

Introduction à la programmation

Travaux pratiques: séance d'introduction

INFO0201-1

G. Allemand, A. Wafflard & R. Chrétien, G. Vanhaele & B.
Baert, X. Baumans
Guillaume.Allemand@uliege.be - Adrien.Wafflard@uliege.be

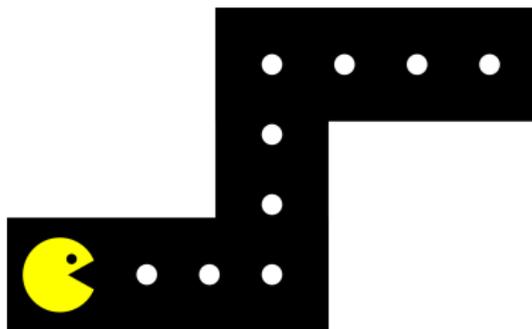


Qu'est-ce que la programmation ?

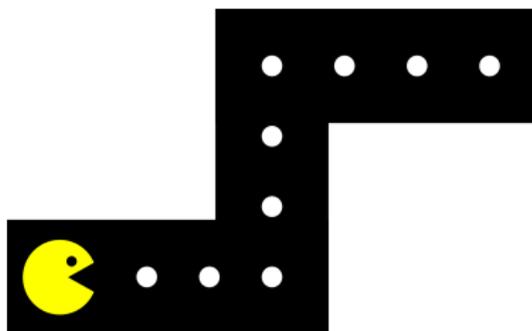
Programmer

Ecrire un programme informatique, c'est établir les étapes précises qu'un ordinateur devra exécuter pour faire quelque chose et les écrire au moyen d'un langage de programmation.

L'ordinateur exécute les étapes qu'on lui donne, une par une, et il ne peut rien faire d'autre que ce qu'on lui dit.



Qu'est-ce que la programmation ?



Exemple: diriger PacMan dans un labyrinthe

- 1 Avancer de trois points
- 2 Tourner à gauche de 90 degrés
- 3 Avancer de trois points
- 4 Tourner à droite de 90 degrés
- 5 Avancer de trois points

Qu'est-ce que la programmation ?

L'ordinateur n'est pas "intelligent", il est **rapide** !

Il ne comprend pas ce qu'il fait ni pourquoi il le fait: il fait ce qu'on lui dit et rien d'autre.

→ **Extrême rigueur** dans les instructions qu'on lui donne !

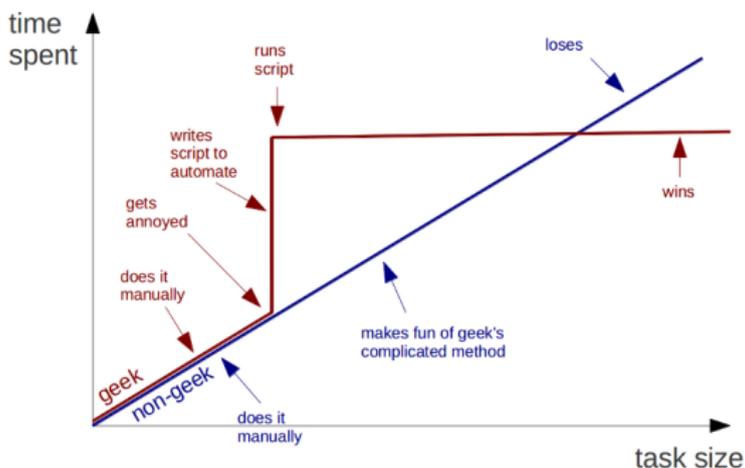
- Si on oublie une étape, l'ordinateur ne l'ajoutera jamais lui-même;
- Si une étape contient une erreur, si petite soit-elle, elle sera commise par l'ordinateur.

Même une simple faute de frappe peut conduire à une erreur dans le programme.

Pourquoi programmer ?

- La programmation est importante en physique (analyse expérience, simulations, traitement de données,...)
- C++ nécessaire pour le cours d'analyse numérique (*Méthodes numériques de la physique*, 2ème BAC)
- Part importante de la cote finale (70%) et interrogations

Geeks and repetitive tasks



Objectif des TPs du cours de programmation

- Acquérir les concepts de base de la programmation informatique
- Apprendre la syntaxe d'un langage de programmation (C/C++)
- Développer des notions de logique et d'algorithmique

En pratique:

- Notions progressives et **cumulatives** lors des TPs, accompagnées d'exercices
- **Projet** - Modalités à venir
- Une simulation d'examen en fin de période

Comment réussir le cours ?

- Observation : ceux qui réussissent le cours ont, en général, réalisé de bons scores durant l'année.
- L'apprentissage de la programmation requiert beaucoup de recul.
 - Nécessité de travailler (pas forcément longtemps mais) très souvent.
 - Ne pas se décourager, le déclic met parfois du temps à arriver.
- Rester curieux (plusieurs solutions à un même problème).
- Faire un maximum d'exercices pour rencontrer un maximum de situations différentes.
- Ne pas hésiter à poser des questions (participation active aux TP).
- Anticipation des séances à venir (préparation à la maison).
- Profiter du projet et de la simulation d'examen pour tester son niveau en conditions réelles.

Comment construire un programme ?

L'ordinateur n'est capable d'exécuter qu'un nombre limité d'opérations différentes

→ ce sont les **instructions fondamentales** du processeur

→ on réalise des instructions plus complexes 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    $0xffffe7960,%ebx
7 callq  400d08 <_ZNSo3putEc@plt>
8 mov    %rax,%rdi
```

Ce **code machine** est très difficile à lire et à comprendre pour un humain

→ On utilise un **langage de programmation**, plus simple à comprendre et 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, ...)
- 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

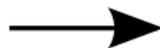
```
1 double truncation_error = fabs(f/h - (fplus +  
   fminus)/(2.*h));  
2 double roundoff_error =  
   std::numeric_limits<double>::epsilon()*f/h;  
3 u = truncation_error / roundoff_error;
```

Etapas fondamentales de la construction d'un programme

2 étapes principales:

- 1 **Ecrire** 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 double u = 1;
57 double derivative = 0;
58
59 int cptr = 0;
60 while( (u < 50) || (u > 200) && (cptr < 100) )
61 {
62     double fplus = (*derivfunction)(x+h);
63     double fplus2 = (*derivfunction)(x+2.*h);
64
65     double fminus = (*derivfunction)(x-h);
66     double fminus2 = (*derivfunction)(x-2.*h);
67
68     derivative = (fplus - fminus)*0.5/h;
69
70     // Evaluation of the error on the derivative from Curtis and Reid, J. Math
71     // Analysis
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     }
82 }
```

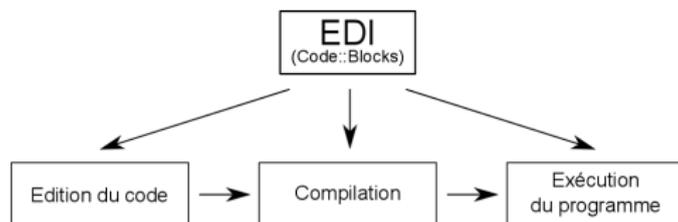


Écrire le code source → Code::Blocks

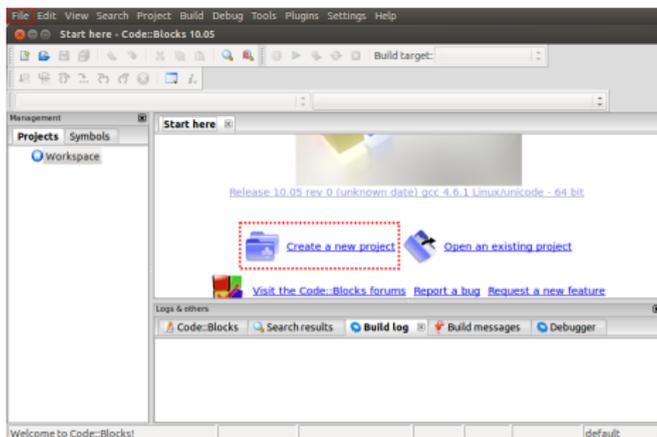
Code::Blocks = **Environnement de Développement Intégré**

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 erreurs de compilation lorsqu'il y en a. On peut ensuite exécuter le programme.

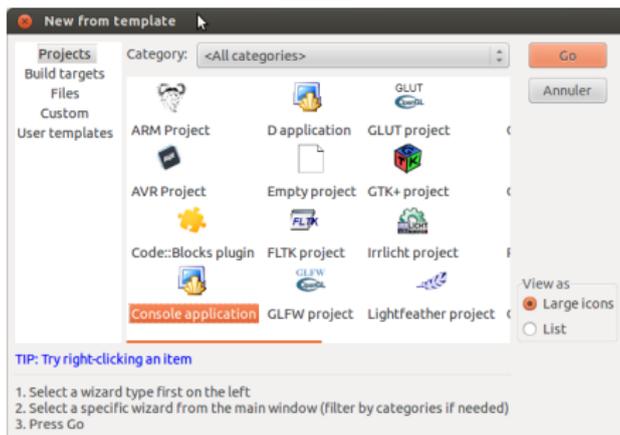


Créer un projet Code::Blocks (1/4)



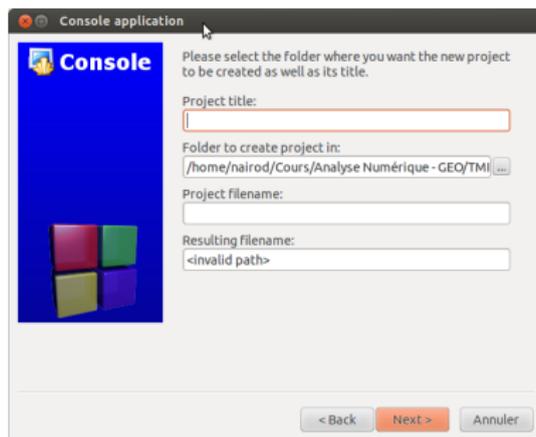
Cliquer sur "Create a new project"

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.

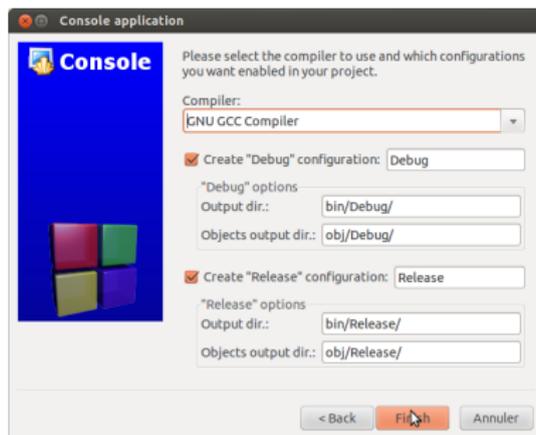
Créer un projet Code::Blocks (3/4)



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.

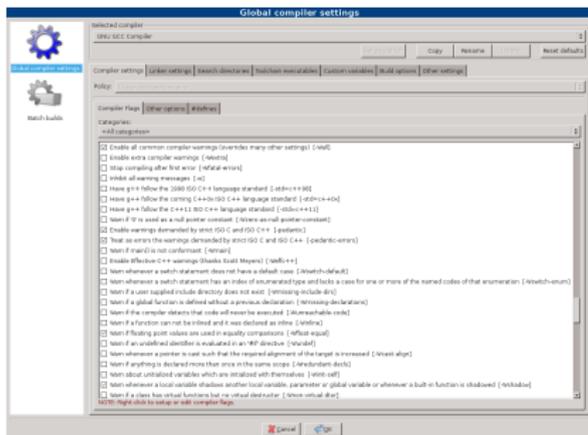
Créer un projet Code::Blocks (4/4)



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

Utiliser Code::Blocks chez soi (compiler flags)

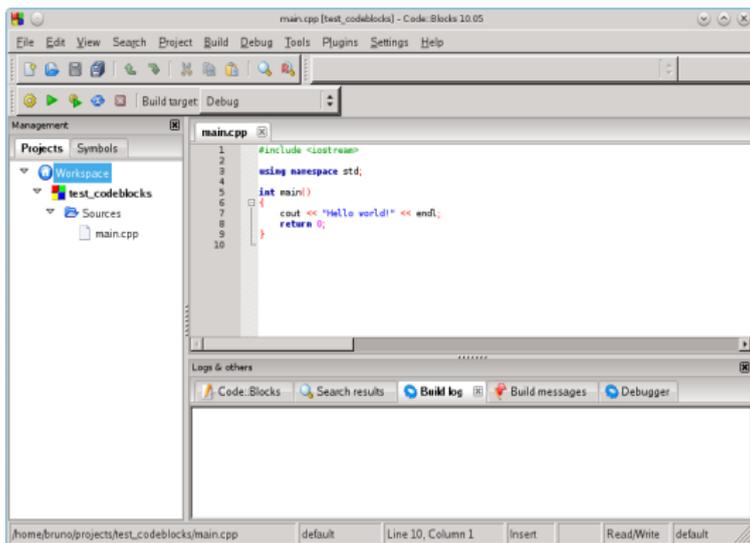
Lorsque vous utiliserez Code::Blocks chez vous, il est conseillé de modifier certains paramètres par défaut afin qu'ils correspondent à ceux utilisés lors des TP (**et de l'examen !**)
Ceux-ci se trouvent dans le menu "Settings/Compiler..."



Les cases supplémentaires à cocher sont:

- -Wall
- -pedantic
- -pedantic-errors
- -Wfloat-equal
- -Wshadow
- -std=c++11

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 }
```

La partie "instructions" du programme Le programme démarre toujours en exécutant une **fonction principale** appelée "**main**" Les instructions de cette fonction sont contenues entre des **accolades** { } L'objet "**cout**" permet l'affichage à l'écran

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!";
```

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

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: " << 3.1415926535897932384626 << 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: " << 3.1415926535897932384626 << endl;  
4 // affichage: Pi: 3.14159  
5 cout.precision(10);  
6 cout << "Pi: " << 3.1415926535897932384626 << endl;  
7 // affichage: Pi: 3.141592653
```

Site internet du cours

Toutes les informations relatives au cours sur :
<http://www.pqs.ulg.ac.be/>

Physique quantique statistique
Cours de physique statistique

Propositions de mémoires 2014-2015

- Dynamique chaotique d'une chaîne d'oscillateurs non linéaires
- Modélisation de la mesure de particules dans un laser à atomes
- Transport mésoscopique d'atomes ultrarabats
- Théorie semi-classique de l'effet tunnel
- Modélisation de gouttes résonnantes dans un bain liquide visqueux

Last Updated on Tuesday, 26 January 2014 08:22

Nos cours (2016-2017)

Introduction à la programmation (bachelier en sciences physiques et master en sciences physiques)
Outils mathématiques de la physique (bachelier en sciences physiques)

Atomes ultrarabats et condensats de Bose-Einstein (bachelier en sciences physiques)

Physique générale III (bachelier en sciences mathématiques)

Mécanique quantique et statistiques relativistes (master en sciences physiques)

Last Updated on Wednesday, 31 February 2017 14:09

MAIN MENU

- Home
- Start
- Research
- Publications
- Services
- Education
- Mémoires
- Location
- Log in

Physique quantique statistique
Cours de physique statistique

Introduction à la programmation (2016-2017)

Cours théorique dispensé à l'intention des étudiants bacheliers en sciences physiques par P. Schlegelack.
Travaux pratiques encadrés par X. Baerman et R. Christen.

- **ÉVALUÉS RELATIFS AU COURS THÉORIQUE ET AUX TRAVAUX PRATIQUES :**

Théorie	TP
Organisation	Organisation
Chapitre 1	TP1 (introduction)
Chapitre 2	TP2
Chapitre 3	TP3
Chapitre 4	TP4
Chapitre 5	TP4 (partiel)
Chapitre 6	TP5 (partiel)
Chapitre 7	TP6
Chapitre 8	TP7
Chapitre 9	TP8
	TP9 (évaluation examen)
	TP10 (évaluation)

- **Résultats des interrogatoires :**
Résultats obtenus (année à year) : 13/05/2017.
- **Code::Blocks - environnement de développement intégré :**
INFO058_Code::Blocks.pdf - Introduction à l'utilisation de Code::Blocks
Télécharger le code::Blocks (pour le forer nommé codeblocks-ex-xxxxxx@ulpg.org)
- **Exercices supplémentaires :**
INFO058 - Liste d'exercices classiquement supplémentaires
- **Recherche de références :**
Le Langage C/C++ (T. Baer, Université de Liège, 2005/6)
Introduction à l'Informatique (H.P. Gansz & F. Mangin, Université de Liège, 2005/6)

Last Updated on Friday, 22 May 2015 12:47

MAIN MENU

- Home
- Start
- Research
- Publications
- Services
- Education
- Mémoires
- Location
- Log in

Exercices

- Construire un projet “Hello World !”, le compiler et l’exécuter;
- Construire un programme affichant les premières décimales du nombre d’or $\Phi = 1.618033988749894848204586\dots$, 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...

```
  ^--^
 (oo)\_____
 (__)\         )\ \
   ||-----w |
   ||         ||
```

Langage de programmation visuel: Scratch

`https://scratch.mit.edu/`

Google : apprentissage du code par les enfants

`https://www.google.com/doodles/
celebrating-50-years-of-kids-coding`