Trabalho Prático 1 - Múltiplas Ordenações

1 Descrição do problema

Você foi contratado pela organização Xulambs para melhorar a infraestrutura de dados da entidade, em particular o cadastro de atendidos pela organização. Cada cadastro é composto de um grande número de informações. O problema que você tem que resolver é gerar 3 listas ordenadas das pessoas atendidas, onde a chave de ordenação de cada lista deve atender a um critério, pro exemplo nome, CPF e endereço.
Assim, o caso de uso a ser implementado é, a partir de uma base de dados xCSV a ser lida de um arquivo, deve ser construída a estrutura de dados para ordenação indireta para cada uma das 3 chaves, realizada a ordenação indireta por cada uma das 3 chaves e imprimir a base de dados ordenada pela chave.
Por exemplo, dada a base de dados de entrada apresentada na Tabela 1, o seu programa deverá gerar as saídas das Tabelas 2, 3 e 4, usando como chave de ordenação Nome, CPF e End, respectivamente.

1.1 Tipo Abstrato de Dados

O TAD OrdenacaoIndireta deve implementar ordenação indireta e permitir ordenações por múltiplas chaves. A estrutura de dados deve suportar ordenação indireta e múltiplas chaves. A Figura 1 ilustra a estrutura de dados a ser especificada e implementada utilizando alocação dinâmica e todos os componentes em memória.
O código a seguir apresenta um exemplo de TAD e sua utilização.

```
typedef struct OrdInd{
 // a definir
} OrdInd_t, *OrdInd_ptr;
59
// Exemplos de funcoes
OrdInd_ptr Cria();
int Destroi (OrdInd_ptr poi);
int CarregaArquivo(OrdInd_ptr poi, char * nomeentrada);
int NumAtributos(OrdInd_ptr poi);
int NomeAtributo(OrdInd_ptr poi, int pos, char * nome);
int CriaIndice (OrdInd_ptr poi, int atribid);
int OrdenaIndice (OrdInd_ptr poi, int atribid);
int ImprimeOrdenadoIndice (OrdInd_ptr poi, int atribid);

int main(int argc, char ** argv){
   char aux[100];
   OrdInd_ptr poi = Cria();
   CarregaArquivo(poi,"entrada.xcsv");
   int numatrib = NumAtributos(poi);
   for (int i = 0; i<numatrib; i++){
     if (NomeAtributo(poi,i,aux)>0){
       if (!strcmp(aux,"Nome")||!strcmp(aux,"CPF")||!strcmp(aux,"End")){
         CriaIndice(poi,i);
         OrdenaIndice (poi,i);
         ImprimeOrdenadoIndice (poi,i);
       }
     }
   }
   Destroi(poi);
}
```

Nome	CPF	End	Outros
Bob	444	Eco St	lkhkjghglkjhl
Chris	333	Alpha St	mnbmbmkhgjgh
Alice	222	Delta St	ituwriquerquiy
Ester	111	Charlie St	yigjhfuyfu
Daniel	555	Beta St	hgdasjhgaeyt

Tabela 1: Base de dados de entrada

Nome	CPF	End	Outros
Alice	222	Delta St	ituwriquerquiy
Bob	444	Eco St	lkhkjghglkjhl
Chris	333	Alpha St	mnbmbmkhgjgh
Daniel	555	Beta St	hgdasjhgaeyt
Ester	111	Charlie St	yigjhfuyfu

Tabela 2: Base de dados ordenada por nome

1.2 Algoritmos de Ordenação

Deverão ser implementados pelo menos 3 algoritmos de ordenação, sendo um deles o QuickSort.

1.3 Avaliação experimental comparativa das versões do aplicativo

Uma vez implementado, a avaliação experimental deve avaliar tanto o seu desempenho quanto a sua localidade de referência. O aspecto desempenho deve ser medido através do tempo execução como um todo e de cada uma das etapas e diferentes critérios de ordenação.
O aspecto localidade de referência deve ser avaliado por métricas como distância de pilha, além de incluir discussões sobre a eficiência da estrutura de dados em executar as várias funções.
Deve ser elaborado um plano experimental que exercite variações nessas dimensões e permita observar os impactos das variações nos valores. A avaliação deve incluir uma discussão dos resultados.
Para facilitar os experimentos serão providos arquivos gerados sinteticamente.
Em termos da carga de trabalho a ser utilizada, devem ser consideradas as seguintes dimensões:

- tamanho da entrada
- tamnho do registro
- algoritmos de ordenação
- natureza da entrada (ordenada, inversamente ordenada, aleatória)

Nome	CPF	End	Outros
Ester	111	Charlie St	yigjhfuyfu
Alice	222	Delta St	ituwriquerquiy
Chris	333	Alpha St	mnbmbmkhgjgh
Bob	444	Eco St	lkhkjghglkjhl
Daniel	555	Beta St	hgdasjhgaeyt

Tabela 3: Base de dados ordenada por CPF

Nome	CPF	End	Outros
Chris	333	Alpha St	mnbmbmkhgjgh
Daniel	555	Beta St	hgdasjhgaeyt
Ester	111	Charlie St	yigjhfuyfu
Alice	222	Delta St	ituwriquerquiy
Bob	444	Eco St	lkhkjghglkjhl

Tabela 4: Base de dados ordenada por endereço

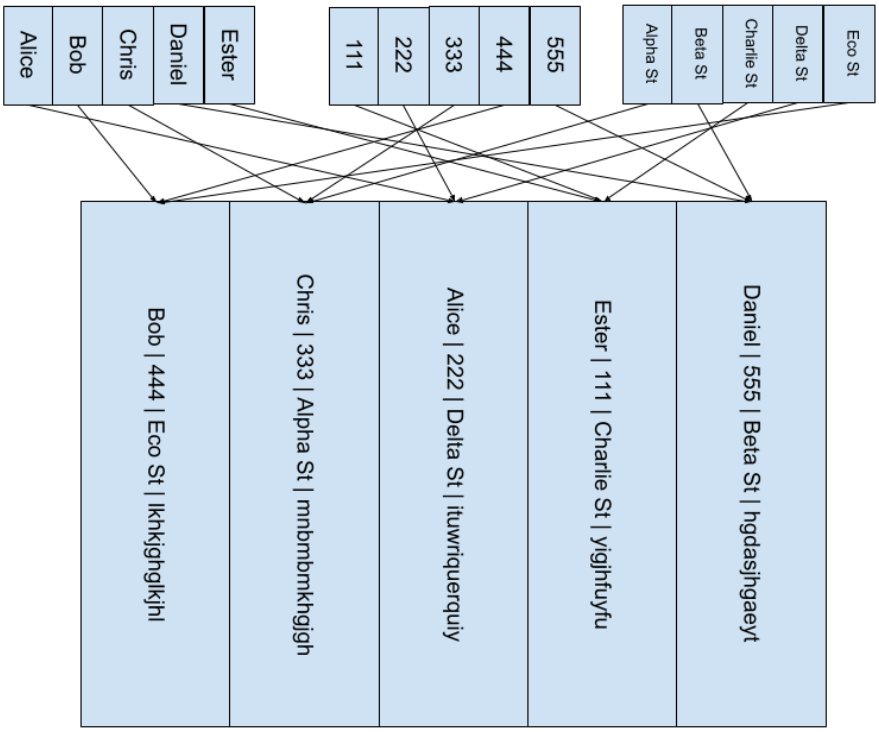

Figura 1: Exemplo de ordenação indireta por múltiplas chaves

1.4 Opcional: Heurística para maximizar a localidade de referência

Um problema da ordenação indireta é que, como as diferentes chaves induzem a diferentes ordenações dos registros. Como o princípio da ordenação indireta é evitar movimentações, é natural que a localidade de referência da impressão do arquivo todo seja ruim.
A parte opcional do TP1 consiste em propor, implementar e avaliar uma heurística que melhore a localidade de referência quando das impressões da base de dados ordenada por diferentes chaves. Você pode considerar que as impressões vão acontecer após as ordenações indiretas e usar as várias relações de precedência para propor uma reorganização da base de dados de forma que a localidade de referência melhore.
A descrição da heurística, sua implementação e avaliação devem ser entregues como um anexo da documentação do seu TP (ou seja, não contam no limite de páginas).

2 Como será feita a entrega

2.1 Submissão

A entrega do TP1 compreende duas submissões:

VPL TP1: Submissão do código a ser submetido até **28/11, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). **Submissões em atraso terão desconto.** Detalhes sobre a submissão do código são apresentados na Seção 2.3.

Relatório TP1: Arquivo PDF contendo a documentação do TP, assim como a avaliação experimental, conforme instruções, a ser submetido até **28/11, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). Não vão ser aceitas submissão em atraso. Detalhes sobre a submissão de relatório são apresentados na Seção 2.2.

2.2 Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
7. **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
9. Número máximo de páginas incluindo a capa: 10

A documentação deve conter a desdos crição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

A documentação deverá ser entregue como uma atividade separada designada para tal no minha.ufmg. A entrega deve ser um arquivo `.pdf`, nomeado `nome_sobrenome_matricula.pdf`, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

2.3 Código

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim as pastas **bin** e **obj** devem estar vazias. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp3.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no moodle. Você também terá à disposição uma VPL de testes para verificar se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não vale pontos e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no Moodle.

3 Avaliação

- Corretude na execução dos casos de teste - (30% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Conteúdo segundo modelo proposto na seção **Documentação** - (20% da nota total)
- Definição e implementação das estruturas de dados e funções - (15% da nota total)
- Apresentação da análise de complexidade das implementações - (5% da nota total)
- Análise experimental - (15% da nota total)
- Aderência às instruções de entrega - (5% da nota total)

Se o programa submetido **não compilar**[1], seu trabalho não será avaliado e sua nota será **0**. Trabalhos entregues com atraso sofrerão penalização de 2^{d-1} pontos, com d = dias úteis de atraso.

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

4 Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

5 FAQ (Frequently asked Questions)

1. Posso utilizar qualquer versão do C++? NÃO, o corretor da VPL utiliza C++11.
2. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM, porém lembre-se que a correção é feita sob o sistema Linux, então certifique-se que seu trabalho está funcional em Linux.
3. Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO.
4. Posso utilizar smart pointers? NÃO.
5. Posso utilizar o tipo String? SIM.
6. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO.
7. Posso utilizar alguma biblioteca para tratar exceções? SIM.
8. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
9. As análises e apresentação dos resultados são importantes na documentação? SIM.
10. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO.
11. Posso fazer o trabalho em dupla ou em grupo? NÃO.
12. Posso trocar informações com os colegas sobre a teoria? SIM.
13. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.