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

OpenGL

Apprendre la programmation 3D en C++ avec OpenGL

# [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](#_Toc511373407)

[Sommaire 3](#_Toc511373408)

[Introduction 4](#_Toc511373409)

[Installation sous Windows avec MinGW 4](#_Toc511373410)

[Téléchargements 4](#_Toc511373411)

[Installation de Notepad++ 4](#_Toc511373412)

[Installation de MinGW 5](#_Toc511373413)

[Installation de FreeGLUT 5](#_Toc511373414)

[Compiler un projet C++ avec NMake 7](#_Toc511373415)

[Compiler un projet C++ avec un seul fichier source 7](#_Toc511373416)

[Compiler un projet C++ avec plusieurs fichiers sources 11](#_Toc511373417)

[Gérer les librairies statiques avec NMake 19](#_Toc511373418)

[Créer une librairie statique en C++ 19](#_Toc511373419)

[Utiliser une librairie statique en C++ 26](#_Toc511373420)

[Gérer les librairies dynamiques avec NMake 29](#_Toc511373421)

[Créer une librairie dynamique en C++ 29](#_Toc511373422)

[Utiliser une librairie dynamique en C++ 36](#_Toc511373423)

[Gérer un projet complet avec NMake 39](#_Toc511373424)

[Compiler un projet complet C++ avec NMake 39](#_Toc511373425)

# [Introduction](#_Sommaire)

Le C++ est un langage de programmation orienté objet. OpenGL est une bibliothèque de création d’applications 3D. Le but de ce tutoriel est de vous apprendre la programmation 3D en C++ avec OpenGL.

**Prérequis :**

Apprendre à compiler un projet C++ avec QMake.

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

## [Téléchargements](#_Sommaire)

**Notepad++ :**

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

**MinGW :**

<http://mingw.org/>

**Qt Creator :**

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

**CMake :**

<https://cmake.org/>

**FreeGLUT :**

<http://freeglut.sourceforge.net/>

**Bullet :**

<https://github.com/bulletphysics/bullet3>

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

**Plugins Notepad++ :**

TextFX

NppExport

## [Installation de MinGW](#_Sommaire)

**Packages MinGW :**

mingw32-base

mingw32-gcc-g++

## [Installation de FreeGLUT](#_Sommaire)

**Dossier construction :**

FreeGLUT/cmd\_cmake.bat

FreeGLUT/cmd\_build.bat

FreeGLUT/build/

**Démarrer CMake :**

FreeGLUT/cmd\_cmake.bat

-----------------------------------------------------

**set** PATH**=**C:\Program Files\CMake\bin;

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

cmake-gui

**pause**

**Configurer FreeGLUT avec CMake :**

Where is the source code

C:/Users/gerar/Downloads/OpenGL/freeglut-3.0.0/freeglut-3.0.0

Where to build the binaries

C:/Users/gerar/Downloads/OpenGL/freeglut-3.0.0/freeglut-3.0.0/build

Configure

Specify the generator for this project

MinGW Makefiles

Specify native compilers

Next

Compilers

C

C:/Qt/Qt5.8.0/Tools/mingw530\_32/bin/gcc.exe

C++

C:/Qt/Qt5.8.0/Tools/mingw530\_32/bin/g++.exe

Finish

Configure

Configure

Generate

**Construire FreeGLUT avec MinGW :**

FreeGLUT/cmd\_build.bat

-----------------------------------------------------

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

**cd** build

mingw32-make

**pause**

**Dossier librairie FreeGLUT :**

FreeGLUT/include/

FreeGLUT/build/lib/

FreeGLUT/build/bin/

# [Créer une fenêtre avec FreeGLUT](#_Sommaire)

## [Afficher une fenêtre](#_Sommaire)

**Objectif :**

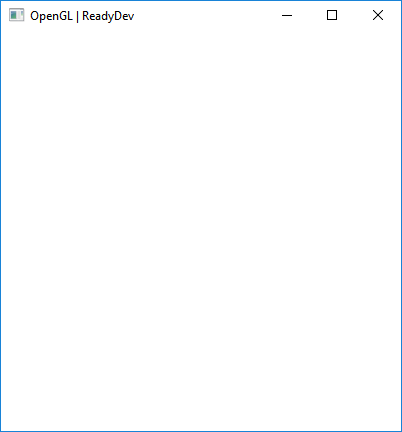
Afficher une fenêtre.

**Implémentation :**

Créer une classe (GWindow).

Afficher une fenêtre (show()).

**Résultat :**



**Dossier projet :**

src/main.cpp

win/GWindow.h

win/GWindow.cpp

win/bin/

win/build/

**src/main.cpp**

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

#include "GWindow.h"

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

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

GWindow**::**Instance**()->**show**(&**argc**,** argv**);**

**return** 0**;**

**}**

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

**src/GWindow.h**

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

#ifndef \_GWindow\_

#define \_GWindow\_

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

#include <GL/freeglut.h>

#include <QString>

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

class GWindow **{**

public**:**

GWindow**();**

**~**GWindow**();**

public**:**

static GWindow**\*** Instance**();**

void show**(**int**\*** argc**,** char**\*\*** argv**);**

static void runDisplay**();**

private**:**

static GWindow**\*** m\_instance**;**

int m\_x**;**

int m\_y**;**

int m\_w**;**

int m\_h**;**

QString m\_title**;**

**};**

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

#endif

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

**src/ GWindow.cpp**

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

#include "GWindow.h"

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

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

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

GWindow**::**GWindow**()** **{**

m\_x **=** 0**;**

m\_y **=** 0**;**

m\_w **=** 400**;**

m\_h **=** 400**;**

m\_title **=** "OpenGL | ReadyDev"**;**

**}**

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

GWindow**::~**GWindow**()** **{**

**}**

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

GWindow**\*** GWindow**::**Instance**()** **{**

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

m\_instance **=** **new** GWindow**;**

**}**

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

**}**

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

void GWindow**::**show**(**int**\*** argc**,** char**\*\*** argv**)** **{**

glutInit**(**argc**,** argv**);**

glutInitDisplayMode**(**GLUT\_DOUBLE **|** GLUT\_RGBA **|** GLUT\_DEPTH**);**

glutInitWindowPosition**(**m\_x**,** m\_y**);**

glutInitWindowSize**(**m\_w**,** m\_h**);**

glutCreateWindow**(**m\_title**.**toStdString**().**c\_str**());**

glutSetOption **(**GLUT\_ACTION\_ON\_WINDOW\_CLOSE**,** GLUT\_ACTION\_GLUTMAINLOOP\_RETURNS**);**

glutDisplayFunc**(**runDisplay**);**

glutMainLoop**();**

**}**

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

void GWindow**::**runDisplay**()** **{**

**}**

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

## [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/manager/GMath.h

src/manager/GMath.cpp

src/manager/GSow.h

src/manager/GShow.cpp

win/Makefile

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/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/Makefile**

GSRC **=** ../src

GBIN **=** bin

GBUILD **=** build

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

GINCS **=** \

/I../src/manager

GOBJS **=** \

$(GBUILD)/main.obj \

$(GBUILD)/GShow.obj \

$(GBUILD)/GMath.obj

all**:** $(GOBJS)

link /OUT:$(GTARGET) $(GOBJS)

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

cl /c /EHsc $(GSRC)/main.cpp /Fo$(GBUILD)/main.obj $(GINCS)

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

cl /c /EHsc $(GSRC)/manager/GShow.cpp /Fo$(GBUILD)/GShow.obj $(GINCS)

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

cl /c /EHsc $(GSRC)/manager/GMath.cpp /Fo$(GBUILD)/GMath.obj $(GINCS)

clean**:**

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

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

cl /c /EHsc ../src/main.cpp /Fobuild/main.obj /I../src/manager

cl /c /EHsc ../src/manager/GShow.cpp /Fobuild/GShow.obj /I../src/manager

cl /c /EHsc ../src/manager/GMath.cpp /Fobuild/GMath.obj /I../src/manager

link /OUT:bin/GProject.exe build/main.obj build/GShow.obj build/GMath.obj

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

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

**Objectif :**

Créer une librairie statique en C++.

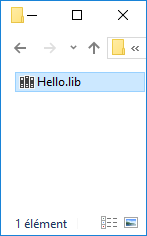
**Implémentation :**

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/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>

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

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/Makefile**

GSRC **=** ../src

GBIN **=** bin

GBUILD **=** build

GTARGET **=** $(GBIN)/Hello.lib

GINCS **=** \

/I../src/manager

GOBJS **=** \

$(GBUILD)/GShow.obj \

$(GBUILD)/GMath.obj

all**:** $(GOBJS)

lib /OUT:$(GTARGET) $(GOBJS)

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

cl /c /EHsc $(GSRC)/manager/GShow.cpp /Fo$(GBUILD)/GShow.obj $(GINCS)

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

cl /c /EHsc $(GSRC)/manager/GMath.cpp /Fo$(GBUILD)/GMath.obj $(GINCS)

clean**:**

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

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

cl /c /EHsc ../src/manager/GShow.cpp /Fobuild/GShow.obj /I../src/manager

cl /c /EHsc ../src/manager/GMath.cpp /Fobuild/GMath.obj /I../src/manager

lib /OUT:bin/Hello.lib build/GShow.obj build/GMath.obj

## [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/Hello.lib

**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 **=** \

/LIBPATH**:**lib/hello/lib Hello.lib

GOBJS **=** \

$(GBUILD)/main.obj

all**:** $(GOBJS)

link /OUT:$(GTARGET) $(GOBJS) $(GLIBS)

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

cl /c /EHsc $(GSRC)/main.cpp /Fo$(GBUILD)/main.obj $(GINCS)

clean**:**

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

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

cl /c /EHsc ../src/main.cpp /Fobuild/main.obj /Ilib/Hello/include

link /OUT:bin/GProject.exe build/main.obj /LIBPATH:lib/hello/lib Hello.lib

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

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

**Objectif :**

Créer une librairie dynamique en C++.

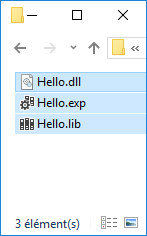
**Implémentation :**

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)/Hello.dll

GINCS **=** \

/I../src/manager

GOBJS **=** \

$(GBUILD)/GShow.obj \

$(GBUILD)/GMath.obj

all**:** $(GOBJS)

link /DLL /OUT:$(GTARGET) $(GOBJS)

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

cl /c /EHsc /DDLL\_APP $(GSRC)/manager/GShow.cpp /Fo$(GBUILD)/GShow.obj $(GINCS)

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

cl /c /EHsc /DDLL\_APP $(GSRC)/manager/GMath.cpp /Fo$(GBUILD)/GMath.obj $(GINCS)

clean**:**

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

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

cl /c /EHsc /DDLL\_APP ../src/manager/GShow.cpp /Fobuild/GShow.obj /I../src/manager

cl /c /EHsc /DDLL\_APP ../src/manager/GMath.cpp /Fobuild/GMath.obj /I../src/manager

link /DLL /OUT:bin/Hello.dll build/GShow.obj build/GMath.obj

## [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/lib/Hello.lib

win/lib/Hello/bin/Hello.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 **=** \

/LIBPATH**:**lib/Hello/lib Hello.lib

GOBJS **=** \

$(GBUILD)/main.obj

all**:** $(GOBJS)

link /OUT:$(GTARGET) $(GOBJS) $(GLIBS)

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

cl /c /EHsc $(GSRC)/main.cpp /Fo$(GBUILD)/main.obj $(GINCS)

clean**:**

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

**win/cmd\_run.bat**

@**echo** off

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

**echo**.

bin\GProject.exe

**echo**.

**pause**

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

cl /c /EHsc ../src/main.cpp /Fobuild/main.obj /Ilib/Hello/include

link /OUT:bin/GProject.exe build/main.obj /LIBPATH:lib/Hello/lib Hello.lib

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

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

**Objectif :**

Compiler un projet complet C++ avec NMake.

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/lib/Hello.lib

win/lib/Hello/bin/Hello.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 **=** \

/LIBPATH**:**lib/Hello/lib Hello.lib

GOBJS **=** \

$(GBUILD)/main.obj \

$(GBUILD)/GProcess.obj

all**:** $(GOBJS)

link /OUT:$(GTARGET) $\*\* $(GLIBS)

{$(GSRC)}.cpp{$(GBUILD)}.obj**:**

cl /c /EHsc $< /Fo$@ $(GINCS)

{$(GSRC)/manager}.cpp{$(GBUILD)}.obj**:**

cl /c /EHsc $< /Fo$@ $(GINCS)

clean**:**

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

**win/cmd\_run.bat**

@**echo** off

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

**echo**.

bin\GProject.exe

**echo**.

**pause**

**Construction du projet :**

cmd\_build.bat

**Construction manuel :**

del /q build\\* bin\\*

cl /c /EHsc ../src\main.cpp /Fobuild/main.obj /I../src/manager /Ilib/Hello/include

cl /c /EHsc ../src/manager\GProcess.cpp /Fobuild/GProcess.obj /I../src/manager /Ilib/Hello/include

link /OUT:bin/GProject.exe build/main.obj build/GProcess.obj /LIBPATH:lib/Hello/lib Hello.lib