Trabalho Prático 01: Manipulação e Organização de Arquivos de Dados
Disciplina: Algoritmos e Estruturas de Dados II

Aluno: Thiago Ker Gama Nunes Carvalho
Matrícula: 23.2.8019

Roteiro de implementação e suas soluções:

1. Geração dos dados
2. Definição dos parâmetros de armazenamento
3. Simulação de escrita
4. Cálculo de estatísticas
5. Exibição dos resultados

1. Geração dos Dados

Para a criação dos registros, utilizei:

- **Biblioteca Faker**: responsável por gerar dados aleatórios baseados em informações reais do contexto brasileiro (nomes, CPFs etc.).
- **Dataclass:** utilizada para estruturar a classe Pessoa, aproximando-se da ideia de struct em C, permitindo um armazenamento organizado e tipado.
- **Biblioteca Random**: utilizada para sortear matrículas, anos de ingresso e coeficientes acadêmicos.
  O código utiliza o encoding latin-1 e o locale pt_BR da biblioteca Faker para garantir compatibilidade na conversão de strings para bytes, evitando caracteres inválidos durante a escrita no arquivo binário.

  A classe Pessoa concentra todos os campos necessários, e cada atributo já é formatado com limites adequados (ex.: nome com até 50 caracteres), garantindo que, ao gerar os registros, nenhum campo ultrapasse o espaço destinado a ele.

2. Entrada de Dados do Usuário

O programa solicita três informações principais:

- Quantidade de registros a serem gerados;
- Modo de armazenamento:
  - (1) Tamanho fixo,
  - (2) Tamanho variável (contíguo ou espalhado);
- Tamanho máximo de cada bloco em bytes.

Além disso, são definidos marcadores especiais:
0xFE: Para marcar o fim de um registro,
0xFF: Para indicar continuação quando um registro variável ultrapassa o limite de um bloco.
Que são utilizados para a organização interna dos dados no arquivo .dat.

3. Simulação de Escrita dos Registros

A primeira etapa da escrita consiste em gerar o cabeçalho do arquivo, contendo o tamanho do bloco e o modo de armazenamento, convertido diretamente para bytes.

- **Modo 1 - Registros de Tamanho Fixo**

Nesse modo, cada registro possui exatamente 169 bytes.
Portanto:

  - O usuário deve escolher um tamanho de bloco no mínimo igual a 169 bytes
  - Caso contrário, o programa encerra com erro

Os registros são montados concatenando todos os campos formatados.
Quando um registro tem espaço sobrando no bloco, esse espaço é preenchido com o caractere #, garantindo que todos os blocos permaneçam uniformes.

A alocação nos blocos é feita sequencialmente:

  - Se o registro cabe no bloco atual, é adicionado
  - Caso contrário, o bloco é fechado e um novo é iniciado

- **Modo 2 - Registros de Tamanho Variável**

Nesse modo, o registro é convertido para uma string única separada por vírgulas e finalizada com o marcador FE.
O usuário pode escolher:

  - (1) Registros contíguos:
    Todo o registro deve caber inteiramente dentro de um único bloco.
    Caso ultrapasse o tamanho máximo, o programa informa o erro.
  - (2) Registros espalhados:
    O registro pode ser fragmentado entre vários blocos.
    Quando o espaço do bloco acaba:
    - Grava-se o trecho possível;
    - Adiciona-se o marcador de continuação FF;
    - O restante vai para o próximo bloco.

Esse processo garante que o arquivo possa armazenar registros maiores do que o próprio bloco.

4. **Cálculo das Estatísticas**
   Após concluir a escrita dos blocos no arquivo .dat, o programa inicia a etapa de análise e geração das estatísticas.

   Os principais cálculos realizados são:
   • Total de blocos utilizados
   É simplesmente o número de blocos que foram preenchidos durante a etapa de armazenamento.
   Esse valor indica o espaço efetivamente ocupado no arquivo, considerando o limite de bytes por bloco.

- **Cálculo dos bytes utilizados**
  - **Modo Fixo:**
    Como cada registro possui exatamente 169 bytes, os bytes úteis são obtidos diretamente pelo produto:
    quantidade_registros × 169.
  - **Modo Variável:**
    O programa recalcula o tamanho real de cada registro (sem os marcadores), somando o tamanho de todos os campos codificados.
    Esse valor representa os *bytes verdadeiramente úteis*, desconsiderando fragmentação ou preenchimento.

   • Ocupação percentual dos blocos
   Para cada bloco, é calculado:
   Ocupação = (tamanho do bloco preenchido / tamanho máximo do bloco) * 100

   Essa métrica permite observar quão cheio cada bloco ficou e identificar possíveis perdas de espaço.

   • Percentual médio de ocupação
   É a média das ocupações individuais dos blocos, permitindo avaliar o nível geral de otimização do armazenamento.
   • Blocos parcialmente utilizados
   Conta quantos blocos não atingiram 100% da capacidade.
   Esse dado é importante principalmente no modo de registros variáveis ou nos últimos blocos, indicando possíveis fragmentações ou sobras.
   • Eficiência de armazenamento
   A eficiência é calculada comparando:

   Eficiência = (bytes úteis / bytes totais disponíveis) * 100

   onde:
- Bytes úteis = tamanho real dos dados;
- Bytes totais disponíveis = número de blocos × tamanho máximo do bloco.

Esse valor mostra quanto do espaço total foi efetivamente utilizado para armazenar informação relevante, desconsiderando fragmentação ou preenchimento.

5. Exibição dos Resultados

- Total de blocos utilizados
- Quantidade de blocos parcialmente ocupados
- Percentual médio de ocupação dos blocos
- Eficiência geral de armazenamento

Github: https://github.com/ThiagoKer/tp01_aeds2