Trabalho Prático 15

Clientes

Algoritmos e Estruturas de Dados I

Prazo: **18** de **novembro** de 2017

1 Introdução

A clientela da papelaria de Pedro está crescendo, por isso ele deseja simplificar a busca por usuários em sua papelaria. O problema é que a base de dados de seus clientes está organizada por ordem de aparição, sem a opção de ordenar por algum campo.

Depois de pesquisar na ferramenta Poodle, Pedro ficou sabendo que existe um método de ordenação muito simples, conhecido por ordenação por seleção[1]. Nele, dado um arranjo A de tamanho t_A, o algoritmo caminha da posição 0 até $t_A - 1$. Para cada posição p do arranjo, ele seleciona o menor item do intervalo de p até $T_A - 1$ e o troca com que está na posição p, como mostra a Figura 1. Nela, a posição p é mostrada pela seta e a posição do menor valor escolhido está em vermelho.

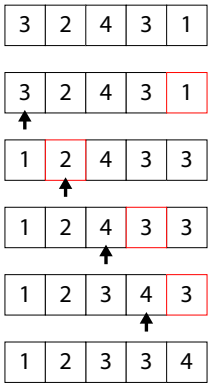

Figura 1: Ordenação por seleção

[1] `https://youtu.be/BSXIolKg5F8?t=25s`

2 Tarefa

Pedro quer que você crie um programa que, dada a lista dos clientes, faça a ordenação por seleção pelo campo requisitado e imprima a lista ordenada.

3 Entrada

A primeira linha da entrada é um número inteiro `N` (`N` nunca será maior que 1000) que indica quantos usuários estão na lista. As próximas `N` linhas são formadas de três *strings* no formato $\texttt{U}_\texttt{p}$ $\texttt{U}_\texttt{s}$ `E`, onde $\texttt{U}_\texttt{p}$ e $\texttt{U}_\texttt{s}$ são o primeiro nome e o sobrenome do usuário. Ambos conterão apenas os caracteres de `a` a `z` (apenas minúsculos) e terão no máximo 20 caracteres. O campo `E` representa o e-mail do usuário. Ele não conterá espaços, mas poderá conter números e caracteres especiais como `"."` (ponto final), `"-"` (hífen), `"_"` (*underscore*) e `"@"` (arroba). O e-mail terá no máximo 100 caracteres. A última linha é um número inteiro que define qual campo deverá ser utilizado para ordenar, onde 0 indica primeiro nome, 1 indica sobrenome e 2 indica e-mail.

4 Saída

A saída deverá ser os cadastros no mesmo formato da entrada ($\texttt{U}_\texttt{p}$ $\texttt{U}_\texttt{s}$ `E`), um por linha, ordenados pelo campo selecionado. Na dúvida, siga o exemplo abaixo.

5 Exemplo

Entrada

```
6
ella dowd ella.dowd.123@email.com
adam pullman pullman_adam@email.com
kylie dowd email-kylie-dowd@server.com
sophie langdon i.am.sophie.langdon@mail.com
alan hill alan.hill@email.com
2
```

Saída

```
alan hill alan.hill@email.com
ella dowd ella.dowd.123@email.com
kylie dowd email-kylie-dowd@server.com
sophie langdon i.am.sophie.langdon@mail.com
adam pullman pullman_adam@email.com
```

Outros exemplos estão disponíveis na página deste trabalho no Moodle.

6 Avisos

Avisos mandatórios para o envio do trabalho:

- O código deve ser escrito em linguagem `C`
- **Você deverá utilizar a função `strcmp` (disponível no cabeçalho `string.h` padrão da linguagem `C`) para comparar duas *strings***
- As entradas que serão utilizadas para teste não conterão erros, então não será necessário testar a validade das mesmas
- Não utilize chamadas da função `system` (por exemplo, `system("pause")`) pois essas podem variar de acordo com o sistema operacional e os programas instalados da máquina onde o programa está rodando
- Deixe seu código bem comentado para facilitar a correção
- Não utilize espaços ou caracteres especiais nos nomes dos arquivos. Utilize apenas os caracteres de `A` a `Z` (tanto maiúsculas como minúsculas) sem acento, os números de `0` a `9` e os caracteres `-` (hífen), _ (*underscore*) e `.` (ponto final)
- Utilize a extensão `.c` para arquivos de código e `.h` para arquivos de cabeçalho, quando aplicar
- Se for submeter os arquivos via upload, não envie um arquivo comprimido (por exemplo, `.zip`, `.rar`, etc.). Utilize os diversos campos da aba "Submissão", um para cada arquivo
- Envie apenas os arquivos `.c` e `.h`
- Não copie o trabalho de algum colega ou baixe da internet. Lembre-se que o prejudicado será você pois o aprendizado obtido nessa disciplina será utilizado durante diversas outras etapas do seu curso

7 Dicas

Dicas importantes para o desenvolvimento deste trabalho:

- Utilize a função `scanf` para ler as entradas do usuário
- Utilize a função `printf` para imprimir os resultados das operações
- Note que `scanf("%s", nome)` lê um arranjo de caracteres (*string*) para a variável `nome`
- A função `strcmp` compara duas *strings* e retorna um número c, onde, se $c < 0$, a primeira *string* é menor alfabeticamente, se $c = 0$, as duas *strings* são iguais e, se $c > 0$, a primeira *string* é maior alfabeticamente

- Recomendo usar um `struct` para guardar os usuários e criar uma função que, dado o `struct` e um campo, retorna a *string* nesse campo na `struct`. Outra possibilidade é utilizar matrizes de *strings*, o que pode ser um pouco mais complicado
- Qualquer dúvida que tiver, utilize o fórum de dúvidas no Moodle. Inicie o assunto do tópico com a tag `[TP15]`
- Caso prefira, participe das monitorias toda quarta das 17h às 18h na sala 2012

8 Checklist

Checklist não exaustiva de passos até a entrega do trabalho:

1. Pesquise o funcionamento das funções citadas acima para facilitar o uso
2. Implemente e compile o programa
3. Teste para o exemplo dado acima. Compare as saídas para garantir o funcionamento correto
4. Faça o mesmo do item anterior para os exemplos disponibilizados na página deste trabalho no Moodle
5. Envie o trabalho pelo Moodle, onde ele será testado automaticamente para todos os casos disponíveis
6. Caso algum teste dê errado, volte ao passo 2

Bom trabalho e divirta-se!