

Plateforme de Développement Continu

Comprendre la Théorie pour mieux Pratiquer
Sciences de l'Ingénieur
Cours - Tutoriels

Cygwin

Apprendre à compiler un projet C++ avec Cygwin

J'aime, Je partage
Montez en Compétences

Auteur

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 | LinkedIn | SiteWeb



Sommaire

Auteur.....	2
Sommaire	3
Introduction	4
Installation sous Windows avec Cygwin.....	4
Téléchargements	4
Installation de Notepad++	4
Installation de Cygwin.....	4
Compiler un projet C++ avec Cygwin	5
Compiler un projet C++ avec un seul fichier source.....	5
Compiler un projet C++ avec plusieurs fichiers sources	8
Gérer les bibliothèques statiques avec Cygwin.....	16
Créer une bibliothèque statique en C++	16
Utiliser une bibliothèque statique en C++	23
Gérer les bibliothèques dynamiques avec Cygwin	26
Créer une bibliothèque dynamique en C++	26
Utiliser une bibliothèque dynamique en C++	33
Gérer un projet complet avec Cygwin.....	36
Compiler un projet complet C++ avec Cygwin	36

Introduction

Le C++ est un langage de programmation orienté objet. Cygwin est un outil de compilation de projet C++. Le but de ce tutoriel est de vous apprendre à compiler un projet C++ avec Cygwin.

Prérequis :

Aucun prérequis n'est nécessaire.

Installation sous Windows avec Cygwin

Téléchargements

Notepad++ :

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

Cygwin :

<http://www.cygwin.com/>

Installation de Notepad++

Plugins Notepad++ :

TextFX

NppExport

Installation de Cygwin

Packages Cygwin :

gcc-core : GNU Compiler Collection (C, OpenMP)

gcc-g++ : GNU Compiler Collection (C++)

gdb : The GNU Debugger

make : The GNU version of the 'make' utility

Compiler un projet C++ avec Cygwin

Compiler un projet C++ avec un seul fichier source

Objectif :

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

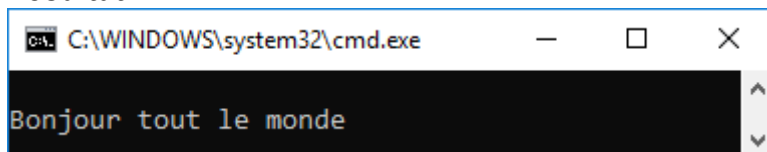
Créer un fichier source en C++.

Créer un fichier de construction Makefile.

Implémentation :

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

Résultat :

A screenshot of a Windows command prompt window. The title bar shows 'C:\WINDOWS\system32\cmd.exe'. The command prompt is open, and the text 'Bonjour tout le monde' is displayed on the screen.

Dossier projet :

src/main.cpp

win/Makefile

win/cmd_build.bat

win/cmd_clean.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;
}
//=====
```

win/Makefile

```
GSRC = ../src
GBIN = bin
GBUILD = build
GTARGET = $(GBIN)/GProject.exe

GOBJS = \
    $(GBUILD)/main.o

all: $(GOBJS)
    g++ -o $(GTARGET) $(GOBJS)
$(GBUILD)/main.o: $(GSRC)/main.cpp
    g++ -c $(GSRC)/main.cpp -o $(GBUILD)/main.o
clean:
    rm -f $(GBUILD)/* $(GBIN)/*
```

win/cmd_build.bat

```
@echo off

call cmd_clean.bat
call cmd_compile.bat
call cmd_run.bat
```

win/cmd_clean.bat

```
@echo off

set PATH=C:\cygwin\bin

make clean
```

win/cmd_compile.bat

```
@echo off

set PATH=C:\cygwin\bin

make
```

win/cmd_run.bat`@echo off``set PATH=C:\cygwin\bin``echo.``bin\GProject.exe``echo.``pause`**Nettoyage du projet :**`cmd_clean.bat`**Compilation du projet :**`cmd_compile.bat`**Exécution du projet :**`cmd_run.bat`**Construction du projet :**`cmd_build.bat`**Construction manuel :**`rm -f build/* bin/*``g++ -c ../src/main.cpp -o build/main.o``g++ -o bin/GProject.exe build/main.o`

Compiler un projet C++ avec plusieurs fichiers sources

Objectif :

Compiler un projet C++ avec plusieurs fichiers sources.

Créer un fichier entête en C++.

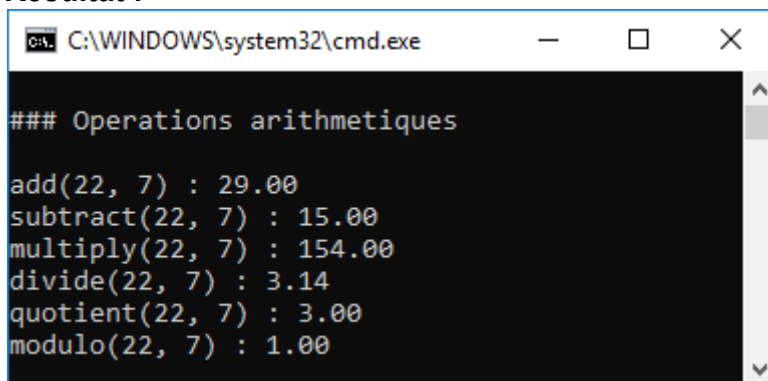
Créer une classe en C++.

Créer un patron Singleton en C++.

Implémentation :

Créer une classe (GMath) permettant de réaliser des opérations arithmétiques sur deux nombres données, Additionner (add()), Soustraire (subtract()), 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 :



```
C:\WINDOWS\system32\cmd.exe

### Operations arithmetiques

add(22, 7) : 29.00
subtract(22, 7) : 15.00
multiply(22, 7) : 154.00
divide(22, 7) : 3.14
quotient(22, 7) : 3.00
modulo(22, 7) : 1.00
```

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.o \
    $(GBUILD)/GShow.o \
    $(GBUILD)/GMath.o

all: $(GOBJS)
    g++ -o $(GTARGET) $(GOBJS)
$(GBUILD)/main.o: $(GSRC)/main.cpp
    g++ -c $(GSRC)/main.cpp -o $(GBUILD)/main.o
$(GINCS)
$(GBUILD)/GShow.o: $(GSRC)/manager/GShow.cpp
    g++ -c $(GSRC)/manager/GShow.cpp -o
$(GBUILD)/GShow.o $(GINCS)
$(GBUILD)/GMath.o: $(GSRC)/manager/GMath.cpp
    g++ -c $(GSRC)/manager/GMath.cpp -o
$(GBUILD)/GMath.o $(GINCS)
clean:
    rm -f $(GBUILD)/* $(GBIN)/*
```

Construction du projet :

cmd_build.bat

Construction manuel :

```
rm -f build/* bin/*
g++ -c ../src/main.cpp -o build/main.o -
I../src/manager
g++ -c ../src/manager/GShow.cpp -o build/GShow.o -
I../src/manager
g++ -c ../src/manager/GMath.cpp -o build/GMath.o -
I../src/manager
g++ -o bin/GProject.exe build/main.o build/GShow.o
build/GMath.o
```

Gérer les bibliothèques statiques avec Cygwin

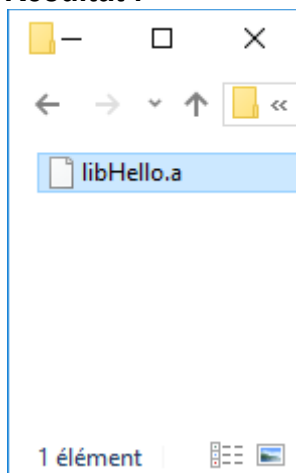
Créer une bibliothèque statique en C++

Objectif :

Créer une bibliothèque statique en C++.

Implémentation :

Créer une classe (GMath) permettant de réaliser des opérations arithmétiques sur deux nombres données, Additionner (add()), Soustraire (subtract()), 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 bibliothèque 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)/libHello.a

GINCS = \
    -I../src/manager

GOBJS = \
    $(GBUILD)/GShow.o \
    $(GBUILD)/GMath.o

all: $(GOBJS)
    ar rcs $(GTARGET) $(GOBJS)
$(GBUILD)/GShow.o: $(GSRC)/manager/GShow.cpp
    g++ -c $(GSRC)/manager/GShow.cpp -o
$(GBUILD)/GShow.o $(GINCS)
$(GBUILD)/GMath.o: $(GSRC)/manager/GMath.cpp
    g++ -c $(GSRC)/manager/GMath.cpp -o
$(GBUILD)/GMath.o $(GINCS)
clean:
    rm -f $(GBUILD)/* $(GBIN)/*
```

Construction du projet :

cmd_build.bat

Construction manuel :

```
rm -f build/* bin/*
g++ -c ../src/manager/GShow.cpp -o build/GShow.o -
I../src/manager
g++ -c ../src/manager/GMath.cpp -o build/GMath.o -
I../src/manager
ar rcs bin/libHello.a build/GShow.o build/GMath.o
```

Utiliser une librairie statique en C++

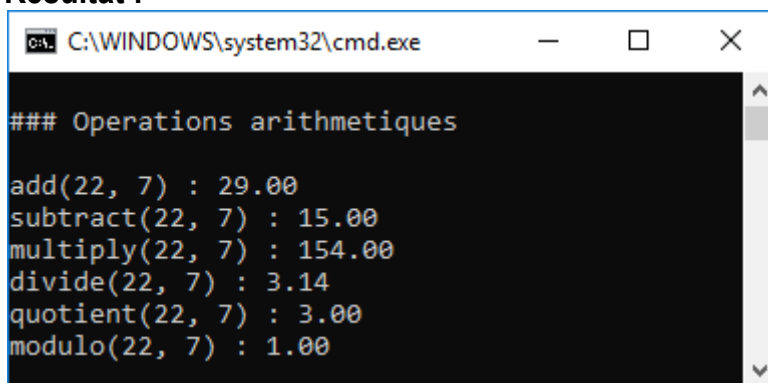
Objectif :

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 :

```
C:\WINDOWS\system32\cmd.exe

### Operations arithmetiques

add(22, 7) : 29.00
subtract(22, 7) : 15.00
multiply(22, 7) : 154.00
divide(22, 7) : 3.14
quotient(22, 7) : 3.00
modulo(22, 7) : 1.00
```

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:
    rm -f $(GBUILD)/* $(GBIN)/*
```

Construction du projet :

cmd_build.bat

Construction manuel :

```
rm -f 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 Cygwin

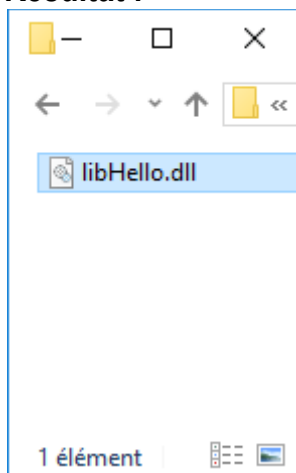
Créer une librairie dynamique en C++

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 (subtract()), 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:
    rm -f $(GBUILD)/* $(GBIN)/*
```

Construction du projet :

cmd_build.bat

Construction manuel :

```
rm -f 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++

Objectif :

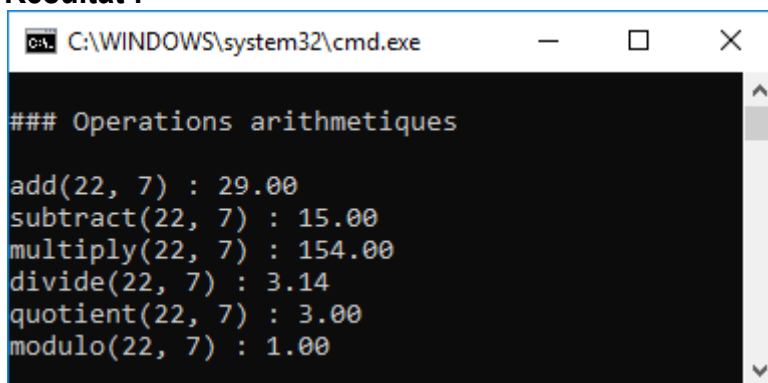
Utiliser une librairie dynamique en C++.

Implémentation :

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

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

Résultat :



```
C:\WINDOWS\system32\cmd.exe

### Operations arithmetiques

add(22, 7) : 29.00
subtract(22, 7) : 15.00
multiply(22, 7) : 154.00
divide(22, 7) : 3.14
quotient(22, 7) : 3.00
modulo(22, 7) : 1.00
```

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:
    rm -f $(GBUILD)/* $(GBIN)/*
```

win/cmd_run.bat

```
@echo off

set PATH=C:\cygwin\bin
set PATH=lib\Hello\bin;%PATH%

echo.
bin\GProject.exe
echo.
pause
```

Construction du projet :

cmd_build.bat

Construction manuel :

```
rm -f 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 Cygwin

Compiler un projet complet C++ avec Cygwin

Objectif :

Compiler un projet complet C++ avec Cygwin.

Utiliser un fichier entête en C++.

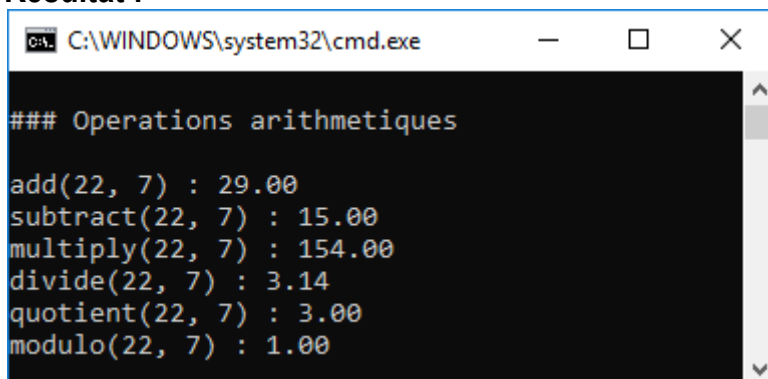
Utiliser une librairie dynamique en C++.

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 donnés (22, 7) et la classe (GShow) de la librairie dynamique (libHello.dll) pour afficher les résultats de calculs.

Résultat :



```
C:\WINDOWS\system32\cmd.exe

### Operations arithmetiques

add(22, 7) : 29.00
subtract(22, 7) : 15.00
multiply(22, 7) : 154.00
divide(22, 7) : 3.14
quotient(22, 7) : 3.00
modulo(22, 7) : 1.00
```

Dossier projet :

```
src/main.cpp
src/manager/GProcess.h
src/manager/GProcess.cpp
win/Makefile
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:
    rm -f $(GBUILD)/* $(GBIN)/*

```

win/cmd_run.bat

```
@echo off
```

```
set PATH=C:\cygwin\bin
```

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

```
echo.
```

```
bin\GProject.exe
```

```
echo.
```

```
pause
```

Construction du projet :

```
cmd_build.bat
```

Construction manuel :

```
rm -f 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
```