Algorithmique & Langage C

IUT GEII S1

1 Notes de cours

(première partie)

cours_informatique_s1_1.21.odp


Licence


•Paternité - Pas d'Utilisation Commerciale -
• Partage des Conditions Initiales à l'Identique 2.0 France

•Vous êtes libres :
• * de reproduire, distribuer et communiquer cette création au public
• * de modifier cette création, selon les conditions suivantes :
•
•**Paternité**. Vous devez citer le nom de l'auteur original.
•
•**Pas d'Utilisation Commerciale**.
•Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.
•
•**Partage des Conditions Initiales à l'Identique**.
•Si vous modifiez, transformez ou adaptez cette création,
•vous n'avez le droit de distribuer la création qui en résulte
•que sous un contrat identiqueà celui-ci.
• * A chaque réutilisation ou distribution, vous devez faire apparaître clairement aux autres les
• conditions contractuelles de mise à disposition de cette création.
•* Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits
•Ce qui précède n'affecte en rien vos droits en tant qu'utilisateur (exceptions au droit d'auteur :
•copies réservées à l'usage privé du copiste, courtes citations, parodie...)
•voir le contrat complet sous : http://fr.creativecommons.org/contrats.htm


Plan du cours

- Algorithmique / Langage de programmation
- Premiers programmes
- Fichier source / Compilateur / Fichier exécutable
- Les variables : nom, type, valeur, déclaration
- Affichages : printf
- Saisies clavier : scanf
- Opérateurs arithmétiques: **+ - * / %**
- L'affectation : **=**
- Expressions
- Expressions et affectations entre types différents

Ressources

Cours, exemples, exercices, sujets des tp, compilateur :

http://arlotto.univ-tln.fr

Sur le net :

http://fr.openclassrooms.com/
http://c.developpez.com/cours/

Un ouvrage bien adapté pour débuter :

"Le livre du C premier langage" Claude Delannoy
Editions Eyrolles ISBN: 2212110529

Algorithme

Définition : Ensemble de règles opératoires
dont l'application permet de résoudre un problème
en un nombre fini d'opérations.

Algorithme : méthode, recette, procédure....

Exemples d'algorithmes :
- recette de cuisine,
- règle de divisibilité par 3
- résolution d'équations du second degré
- méthode de dépannage
- .....

Langage de programmation

Un algorithme est indépendant du langage dans lequel il est décrit.

Un algorithme peut être décrit en langage naturel mais pour pouvoir être exécuté par un ordinateur il faut le traduire dans un *langage de programmation*.

Langage de programmation :
Ensemble de symboles et de règles permettant de décrire un algorithme.
Il existe de nombreux langages de programmation...

Nous commencerons par apprendre le ***langage C***.
C'est un langage généraliste de base que tous les informaticiens connaissent.
C'est le langage le plus utilisé en électronique (micro,dsp,...)

Hello world !

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
    {
        printf("Hello world !\n") ;
        return 0 ;
     }
```

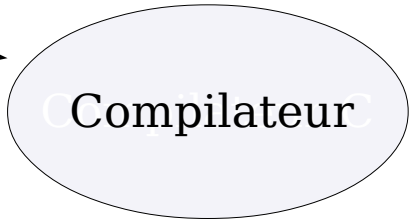

source -> compilateur -> exécutable

Fichier source

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
    {
        printf("Hello World !\n")
        return 0 ;
     }
```

Hello.c



Fichier exécutable

```
00111101
00011100
11001011
01110000
00001111
11001010
11100001
00000100
```

Hello.exe


source -> compilateur -> exécutable

Fichier source : un algorithme écrit en langage de programmation (en C)
c'est un fichier texte (lisible avec n'importe quel éditeur de texte (notepad). Il porte l'extension **.c** en langage C.

Compilateur : logiciel qui "comprend" le fichier source pour en générer un fichier exécutable.

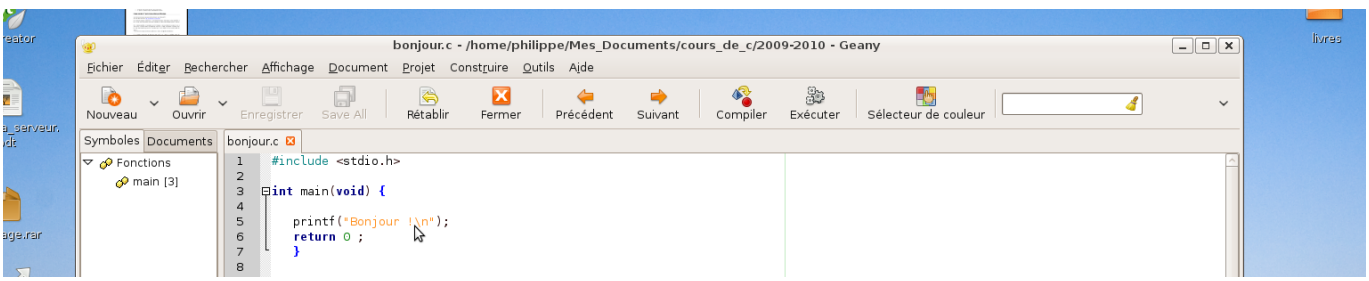
Fichier exécutable : fichier qui contient les instructions exécutées par le processeur.
(sous Windows un exécutable porte l'extension .exe, sous Linux le nom d'un exécutable peut être quelconque : des bits (x) définissent le droit d'exécution pour chaque utilisateur. (visibles par la commande : ls -l))

Environnements utilisés en tp

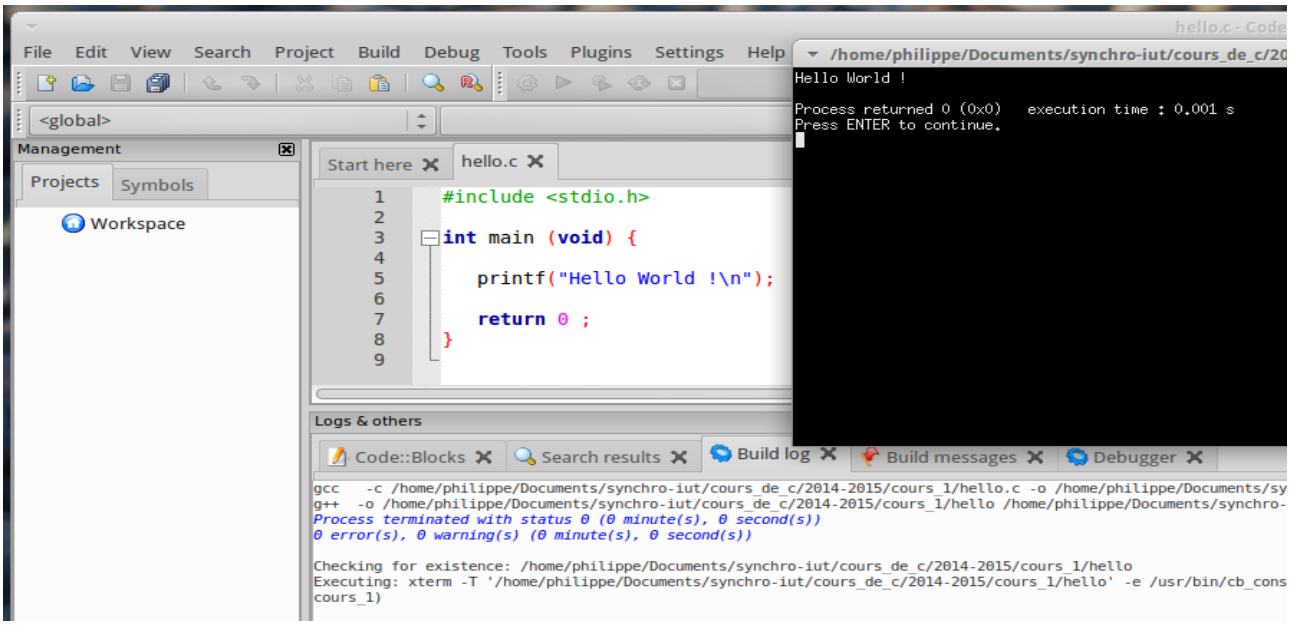
Pour les postes sous **Linux**, on peut utiliser directement gcc en ligne de commande et l'éditeur gedit (ou mousepad) :

```
$gedit tp1.c &      // tapez votre programme et sauvez-le
$gcc -Wall -o tp1 tp1.c
$./tp1
```

Ou bien l'environnement open source intégré très simple et très convivial : **Geany**.



On utilise aussi codeblocks qui fonctionne avec Windows et Linux. Voir : www.codeblocks.org



Faire du langage C chez soi

Vous utilisez Windows :

Vous avez **Windows 10,8,7 ou XP** :

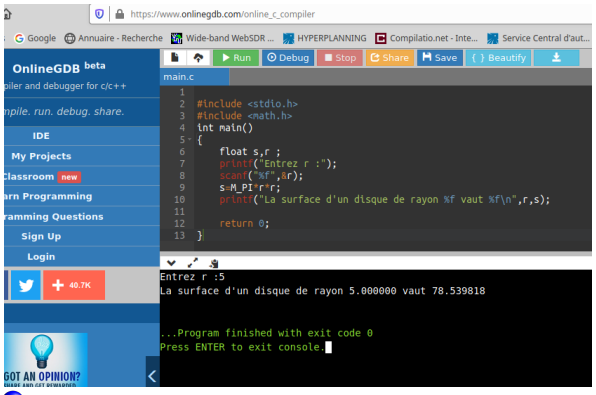
Installez **Code ::Blocks** et travaillez comme en tp.

Téléchargez : http://www.codeblocks.org/downloads/26
Attention prendre la version **avec mingw** :
codeblocks-17.12mingw-setup.exe
(ou plus récente)

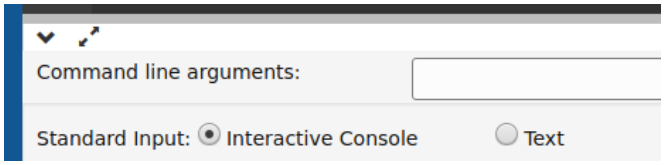
Si vous avez besoin d'aide, notez bien les messages d'erreurs. Si vous nous dites seulement "ça ne marche pas", on ne pourra pas vous aider !

En utilisant un ***compilateur en ligne*** vous pouvez faire tous les exercices sans rien installer. Cette méthode est compatible avec tous les OS. Il suffit d'avoir une connexion internet.
Par exemple :
https://www.onlinegdb.com/online_c_compiler

Il en existe plusieurs.
Faire une recherche "online c compiler".



Pour pouvoir entrer un texte avec un scanf sélectionnez la "console interactive".




Vous pouvez utilisez **linux** par exemple avec Xubuntu 20.04
Dans ce cas **Code::Blocks** ou **Geany** sont disponibles.
Tapez simplement `:$ sudo apt-get install codeblocks`

Pour utiliser gcc, voir aussi (document un peu ancien mais globalement valable) :
http://arlotto.univ-tln.fr/ressources/cours_de_c/compilateurs/linux/tpc_avec_linux.1.2.pdf

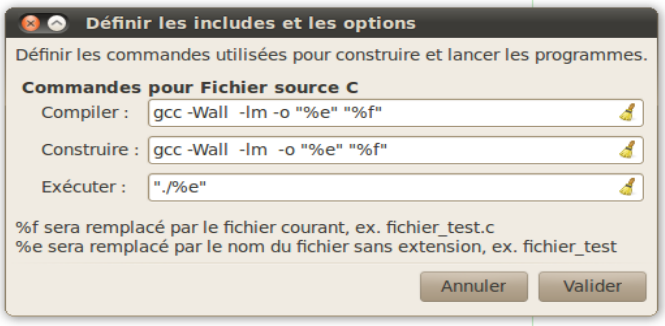
Vous pouvez démarrer Linux sur un pc équipé de Windows sans rien changer à sa configuration en utilisant un LiveCD ou une clé USB.
Voir : http://doc.ubuntu-fr.org/live_usb
http://doc.ubuntu-fr.org/live_cd

Vous avez un Mac ??? => Linux s'installe aussi sur les MAC :-)
Sinon gcc devrait fonctionner comme sur tous les unix-like.
Il y aussi xcode pour mac.


En plus de faire du C, vous voulez redécouvrir l'informatique
=> **Optez pour Linux, ça n'a rien à voir...**
Par exemple avec Ubuntu : http://www.ubuntu-fr.org/

Installez Code::Blocks : `$sudo apt-get install codeblocks`

Ou installez geany : `$ sudo apt-get install geany`
Ensuite dans "*Construire->Définir les includes et les options*"
Recopier la ligne de commande de Construire vers Compiler :



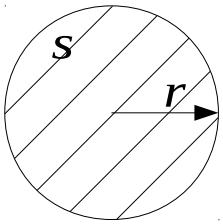
Ajouter également l'option **-lm** pour pouvoir utiliser la librairie mathématique

Un premier programme : surface d'un disque

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
    {
        float rayon, s ;
        printf("Valeur du rayon (m) : ") ;
        scanf("%f",&rayon) ;
        s = 3.14159 * rayon * rayon ;
        printf("La surface vaut %f m2\n",s);
        return 0 ;
     }
```

$$s = \pi r^2$$



Notion de bloc

Bloc : portion de programme comprise entre { et }

```
{
....
}
....
....
{
    .... {
          .....
        }
}
```

Les blocs rouge et vert sont ***disjoints***

Le bloc bleu est ***inclus*** dans le bloc vert.

Un bloc d'instruction se comporte comme une instruction unique


Les variables

Les variables sont les éléments de bases que manipule le programme. Ce sont des *cases mémoires* de l'ordinateur.

Une variable possède :

Un nom :
unique qui permet de la désigner dans le programme

Un type :
qui détermine ces propriétés :
valeurs possibles, limites, règles de calculs,.....

Une valeur :
La valeur d'une variable se conserve (mémoire) jusqu'à ce qu'elle soit modifiée par une opération d'écriture (affectation)


Les variables (nom)

Le nom : unique dans le bloc

caractères possibles : [a-z] , [A-Z] , [0-9]
et le caractères souligné _

Ne doit pas commencer par un chiffre.

Tous les compilateurs acceptent des noms
d'au moins 32 caractères (souvent beaucoup plus)

Par convention on n'utilise pas de nom entièrement
en majuscules pour les variables.

Choisir un nom évocateur de la fonction de la variable :

NbVoituresRouges TempMaxi temp_maxi surface_m2

Les variables (type)

Le type :
int : représente un partie des entiers relatifs.

au minimum [-32768;+32767] (16 bits)
souvent beaucoup plus $[-2^{31};+2^{31}-1]$ (32bits)
attention : ensemble fini.
plus grand + 1 = plus petit !!

float : représente un partie des nombres réels.

Au minimum $[-10^{+37};+10^{+37}]$
Précision limitée à 6 chiffres : tous les nombres ne sont pas représentables => Erreur d'arrondis

double :
idem à float mais meilleure précision (10 chiffres)

Les variables (valeur)

Valeur : c'est la valeur que contient la variable à un moment donnée du déroulement du programme.

Pour une variable déclarée dans un bloc, la valeur est quelconque à l'initialisation (au début de l'exécution du bloc).

La valeur se conserve jusqu'à ce que survienne une écriture.

```
{
...
int var ;   // var  est quelconque
var = 5;  //  var vaut 5
....
....          // var vaut toujours 5
var = 8 ; // var vaut 8
.....
```

Déclarations de variables

Pour être connue du compilateur, une variable doit avoir préalablement été ***déclarée***.

Instruction de déclaration : <type> <nom>[=<valeur>] ;

Exemples :

```
int var ;
int var2 = 9 ;
float x ;
double pression, temperature ;
int nombre, i=-78, j ;
float z = 42.56 , h = -1.23E-6 ;
```

Déclarations de variables

Norme Ansi **C89** : les instructions de déclarations doivent toutes être regroupées en *début de bloc*.
(tout de suite après un crochet ouvrant {)

Norme Ansi **C99** : les instructions de déclarations peuvent être placées n'importe où dans le bloc. La seule contrainte est que la déclaration doit se trouver ***avant*** l'utilisation.
(cas également du c++)

Les compilateurs gcc et mingw répondent à la norme C99.

Certains compilateurs C pour microcontrôleurs sont à la norme C89 (utilisés en S2,S3,S4).

Le compilateur Arduino est du c++.

Affichage sur l'écran

La fonction **printf** permet d'afficher :

un texte fixe (placé entre " et ") :

```
printf("bonjour monsieur");    affiche bonjour monsieur
```

cas particuliers :

\n	va à la ligne
\"	affiche "

```
printf("\"bonjour\"\n\n\"monsieur");  affiche    "bonjour"

                                                 "monsieur
```

Affichage sur l'écran

La fonction **printf** permet d'afficher :

la valeur d'une variable en utilisant les ***formateurs*** %...

```
int i=7;
printf("%d",i);   affiche 7
```

%d pour afficher un entier en décimal

%x pour afficher un entier en hexadécimal

%f pour afficher un float ou un double

%% pour afficher %

il existe de nombreux autres formateurs .....

Affichage sur l'écran

On peut mêler du texte et des formateurs.
Lorsqu'il y a plusieurs formateurs, le premier se rapporte à la première variable, le deuxième à la deuxième variable,...

```
int i = 4 , j = 5 ;
float temp = -1.43 ;
double z = 12.32 ;

printf("i=%d , temp=%f  j=%d \net z vaut %f\n", i , temp , i , z ) ;
```

affiche

```
i=4 , temp= -1.430000 j=4
et z vaut 12.320000
```

Saisie d'une valeur à partir du clavier

On utilise la fonction scanf :

```
int i ;

scanf("%d",&i);
```

1/ stoppe l'exécution du programme en attente de l'appui sur la touche Entrée

2/ tente* de convertir les caractères tapés avant Entrée selon le format spécifié et place le résultat dans la variable spécifiée.

* si la conversion échoue la variable n'est pas modifiée


Saisie d'une valeur à partir du clavier

Saisie d'une valeur décimale dans un int : %d

```
int i ;
....
scanf("%d",&i);
```

Saisie d'une valeur réelle dans un float : %f

```
float x ;
.....
scanf("%f",&x);
```

Saisie d'une valeur réelle dans un double : %lf

```
double z ;
.....
scanf("%lf",&z);
```


Les opérateurs arithmétiques

Addition : +
Soustraction : -
Multiplication : *
Division : /
lorsque un des opérandes est réel / représente la division réelle
lorsque les deux opérandes sont entiers / représente la division euclidienne (quotient)

Reste dans la division euclidienne : %
(entre opérandes entiers uniquement)

dividende diviseur

reste quotient

Expression

Une expression est une construction formée à partir d'opérateurs et de variables, constantes ou d'autres expressions.

Exemples d'expressions :

x+2 2*x+y*z 3*(x+2)/4 5*x 4+7 1

Une expression possède un **type** et une **valeur**.

Lorsque les opérandes ont tous le même type,
Le type de l'expression est celui des opérandes.
(Lorsque les types diffèrent des règles de conversions s'appliquent)

Déterminer la valeur d'une expression : **évaluer** l'expression

L'affectation

C'est l'opération qui permet de stocker une valeur dans une variable. Elle est représentée par le caractère = .
Forme générale :
<variable> = <expression> ;

Cette opération se déroule en *deux temps* :

1/ l'expression est évaluée

puis

2/ la valeur de l'expression est écrite dans la variable

L'affectation

```
int i , j ;
```

`i = 3 ;` L'expression est évaluée, on trouve 3, donc i vaut 3

`j= i + 4 ;` L'expression est évaluée, on trouve 7, donc j vaut 7

`i = i + 1 ;` L'expression est évaluée, on trouve 4 (3+1), donc i vaut 4

`j = i + 2*j ;` L'expression est évaluée, on trouve 18, donc j vaut 18

Affectation entre types différents

Règle : l'affectation force une conversion dans le type d'arrivé (de gauche).

Attention :

La conversion n'est pas toujours possible.

Quand elle est possible, elle n'a pas toujours le sens mathématique que l'on pourrait souhaiter.

Quelques conversions

De int vers float (ou double)
De float vers double :
Pas de problème. Pas de perte d'information.
Ce sont des *conversions non dégradantes*.

De float (ou double) vers int :
Il y a suppression de la partie décimale.
(et pas "le plus proche")
C'est une *conversion dégradante.*

De double vers float :
conversion vers la valeur représentable la plus proche.
Perte de précision.
C'est une *conversion dégradante*

Expressions entre types différents

Règle : on applique des conversions non dégradantes tant que c'est possible.

Exemple :

```
int i = 3, j =1 , k , m , n, p ;
float x = 1  , y=1.6 , z , t , u , v , w ;

k = j /  i  ;
z = j /  i  ;
t = x / i  ;
m = y ;
n = y * i ;
u = (y * 10) / i ;
v =  y * (10 / i ) ;
w = y * (10.0 / i ) ;
p = (y * 10) / i ;
```

Trouvez les valeurs de k,m,n,p et z,t,u,v,w.
Vérifiez vos résultats avec un programme.

Travail à effectuer avant la prochaine séance

- Installez **CodeBlocks** ou utilisez le compilateur en ligne :

https://www.onlinegdb.com/online_c_compiler

- Compilez un premier programme (par exemple celui de la page7) , exécutez-le puis essayez de le modifier.
(Notez bien les messages d'erreurs en cas de problème)

http://arlotto.univ-tln.fr/ressources/cours_de_c/période_1/

- Essayez de faire les premiers programmes des TP :

Il est impératif de commencer très rapidement.