READYDEV

[GitHub](https://github.com/gkesse) | [LinkedIn](https://www.linkedin.com/in/tia-gerard-kesse/) | [SiteWeb](http://readydev.freeboxos.fr/)

Par Gérard KESSE

Plateforme de Développement Continu

Comprendre la Théorie pour mieux Pratiquer

Sciences de l'Ingénieur

Cours - Tutoriels

READYDEV

[GitHub](https://github.com/gkesse) | [LinkedIn](https://www.linkedin.com/in/tia-gerard-kesse/) | [SiteWeb](http://readydev.freeboxos.fr/)

Par Gérard KESSE

J'aime, Je partage

Montez en Compétences

QMake

Apprendre à compiler un projet C++ avec QMake

# [Auteur](#_Sommaire)

Je suis **Gérard KESSE**,

Ingénieur en Développement Informatique C/C++/Qt,

Avec à la fois des compétences en Systèmes Embarqués et en Robotique.

Formé à Polytech'Montpellier, Je suis un professionnel de conception de projets logiciel applicatif ou embarqué dans les secteurs de l'Aéronautique, de la Robotique, des Drones et de la Vision par Ordinateur. Aussi, Je reste ouvert à d'autres types de secteurs tels que l'Énergie et les Finances.

Les Sciences de l’Ingénieur sont au cœur du métier d’ingénieur. Sur le site

**ReadyDev**, la Plateforme de Développement Continu, dont j'en suis le concepteur, vous trouverez des cours et des tutoriels adaptés aux sciences de l’ingénieur.

J'aime, Je partage.

Gérard KESSE

[GitHub](https://github.com/gkesse) | [LinkedIn](https://www.linkedin.com/in/tia-gerard-kesse/) | [SiteWeb](http://readydev.freeboxos.fr/)

[](http://readydev.freeboxos.fr/)

# Sommaire

[Auteur 2](#_Toc511210461)

[Sommaire 3](#_Toc511210462)

[Introduction 4](#_Toc511210463)

[Installation sous Windows avec QMake 4](#_Toc511210464)

[Téléchargements 4](#_Toc511210465)

[Installation de Notepad++ 4](#_Toc511210466)

[Installation de QMake 4](#_Toc511210467)

[Compiler un projet C++ avec QMake 5](#_Toc511210468)

[Compiler un projet C++ avec un seul fichier source 5](#_Toc511210469)

[Compiler un projet C++ avec plusieurs fichiers sources 8](#_Toc511210470)

[Gérer les librairies statiques avec QMake 16](#_Toc511210471)

[Créer une librairie statique en C++ 16](#_Toc511210472)

[Utiliser une librairie statique en C++ 23](#_Toc511210473)

[Gérer les librairies dynamiques avec QMake 26](#_Toc511210474)

[Créer une librairie dynamique en C++ 26](#_Toc511210475)

[Utiliser une librairie dynamique en C++ 33](#_Toc511210476)

[Gérer un projet complet avec QMake 36](#_Toc511210477)

[Compiler un projet complet C++ avec QMake 36](#_Toc511210478)

# [Introduction](#_Sommaire)

Le C++ est un langage de programmation orienté objet. QMake est un outil de compilation de projet C++ fourni par Qt Creator. Le but de ce tutoriel est de vous apprendre à compiler un projet C++ avec QMake.

**Prérequis :**

Aucun prérequis n’est nécessaire.

# [Installation sous Windows avec QMake](#_Sommaire)

Qt Creator est un environnement de développement intégré multiplateforme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++. Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git.

## [Téléchargements](#_Sommaire)

**Notepad++ :**

<https://notepad-plus-plus.org/fr/>

**Qt Creator :**

<https://download.qt.io/archive/qt/>

## [Installation de Notepad++](#_Sommaire)

**Plugins Notepad++ :**

TextFX

NppExport

# [Compiler un projet C++ avec QMake](#_Sommaire)

## [Compiler un projet C++ avec un seul fichier source](#_Sommaire)

**Objectif :**

Compiler un projet C++ avec un seul fichier source.

Créer un fichier source en C++.

Créer un fichier de construction Makefile.

**Implémentation :**

Afficher un message dans la console (Bonjour tout le monde).

**Résultat :**



**Dossier projet :**

src/main.cpp

src/GMain.pri

win/GProject.pro

win/cmd\_build.bat

win/cmd\_clean.bat

win/cmd\_qmake.bat

win/cmd\_compile.bat

win/cmd\_run.bat

win/bin/

win/build/

**src/main.cpp**

//===============================================

#include <iostream>

//===============================================

**using** **namespace** std**;**

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

cout **<<** "Bonjour tout le monde\n"**;**

**return** 0**;**

**}**

//===============================================

**src/GMain.pri**

SOURCES += \

$$PWD/main.cpp

**win/GProject.pro**

TEMPLATE = app

TARGET = GProject

DESTDIR = bin

OBJECTS\_DIR = build

MOC\_DIR = build

UI\_DIR = build

RCC\_DIR = build

CONFIG += release console

include(../src/GMain.pri)

**win/cmd\_build.bat**

@**echo** off

**call** cmd\_clean.bat

**call** cmd\_qmake.bat

**call** cmd\_compile.bat

**call** cmd\_run.bat

**win/cmd\_clean.bat**

@**echo** off

**del** /q bin\**\***

**del** /q build\**\***

**del** /q Makefile**\***

**win/cmd\_qmake.bat**

@**echo** off

**set** PATH**=**C:\Qt\Qt5.8.0\5.8\mingw53\_32\bin

qmake -r -spec win32-g**++**

**win/cmd\_compile.bat**

@**echo** off

**set** PATH**=**C:\Qt\Qt5.8.0\Tools\mingw530\_32\bin

mingw32-make

**win/cmd\_run.bat**

@**echo** off

**set** PATH**=**C:\Qt\Qt5.8.0\Tools\mingw530\_32\bin

**echo**.

bin\GProject.exe

**echo**.

**pause**

**Nettoyage du projet :**

cmd\_clean.bat

**Génération du Makefile :**

cmd\_qmake.bat

**Compilation du projet :**

cmd\_compile.bat

**Exécution du projet :**

cmd\_run.bat

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q bin\\*

del /q build\\*

del /q Makefile\*

qmake -r -spec win32-g++

mingw32-make

## [Compiler un projet C++ avec plusieurs fichiers sources](#_Sommaire)

**Objectif :**

Compiler un projet C++ avec plusieurs fichiers sources.

Créer un fichier entête en C++.

Créer une classe en C++.

Créer un patron Singleton en C++.

**Implémentation :**

Créer une classe (GMath) permettant de réaliser des opérations arithmétiques sur deux nombres données, Additionner (add()), Soustraire (substract()), Multiplier (multiply()), Diviser (divide()), Déterminer le Quotient (quotient()), Déterminer le Modulo (modulo()).

Créer une classe (GShow) permettant d’afficher des données dans la console (show()).

Utiliser la classe (GMath) pour réaliser des opérations arithmétiques sur deux nombres données (22, 7). Utiliser la classe (GShow) pour afficher les résultats de calculs.

**Résultat :**



**Dossier projet :**

src/main.cpp

src/GMain.pri

src/manager/GMath.h

src/manager/GMath.cpp

src/manager/GSow.h

src/manager/GShow.cpp

src/manager/GManager.pri

win/GProject.pro

win/bin/

win/build/

**src/main.cpp**

//===============================================

#include "GShow.h"

#include "GMath.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

double m\_data**;**

GShow**::**Instance**()->**show**(**"### Operations arithmetiques"**);**

m\_data **=** GMath**::**Instance**()->**add**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "add(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**subtract**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "subtract(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**multiply**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "multiply(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**divide**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "divide(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**quotient**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "quotient(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**modulo**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "modulo(22, 7)"**);**

**return** 0**;**

**}**

//===============================================

**src/GMain.pri**

include(manager/GManager.pri)

SOURCES += \

$$PWD/main.cpp

**src/manager/GMath.h**

//===============================================

#ifndef \_GMath\_

#define \_GMath\_

//===============================================

#include <cmath>

//===============================================

class GMath **{**

private**:**

GMath**();**

public**:**

**~**GMath**();**

public**:**

static GMath**\*** Instance**();**

public**:**

double add**(**const double**&** a**,** const double**&** b**);**

double subtract**(**const double**&** a**,** const double**&** b**);**

double multiply**(**const double**&** a**,** const double**&** b**);**

double divide**(**const double**&** a**,** const double**&** b**);**

double quotient**(**const double**&** a**,** const double**&** b**);**

double modulo**(**const int**&** a**,** const int**&** b**);**

private**:**

static GMath**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GMath.cpp**

//===============================================

#include "GMath.h"

//===============================================

GMath**\*** GMath**::**m\_instance **=** 0**;**

//===============================================

GMath**::**GMath**()** **{**

**}**

//===============================================

GMath**::~**GMath**()** **{**

**}**

//===============================================

GMath**\*** GMath**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GMath**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

double GMath**::**add**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **+** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**subtract**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **-** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**multiply**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **\*** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**divide**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **/** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**quotient**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** floor**(**divide**(**a**,** b**));**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**modulo**(**const int**&** a**,** const int**&** b**)** **{**

int m\_data **=** a **%** b**;**

**return** double**(**m\_data**);**

**}**

//===============================================

**src/manager/GShow.h**

//===============================================

#ifndef \_GShow\_

#define \_GShow\_

//===============================================

#include <iostream>

#include <iomanip>

#include <string>

//===============================================

**using** **namespace** std**;**

//===============================================

class GShow **{**

private**:**

GShow**();**

public**:**

**~**GShow**();**

public**:**

static GShow**\*** Instance**();**

public**:**

void show**(**const string**&** data**,** const string**&** name **=** ""**);**

void show**(**const double**&** data**,** const string**&** name **=** ""**);**

private**:**

static GShow**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GShow.cpp**

//===============================================

#include "GShow.h"

//===============================================

GShow**\*** GShow**::**m\_instance **=** 0**;**

//===============================================

GShow**::**GShow**()** **{**

**}**

//===============================================

GShow**::~**GShow**()** **{**

**}**

//===============================================

GShow**\*** GShow**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GShow**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

void GShow**::**show**(**const string**&** data**,** const string**&** name**)** **{**

**if(**name **!=** ""**)** cout **<<** name **<<** " : "**;**

cout **<<** data **<<** "\n"**;**

**}**

//===============================================

void GShow**::**show**(**const double**&** data**,** const string**&** name**)** **{**

**if(**name **!=** ""**)** cout **<<** name **<<** " : "**;**

cout **<<** fixed **<<** setprecision**(**2**)** **<<** data **<<** "\n"**;**

**}**

//===============================================

**src/manager/GManager.pri**

INCLUDEPATH += \

$$PWD

HEADERS += \

$$PWD/GMath.h \

$$PWD/GShow.h

SOURCES += \

$$PWD/GMath.cpp \

$$PWD/GShow.cpp

**win/GProject.pro**

TEMPLATE = app

TARGET = GProject

DESTDIR = bin

OBJECTS\_DIR = build

MOC\_DIR = build

UI\_DIR = build

RCC\_DIR = build

CONFIG += release console

include(../src/GMain.pri)

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q bin\\*

del /q build\\*

del /q Makefile\*

qmake -r -spec win32-g++

mingw32-make

# [Gérer les librairies statiques avec QMake](#_Sommaire)

## [Créer une librairie statique en C++](#_Sommaire)

**Objectif :**

Créer une librairie statique en C++.

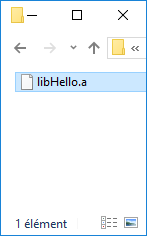
**Implémentation :**

Créer une classe (GMath) permettant de réaliser des opérations arithmétiques sur deux nombres données, Additionner (add()), Soustraire (substract()), Multiplier (multiply()), Diviser (divide()), Déterminer le Quotient (quotient()), Déterminer le Modulo (modulo()).

Créer une classe (GShow) permettant d’afficher des données dans la console (show()).

Regrouper ces deux classes dans une librairie statique (libHello.a).

**Résultat :**



**Dossier projet :**

src/GMain.pri

src/manager/GMath.h

src/manager/GMath.cpp

src/manager/GSow.h

src/manager/GShow.cpp

src/manager/GManager.pri

win/GProject.pro

win/bin/

win/build/

**src/GMain.pri**

include(manager/GManager.pri)

**src/manager/GMath.h**

//===============================================

#ifndef \_GMath\_

#define \_GMath\_

//===============================================

#include <cmath>

//===============================================

class GMath **{**

private**:**

GMath**();**

public**:**

**~**GMath**();**

public**:**

static GMath**\*** Instance**();**

public**:**

double add**(**const double**&** a**,** const double**&** b**);**

double subtract**(**const double**&** a**,** const double**&** b**);**

double multiply**(**const double**&** a**,** const double**&** b**);**

double divide**(**const double**&** a**,** const double**&** b**);**

double quotient**(**const double**&** a**,** const double**&** b**);**

double modulo**(**const int**&** a**,** const int**&** b**);**

private**:**

static GMath**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GMath.cpp**

//===============================================

#include "GMath.h"

//===============================================

GMath**\*** GMath**::**m\_instance **=** 0**;**

//===============================================

GMath**::**GMath**()** **{**

**}**

//===============================================

GMath**::~**GMath**()** **{**

**}**

//===============================================

GMath**\*** GMath**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GMath**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

double GMath**::**add**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **+** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**subtract**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **-** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**multiply**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **\*** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**divide**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **/** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**quotient**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** floor**(**divide**(**a**,** b**));**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**modulo**(**const int**&** a**,** const int**&** b**)** **{**

int m\_data **=** a **%** b**;**

**return** double**(**m\_data**);**

**}**

//===============================================

**src/manager/GShow.h**

//===============================================

#ifndef \_GShow\_

#define \_GShow\_

//===============================================

#include <iostream>

#include <iomanip>

#include <string>

//===============================================

**using** **namespace** std**;**

//===============================================

class GShow **{**

private**:**

GShow**();**

public**:**

**~**GShow**();**

public**:**

static GShow**\*** Instance**();**

public**:**

void show**(**const string**&** data**,** const string**&** name **=** ""**);**

void show**(**const double**&** data**,** const string**&** name **=** ""**);**

private**:**

static GShow**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GShow.cpp**

//===============================================

#include "GShow.h"

//===============================================

GShow**\*** GShow**::**m\_instance **=** 0**;**

//===============================================

GShow**::**GShow**()** **{**

**}**

//===============================================

GShow**::~**GShow**()** **{**

**}**

//===============================================

GShow**\*** GShow**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GShow**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

void GShow**::**show**(**const string**&** data**,** const string**&** name**)** **{**

**if(**name **!=** ""**)** cout **<<** name **<<** " : "**;**

cout **<<** data **<<** "\n"**;**

**}**

//===============================================

void GShow**::**show**(**const double**&** data**,** const string**&** name**)** **{**

**if(**name **!=** ""**)** cout **<<** name **<<** " : "**;**

cout **<<** fixed **<<** setprecision**(**2**)** **<<** data **<<** "\n"**;**

**}**

//===============================================

**win/manager/GManager.pri**

INCLUDEPATH += \

$$PWD

HEADERS += \

$$PWD/GMath.h \

$$PWD/GShow.h

SOURCES += \

$$PWD/GMath.cpp \

$$PWD/GShow.cpp

**win/GProject.pro**

TEMPLATE = lib

TARGET = Hello

DESTDIR = bin

OBJECTS\_DIR = build

MOC\_DIR = build

UI\_DIR = build

RCC\_DIR = build

CONFIG += release console staticlib

include(../src/GMain.pri)

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q bin\\*

del /q build\\*

del /q Makefile\*

qmake -r -spec win32-g++

mingw32-make

## [Utiliser une librairie statique en C++](#_Sommaire)

**Objectif :**

Utiliser une librairie statique en C++.

**Implémentation :**

Utiliser la classe (GMath) de la librairie statique (libHello.a) pour réaliser des opérations arithmétiques sur deux nombres données (22, 7).

Utiliser la classe (GShow) de la librairie statique (libHello.a) pour afficher les résultats de calculs.

**Résultat :**



**Dossier projet :**

src/main.cpp

win/Makeifle

win/bin/

win/build/

**Dossier librairie :**

win/lib/

win/lib/Hello/include/GMath.h

win/lib/Hello/include/GShow.h

win/lib/Hello/lib/libHello.a

**src/main.cpp**

//===============================================

#include "GShow.h"

#include "GMath.h"

//===============================================

**using** **namespace** std**;**

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

double m\_data**;**

GShow**::**Instance**()->**show**(**"### Operations arithmetiques\n"**);**

m\_data **=** GMath**::**Instance**()->**add**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "add(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**subtract**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "subtract(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**multiply**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "multiply(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**divide**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "divide(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**quotient**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "quotient(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**modulo**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "modulus(22, 7)"**);**

**return** 0**;**

**}**

//===============================================

**win/Makefile**

GSRC **=** ../src

GBIN **=** bin

GBUILD **=** build

GTARGET **=** $(GBIN)/GProject.exe

GINCS **=** \

-Ilib/Hello/include

GLIBS **=** \

-Llib/Hello/lib -lHello

GOBJS **=** \

$(GBUILD)/main.o

all**:** $(GOBJS)

g++ -o $(GTARGET) $(GOBJS) $(GLIBS)

$(GBUILD)/main.o**:** $(GSRC)/main.cpp

g++ -c $(GSRC)/main.cpp -o $(GBUILD)/main.o $(GINCS)

clean**:**

del /q $(GBUILD)\\* $(GBIN)\\*

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

g++ -c ../src/main.cpp -o build/main.o -Ilib/Hello/include

g++ -o bin/GProject.exe build/main.o -Llib/Hello/lib –lHello

# [Gérer les librairies dynamiques avec QMake](#_Sommaire)

## [Créer une librairie dynamique en C++](#_Sommaire)

**Objectif :**

Créer une librairie dynamique en C++.

**Implémentation :**

Créer une classe (GMath) permettant de réaliser des opérations arithmétiques sur deux nombres données, Additionner (add()), Soustraire (substract()), Multiplier (multiply()), Diviser (divide()), Déterminer le Quotient (quotient()), Déterminer le Modulo (modulo()).

Créer une classe (GShow) permettant d’afficher des données dans la console (show()).

Regrouper ces deux classes dans une librairie dynamique (libHello.dll).

**Résultat :**



**Dossier projet :**

src/manager/GMath.h

src/manager/GMath.cpp

src/manager/GSow.h

src/manager/GShow.cpp

win/Makefile

win/bin/

win/build/

**src/manager/GMath.h**

//===============================================

#ifndef \_GMath\_

#define \_GMath\_

//===============================================

#include <cmath>

//===============================================

#ifdef DLL\_APP

#define DLL\_API \_\_declspec(dllexport)

#else

#define DLL\_API \_\_declspec(dllimport)

#endif

//===============================================

class DLL\_API GMath **{**

private**:**

GMath**();**

public**:**

**~**GMath**();**

public**:**

static GMath**\*** Instance**();**

public**:**

double add**(**const double**&** a**,** const double**&** b**);**

double subtract**(**const double**&** a**,** const double**&** b**);**

double multiply**(**const double**&** a**,** const double**&** b**);**

double divide**(**const double**&** a**,** const double**&** b**);**

double quotient**(**const double**&** a**,** const double**&** b**);**

double modulo**(**const int**&** a**,** const int**&** b**);**

private**:**

static GMath**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GMath.cpp**

//===============================================

#include "GMath.h"

//===============================================

GMath**\*** GMath**::**m\_instance **=** 0**;**

//===============================================

GMath**::**GMath**()** **{**

**}**

//===============================================

GMath**::~**GMath**()** **{**

**}**

//===============================================

GMath**\*** GMath**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GMath**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

double GMath**::**add**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **+** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**subtract**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **-** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**multiply**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **\*** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**divide**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** a **/** b**;**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**quotient**(**const double**&** a**,** const double**&** b**)** **{**

double m\_data **=** floor**(**divide**(**a**,** b**));**

**return** m\_data**;**

**}**

//===============================================

double GMath**::**modulo**(**const int**&** a**,** const int**&** b**)** **{**

int m\_data **=** a **%** b**;**

**return** double**(**m\_data**);**

**}**

//===============================================

**src/manager/GShow.h**

//===============================================

#ifndef \_GShow\_

#define \_GShow\_

//===============================================

#include <iostream>

#include <iomanip>

#include <string>

//===============================================

#ifdef DLL\_APP

#define DLL\_API \_\_declspec(dllexport)

#else

#define DLL\_API \_\_declspec(dllimport)

#endif

//===============================================

**using** **namespace** std**;**

//===============================================

class DLL\_API GShow **{**

private**:**

GShow**();**

public**:**

**~**GShow**();**

public**:**

static GShow**\*** Instance**();**

public**:**

void show**(**const string**&** data**,** const string**&** name **=** ""**);**

void show**(**const double**&** data**,** const string**&** name **=** ""**);**

private**:**

static GShow**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GShow.cpp**

//===============================================

#include "GShow.h"

//===============================================

GShow**\*** GShow**::**m\_instance **=** 0**;**

//===============================================

GShow**::**GShow**()** **{**

**}**

//===============================================

GShow**::~**GShow**()** **{**

**}**

//===============================================

GShow**\*** GShow**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GShow**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

void GShow**::**show**(**const string**&** data**,** const string**&** name**)** **{**

**if(**name **!=** ""**)** cout **<<** name **<<** " : "**;**

cout **<<** data **<<** "\n"**;**

**}**

//===============================================

void GShow**::**show**(**const double**&** data**,** const string**&** name**)** **{**

**if(**name **!=** ""**)** cout **<<** name **<<** " : "**;**

cout **<<** fixed **<<** setprecision**(**2**)** **<<** data **<<** "\n"**;**

**}**

//===============================================

**win/Makefile**

GSRC **=** ../src

GBIN **=** bin

GBUILD **=** build

GTARGET **=** $(GBIN)/libHello.dll

GINCS **=** \

-I../src/manager

GOBJS **=** \

$(GBUILD)/GShow.o \

$(GBUILD)/GMath.o

all**:** $(GOBJS)

g++ -shared -o $(GTARGET) $(GOBJS)

$(GBUILD)/GShow.o**:** $(GSRC)/manager/GShow.cpp

g++ -c -DDLL\_APP $(GSRC)/manager/GShow.cpp -o $(GBUILD)/GShow.o $(GINCS)

$(GBUILD)/GMath.o**:** $(GSRC)/manager/GMath.cpp

g++ -c -DDLL\_APP $(GSRC)/manager/GMath.cpp -o $(GBUILD)/GMath.o $(GINCS)

clean**:**

del /q $(GBUILD)\\* $(GBIN)\\*

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

g++ -c -DDLL\_APP ../src/manager/GShow.cpp -o build/GShow.o -I../src/manager

g++ -c -DDLL\_APP ../src/manager/GMath.cpp -o build/GMath.o -I../src/manager

g++ -shared -o bin/libHello.dll build/GShow.o build/GMath.o

## [Utiliser une librairie dynamique en C++](#_Sommaire)

**Objectif :**

Utiliser une librairie dynamique en C++.

**Implémentation :**

Utiliser la classe (GMath) de la librairie dynamique (libHello.dll) pour réaliser des opérations arithmétiques sur deux nombres données (22, 7).

Utiliser la classe (GShow) de la librairie dynamique (libHello.dll) pour afficher les résultats de calculs.

**Résultat :**



**Dossier projet :**

src/main.cpp

win/Makeifle

win/bin/

win/build/

**Dossier librairie :**

win/lib/

win/lib/Hello/include/GMath.h

win/lib/Hello/include/GShow.h

win/lib/Hello/bin/libHello.dll

**src/main.cpp**

//===============================================

#include "GShow.h"

#include "GMath.h"

//===============================================

**using** **namespace** std**;**

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

double m\_data**;**

GShow**::**Instance**()->**show**(**"### Operations arithmetiques\n"**);**

m\_data **=** GMath**::**Instance**()->**add**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "add(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**subtract**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "subtract(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**multiply**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "multiply(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**divide**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "divide(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**quotient**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "quotient(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**modulo**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "modulus(22, 7)"**);**

**return** 0**;**

**}**

//===============================================

**win/Makefile**

GSRC **=** ../src

GBIN **=** bin

GBUILD **=** build

GTARGET **=** $(GBIN)/GProject.exe

GINCS **=** \

-Ilib/Hello/include

GLIBS **=** \

-Llib/Hello/bin -lHello

GOBJS **=** \

$(GBUILD)/main.o

all**:** $(GOBJS)

g++ -o $(GTARGET) $(GOBJS) $(GLIBS)

$(GBUILD)/main.o**:** $(GSRC)/main.cpp

g++ -c $(GSRC)/main.cpp -o $(GBUILD)/main.o $(GINCS)

clean**:**

del /q $(GBUILD)\\* $(GBIN)\\*

**win/cmd\_run.bat**

@**echo** off

**set** PATH**=**C:\QMake\bin

**set** PATH**=**lib\Hello\bin;**%PATH%**

**echo**.

bin\GProject.exe

**echo**.

**pause**

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

g++ -c ../src/main.cpp -o build/main.o -Ilib/Hello/include

g++ -o bin/GProject.exe build/main.o -Llib/Hello/bin -lHello

# [Gérer un projet complet avec QMake](#_Sommaire)

## [Compiler un projet complet C++ avec QMake](#_Sommaire)

**Objectif :**

Compiler un projet complet C++ avec QMake.

Utiliser un fichier entête en C++.

Utiliser une librairie dynamique en C++.

Utiliser un fichier de construction Makefile.

**Implémentation :**

Créer une classe (GProcess) utilisant la classe (GMath) de la librairie dynamique (libHello.dll) pour réaliser des opérations arithmétiques sur deux nombres données (22, 7) et la classe (GShow) de la librairie dynamique (libHello.dll) pour afficher les résultats de calculs.

**Résultat :**



**Dossier projet :**

src/main.cpp

src/manager/GProcess.h

src/manager/GProcess.cpp

win/Makeifle

win/bin/

win/build/

**Dossier librairie :**

win/lib/

win/lib/Hello/include/GMath.h

win/lib/Hello/include/GShow.h

win/lib/Hello/bin/libHello.dll

**src/main.cpp**

//===============================================

#include "GProcess.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

GProcess**::**Instance**()->**run**();**

**return** 0**;**

**}**

//===============================================

**src/manager/GProcess.h**

//===============================================

#ifndef \_GProcess\_

#define \_GProcess\_

//===============================================

class GProcess **{**

private**:**

GProcess**();**

public**:**

**~**GProcess**();**

public**:**

static GProcess**\*** Instance**();**

public**:**

void run**();**

private**:**

static GProcess**\*** m\_instance**;**

**};**

//===============================================

#endif

//===============================================

**src/manager/GProcess.cpp**

//===============================================

#include "GProcess.h"

#include "GShow.h"

#include "GMath.h"

//===============================================

GProcess**\*** GProcess**::**m\_instance **=** 0**;**

//===============================================

GProcess**::**GProcess**()** **{**

**}**

//===============================================

GProcess**::~**GProcess**()** **{**

**}**

//===============================================

GProcess**\*** GProcess**::**Instance**()** **{**

**if(**m\_instance **==** 0**)** **{**

m\_instance **=** **new** GProcess**;**

**}**

**return** m\_instance**;**

**}**

//===============================================

void GProcess**::**run**()** **{**

double m\_data**;**

GShow**::**Instance**()->**show**(**"### Operations arithmetiques\n"**);**

m\_data **=** GMath**::**Instance**()->**add**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "add(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**subtract**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "subtract(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**multiply**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "multiply(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**divide**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "divide(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**quotient**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "quotient(22, 7)"**);**

m\_data **=** GMath**::**Instance**()->**modulo**(**22**,** 7**);**

GShow**::**Instance**()->**show**(**m\_data**,** "modulus(22, 7)"**);**

**}**

//===============================================

**win/Makefile**

GSRC **=** ../src

GBIN **=** bin

GBUILD **=** build

GTARGET **=** $(GBIN)/GProject.exe

GINCS **=** \

-I../src/manager \

-Ilib/Hello/include

GLIBS **=** \

-Llib/Hello/bin -lHello

GOBJS **=** \

$(patsubst $(GSRC)/%.cpp, $(GBUILD)/%.o, $(wildcard $(GSRC)/\*.cpp)) \

$(patsubst $(GSRC)/manager/%.cpp, $(GBUILD)/%.o, $(wildcard $(GSRC)/manager/\*.cpp))

all**:** $(GOBJS)

g++ -o $(GTARGET) $(GOBJS) $(GLIBS)

$(GBUILD)/%.o**:** $(GSRC)/%.cpp

g++ -c $< -o $@ $(GINCS)

$(GBUILD)/%.o**:** $(GSRC)/manager/%.cpp

g++ -c $< -o $@ $(GINCS)

clean**:**

del /q $(GBUILD)\\* $(GBIN)\\*

**win/cmd\_run.bat**

@**echo** off

**set** PATH**=**C:\QMake\bin

**set** PATH**=**lib\Hello\bin;**%PATH%**

**echo**.

bin\GProject.exe

**echo**.

**pause**

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

g++ -c ../src/main.cpp -o build/main.o -I../src/manager -Ilib/Hello/include

g++ -c ../src/manager/GProcess.cpp -o build/GProcess.o -I../src/manager -Ilib/Hello/include

g++ -o bin/GProject.exe build/main.o build/GProcess.o -Llib/Hello/bin –lHello