UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR | COLLÈGE STEE SCIENCES ET TECHNOLOGIES POUR L'ÉNERGIE ET L'ENVIRONNEMENT

Programmation orientée objet - Projet - EDP

Samson Pierre <samson.pierre@univ-pau.fr>

01/04/2022

Vous devez créer un programme nommé EDP (Envelope Delivery Program) qui doit lire un fichier XML dont le chemin est passé en ligne de commande. Ce fichier XML contient des données concernant une entreprise et les enveloppes qu'elle livre. Ce programme doit proposer de saisir des commandes pour afficher des informations contenues dans ce fichier XML.

Le travail à rendre

L'archive à rendre doit contenir exactement ces fichiers :

```
$ tar tf projet-wilfrid-lefer-samson-pierre.tar.xz | sort
projet-wilfrid-lefer-samson-pierre/
projet-wilfrid-lefer-samson-pierre/address.cpp
projet-wilfrid-lefer-samson-pierre/address.h
projet-wilfrid-lefer-samson-pierre/company.cpp
projet-wilfrid-lefer-samson-pierre/company.h
projet-wilfrid-lefer-samson-pierre/company-ups.xml
projet-wilfrid-lefer-samson-pierre/edp.cpp
projet-wilfrid-lefer-samson-pierre/envelope-c4.cpp
projet-wilfrid-lefer-samson-pierre/envelope-c4.h
projet-wilfrid-lefer-samson-pierre/envelope.cpp
projet-wilfrid-lefer-samson-pierre/envelope-dl.cpp
projet-wilfrid-lefer-samson-pierre/envelope-dl.h
projet-wilfrid-lefer-samson-pierre/envelope.h
projet-wilfrid-lefer-samson-pierre/Makefile
projet-wilfrid-lefer-samson-pierre/str2i-error.cpp
projet-wilfrid-lefer-samson-pierre/str2i-error.h
projet-wilfrid-lefer-samson-pierre/str2l-error.cpp
projet-wilfrid-lefer-samson-pierre/str2l-error.h
$
```

Les prénoms `wilfrid` et `samson` ainsi que les noms `lefer` et `pierre` sont à remplacer par les vôtres. Le fichier `Makefile` contient les règles dont se sert le programme `make`. Les fichiers `address.h`, `company.h`, `envelope-c4.h`, `envelope-dl.h`, `envelope.h`, `str2i-error.h` et `str2l-error.h` contiennent respectivement les déclarations pour une adresse, une entreprise, une enveloppe format C4, une enveloppe format DL, une enveloppe, une exception de conversion d'une chaîne de caractères vers un entier et une exception de conversion d'une chaîne de caractères vers un entier long. Les fichiers `address.cpp`, `company.cpp`, `envelope-c4.cpp`, `envelope-dl.cpp`, `envelope.cpp`, `str2i-error.cpp` et `str2l-error.cpp` contiennent respectivement les définitions pour une adresse, une entreprise, une enveloppe format C4, une enveloppe format DL, une enveloppe, une exception de conversion d'une chaîne de caractères vers un entier et une exception de conversion d'une chaîne de caractères vers un entier long. Le fichier `edp.cpp` contient les définitions pour le programme (notamment celle de la fonction principale). Le fichier `company-ups.xml` contient des données concernant une entreprise et les enveloppes qu'elle livre.

La compilation du projet doit se passer ainsi :

```
$ make
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o address.o address.cpp
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o company.o company.cpp
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o envelope.o
envelope.cpp
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o envelope-c4.o
envelope-c4.cpp
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o envelope-dl.o
envelope-dl.cpp
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o str2i-error.o
str2i-error.cpp
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -c -o str2l-error.o
str2l-error.cpp
```

```
g++ -std=c++98 -pedantic -Wall -Werror -g `pkg-config pugixml --cflags` -o edp.out edp.cpp
address.o company.o envelope.o envelope-c4.o envelope-dl.o str2i-error.o str2l-error.o `pkg-config
pugixml --libs-only-L` `pkg-config pugixml --libs-only-l`
$
```

Voici les classes autorisées pour ce projet :

```
pugi::xml_attribute
pugi::xml_document
pugi::xml_node
pugi::xml_parse_result
std::exception
std::ostringstream
std::string
std::vector
```

Voici les fonctions autorisées pour ce projet :

```
char *fgets(char *s, int size, FILE *stream);
int getchar(void);
int strcmp(const char *s1, const char *s2);
size_t strlen(const char *s);
char *strstr(const char *haystack, const char *needle);
long int strtol(const char *nptr, char **endptr, int base);
```

Pour obtenir une bonne note, vous devez respecter ces règles :
- les fichiers doivent être encodés en UTF-8
- les fichiers ne doivent pas contenir de fautes d'orthographe
- les noms de fichiers doivent être ceux indiqués dans ce sujet
- les fichiers d'en-tête doivent être identiques à ceux du sujet
- le fichier XML doit être identique à celui du sujet
- les options de compilation doivent être celles précisées dans ce sujet
- les affichages à l'écran doivent correspondre à ceux du sujet
- la solution doit se rapprocher au maximum de ce qui est demandé dans ce sujet
- la solution doit ignorer tout noeud ou attribut qui ne décrit pas une entreprise et les enveloppes qu'elle livre
- la solution doit accepter tout fichier XML similaire à celui du sujet (`company-ups.xml` n'est qu'un exemple)
- le code doit être correctement indenté
- le code doit être homogène concernant le nom des variables, les espaces, ...
- la mémoire allouée doit être correctement libérée
- les classes et fonctions que vous utilisez dans votre code doivent figurer parmi celles autorisées dans ce sujet
- la valeur de retour des fonctions pouvant échouer doit être vérifiée afin de traiter les erreurs
- l'affichage des messages d'erreur doit se faire uniquement dans la fonction principale
- le format des messages d'erreur doit être identique à celui présenté dans ce sujet
- les messages d'erreur doivent être écrits dans le flux d'erreur standard
- les autres messages doivent être écrits dans le flux de sortie standard
- le code retourné par un processus si erreur doit être `1`
- le code retourné par un processus si succès doit être `0`
- l'archive ne doit pas contenir d'autres fichiers que ceux demandés dans ce sujet
- l'archive doit être envoyée au plus tard le 22/04/2022 à 23:59
- l'archive doit être envoyée par e-mail à l'adresse `samson.pierre@univ-pau.fr`
- le sujet de l'e-mail doit être `POO - Projet - Wilfrid Lefer - Samson Pierre`
- les prénoms `Wilfrid` et `Samson` sont à remplacer par les vôtres
- les noms `Lefer` et `Pierre` sont à remplacer par les vôtres
- le travail est à réaliser impérativement en binôme
- le binôme doit travailler en collaboration via un projet privé sur la forge `https://git.univ-pau.fr/`
- tout travail similaire à un autre sera sanctionné par une note nulle (0/20)
- tout étudiant signalé comme n'ayant pas participé suffisamment au travail sera sanctionné par une note nulle (0/20)

Le programme

Le programme doit lire un fichier XML dont le chemin est passé en ligne de commande. Si le nombre de paramètres passés en ligne de commande est différent de un, le programme doit afficher un message d'erreur et quitter. Idem si l'analyse du fichier échoue (exemple : s'il n'existe pas).

```
$ ./edp.out
./edp.out: invalid number of arguments
$ ./edp.out file.xml
./edp.out: unable to parse the document
```

```
$
```

Le programme doit proposer de saisir des commandes :

```
$ ./edp.out company-ups.xml
EDP>
```

La commande `h` permet d'obtenir cet affichage :

```
$ ./edp.out company-ups.xml
EDP> h
e: prints the envelopes
ec CODE: prints the envelopes with the sender postal code equal to CODE
ecge CODE: prints the envelopes with the sender postal code greater than or equal to CODE
ecgt CODE: prints the envelopes with the sender postal code greater than CODE
ecle CODE: prints the envelopes with the sender postal code less than or equal to CODE
eclt CODE: prints the envelopes with the sender postal code less than CODE
en NAME: prints the envelopes with the sender name containing NAME
h: prints this help
i: prints information about the company
n: prints the company name
q: quits EDP
v: prints the EDP version
w: prints the company web address
EDP>
```

La commande `v` permet d'obtenir cet affichage :

```
$ ./edp.out company-ups.xml
EDP> v
EDP (Envelope Delivery Program) 20220306

Copyright (C) 2022 Wilfrid Lefer and Samson Pierre.

Written by Wilfrid Lefer <wilfrid.lefer@univ-pau.fr> and Samson Pierre <samson.pierre@univ-pau.fr>.
EDP>
```

Les prénoms `Wilfrid` et `Samson`, les noms `Lefer` et `Pierre` ainsi que les adresses e-mail `wilfrid.lefer@univ-pau.fr` et `samson.pierre@univ-pau.fr` sont à remplacer par les vôtres. La version `20220306` est à remplacer par la version de votre programme.

La commande `q` permet de quitter le programme :

```
$ ./edp.out company-ups.xml
EDP> q
$
```

Les autres commandes affichent un contenu spécifique à l'entreprise et les enveloppes qu'elle livre :

```
$ ./edp.out company-ups.xml
EDP> e
((Alain Verse, 1 rue de la programmation, 64320, Bizanos, France) -> (Bernard Tichaut, 3 rue de l'objet, 64140, Lons, France) [low] 229x324 mm)
((Camille Onnette, 2 rue de l'instruction, 64000, Pau, France) -> (Laure Dure, 4 rue de l'expression, 64110, Uzos, France) [high] 110x220 mm)
EDP> ec 64320
((Alain Verse, 1 rue de la programmation, 64320, Bizanos, France) -> (Bernard Tichaut, 3 rue de l'objet, 64140, Lons, France) [low] 229x324 mm)
EDP> ecge 64320
((Alain Verse, 1 rue de la programmation, 64320, Bizanos, France) -> (Bernard Tichaut, 3 rue de l'objet, 64140, Lons, France) [low] 229x324 mm)
EDP> ecgt 64320
EDP> ecle 64320
((Alain Verse, 1 rue de la programmation, 64320, Bizanos, France) -> (Bernard Tichaut, 3 rue de l'objet, 64140, Lons, France) [low] 229x324 mm)
((Camille Onnette, 2 rue de l'instruction, 64000, Pau, France) -> (Laure Dure, 4 rue de l'expression, 64110, Uzos, France) [high] 110x220 mm)
EDP> eclt 64320
((Camille Onnette, 2 rue de l'instruction, 64000, Pau, France) -> (Laure Dure, 4 rue de l'expression, 64110, Uzos, France) [high] 110x220 mm)
EDP> en Al
((Alain Verse, 1 rue de la programmation, 64320, Bizanos, France) -> (Bernard Tichaut, 3 rue de l'objet, 64140, Lons, France) [low] 229x324 mm)
EDP> i
(UPS, https://www.ups.com/, (((Alain Verse, 1 rue de la programmation, 64320, Bizanos, France) -> (Bernard Tichaut, 3 rue de l'objet, 64140, Lons, France) [low] 229x324 mm), ((Camille Onnette, 2 rue de l'instruction, 64000, Pau, France) -> (Laure Dure, 4 rue de l'expression, 64110, Uzos, France) [high] 110x220 mm)))
EDP> n
UPS
```

```
EDP> w
https://www.ups.com/
EDP>
```

Si la commande saisie n'existe pas, le programme doit afficher un message d'erreur. Idem si le paramètre de la commande manque, si le paramètre de la commande est incorrect, si le paramètre de la commande dépasse la plage de valeurs ou si la commande dépasse 18 caractères :

```
$ ./edp.out company-ups.xml
EDP> hello
./edp.out: invalid command
EDP> ec
./edp.out: missing parameter for the ec command
EDP> ec abc
./edp.out: invalid parameter for the ec command
EDP> ec 2147483648
./edp.out: invalid parameter for the ec command
EDP> ec 1234567890123456
./edp.out: too many characters for the command
EDP>
```

Si lors de l'analyse du fichier XML, une fonction autre que la fonction principale échoue à convertir une chaîne de caractères vers un entier ou un entier long à cause d'un débordement par le bas, d'un débordement par le haut ou d'un caractère invalide, cette fonction doit lever une exception qui doit être traitée dans la fonction principale :

```
$ ./edp.out company-ups.xml
./edp.out: an exception occurred (str2i_error: unable to convert the string "2147483648" to an int)
$ ./edp.out company-ups.xml
./edp.out: an exception occurred (str2l_error: unable to convert the string "9223372036854775808"
to a long)
$ ./edp.out company-ups.xml
./edp.out: an exception occurred (str2l_error: unable to convert the string "abc" to a long)
$
```

Les fichiers d'en-tête

Voici le fichier d'en-tête `address.h` :

```
/**
 * \file address.h
 */
#ifndef ADDRESS_H
#define ADDRESS_H
#include <string> // for string
using namespace std; // for string
/**
 * An address.
 */
class address_t
{
        string city; /**< The city (example: "Bizanos"). */
        string country; /**< The country (example: "France"). */
        string name; /**< The name (example: "Alain Verse"). */
        int postal_code; /**< The postal code (example: "64320"). */
        string street; /**< The street (example: "1 rue de la programmation"). */
        public:
                /**
                 * Constructs a new address.
                 * Initializes its city, country, name and street to "undefined".
                 * Initializes its postal code to 0.
                 */
                address_t();
                /**
                 * Gets the city for this address.
                 * \return The city for this address.
                 */
                string get_city() const;
                /**
                 * Gets the country for this address.
                 * \return The country for this address.
                 */
                string get_country() const;
                /**
                 * Gets the name for this address.
                 * \return The name for this address.
```

```
38                  */
39                 string get_name() const;
40                 /**
41                  * Gets the postal code for this address.
42                  * \return The postal code for this address.
43                  */
44                 int get_postal_code() const;
45                 /**
46                  * Gets the street for this address.
47                  * \return The street for this address.
48                  */
49                 string get_street() const;
50                 /**
51                  * Sets the city for this address.
52                  * \param city The city for this address.
53                  */
54                 void set_city(string city);
55                 /**
56                  * Sets the country for this address.
57                  * \param country The country for this address.
58                  */
59                 void set_country(string country);
60                 /**
61                  * Sets the name for this address.
62                  * \param name The name for this address.
63                  */
64                 void set_name(string name);
65                 /**
66                  * Sets the postal code for this address.
67                  * \param postal_code The postal code for this address.
68                  */
69                 void set_postal_code(int postal_code);
70                 /**
71                  * Sets the street for this address.
72                  * \param street The street for this address.
73                  */
74                 void set_street(string street);
75 };
76 /**
77  * Inserts an address into an output stream.
78  * The inserted string is "(name, street, postal_code, city, country)".
79  * \param os The output stream.
80  * \param address The address.
81  * \return The output stream.
82  */
83 ostream &operator<<(ostream &os, const address_t &address);
84 #endif
```

Voici le fichier d'en-tête `company.h` :

```
1  /**
2   * \file company.h
3   */
4  #ifndef COMPANY_H
5  #define COMPANY_H
6  #include <string> // for string
7  #include <vector> // for vector
8  #include "envelope.h" // for envelope_t
9  using namespace std; // for string
10 /**
11  * A company.
12  */
13 class company_t
14 {
15         vector<envelope_t> envelopes; /**< The envelopes. */
16         string name; /**< The name (example: "UPS"). */
17         string web; /**< The web address (example: "https://www.ups.com/"). */
18         public:
19                 /**
20                  * Constructs a new company.
21                  * Initializes its name and web address to "undefined".
22                  */
23                 company_t();
24                 /**
25                  * Gets the name for this company.
26                  * \return The name for this company.
27                  */
```

```
28                  string get_name() const;
29                  /**
30                   * Gets the web address for this company.
31                   * \return The web address for this company.
32                   */
33                  string get_web() const;
34                  /**
35                   * Handles the e command for this company.
36                   * Iterates on envelopes to call the handle_e method on its elements.
37                   */
38                  void handle_e() const;
39                  /**
40                   * Handles the i command for this company.
41                   * Calls the << overloaded operator on this company.
42                   */
43                  void handle_i() const;
44                  /**
45                   * Handles the n command for this company.
46                   */
47                  void handle_n() const;
48                  /**
49                   * Handles the ec command for this company.
50                   * Iterates on envelopes to call the handle_ec method on its elements.
51                   * \param postal_code The ec command parameter.
52                   */
53                  void handle_ec(int postal_code) const;
54                  /**
55                   * Handles the ecge command for this company.
56                   * Iterates on envelopes to call the handle_ecge method on its elements.
57                   * \param postal_code The ecge command parameter.
58                   */
59                  void handle_ecge(int postal_code) const;
60                  /**
61                   * Handles the ecgt command for this company.
62                   * Iterates on envelopes to call the handle_ecgt method on its elements.
63                   * \param postal_code The ecgt command parameter.
64                   */
65                  void handle_ecgt(int postal_code) const;
66                  /**
67                   * Handles the ecle command for this company.
68                   * Iterates on envelopes to call the handle_ecle method on its elements.
69                   * \param postal_code The ecle command parameter.
70                   */
71                  void handle_ecle(int postal_code) const;
72                  /**
73                   * Handles the eclt command for this company.
74                   * Iterates on envelopes to call the handle_eclt method on its elements.
75                   * \param postal_code The eclt command parameter.
76                   */
77                  void handle_eclt(int postal_code) const;
78                  /**
79                   * Handles the en command for this company.
80                   * Iterates on envelopes to call the handle_en method on its elements.
81                   * \param name The en command parameter.
82                   */
83                  void handle_en(string name) const;
84                  /**
85                   * Handles the w command for this company.
86                   */
87                  void handle_w() const;
88                  /**
89                   * Accesses to the element of envelopes at the specified index for this company.
90                   * Calls the at method on envelopes.
91                   * \param index The index.
92                   * \return The element of envelopes at the specified index for this company.
93                   */
94                  envelope_t envelopes_at(int index) const;
95                  /**
96                   * Adds an element at the end of envelopes for this company.
97                   * Calls the push_back method on envelopes.
98                   * \param envelope The element.
99                   */
100                 void envelopes_push_back(envelope_t envelope);
101                 /**
102                  * Gets the size of envelopes for this company.
103                  * Calls the size method on envelopes.
```

```
104                  * \return The size of envelopes for this company.
105                  */
106                 int envelopes_size() const;
107                 /**
108                  * Sets the name for this company.
109                  * \param name The name for this company.
110                  */
111                 void set_name(string name);
112                 /**
113                  * Sets the web address for this company.
114                  * \param web The web address for this company.
115                  */
116                 void set_web(string web);
117 };
118 /**
119  * Inserts a company into an output stream.
120  * The inserted string is "(name, web, (envelopes1, envelopes2, ...))".
121  * Calls the << overloaded operator on the envelopes.
122  * \param os The output stream.
123  * \param company The company.
124  * \return The output stream.
125  */
126 ostream &operator<<(ostream &os, const company_t &company);
127 #endif
```

Voici le fichier d'en-tête `envelope-c4.h` :

```
1  /**
2   * \file envelope-c4.h
3   */
4  #ifndef ENVELOPE_C4_H
5  #define ENVELOPE_C4_H
6  #include "envelope.h" // for envelope_t
7  /**
8   * An envelope in the C4 format.
9   */
10 class envelope_c4_t: public envelope_t
11 {
12         public:
13                 /**
14                  * Constructs a new envelope in the C4 format.
15                  * Initializes its height to "324" and width to "229".
16                  */
17                 envelope_c4_t();
18 };
19 #endif
```

Voici le fichier d'en-tête `envelope-dl.h` :

```
1  /**
2   * \file envelope-dl.h
3   */
4  #ifndef ENVELOPE_DL_H
5  #define ENVELOPE_DL_H
6  #include "envelope.h" // for envelope_t
7  /**
8   * An envelope in the DL format.
9   */
10 class envelope_dl_t: public envelope_t
11 {
12         public:
13                 /**
14                  * Constructs a new envelope in the DL format.
15                  * Initializes its height to "220" and width to "110".
16                  */
17                 envelope_dl_t();
18 };
19 #endif
```

Voici le fichier d'en-tête `envelope.h` :

```
1  /**
2   * \file envelope.h
3   */
4  #ifndef ENVELOPE_H
5  #define ENVELOPE_H
6  #include <string> // for string
7  #include "address.h" // for address_t
```

```
8  using namespace std; // for string
9  /**
10  * A format.
11  */
12 enum priority_t
13 {
14         undefined,
15         high,
16         low,
17         medium
18 };
19 /**
20  * An envelope.
21  */
22 class envelope_t
23 {
24         private:
25                 priority_t priority; /**< The priority (example: "low"). */
26                 address_t recipient; /**< The recipient address. */
27                 address_t sender; /**< The sender address. */
28         protected:
29                 int height; /**< The height (in mm, example: "324"). */
30                 int width; /**< The width (in mm, example: "229"). */
31         public:
32                 /**
33                  * Constructs a new envelope.
34                  * Initializes its height and width to "0".
35                  */
36                 envelope_t();
37                 /**
38                  * Gets the height for this envelope.
39                  * \return The height for this envelope.
40                  */
41                 int get_height() const;
42                 /**
43                  * Gets the priority for this envelope.
44                  * \return The priority for this envelope.
45                  */
46                 priority_t get_priority() const;
47                 /**
48                  * Gets the recipient address for this envelope.
49                  * \return The recipient address for this envelope.
50                  */
51                 address_t get_recipient() const;
52                 /**
53                  * Gets the sender address for this envelope.
54                  * \return The sender address for this envelope.
55                  */
56                 address_t get_sender() const;
57                 /**
58                  * Gets the width for this envelope.
59                  * \return The width for this envelope.
60                  */
61                 int get_width() const;
62                 /**
63                  * Handles the e command for this envelope.
64                  * Calls the << overloaded operator on this envelope.
65                  */
66                 void handle_e() const;
67                 /**
68                  * Handles the ec command for this envelope.
69                  * Calls the handle_e method on this envelope if the condition is satisfied.
70                  * \param postal_code The ec command parameter.
71                  */
72                 void handle_ec(int postal_code) const;
73                 /**
74                  * Handles the ecge command for this envelope.
75                  * Calls the handle_e method on this envelope if the condition is satisfied.
76                  * \param postal_code The ecge command parameter.
77                  */
78                 void handle_ecge(int postal_code) const;
79                 /**
80                  * Handles the ecgt command for this envelope.
81                  * Calls the handle_e method on this envelope if the condition is satisfied.
82                  * \param postal_code The ecgt command parameter.
83                  */
```

```
84                  void handle_ecgt(int postal_code) const;
85                  /**
86                   * Handles the ecle command for this envelope.
87                   * Calls the handle_e method on this envelope if the condition is satisfied.
88                   * \param postal_code The ecle command parameter.
89                   */
90                  void handle_ecle(int postal_code) const;
91                  /**
92                   * Handles the eclt command for this envelope.
93                   * Calls the handle_e method on this envelope if the condition is satisfied.
94                   * \param postal_code The eclt command parameter.
95                   */
96                  void handle_eclt(int postal_code) const;
97                  /**
98                   * Handles the en command for this envelope.
99                   * Calls the handle_e method on this envelope if the condition is satisfied.
100                  * \param name The en command parameter.
101                  */
102                 void handle_en(string name) const;
103                 /**
104                  * Sets the priority for this envelope.
105                  * \param priority The priority for this envelope.
106                  */
107                 void set_priority(priority_t priority);
108                 /**
109                  * Sets the recipient address for this envelope.
110                  * \param recipient The recipient address for this envelope.
111                  */
112                 void set_recipient(address_t recipient);
113                 /**
114                  * Sets the sender address for this envelope.
115                  * \param sender The sender address for this envelope.
116                  */
117                 void set_sender(address_t sender);
118 };
119 /**
120  * Inserts an envelope into an output stream.
121  * The inserted string is "(sender -> recipient [priority] widthxheight)".
122  * \param os The output stream.
123  * \param envelope The envelope.
124  * \return The output stream.
125  */
126 ostream &operator<<(ostream &os, const envelope_t &envelope);
127 #endif
```

Voici le fichier d'en-tête `str2i-error.h` :

```
1  /**
2   * \file str2i-error.h
3   */
4  #ifndef STR2I_ERROR_H
5  #define STR2I_ERROR_H
6  #include <string> // for string
7  using namespace std; // for string
8  /**
9   * A string to int conversion exception.
10  */
11 class str2i_error: public exception
12 {
13         string str; /**< The string describing this string to int conversion exception. */
14         public:
15                 /**
16                  * Constructs a new string to int conversion exception.
17                  * \param str The string that cannot be converted to an int.
18                  */
19                 str2i_error(string str);
20                 /**
21                  * Destroys an existing string to int conversion exception.
22                  */
23                 ~str2i_error() throw ();
24                 /**
25                  * Creates a string describing this string to int conversion exception.
26                  * This string is "str2i_error: unable to convert the string \"str\" to an int".
27                  * \return The string describing this string to int conversion exception.
28                  */
29                 const char *what() const throw ();
30 };
```

```
#endif
```

Voici le fichier d'en-tête `str2l-error.h` :

```
/**
 * \file str2l-error.h
 */
#ifndef STR2L_ERROR_H
#define STR2L_ERROR_H
#include <string> // for string
using namespace std; // for string
/**
 * A string to long conversion exception.
 */
class str2l_error: public exception
{
        string str; /**< The string describing this string to long conversion exception. */
        public:
                /**
                 * Constructs a new string to long conversion exception.
                 * \param str The string that cannot be converted to a long.
                 */
                str2l_error(string str);
                /**
                 * Destroys an existing string to long conversion exception.
                 */
                ~str2l_error() throw ();
                /**
                 * Creates a string describing this string to long conversion exception.
                 * This string is "str2l_error: unable to convert the string \"str\" to a long".
                 * \return The string describing this string to long conversion exception.
                 */
                const char *what() const throw ();
};
#endif
```

Le fichier XML

Voici le fichier XML `company-ups.xml` :

```
<company name="UPS">
        <envelopes>
                <envelope format="c4">
                        <priority>low</priority>
                        <recipient>
                                <city>Lons</city>
                                <country>France</country>
                                <name>Bernard Tichaut</name>
                                <postal-code>64140</postal-code>
                                <street>3 rue de l'objet</street>
                        </recipient>
                        <sender>
                                <city>Bizanos</city>
                                <country>France</country>
                                <name>Alain Verse</name>
                                <postal-code>64320</postal-code>
                                <street>1 rue de la programmation</street>
                        </sender>
                </envelope>
                <envelope format="dl">
                        <priority>high</priority>
                        <recipient>
                                <city>Uzos</city>
                                <country>France</country>
                                <name>Laure Dure</name>
                                <postal-code>64110</postal-code>
                                <street>4 rue de l'expression</street>
                        </recipient>
                        <sender>
                                <city>Pau</city>
                                <country>France</country>
                                <name>Camille Onnette</name>
                                <postal-code>64000</postal-code>
                                <street>2 rue de l'instruction</street>
                        </sender>
```

```
36                  </envelope>
37          </envelopes>
38          <web>https://www.ups.com/</web>
39  </company>
```