Introduction à l'informatique
Travaux pratiques: séance introductive
INFO0205-1

X. Baumans
(xavier.baumans@ulg.ac.be)

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques
- Présence fortement conseillée => a priori positif

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques
- Présence fortement conseillée => a priori positif
- Vendredi 8h30-10h30 [12/02 ; 19/02 ; 26/02 ; 04/03 ; 11/03 ; 18/03 ; 25/03 ; 15/04 ; 22/04]

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques
- Présence fortement conseillée => a priori positif
- Vendredi 8h30-10h30 [12/02 ; 19/02 ; 26/02 ; 04/03 ; 11/03 ; 18/03 ; 25/03 ; 15/04 ; 22/04]
- Salle 4.15 (B5a)

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques
- Présence fortement conseillée => a priori positif
- Vendredi 8h30-10h30 [12/02 ; 19/02 ; 26/02 ; 04/03 ; 11/03 ; 18/03 ; 25/03 ; 15/04 ; 22/04]
- Salle 4.15 (B5a)

Examen

- Durée : 4h, date à définir

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques
- Présence fortement conseillée => a priori positif
- Vendredi 8h30-10h30 [12/02 ; 19/02 ; 26/02 ; 04/03 ; 11/03 ; 18/03 ; 25/03 ; 15/04 ; 22/04]
- Salle 4.15 (B5a)

Examen

- Durée : 4h, date à définir
- Examen TP = 70% de la cote totale

Organisation

En pratique...

- 9 séances x 2h = 18h de travaux pratiques
- Présence fortement conseillée => a priori positif
- Vendredi 8h30-10h30 [12/02 ; 19/02 ; 26/02 ; 04/03 ; 11/03 ; 18/03 ; 25/03 ; 15/04 ; 22/04]
- Salle 4.15 (B5a)

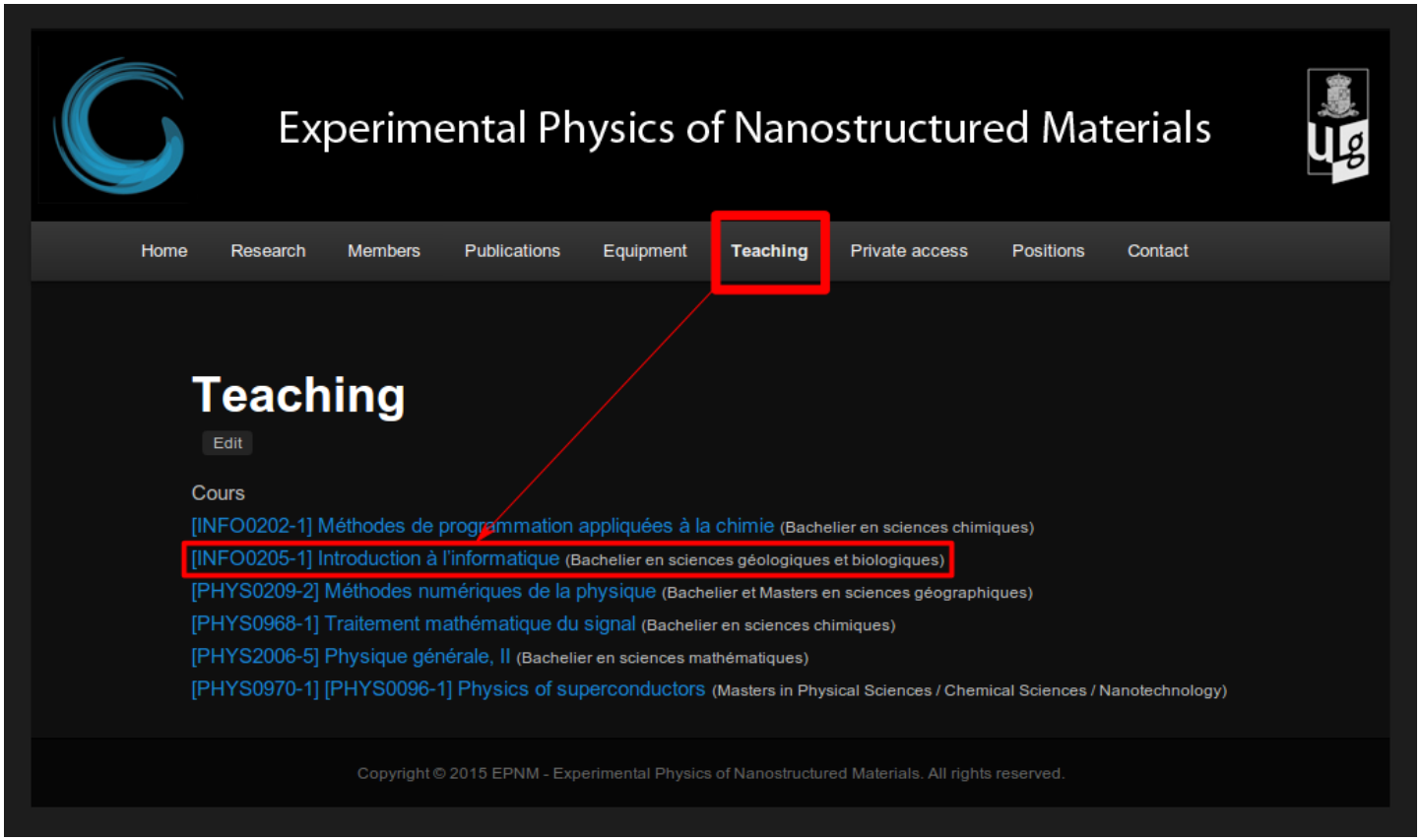
Examen

- Durée : 4h, date à définir
- Examen TP = 70% de la cote totale
- Avoir sa carte d'étudiant avec soi !

Notes de TP

Slides de TP accessibles à l'adresse :

http://www.mate.ulg.ac.be



Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier
  - Déclaration de variables, opérateurs

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier
  - Déclaration de variables, opérateurs
  - Structures de contrôle (if, for, while,...)

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier
  - Déclaration de variables, opérateurs
  - Structures de contrôle (if, for, while,...)
  - Tableaux

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier
  - Déclaration de variables, opérateurs
  - Structures de contrôle (if, for, while,...)
  - Tableaux
  - Fonctions

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier
  - Déclaration de variables, opérateurs
  - Structures de contrôle (if, for, while,...)
  - Tableaux
  - Fonctions
  - Lecture/écriture dans un fichier

Programme

- Introduction à la programmation (C++)
  - Syntaxe (règles d'écriture)
  - Affichage console, lecture au clavier
  - Déclaration de variables, opérateurs
  - Structures de contrôle (if, for, while,...)
  - Tableaux
  - Fonctions
  - Lecture/écriture dans un fichier
  - ...

C'est quoi "programmer" ?

Programme informatique

Suite d'instructions utilisées par l'ordinateur pour effectuer un traitement donné.

C'est quoi "programmer" ?

Programme informatique

Suite d'instructions utilisées par l'ordinateur pour effectuer un traitement donné.

La machine suit le schéma qu'on lui donne. Elle exécute les étapes imposées, une par une, ni plus ni moins.

C'est quoi "programmer" ?

Programme informatique

Suite d'instructions utilisées par l'ordinateur pour effectuer un traitement donné.

La machine suit le schéma qu'on lui donne. Elle exécute les étapes imposées, une par une, ni plus ni moins.

Exemple : Recette culinaire exécutée par un débutant...

C'est quoi "programmer" ?

Programme informatique

Suite d'instructions utilisées par l'ordinateur pour effectuer un traitement donné.

La machine suit le schéma qu'on lui donne. Elle exécute les étapes imposées, une par une, ni plus ni moins.

Exemple : Recette culinaire exécutée par un débutant...

L'ordinateur n'a pas d'esprit d'initiative : il fait **exactement** ce qu'on lui dit et **rien d'autre** ! Néanmoins, point positif : il peut le faire un très grand nombre de fois sans sourciller, et ce très rapidement ! (cf. fréquence du processeur)

C'est quoi "programmer" ?

Programme informatique

Suite d'instructions utilisées par l'ordinateur pour effectuer un traitement donné.

La machine suit le schéma qu'on lui donne. Elle exécute les étapes imposées, une par une, ni plus ni moins.

Exemple : Recette culinaire exécutée par un débutant...

L'ordinateur n'a pas d'esprit d'initiative : il fait **exactement** ce qu'on lui dit et **rien d'autre** ! Néanmoins, point positif : il peut le faire un très grand nombre de fois sans sourciller, et ce très rapidement ! (cf. fréquence du processeur)

! ! ! → Soyez rigoureux avec vos instructions ! ! !

Implication des TP info et pourquoi programmer ?

- La programmation est importante en géologie (modélisation, simulation,... Ex : tectonique des plaques)

Implication des TP info et pourquoi programmer ?

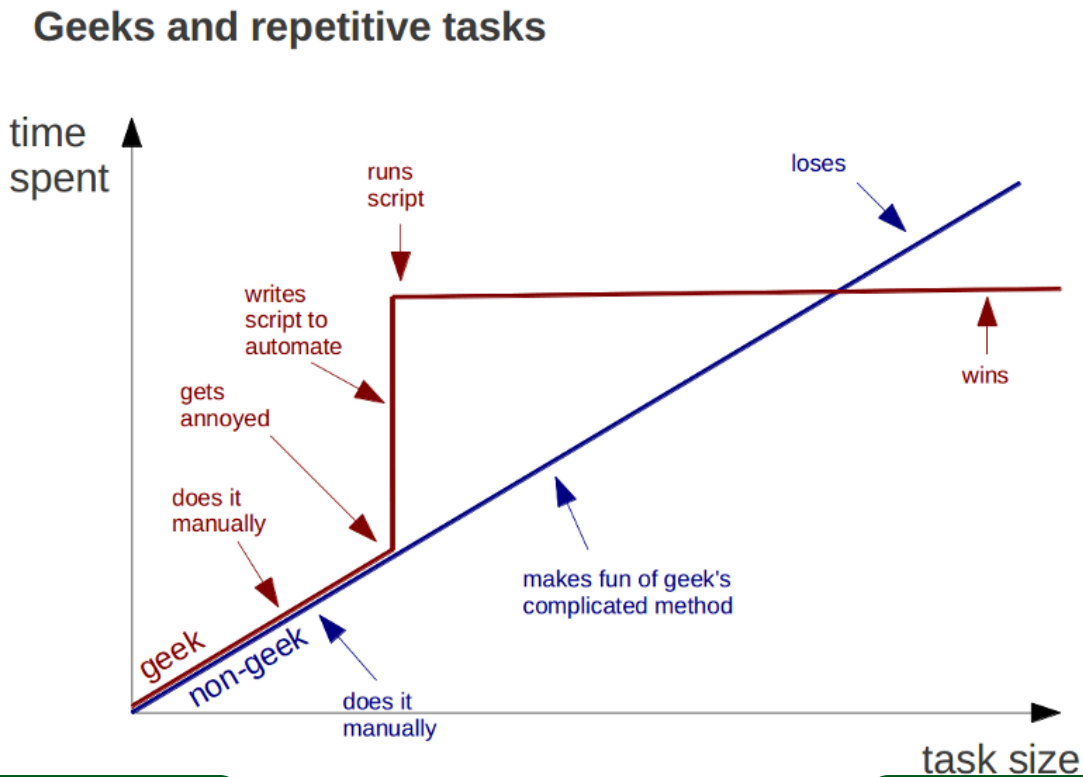
- La programmation est importante en géologie (modélisation, simulation,... Ex : tectonique des plaques)
- Beaucoup de logiciels édités dans le domaine (géostatistique, modélisation stratigraphique, modeleur aux éléments finis,...)

Implication des TP info et pourquoi programmer ?

- La programmation est importante en géologie (modélisation, simulation,... Ex : tectonique des plaques)
- Beaucoup de logiciels édités dans le domaine (géostatistique, modélisation stratigraphique, modeleur aux éléments finis,...)
- Partie pratique importante dans la cote finale du cours (70%)

Implication des TP info et pourquoi programmer ?

- La programmation est importante en géologie (modélisation, simulation,... Ex : tectonique des plaques)
- Beaucoup de logiciels édités dans le domaine (géostatistique, modélisation stratigraphique, modeleur aux éléments finis,...)
- Partie pratique importante dans la cote finale du cours (70%)



Comment programmer ?

Résoudre un problème

1. Analyser et décortiquer le problème

Comment programmer ?

Résoudre un problème

1. Analyser et décortiquer le problème
2. Penser son code (raisonnement logique et intuitif)

Comment programmer ?

Résoudre un problème

1. Analyser et décortiquer le problème
2. Penser son code (raisonnement logique et intuitif)
3. Implémenter (contrôler et commenter chaque étape)

Comment programmer ?

Résoudre un problème

1. Analyser et décortiquer le problème
2. Penser son code (raisonnement logique et intuitif)
3. Implémenter (contrôler et commenter chaque étape)
4. Compiler / Exécuter

Comment programmer ?

Résoudre un problème

1. Analyser et décortiquer le problème
2. Penser son code (raisonnement logique et intuitif)
3. Implémenter (contrôler et commenter chaque étape)
4. Compiler / Exécuter
5. Débugger si nécessaire (retour au point 2)

Langages de programmation

Types de langage

- Langage structuré (C,Fortran,...)

Langages de programmation

Types de langage

- Langage structuré (C,Fortran,...)
- Langage orienté objet (C++, C#, Objective-C, Java,...)

Langages de programmation

Types de langage

- Langage structuré (C,Fortran,...)
- Langage orienté objet (C++, C#, Objective-C, Java,...)

Programmation C++ pour ce cours

- C étendu (langage structuré)
- Notion de classe et objet

Construction d'un programme

Un ordinateur n'est capable d'exécuter qu'un nombre limité d'opérations différentes
→ ce sont les **instructions fondamentales** du processeur
⟶ les instructions plus complexes sont réalisées en les combinant

```
1  mov     -0x18(%rax),%rax
2  mov     0x603170(%rax),%rbx
3  test    %rbx,%rbx
4  je      401c63 <main+0x173>
5  mov     $0x603080,%edi
6  mov     $0xfffe7960,%ebx
7  callq   400d08 <_ZNSo3putEc@plt>
8  mov     %rax,%rdi
```

Ce **code machine** est très difficile à comprendre pour un être humain
⟶ On utilise un **langage de programmation** plus simple à comprendre et à utiliser

Langages de programmation : C/C++

Un langage de programmation est constitué

- d'un ensemble de **mots-clés** qui correspondent à des instructions (`if`, `else`, `while`, `...`)

Langages de programmation : C/C++

Un langage de programmation est constitué

- d'un ensemble de **mots-clés** qui correspondent à des instructions (`if`, `else`, `while`, `...`)
- de règles pour combiner les éléments du langage : la **syntaxe**

Langages de programmation : C/C++

Un langage de programmation est constitué

- d'un ensemble de **mots-clés** qui correspondent à des instructions (`if`, `else`, `while`, `...`)
- de règles pour combiner les éléments du langage : la **syntaxe**
- d'**identifiants** pour les variables, les fonctions, etc...

Langages de programmation : C/C++

Un langage de programmation est constitué

- d'un ensemble de **mots-clés** qui correspondent à des instructions (`if`, `else`, `while`, `...`)
- de règles pour combiner les éléments du langage : la **syntaxe**
- d'**identifiants** pour les variables, les fonctions, etc...

Nous utiliserons le langage **C/C++**, qui possède une syntaxe et des mots-clés qui lui sont propres

```
1 double trunc_error = fabs(f/h - (fplus +
      fminus)/(2.*h));
2 double roundoff_error =
      std::numeric_limits<double>::epsilon()*f/h;
3 u = trunc_error / roundoff_error;
```

Étapes fondamentales de la construction d'un programme

2 étapes principales :

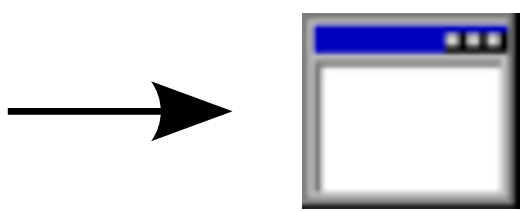
1. **Écrire** le "code source" du programme dans un fichier texte
   → Un ou des fichier(s) texte qui contiennent le programme écrit dans un langage de programmation

Étapes fondamentales de la construction d'un programme

2 étapes principales :

1. **Écrire** le "code source" du programme dans un fichier texte
   → Un ou des fichier(s) texte qui contiennent le programme écrit dans un langage de programmation
2. "**Compiler**" le code source pour en faire un programme exécutable
   → Le compilateur transforme les fichiers sources en un code exécutable par l'ordinateur

```
56
57      double u = 1;
58      double derivative = 0;
59
60      int cptr = 0;
61      while( ((u < 50) || (u > 200)) && (cptr < 100))
62      {
63        double fplus = (*derivfunction)(x+h);
64        double fplus2 = (*derivfunction)(x+2.*h);
65
66        double fminus = (*derivfunction)(x-h);
67        double fminus2 = (*derivfunction)(x-2.*h);
68
69        derivative = (fplus - fminus)*0.5/h;
70
71        // Evaluation of the error on the derivative from Curtis and Reid, J. Math
72        if( f == fminus == fplus )
73        {
74          std::cout << "Function seems to be flat ==> derivative = 0" << endl;
75          u = 0;
76        }
77        else
78        {
79          double trunc_error = fabs(f/h - (fplus + fminus)/(2.*h));
80          double roundoff_error = std::numeric_limits<double>::epsilon()*f/h;
81
```



Écrire le code source → Code : :Blocks

Code : :Blocks = **E**nvironnement de **D**éveloppement **I**ntégré

Il regroupe :

Écrire le code source → Code : :Blocks

Code : :Blocks = **E**nvironnement de **D**éveloppement **I**ntégré

Il regroupe :

- Un **éditeur de texte** : il facilite l'écriture du code source (coloration syntaxique, numérotation des lignes, indentation automatique, ...)

Écrire le code source → Code : :Blocks

Code : :Blocks = **E**nvironnement de **D**éveloppement **I**ntégré

Il regroupe :

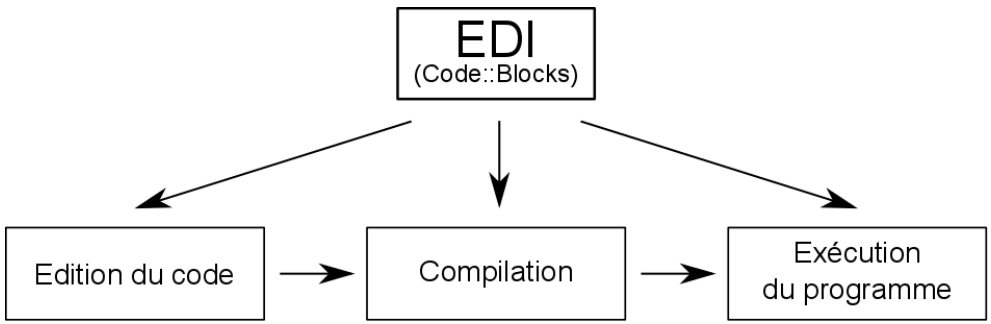
- Un **éditeur de texte** : il facilite l'écriture du code source (coloration syntaxique, numérotation des lignes, indentation automatique, ...)
- Une interface avec un **compilateur** : il permet de compiler directement les codes sources écrits dans l'éditeur. Une fenêtre montre les éventuelles erreurs de compilation. On peut ensuite exécuter le programme.

Écrire le code source → Code : :Blocks

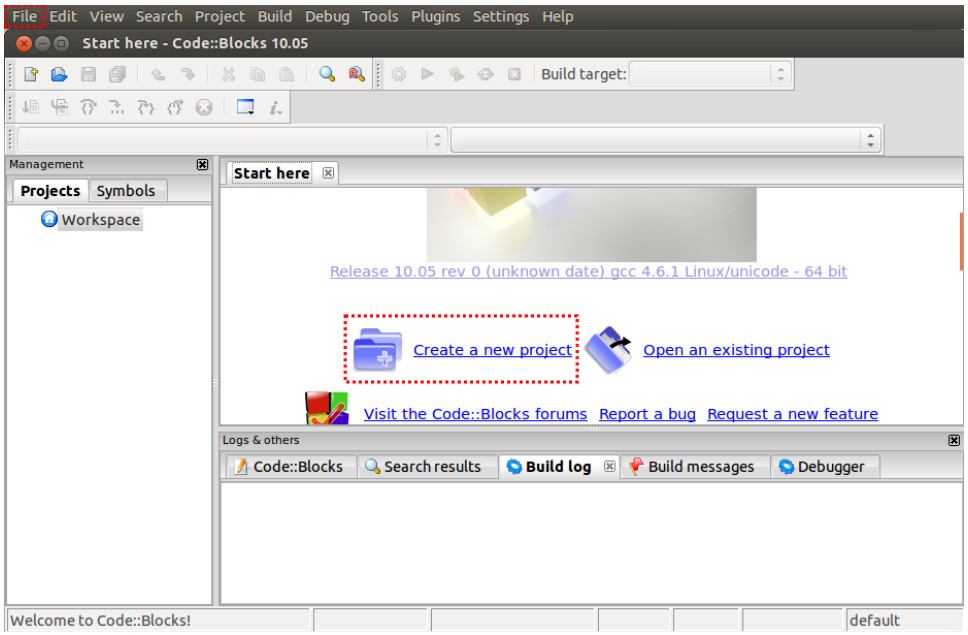
Code : :Blocks = **E**nvironnement de **D**éveloppement **I**ntégré

Il regroupe :

- Un **éditeur de texte** : il facilite l'écriture du code source (coloration syntaxique, numérotation des lignes, indentation automatique, ...)
- Une interface avec un **compilateur** : il permet de compiler directement les codes sources écrits dans l'éditeur. Une fenêtre montre les éventuelles erreurs de compilation. On peut ensuite exécuter le programme.

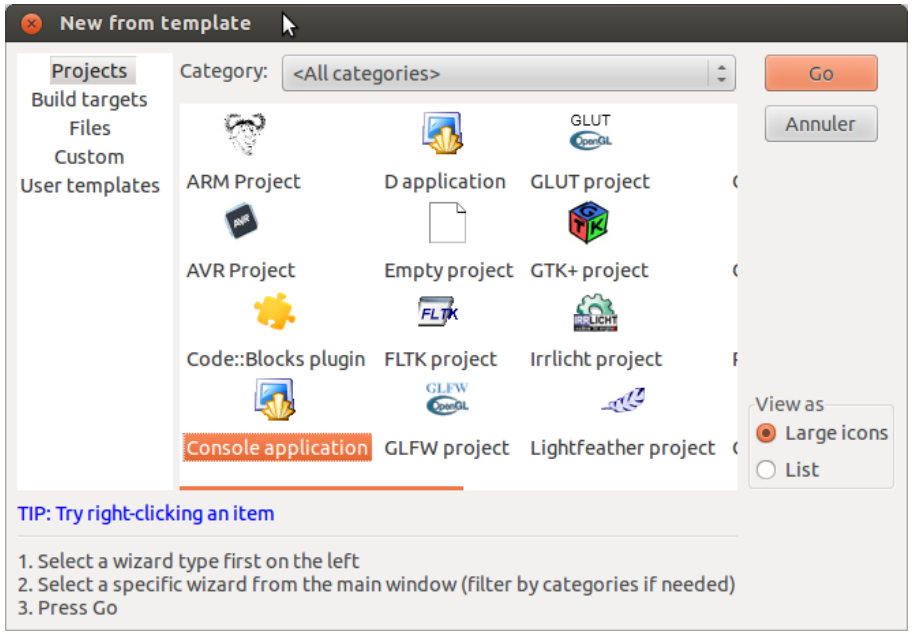


Créer un projet Code : :Blocks (1/4)



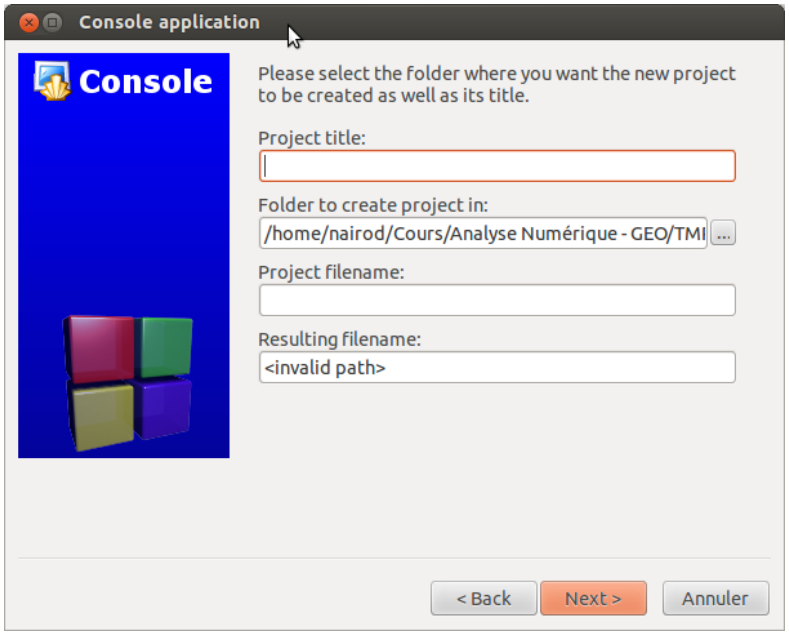
Cliquer sur "`Create a new project`"

Créer un projet Code : :Blocks (2/4)



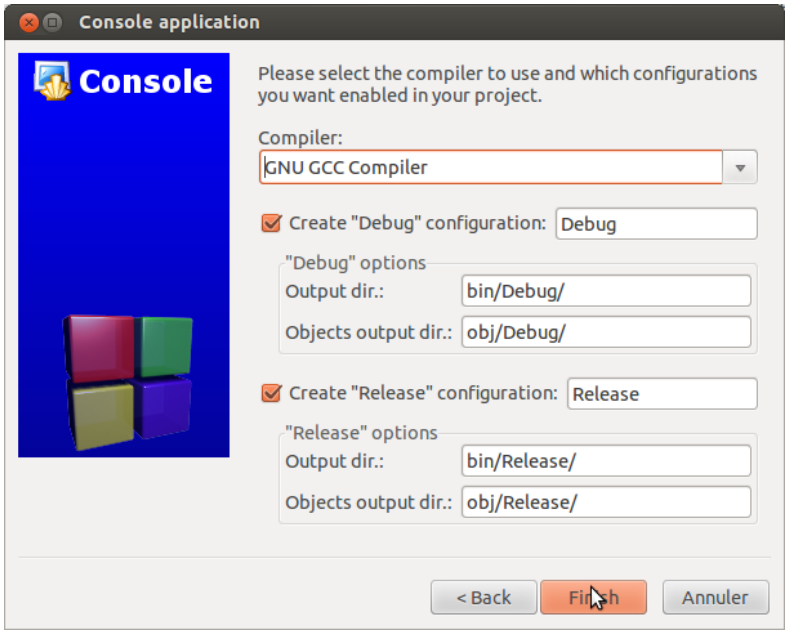
Dans la catégorie "`Projects`", choisir le type "`Console application`", pour construire un projet permettant de réaliser des affichages et des saisies au clavier dans un terminal.

Créer un projet Code : :Blocks (3/4)



Choisir un nom de projet pour le champ "`Project title`" et choisir le dossier dans lequel enregistrer le projet avec le champ "`Folder to create project in`".
Laisser les valeurs par défaut pour les deux autres champs.

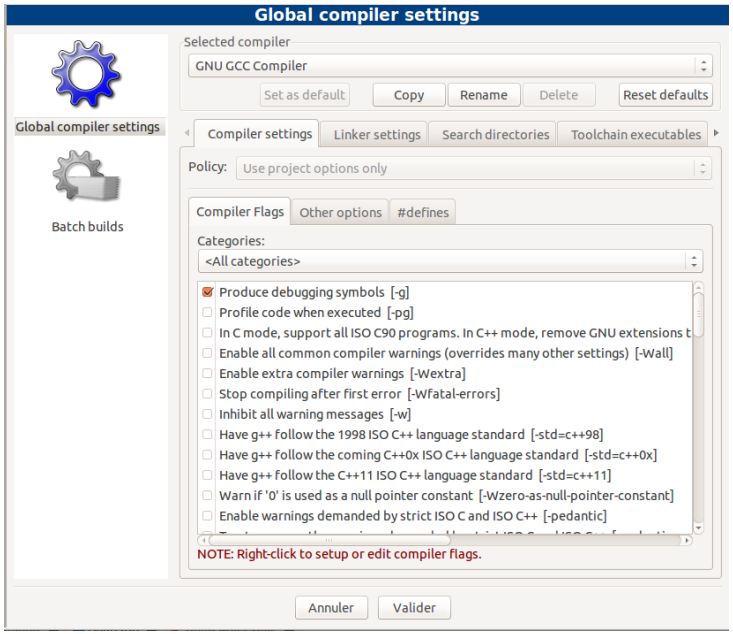
Créer un projet Code : :Blocks (4/4)



Ne rien modifier dans cette fenêtre et cliquer sur "`Finish`"

Paramètres par défaut du compilateur

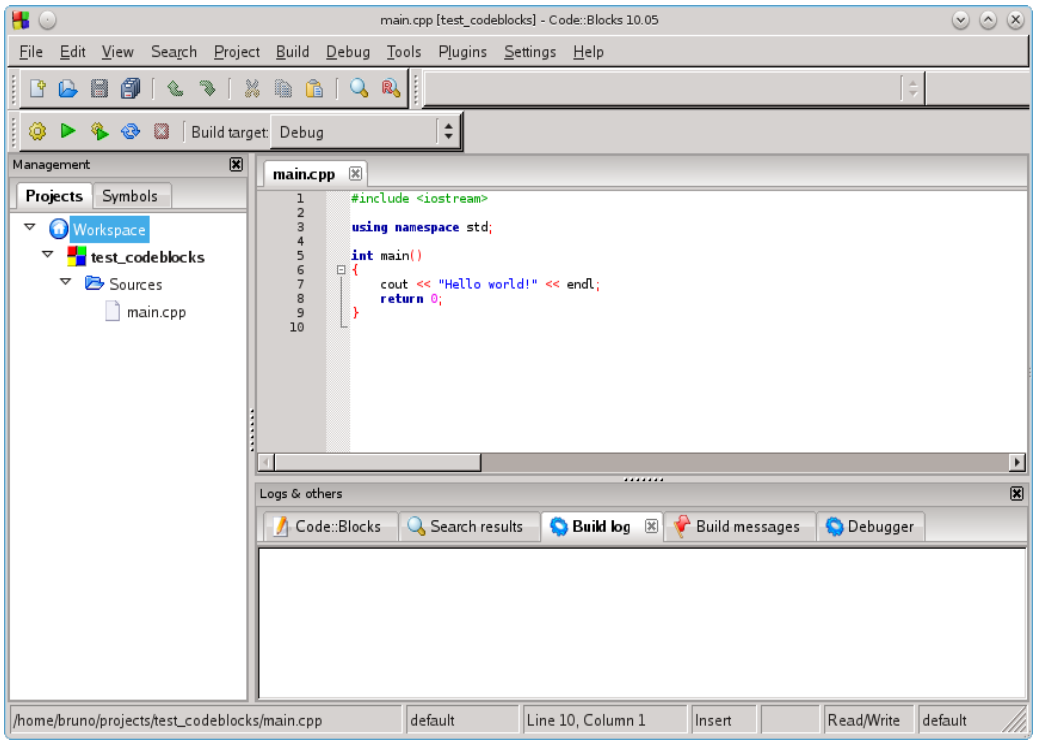
Settings → *Compiler* → Panneau *"Compiler settings"* → Onglet *"Compiler Flags"*...



Paramètres à cocher si vous utilisez votre ordinateur personnel :

- -Wall
- -pedantic
- -pedantic-errors
- -Wfloat-equal
- -Wshadow

Premier programme : Hello World !



Lors de la création d'un nouveau projet, Code : :Blocks le complète automatiquement avec un petit programme : "**Hello World !**"
Tous les programmeurs commencent leur apprentissage de la programmation en écrivant ce petit programme.

Hello World !

Le code source du programme “Hello World” en C/C++ est le suivant :

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

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

La partie "instructions" du programme

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

Le programme démarre toujours en exécutant une **fonction principale** appelée "**main**"

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

Les instructions de cette fonction sont contenues entre des **accolades {...}**

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

L'objet "**cout**" permet l'affichage à l'écran.

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

L'instruction "**return x ;**" est la dernière instruction de la fonction main. Elle indique quelle valeur la fonction va retourner (0).

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

L'instruction pre-processeur "**#include**" permet d'inclure des librairies externes contenant des fonctionalités supplémentaires. Ici "**iostream**" permet de réaliser des entrées-sorties avec la console.

Hello World !

Le code source du programme "Hello World" en C/C++ est le suivant :

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

"**using namespace**" permet de signaler au compilateur qu'on utilisera un espace de nom appelé "**std**" pour accéder aux fonctions de "**iostream**". Sans cela, on devrait écrire "**std : :cout**".

Instructions en C/C++

- Les instructions sont séparées par des **points-virgules** " ;"
- Les accolades {...} définissent des blocs d'instructions
  Ex. les instructions de la fonction **main()** sont entourées par des accolades
- Le nombre d'espaces entre les mots-clés, identifiants, etc... n'a pas d'importance

```
1  cout << "Hello world!";
2
```

```
1  cout    <<      "Hello world!";
2
```

Fonction main

La fonction **main()** est la fonction principale du programme.

```
1 int main()
2 {
3   // instructions du programme
4   return 0;
5 }
```

Elle doit toujours exister et elle est la première à être exécutée.

Lorsqu'elle se termine, elle donne comme résultat un nombre entier (**int**) qui permet de savoir si son exécution s'est déroulée sans problème (valeur 0) ou si des erreurs se sont produites (valeur positive).

Affichage dans la console : cout (1/2)

"**cout**" permet d'afficher des informations sur l'écran.
Les informations à afficher sont juxtaposées l'une derrière l'autre et séparées par les caractères <<.

```
1 cout << "Du texte" << " qui s'affiche" << endl;
```

L'identifiant "**endl**", lorsqu'il est passé à l'objet **cout**, provoque un retour à la ligne.

```
1 cout << "La ligne 1" << " qui s'affiche" << endl;
2 cout << "La ligne 2" << " s'affiche aussi" << endl;
```

Le caractère spécial "**\t**" permet d'insérer une tabulation dans l'affichage.

```
1 cout << "Texte 1\tTexte 2\tTexte 3" << endl;
```

Affichage dans la console : cout (2/2)

On peut également afficher des nombres

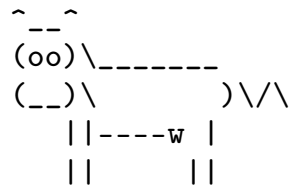
```
1 cout << "Le nombre " << 361 << " s'affiche" << endl;
2 cout << "Pi: " << 3.14159265358979323846 << endl;
```

L'instruction **cout.precision(x)** permet de définir le nombre de chiffres significatifs à afficher

```
1 cout << "Le nombre " << 361 << " s'affiche" << endl;
2 // affichage: Le nombre 361 s'affiche
3 cout << "Pi: " << 3.14159265358979323846 << endl;
4 // affichage: Pi: 3.14159
5 cout.precision(10);
6 cout << "Pi: " << 3.14159265358979323846 << endl;
7 // affichage: Pi: 3.141592653
```

Exercices

- Construire un projet "Hello World !", le compiler et l'exécuter ;
- Construire un programme affichant les premières décimales du nombre d'or $\Phi = 1.61803398874989484820458 6...$, avec successivement
  - la précision par défaut de **cout** ;
  - 15 chiffres significatifs ;
  - 10 chiffres significatifs.
- Construire un programme affichant à l'écran un dessin de votre choix à l'aide de caractères tels que *, +, _, -, etc...



- *Supplément : Déclarer la variable entière* ***a*** *en utilisant l'instruction "int a ;". Ensuite, à l'aide de "cin » a ;", demander à l'utilisateur de rentrer la précision (a) qu'il désire pour le nombre d'or et utiliser cette valeur pour l'affichage de celui-ci.*

Note : **cin** a la même structure que **cout** sauf pour le sens des guillemets. Au lieu d'afficher dans la console ce qui lui est renseigné , il permet à l'utilisateur de rentrer une valeur au clavier dans la console... La valeur est alors assignée à une variable préalablement déclarée.