Serviço Público Federal
Ministério da Educação
Fundação Universidade Federal de Mato Grosso do Sul

Nome: Raul Leão Chagas
RGA: 2021.1906.064-6’
Professora: Daniela Luiza Catelan

Cocktail Sort

1. Introdução

Este relatório tem como objetivo detalhar o funcionamento e o processo de desenvolvimento do programa em MARIE que realiza a ordenação de uma lista de números inteiros utilizando o algoritmo Cocktail Sort.

2. Explicação do funcionamento do Cocktail sort

O código implementa o algoritmo Cocktail Sort, que ordena a lista de forma unidirecional. Ele recebe o número de elementos e os valores da lista como entrada, realiza a ordenação e exibe a lista ordenada.

- **Entrada de Dados:** O programa solicita ao usuário a quantidade de elementos e os valores da lista.
- **Inicialização:** Variáveis necessárias são inicializadas, incluindo um contador e um indicador de troca realizada.
- **Loop de Ordenação:** O programa alterna passadas de comparação e troca, movendo os maiores elementos para o final e os menores para o início da lista.
- **Verificação de Trocas:** Se nenhuma troca for realizada em uma passada, a ordenação é considerada concluída.
- **Saída do Resultado:** A lista ordenada é exibida na tela.

3. Explicação e Organização do Código

- **Descrição das Variáveis:**
- **Entrada:**
  - numElementos: Número total de elementos da lista.
  - Lista de valores: Valores dos elementos da lista.
- **Saída:**
  - Lista ordenada: A lista de elementos em ordem crescente.

Clareza de comentário:

O código foi comentado linha a linha informando o que acontece ao decorrer da execução do código, veja os exemplos a seguir:

```
/ Quantidade de iterações
input_usuario,Input                    / Solicita ao usuário a quantidade de elementos
a serem inseridos
    Store numElementos          / Armazena a quantidade de elementos em numElementos
    Store contador              / contador = numElementos
    Load inicioLista
    Store enderecoAtual         / enderecoAtual = inicioLista
```

```
loop_input, Load contador      / Início do loop de entrada
    Skipcond 800               / Se contador > 0, continua o loop
    Jump pre_ordenacao         / Se contador <= 0, termina o loop
    Subt UM
    Store contador             / contador = contador - 1
    Input                      / Solicita ao usuário o próximo valor
    Output
    StoreI enderecoAtual       / Armazena o valor na posição atual da lista
    Load enderecoAtual
    Add UM
    Store enderecoAtual        / enderecoAtual = enderecoAtual + 1
    Jump loop_input            / Repete o loop
```

Algoritmo:

O algoritmo segue parcialmente os passos do Cocktail Sort. Inicialmente, os elementos são inseridos na lista. Em seguida, a ordenação é realizada unidirecional, e finalmente a lista ordenada é impressa.

Funções, Parâmetros, Variáveis:

- numElementos: Número total de elementos a serem ordenados.
- trocaRealizada: Flag para verificar se trocas foram realizadas em uma passada.
- inicioLista: Endereço do primeiro elemento da lista.
- enderecoAtual: Endereço do elemento atual da lista sendo processado.
- proxEndereco: Endereço do próximo elemento a ser comparado.
- proxValor: Valor do próximo elemento a ser comparado.
- numComparacoes: Número de comparações restantes em uma passada.
- contador: Contador geral para controlar loops.
- ZERO: Constante 0.
- UM: Constante 1.

4. Resultados e Discussão:

O programa implementado calcula corretamente a ordenação da lista utilizando uma abordagem unidirecional do método de ordenação Cocktail Sort. Devido à complexidade e ao tempo disponível, não foi possível desenvolver o algoritmo de forma 100% bidirecional. Vale ressaltar que a implementação possui algumas limitações, quando o programa é executado, o algoritmo pede a quantidade de números que o usuário precisa digitar para realizar a ordenação, caso esse número seja zero ou outro número negativo, o programa quebra.

5. Dificuldades Encontradas:

Apesar de o código ter sido implementado parcialmente com sucesso, algumas dificuldades foram enfrentadas ao longo do processo. A falta de conhecimento em Assembly (MARIE) tornou a dificultou a implementação consideravelmente, pois a sintaxe e a lógica de controle em Assembly são bastante diferentes das linguagens de programação de alto nível (Java, Python, JavaScript). Além disso, encontrar recursos e material de estudo sobre MARIE foi difícil, já que há menos documentação e tutoriais disponíveis para MARIE em comparação com linguagens de programação mais populares, como JavaScript, Java e Python. A complexidade da linguagem Assembly também

contribuiu para as dificuldades enfrentadas, já que a programação em Assembly é mais complexa do que em linguagens de alto nível. Em continuidade, durante a implementação, foram encontradas várias dificuldades, incluindo a manipulação de endereços de memória, que se mostrou desafiadora ao controlar os endereços para acessar e trocar elementos corretamente. Também foi difícil garantir que os loops alternassem corretamente entre passadas e atualizassem as variáveis conforme esperado.

6. Soluções de Implementação:

Para superar essas dificuldades, foram adotadas as seguintes abordagens: a utilização de variáveis auxiliares para armazenar endereços e valores intermediários, a estruturação clara dos loops com definição das condições de saída para garantir o funcionamento correto, e a realização de testes contínuos para validar cada passo do algoritmo e assegurar a execução correta e a atualização das variáveis.

7. Conclusão:

Este relatório fornece uma análise detalhada da implementação do algoritmo de ordenação Cocktail Sort. O código foi explicado em termos de seu funcionamento, estrutura, variáveis e algoritmo.