ENIAC
Centro Universitário de Excelência

SISTEMAS DE INFORMAÇÃO

DAVID MARTINS DOS SANTOS - 240902022

PORTFÓLIO LINGUAGEM C AVANÇADO

Guarulhos
2024

DAVID MARTINS DOS SANTOS

PORTFÓLIO
PORTFÓLIO LINGUAGEM C AVANÇADO

Trabalho apresentado ao Curso Sistemas de informação do Centro Universitário ENIAC para a disciplina PORTFÓLIO LINGUAGEM C AVANÇADO.

Prof. NELSON LUZETTI CRIADO

Guarulhos

2024

ENIAC
Centro Universitário de Excelência

Algoritmo de Ordenação: Bubble Sort

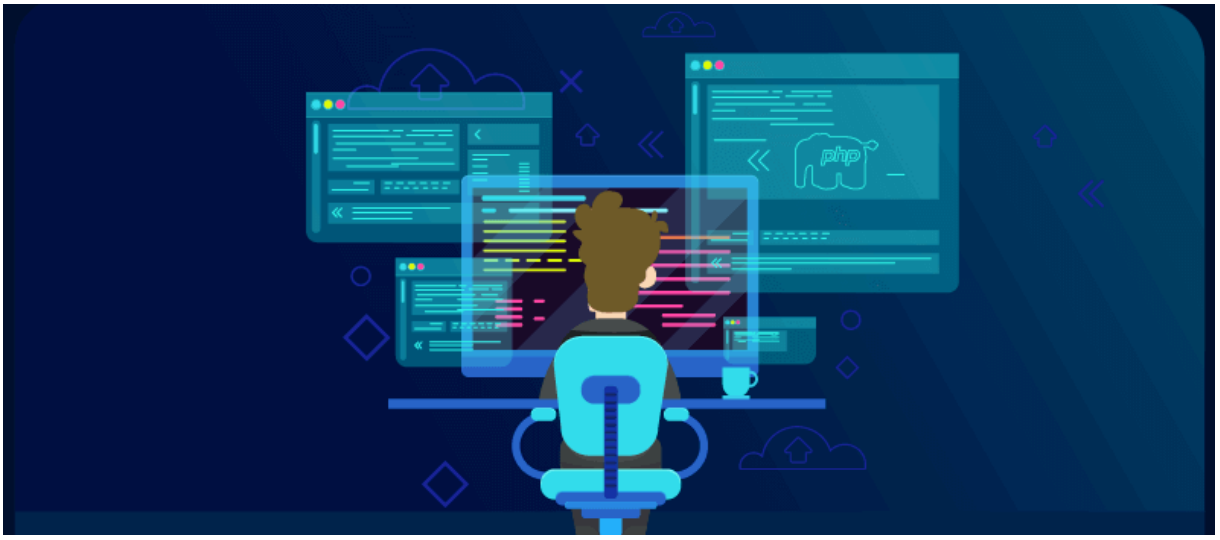
O algoritmo de ordenação Bubble Sort é um dos mais simples algoritmos de ordenação. Ele funciona percorrendo repetidamente a lista, comparando elementos adjacentes e trocando-os se estiverem na ordem errada. O processo é repetido até que nenhuma troca seja necessária, o que indica que a lista está ordenada.

A complexidade do Bubble Sort é de O(n^2), onde "n" representa o número de elementos na lista a ser ordenada. Isso significa que o tempo de execução do algoritmo cresce quadraticamente com o tamanho da entrada. A justificativa dessa complexidade se baseia no fato de que o algoritmo tem dois loops aninhados: um para percorrer a lista e outro para comparar os elementos adjacentes. Como cada elemento precisa ser comparado com todos os outros, o número total de comparações é proporcional a n * (n-1), que é aproximadamente n^2 para grandes valores de n.

Ponteiros em Linguagem C

Em linguagem C, um ponteiro é uma variável que contém o endereço de memória de outra variável. Eles são amplamente utilizados para manipulação de dados, alocação dinâmica de memória e passagem de parâmetros para funções.

Aqui está um exemplo simples de como os ponteiros são utilizados em C:



```
 include <stdio.h>

int main
 int       10
 int


 printf "O valor de num é: %d\n"
 printf "O valor de num acessado através do ponteiro é: %d\n"

 return 0
```

Alocação Estática e Alocação Dinâmica

A alocação estática ocorre em tempo de compilação e é feita quando o tamanho e a quantidade de memória necessários são conhecidos antes da execução do programa. Em C, isso é feito através da declaração de variáveis ou arrays.

Por exemplo:

```
int array 100
```

Já a alocação dinâmica ocorre em tempo de execução e é feita quando o tamanho e a quantidade de memória necessários só podem ser determinados durante a execução do programa. Em C, isso é feito usando as funções `malloc()`, `calloc()` ou

`realloc()` da biblioteca `<stdlib.h>`.

Po exemplo:

```
int
        int     malloc 100    sizeof int
```

Vantagens e Desvantagens:

- ☐ Alocação Estática:
    a. Vantagens:
        - ☐ Simplicidade: A alocação estática é simples e direta.
        - ☐ Desempenho: Acesso rápido à memória.
    b. Desvantagens:
        - ☐ Limitações de tamanho: O tamanho do array precisa ser conhecido em tempo de compilação e não pode ser alterado durante a execução do programa.
        - ☐ Inflexibilidade: Não é possível liberar a memória alocada durante a execução do programa.
- ☐ Alocação Dinâmica:
    a. Vantagens:
        - ☐ Flexibilidade: O tamanho e a quantidade de memória podem ser determinados durante a execução do programa.
        - ☐ Eficiência no uso da memória: A memória é alocada apenas quando necessária.
    b. Desvantagens:
        - ☐ Gerenciamento manual: A responsabilidade de liberar a memória alocada é do programador, o que pode levar a vazamentos de memória.
        - ☐ Possibilidade de fragmentação de memória: Se não for feita uma gestão cuidadosa da alocação e liberação de memória, pode ocorrer fragmentação de memória, diminuindo a eficiência do programa.

Conclusão

Durante a realização deste trabalho, pude aprofundar meu conhecimento sobre o algoritmo de ordenação Bubble Sort, ponteiros em linguagem C e os conceitos de alocação estática e dinâmica de memória. Foi interessante perceber como esses conceitos fundamentais são aplicados no desenvolvimento de software e como cada um deles tem suas vantagens e desvantagens.

Em particular, compreendi melhor a importância da eficiência na escolha de algoritmos de ordenação, considerando sua complexidade e desempenho. Além disso, percebi a flexibilidade que os ponteiros proporcionam em C, permitindo manipular diretamente a memória do programa. Quanto à alocação de memória, ficou claro que a escolha entre estática e dinâmica depende das necessidades específicas do programa e das características do problema a ser resolvido.

No geral, este trabalho contribuiu significativamente para minha compreensão dos fundamentos da programação em C e dos conceitos essenciais para o desenvolvimento de software eficiente e robusto.