

Chapitre 1

Introduction

Auteur : Marco Lavoie

Adaptation : Sébastien Bois, Hiver 2021



Langage C++
25908 IFM



Objectifs

2

- Écrire des programmes simples en C++
- Utiliser des instructions d'entrées/sorties simples
- Se familiariser avec les types de données fondamentaux
- Utiliser les opérateurs arithmétiques
- Utiliser l'instruction de prise de décision
- Compiler et exécuter un programme



Aperçu

3

- Introduction à la programmation en C++
- Programmes simples
 - Commentaires
 - Directives de précompilation
 - Instructions d'entrées/sorties console
- Opérateurs arithmétiques
 - Priorité des opérateurs
- Structure conditionnelles `if/else`
 - Opérateurs relationnels
- *Microsoft Visual Studio*
 - Concevoir et exécuter un projet C++
- Conventions d'écriture pour ce cours



Un premier programme

4

- Programme `bonjour.cpp`

```
// Premier programme en C++
#include <iostream>

int main() {
    std::cout << "Bonjour\n";

    return 0; // Indique le succès
}
```

- Affiche dans la console

```
C:\>bonjour.exe
Bonjour
C:\>
```



Un premier programme (suite)

5

- Éléments de base du programme

- Commentaires

```
// Premier programme en C++
#include <iostream>

int main() {
    std::cout << "Bonjour\n";

    return 0; // Indique le succès au OS
}
```

- Deux formes de commentaires

- `//` commente ce qui suit jusqu'à la fin de la ligne
 - `/*` et `*/` commentent tout ce qu'il y a entre les deux

- Exemples équivalents

```
// Ligne #1      ⇔      /* Ligne #1
// Ligne #2                Ligne #2 */
```



Un premier programme (suite)

6

- Directives de précompilation

- Débutent toutes par **#** en début de ligne

```
// Premier programme en C++  
#include <iostream>
```

```
int main() {  
    std::cout << "Bonjour\n";
```

- **#include** indique

```
return 0; // Indique le succès au OS
```

```
}
```

d'inclure le fichier d'en-tête **<iostream>**

- Ce fichier contient le code source associé aux flux d'entrées/sorties tels que `std::cout`
 - Ce fichier est fourni avec le compilateur C++
- Si on oublie d'inclure le fichier **<iostream>**, le compilateur ne comprend pas `std::cout`



Un premier programme (suite)

7

- Fonction principale

- L'exécution débute à la première instruction de la fonction `main()`
- Le type de la valeur de retour (`int`) précède le nom de la fonction
- L'instruction `return` permet de quitter la fonction (en indiquant le résultat retourné)
- Les instructions sont regroupées en blocs entre `{` et `}`

```
// Premier programme en C++
#include <iostream>

int main() {
    std::cout << "Bonjour\n";

    return 0; // Indique le succès
}
```



Un premier programme (suite)

8

- Instruction de sortie

- `std::cout` fait référence au *flux de sortie* `cout` de l'*espace de nom* `std`

```
// Premier programme en C++
#include <iostream>

int main() {
    std::cout << "Bonjour\n";

    return 0; // Indique le succès
}
```

- L'espace de nom `std` est défini dans le fichier d'en-tête **<iostream>**
- L'opérateur `<<` dirige la chaîne **"Bonjour\n"** vers le flux de sortie `cout`
- `cout` reconnaît les séquences d'échappement

\n Retour de chariot

\" La guillemet (")

\t Tabulation

**** Le backslash (\)



Un second programme

9

- `addition.cpp` : addition de deux entiers

```
// Programme d'addition
#include <iostream>

int main()
{
    int entier1, entier2, somme;           // déclaration

    std::cout << "Entrez le premier entier\n"; // invite
    std::cin >> entier1;                   // lecture d'un entier
    std::cout << "Entrez le second entier\n"; // invite
    std::cin >> entier2;                   // lecture d'un entier
    somme = entier1 + entier2;             // affectation de la somme
    std::cout << "La somme vaut " << somme
              << std::endl;               // affiche la somme

    return 0; // indique que le programme s'est terminé avec succès
}
```



Un second programme (suite)

10

- Affiche dans la console

```
C:\>addition.exe
Entrez le premier entier
45
Entrez le second entier
72
La somme vaut 117
C:\>
```

- Éléments de base du programme

- Déclaration de variables

- `int` pour entier

- Autres types communs

`long, float, double, char`

```
int main()
{
    int entier1, entier2, somme;

    std::cout << "Entrez le prem
```



Un second programme (suite)

11

- Instruction de saisie (lecture)
 - `std::cin` fait référence au *flux d'entrée* `cin` de l'*espace de nom* `std`
 - L'opérateur `>>` dirige la valeur saisie du flux d'entrée `cin` vers la variable
- Notez la direction de l'opérateur de flux
 - Vers le flux de sortie : `std::cout << somme;`
 - En provenance du flux d'entrée : `std::cin >> entier1;`
 - Attention de ne pas confondre les deux
 - Ça ne compilera pas
 - Permettent aussi l'enchaînement

```
std::cout << "La somme vaut " << somme << std::endl;
std::cin >> entier1 >> entier2;
```



Un second programme (suite)

12

- Instruction de saisie (suite)
 - `std::cin` lit les touches entrées jusqu'à ce qu'un caractère « non valide » soit rencontré
 - Exemples : si l'input est **123.456**

<code>int a;</code> <code>std::cin >> a;</code>	}	<code>a = 123</code>
<code>float a;</code> <code>std::cin >> a;</code>	}	<code>a = 123.456</code>
<code>char a;</code> <code>std::cin >> a;</code>	}	<code>a = '1'</code>
<code>int a, b;</code> <code>char c;</code> <code>std::cin >> a >> c >> b;</code>	}	<code>a = 123, c = '.' et b = 456</code>



Un second programme (suite)

13

- Notes supplémentaires

- `std::endl` correspond à `"\n"`

```
std::cout << "La somme vaut " << somme << std::endl;
```

- Une instruction peut se propager sur plus d'une ligne

- Le point-virgule (;) délimite les instructions
 - Une ligne peut même contenir plus d'une instruction

```
int v; std::cout << "Valeur? "; std::cin >> v;
```

- Cependant c'est fortement déconseillé



Pourquoi tous ces `std::` ?

14

- On peut éviter de répéter `std::` en utilisant **`using`**

```
// Programme d'addition
#include <iostream>

using std::cin;
using std::cout;
using std::endl;

int main()
{
    int entier1, entier2, somme;

    cout << "Entrez le premier entier\n";
    cin >> entier1;
    cout << "Entrez le second entier\n";
    cin >> entier2;
    somme = entier1 + entier2;
    cout << "La somme vaut " << somme
        << endl;

    return 0;
}
```



Opérateurs arithmétiques

15

- Opérateurs de base
 - Supposons : `int a = 21;`

Opérateur	Opération	Exemple	Résultat
+	Addition	<code>a + 7</code>	28
-	Soustraction	<code>a - 7</code>	14
*	Multiplication	<code>a * 7</code>	147
/	Division	<code>a / 7</code>	3
()	Parenthèses	<code>(a + 5) * 2</code>	52
%	Modulo	<code>(a - 2) % 7</code>	5



Opérateurs arithmétiques (suite)

16

- Particularités de la division
 - Si les deux opérandes sont entiers, alors une division entière est effectuée
 - La partie fractionnelle du résultat est ignorée

```
int    i = 22;  
float  f = 22.0;
```

```
std::cout << i / 5    << std::endl  
          << f / 5    << std::endl  
          << i / 5.    << std::endl;
```

```
4  
4.4  
4.4
```

- **Attention** : *la division entière est souvent cause d'erreurs de programmation*



Opérateurs arithmétiques (suite)

17

- Priorité des opérateurs mathématiques

Opérateurs	Niveau de priorité*	Ordre d'évaluation
()	2	Leur contenu est évalué en premier. Lorsque imbriquées, celles les plus intérieures sont évaluées en premier. Celles au même niveau sont évaluées de gauche à droite.
*, / ou %	6	Évalués en second, de gauche à droite.
+ ou -	7	Évalués en dernier, de gauche à droite.

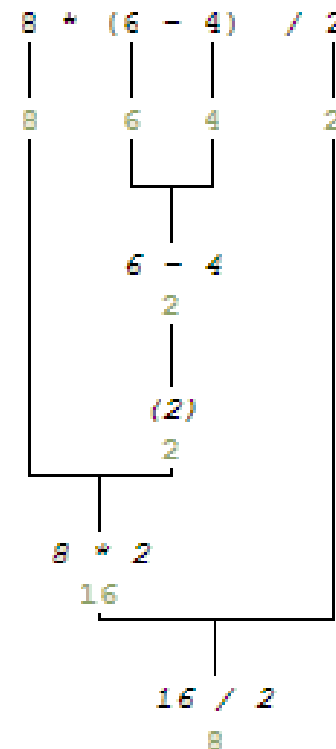
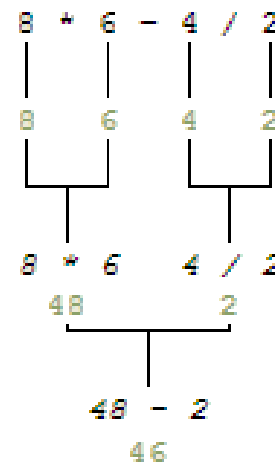
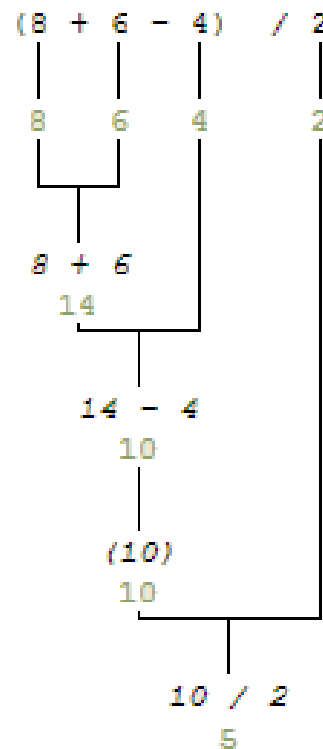
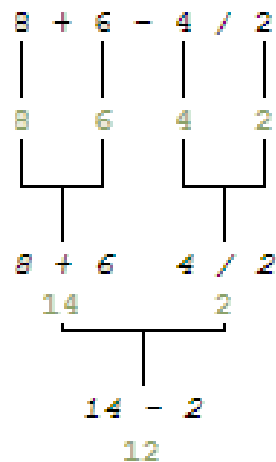
** Niveaux de priorité 1 à 18, 1 étant la priorité la plus élevée*



Opérateurs arithmétiques (suite)

18

- Exemples d'évaluation





Structure conditionnelle

19

- Opérateurs relationnels

- Exploités pour exprimer des conditions

Opérateur	Opération	Exemple
>	Plus grand que	a > 7
<	Plus petit que	a < 7
>=	Plus grand ou égale à	a >= 7
<=	Plus petit ou égale à	a <= 7
==	Égalité	a == 7
!=	Inégalité	a != 7

- Erreurs de programmation courantes

- Confondre l'opérateur d'égalité (==) avec celui d'affectation (=)
- Plus grand ou égal est >=, et non =>



Structure conditionnelle (suite)

20

- Le `if/if-else` de base

```
// Programme d'addition
#include <iostream>

int main()
{
    int entier1, entier2;                // déclaration

    std::cout << "Entrez deux entiers: "; // invite
    std::cin >> entier1 >> entier2;      // lecture d'un entier

    // Afficher la plus grande des deux valeurs
    std::cout << "La plus grande valeur est ";
    if ( entier1 > entier2 )
        std::cout << entier1 << std::endl;
    else
        std::cout << entier2 << std::endl;

    return 0; // indique que le programme s'est terminé avec succès
}
```



Structure conditionnelle (suite)

21

- Erreurs de programmation courantes

- Confondre = avec ==

```
a = 0;  
if ( a = 0 )  
    std::cout << "a est 0" << std::end;
```

N'affiche pas a est 0

- Le point-virgule de trop

```
a = 1;  
if ( a == 0 );  
    std::cout << "a est 0" << std::end;
```

Affiche a est 0

- Le bloc manquant ({ })

```
a = 0;  
if ( a == 1 ) {  
    std::cout << "a est 1" << std::end;  
    std::cout << "a invalide" << std::end;  
}
```

Le deuxième std::cout
n'est pas dans le if



Priorité des opérateurs

22

- Incluant les opérateurs relationnels, de flux et l'affectation

Opérateurs	Niveau de priorité*	Associativité
()	2	De gauche à droite
* / %	6	De gauche à droite
+ -	7	De gauche à droite
<< >>	8	De gauche à droite
< <= > >=	9	De gauche à droite
== !=	10	De gauche à droite
=	17	De droite à gauche

- Exemple

```
int a, b, c, d;  
a = b = c = d = 1 + 1; // initialise toutes les variables à 2
```



Environnement de développement

23

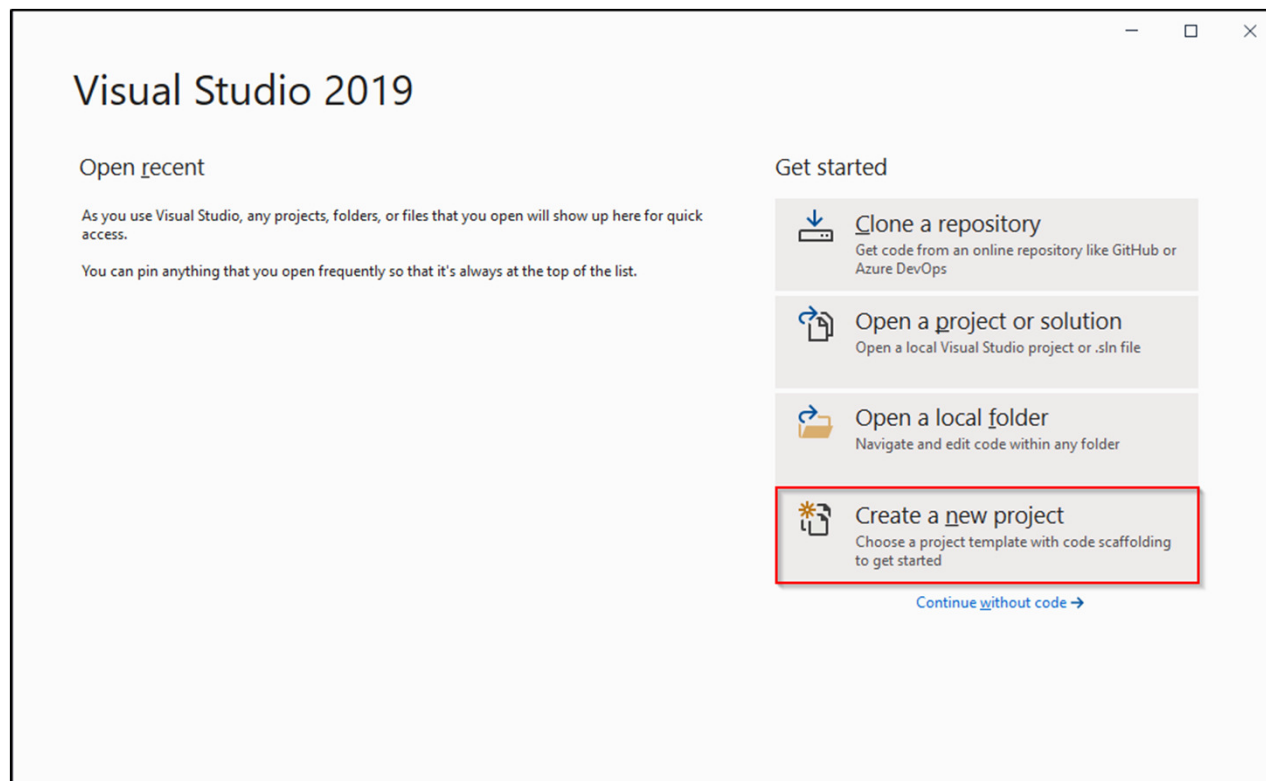
- Le langage C++ est disponible sur la plupart des plate-formes
 - **Windows** : plusieurs fournisseurs (*Microsoft, Waterloo, Intel, GNU Open Source, etc.*)
 - **Linux** : gcc (*GNU Open Source*)
- Dans le cours 25908 IFM, nous utilisons *Microsoft Visual Studio*
 - Interface graphique convivial
 - Outils de débogage sophistiqués
 - Couramment exploité dans l'industrie



Microsoft Visual Studio

24

- Accueil





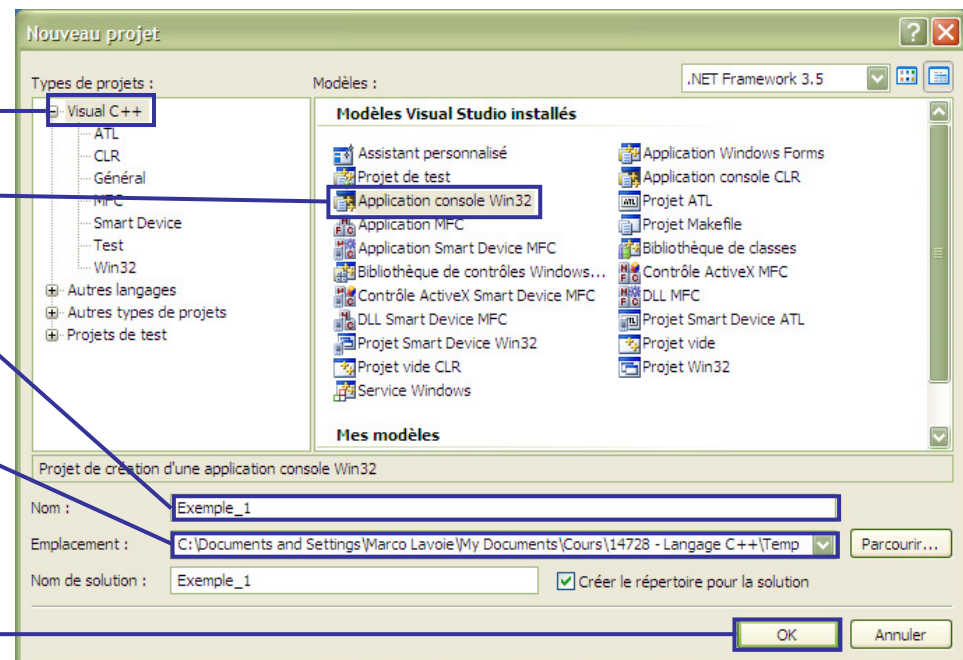
Visual Studio

Créer un nouveau projet

25

- Créer un nouveau projet console C++
 - Menu : Fichier » Nouveau » Projet...
 - Type de projet :

1. Langage C++
2. Application console
3. Nom du projet
4. Répertoire où stocker le projet
5. Passer à l'étape suivante



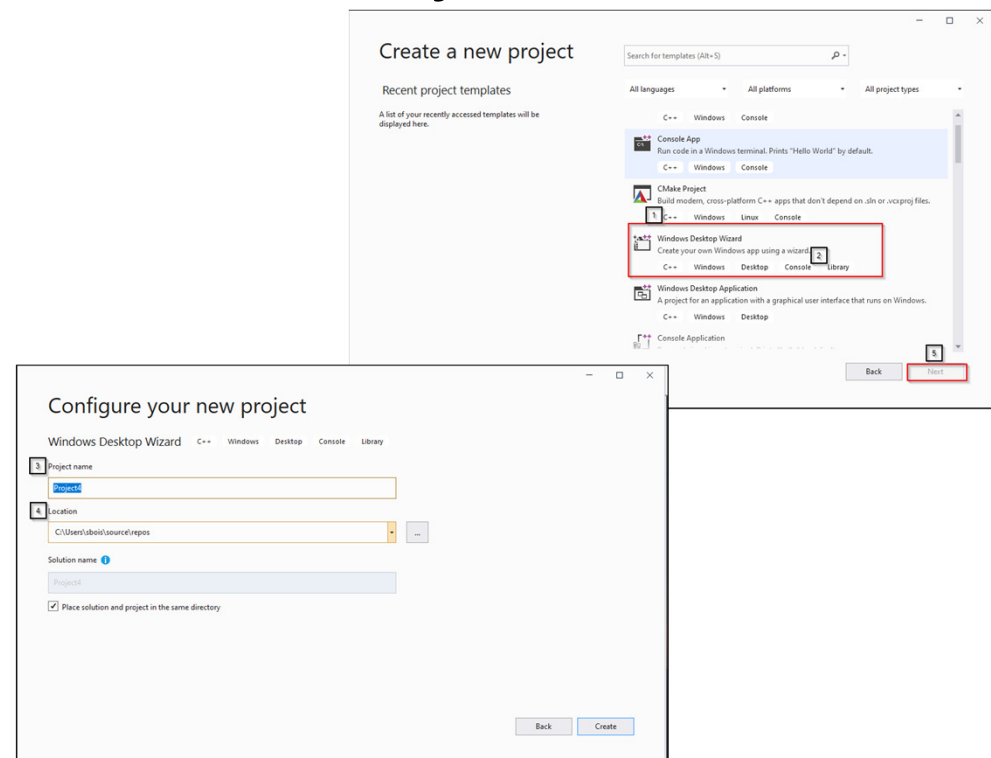


Visual Studio 2019

Créer un nouveau projet

26

- Créer un nouveau projet console C++
 - Menu : Fichier » Nouveau » Projet...
 - Type de projet :
 1. Langage C++
 2. Application console
 3. Nom du projet
 4. Répertoire où stocker le projet
 5. Passer à l'étape suivante



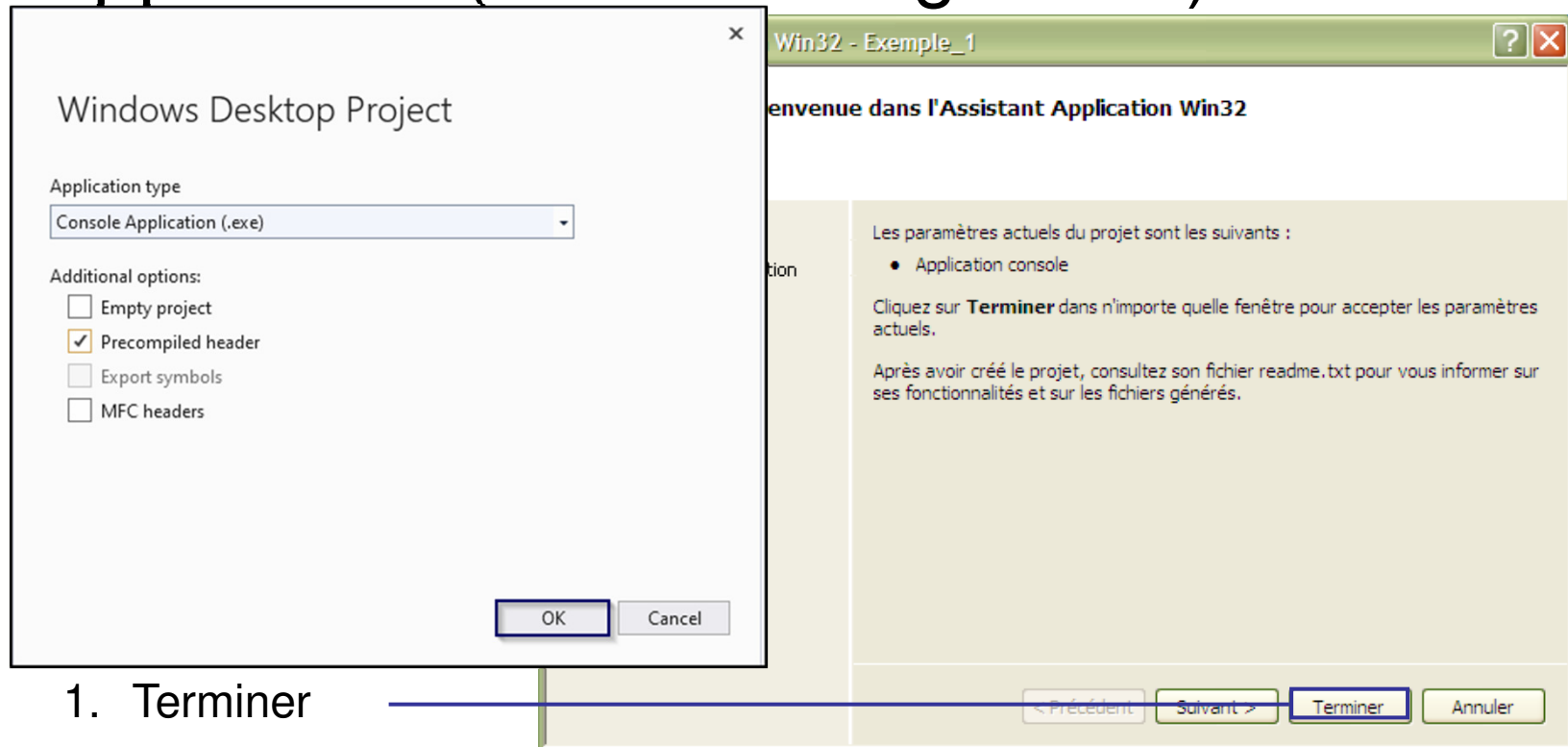


Visual Studio

Créer un nouveau projet (suite)

27

- Accepter les paramètres par défaut de l'application (aucun changement)





Visual Studio

Créer un nouveau projet (suite)

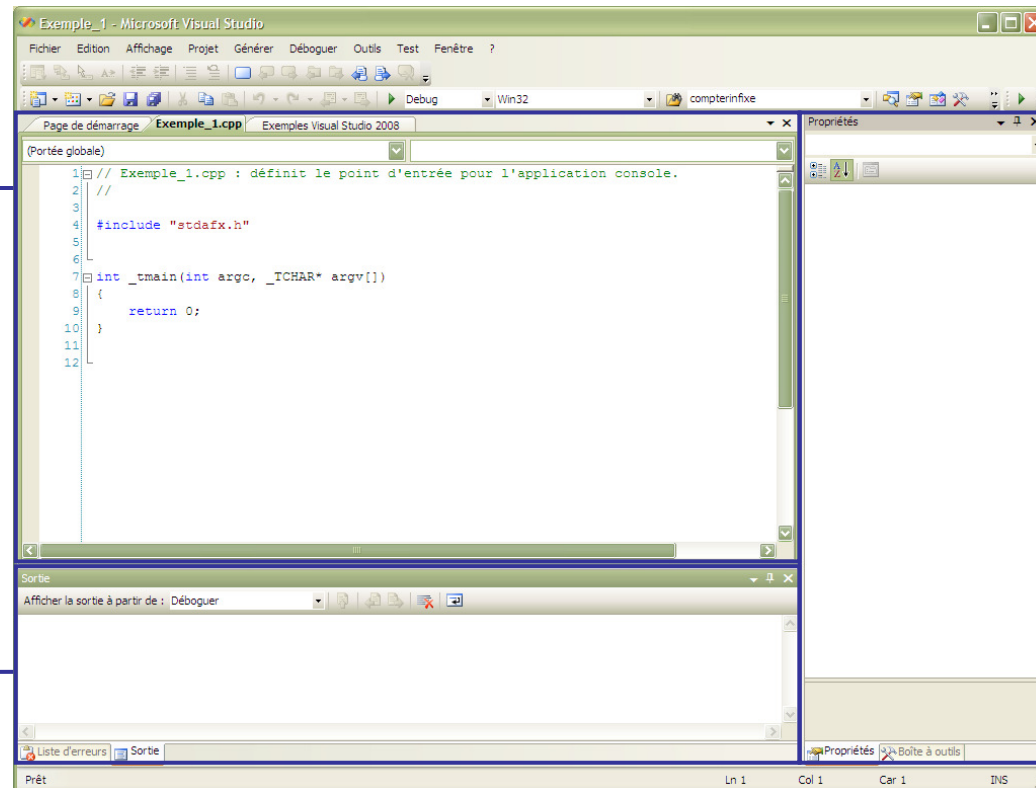
28

- Principaux composants de l'environnement de développement

Éditeur

Sorties

Explorateur
de solution





Visual Studio

Qu'est-ce qu'une solution?

29

- *Visual Studio* permet de regrouper plusieurs projets dans une même *solution*
 - Chaque projet peut utiliser un langage de programmation distinct
 - Des projets de tests peuvent être inclus dans la solution
 - Des projets peuvent partager des fichiers communs
- Dans le cadre de 25908 IFM, nous concevons des solutions à un seul projet

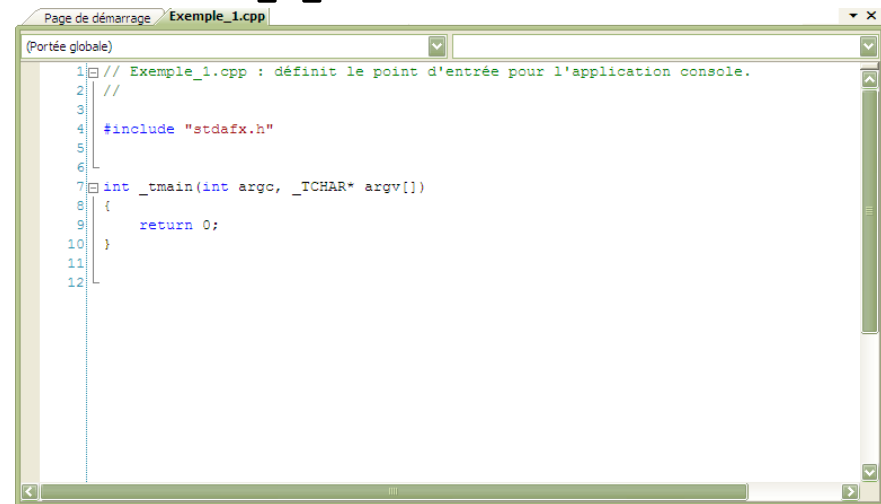


Visual Studio

Squelette de code source

30

- Le projet contient un fichier `.cpp` contenant la routine `main()`
 - Le nom de ce fichier correspond au nom du projet
 - Le fichier en-tête `"stdafx.h"` est requis par VS (`"pch.h"` dans VS 2019)
 - L'en-tête de la routine principale est unique à VS
 - Nous verrons plus tard le rôle de `argc` et `argv[]`



```
1 // Exemple_1.cpp : définit le point d'entrée pour l'application console.
2 //
3
4 #include "stdafx.h"
5
6
7 int _tmain(int argc, _TCHAR* argv[])
8 {
9     return 0;
10 }
11
12
```


```
int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}
```

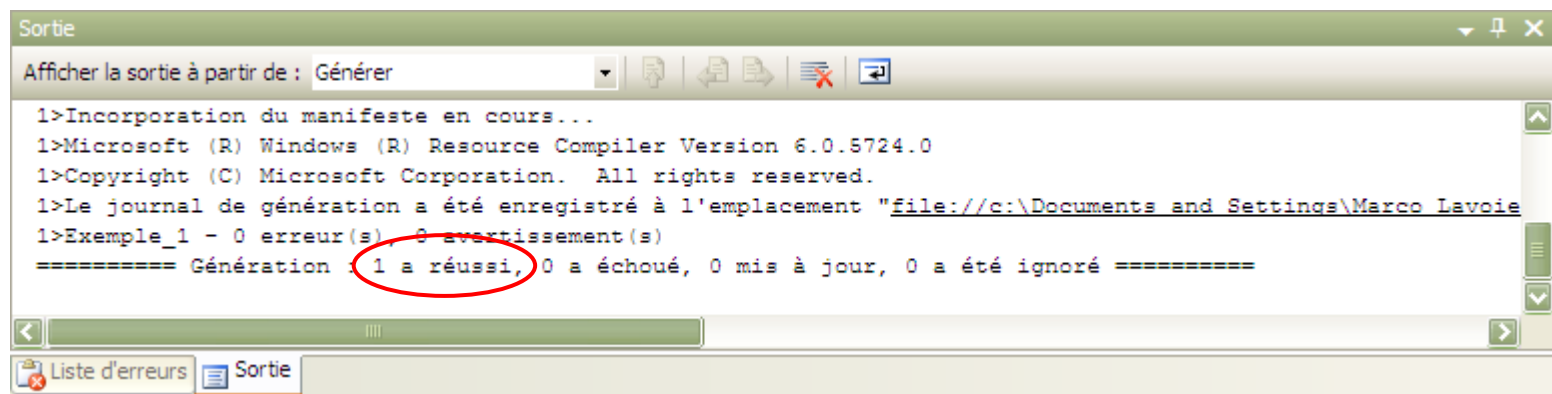


Visual Studio

Compilation du projet

31

- La compilation s'assure qu'il n'y a pas d'erreurs de syntaxe
- Menu : **Générer » Générer** la solution
 - Alternativement, pressez la touche 
 - La fenêtre de sorties affiche les résultats de la compilation (dont les messages d'erreurs s'il y a lieu)




```
Sortie
Afficher la sortie à partir de : Générer
1>Incorporation du manifeste en cours...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0
1>Copyright (C) Microsoft Corporation. All rights reserved.
1>Le journal de génération a été enregistré à l'emplacement "file:///c:/Documents and Settings/Marco Lavoie
1>Exemple_1 - 0 erreur(s), 0 avertissement(s)
===== Génération : 1 a réussi, 0 a échoué, 0 mis à jour, 0 a été ignoré =====
```



Visual Studio

Exécution du projet

32

- Lorsque le projet est compilé avec succès
- Menu : **Déboguer » Démarrer** le débogage
 - Alternativement, pressez la touche 
 - On verra plus tard la distinction entre déboguer et exécuter
- L'exécution est rapide
 - La console d'exécution n'est que brièvement visible car le programme ne fait rien!

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    return 0;  
}
```




Visual Studio

Exécution du projet (suite)

33

- Pour éviter que la console d'exécution soit fermée en fin d'exécution

```
// Exemple_1.cpp : définit le point d'entrée pour l'application console
//

#include "stdafx.h"
#include <iostream>
#include <conio.h>

int _tmain(
{
    // Attendre confirmation pour fermer la console
    std::cout << "Pressez une touche pour terminer..." << std::endl;
    _getch();

    return 0;
}
```



Conventions d'écriture

34

- Similaires à ce que vous avez probablement vu dans d'autres cours ...
 - Chaque fichier doit débuter par un commentaire d'en-tête

```
/*=====
Programmeur : Marco Lavoie
Fichier      : Exemple_1.cpp
Description  : Programme d'addition de deux entiers
Date        : 2010/01/01
=====*/
```

- Le programme doit être suffisamment documenté pour permettre à un individu, autre que celui qui l'a écrit, de comprendre le fonctionnement du code



Tabulation



Conventions d'écriture (suite)

36

- Chaque structure de contrôle moindrement complexe doit être précédée d'un commentaire décrivant son rôle

```
// Contenu de source copié à l'envers
// dans dest
for( int i=0 ; i<n ; i++ )
    dest[i] = source[n-i];
```

- Chaque routine doit être précédée d'un commentaire descriptif

```
// Routine principale d'addition
int main()
{
    int entier1, entier2, somme;                // déclaration

    std::cout << "Entrez le premier entier\n"; // invite
```



Conventions d'écriture (suite)

37

- Nomenclature des identificateurs
 - **Les noms de classes:** Commencent par une majuscule et peuvent être formés par la concaténation de plusieurs mots. La première lettre de chaque mot est majuscule et toutes les autres sont minuscules. (ex: **Graphics**, **VolCommercial**, **IOException**, etc.)
 - **Les noms de méthodes (fonctions) et de variables:** Même chose que pour les noms de classes sauf que la première lettre doit être minuscule. (ex: **getGraphics**, **calculerLaMoyenne**, **moyenne**, etc.)



Conventions d'écriture (suite)

38

- Nomenclature des identificateurs (suite)
 - **Les noms de variables constantes:** Toutes les lettres sont majuscules et les mots, s'il y en a plus d'un, sont séparés par des caractères de soulignement. (ex: **PI**, **E**, **NOMBRE_MAX**, etc.)
 - Un identificateur ne doit jamais contenir plus de 31 caractères (lisibilité et portabilité)
- Le professeur considérera toujours ces conventions lors de la correction des travaux
 - *Un travail qui ne respecte pas l'une ou l'autre des conventions prescrites peut ne pas obtenir une note parfaite*



Exercice #1.1

39

- Solutionnez l'exercice distribué par l'instructeur
 - Créer un nouveau projet console VS
 - Solutionner le problème tel que décrit
 - N'oubliez pas les conventions d'écriture
 - Soumettez votre projet selon les indications de l'instructeur
 - **Attention** : respectez à la lettre les instruction de l'instructeur sur la façon de soumettre vos travaux, *sinon la note EC sera attribuée à ceux-ci*



Erreurs de programmation

40

- Ne pas oublier d'inclure **<iostream>**
- Attention à la division entière (/) lorsque les deux opérandes sont entiers
- Ne pas confondre l'affectation (=) avec l'opérateur d'égalité (==)
- Ne pas oublier qu'une expression est évaluée en fonction de la priorité de ses opérateurs
- Attention au délimiteur d'instructions (;)
- Ne pas oublier les accolades pour délimiter les blocs d'instructions ({ })



Bonnes pratiques de programmation

41

- Exploiter judicieusement l'indentation
- Respecter les convention d'écriture
- Il n'y a jamais trop de commentaires
- Aérer le code
 - en insérant judicieusement des lignes vides
 - en insérant des espaces entre les opérateurs et après les virgules d'une expression
 - en évitant d'avoir plusieurs instructions par ligne
 - en répartissant une longue instruction sur plusieurs lignes
- En cas de doute sur les priorités d'évaluation, exploiter les parenthèses



Exercice #1.2

42

- Solutionnez l'exercice distribué par l'instructeur
 - Créer un nouveau projet console VS
 - Solutionner le problème décrit, mais avec la variante suivante
 - *Déterminer les équivalents entiers de toutes les lettres majuscules, minuscules ainsi que ceux de tous les chiffres*
 - N'oubliez pas les conventions d'écriture
 - Soumettez votre projet selon les indications de l'instructeur
 - **Attention** : respectez à la lettre les instructions de l'instructeur sur la façon de soumettre vos travaux, *sinon la note EC sera attribuée à ceux-ci*



Pour la semaine prochaine...

43

- Vous devez relire le contenu de la présentation du chapitre 1
 - Il y aura un **quiz** sur ce contenu au prochain cours
 - À livres et ordinateurs fermés (5 minutes donc vous connaissez la réponse ou pas)