Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Verão de 2020/2021

Série de Exercícios 3

Construa os programas e a biblioteca indicados na Parte II, usando a linguagem C, tendo o cuidado de eliminar repetições no código fonte e de isolar funcionalidades distintas em diferentes ficheiros fonte. Entregue o código desenvolvido, devidamente indentado e comentado, bem como os **`Makefile`** para gerar os executáveis e as bibliotecas a partir do código fonte. Assegure-se de que o compilador não emite qualquer aviso sobre o seu código com a opção **`-Wall`** activa, e de que no final da execução do programa não existem recursos por libertar (memória alocada dinamicamente e ficheiros abertos). Deve verificar essa situação com o utilitário Valgrind. Em caso de erro irrecuperável, o programa não deve terminar descontroladamente, deve, no mínimo, emitir uma mensagem informativa e em seguida terminar. O código desenvolvido será valorizado segundo os seguintes critérios, em importância decrescente: correção, eficiência, clareza.

Encoraja-se a discussão dos problemas e das respectivas soluções com outros colegas (tenha em consideração que a partilha directa de soluções implica, no mínimo, a anulação das entregas dos alunos envolvidos).

O produto final desta série de exercícios é uma biblioteca que implementa serviços de consulta de informação sobre futebol, um programa que utiliza essa biblioteca e um *shared object* que acrescenta uma nova funcionalidade ao programa, na forma de *plug-in*. A informação futebolística pode ser obtida no serviço disponibilizado pelo site football-data.org, utilizando uma API Web. A informação sobre este serviço pode ser obtida a partir do site https://www.football-data.org.

Parte I - Preparação do ambiente de desenvolvimento

O acesso à informação usa o protocolo HTTP (*Hypertext Transfer Protocol*) para aceder aos serviços definidos por uma API REST (*Representational State Transfer*). É prática comum que os serviços suportados em API REST usem o paradigma pergunta/resposta. A pergunta é representada pelo URL (*Uniform Resource Locator*) que é usado no pedido HTTP GET enviado ao servidor; a resposta ao pedido é codificada no formato JSON (*JavaScript Object Notation*), de acordo com o esquema definido pela API.

A documentação da API a utilizar está disponível em https://openweathermap.org/api.

CURL

O acesso ao serviço é feito estabelecendo ligações ao servidor usando o protocolo HTTP. Para suportar as comunicações com o servidor deverá ser utilizada a biblioteca *open source* **`libcurl`**.

Instalação: **`$ sudo apt-get install libcurl4-gnutls-dev`**

Documentação: libcurl - the multiprotocol file transfer library.

JSON

Para interpretar as respostas do servidor em formato JSON deverá ser utilizada a biblioteca *open source* **`jansson`**.

Instalação: **`$ sudo apt-get install libjansson-dev`**

Documentação: Jansson Documentation

Tal como noutras linguagens, a indentação da escrita em formato JSON facilita a leitura direta por parte do humano. Sugere-se a utilização de um visualizador JSON, que pode ser instalado no *browser* na forma de *plug-in*.

Valgrind

Para verificar se um programa liberta toda a memória alocada dinamicamente deverá utilizar a ferramenta **`valgrind`**.

Instalação: **`$ sudo apt-get install valgrind`**

Documentação: **`$ man valgrind`**

Parte II - Realização

1. Utilizando as bibliotecas **`libcurl`** e **`jansson`**, implemente a função **`http_get_json_data`**, que realiza um pedido HTTP GET ao URL especificado através do parâmetro **`url`**, que deve corresponder a um recurso HTTP do tipo **`application/json`**, e contendo os cabeçalhos HTTP indicados no parâmetro **`request_headers`**. Retorna o ponteiro para uma instância do tipo **`json_t`** (definido no âmbito da biblioteca **`jansson`**) com o conteúdo da resposta. Se ocorrer um erro durante a transferência, a função retorna **`NULL`** e mostra em **`stderr`** a mensagem que indica a razão do erro. O parâmetro **`request_headers`** é um *array* de ponteiros para *strings* terminado pelo ponteiro **`NULL`**.

   ```
   json_t *http_get_json_data(const char *url, char *request_headers[]);
   ```

   Para teste utilize um programa que aceda ao recurso cujo URL é http://api.football-data.org/v2/competitions/ e afixe na consola as competições portuguesas.

2. Utilizando a função do ponto anterior, implemente as funções **`competitions_get`** , **`teams_get`** e **`matches_get`**.

   Utilize os seguintes tipos de dados para representar, respetivamente, uma competição; uma equipa; um jogo:

   ```
   typedef struct competition {
      int id;
      const char *name;
   } Competition;

   typedef struct team {
      int id;
      const char *name;
   } Team;

   typedef struct match {
      int id;
      const char *date;
      const char *status;
      const char *home_team, *away_team;
   } Match;
   ```

   A função **`competitions_get`** retorna informação sobre competições. Esta informação é obtida diretamente em **api.football-data.org/v2/competitions**.

   ```
   Competitions *competitions_get();
   ```

   A função **`teams_get`** retorna informação sobre as equipas que participam numa dada competição. Esta informação é obtida em **api.football-data.org/v2/competitions/{id}/teams/**, sendo **{id}** o identificador numérico da competição.

   ```
   Teams *teams_get(int competition_id);
   ```

   A função **`matches_get`** retorna informação sobre os jogos de uma competição. Esta informação é obtida em **api.football-data.org/v2/competitions/{id}/matches/**, sendo **{id}** o identificador numérico da competição.

```
Matches *matches_get(int competition_id);
```

Deve definir os tipos **`Competitions`**, **`Teams`** e **`Matches`** de modo a representarem a informação sobre todos os elementos do respetivo conjunto.

Na programação destas funções deve sempre libertar a memória alocada dinamicamente no momento em que esta deixe de ser utilizada.

Utilize um programa de teste que afixe na consola a lista das competições e a lista das equipas e dos jogos de uma dada competição. O identificador do campeonato nacional é 2017.

3. Construa uma biblioteca de ligação dinâmica (*shared object*) com as funções definidas nos pontos anteriores e com as funções auxiliares que entender necessárias. Na organização do código, tenha em consideração que deve evitar repetições de código fonte.

4. Desenvolva um programa que permanece em execução, aceitando e processando comandos para apresentação de informação na consola.

   O programa deve aceitar os seguintes comandos:

c	Listar as competições.
e {competition_id}	Listar as equipas participantes na competição especificada.
j {competition_id}	Listar os jogos da competição especificada.
s	Termina a execução do programa.

   Na construção do programa deve-se utilizar a biblioteca produzida no ponto anterior.

   Na execução de comandos posteriores deve-se reutilizar a informação adquirida em comandos anteriores, minimizando assim o número de pedidos ao servidor. Para retenção da informação deve criar em memória uma estrutura de dados adequada.

   Ao terminar a execução, o programa deve explicitamente libertar a memória alocada dinamicamente.

5. Modifique o programa anterior de modo que possam ser acrescentados novos comandos na forma de *plug-in*.

   Exemplifque a utilização dessa funcionalidade acrescentando um novo comando que lista apenas os jogos de uma dada equipa, para uma dada competição.

Data limite de entrega: 26 de Junho de 2021

ISEL, 30 de Maio de 2021