INSA Lyon
20, avenue Albert Einstein
69621 Villeurbanne Cedex

Livrable de projet

Grammaire et langage

« Analyse de fichiers XML »

du 14 Mars au XX Mars 2014

Binôme :
Guillaume Abadie
Thierry Cantenot
Juliette Courlet
Rémi Domingues
Adrien Duffy-Coissard
Ahmed Kachkach

Enseignants :
Nabila Benharkat
Eric Guérin

Année scolaire 2013-2014

Sommaire

1. Chapitre trololo

Trololo et tralala.

2. Documentation sommaire doxygen

2.1 Brefs points essentiels

Globalement, un commentaire se fera dans ce style la, avant une methode ou en tete de fichier :

```
/**
 * ... text ...
 */
```

Si vous voulez faire apparaitre un point important et resumant d abord, pour ensuite mettre des details il faut mettre :

```
/*! \brief Brief description.
 *         Brief description continued.
 *
 *  Detailed description starts here.
 */
```

Des commentaires peuvent etre ajoutes dans les fonctions, par exemple apres certains attributs. Il faut alors placer un $\ddot{<}\ddot{}$:

```
int var; /**< Detailed description after the member */
```

Lorsque vous documentez une enum, classe... il faut utiliser toujours le commentaire de base avec
brief, mais en ajoutant au debut :

```
/**
    \struct to document a C-struct.
    \union to document a union.
    \enum to document an enumeration type.
    \fn to document a function.
    \var to document a variable or typedef or enum value.
    \def to document a #define.
    \typedef to document a type definition.
    \file to document a file.
    \namespace to document a namespace.
    \package to document a Java package.
    \interface to document an IDL interface.
    \bug to signal a bug
*/
```

2.2 Templates c++

Tout d'abord, nous obtenons un template de fichier de cette forme :

```
/**
 * \file main.c  ( specifie que vous desirez une doc sur ce fichier)
 * \brief Programme de tests.  (description breve du fichier)
 * \author Franck.H
 * \version 0.1
 * \date 11 septembre 2007
 *
 * Programme de test pour l'objet de gestion des chaines de caracteres Str_t.
 *
 */
```

A placer au tout debut, il permet d'initier une documentations Doxygen sur le fichier, notamment grace a la balise
file, a ne pas ommettre .

Au final, on obtient un template de methode de ce type, avec le commentaire des parametres (entree ou sortie) :

```
/**
 * \brief       Calcule la distance entre deux points
 * \details    La distance entre les \a point1 et \a point2 est calculee par l'
    intermediaire
 *                  des coordonnees des points. (cf #Point)
 * \param[in]   nom_param1         nom_param 1 pour le calcul de distance.
 * \param[out]    nom_param2         nom_param 2 retourner une valeur.
 * \return    Un \e float representant la distance calculee.
 */
```

2.3 Exemples

En utilisant les balises decrites au dessus, nous pouvons deduire du commentaire :

2.3.1 Un enum

```
/**
 * \enum Str_err_e
 * \brief Constantes d'erreurs.
 *
 * Str_err_e est une serie de constantes predefinie pour diverses futures
 * fonctions de l'objet Str_t.
 */
typedef enum
{
   STR_NO_ERR,    /*!< Pas d'erreur. */
   STR_EMPTY_ERR, /*!< Erreur: Objet vide ou non initialise. */

   NB_STR_ERR     /*!< Nombre total de constantes d'erreur. */
}
Str_err_e;
```

2.3.2 Une struct

```
/**
 * \struct Str_t
 * \brief Objet chaine de caracteres.
 *
 * Str_t est un petit objet de gestion de chaines de caracteres.
 * La chaine se termine obligatoirement par un zero de fin et l'objet
 * connait la taille de chaine contient !
 */
```

```
typedef struct
{
   char * sz;  /*!< Chaine avec  caractere null de fin de chaine. */
   size_t len; /*!< Taille de la chaine sz sans compter le zero de fin. */
}
Str_t;
```

Il est possible de signaler un bug :

```
/**
\bug { bug description }
*/
```

2.3.3 Une fonction statique

```
/**
 * \fn static Str_t * str_new (const char * sz)
 * \brief Fonction de creation d'une nouvelle instance d'un objet Str_t.
 *
 * \param sz Chaine a stocker dans l'objet Str_t, ne peut etre NULL.
 * \return Instance nouvelle allouee d'un objet de type Str_t ou NULL.
 */
static Str_t * str_new (const char * sz)
{
   Str_t * self = NULL;

   if (sz != NULL && strlen (sz) > 0)
   {
      self = malloc (sizeof (* self));

      if (self != NULL)
      {
         self->len = strlen (sz);
         self->sz = malloc (self->len + 1);

         if (self->sz != NULL)
         {
            strcpy (self->sz, sz);
         }
         else
         {
            str_destroy (& self);
         }
      }
   }

   return self;
}
```

2.3.4 exemple c++ complet

```
#ifndef CPLAYER_H_
#define CPLAYER_H_

/*!
 * \file CPlayer.h
 * \brief Lecteur de musique de base
 * \author hiko-seijuro
 * \version 0.1
 */
#include <string>
#include <list>

/*! \namespace player
 *
```

```
 * espace de nommage regroupant les outils composants
 * un lecteur audio
 */
namespace player
{
  /*! \class CPlayer
   * \brief classe representant le lecteur
   *
   *  La classe gere la lecture d'une liste de morceaux
   */
  class CPlayer
  {
  private:
    std::list<string> m_listSongs; /*!< Liste des morceaux*/
    std::list<string>::iterator m_currentSong; /*!< Morceau courant */


  public:
    /*!
     *  \brief Constructeur
     *
     *  Constructeur de la classe CPlayer
     *
     *  \param listSongs : liste initial des morceaux
     */
    CPlayer(std::list<string> listSongs);

    /*!
     *  \brief Destructeur
     *
     *  Destructeur de la classe CPlayer
     */
    virtual ~CPlayer();

  public:
    /*!
     *  \brief Ajout d'un morceau
     *
     *  Methode qui permet d'ajouter un morceau a liste de
     *  lecture
     *
     *  \param strSong : le morceau a ajouter
     *  \return true si morceau deja present dans la liste,
     *  false sinon
     */
    bool add(std::string strSong);

    /*!
     *  \brief Morceau suivant
     *
     *  Passage au morceau suivant
     */
    void next();

    /*!
     *  \brief Morceau precedent
     *
     *  Passage au morceau precedent
     */
    void previous();

    /*!
     *  \brief Lecture
     *
     *  Lance la lecture de la liste
     */
    void play();

    /*!
     *  \brief Arret
```

```
     *
     *  Arrete la lecture
     */
    void stop();
  };
};

#endif
```