Sistema de Gerenciamento de Clientes em Linguagem C para o Studio LB Treinamento Personalizado

Felipe Shoji Pires Endo, Letícia Monteiro Gomes da Silva
Dr. Aldo Henrique Mendes (Orientador)
Centro Universitário Euro-Americano (Unieuro), Brasília, DF, Brasil

Resumo—Este trabalho apresenta o desenvolvimento de um sistema de gerenciamento de clientes em linguagem C, denominado LB Treinamento Personalizado. O sistema implementa autenticação de usuário, cadastro dinâmico de clientes e listagem, utilizando estruturas, ponteiros e alocação dinâmica de memória por meio das funções calloc, realloc e free. A leitura de dados foi realizada com fgets, garantindo entrada de textos com espaços e tratamento de buffer. Foram realizados testes funcionais com 2, 5 e 10 cadastros, observando expansão dinâmica do vetor e tempo médio de execução inferior a 1 segundo. O projeto consolida conceitos de programação estruturada e gerenciamento de memória.

Palavras-chave—Linguagem C; Struct; Ponteiros; Alocação dinâmica; Cadastro de clientes.

I. Introdução

A informatização de processos administrativos é fundamental para pequenas empresas e estúdios de treinamento personalizado, contribuindo para organização de dados e melhoria na gestão de clientes [1]. A linguagem C, por permitir acesso direto à memória e manipulação eficiente de estruturas de dados, é amplamente utilizada no ensino da programação [2].

Este trabalho tem como objetivo desenvolver um sistema denominado LB Treinamento Personalizado, capaz de realizar autenticação de usuário, cadastro e listagem de clientes, utilizando structs, ponteiros e alocação dinâmica.

II. Metodologia

O desenvolvimento foi realizado em linguagem C com compilação no ambiente Code::Blocks. O sistema foi dividido em dois módulos principais: login e gerenciamento de clientes.

A. Estruturas de Dados

Foram definidas duas estruturas principais. A struct Usuario armazena os campos usuário, senha e email. A struct Cliente armazena matrícula, nome, data de nascimento, telefone, email e anamnese [2].

B. Uso de Ponteiros

O sistema utiliza um ponteiro Cliente *clientes para armazenamento dinâmico, além de um ponteiro auxiliar na função main para controle do fluxo principal. A passagem por referência é empregada nas funções de cadastro e listagem [1].

C. Alocação Dinâmica de Memória

A alocação inicial é realizada com calloc. Quando a capacidade máxima é atingida, o vetor é expandido com realloc, dobrando a capacidade. Ao término da execução, free é utilizado para liberar a memória alocada, evitando vazamentos [3].

D. Entrada de Dados

A leitura é realizada com fgets, permitindo entrada de textos com espaços. O tamanho do vetor é informado para garantir leitura correta e evitar inconsistências de buffer [2].

III. Resultados

Foram realizados testes com diferentes quantidades de cadastros para verificar o comportamento da alocação dinâmica. A Tabela I sintetiza os resultados obtidos.

TABELA I

Resultados dos Testes Funcionais

Teste	Qtd. clientes	Resultado
Teste 1	2	Funcionamento correto
Teste 2	5	Expansão dinâmica
Teste 3	10	Funcionamento estável

Durante os testes, a memória foi expandida automaticamente quando o limite inicial foi atingido. O tempo médio de execução permaneceu inferior a 1 segundo em todos os casos.

As Figuras 1 a 4 ilustram, respectivamente, a tela de login, o menu principal, o cadastro de cliente e a listagem de clientes em terminal.

IV. Discussão

Os resultados demonstram que a alocação dinâmica proporciona maior flexibilidade quando comparada a vetores estáticos, permitindo crescimento sob demanda [1]. O sistema apresenta limitações como ausência de persistência em arquivos e ausência de interface gráfica, que podem ser superadas em trabalhos futuros [4].

Outra limitação é a autenticação simples, sem criptografia de senha. Para versões futuras, recomenda-se a aplicação de

funções de hash criptográfico, além da utilização de banco de dados para persistência [4].

V. Conclusão

O sistema desenvolvido permitiu a aplicação prática de conceitos fundamentais da linguagem C, incluindo structs, ponteiros e alocação dinâmica. Os testes confirmaram funcionamento estável e eficiente. O projeto contribuiu para o aprendizado prático de programação estruturada e gerenciamento de memória.

Referências

[1] B. W. Kernighan e D. M. Ritchie, The C Programming Language, 2ª ed. Englewood Cliffs, NJ: Prentice Hall, 1988.

[2] P. Deitel e H. Deitel, C: Como Programar, 6ª ed. São Paulo: Pearson, 2011.

[3] A. Silberschatz, P. B. Galvin e G. Gagne, Operating System Concepts, 9ª ed. Hoboken, NJ: Wiley, 2015.

[4] A. S. Tanenbaum, Structured Computer Organization, 6ª ed. Boston: Pearson, 2013.

[5] A. Henrique. Aldo Henrique [Online]. Disponível em: https://aldohenrique.com.br/. Acesso em: 6 mai. 2026.