Desenvolvimento e Análise de um Sistema de Agenda em Linguagem C

Rafael Vinicius Gomes de Carvalho, Vinicius Cintra Guimarães Lima
Dr. Aldo Henrique Mendes (Orientador)
Centro Universitário Euro-Americano (Unieuro), Brasília, DF, Brasil

Resumo—Este trabalho apresenta o desenvolvimento de um sistema de agenda em linguagem C voltado à gestão de compromissos pessoais. A solução utiliza estruturas (struct), definição de tipos (typedef), ponteiros e alocação dinâmica de memória por meio das funções calloc, realloc e free. O sistema oferece funcionalidades de cadastro, listagem e busca de contatos, com menu interativo implementado por switch-case. Os testes realizados verificaram a correta alocação e dealocação de memória, bem como a integridade das operações de busca. Os resultados indicam que a aplicação executa as operações em tempo inferior a 1 segundo e mantém uso de memória otimizado, demonstrando que C continua sendo uma linguagem adequada para o ensino prático de estruturas de dados e gerenciamento de memória.

Palavras-chave—Linguagem C; Alocação dinâmica; Estruturas de dados; Ponteiros; Sistemas de agenda.

I. Introdução

A gestão de compromissos é atividade essencial no contexto pessoal e profissional. Sistemas de agenda informatizados permitem maior organização e produtividade quando comparados a métodos manuais [1]. A linguagem C, criada por Kernighan e Ritchie, oferece controle direto sobre memória e estruturas de dados, características desejáveis para aplicações que exigem desempenho e baixo consumo de recursos [1].

O presente trabalho tem como objetivo desenvolver e analisar um sistema de agenda em C aplicando os conceitos de struct, typedef, ponteiros, ponteiros duplos e alocação dinâmica. A escolha da linguagem se justifica por sua flexibilidade no gerenciamento de memória e por sua relevância no ensino de fundamentos de programação [2].

II. Metodologia

O sistema foi implementado em linguagem C, com compilação realizada pelo GCC. A arquitetura adotada é modular, dividida em funções específicas para cadastro, listagem e busca de contatos.

Os dados foram organizados por meio de struct e typedef, contendo os campos nome, telefone e email. A alocação dinâmica utiliza calloc para a alocação inicial do vetor de contatos, realloc para expansão da capacidade e free para liberação ao término da execução. O acesso aos dados é realizado por meio de ponteiros e ponteiros duplos, permitindo modificação por referência.

A busca por contatos utiliza a função strcmp para comparação de strings. O menu interativo é controlado por switch-case, alternando entre as opções disponíveis até que o usuário escolha encerrar o programa [2].

III. Resultados

O sistema desenvolvido executa corretamente as operações de cadastro, listagem e busca. A alocação dinâmica permitiu o crescimento do vetor de contatos conforme a necessidade, sem necessidade de tamanho pré-definido. A liberação de memória com free foi confirmada por meio de testes manuais de execução, sem ocorrência de falhas.

Os tempos de resposta observados em ambiente de teste foram inferiores a 1 segundo para as operações principais, com utilização de memória proporcional ao número de contatos cadastrados. A Figura 1 ilustra a execução do sistema em terminal, exibindo o menu, o cadastro de um contato e a busca subsequente.

Fig. 1. Execução do sistema de agenda em terminal: cadastro, listagem e busca de contatos.

IV. Discussão

A solução desenvolvida apresenta abordagem prática para a gestão de compromissos, com organização eficiente dos dados. O uso de structs e ponteiros permitiu encapsulamento e manipulação direta da memória, conforme princípios discutidos por Kernighan e Ritchie [1].

Entre as limitações identificadas destaca-se a ausência de persistência de dados, uma vez que as informações cadastradas são perdidas ao encerramento do programa. Trabalhos futuros podem incorporar persistência via arquivos binários ou bancos de dados, além de interfaces gráficas e algoritmos de busca mais sofisticados [2].

V. Conclusão

O sistema de agenda implementado em C demonstrou ser uma solução funcional para gestão de compromissos, validando a hipótese de que a linguagem permanece adequada para o aprendizado prático de estruturas de dados e alocação dinâmica. As técnicas aplicadas (struct, typedef, ponteiros, calloc, realloc, free, strcmp e switch-case) mostraram-se eficazes para os requisitos propostos.

Como trabalhos futuros, sugere-se a integração com persistência em arquivos, implementação de interface gráfica e adição de funcionalidades de backup e recuperação.

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.