os testes foram realizados em um ambiente de desenvolvimento local, verificando a consistência e a funcionalidade do sistema conforme os requisitos especificados.

7. IMPLANTAÇÃO

7.1. Diagrama de Implantação

O sistema é implantado em um computador com um compilador C e acesso ao terminal.

7.2. Manual de Implantação

1. **Pré-requisitos**:
   - Compilador C (como GCC).
   - Arquivo de questões (`questoes_formatadas_iso_8859_1.txt`).
   -
2. **Compilação do Código**:

```
gcc simulador_provas.c -o simulador_provas
```

3. **Execução do Programa**:

```
./simulador_provas
```

8. MANUAL DO USUÁRIO

8.1. Iniciar

Prova

- **Passo 1**: Selecione a opção ‘Iniciar Prova’ no menu principal.
- **Passo 2**: Digite seu nome quando solicitado.
- **Passo 3**: Responda a 30 questões, digitando ‘C’ para Certo ou ‘E’ para Errado.
- **Passo 4**: Veja seu resultado e gabarito ao final.

8.2. Ver Ranking

- **Passo 1**: Selecione a opção ‘Ver Ranking’ no menu principal.
- **Passo 2**: Veja a lista dos melhores desempenhos.

8.3. Responder Questões Aleatórias

- **Passo 1**: Selecione a opção ‘Responder Questões Aleatórias’ no menu principal.
- **Passo 2**: Responda questões aleatórias sem limite de tempo.
- **Passo 3**: Digite ‘C’ para Certo ou ‘E’ para Errado.

8.4. Ajuda

- **Passo 1**: Selecione a opção ‘Ajuda’ no menu principal.
- **Passo 2**: Leia as instruções detalhadas sobre como utilizar o simulador de provas.

9. CONCLUSÕES E CONSIDERAÇÕES FINAIS

O simulador de provas desenvolvido atende aos requisitos especificados, proporcionando uma ferramenta prática e eficaz para a preparação de candidatos a concursos públicos. Futuras melhorias podem incluir uma interface gráfica, adição de mais funcionalidades como análise de desempenho e integração com outras ferramentas de estudo.