

## Contenu Cours Table des matières

**Auteur: Karim NASR**

### **HISTORIQUE DU DOCUMENT**

- 2/09/024. Ajout des fonctionnalités liées à ECOLE**
- 30/08/024. Referentiel initial Ajout des fonctionnalités liées à CLASSE**
- 15/09/024. Mise à jour avec la table des matières**
- 27/10/024. Mise à jour ajout d'exemple afin de gérer une base de donnée SQLite avec 2 exemple TP-SQL-CPP/TP-Sqlite-modern-cpp**

### **Chapitre1: Présentation du langage C++**

#### **1- Programmation structurée et Programmation orientée objet**

- 1.1 Problématique de la programmation
- 1.2 La programmation structurée
- 1.3 Les apports de la programmation orientée objet
  - 1.3.1 Objet
  - 1.3.2 Encapsulation
  - 1.3.3 Classe
  - 1.3.4 Héritage
  - 1.3.5 Polymorphisme
- 1.4 P.O.O,langage de programmation et C++

#### **2-C++ et la programmation structurée**

#### **3-C++ et la programmation orientée Objet**

#### **4-C et C++**

#### **5-C++ et les bibliothéqies standards**

### **Chapitre2: Généralités du langage C++**

#### **1- Présentation par l'exemple de quelques instructions du langage C++**

- 1.1 Un exemple de programme en C++
- 1.2 Structure d'un programme en langage C++
- 1.3 Déclaration
- 1.4 Pour écrire des informations: utiliser le flot cout
- 1.5 Pour faire une répétition:l'instruction for
- 1.6 Pour lire des informations:utiliser le flot cin
- 1.7 Pour faire des choix: l instruction if
- 1.8 les directives à destination du préprocesseur
- 1.9 l'instruction using
- 1.10 Exemple de programme utilisant le type caractère

#### **2-Quelques règles d'écriture**

- 1.1 Les identificateurs
- 1.2 Les mots clés
- 1.3 Les séparateurs

#### **3-C++ et la programmation orientée Objet**

#### **4-C et C++**

#### **5-C++ et les bibliothéqies standards**

## **Chapitre5: Les entrées-sorties conversationnelles de C++**

### **1- Affichage à l'écran**

- 1.1 Exemple 1
- 1.2 Exemple 2
- 1.3 Les possibilités d'écriture sur cout

### **1- Lecture au clavier**

- 2.1 Introduction
- 2.2 Les différentes possibilités de lecture sur cin
- 2.3 Notions de tampon et de caractères séparateurs
- 2.4 Premières règles utilisées par >>
- 2.5 Présence d'un caractère invalide dans une donnée
- 2.6 Les risques induits par la lecture au clavier
  - 2.6.1 Manque de synchronisme entre clavier et écran
  - 2.6.2 Blocage de la lecture
  - 2.6.3 Boucle infinie sur un caractère invalide

## **Chapitre6: Les instructions de contrôle**

### **1- Les blocs d'instruction**

- 1.1 Blocs d'instructions
- 1.2 Déclaration dans un bloc

### **2- L'instruction if**

- 2.1 Syntaxe de l'instruction if
- 2.2 Exemples
- 2.3 Imbrication des instructions if

### **3- L'instruction switch**

- 3.1 Exemple d'introduction de l'instruction switch
- 3.2 Syntaxe de l'instruction switch

### **4- L'instruction do while**

- 4.1 Exemple d'introduction de l'instruction do while
- 4.2 Syntaxe de l'instruction do while

### **5- L'instruction while**

- 5.1 Exemple d'introduction de l'instruction while
- 5.2 Syntaxe de l'instruction while

### **6- L'instruction For**

- 6.1 Exemple d'introduction de l'instruction For
- 6.2 L'instruction for en général
- 6.3 Syntaxe de l'instruction For

### **7- Les instructions de boucle**

- 7.1 Exemple d'introduction de l'instruction For

- 7.2 l'instruction for en général
- 7.3 Syntaxe de l'instruction For

## **Chapitre7: Les Fonctions**

### **1- Exemple de définition et d'utilisation d'une fonction**

### **2- Quelques règles**

- 2.1 Arguments muets et arguments effectifs
- 2.2 L'instruction return
- 2.3 cas des fonctions sans valeur de retour ou sans arguments

### **3- Les fonctions et leurs déclarations**

- 3.1 Les différentes façons de déclarer une fonction
- 3.2 Ou placer les déclarations d'une fonction
- 3.3 Contrôles et conversions induites par le prototype

### **4- Transmission des arguments par valeur**

- 4.1 Cas général
- 4.2 Transmission par valeur et constantes des arguments
  - 4.2.1 Cas des arguments effectifs constants
  - 4.2.2 Cas des arguments muets constants

### **5- Transmission des arguments par référence**

- 5.1 Exemple de transmission par référence
- 5.2 Propriété de la transmission par référence d'un argument
- 5.3 Référence à un argument muet constant
- 5.4 Introduction de risques indirects

### **6- Les variables globales**

- 6.1 Exemple d'utilisation de variables globales
- 6.2 La portée des variables globales
- 6.3 La classe d'allocation des variables globales

### **7- Les variables locales**

- 7.1 La portée des variables locales
- 7.2 Les variables locales automatiques
- 7.3 Les variables locales statiques
- 7.4 Les variables locales à un bloc
- 7.5 Les cas des fonctions récursives

### **8- Transmission par référence d'une valeur de retour**

- 8.1 Introduction
- 8.2 conséquence dans la définition de la fonction
- 8.3 conséquence dans l'utilisation de la fonction
- 8.4 Exemple
- 8.5 Valeur de retour constante

## **9- Initialisation des variables**

- 8.1 Les variables de classe statique

## **11- Surdefinition de fonctions**

- 11.1 Mise en oeuvre de la surdefinition de fonctions
- 11.2 Exemple de choix d'une fonction surdéfinie
- 11.3 Règles de recherche d'une fonction surdéfinie
  - 11.3.1 Cas des fonctions à un argument
  - 11.3.2 Cas des fonctions à plusieurs arguments

## **12- Les fonctions et la déclaration auto (c++ 11)**

- 12.1 Déclaration automatique du type des valeurs de retour
- 12.2 Déclaration automatique du type des arguments muets
- 12.3 Combinaison des deux possibilités

## **13- Les fonctions déclarées constexpr (c++ 11)**

## **14- La référence d'une manière générale**

- 14.1 Déclaration de variables de type référence
- 14.2 Initialisation de référence

## **15- Les fonctions à arguments variables**

- 15.1 Le type initialize\_list (C++ 11)
- 15.2 Application à une fonction à nombre variables d'arguments (C++11)
- 15.3 Les anciennes fonctions va\_start et va\_arg

## **16- Conséquences de la compilation séparée**

- 16.1 Compilation séparée et prototypes
- 16.2 Fonction manquante lors de l'édition de liens
- 16.3 Le mécanisme de la surdéfinition de fonctions
- 16.4 Compilation séparée et variables globales
  - 16.4.1 La portée d'une variable globale-la déclaration extern
  - 16.4.2 Les variables globales et l'édition de liens
  - 16.4.3 Les variables globales cachées - la déclaration static

## **17- La spécification inline**

## **18- Terminaison d'un programme**

## **Chapitre8: Le type String**

### **1- Déclaration et initialisation**

### **2- Lecture et écriture de chaines**

### **3- Affectation de chaines**

### **4- Les fonctions size et empty**

### **5- Concaténation de chaines**

### **6- Accès aux caractéristiques d'une chaine**

#### 6.1 Acces à un caractère de rang donné

##### 6.1.1 Généralités

##### 6.1.2 Absence de contrôle d'indice

##### 6.1.3 Exemple

#### 6.2 Traitement de tous les caractères d'une chaine

##### 6.2.1 Cas général

##### 6.1.2 Absence de contrôle d'indice

##### 6.1.3 Exemple

### **8- Les autres possibilités du type string**

## **Chapitre9: Les Pointeurs natifs**

### **1- Notion de pointeur-les opérateurs \* et &**

#### 1.1 Introduction

#### 1.2 Déclaration multiples et emploi des opérateurs \* et &

#### 1.3 Exemple

### **2- Affectation et comparaison de pointeurs**

#### 2.1 Affectation de pointeurs

#### 2.2 Comparaison de pointeurs

#### 2.3 Le pointeur Nul

#### 2.4 Conversion implicite en bool

### **3- Les conversions entre pointeurs**

### **4- Les pointeurs génériques**

### **5- Pointeurs et constance**

#### 5.1 Pointeur sur un élément constant

#### 5.2 Pointeur constant

#### 5.3 Pointeur constant sur un élément constant

#### 5.4 Constexpr et les pointeurs (C++11)

### **6- Comment simuler une transmission par adresse avec un pointeur**

### **7- Pointeurs et surdéfinition de fonctions**

### **8- Utilisation des pointeurs sur des fonctions**

#### 8.1 Paramétrage d'appel de fonctions

#### 8.2 Fonctions transmises en argument

## **9- Les expressions lambdas (C++11)**

- 9.1 Exemple Introductif
- 9.2 La liste de capture d'une expression lambda
  - 9.2.1 Expression lambda nommée
  - 9.2.2 Liste de capture

## **Chapitre10: La gestion dynamique**

### **1- Les opérateurs new et delete pour les types scalaires**

- 1.1 Présentation de new et delete
- 1.2 Exemple
  - 1.2.1 Exemple1
  - 1.2.2 Exemple2
  - 1.2.3 Exemple3
- 1.3 En cas de manque de mémoire

### **2- Les pointeurs intelligents (C++11)**

#### **3- Le type unique\_ptr (C++11)**

- 3.1 Présentation générale
- 3.2 Fiabilisation des schémas précédents
  - 3.2.1 Exemple1
  - 3.2.2 Exemple2
- 3.3 Initialisation d'un unique\_ptr
  - 3.2.1 Initialisation avec new
  - 3.2.2 Initialisation avec make\_unique
  - 3.3.3 Récapitulatif
- 3.4 Propriété des unique\_ptr
  - 3.4.1 Adresse contenue dans un unique\_ptr
  - 3.4.2 Comparaison
- 3.5 Transfert de propriétés
  - 3.5.1 Par affectation avec move
  - 3.5.2 Par appel de fonctions
  - 3.5.3 Cas particulier de la valeur de retour d'une fonction

#### **4- Le type shared\_ptr (C++11)**

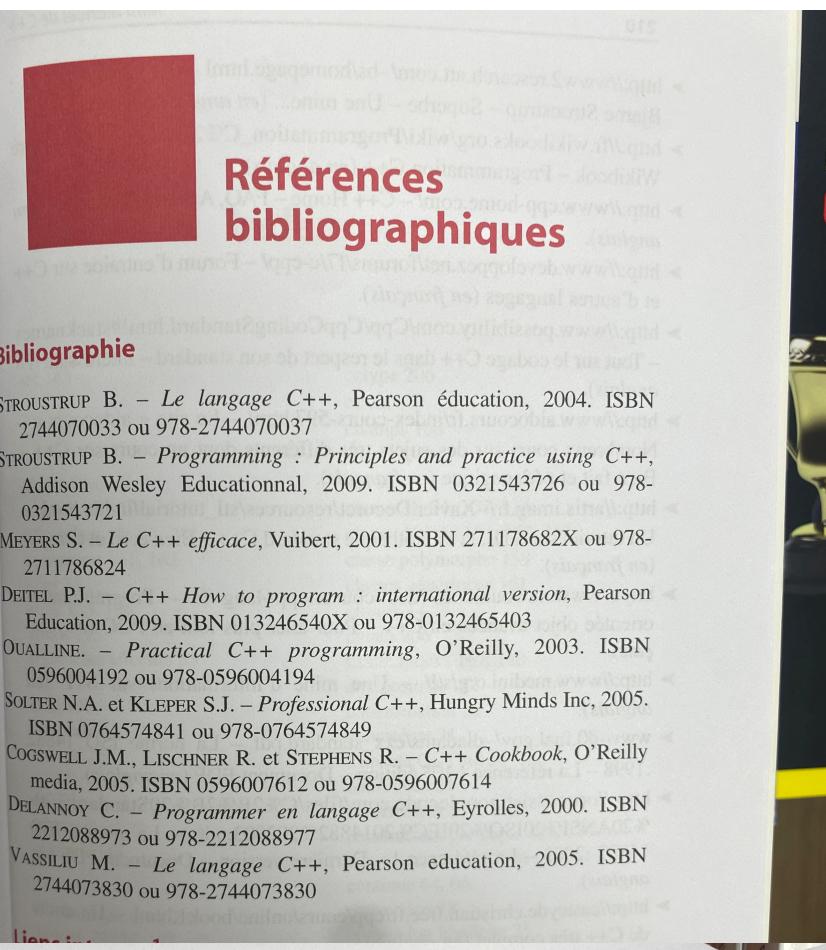
- 4.1 Déclaration et initialisation
- 4.2 Utilisation
- 4.3 L affectation et le compteur de références
- 4.4 Exemple
- 4.5 Transmission par valeur
- 4.6 Conversion entre shared\_ptr et unique\_ptr

#### **5- Quelques précautions (C++11)**

#### **6- Pointeurs intelligents et cycles (C++11)**

#### **7-Les supresseurs (C++11)**

édéfinies
cmath
cmath
cstdlib
cstring
cstring
cstring
cstring
cstdlib
cmath
cmath



210

Mini Manuel de C++

- <http://www2.research.att.com/~bs/homepage.html> – Le site du maître : Bjarne Stroustrup – Superbe – Une mine... (*en anglais*).
- [http://fr.wikibooks.org/wiki/Programmation\\_C%2B%2B](http://fr.wikibooks.org/wiki/Programmation_C%2B%2B) – Un autre Wikibook – Programmation C++ (*en anglais*)
- <http://www.cpp-home.com/> – C++ Home – FAQ, Articles, Tutoriels... (*en anglais*).
- <http://www.developpez.net/forums/f7/c-cpp/> – Forum d'entraide sur C++ et d'autres langages (*en français*).
- <http://www.possibility.com/Cpp/CppCodingStandard.html#stacknames> – Tout sur le codage C++ dans le respect de son standard – Intéressant (*en anglais*).
- <http://www.aidocours.fr/index-cours-597.html> – Le site « aidocours » – Nombreux cours sur des sujets très différents dont un cours sur C++ – Bien fait et pédagogique (*en français*).
- <http://artis.imag.fr/~Xavier.Decoret/resources/stl Tutorial/index.html> – Un tutoriel sur STL, la bibliothèque standard C++ – Bien écrit et complet (*en français*).
- <http://www.nawouak.net/?doc=course.cpp+lang=fr> – Programmation orientée objet avancée en C++ – Pour aller plus loin en POO (*en français*).
- <http://www.medini.org/stl/> – Une mine d'informations sur STL (*en anglais*).
- [http://www-d0.fnal.gov/~dladams/cxx\\_standard.pdf](http://www-d0.fnal.gov/~dladams/cxx_standard.pdf) – La norme ISO 14882 :1998 – La référence ! 1<sup>ère</sup> édition – Document PDF (*en anglais*).
- <http://openassist.googlecode.com/files/C%2B%2B%20Standard%20-%20ANSI%20ISO%20IEC%2014882%202003.pdf> – La norme ISO 14882 :2003 – La référence ! – Dernière version – Document PDF (*en anglais*).
- <http://casteyde.christian.free.fr/cpp/cours/online/book1.html> – Un cours de C++ très complet (*en anglais*).
- <http://www.cplusplus.com/reference/stl/> – Tout sur C++ – Énorme... (*en anglais*).

## 8-Le type auto\_ptr

### Chapitre11: Les vecteurs,les tableaux natifs et les chaînes C

#### 1- Les vecteurs

##### 1.1 Exemple de présentation des vecteurs

##### 1.2 Les éléments et les indices d'un vecteur

###### 1.2.1

###### Quelques règles

###### 1.2.2

###### Absence de contrôles d'indice

##### 1.3 Initialisation d'un vecteur

##### 1.4 Parcours d'un vecteur avec for pour les séquences

##### 1.5 Affectation de vecteurs

##### 1.6 Vecteur transmis en argument d'une fonction

###### 1.6.1 Par défaut la transmission se fait par valeur

###### 1.6.2

###### Transmission d'un vecteur par références ou par pointeur

##### 1.7 Autres possibilités d'un type vector

#### 2- Les tableaux natifs

##### 2.1 Les tableaux natifs et exemple

###### 2.1.2

###### Quelques règles

###### 2.1.3

###### Parcours des

éléments avec for pour les séquences (C++11)

- 2.2 Les tableaux natifs à plusieurs indices
  - 2.2.1 leurs déclarations
  - 2.2.2 Arrangement en mémoire des tableaux à plusieurs indices
  - 2.2.2 Parcours des éléments de tableau à plusieurs indices
- 2.3 Initialisation des tableaux natifs
  - 2.3.1 Initialisation de tableaux natifs à un indice
  - 2.3.2 Initialisation de tableaux natifs à plusieurs indices
  - 2.3.2 Initialisateurs et classe d'allocation

## **Chapitre12: Classes et objets**

### **1- La notion de classe**

- 1.1 Definition d'une classe point
  - 1.1.1 Declaration de la classe point
  - 1.1.2 Definition des fonctions membres de la classe
- 1.3 Exemple récapitulatif
- 1.4 La déclaration d'une classe d'une manière générale

### **2- Les structures**

### **3- Affectation d'objets**

### **4- Notion de constructeur et de destructeur**

- 4.1 Introduction
- 4.2 Exemple de classe comportant un constructeur
- 4.3 Initialisation des membres dans l'entête du constructeur
- 4.4 Construction et destruction des objets
- 4.5 roles du constructeur et destructeur
- 4.6 quelques règles

### **5- Objets transmis en argument d'une fonction**

- 5.1 Cas de la transmission par valeur
- 5.2 Cas de la transmission par référence
- 5.3 Cas de la transmission par pointeur:l'opérateur ->

### **6- Les membres données statiques**

- 6.1 Le qualificatif static pour un membre donnée
- 6.2 Initialisation des membres données statiques
- 6.3 Exemple

### **7- Exploitation d'une classe**

- 7.1 la classe comme composant logiciel
- 7.2 Protection contre les inclusions multiples
- 7.3 Cas des membres données statiques
- 7.4 Modification d'une classe
  - 7.4.1 Notion d'interface et d'implémentation
  - 7.4.2 Modélisation d'une classe sans modification de son interface
  - 7.4.3 Modification d'une classe avec modification de son interface

## **Chapitre13: Les propriétés des fonctions membres**

- 1- surdefinition des fonctions membres**
- 2- Arguments par défaut**
- 3- Les fonctions membres en ligne**
- 4- Constructeurs délégués (C++ 11)**
- 5- Cas des objets transmis en argument d'une fonction membre**
- 6- Mode de transmission des objets en argument**
  - 6.1 Transmission de l'adresse d'un objet
  - 6.2 Transmission par référence
  - 6.3 Les problèmes posés par la transmission par valeur
- 7- Lorsqu'une fonction renvoie un objet**
- 8- Autoréférence: le mot clé this**
- 9- Les fonctions membres statiques**

## **Chapitre14: Construction, destruction et initialisation des objets**

### **1- Les objets automatiques et statiques**

- 1.1 Durée de vie
- 1.2 Appel des constructeurs et des destructeurs
- 1.3 Exemple

### **2- Les objets dynamiques**

- 2.1 Cas d'une classe sans constructeur
- 2.2 Cas d'une classe avec constructeur
  - 2.2.1 cas des pointeurs natifs
  - 2.2.2 Avec les pointeurs intelligents
  - 2.2.3 Exemple

### **3- Le constructeur de recopie**

- 3.1 Il n'existe pas de constructeur approprié
- 3.2 Il existe un constructeur approprié
- 3.3 Comment interdire la construction par recopie

### **4- Exemple de constructeur de recopie**

- 4.1 Comportement du constructeur de recopie par défaut
  - 4.1.1 version pointeurs natifs
  - 3.2 Il version unique\_ptr (C++11)
- 4.2 Définition d'un constructeur de recopie

### **5- Initialisation d'un objet lors de sa déclaration**

- 5.1 Cas d'un constructeur à un seul argument
- 5.2 Cas d'un constructeur à plusieurs arguments
- 5.3 Le mot clé default pour un constructeur (C++ 11)
- 5.4 Cas particulier des classes agrégats
- 5.5 Constructeur avec initialize\_list (C++11)

### **6- Objets membres**

- 6.1 Introduction
- 6.2 Mise en œuvre des constructeurs et des destructeurs

6.3 Le constructeur de recopie

## 7- Les tableaux et vecteurs d'objets

7.1 Notation

7.2 Constructeurs et initialiseurs

7.3 Cas des tableaux dynamiques d'objets

## 8- Les objets temporaires

## 9- Dissocier l'allocation mémoire de la construction

## Chapitre15: Les fonctions amies

### 1- Exemple de fonction indépendante amie d'une classe

### 2- Les différentes situations d'amitié

2.1 Fonction membre d'une classe, amie d'une autre classe

2.2 Fonction amie de plusieurs classes

2.3 Toute les fonctions d'une classe amies d'une autre classe

### 3- Exemple

3.1 Fonction amie indépendante

3.2 Fonction amie ,membre d'une classe

## 4- Exploitation de classes disposant de fonctions amies

## Chapitre35: Introduction aux threads (C++ 11)

### 1- Thread depuis une fonction ou un objet fonction

2.1 Généralités

2.2 Transmission d'argument à un thread

2.3 Mise en sommeil d'un thread

### 2- Thread depuis une fonction membre

### 3- Thread et exceptions

### 4- Partage de donnée entre thread et verrous mutex

### 5- Prise en compte des exceptions avec verrou lock\_guard

### 6- Les variables atomic

### 7- Transfert d'informations au retour d'un thread

7.1 Utilisation de la fonction async

7.2 Démarche plus générale

Les projets	14
Choisissez votre IDE	14
Code Blocks (Windows,macos,linux)	15
Xcode (macos seulement)	24

### **3 Votre premier programme**

lien

[La programmation en C++ moderne • Bibliothèque • Zeste de Savoir](#)

Premier programme de moyenne

<https://informaticienzero.github.io/c++-avec-openclassrooms-ou-comment-perdre-son-temps/>

Le monde merveilleux de la console.	13
Les programmes graphiques	14
Les programmes en console	14
Notre première cible : les programmes en console	
Création et lancement d'un premier projet	15

Voir Programme « Hello world » et TP-1/2, TP Gestion classe Master

Lien Github <https://github.com/karnasrr/travaux-c-.git>

Utilisation environnement Xcode

Création d'un projet	24
Lancement du programme	
Explication sur ce premier code source	
La ligne include	
La ligne using namespace	
La ligne Int Main	
La ligne cout	
La ligne return	
Commentez vos programmes	

Création du projet TP Gestion classe Master (dans environnement Xcode Apple).

Le projet consiste à saisir des écoles, classes et des étudiants et les affecter à des classes

Possibilité d'afficher les classes par ordres croissant

Possibilité d'afficher les écoles par ordres croissant

Ce programme permet de comprendre les notions du C++ à travers un menu de saisie.

Voir les sources (TP3 Gestion Classe Master en PDF main.pdf + revision0.pdf )

<https://github.com/karnasrr/travaux-c-.git>

Les possibilités du Menu sont celles ci:

- 1- ajouter une classe
- 2- afficher les classes
- 3- ajouter un étudiant dans une classe
- 4- lister les étudiants d'une classe
- 5- afficher les classes par ordre croissant
- 6- Quitter

**Evolutions du 2/09/2024:**

- 1- ajouter une Ecole
- 2- afficher les écoles
- 3- ajouter un etudiant dans une école
- 4- lister les etudiants d'une école
- 5- afficher les écoles par ordre croissant
- 6- Quitter

Le contenu permet de connaitre les instructions de base type *cout,switch,les structures.*

L instruction *using namespace*

Les premieres instruction pour utiliser les *tableaux*

*La ligne include*

*La ligne using namespace*

*La ligne Int Main*

*La ligne cout*

*La ligne return*

*Les boucles do while, for...*

Pour les instructions se referer aux fichiers PDFs cités dans TP3 Gestion Master.

En option de cette section, rentrer dans son compte Gitlab les sources et créer un répertoire documents.

Suivre les instructions suivantes pour les comandes shell sous le terminal:

```
-cd Travaux\ C++
-git add TP\ n3-Gestion\ classe\ Master/
-git add -a
-git commit -m "commit"
-git config --list
-git init
-git remote add origine https://github.com/karnasrr/travaux-c-.git
-git branch -M main
-git config --global user.name
-git push -u origine main
-A l invite: entrer User NAmE: Karim NASR et pswd
token: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
```

```
new token 6/09: ghp_AcDliOVxvPxILye05QyAlYWZmKKNrw2aqLTX
new token 11/11: ghp_X8715dZPWITBn2OlqRvZgYqKZwUxtY2rXgD4
```

Option 3: Le precedent projet etait basee uniquement sur de l'encapsulation de structure,  
Il fautdra desormais le transformer en mode objets (avec un systeme heritage).

Option4: cf extrait YouTube de TP QT Exemple 2 (arrete à 11:21) pour faire des labels et  
des Frames d'affichage de Menu + « affichage de Hello World ».

### TP extract du C++ moderne

<https://github.com/karnasrr/travaux-c->

Au 09/09/024

La creation d une classe

Se referer au depot pour voir les exercices des constructeurs de recopie er desdestructeurs de  
différentes manières (par pointeurs, par recopie)

Se referer au programme creation d'objets temporaire et destruction immédiate (IMG 1753).

Au 11/09/024

Se referer a l exemple sur les patrons de fonctions à un type classe (IMG-1545) et un exemple classique (pas sur type classe) dans dépôt Git.

Au 15/09/024

Chap11 exercices sur les vecteurs

[TP extrait du site de Genelaxis et Bogolo pour la gestion de threads](#)

Au 13/09/024

Se referer au dépôt GitHub pour la gestion de threads

[travaux-c-/TP-C++Moderne/TP-Threads](#)

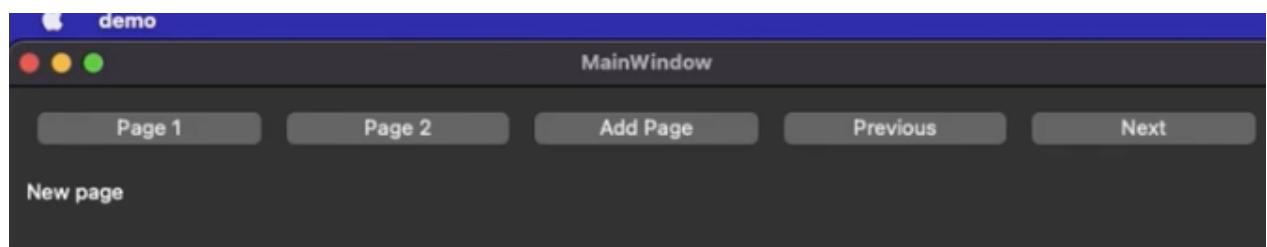
### 3 bis Maquettage IHM (Sous QT)

Vor les designs des Fenêtres sous formalisme QT.

Tutoriel framework GUI Qt Français | Chapitre 5 | Gestion des pages avec QStackedWidget  
(Extrait de Codersland FR - chapitre 5)

IE TUTORIAL YOUTUBE PRECEDENT CHAPITRE 5 PERMET DE CREER DES onglets (et naviguer à l'aide d'onglets sur differentes pages et d'afficher ces tables) page 1...4 (next/ previous) voir la video.

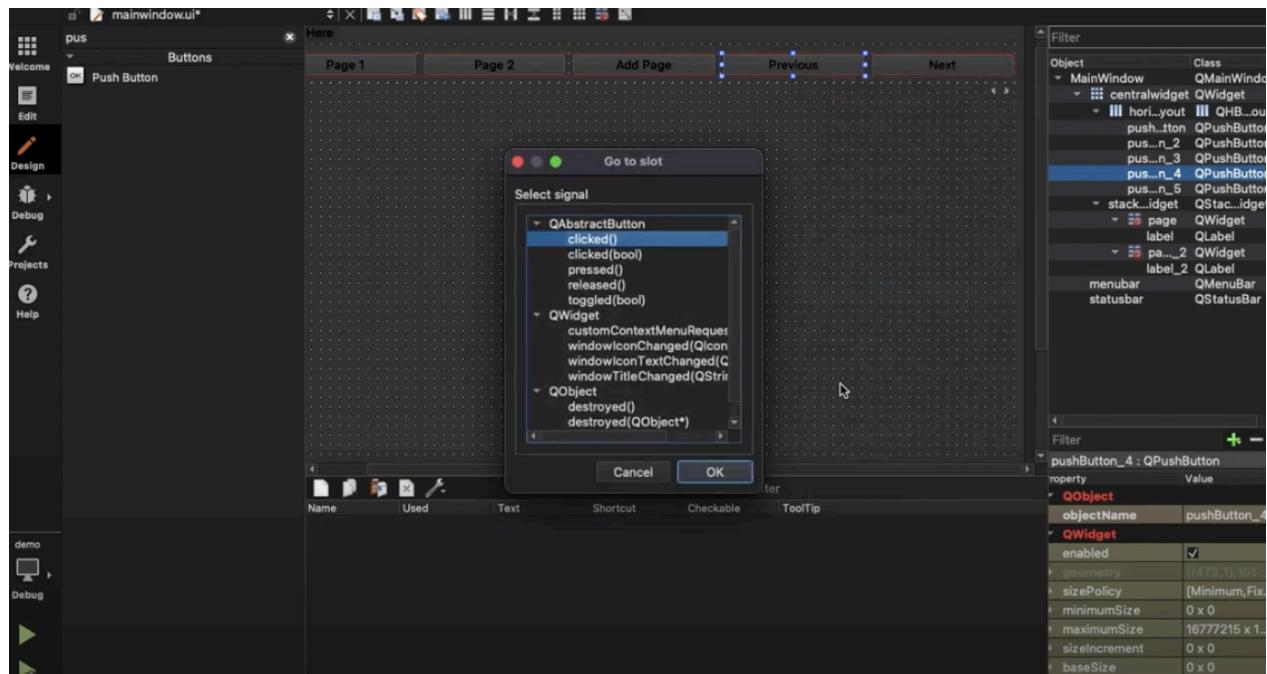
[TP- Chapitre 5 \(parcours de plusieurs pages\)](#)



- vue video chapitre 5.

(Voir dossier images & notes)

<https://github.com/karnasrr/travaux-c-/tree/main>



- utilisation de Qframe & QLabel.

```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 #include <QFrame>
5
6 MainWindow::MainWindow(QWidget *parent)
7     : QMainWindow(parent)
8     , ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11 }
12
13 ~MainWindow()
14 {
15     delete ui;
16 }
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     ui->stackedWidget->setCurrentIndex(0);
21 }
22
23 void MainWindow::on_pushButton_2_clicked()
24 {
25 }

```

- images des méthodes utilisées par les slots dans MainWindow, utilisation également de Qframe. Index(0) pour afficher la page 1.

*Voir le slide 34 de TVAIRA sur la connexion signal/slot et la méthode connect():*

*Voir le slide 36 de TVAIRA sur les signaux personnalisés:*

Il est possible de créer ses propres signaux et slots.

### TP exemple sur les slots

```

#include <QObject>
#include <QDebug>
class MaClasse : public QObject {
    Q_OBJECT
private:
    int numero;
public:
    MaClasse( int numero=0, QObject *parent=0 ) : QObject(parent), numero(numero) {}
    void emettre() {
        emit send(numero); // envoie le signal send avec la valeur de la variable numero
    }
signals:
    void send( int ); // un signal personnalisé
public slots:
    void receive( int valeur ) // un slot personnalisé {
        qDebug() << "signal recu " << valeur;
    }
};

```

*Slide 39 montre exemple du mécanisme Signal/slot*

```

MaClasse monObjet1(1), monObjet2(2); // instacie 2 objets
// le signal et le slot doivent étre compatibles (même signature)
QObject::connect(&monObjet1, SIGNAL(send(int)), &monObjet2, SLOT(receive(int)));
monObjet1.emettre();
// la méthode emettre() de l'objet1 enverra le signal send avec la valeur 1,
// ce qui déclenchera l'exécution du slot receive de l'objet2 et
// cela affichera "signal recu 1"

```

Le slide 40 montre le mécanisme de connexions.

### TP sur label:

*Extrait des slides de TAVAIRA ([tvaira.free.fr](http://tvaira.free.fr))*



### [Programme \(Main.cpp\)](#)

```

#include <QApplication>
#include "MyMainWindow.h"

int main(int argc, char **argv) {
    QApplication app(argc, argv); // mon objet application
    MyMainWindow w; // mon objet fenêtre
    w.show(); // affichage
    return app.exec(); // boucle
}

```

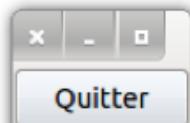
### TP sur une fenêtre avec Bouton (nommé Quitter)

### [Programme \(Main.cpp\)](#)

```

#include <QApplication>
#include <QPushButton>
int main(int argc, char **argv) {
    QApplication app(argc, argv); QPushButton bouton("Quitter");
    // on connecte le signal clicked() de l'objet bouton
    // au slot quit() de l'objet app (ici le pointeur qApp) QObject::connect(&bouton,
    // SIGNAL(clicked()), qApp, SLOT(quit()));
    bouton.show(); return app.exec();
}

```



**Cours** <https://perso.telecom-paristech.fr/elc/qt/Qt-Graphique.pdf>  
<http://tvaira.free.fr/dev/qt/presentation-qt.pdf>

### Contenu (TVEIRA)

#### **Qu'est-ce qu'un projet Qt ? (Slide 41)**

Qmake et Makefile (slides 41 à 44)  
QtCreator (slide 45/46)  
QtDesigner (slide 47)

## **Les classes QLayout**

### **(Slide 51/52)**

Les différents Layouts

Intérêts des layouts (slide 51)

Création des Widgets (slide 56)

Gestion du positionnement

Transition Qt4/Qt5 (slide 58)

Lien docs QT4

Lien docs QT4/5

Autre liens

## Contenu (C++, programmation QT)

Le fichier .pro Les fichiers de conception graphique (ui) Les fichiers de gestion d'internationalisation (ts et qm) Les types de bases du langage (qint, qfloat ...)

La compilation avec qmake. La classe QObject L'introspection avec Qt La gestion de la mémoire

Les composants de base de l'IHM (QMainWindow, QFrame, QLabel ...) La gestion du positionnement des composants (QLayout) Les boîtes de dialogue (QDialog) Les menus (QMenu) Les outils de conception visuelle de Qt (Qt Designer ...)

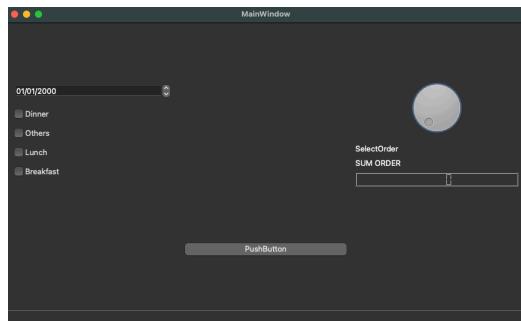
Notions de signal et slot Déclaration de signaux et de slots Installer des filtres d'événement Accéder à l'application pendant un traitement lourd ( QTimer )

Au 5/09/024 -> 14h30 arrêt slide 29.

Au 6/09/024 -> 15h30 cours Qt de [tvaira.free.fr](http://tvaira.free.fr) (MyMainWindow.cpp) slide 31/63.

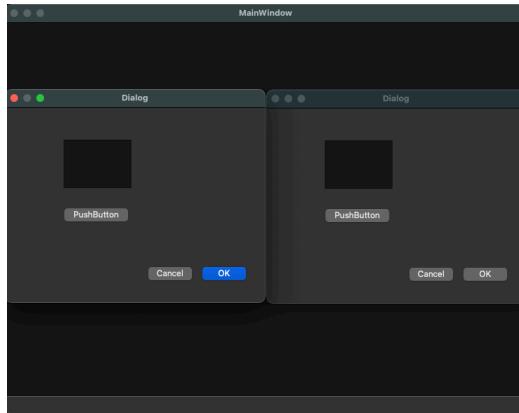
Slide 32 -deuxieme fenêtre (affiche Label)

Au 16/09/024 -> se referer à l'exemple sur Qt sur le projet (*QtThreeWidgetTutorial*) également dans GitHub avec la documentation montrant les options checkbox et l'affichage sur un LCD ainsi qu'un curseur affichant sur LCD.  
La documentation s'appelle *QtThreeWidgetTutorial*.



Au 20/09/024 -> se referer à l'exemple sur Qt sur le projet (displayingWindows)

Creation d'un projet nommé DisplayingWindows sous Qt  
pour la création de boites de dialogues extrait de youtube  
VoidRealms  
// creation d'un nouveau formulaire de dialog dans formulaire et le fichier  
//dialog.h  
// modification de mainwindow.h pour intégrer Dialog \*mDialog;



Au 20/09/024 ->

Le dossier ReadingWritingTextFile est extrait de YouTube

On écrit et lit dans un fichier « test.txt » avec OStream, si on change le nom du fichier alors message « file not found »

Au 27/10/024 Gestion de la base de donnée SQLlite et d'un wrapper

Ajout d'un exemple simple TP-SQL-CPP pour la gestion d'une base test.db et la table COMPANY

Ajout d'exemple d'un wrapper Sqlite TP-Sqlite-modern-cpp extrait du lien

[https://github.com/SqliteModernCpp/sqlite\\_modern\\_cpp/tree/dev](https://github.com/SqliteModernCpp/sqlite_modern_cpp/tree/dev)

#### 4 utiliser la mémoire

Qu est qu une variable

Les noms de variables

Les types de variables

Déclarer une variable

Le cas de chaines de caractères

une astuce pour gagner de la place

Déclarer sans installer

Afficher la valeur d une variable

Les references

En resume

#### 5 une vraie calculatrice

Demandez des informations à l'utilisateur

Lecture depuis la console

Une astuce pour les chevrons

D'autre variables

Le problème des espaces

Demandez dabord la valeur de pi

Modifier les variables  
Changer le contenu d'une variable  
Une vraie calculatrice de base  
Les constantes  
Déclarer une constante  
Un premier exercice  
Les raccourcis  
L'incrémentation  
La décrémentation  
Les autres opérations  
Encore plus de maths  
L'entête Cmath  
Racine carre  
Autre fonction présente dans cmath  
Le aco de la fonction puissance  
En résumé

## 6 les structures de contrôles

Les conditions  
La condition if  
La condition switch  
Booleens et combinaisons de conditions  
Les booleens  
Combiner des conditions  
Les boucles  
La boucle while  
La boucle do while  
La boucle for  
En résumé

## 7 Decouper les programmes en fonction

Créer et utiliser une fonction  
Présentation des fonctions  
Définir une fonction  
Créer une fonction toute simple  
Appeler une fonction  
Une fonction qui prend plusieurs paramètres  
Une fonction sans arguments  
Une fonction qui ne renvoie rien

Quelques exemples d'application  
Calculer le carré d'un nombre  
Des fonctions différentes avec des variables portant le même nom  
Une fonction à deux arguments

Notions avancées  
Passage par valeur  
Passage par référence  
Passage par référence constante

Utiliser plusieurs fichiers sources  
Les fichiers nécessaires  
Déclarer la fonction dans le fichier  
Commenter son code  
Des valeurs par défaut pour les arguments  
Les valeurs par défaut  
Cas particulier, attention danger  
En résumé

## 8 les tableaux

Les tableaux statiques  
Yn exemple d'utilisation  
Déclarer un tableau statique  
Acceder aux elements d un tableau  
Parcourir un tableau  
Un petit exemple  
Les tableaux et les fonctions  
Les tableaux dynamiques  
Declarer un tableau dynamique  
Acceder aux elements d'un tableau  
Changer la taille  
Exercice calculer une moyenne  
Les tableaux dynamiques et les fonctions  
Les tableaux multidimensionnels  
Declarer un tableau multidimentionnel  
Acceder aux elements  
Aller plus vite  
Les chaines utilisees comme des tableaux  
Acceder au caracteres  
Les fonctions  
En Resume

## 9 lire et modifier des fichiers

[Lisez et modifiez des fichiers - Apprenez à programmer en C++ - OpenClassrooms](#)  
Ecrire dans un fichier  
L entete Fstream  
Ouvrir un fichier en ecriture  
Ecrire dans un flux  
Les differents modes d'ouverture  
Lire un fichier  
Ouvrir un fichier en lecture  
Lire le contenu  
Lire un fichier en entier  
Quelques astuces  
Fermer prematurément un fichier  
Le curseur dans le fichier  
Connaitre sa position  
Se deplacer  
Connaitre la taille d'un fichier  
En resume

## 10 TP le mot mystere

Preparatifs et conseils  
Principe du jeu 'le mot mystere «  
Quelques conseils pour bien demarrer  
Reperer les etapes du programme  
Creer un canevas de code avec les etapes  
Un peu d'aide pour melanger les lettres  
Correction  
Le code  
Les explications  
Aller plus loin

## 11 les pointeurs

Une question d'adresse  
Afficher l adresse  
Les pointeurs  
Declarer un pointeur

Quand utiliser des pointeurs  
Partager une variable  
Choisir parmi plusieurs éléments  
En résumé

Deuxième partie la programmation objet

## 12 La vérification sur les chaînes enfin dévoilée

Des objets .... Pour quoi faire? ....  
Comprendre le fonctionnement du type string  
Pour un ordinateur les lettres n'existent pas  
Les textes sont des tableaux de caractères  
Créer et utiliser des objets string  
Créer un objet string  
Concaténation de chaînes  
Comparaison de chaînes  
Attributs et méthodes  
Quelques méthodes utiles du type string  
En résumé

## 13 Les classes (partie 1/2)

Qu'est-ce qu'une classe  
Quelle classe créer  
Définition d'une classe  
Ajout de méthodes et d'attributs  
Les attributs  
Les méthodes  
Droits d'accès et encapsulation  
Les droits d'accès  
L'encapsulation  
Séparer prototypes et définitions  
Personnage.hpp  
Personnage.cpp  
main.cpp

## 14 Les classes (partie 2/2)

Constructeur et destructeur  
Le constructeur  
Le destructeur  
Les méthodes constantes  
Associer des classes entre elles  
La classe Arme  
Adapter la classe Personnage pour utiliser la classe Arme  
Afficher l'état des personnages  
Ajout du prototype  
Implementation

Appel de AfficherEtat dans le main()  
Résumé de la structure du code  
En résumé

## 15 La surcharge d'opérateurs

Petits préparatifs  
Principe de la surcharge d'opérateurs  
La classe Durée pour nos exemples  
Les opérateurs de comparaison  
L'opérateur ==  
L'opérateur Different  
L'opérateur <

- Les autres operateurs de comparaison
- Les opérateurs arithmétiques
  - L'opérateur d'addition +
  - Les autres opérateurs arithmétiques
- Les opérateurs de flux
  - Définir ses propres opérateurs pour le cout
  - Implementation d'opérateur <<
  - Afficher des objets Duree dans le main()

En résumé

## 16 TP la programmation POO en pratique

- Préparatifs
- Description de la classe Fraction
- Créer un nouveau projet
- Le code de base des fichiers
  - Construction de la classe
  - Choix des attributs
- Corréction
- Les constructeurs
- Afficher une fraction
- L'opérateur de multiplication
- L'opérateur de comparaison
- Aller plus loin

## 17 classes et pointeurs

- Pointeur d'une classe vers une autre classe
- Gérer l'allocation dynamique
  - Allouer de la mémoire pour l'objet
  - Desallouer de la mémoire pour l'objet
  - Adapter les méthodes
  - Le pointeur this
- Le constructeur de copie
- Le problème
- Créer le constructeur de copie
- Terminer le constructeur de copie
- L'opérateur d'affectation

En résumé

## 18 Le héritage

- Exemple de héritage simple
- Comment reconnaître un héritage
- Notre exemple la classe personnage
- La classe guerrier hérite de personnage
- La classe Personnage hérite aussi de Personnage
- La dérivation de type
- Héritage et constructeurs
  - Appeler le constructeur de la classe mère
  - Transmission de paramètres
- La portée protected
- Le masquage
  - Une fonction de la classe mère
  - La fonction est héritée dans les classes filles
  - Le masquage
  - Le démasquage

En Résumé

## 19 Le polymorphisme

- La résolution des liens
- La résolution statique des liens

- La resolution dynamique des liens
- Les methodes speciales
  - Le cas des constructeurs
  - Le cas des destructureurs
  - Le code ameliore
- Les collections heterogenes
  - Le retour des pointeurs
  - Utiliser la collection
- Les fonctions virtuelles pires
  - Le probleme des roues
  - Les classes abstraites
- A vous de jouer
- En resume

## 20 Elements statiques et amitie

- Les methodes statiques
- Creer une methode statique
- Les attributs statiques
- Creer un attribut statique dans une classe
- L amitie

- Qu est que l amitie
- Retour sur la classe Duree
- Declarer une fonction amie d une classe
- L amitie et la responsabilite

En Resume

## 21 La STL

[Nawouak.net \[course/cpp/stl\] \[fr\]](#)  
(Voir Cpp.pdf)

## 22 Patrons de composants et classes generiques

(Voir Cpp.pdf)

## 23 QT

### 23.1 Exemple Traffic Light

Extrait du lien suivant:

<https://qt.developpez.com/doc/4.7/statemachine-trafficlight/> (traffic light exemple wuth Qt)

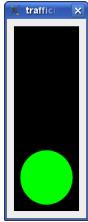
```
int main(int argc, char **argv)
{
    QApplication app(argc, argv);

    TrafficLight widget;
    widget.resize(110, 300);
    widget.show();

    return app.exec();
}
```

The main() function constructs a TrafficLight and shows it.

The Traffic Light example shows how to use [The State Machine Framework](#) to implement the control flow of a traffic light.



```
class LightWidget : public QWidget
{
    Q_OBJECT
    Q_PROPERTY(bool on READ isOn WRITE setOn)
public:
    LightWidget(const QColor &color, QWidget *parent = 0)
        : QWidget(parent), m_color(color), m_on(false) {}

    bool isOn() const
        { return m_on; }
    void setOn(bool on)
    {
        if (on == m_on)
            return;
        m_on = on;
        update();
    }

public slots:
    void turnOff() { setOn(false); }
    void turnOn() { setOn(true); }

protected:
    virtual void paintEvent(QPaintEvent *)
    {
        if (!m_on)
            return;
        QPainter painter(this);
        painter.setRenderHint(QPainter::Antialiasing);
        painter.setBrush(m_color);
        painter.drawEllipse(0, 0, width(), height());
    }

private:
    QColor m_color;
    bool m_on;
};

class TrafficLightWidget : public QWidget
{
public:
    TrafficLightWidget(QWidget *parent = 0)
        : QWidget(parent)
    {
        QVBoxLayout *vbox = new QVBoxLayout(this);
        m_red = new LightWidget(Qt::red);
        vbox->addWidget(m_red);
        m_yellow = new LightWidget(Qt::yellow);
        vbox->addWidget(m_yellow);
        m_green = new LightWidget(Qt::green);
        vbox->addWidget(m_green);
        QPalette pal = palette();
        pal.setColor(QPalette::Background, Qt::black);
        setPalette(pal);
        setAutoFillBackground(true);
    }

    LightWidget *redLight() const
        { return m_red; }
    LightWidget *yellowLight() const
```

```

    { return m_yellow; }
LightWidget *greenLight() const
    { return m_green; }

private:
    LightWidget *m_red;
    LightWidget *m_yellow;
    LightWidget *m_green;
};

The TrafficLightWidget class represents the visual part of the traffic light; it's a widget that contains three lights arranged vertically, and provides accessor functions for these.

```

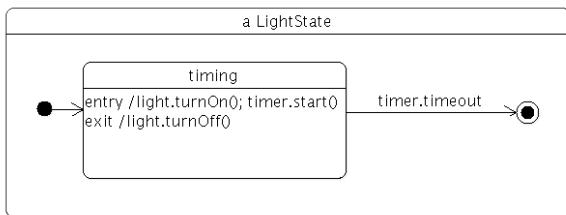
The TrafficLightWidget class represents the visual part of the traffic light; it's a widget that contains three lights arranged vertically, and provides accessor functions for these.

```

QState *createLightState(LightWidget *light, int duration, QState *parent = 0)
{
    QState *lightState = new QState(parent);
    QTimer *timer = new QTimer(lightState);
    timer->setInterval(duration);
    timer->setSingleShot(true);
    QState *timing = new QState(lightState);
    QObject::connect(timing, SIGNAL(entered()), light, SLOT(turnOn()));
    QObject::connect(timing, SIGNAL(exited()), timer, SLOT(start()));
    QObject::connect(timer, SIGNAL(timeout()), light, SLOT(turnOff()));
    QFinalState *done = new QFinalState(lightState);
    timing->addTransition(timer, SIGNAL(timeout()), done);
    lightState->setInitialState(timing);
    return lightState;
}

```

The createLightState() function creates a state that turns a light on when the state is entered, and off when the state is exited. The state uses a timer, and as we shall see the timeout is used to transition from one LightState to another. Here is the statechart for the light state:



## **23 References Bibliographiques**

## **24 liens internet**

[www.cppfrance.com](http://www.cppfrance.com)

<http://casteyde.christian.free.fr/cpp/cours/online.book1.html> un cours complet

<http://www.cplusplus.com/reference/stl> Tout sur c++ Enorme en anglais

<http://www.medini.org/stl> une mine d informations sur stl en anglais

<http://www.nawouak.net/?doc=course.cpp+lang-fr> Programmation orientee objet avancée en C++

<http://www.aidocours.fr/index-cours-597.html> nombreux cours sur sujets différents dont c++

<http://www.cpp-home.com/> C++ Home FAQ articles

[http://artis.imag.fr/xavuer.decorer/ressources/stl\\_tutorial/index.html](http://artis.imag.fr/xavuer.decorer/ressources/stl_tutorial/index.html) tutorial sur STL

<https://perso.telecom-paristech.fr/elc/qt/Qt-Graphique.pdf>

[La programmation en C++ moderne • Bibliothèque • Zeste de Savoir](#)

[Installer Qt sur Mac avec Xcode ou QtCreator.](#)

[Qt without XCode - how to use Qt Creator for macOS software development without installing XCode • GitHub](#)

Youtube

Creating Qt Widget Application (GUI) on macOS [Tommy Ngo](#)

How To Install Qt Creator on Mac / MacOS (2024) [ProgrammingKnowledge](#)

Reference qt install on Mac

[QT C++ GUI Tutorial For Beginners](#)

Qt [project.org](http://qt-project.org)

[Qt Creator Documentation](#)

[Liens utiles pour QT](#)

- <https://qt.developpez.com/doc/4.7/statemachine-trafficlight/> (traffic light exemple with Qt)

- [Site Télécom: http://www.telecom-paristech.fr/~elc/qt](http://www.telecom-paristech.fr/~elc/qt)

- <http://tvaira.free.fr/dev/qt/presentation-qt.pdf>

- [Site général Qt: http://www.qt.io/](http://www.qt.io/)

- [ToolkitQt:](#)

- Page principale : <http://doc.qt.io/qt-5/>
- Toutes les classes (par modules) : <http://doc.qt.io/qt-5/modules-cpp.html>
- QtWidgets :
- Documentation : <http://doc.qt.io/qt-5/qtwidgets-index.html>
- Liste des classes : <http://doc.qt.io/qt-5/qtwidgets-module.htm>

## ANNEXE - Section dediee à GitLab

### Pour resoudre message erreur

Messahe 1 **Updates were rejected because a pushed branch tip is behind its remote**

```
error: failed to push some refs to 'https://github.com/kimnasr/travaux-c-.git'
hint: Updates were rejected because a pushed branch tip is behind its
remote
hint: counterpart. Check out this branch and integrate the remote
changes
hint: (e.g. 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for
details.
```

### **Il faut faire les instructions en Gras**

```
iMac-de-nasr:Travaux C++ nasr$ rm -fr ".git/rebase-merge"
iMac-de-nasr:Travaux C++ nasr$ git pull --rebase
You are not currently on a branch.
Please specify which branch you want to rebase against.
See git-pull(1) for details.
```

git pull <remote> <branch>

```
iMac-de-nasr:Travaux C++ nasr$ git pull origine main
From https://github.com/karnasrr/travaux-c-
 * branch           main      -> FETCH_HEAD
 + 754af4d...ecbed36 main      -> origine/main  (forced update)
Already up to date.
iMac-de-nasr:Travaux C++ nasr$ git pull --rebase
You are not currently on a branch.
Please specify which branch you want to rebase against.
See git-pull(1) for details.
```

git pull <remote> <branch>

```
iMac-de-nasr:Travaux C++ nasr$ git init
```

```

Reinitialized existing Git repository in /Users/nasr/Documents/
PremierProjet/Travaux C++.git/
iMac-de-nasr:Travaux C++ nasr$ git add -A
iMac-de-nasr:Travaux C++ nasr$ git commit -m "Add your commit"
HEAD detached from 24e94ee
nothing to commit, working tree clean
iMac-de-nasr:Travaux C++ nasr$ git branch -M main
fatal: cannot rename the current branch while not on any.
iMac-de-nasr:Travaux C++ nasr$ git branch
* (HEAD detached from 24e94ee)
  main
iMac-de-nasr:Travaux C++ nasr$ git push origin main --force
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 6.04 MiB | 31.09 MiB/s, done.
Total 15 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
remote: This repository moved. Please use the new location:
remote:   https://github.com/karnasrr/travaux-c-.git
To https://github.com/kimnasr/travaux-c-.git
 + ecbed36...0b0a5ab main -> main (forced update)
iMac-de-nasr:Travaux C++ nasr$
```

Messahe 2 Resolve all conflicts manually, mark them as resolved with

```

warning: Cannot merge binary files: .DS_Store (HEAD vs. 4dbad7a (ajout
d'un repertoire Documentations Projet))
Auto-merging .DS_Store
CONFLICT (content): Merge conflict in .DS_Store
error: could not apply 4dbad7a... ajout d'un repertoire Documentations
Projet
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git
rebase --abort".
Could not apply 4dbad7a... ajout d'un repertoire Documentations Projet
iMac-de-nasr:Travaux C++ nasr$
iMac-de-nasr:Travaux C++ nasr$ git rebase --continue
.DS_Store: needs merge
You must edit all merge conflicts and then
mark them as resolved using git add
iMac-de-nasr:Travaux C++ nasr$ git add .DS_Store
iMac-de-nasr:Travaux C++ nasr$ git commit -m "merge"
[detached HEAD 821a5da] merge
 4 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 Documentation-Projets/Contenu cours Table Matiere C+
+-extrait decitre .pdf
  create mode 100644 Documentation-Projets/Programme C++, programmation
QT.pdf
```

```
create mode 100644 Documentation-Projets/cpp.pdf
iMac-de-nasr:Travaux C++ nasr$ git pull --rebase
You are not currently on a branch.
Please specify which branch you want to rebase against.
See git-pull(1) for details.
```

```
git pull <remote> <branch>
```

```
iMac-de-nasr:Travaux C++ nasr$ git pull dev main
From https://github.com/karnasrr/travaux-c-
 * branch           main      -> FETCH_HEAD
Already up to date.
iMac-de-nasr:Travaux C++ nasr$ git pull --rebase
You are not currently on a branch.
Please specify which branch you want to rebase against.
See git-pull(1) for details.
```

```
git pull <remote> <branch>
```

```
iMac-de-nasr:Travaux C++ nasr$ git init
Reinitialized existing Git repository in /Users/nasr/Documents/
PremierProjet/Travaux C++/.git/
iMac-de-nasr:Travaux C++ nasr$ git add -A
iMac-de-nasr:Travaux C++ nasr$ git branch -M main
fatal: cannot rename the current branch while not on any.
iMac-de-nasr:Travaux C++ nasr$ git push dev main --force
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 4 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (26/26), 36.65 MiB | 30.66 MiB/s, done.
Total 26 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
To https://github.com/karnasrr/travaux-c-.git
 + 0b0a5ab...8667c9a main -> main (forced update)
iMac-de-nasr:Travaux C++ nasr$ git log --oneline --graph --decorate --
all
* 821a5da (HEAD) merge
* 0b0a5ab (origine/main, origin/main) ajout fichier
* 24e94ee Ajout TP livre C++ moderne- TPn1-p253
* 754af4d (codes/main) Add documentations for supporting project Qt/C++.
| * 8667c9a (dev/main, main) commit
| | * 928c254 (refs/stash) WIP on main: b329b61 commit changes
| |
| | * 773e54e index on main: b329b61 commit changes
| |
| * b329b61 commit changes
| | * ecbed36 (program/main, or/main, codess/main) ajout documentations
| |
| * a8ed88d (BRANCHE) suppression d'un repertoire Documentations Projet
| * 4dbad7a ajout d'un repertoire Documentations Projet
|
* d7ce0f6 ajout d'un troisieme exemple(exemple2) QT-gestion de labels et
frames/melange avec les sources de Xcode
* b1e8603 ajout d'un troisieme exemple QT-gestion de labels et frames/
melange avec les sources de Xcode
```

- \* [70bf16a](#) ajout d'un deuxième exemple QT-gestion de labels et frames
- \* [b94bc3b](#) ajout d'un premier exemple QT-gestion de labels et frames
- \* [8dfde1c \(orig/main\)](#) troisième version,ajout de main.cpp
- \* [d4ee9f9](#) deuxième version,ajout de main.cpp
- \* [c8920cb](#) deuxième version
- \* [d894ceb](#) commit
- \* [7d579f9](#) first release
- \* [495bcd5](#) first commit