OBJETIVO

Relembrar os conceitos básicos de estruturas de dados e ponteiros obtidos nas disciplinas de AED1 e AED2 através da implementação de uma lista encadeada simples que armazene as informações de um evento (`double time, int target, int type`). Este laboratório também introduzirá o padrão de entrada de dados usando arquivos que será usado nos próximos laboratórios.

QUESTÃO 1

Implemente uma estrutura chamada `evento_t` para armazenar um evento. Esta estrutura deverá ter os campos `tempo` (`double`), `alvo` (`int`) e `tipo` (`int`).

Implemente uma estrutura chamada `lista_eventos_t` para a lista encadeada. Esta estrutura deverá ter um campo para apontar para um `evento` (`evento_t*`) e um campo para apontar para o `proximo` evento (`lista_eventos_t*`). Uma lista nova/vazia irá apontar para `NULL`, portanto não há necessidade de se implementar uma função que crie uma nova lista. Implemente uma função para adicionar um evento no início da lista. Esta função deverá ter a seguinte interface:

```
bool lista_eventos_adicionar_inicio(evento_t *evento, lista_eventos_t **lista);
```

onde `*evento` é o evento a ser adicionado e `**lista` é um ponteiro para um ponteiro para a lista.

Implemente também uma função para listar os eventos de uma lista. Esta função deverá ter a seguinte interface:

```
void lista_eventos_listar(lista_eventos_t *lista);
```

Leia um arquivo de entrada cujo nome será passado pela linha de comando (`argv[1]`). Cada linha deste arquivo terá a seguinte sintaxe:

```
%3.6f\t%d\t%d\n
```

onde o primeiro valor (`%3.6f`) será o tempo de um evento, o segundo (`%d`) será o alvo e o terceiro (`%d`) será o tipo do evento. Tudo separado por tabs (`\t`) e seguido de uma linha nova (`\n`). Para cada linha, adicione o evento na lista (`lista_eventos_adicionar_inicio`). Após ler todo o arquivo de entrada, liste todos os eventos da lista criada (`lista_eventos_listar`).

Dicas:

- Para usar o tipo de dado `bool`, inclua o cabeçalho `stdbool.h` em seu código.
- Em todas as estruturas, use o `typedef` para definir a estrutura como um tipo. Exemplo:

```
struct evento_t { … };
typedef struct evento_t evento_t;
```

- Na função `lista_eventos_adicionar_inicio`, usamos um ponteiro para um ponteiro. Porque? R: Porque como o novo elemento estará no início da lista, ele será, na verdade, o (novo) ponteiro da lista. Para mudar o ponteiro de uma lista, é necessário passar o ponteiro dela como referência. Portanto, temos um ponteiro para ponteiro. Para acessar o primeiro elemento (e a lista), usamos `*lista`. Exemplo:

```
lista_eventos_t *item_novo = malloc(sizeof(lista_eventos_t));      // Aloca o novo item
if (item_novo == NULL) return false;                               // Falta memória?
item_novo->evento = evento;                                        // Seta o novo item
item_novo->proximo = *lista;                // O próximo do novo item será a lista
*lista = item_novo;                         // Aqui, estamos mudando o ponteiro da lista
return true;
```

- Para ler as linhas do arquivo de entrada, abra-o com o `fopen` e leia cada linha no formato especificado (`fscanf`) enquanto `feof(arquivo_entrada)` for falso.

QUESTÃO 2

Implemente uma função para adicionar um evento no final da lista. Esta função deverá ter a seguinte interface:

```
bool lista_eventos_adicionar_fim(evento_t *evento, lista_eventos_t **lista);
```

Leia um arquivo de entrada no mesmo formato que na Questão 1. A cada linha, execute a inserção no fim. Após ler todo o arquivo, liste todos os eventos.

Dicas:

- Se a `lista` for `NULL`, faça-a apontar para o novo item.
- Se não, percorra-a enquanto `item_atual->proximo != NULL`
- Aí, `item_atual->proximo = item_novo`

QUESTÃO 3

Implemente uma função para adicionar um evento de forma ordenada (crescente) na lista (que estará ordenada) de acordo com o campo `tempo` da estrutura `evento_t`. Esta função deverá ter a seguinte interface:

```
bool lista_eventos_adicionar_ordenado(evento_t *evento, lista_eventos_t **lista);
```

Leia um arquivo de entrada no mesmo formato que na Questão 1. A cada linha, execute a inserção ordenada. Após ler todo o arquivo, liste todos os eventos.

Dicas:

- Se a `lista` for `NULL`, faça-a apontar para o novo item.
- Se o tempo do novo evento for menor do que o tempo do *primeiro* item da lista, adicione o item no início da lista.
- Se não, percorra a lista enquanto
  `item_atual->proximo != NULL && item_atual->proximo->evento->tempo < evento->tempo`
- Adicione como próximo do `item_atual`.

ENTREGA DO LABORATÓRIO

Envie, até 15/04/21 às 23:59, o código-fonte para horacio@icomp.ufam.edu.br com o assunto “Entrega do 1o Laboratório de LPA”.