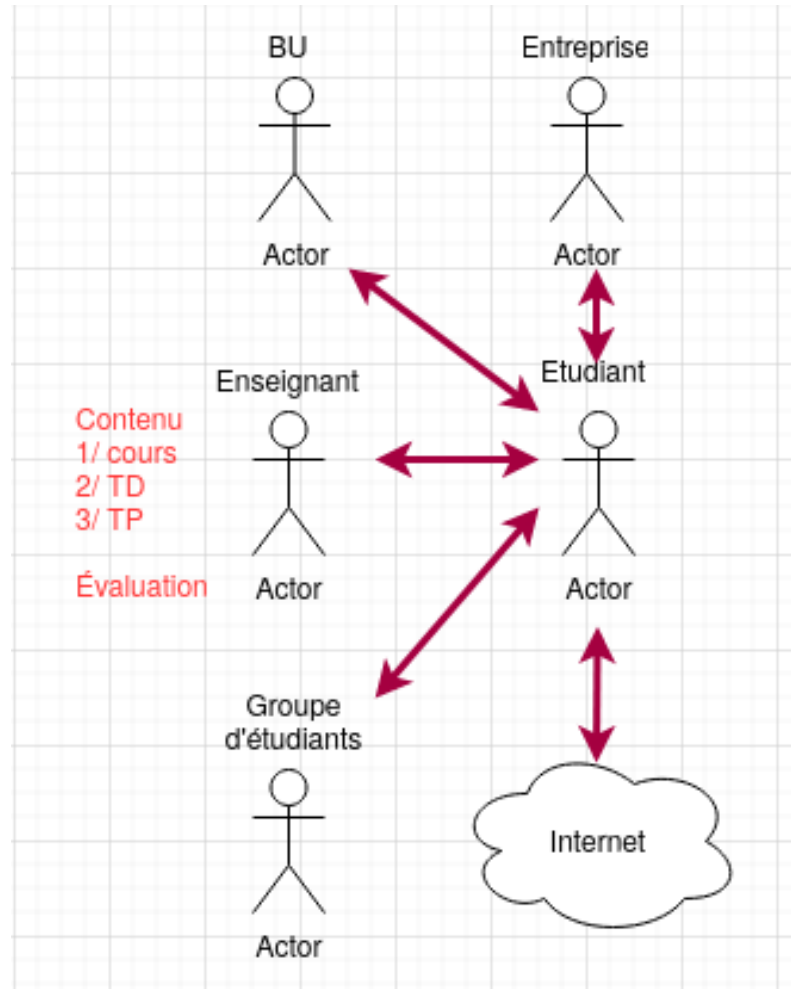


POO & C++

Programmation Orientée Objet & Langage C++

Le socio-constructivisme



POO

La programmation orientée objet (POO) est un modèle de langage de programmation (on parle de paradigme) qui s'articule autour d'objets et de données, plutôt que d'actions et de logique.

« Par le passé », un programme était une procédure logique qui récupérait des données en entrée, les traitait puis produisait des données en sortie.

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

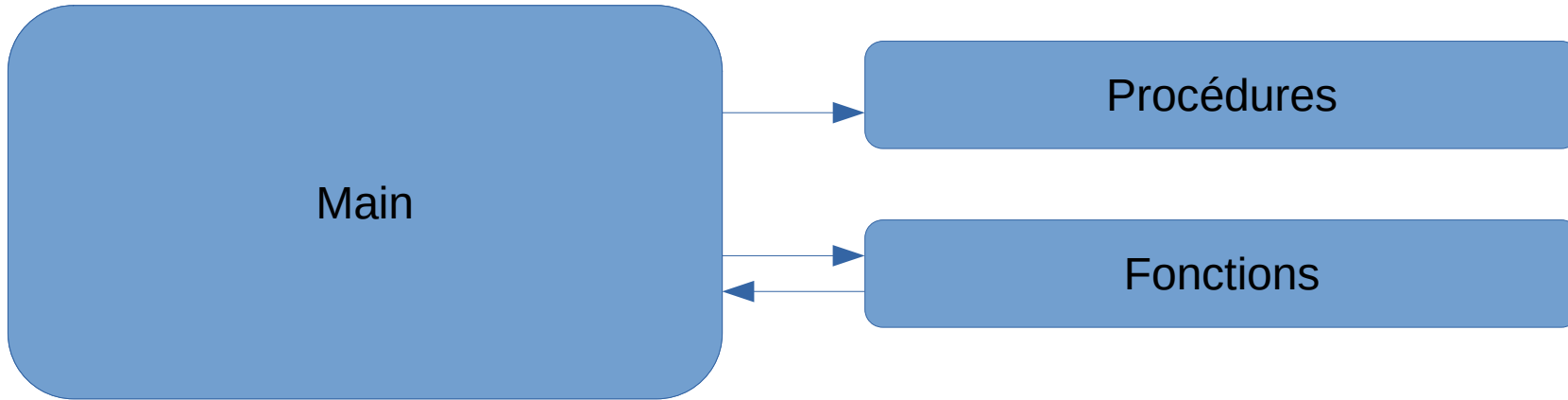
Les programmations

Il existe 2 types de programmation :

- La programmation procédurale
- La programmation orientée objet

Les programmations

- La programmation procédurale
 - La programmation orientée objet



Programmation procédurale

Étudions un exemple

Programmation procédurale

```
#include <iostream>
#include <cstring> // Pour strcpy et strlen
```

```
using namespace std;
```

```
// Définition d'une structure pour stocker les informations d'un étudiant
```

```
struct Etudiant {
    char nom[50];
    int age;
    float moyenne;
};
```

en procédurale

```
#include <iostream>
#include <cstring> // Pour strcpy et strlen

using namespace std;

// Définition d'une structure pour stocker les informations d'un étudiant
struct Etudiant {
    char nom[50];
    int age;
    float moyenne;
};

// Définition de constantes
const int MAX_ETUDIANTS = 100;
int nombreEtudiants = 0;
Etudiant listeEtudiants[MAX_ETUDIANTS]; // Tableau statique pour stocker les étudiants
```


Programmation procédurale

```
// Fonction pour ajouter un étudiant
void ajouterEtudiant(const char* nom, int age, float moyenne) {
    if (nombreEtudiants < MAX_ETUDIANTS) {
        strcpy(listeEtudiants[nombreEtudiants].nom, nom);
        listeEtudiants[nombreEtudiants].age = age;
        listeEtudiants[nombreEtudiants].moyenne = moyenne;
        nombreEtudiants++;
        cout << "Etudiant ajouté avec succès.\n";
    } else {
        cout << "Liste d'étudiants pleine.\n";
    }
}
```

Programmation procédurale

```
// Fonction pour afficher la liste des étudiants
void afficherEtudiants() {
    if (nombreEtudiants == 0) {
        cout << "Aucun étudiant enregistré.\n";
        return;
    }

    cout << "\nListe des étudiants :\n";
    for (int i = 0; i < nombreEtudiants; i++) {
        cout << "Nom : " << listeEtudiants[i].nom
            << ", Age : " << listeEtudiants[i].age
            << ", Moyenne : " << listeEtudiants[i].moyenne << endl;
    }
}
```

Programmation procédurale

```
int main() {  
    ajouterEtudiant("Alice", 20, 15.5);  
    ajouterEtudiant("Bob", 22, 12.3);  
    ajouterEtudiant("Charlie", 19, 14.8);  
  
    afficherEtudiants();  
  
    return 0;  
}
```

Programmation procédurale

Explication :

- Structure **Etudiant** : Regroupe les informations d'un étudiant.
- Fonction **ajouterEtudiant** : Ajoute un étudiant au tableau statique.
- Fonction **afficherEtudiants** : Affiche la liste des étudiants.
- Tableau **listeEtudiants** : Utilisé pour stocker un nombre fixe d'étudiants (approche procédurale).
- **main()** : Ajoute trois étudiants et affiche la liste.

Pourquoi c'est une approche procédurale ?

- Pas de programmation orientée objet (pas de classes ni d'encapsulation).
- Utilisation de structures pour stocker des données, mais sans méthodes associées.
- Toutes les actions sont réalisées à travers des fonctions globales.

La programmation orientée objet

Principaux changements :

- Encapsulation des données : Les attributs (variables) sont privés et accessibles via des méthodes (fonctions) public => l'encapsulation permet une meilleure hiérarchie et protection des données.
- Utilisation de méthodes pour gérer les étudiants.
- Les méthodes pourront être stockées dans d'autres fichiers (ou librairies)
- Stockage dynamique avec **vector** pour plus de flexibilité.

La programmation orientée objet

```
#include <iostream>
#include <vector>
using namespace std;
// Classe représentant un étudiant
class Etudiant {
private:
    string nom;
    int age;
    float moyenne;
```

La programmation orientée objet

```
#include <iostream>
#include <vector>
using namespace std;

// Classe représentant un étudiant
class Etudiant {
private:
    string nom;
    int age;
    float moyenne;
public:
    // Constructeur
    Etudiant(string _nom, int _age, float _moyenne) {
        nom = _nom;
        age = _age;
        moyenne = _moyenne;
    }
```

```
// Méthode pour afficher les informations de l'étudiant
void afficher() const {
    cout << "Nom : " << nom
        << ", Age : " << age
        << ", Moyenne : " << moyenne << endl;
}
```

};

Fin de la classe **Etudiant**

On a créé une classe **Etudiant**, avec un constructeur pour créer des étudiants et une méthode (fonction) pour afficher les étudiants.
Les attributs sont protégés et ne sont accessibles que lors de la création des étudiants (objets)

La programmation orientée objet

```
#include <iostream>
#include <vector>
using namespace std;

// Classe pour gérer une liste d'étudiants
class GestionEtudiants {
private:
    vector<Etudiant> listeEtudiants; // Utilisation d'un vecteur pour stocker

public:
    // Ajouter un étudiant à la liste
    void ajouterEtudiant(string nom, int age, float moyenne) {
        listeEtudiants.push_back(Etudiant(nom, age, moyenne));
        cout << "Etudiant ajouté avec succès.\n";
    }
};
```

```
// Afficher tous les étudiants
void afficherEtudiants() const {
    if (listeEtudiants.empty()) {
        cout << "Aucun étudiant enregistré.\n";
        return;
    }
    cout << "\nListe des étudiants :\n";
    for (const auto& etudiant : listeEtudiants) {
        etudiant.afficher();
    }
}

};
```

Fin de la classe **GestionEtudiants**

};

La programmation orientée objet

```
// Fonction principale  
int main() {  
    GestionEtudiants gestion;  
  
    // Ajout d'étudiants  
    gestion.ajouterEtudiant("Alice", 20, 15.5);  
    gestion.ajouterEtudiant("Bob", 22, 12.3);  
    gestion.ajouterEtudiant("Charlie", 19, 14.8);  
  
    // Affichage de la liste des étudiants  
    gestion.afficherEtudiants();  
  
    return 0;  
}
```

→ Voir exemple1.cpp

Comparaison de la version procédurale et de la version objet

	Approche procédurale	Approche objet
stockage	tableau statique Etudiant listeEtudiants[MAX_ETUDIANTS]	Stockage dynamique vector<Etudiant>
Encapsulation	Aucune, accès direct aux données	Données privées, accès via méthodes
Gestion des étudiants	Fonctions globales	Méthodes de la classe GestionEtudiants
Réutilisabilité	Faible	Élevée (la classe peut être réutilisée ailleurs)

Avantages de l'approche orientée objet

Encapsulation : Meilleure protection des données.

Extensibilité : Facile à ajouter de nouvelles fonctionnalités (ex: tri, recherche).

Flexibilité : vector permet d'ajouter autant d'étudiants que nécessaire.

Lisibilité : Code mieux structuré et plus modulaire ==>> permet un travail en équipe.

C++

Le C++ moderne est en pleine expansion. On le retrouve sur :

- les objets connectés (IoT)
- Les smartphones;
- Les postes de travail;
- Les serveurs;
- Le cloud et l'intelligence artificielle (IA).

C++ est le langage le plus puissant et le plus rapide qui existe. Il tire parti au maximum de l'architecture matérielle et des processeurs avec une empreinte mémoire limitée.

C++

Qu'est-ce que le C++ ?

Le C++ est un langage de programmation orienté objet, développé par Bjarne Stroustrup en 1983. Il est basé sur le langage C et offre des fonctionnalités avancées comme la programmation orientée objet, la gestion fine de la mémoire et l'utilisation de bibliothèques standard et tierces.

Différences entre Python et C++

- Typage : C++ est fortement typé, contrairement à Python.
- Gestion de la mémoire : Le C++ n'a pas de ramasse-miettes automatique comme Python.
- Compilé vs Interprété : Le C++ est un langage compilé, ce qui permet une exécution plus rapide.
- Syntaxe : Plus stricte que Python.

C++

Structure de base d'un programme

```
#include <iostream>
using namespace std;

int main() {
    cout << "Bonjour, monde !" << endl;
    return 0;
}
```

Structure de base d'un programme

C++

```
#include <iostream>
using namespace std;

int main() {
    cout << "Bonjour, monde !" << endl;
    return 0;
}
```

Variables et Types

int, double, char, bool, string

Types avancés : long, unsigned, float, wchar_t

NB: déclaration obligatoire (typage fort)

&

variables locales vs variables globales....

Structure de base d'un programme

```
#include <iostream>
using namespace std;

int main() {
    cout << "Bonjour, monde !" << endl;
    return 0;
}
```

C++

Variables et Types

int, double, char, bool, string

Types avancés : long, unsigned, float, wchar_t

NB: déclaration obligatoire (typage fort)

&

variables locales vs variables globales....

Opérateurs

Arithmétiques : +, -, *, /, %

Comparaison : ==, !=, <, >, <=, >=

Logiques : &&, ||, !

Bits : &, |, ^, <<, >>

Structure de base d'un programme

```
#include <iostream>
using namespace std;

int main() {
    cout << "Bonjour, monde !" << endl;
    return 0;
}
```

Opérateurs

Arithmétiques : +, -, *, /, %

Comparaison : ==, !=, <, >, <=, >=

Logiques : &&, ||, !

Bits : &, |, ^, <<, >>

C++

Variables et Types

int, double, char, bool, string

Types avancés : long, unsigned, float, wchar_t

Conditions et Boucles

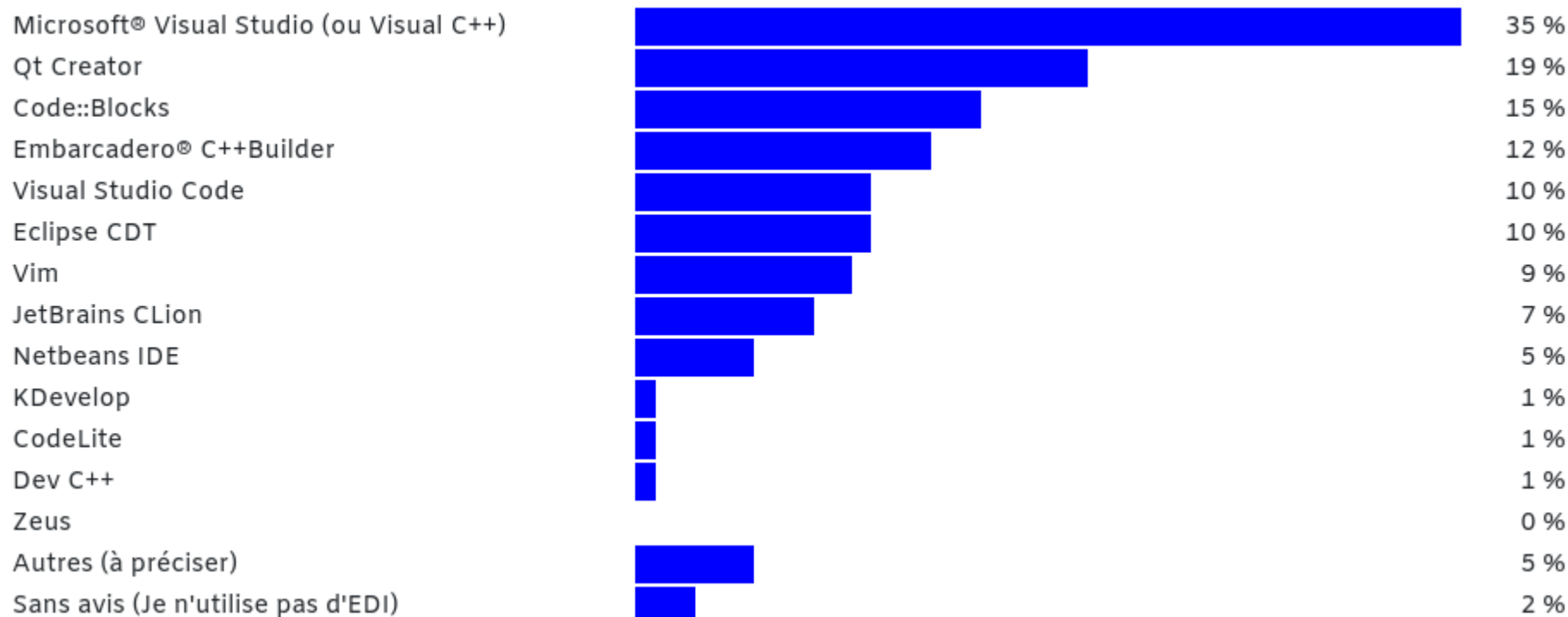
```
if (a > b) {
    cout << "a est plus grand que b";
} else {
    cout << "b est plus grand";
}
```

```
for (int i = 0; i < 10; i++) {
    cout << i << " ";
}
while (condition) {
    // Instructions
}
do {
    // Instructions
} while (condition);
```

C++

Les EDI

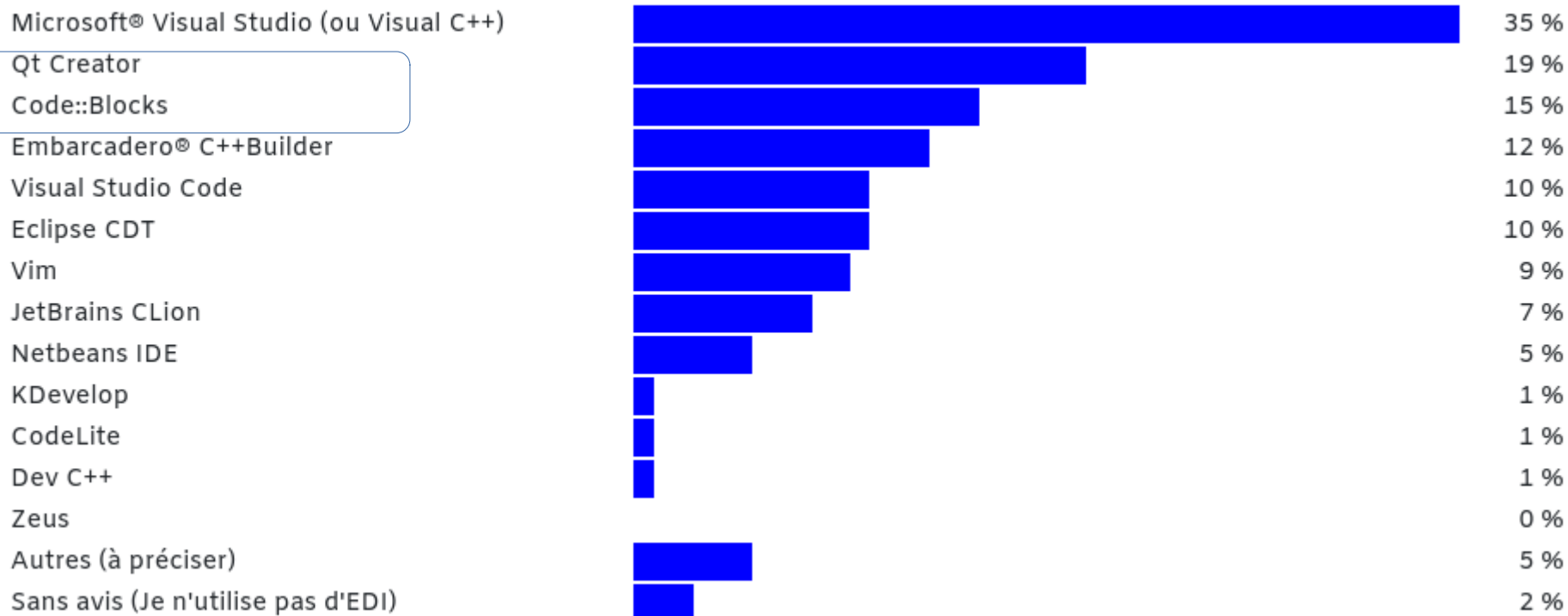
Quels environnements de développement intégrés C/C++ utilisez-vous en 2018 ? Et pourquoi ?



C++

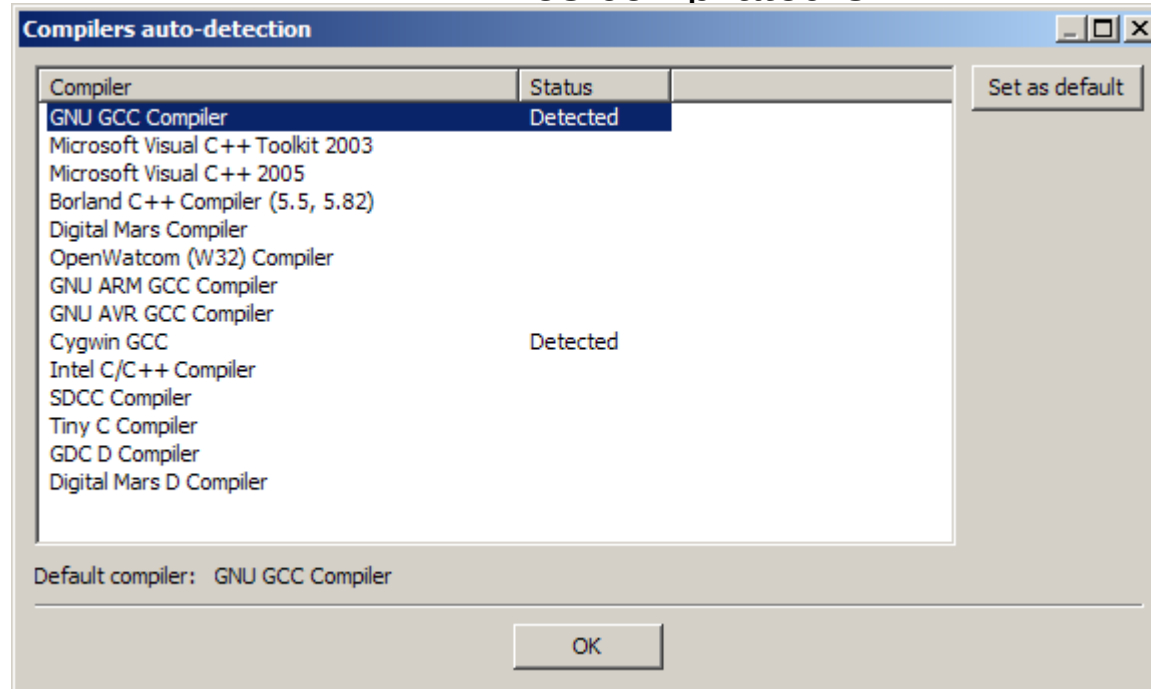
Les EDI

Quels environnements de développement intégrés C/C++ utilisez-vous en 2018 ? Et pourquoi ?



C++

Les compilateurs



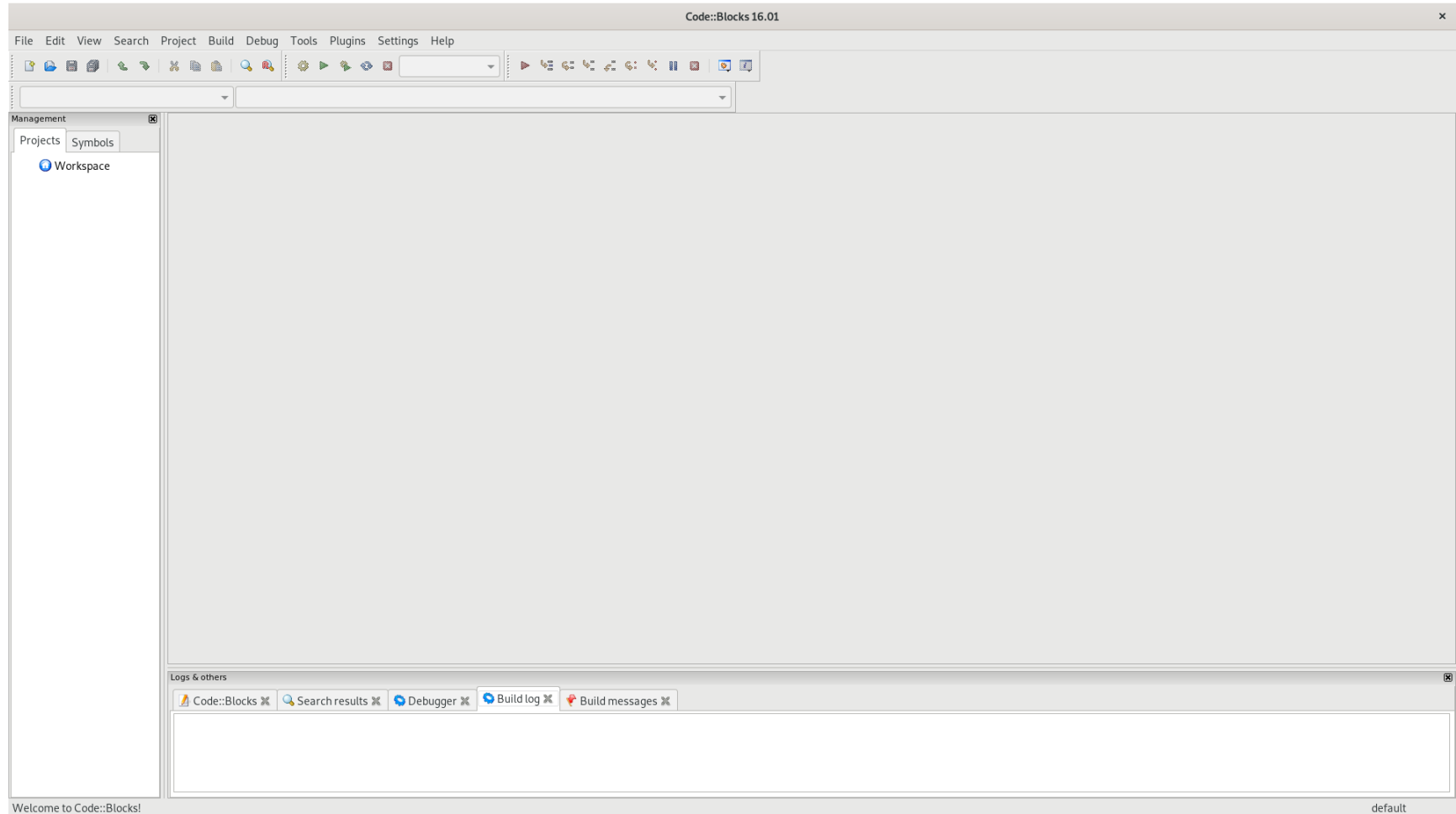
GNU Compiler Collection ou GCC est capable de compiler le C, C++,.....

→ Il est multi-plateforme

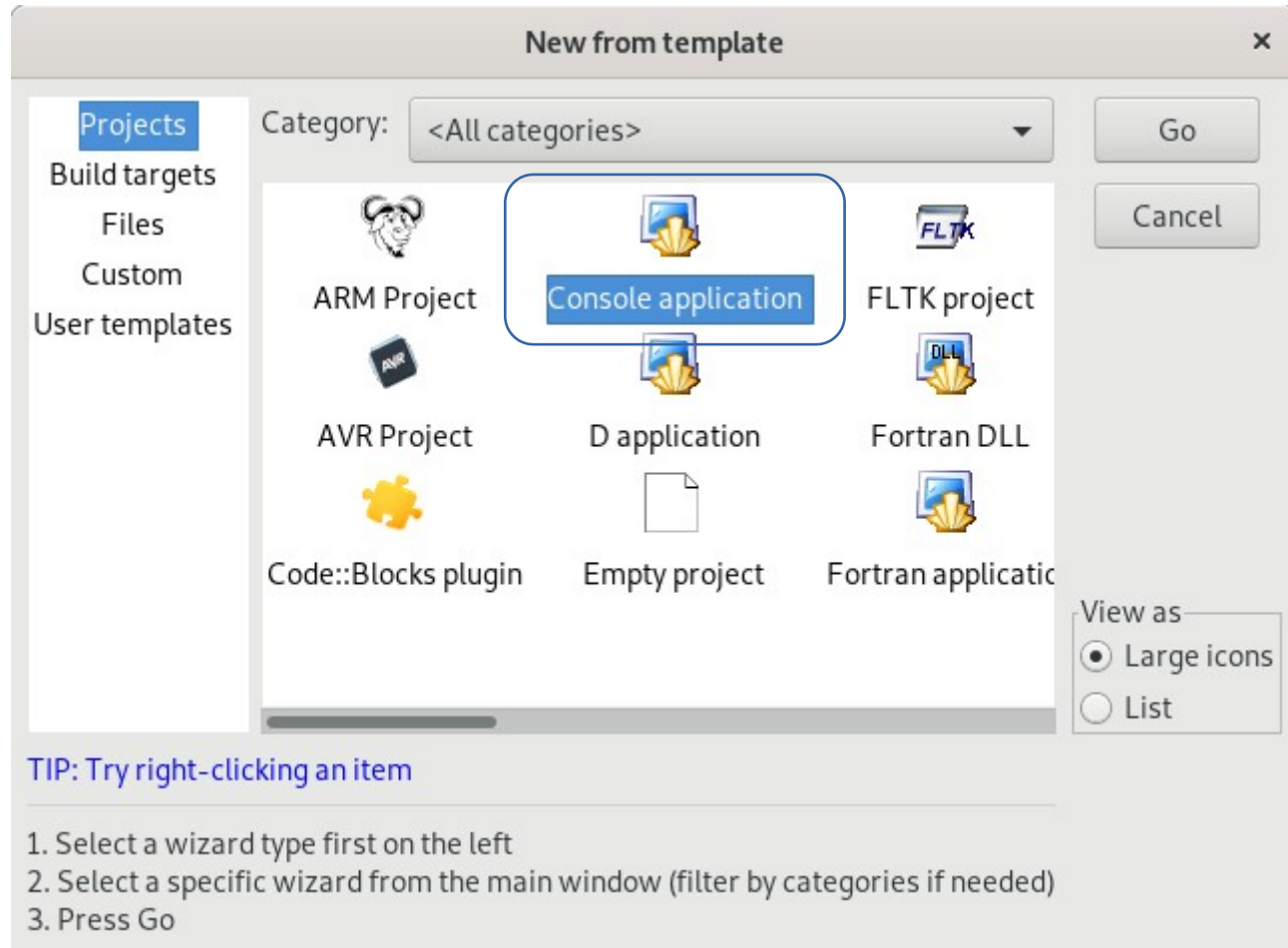
MinGW (Minimalist GNU for Windows)...



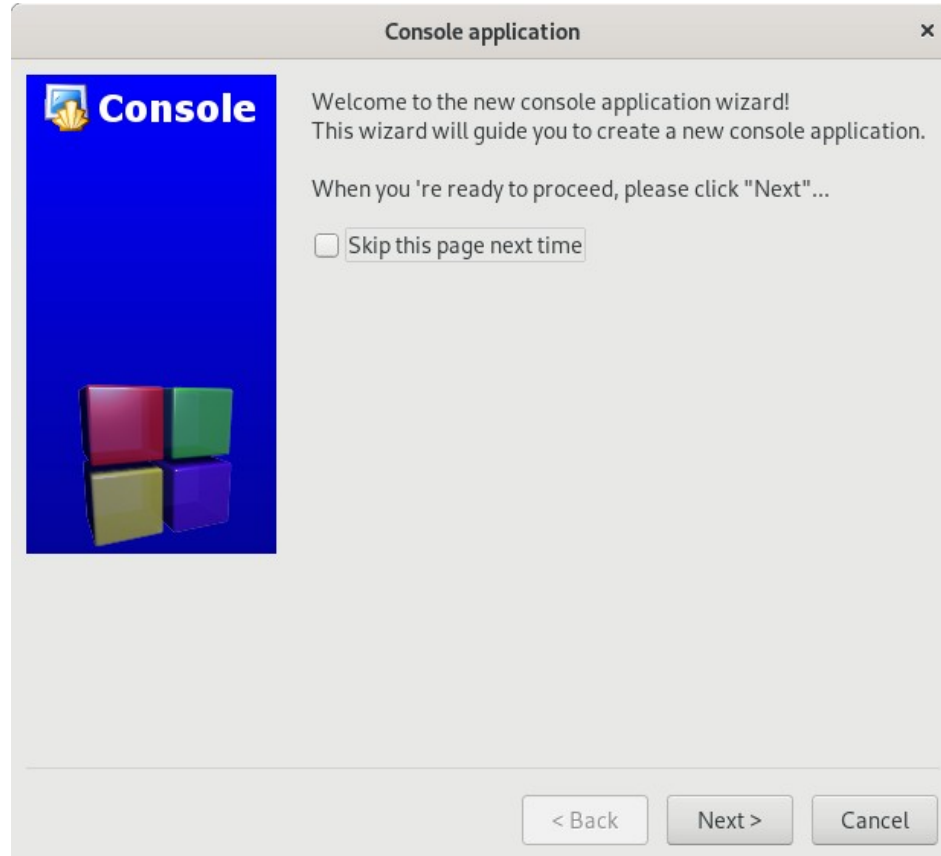
Un exemple (Code::Blocks sous Linux)



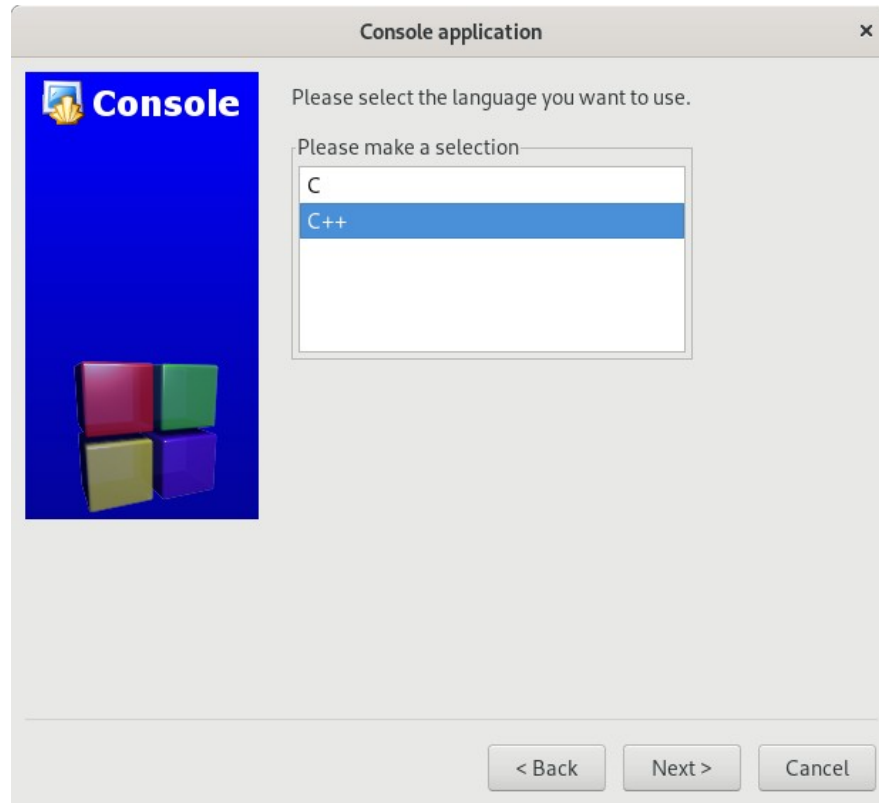
Un exemple (New Project)



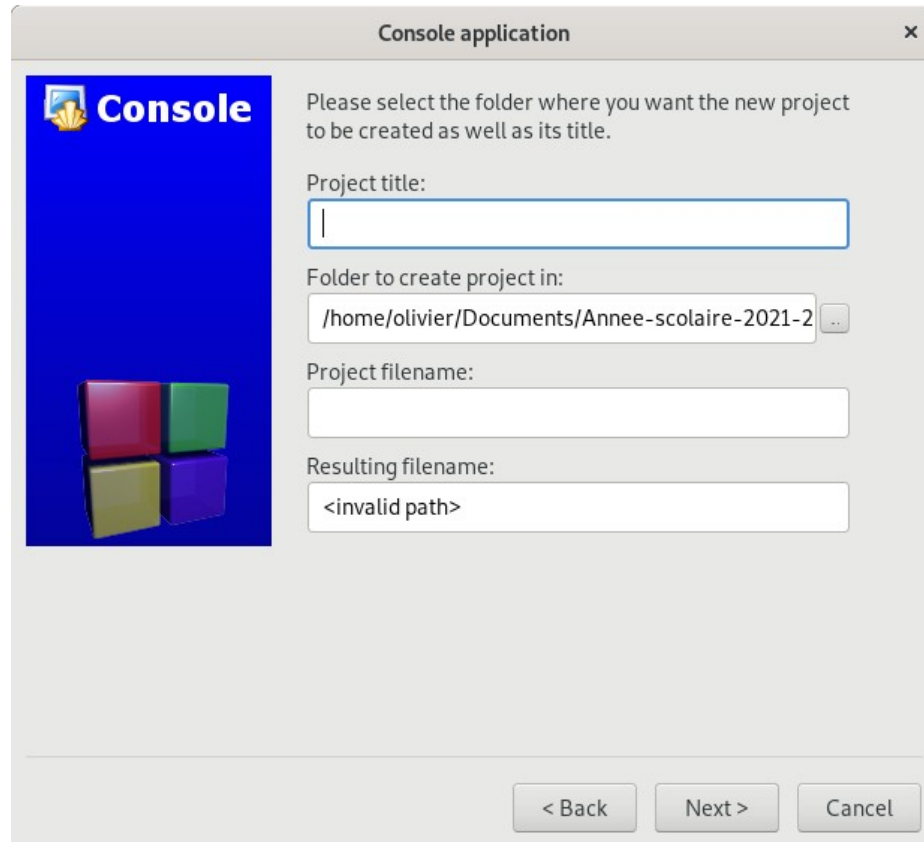
Un exemple (?)



Un exemple (C++)



Un exemple (C++)



The image shows a 'Console application' wizard window. On the left is a blue sidebar with the 'Console' logo and a 3D cube icon. The main area contains instructions and input fields for creating a new project.

Console application [X]

Please select the folder where you want the new project to be created as well as its title.

Project title:

Folder to create project in:
 ..

Project filename:

Resulting filename:

< Back Next > Cancel

Un exemple (C++)

Console application

Please select the folder where you want the new project to be created as well as its title.

Project title:
cours1_exemple1

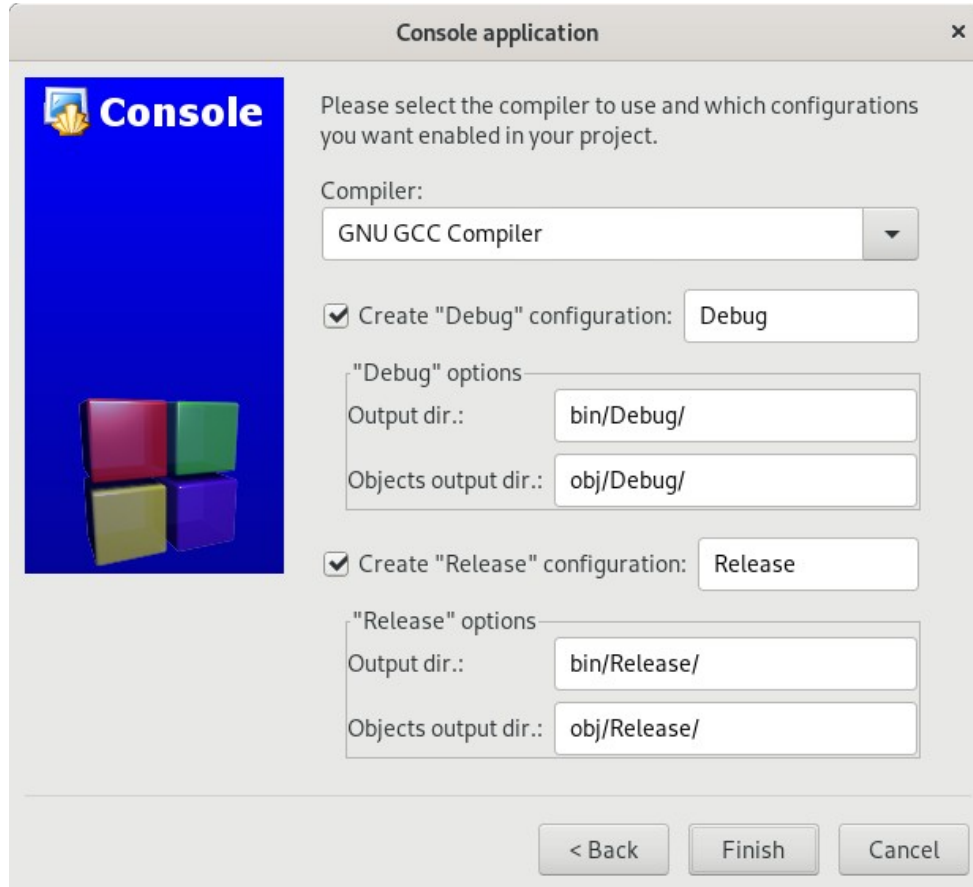
Folder to create project in:
/home/olivier/Documents/Annee-scolaire-2021-2

Project filename:
cours1_exemple1.cbp

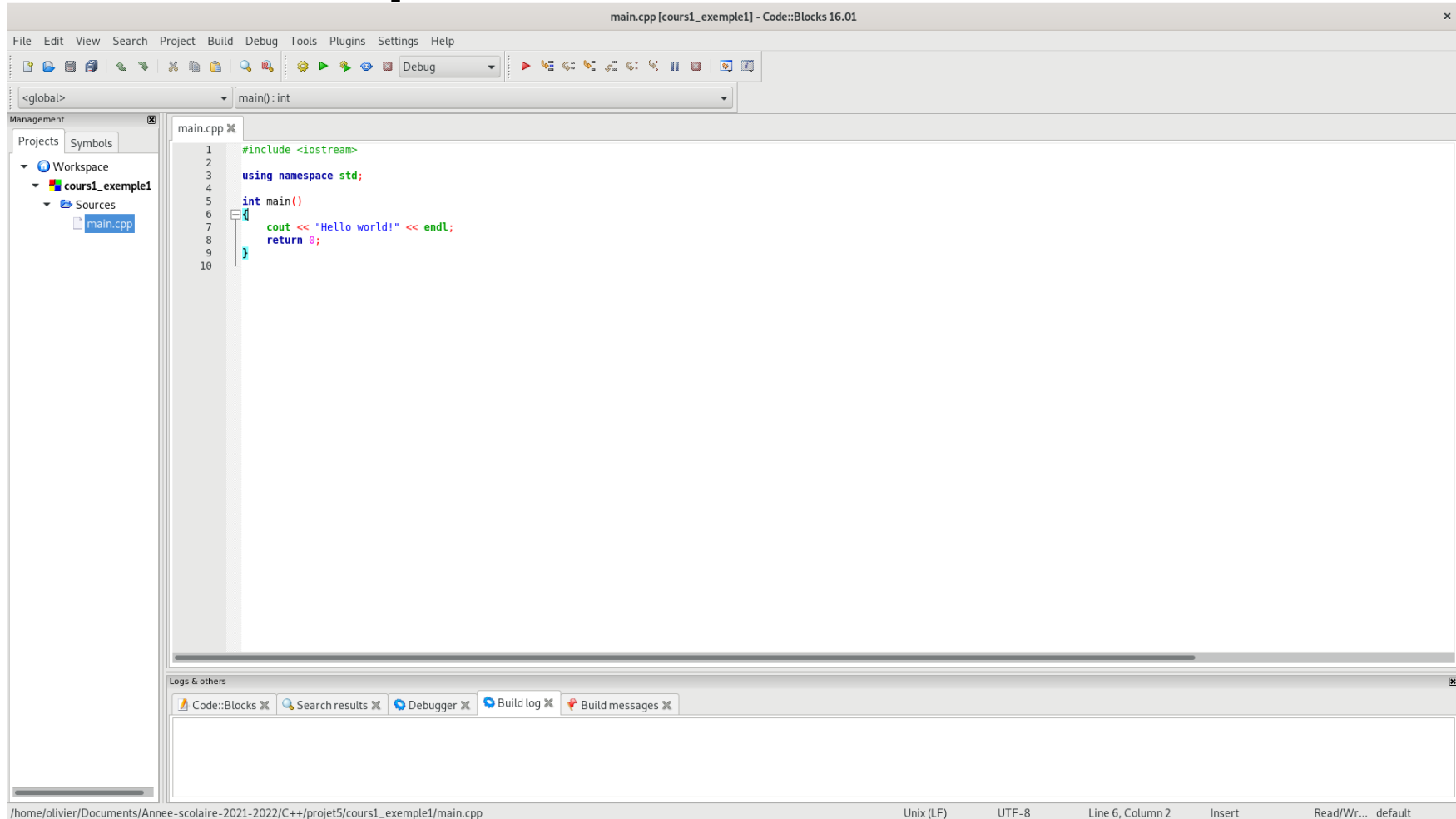
Resulting filename:
/home/olivier/Documents/Annee-scolaire-2021-202

< Back Next > Cancel

Un exemple (C++)



Un exemple (main.cpp dans le répertoire Sources)



Un exemple (main.cpp dans le répertoire Sources)

main.cpp X

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

Build and Run
(F9)

cours1_exemple1 X

```
Hello world!
Process returned 0 (0x0)   execution time : 0.001 s
Press ENTER to continue.
```

Un exemple (main.cpp dans le répertoire Sources)

main.cpp X

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

Directive de pré processeur (travail avant le compilateur)

<https://www.cplusplus.com/reference/iostream/>

Utilisation de l'espace des noms standard (lié à iostream)

<https://www.cplusplus.com/doc/oldtutorial/namespaces/>

Prog. principal

endl : fin de ligne

cout pour afficher à l'écran
(permis car iostream + using namespace std)

Fin du « int main () » car cette fonction main(), du type int doit retourner un entier

Le même exemple (standard C++ 2011)

```
1 #include <iostream>
2
3
4 int main()
5 {
6     std::cout << "Hello world!" << std::endl;
7     return 0;
8 }
9
```

Directive de pré processeur (travail avant le compilateur)
<https://www.cplusplus.com/reference/iostream/>

Prog. principal

```
cours1_exemple1 x
Hello world!
Process returned 0 (0x0)   execution time : 0.001 s
Press ENTER to continue.
```

std ::endl : fin de ligne

std ::cout pour afficher à l'écran
(permis car iostream + using namespace std)

Objet : <https://www.cplusplus.com/reference/iostream/cout/>

Fin du « int main () » car cette fonction main(), du type int doit retourner un entier

La syntaxe du langage

Un langage informatique est composé de :

- Mots-clés :
 - 👉 Constituent le vocabulaire du langage (exemple : main, int, cout,...)
- Syntaxe : structures et règles
 - 👉 La « grammaire » du langage
(= la forme requise des instructions)
- Conventions
 - 👉 Constituent des règles de notations adoptées par tous les programmeurs

C++

Les variables

Les variables

- Une variable est un endroit de la mémoire auquel on a donné un nom de sorte que l'on puisse y faire facilement référence dans le programme
- Créer une variable implique de lui donner :
un nom + une valeur + un type
- La valeur d'une variable peut changer (varier) au cours du temps et de l'exécution du programme.
si elle ne change pas : « const int » par exemple
`const double PI = 3.1415926535897 ;`
`#define PI = 3.1415926535897 ;`
`int nombre ;`
`unsigned int mon_nombre = 1 ;`
- Par contre son type ne change pas !
- Convention : 1^{ère} lettre en minuscule.
- Autre convention : 1^{ère} lettre en minuscule qui est le type : `int iNombre ;`

Les variables

[\[source\]](#)

Type de donnée	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32 768 à 32 767
unsigned short int	Entier court non signé	2	0 à 65 535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32 768 à 32 767 -2 147 483 648 à 2 147 483 647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65 535 0 à 4 294 967 295
long int	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$-3.4 \cdot 10^{-38}$ à $3.4 \cdot 10^{38}$
double	Flottant double	8	$-1.7 \cdot 10^{-308}$ à $1.7 \cdot 10^{308}$
long double	Flottant double long	10	$-3.4 \cdot 10^{-4932}$ à $3.4 \cdot 10^{4932}$
bool	Booléen	Même taille que le type <i>int</i> , parfois 1 sur quelques compilateurs	Prend deux valeurs : <i>'true'</i> et <i>'false'</i> mais une conversion implicite (valant 0 ou 1) est faite par le compilateur lorsque l'on affecte un entier (en réalité toute autre valeur que 0 est considérée comme égale à <i>True</i>).

Les variables

Les conversions de types de données

Conversion implicite :

```
int x ;  
X = 8.32 ;  
X contiendra 8
```

Conversion explicite :

```
int x ;  
X = int(8.32) ;  
X contiendra 8
```

```
int x ;  
X = (int)8.32 ;  
X contiendra 8
```

[*source*]

Les variables

La portée d'une variable

Une variable peut-être accessible de n'importe où (on parle de **variable globale**) ou seulement dans une fonction (on parle de **variable locale**).

Il existe l'opérateur de résolution de portée **::** qui permet d'utiliser la variable globale lorsqu'elle a le même nom que celui de la variable locale.

Exemple :

```
int toto = 13 ; // création et initialisation de la variable globale
```

```
int main()
```

```
{
```

```
    int toto = 1 ; // création et initialisation de la variable locale
```

```
    toto = toto +::toto ; // on ajoute la variable globale à la variable locale...
```

```
}
```

Les variables

La portée d'une variable

```
1  #include <iostream>
2  int toto = 13;
3
4  int main()
5  {
6      int toto = 1;
7      std::cout << "toto = " << toto << std::endl;
8
9      toto = toto + ::toto;
10     std::cout << "toto = " << toto << std::endl;
11
12     return 0;
13 }
14
```

cours1_exemple1

```
toto = 1
toto = 14

Process returned 0 (0x0)   execution time : 0.001 s
Press ENTER to continue.
```

La portée d'une variable :

C++ en ligne

<https://www.programiz.com/cpp-programming/online-compiler/>

```
#include <iostream>
int toto = 13;
```

```
int main() {
    // Write C++ code here
    int toto = 1;
    std::cout << "toto = " << toto << std::endl;
    toto = toto + ::toto;
    std::cout << "toto = " << toto << std::endl;
    return 0;
}
```

toto = 1
toto = 14

La portée d'une variable :

C++ en ligne

```
#include <iostream>
```

```
int toto=1;
```

```
/*interdit de faire ceci:
```

```
std::cout << "Saisissez la 1ère valeur: ";
```

```
std::cin >> toto;
```

```
*/
```

```
int main() {
```

```
    std::cout << "Saisissez la 1ère valeur: ";
```

```
    std::cin >> ::toto;
```

```
    int toto;
```

```
    std::cout << "Saisissez la 2nde valeur: " ;
```

```
    std::cin >> toto;
```

```
    std::cout << "toto = " << toto << std::endl;
```

```
    toto = toto + ::toto;
```

```
    std::cout << "toto = " << toto << std::endl;
```

```
    return 0;
```

```
}
```

Saisissez la 1ère valeur: 12

Saisissez la 2nde valeur: 5

toto = 5

toto = 17

Rappel sur les variables

[\[source\]](#)

Type de donnée	Signification	Taille (en octets)	Plage de valeurs acceptée
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32 768 à 32 767
unsigned short int	Entier court non signé	2	0 à 65 535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32 768 à 32 767 -2 147 483 648 à 2 147 483 647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65 535 0 à 4 294 967 295
long int	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$-3.4 \cdot 10^{-38}$ à $3.4 \cdot 10^{38}$
double	Flottant double	8	$-1.7 \cdot 10^{-308}$ à $1.7 \cdot 10^{308}$
long double	Flottant double long	10	$-3.4 \cdot 10^{-4932}$ à $3.4 \cdot 10^{4932}$
bool	Booléen	Même taille que le type <i>int</i> , parfois 1 sur quelques compilateurs	Prend deux valeurs : <i>'true'</i> et <i>'false'</i> mais une conversion implicite (valant 0 ou 1) est faite par le compilateur lorsque l'on affecte un entier (en réalité toute autre valeur que 0 est considérée comme égale à <i>True</i>).

Les opérateurs et expressions

Opérateurs de calcul

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
+	opérateur d'addition	Ajoute deux valeurs	x+3	10
-	opérateur de soustraction	Soustrait deux valeurs	x-3	4
*	opérateur de multiplication	Multiplie deux valeurs	x*3	21
/	opérateur de division	Divise deux valeurs	x/3	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	x=3	Met la valeur 3 dans la variable x

Exemple : `int x = 7 ;`
 `float div = x/5 ;`
 `int reste = x % 5 ;`

Opérateurs d'assignation

Opérateur	Effet
+=	additionne deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat dans la variable
*=	multiplie deux valeurs et stocke le résultat dans la variable
/=	divise deux valeurs et stocke le résultat dans la variable

Exemple : `x=x+2 ;`
 `x+=2 ;`

Opérateurs d'incrémentement ou de décrémentement

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentement	Augmente d'une unité la variable	x++	8
--	Decrémentement	Diminue d'une unité la variable	x--	6

Les opérateurs et expressions

Opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
== A ne pas confondre avec le signe d'affectation (=) !	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	x==3	Retourne 1 si x est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	x<3	Retourne 1 si x est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=3	Retourne 1 si x est inférieur ou égal à 3, sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	x>3	Retourne 1 si x est supérieur à 3, sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	x>=3	Retourne 1 si x est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	x!=3	Retourne 1 si x est différent de 3, sinon 0

Opérateurs logiques

Opérateur	Dénomination	Effet	Syntaxe
 	OU logique	Vérifie qu'une des conditions est réalisée	((condition1) condition2))
&&	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1)&&condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

Exercice

Écrire un programme qui demande à l'utilisateur son age et qui calcul son année de naissance :

- la variable « *son_age* » est de type *unsigned int*
- la variable de sortie est de type *unsigned int* et est « *son_annee2naissance* »
- l'année en cours se nomme « *année_actuelle* » et est une constante....

Pour programmer :

- `std::cout << "Quel age avez-vous ? "<< std::endl ;`
- `std::cin >> son_age ;`
- `std::cout << "Votre année de naissance est " << son_annee2naissance << std::endl ;`

```

1  /*****
2
3      | | | | | Online C++ Compiler.
4      | | | | | Code, Compile, Run and Debug C++ program online.
5      | | | | | Write your code in this editor and press "Run" button to compile and execute it.
6
7      *****/
8
9  #include <iostream>
10
11
12  using namespace std; //à retirer
13  unsigned int son_age, son_annee2naissance;
14  const unsigned int annee_actuelle = 2022;
15
16  int main()
17  {
18      std::cout << "Programme de calcul de l'année de naissance" << std::endl;
19
20      std::cout << "Quel est votre age? ";
21      std::cin >> son_age;
22      son_annee2naissance = annee_actuelle - son_age;
23      std::cout << "Votre année de naissance est " << son_annee2naissance << std::endl;
24
25
26      return 0;
27  }

```

```

1  /*****
2
3      | | | | | Online C++ Compiler.
4      | | | | | Code, Compile, Run and Debug C++ program online.
5      | | | | | Write your code in this editor and press "Run" button to compile and execute it.
6
7      *****/
8
9  #include <iostream>
10
11
12  using namespace std; //à retirer
13  unsigned int son_age, son_annee2naissance;
14  const unsigned int annee_actuelle = 2022;
15
16  int main()
17  {
18      std::cout << "Programme de calcul de l'année de naissance" << std::endl;
19
20      std::cout << "Quel est votre age? ";
21      std::cin >> son_age;
22      //son_annee2naissance = annee_actuelle - son_age;
23      std::cout << "Votre année de naissance est " << annee_actuelle - son_age << std::endl;
24
25
26      return 0;
27  }

```

2nde solution

https://www.onlinegdb.com/online_c++_compiler/

Bilan

- On a réalisé un exemple avec des variables globales
- 1 variable est une constante et on l'a annoncé au compilateur qu'elle ne changera pas au cours de l'exécution du programme
- `#include <iostream>` permet d'utiliser les entrées et sorties du PC (cin, cout) : c'est une librairie... (ou bibliothèque)
- `using namespace std;` n'est pas à utiliser dans les dernières versions de C++ (mais ça fonctionne...)
- `std::cout <<`
- `std::cin >>`
- `std::endl ;`
- `return 0 ;` C'est la fin du `main()` → `main` = principal

Notre calcul n'est pas exact dans tous les cas : il faut faire des tests sur le mois actuel

Les expressions conditionnelles

Le **if ... else**

```
if (condition réalisée ?) {  
    liste d'instructions  
}  
  
else {  
    autre série d'instructions  
}
```

Exemple 1 :

```
if (etudiant == homme) || (etudiant == femme)  
    {  
        .... ;  
    }  
else  
    {  
        .... ;  
    }
```


Les expressions conditionnelles

Le **if ... else**

```
if (condition réalisée ?) {  
    liste d'instructions  
}  
  
else {  
    autre série d'instructions  
}
```

Exemple 2 :

```
if (etudiant == homme) || (etudiant == femme)  
{  
    .... ;  
}  
else if (etudiant == homme) && (etudiant == femme)  
{  
    .... ;  
}  
else  
{  
    .... ;  
}
```

Les expressions conditionnelles

Le **switch case**

```
switch (Variable) {  
  
    case Valeur1 :  
        Liste d'instructions  
        break;  
  
    case Valeur2 :  
        Liste d'instructions  
        break;  
  
    default:  
        Liste d'instructions  
        break;  
}
```

Exemple :
switch (age) {

```
    case 0 :  
        Liste d'instructions  
        break;  
  
    case 1 :  
        Liste d'instructions  
        break;  
  
    default:  
        Liste d'instructions  
        break;  
}
```

Les boucles

La boucle **for**

```
for (compteur; condition; modification du compteur)
{
    liste d'instructions
}
```

Exemple :

```
for (int i=1; i<6; i++)
{
    printf("%d", i);
}
```

Compteur : initialisation du compteur de boucles

Condition : condition pour rester dans la boucle

Modification du compteur : incrémentation ou décrémentation en général

Les boucles

La boucle **while**

```
while (condition réalisée ?)
{
    liste d'instructions
}
```

Exemple :

```
x=1;

while (x<=10) {
    if (x == 7) {
        printf("Division par zéro !");
        x++ ;
        continue;

    }
    a = 1/(x-7);

    printf("%d", a);

    x++;

}
```

Exercice (suite)

Écrire un programme qui demande à l'utilisateur son age et qui calcul son année de naissance :

- la variable « *son_age* » est de type *unsigned int*
- la variable de sortie est de type *unsigned int* et est « *son_annee2naissance* »
- l'année en cours se nomme « *année_actuelle* » et est une constante....
- le mois en cours se nomme « *mois_actuel* » et est une constante....

Pour programmer :

- `std::cout << "Quel age avez-vous ? "<< std::endl ;`
- `std::cin >> son_age ;`
- `std::cout << "Votre année de naissance est « " << son_annee2naissance << std::endl ;`

Exercice (suite)

Écrire un programme qui demande à l'utilisateur son age et qui calcul son année de naissance :

- la variable « *son_age* » est de type *unsigned int*
- la variable de sortie est de type *unsigned int* et est « *son_annee2naissance* »
- l'année en cours se nomme « *année_actuelle* » et est une constante....
- le mois en cours se nomme « *mois_actuel* » et est une constante....

Pour programmer :

- `std::cout << "Quel age avez-vous ? "<< std::endl ;`
- `std::cin >> son_age ;`
- `std::cout << "Votre année de naissance est « " << son_annee2naissance << std::endl ;`

Les fonctions

La déclaration de la fonction :

```
type_de_donnee Nom_De_La_Fonction(type1 argument1, type2 argument2, ...)  
{  
    liste d'instructions ;  
}
```

L'appel de la fonction (dans le main par exemple) :

```
Nom_De_La_Fonction() ;
```

Les fonctions

Un exemple

```
#include <iostream>
using namespace std;

int doubleur(int a_locale_doubleur)
{
    return 2*a_locale_doubleur;
}

int main()
{
    int nombre = 21;
    cout << doubleur(nombre) << endl; // 42.
    return 0;
}
```


Les fonctions

Un autre exemple

```
#include <iostream>
using namespace std;

// définition de la fonction f :
double f(double x, double y)
{
    double a;
    a = x*x + y*y;
    return a;
}

int main()
{
    double u, v, w;
    cout << "Tapez la valeur de u : "; cin >> u;
    cout << "Tapez la valeur de v : "; cin >> v;

    w = f (u, v); //appel de notre fonction

    cout << "Le résultat est " << w << endl;
    return 0;
}
```

Les fonctions

Encore un autre exemple

```
#include <iostream>
using namespace std;

// prototype de la fonction f :
double f(double x, double y); // attention à ne pas oublier le ;

int main()
{
    double u, v, w;
    cout << "Tapez la valeur de u : "; cin >> u;
    cout << "Tapez la valeur de v : "; cin >> v;

    w = f (u, v); //appel de notre fonction

    cout << "Le résultat est " << w << endl;
    return 0;
}

// définition de la fonction f :
double f(double x, double y)
{
    double a;
    a = x*x + y*y;
    return a;
}
```

Les fonctions (Exercice v1)

Réalisez un programme de résolution d'une équation du 2nd degré avec l'affichage du discriminant, et les 2 résultats possibles

Les variables seront des « float » ou des « double »

Il faudra utiliser la librairie iostream et cmath (pour la racine carrée (sqrt))

Si le discriminant est négatif, on affiche un « pas de solutions dans le réel »

Structure du programme :

Dans la main, on retrouve les questions sur les valeurs de « a », « b » et « c »,

On retrouve l'appel à la fonction

On retrouve le return 0 ;

Structure de la fonction :

On y retrouve tous les calculs, tests et affichage

Les fonctions (Exercice v2)

Réalisez un programme de résolution d'une équation du 2nd degré avec l'affichage du discriminant, et les 2 résultats possibles

Les variables seront des « float » ou des « double »

Il faudra utiliser la librairie iostream et cmath (pour la racine carrée (sqrt))

Si le discriminant est négatif, on affiche un « pas de solutions dans le réel »

Structure du programme :

Dans la main, on retrouve les questions sur les valeurs de « a », « b » et « c »,

On retrouve l'appel à la fonction

On y retrouve l'affichage :-)

On retrouve le return 0 ;

Structure de la fonction :

On y retrouve tous les calculs, tests

Les fonctions récursives

Récuratif : Qui peut être répété un nombre indéfini de fois par l'application de la même règle.

```
1 - /*****  
2  
3      Online C++ Compiler.  
4      Code, Compile, Run and Debug C++ program online.  
5      Write your code in this editor and press "Run" button to compile and execute it.  
6  
7      *****/  
8  
9  #include <iostream>  
10  
11  int Fact(int n)  
12  {  
13  
14      if(n == 1)  
15          return 1;  
16      else  
17          return (n*Fact(n-1));  
18  }  
19  
20  int main()  
21  {  
22      std::cout<< Fact(5);  
23      return 0;  
24  }  
25
```

```
#include <iostream>
```

```
int Fact(int n)  
{
```

```
    if(n == 1)  
        return 1;  
    else  
        return (n*Fact(n-1));  
}
```

```
int main()  
{  
    std::cout<< Fact(5);  
    return 0;  
}
```

TD (à la maison)

Réalisez un programme qui calcul la moyenne des notes d'un étudiant. Les notes ne peuvent pas être négative (si elles sont mal saisies, on affiche un message d'erreur) et les notes ne peuvent pas être au dessus de 20 (si elles sont mal saisies, on affiche un message d'erreur).

Le programme boucle en continu et affiche donc la moyenne à chaque saisie de l'utilisateur.

Pour arrêter le programme, l'utilisateur devra appuyer sur la touche P

- le main ne comporte que très peu de lignes
- fonctions obligatoires
- utilisez des caractères du type * _ pour faire des affichages un peu « fun »

Enjoy