Programmazione di sistema

Anno accademico 2018-2019

Esercitazione 2

Si realizzi una libreria per la descrizione di un filesystem contenente directory e file regolari. Per lo svolgimento dell'esercitazione è **necessario utilizzare contenitori della STL e smart pointer**.

La classe astratta **Base** è la base comune da cui derivano Directory e File e non è istanziabile. Offre le seguenti funzioni membro pubbliche:

- `std::string getName() const` – restituisce il nome dell'oggetto
- `virtual int mType() const = 0` – metodo virtuale puro di cui fare override nelle classi derivate; restituisce il tipo dell'istanza (Directory o File) codificato come intero
- `virtual void ls(int indent=0) const = 0` – metodo virtuale puro di cui fare override nelle classi derivate.

La classe **Directory** deriva da Base ed è una classe singleton, il cui costruttore è protetto. Mantiene come membri privati una collezione di shared_ptr ad altri file di tipo Directory e File, uno weak_ptr alla directory genitore e uno weak_ptr a sé stessa. Il metodo statico getRoot() permette di accedere a una singola istanza (corrispondente alla cartella "/") attraverso la quale si può interagire con il modello.
La classe Directory espone le seguenti funzioni membro pubbliche:

- `static std::shared_ptr<Directory> getRoot()` – crea, se ancora non esiste, l'oggetto di tipo Directory e ne restituisce lo smart pointer.
- `std::shared_ptr<Directory> addDirectory(std::string nome)` – crea un nuovo oggetto di tipo Directory, il cui nome è desunto dal parametro, e lo aggiunge alla cartella corrente. Se risulta già presente, nella cartella corrente, un oggetto con il nome indicato, solleva un'eccezione.
- `std::shared_ptr<File> addFile(std::string nome, uintmax_t size)` – aggiunge alla Directory un nuovo oggetto di tipo File, ricevendone come parametri il nome e la dimensione in byte; l'aggiunta di un File con nome già presente nella cartella corrente non è permessa e causa un'eccezione
- `std::shared_ptr<Base> get(std::string name)` – restituisce uno smart pointer all'oggetto (Directory o File) di nome "name" contenuto nella directory corrente. Se inesistente, restituisce uno shared_ptr vuoto. I nomi speciali "`..`" e "`.`" permettono di ottenere rispettivamente lo shared_pointer alla directory genitore di quella corrente, e quello all'istanza stessa.
- `std::shared_ptr<Directory> getDir(std::string name)` – funziona come il metodo get(nome), facendo un dynamic_pointer_cast dal tipo Base al tipo Directory
- `std::shared_ptr<File> getFile(std::string name)` – funziona come il metodo get(nome), facendo un dynamic_pointer_cast dal tipo Base al tipo File

- **void remove**(std::string nome) – rimuove dalla collezione di figli della directory corrente l'oggetto (Directory o File) di nome "nome". La rimozione degli oggetti di nome "**..**" e "**.**" non è permessa e causa un'eccezione
- **void ls**(int indent=0) **const override** – implementa il metodo virtuale puro della classe Base; elenca ricorsivamente File e Directory figli della directory corrente, indentati in modo appropriato

Anche la classe **File** deriva da Base e offre le seguenti funzioni membro pubbliche aggiuntive:

- **uintmax_t getSize**() **const** – restituisce la dimensione del file
- **void ls**(int indent=0) **const override** – implementa il metodo virtuale puro della classe Base; stampa nome e dimensione del file con indentazione appropriata

Nota. La relazione di possesso del contenitore verso gli oggetti contenuti si può realizzare tramite shared_pointer. Per evitare cicli, "self" e la relazione "ha-genitore" vanno invece rappresentate tramite weak_ptr.

Esempio di uso

```
std::shared_ptr<Directory> root = Directory::getRoot();
auto alfa = root->addDirectory("alfa");
alfa->addDirectory("beta")->addFile("beta1",100);
alfa->getDir("beta")->addFile("beta2", 200);
alfa->getDir("..")->ls();
alfa->remove("beta");
root->ls();
```

Output

```
[+] /
       [+] alfa
       [+] beta
              beta1 100
              beta2 200

[+] /
       [+] alfa
```

Competenze da acquisire

- Eredità e polimorfismo
- Uso dei contenitori della Standard Template Library
- Uso di smart pointer
- Uso delle eccezioni