Fatec Araras Antonio Brambilla

Centro Paula Souza

Documentação do Sistema de Controle de Sócios e Portaria

Versão: v1.0

Maikon Fabrício Gino

ARARAS/SP
DEZEMBRO/2023

SUMÁRIO

1. Introdução..... 03

2. Bibliotecas Utilizadas..... 03

2.1. <iostream>..... 03

2.2. <fstream>..... 03

2.3. <cstring>..... 04

2.4. <locale>.....04

2.5. using namespace std..... 04

3. Estrutura do Código..... 04

3.1. Constantes..... 04

3.2. Estruturas de Dados..... 04

3.2.1. struct Socio..... 04

3.2.2. struct RegistroPortaria..... 05

4. Funções..... 05

4.1. ‘inserirSocio’..... 05

4.2. ‘listarSocios’..... 05

4.3. ‘removerSocio’..... 05

4.4. ‘registrarEntrada’..... 05

4.5. ‘registrarSaida’..... 05

4.6. ‘gerarRelatorio’..... 06

5. Geração de relatório (arquivo .txt) 06

6. Função ‘main’..... 07

7. Utilização do Sistema..... 07

7.1. Inserir Sócio (‘opcao = 1’)..... 07

7.2. Listar Sócios (‘opcao = 2’)..... 07

7.3. Remover Sócio (‘opcao = 3’)..... 07

7.4. Registrar Entrada (‘opcao = 4’)..... 07

7.5. Registrar Saída (‘opcao = 5’)..... 08

7.6. Gerar Relatório (‘opcao = 6’)..... 08

7.7. Sair do Programa (‘opcao = 0’)..... 08

8. Considerações Finais..... 08

1. Introdução

O Sistema de Controle de Sócios e Portaria foi desenvolvido em C++ para gerenciar informações de sócios em um clube, proporcionando funcionalidades como inserção, remoção, listagem, e o registro de entradas e saídas na portaria.

Esta documentação oferece uma visão detalhada da estrutura do código, das estruturas de dados e das funções implementadas, bem como um guia claro para a utilização do sistema. Desenvolvido para proporcionar uma abordagem organizada e intuitiva, esse sistema visa otimizar a administração de dados dos associados, promovendo transparência, agilidade e segurança nas operações diárias.

Em ambientes clubísticos, a gestão de sócios e controle de acesso são elementos cruciais para assegurar uma experiência harmoniosa e satisfatória.

Ao longo desta documentação, exploraremos minuciosamente cada aspecto do código-fonte, desde a escolha criteriosa das bibliotecas até a implementação de estruturas de dados eficazes. Cada função e bloco de código foi concebido com a finalidade de garantir não apenas a funcionalidade técnica, mas também a compreensão intuitiva do programa, tornando-o acessível mesmo para aqueles que não possuem profundo conhecimento em programação.

Este software, em essência, visa ser uma ferramenta indispensável para a modernização e aprimoramento dos processos internos de clubes, proporcionando uma solução tecnológica eficaz e adaptável às necessidades específicas de cada instituição. A seguir, apresentaremos uma análise detalhada de cada componente do código, proporcionando uma compreensão abrangente e transparente do Sistema de Controle de Sócios e Portaria.

2. Bibliotecas Utilizadas

2.1. <iostream>

Responsável por fornecer funcionalidades de entrada e saída padrão. Utilizada para interação com o usuário através da console.

2.2. <fstream>

Usada para operações de leitura e escrita em arquivos, possibilitando o armazenamento de informações relevantes.

2.3. <cstring>

Fornece funções para manipulação de strings. Utilizada para operações específicas relacionadas a cópias e manipulação de dados de strings.

2.4. <locale>

Permite a configuração da localidade para tratar caracteres especiais e formatação de acordo com a região.

2.5. using namespace std

A instrução using namespace std simplifica o uso de elementos do namespace padrão (std) em C++. Ela permite o acesso direto a recursos como cout, cin e outros sem a necessidade de usar o prefixo std::.

3. Estrutura do Código

3.1. Constantes

- ‘MAX_SOCIOS’: Limite máximo de sócios suportados.

- ‘MAX_REGISTROS’: Limite máximo de registros de portaria suportados.

- ‘MAX_NOME_TITULAR’: Tamanho máximo do nome do titular do título do sócio.

- ‘LISTA_SOCIO’: Caminho do arquivo para salvar o relatório de visitas.

3.2. Estruturas de Dados

3.2.1. struct Socio

Representa a estrutura de dados que armazena informações sobre um sócio, como número de título, nome do titular, números de dependentes e status de pagamento de mensalidade**.**

- ‘numeroTitulo’: Número de identificação do título do sócio.

- ‘nomeTitula’ Nome do titular do título do sócio.

- ‘numeroDependente1’ e ‘numeroDependente2’: Números de identificação dos dependentes.

- ‘pagamentoMensalidade’: Indica se o sócio está em dia com a mensalidade.

3.2.2. struct RegistroPortaria

Define a estrutura de dados para armazenar informações de registros na portaria, incluindo o número de título, nome e se a entrada foi registrada (true para entrada, false para saída).

- ‘numeroTitulo’: Número de identificação do título do sócio associado ao registro.
- ‘nome’: Nome do titular do título do sócio associado ao registro.
- ‘entrada’: Indica se o registro é de entrada (‘true’) ou saída (‘false’).

4. Funções

4.1. ‘inserirSocio’: Responsável por inserir um novo sócio no sistema, caso o limite máximo de sócios ainda não tenha sido atingido.

```
void inserirSocio(Socio socios[], int& totalSocios, Socio novoSocio);
```

4.2. ‘listarSocios’: Exibe os dados de todos os sócios cadastrados, incluindo número de título, nome do titular, números de dependentes e status de pagamento de mensalidade.

```
void listarSocios(const Socio socios[], int totalSocios);
```

4.3. ‘removerSocio’: Remove um sócio com base no número de título fornecido. Se o sócio não for encontrado, uma mensagem apropriada é exibida.

```
void removerSocio(Socio socios[], int& totalSocios, int numeroTitulo);
```

4.4. ‘registrarEntrada’: Registra a entrada de um sócio na portaria, verificando se o sócio existe, está em dia com as mensalidades e se a capacidade máxima de registros não foi alcançada.

```
void registrarEntrada(const Socio socios[], int totalSocios, RegistroPortaria registros[], int& totalRegistros, int numeroTitulo);
```

4.5. ‘registrarSaida’: Registra a saída de um sócio na portaria, se ele foi previamente registrado na entrada.

```
void registrarSaida(RegistroPortaria registros[], int& totalRegistros, int& totalRegistros, int numeroTitulo);
```

4.6. ‘gerarRelatorio’: Gera um relatório de visitas, salvando as informações no arquivo especificado em “LISTA_SOCIO” (.txt). Caso não seja possível abrir o arquivo, uma mensagem de erro é exibida.

```
void gerarRelatorio(const RegistroPortaria registros[], int totalRegistros[], int totalRegistros);
```

5. Geração de relatório (arquivo .txt)

A parte do código que está relacionada à geração de um relatório de visitas e à gravação desse relatório em um arquivo.

5.1. ofstream arquivo(LISTA_SOCIO);: Aqui, um objeto da classe ofstream (que é usada para escrever em arquivos) é criado com o nome arquivo e associado ao arquivo cujo caminho é especificado pela constante LISTA_SOCIO. Isso significa que todas as operações de escrita subsequentes no objeto arquivo serão refletidas no arquivo associado.

5.2. if (!arquivo.is_open()) {...}: Verifica se o arquivo foi aberto com sucesso. Se houver algum problema na abertura do arquivo (por exemplo, se o caminho do arquivo estiver incorreto), uma mensagem de erro é exibida, e a função retorna, interrompendo a geração do relatório.

5.3. streambuf* coutbuf = cout.rdbuf();: Salva o buffer atual da saída padrão (cout) no ponteiro coutbuf. Isso é feito para que o conteúdo que normalmente seria impresso no console seja redirecionado para o arquivo.

5.4. cout.rdbuf(arquivo.rdbuf());: Redireciona a saída padrão (cout) para o buffer do arquivo, efetivamente redirecionando qualquer saída para o arquivo associado.

A partir deste ponto até cout.rdbuf(coutbuf);, o código está escrevendo no arquivo ao invés do console. Ele gera o relatório de visitas, percorrendo os registros e imprimindo informações relevantes no formato desejado.

5.5. cout.rdbuf(coutbuf);: Restaura o buffer original da saída padrão (cout), desfazendo o redirecionamento para o arquivo.

5.6. arquivo.close();: Fecha o arquivo após a conclusão da escrita.

5.7. cout << "Relatório salvo com sucesso em " << LISTA_SOCIO << "." << endl;: Exibe uma mensagem indicando que o relatório foi salvo com sucesso no arquivo especificado.

Em resumo, essa parte do código redireciona temporariamente a saída padrão para um arquivo, cria e preenche um relatório de visitas nesse arquivo, e, em seguida, restaura a saída padrão de volta ao console. O objetivo final é gerar um relatório de visitas e salvá-lo em um arquivo específico.

6. Função *'main'*

Lógica principal do programa, oferecendo um menu interativo para o usuário.

A função principal (main) é o ponto de entrada do programa. Ela inicia com a configuração da localidade para possibilitar a correta exibição de caracteres especiais. Em seguida, declara as variáveis principais, como arrays de sócios e registros, e inicia um loop do menu principal que permite ao usuário interagir com o sistema.

O menu oferece opções para inserir um novo sócio, listar sócios, remover sócio, registrar entrada, registrar saída, gerar relatório e sair do programa. Cada opção aciona a execução da função correspondente.

O loop continua até que o usuário escolha sair (opção 0).

O código é projetado de maneira modular, facilitando a compreensão e manutenção do sistema de controle de sócios e portaria.

7. Utilização do Sistema

7.1. Inserir Sócio ('opcao = 1'):

- Solicita informações e adiciona um novo sócio.

7.2. Listar Sócios ('opcao = 2'):

- Exibe detalhes de todos os sócios cadastrados.

7.3. Remover Sócio ('opcao = 3'):

- Solicita o número do título e remove o sócio do sistema.

7.4. Registrar Entrada ('opcao = 4'):

- Solicita o número do título e registra a entrada se o sócio estiver em dia.

7.5. Registrar Saída (‘opcao = 5’):

- Solicita o número do título e registra a saída se o sócio estiver como entrada.

7.6. Gerar Relatório (‘opcao = 6’):

- Gera um relatório de visitas, salva as informações em um arquivo de texto.

7.7. Sair do Programa (‘opcao = 0’):

- Finaliza a execução do programa.

8. Considerações Finais

Ao longo da documentação, cada componente do código foi detalhado de forma a proporcionar uma compreensão completa do funcionamento do programa.

A escolha das bibliotecas, como <iostream> e <fstream>, visa facilitar a interação com o usuário e a manipulação de arquivos. A utilização da biblioteca <locale> demonstra a preocupação com a correta formatação de caracteres especiais, adequando-se às convenções regionais.

As estruturas de dados struct Socio e struct RegistroPortaria foram projetadas para armazenar as informações necessárias de forma organizada, facilitando as operações de inserção, remoção e registro de entrada/saída na portaria.

As funções implementadas refletem a modularidade do código, facilitando sua compreensão e manutenção. Cada função desempenha um papel específico, contribuindo para a execução eficiente do programa.

A função principal (main) controla o fluxo do programa e oferece ao usuário um menu intuitivo para interação. O uso do using namespace std simplifica o código, tornando mais legível e concisa a utilização de recursos do namespace padrão.

O Sistema de Controle de Sócios e Portaria oferece funcionalidades essenciais para o gerenciamento eficiente de sócios em um clube. Para futuras expansões, sugere-se a análise de requisitos específicos do clube e a possível implementação de funcionalidades adicionais, como controle financeiro e relatórios mais detalhados. Esta documentação visa facilitar o entendimento e a manutenção do sistema, tornando-o mais eficaz e orientado para o usuário.