Projet de Programmation

Benoit Donnet

Année Académique 2023 - 2024

Agenda

Partie 2: Outils

Agenda

- Chapitre 4: Documentation
  - Introduction
  - Recommandations
  - `Doxygen`

Agenda

- Chapitre 4: Documentation
  - Introduction
  - Recommandations
  - `Doxygen`

Introduction

- Quelques témoignages
  - j'ai écrit ce code l'an dernier. Je ne comprends plus comment il fonctionne :-s
  - mon binôme a écrit beaucoup de code... mais je ne sais pas comment l'utiliser :-(
  - cette bibliothèque trouvée sur le Net n'est pas assez bien documentée, alors j'ai préféré écrire la mienne! :-)
    - ✓ tu as documenté ta bibliothèque? :-o
    - ✓ non plus :-s

Introduction (2)

- *Programs must be written for people to read and only incidentally for machines to execute*
  - Hal Abelson
- Le code écrit va être lu
  - par les autres membres du projet
  - par le programmeur lui-même (quelques mois plus tard)
  - par d'autres développeurs
  - par l'équipe pédagogique

Introduction (3)

- Ce qui était clair pour vous quand vous avez écrit la fonction/procédure
  - ne l'est peut-être pas pour les autres membres du projet
  - ne l'est peut-être pas pour vous-même
    - ✓ quelques mois plus tard
  - ne l'est peut-être pas pour d'autres développeurs
  - ne l'est peut-être pas pour l'équipe pédagogique
- Dans votre carrière professionnelle
  - vous allez passer plus de temps à reprendre du code qu'à en écrire
  - les gens vont passer plus de temps à reprendre votre code que vous à l'écrire...

Introduction (4)

- Conclusion 1
  - documentez votre code!
- Conclusion 2
  - documentez votre code!!
- Conclusion 3
  - documentez votre code!!!

Agenda

- Chapitre 4: Documentation
  - Introduction
  - Recommandations
  - Doxygen

Recommandations

- Indentation convenable du code
- Possibilité d'utiliser l'indentation automatique
  - disponible dans tout IDE
  - disponible sous vi
    - .vimrc
- Consistance
  - choisir une norme de présentation et la respecter

Recommandations (2)

- Pas bien

```
int bs(int *a, int s, int t)
{  int l = 0; int h=s-1;
  while(l <= h) {
  int m = l+(h-1)/2;
  if (t<a[m]) h = m - 1;
else if(t>a[m] ) l = m + 1;
    else return m;} return -1;
}
```

- Bien

```
int bs(int *a, int s, int t){
  int l = 0;
  int h = s-1;
  while(l <= h){
    int m = l+(h-1)/2;
    if(t < a[m])
      h = m - 1;
    else
      if(t > a[m])
        l = m + 1;
      else
        return m;
  }
  return -1;
}
```


Recommandations (3)

- Ecrivez vos commentaires comme s'ils étaient pour vous

```
//binary search of the value t in the array a of size s
int bs(int *a, int s, int t){
  int l = 0; //low index
  int h = s-1; //high index
  while(l <= h){
    int m = l+(h-1)/2; //middle index
    if(t < a[m])
      h = m - 1;
    else
      if(t > a[m])
        l = m + 1;
      else
        return m;
  }//end while
  return -1;
}//end bs()
```

Recommandations (4)

- Faites en sorte que le code parle de lui-même
  - bon code == besoin de moins de commentaires

```
//binary search of the value target in the array a
int binary_search(int *a, int size, int target){
  int low = 0;
  int high = size-1;
  while(low <= high){
    int middle = low+(high-1)/2;
    if(target < a[middle])
      high = middle - 1;
    else
      if(target > a[middle])
        low = middle + 1;
       else
        return middle;
  }//end while
  return -1;
}//end binary_search()
```


Recommandations (5)

- Définir le contrat de chaque fonction/procédure de manière non-ambigüe
  - un autre programmeur doit être capable d'écrire la fonction/procédure uniquement sur base des commentaires

Recommandations (6)

```
/*
 * Description: algorithm for quickly locating a particular
 * value in a sorted array.
 * Parameters:
 *  - a, an array sorted in the increasing order
 *  - size, number of elements in a (size >= 0)
 *  - target, value searched in the array
 * Return: the index of an array cell containing the target
 * value if any, -1 else.
 */
int binary_search(int *a, int size, int target){
  int low = 0;
  ...
}//end binary_search()
```


Agenda

- Chapitre 4: Documentation
  - Introduction
  - Recommandations
  - Doxygen
    - ✓ Généralités
    - ✓ Base
    - ✓ Documentation Module
    - ✓ Documentation Fonction
    - ✓ Documentation Structure

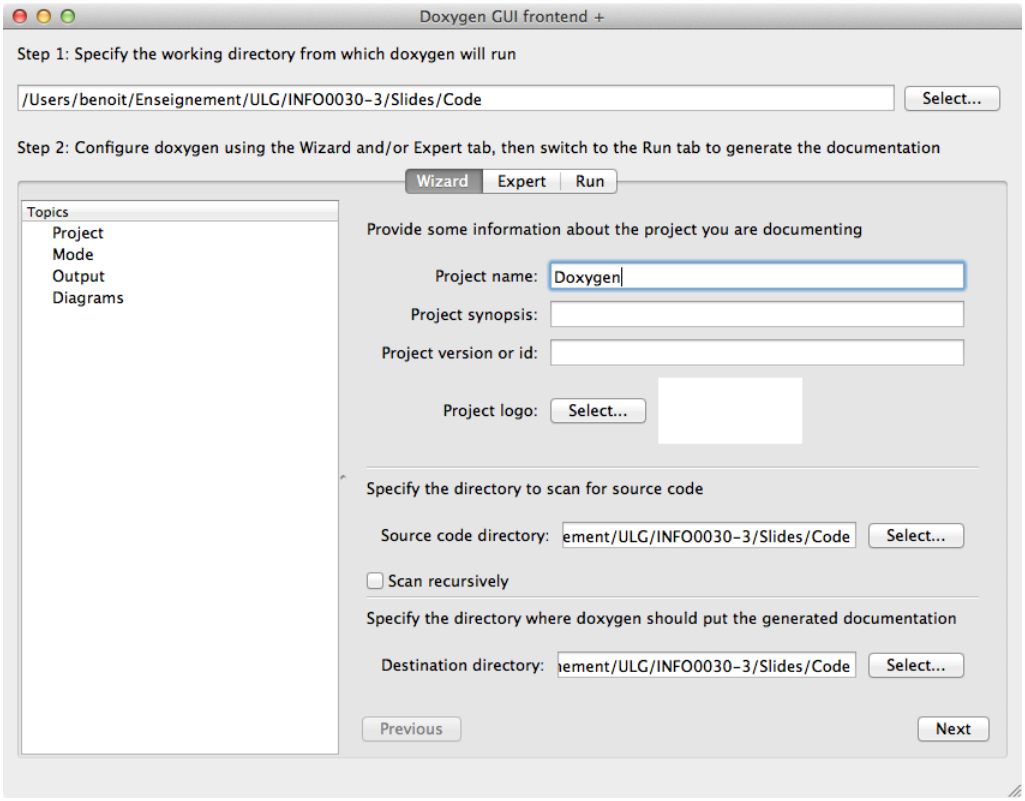
Généralités

- `Doxygen`?
  - générateur de documentation
  - la documentation est écrite dans le code source
    - ✓ facile à maintenir
  - documentation formatée
    - ✓ utilisation de tags bien connus
  - output dans différents formats
    - ✓ HTML, LaTeX, rtf, PDF, man page, ...
  - applicable à plusieurs langages de programmation
    - ✓ C, C++, C#, Java, Python, PHP, ...
- Disponible pour toutes les plateformes
  - http://www.doxygen.org
  - http://www.stack.nl/~dimitri/doxygen/manual/

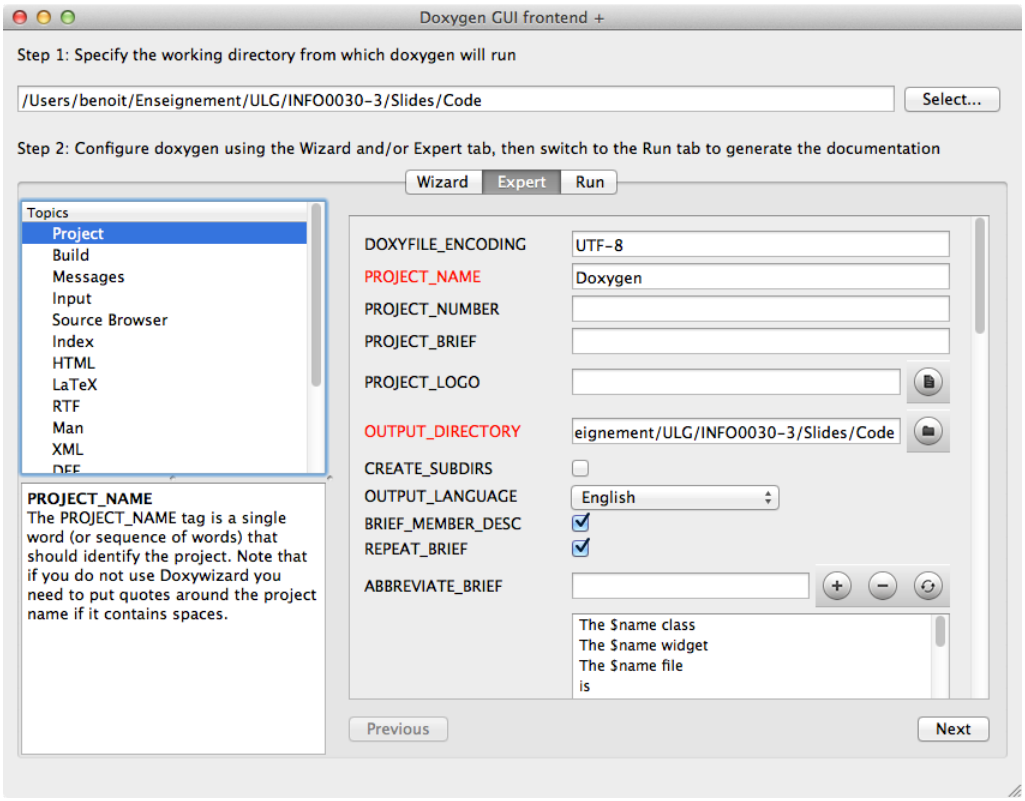
Généralités (2)

- Fonctionne
  - en ligne de commande
    - ✓ nécessite la création d'un fichier de configuration
  - via `DoxyWizard`
    - ✓ disponible pour toutes les plateformes

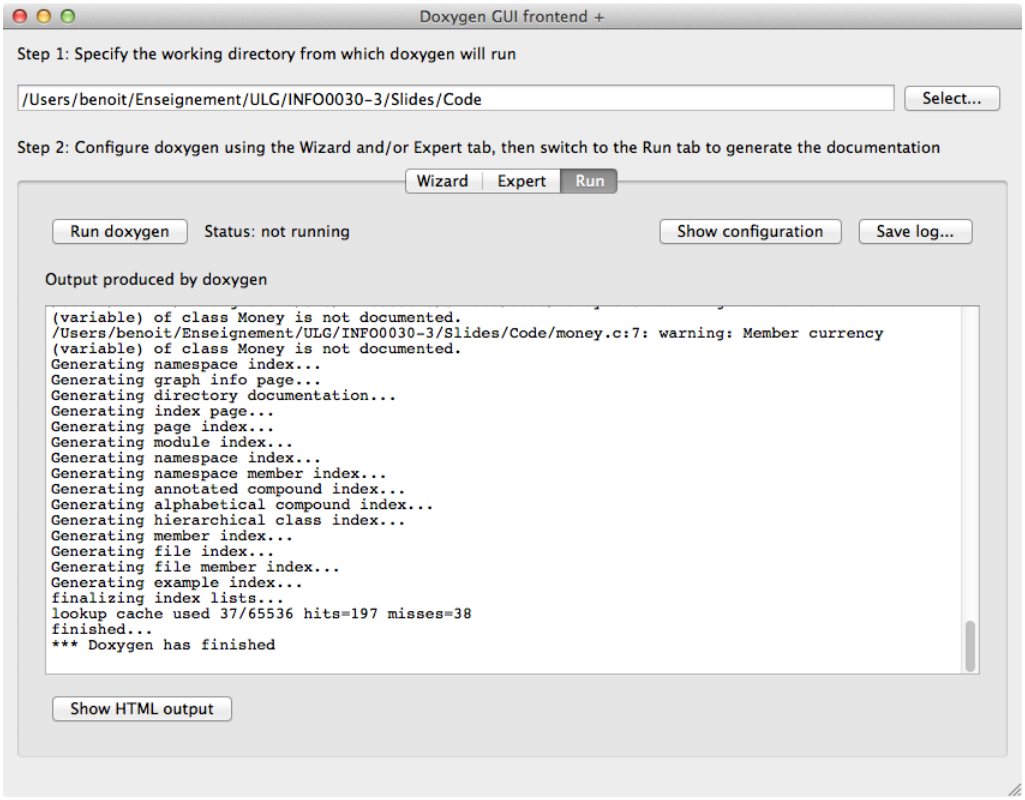
Généralités (3)



Généralités (4)



Généralités (5)



Bases

- Toute documentation `Doxygen` doit être "emballée"

```
/**
 * Documentation Doxygen
 *
 * ...
 *
 *
 */
```

Bases (2)

- `Doxygen` fonctionne par lecture de commandes particulières
  - permet un formatage de l'output
  - permet de structurer la documentation
    - ✓ standard maintenu tout au long du projet
- Chaque commande porte un nom particulier et est précédée de `\`

Bases (3)

- Exemples de commandes
  - `\brief`
    - ✓ paragraphe donnant une brève description
  - `\author`
    - ✓ paragraphe donnant le nom de l'auteur
  - `\fn`
    - ✓ paragraphe qui commence la documentation d'une fonction/procédure
  - `\param`
    - ✓ paragraphe décrivant un paramètre d'une fonction/procédure
  - `\result`
    - ✓ paragraphe décrivant le retour d'une fonction
- Il est possible d'étendre l'offre de commandes

Bases (4)

- Il est possible d'inclure des formules mathématiques dans la documentation
  - se fait en LaTeX
  - T. Oetiker, H. Partl, I. Hyna, E. Schlegl. A Not so Short Introduction to LaTeX. April 2011. cfr. http://tobi.oetiker.ch/lshort/lshort.pdf
- Exemple

```
/**
 * The distance between \f$(x_1,y_1)\f$ and \f$(x_2,y_2)\f$
 * is \f$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}\f$
 *
 * ...
 *
 *
 */
```


Documentation Module

- Chaque header d'un module doit avoir une description globale
- Cette description doit au moins comprendre les informations suivantes:
  - nom du fichier
  - brève description du module/header
  - nom de l'auteur
  - version
  - date

Documentation Module (2)

```
#ifndef __MONEY__
#define __MONEY__

/**
 * \file money.h
 * \brief Module for managing money
 * \author Benoit Donnet -- Université de Liège (ULg)
 * \version 0.1
 * \date 13/03/2013
 *
 * Manages money, as an opaque data structure.
 */
```


Documentation Module (3)

Detailed Description

Module for managing money.

Author
Benoit Donnet – Université de Liège (ULg)

Version
0.1

Date
13/03/2013

Manages money, as an opaque data structure.

Documentation Fonction

- Chaque fonction/procédure d'un module doit être documentée
- Cette description doit au moins comprendre les informations suivantes:
  - prototype de la fonction
  - brève description de l'objectif de la fonction
  - description des paramètres
  - description du retour (si s'applique)

Doc. Fonction (2)

```
/**
 * \fn Money *create_money(int amount, char *currency)
 * \brief Creates a Money data structure.
 *
 * \param amount, The amount of money (>=0).
 * \param currency, the money currency (!=NULL).
 *
 * \return A pointer to a well defined money data structure.
 *         NULL otherwise.
 */
Money *create_money(int amount, char *currency);
```

Doc. Fonction (3)

```
Money* create_money ( int      amount,
                      char *   currency
                    )
```

Creates a Money data structure.

Parameters

amount,The amount of money (>=0).

currency,the money currency (!=NULL).

Returns

A pointer to a well defined money data structure. NULL otherwise.

Documentation Structure

- Chaque structure d'un module doit être documentée
- Cette description doit au moins comprendre les informations suivantes:
  - le nom de la structure
  - brève description de la structure
- Si la structure n'est pas opaque, il est possible de documenter chaque champ

Doc. Structure (3)

```
/**
 * \struct Money
 * \brief Data structure representing money.
 */
typedef struct Money_t Money;
```

```
struct Money_t{
    int amount; /*!< The amount of money */
    char *currency; /*!< The money currency (EUR, $, ...) */
};
```


Doc. Structure (3)

Money_t Struct Reference

Public Attributes

int	**amount**
char *	**currency**

Member Data Documentation

int Money_t::amount

The amount of money

char* Money_t::currency

The money currency (EUR, $, ...)

The documentation for this struct was generated from the following file:

- money.c