

Construa os programas e as bibliotecas 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 irreversível, 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.

A realização deste trabalho consiste na programação dos artefactos especificados ao longo das alíneas, destinados à construção de um programa de simulação de uma loja. O programa interage com o site [dummy JSON](https://dummyjson.com/), utilizando uma API web, trocando dados fictícios.

A entrega deve consistir num ficheiro em formato **zip** contendo uma diretoria como raiz, com seis diretorias no seu interior, uma por cada exercício. O conteúdo de cada diretoria deve ser composto pelo código fonte, *makefile* e eventuais bibliotecas necessárias para gerar o programa executável especificado nesse exercício.

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://dummyjson.com/docs>.

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](https://curl.se/libcurl/).

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](https://jansson.org/)

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`, que realiza um pedido HTTP GET ao URL especificado através do parâmetro `url`, que deve corresponder a um recurso HTTP do tipo `application/json`. 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.

```
json_t *http_get_json(const char *url);
```

Para teste realize um programa que acesse a informação do produto 1, cujo URL é <https://dummyjson.com/products/1> e afixe na consola alguma da informação obtida.

2. Utilizando as bibliotecas `libcurl` e `jansson`, implemente a função `http_post_json`, que realiza um pedido HTTP POST ao URL especificado através do parâmetro `url`. O parâmetro `data` é um ponteiro para uma instância do tipo `json_t` (definido no âmbito da biblioteca `jansson`) e representa os dados a enviar no pedido em formato JSON. Se ocorrer um erro durante a transferência, a função retorna `false` e mostra em `stderr` a mensagem que indica a razão do erro.

```
bool http_post_json(const char *url, json_t *data);
```

Para teste realize um programa que acrescente um novo carrinho de compras com o conteúdo `{"userId": 1, "products": [{"id": 144, "quantity": 4}]}` através do URL <https://dummyjson.com/carts/add> e afixe na consola o resultado da operação.

3. Utilizando a função do ponto anterior, implemente as funções `products_get`, `users_get` e `cart_put`.

Na elaboração destas funções, utilize os seguintes tipos de dados para representar, respetivamente, um produto; um utilizador; um carrinho de compras:

```
typedef struct {  
    int id;  
    float price;  
    const char *description;  
    const char *category;  
} Product;
```

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

```
typedef struct {  
    int user_id;  
    size_t n_products;  
    struct {  
        int id;  
        size_t quantity;  
    } products[];  
} Cart;
```

A função `products_get` retorna a informação sobre todos os produtos. Esta informação é obtida em <https://dummyjson.com/products>. Em caso de erro a função devolve `NULL`.

```
Products *products_get();
```

A função **users_get** retorna informação sobre todos os utilizadores. Esta informação é obtida em <https://dummyjson.com/users>. Em caso de erro a função devolve **NULL**.

Users *user_get();

Deve definir os tipos **Products** e **Users** de modo a representarem a informação sobre todos os elementos do respetivo conjunto.

A função **cart_put** acrescenta um novo carrinho de compras. Esta informação é enviada para <https://dummyjson.com/carts/add>. Devolve um valor booleano indicando o êxito da operação.

bool cart_put(Cart *cart);

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

Realize um programa de teste que crie um ficheiro para os utilizadores e outro para produtos obtidos. Os ficheiros devem ser formatados em formato **csv**. Inclua também o teste da função **cart_put** que acrescente um carrinho de compras com os dados lidos de um ficheiro.

Nota: Como alguns campos contêm o caractere vírgula, pode usar outro caractere como separador para evitar colisão. Por exemplo, o caractere TAB (t).

4. 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.

Utilize o programa de teste anterior para verificar o correto funcionamento da biblioteca.

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

O programa deve aceitar os seguintes comandos:

Utilizadores	Listar os utilizadores
uTilizador <identificador>	Assumir o utilizador indicado como o utilizador corrente.
Produtos <categoria> <critério>	Listar os produtos da categoria indicada segundo o critério de ordenação indicado. Se a categoria não for reconhecida lista todos os produtos. Os critérios possíveis são “preço crescente” ou “preço decrescente”. Utilize os sinais < e > para indicar.
Carrinho	Listar os produtos que estão no carrinho de compras (descrição, preço, quantidade).
cOmprar <produto> <quantidade>	Acrescentar um produto ao carrinho de compras.
Finalizar	Finalizar compra.
Terminar	Termina a execução do programa.

Na construção deste programa deve-se utilizar a biblioteca produzida no exercício anterior.

Na execução de comandos posteriores deve-se reutilizar a informação adquirida em comandos anteriores, minimizando assim o número de pedidos remotos. Para retenção da informação deve criar em memória uma estrutura de dados adequada. Na construção dessa estrutura de dados estabeleça referências com ponteiros entre as entidades, de modo a suprimir ou minimizar as procuras exaustivas. Por exemplo, na variável que representa um carrinho deve existir ponteiros para as variáveis que representam produtos.

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

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

Exemplifique a utilização dessa funcionalidade acrescentando um novo comando que lista na consola as categorias de produtos, que estão no carrinho.

Data limite de entrega: 15 de dezembro de 2025

ISEL, 11 de novembro de 2024