Projeto: Agenda
Tópicos de Programação

Entrega: 04/02/2016

Como projeto do Curso de Versão de Tópicos de Programação de 2016, vamos fazer uma agenda telefônica. Essa agenda será utilizada via terminal, com comandos como "`novo`", "`listar`", "`apagar`", etc. Por exemplo:

```
> novo
Nome: Diogo
Sobrenome: Kykuta
Endereco: Rua do Matao, 1010
Telefone: nao informado
e-mail: haruki@ime.usp.br
Salvo!

> novo
Nome: Caio
Sobrenome: Braz
Endereco: Rua do Matao, 1010
Telefone: (11) 9 9999 9999
e-mail: nao informado
Salvo!

> listar
Nome  | Sobrenome  | Endereco           | Telefone         | e-mail
Diogo | Kykuta     | Rua do Matao, 1010 | nao informado    | haruki@ime.usp.br
Caio  | Braz       | Rua do Matao, 1010 | (11) 9 9999 9999 | nao informado

> apagar nome
Nome: Diego
Nao existe ninguem com nome Diego

> apagar sobrenome
Sobrenome: Kykuta
Apagado

> listar
Nome  | Sobrenome  | Endereco           | Telefone         | e-mail
Caio  | Braz       | Rua do Matao, 1010 | (11) 9 9999 9999 | nao informado

> sair
```

O reconhecimento dos comandos já está implementado no arquivo `main.c` fornecido. Esses comandos estão mapeados para chamar uma certa função dentro desse arquivo.

Por exemplo, ao usar o comando "`apagar`" da agenda, será chamada a função `deleteEntry`.

Todas essas funções possuem comentários "`// ME COMPLETE!`" para indicar onde você deve mexer, minimamente, para fazer o projeto. Elas recebem como parâmetro um parâmetro `ag` do tipo `agenda`, esse tipo `agenda` será criado por vocês, mas mais para frente no enunciado tem explicações mais detalhadas. O `main()` desse arquivo é complicado mesmo, não se preocupe! Você não vai precisar mexer nele.

1 Tarefas

Para facilitar um pouco, seguem algumas tarefas, organizadas em uma ordem que vai te permitir fazer o projeto durante o andamento do curso:

1.1 Entrada da agenda

Precisamos criar uma `struct` para armazenar os dados de uma entrada da agenda e precisamos de uma função para imprimir uma linha bonita.

Dica: O `"%s"` pode ser modificado para forçar usar uma certa quantidade de caracteres. Alem disso, pode ser alinhado a esquerda e a direita:

```
printf("%s %s\n", "oi", "tchau")
oi\n
printf("%10s %10s\n", "oi", "tchau")
        oi     tchau\n
printf("%-10s %-10s\n", "oi", "tchau")
oi        tchau     \n
```

Com essa parte feita, dá pra fazer (de modo incompleto) o `novo`, por exemplo

1.2 Lista ligada

Para armazenar as diversas entradas da agenda, usaremos uma lista ligada. Sim, precisa ser lista ligada

Com isso, já podemos começar a preencher várias outras funções.

1.3 Agenda

Já podemos criar a `struct` da agenda agora! Além disso, devemos criar uma agenda para usarmos nas funções. Faça isso na função `newAgenda`, que também está marcada para ser completada. Ela já está sendo usada, basta completar.

1.4 Ordenação

Oras, tá difícil procurar com tudo bagunçado! Quero as entradas ordenadas, normalmente, por nome. Então modifique sua lista ligada para manter as entradas sempre ordenadas por nome!

1.5 Ordenação 2

Apesar disso, as vezes posso querer por outro campo, como por exemplo, por sobrenome, pelo email, pela data de nascimento, ... data de nascimento!?!? Mas não tinha isso antes!! então modifique a função `listar` para ordenar por algum campo. Como eu faço isso?!?

Podemos pegar o primeiro argumento passado (já vou explicar melhor isso) para a função `listar` (assim como pra `apagar` podemos fazer isso), por exemplo:
`listar sobrenome`
Lista ordenando por sobrenome
`listar email`
Lista ordenando por email

2 As funções

As funções que já estão criadas recebem vários argumentos:

`char* cmd`

Esse é o nome do comando digitado (por exemplo `"novo"`, `"listar"`, ...)

`int argc` e `char** argv`

`argc` é o número de argumentos passados para o comando e `argv` é o vetor desses argumentos. `argv` tem exatamente `argc` elementos. Por exemplo:

`novo`

Possui `argc = 0` e `argv = {}`

`listar sobrenome`

Possui `argc = 1` e `argv = {"sobrenome"}`

`apagar nome sobrenome`

Possui `argc = 2` e `argv = {"nome", "sobrenome"}`

Dica: Na dúvida, coloque um `printf` no começo de um dos comandos para você poder entender o que está acontecendo.

3 Partes Obrigatórias

É obrigatório que seu programa seja feito usando lista ligada e apresente as funcionalidades acima.

3,0 Funcionalidades (não importa como), sendo
- 1,0 `novo`
- 1,0 `apagar`
- 1,0 `listar`

1,0 struct para armazenar as informações da agenda

1,0 struct da agenda

3,0 lista ligada, sendo
- 1,0 struct da celula da lista ligada

1,0 inserção

1,0 remoção

1,0 lista ligada mantida ordenada

1,0 ordenação na listagem por outro campo

Isso soma 10,0, mas mesmo assim isso não garante que você vai tirar nota máxima. Teremos descontos, alguns exemplos de descontos estão listados abaixo, mas não é a lista completa de descontos (ainda):

-0,5 cada malloc sem free

até -2,0 por indentação errada

até -2.0 por código ilegível

4 Partes Opcionais

Algumas sugestões de como continuar o projeto. Não vale nota, mas você pode deixar o monitor tão feliz que ele pode esquecer de olhar algum desconto ;)

- Faça salvar a agenda em disco, e abra o arquivo lido
- Adicione a funcionalidade de pesquisar um único contato a partir do nome
- Invente novas funcionalidades para a agenda!