

Chapitre 3:

Généralités sur le

langage C

Historique

- **Naissance** : 1970
- **Père** : D. RITCHIE et B.W. Kernighan
- premières normes en 1978 puis norme ANSI en 1989 et ISO en 1990.
- **Objectif** : Réécriture d'UNIX (Ken THOMPSON) écrit en assembleur et langage B (langage non typé).
- **Conséquences** : C est un langage évolué, capable de traiter de manière standard des informations de bas niveau très proches de la structure matérielle des ordinateurs (mots, adresses, registres, octets, bits ...)

Bibliographie

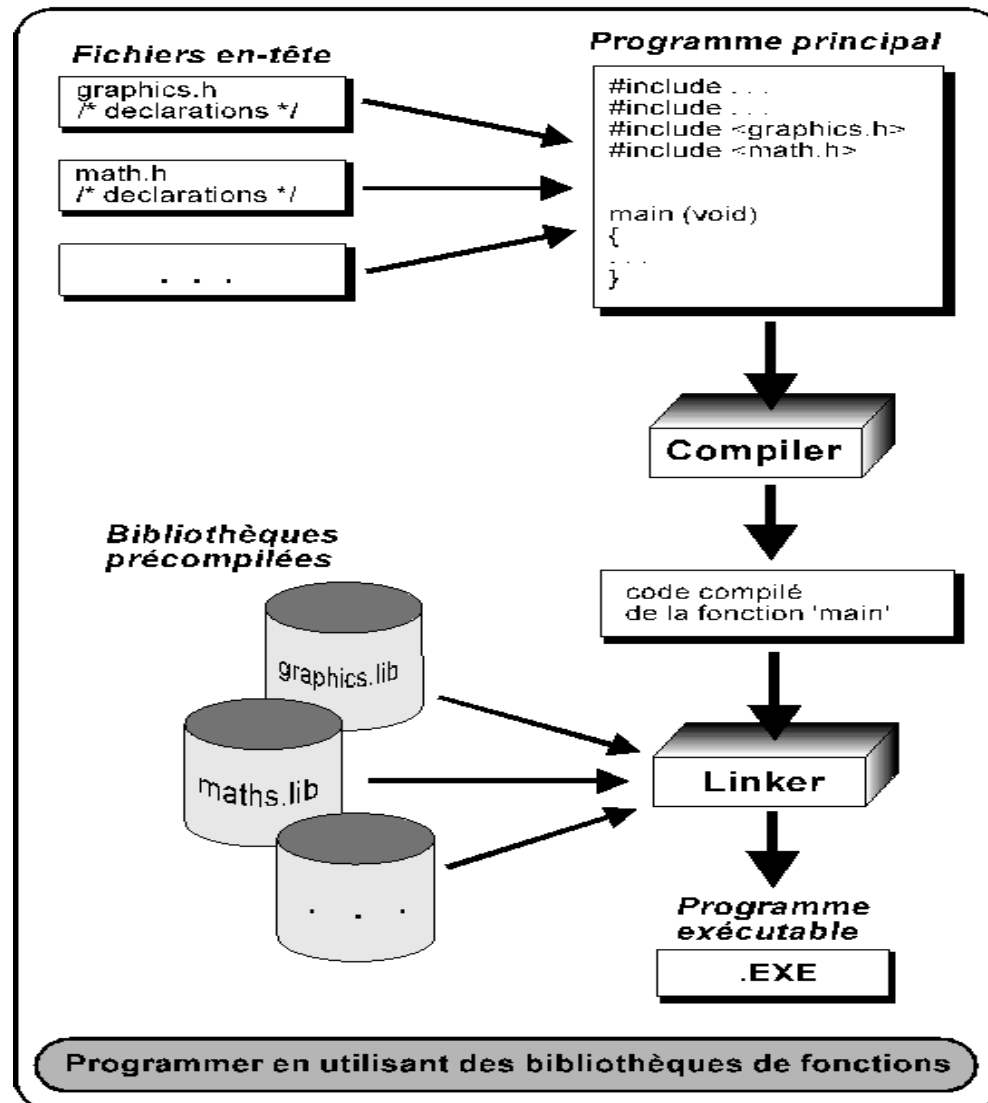
1. Le livre du C premier langage pour les débutants en programmation.
2. Programmer en langage C de [Claude DELANNOY](#)
3. Internet, Commentcamarche, [OpenClassroom](#),...
4. Le langage C norme ANSI (2ème édition) Kernighan & Ritchie (masson)
5. Programmation en C Gottfried (Schaum)

Compilation

- Le C est un langage compilé. Cela signifie qu'un programme C est décrit par un fichier texte, appelé fichier source. Ce fichier n'étant évidemment pas exécutable par le microprocesseur, il faut le traduire en langage machine c'est à dire le compiler.
- La compilation consiste à traduire le programme source en langage machine, en faisant appel à un programme nommé **compilateur**. Cette opération génère deux fichiers:

Un fichier objet et fichier exécutable

Compilation



Logiciels nécessaires

1. Un **éditeur de texte** pour écrire le code source du programme en C.
2. Un **compilateur** pour transformer (« compiler ») votre code source en binaire.
3. Un **débugger** (« Débogueur » ou « Débugueur » en français) pour vous aider à traquer les erreurs.

➡ utiliser un IDE (ou en français « EDI » pour « Environnement de Développement Intégré »).

Logiciels nécessaires

- **Code::Blocks** est un **IDE** libre et gratuit, disponible pour Windows, Mac et Linux.
- Rendez-vous sur la page de téléchargement de code-blocks:

<http://www.codeblocks.org/downloads/binaries>

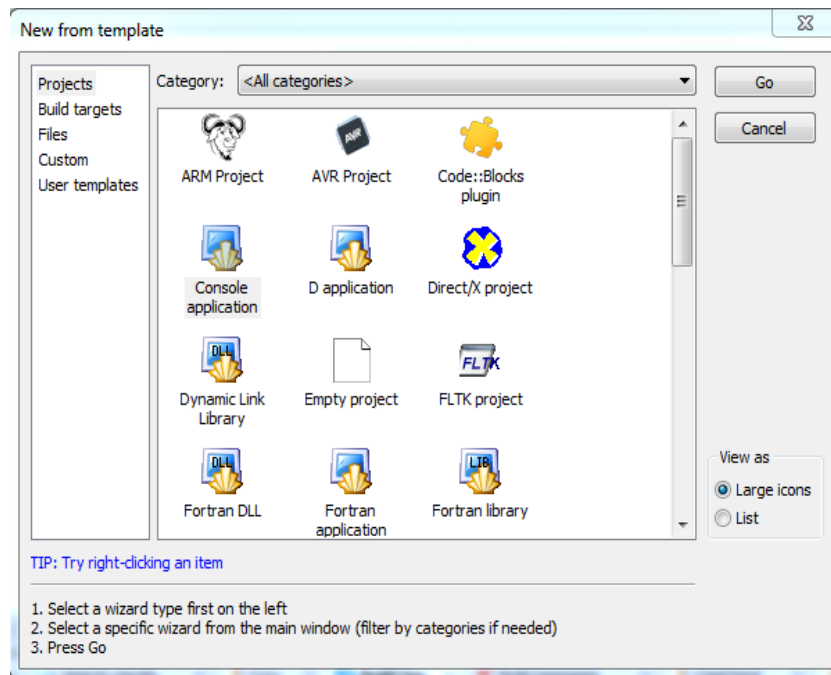
- Repérez la section Windows et télécharger le logiciel dont le nom contient **mingw**.

Premiers pas

- Les projets
- ✓ Quand vous réalisez un programme, on dit que vous travaillez sur un projet. Un projet est constitué de plusieurs fichiers de code source : des fichiers .c, .h, les images du programme, etc.
- ✓ Le rôle d'un IDE est de rassembler tous ces fichiers au sein d'une même interface.

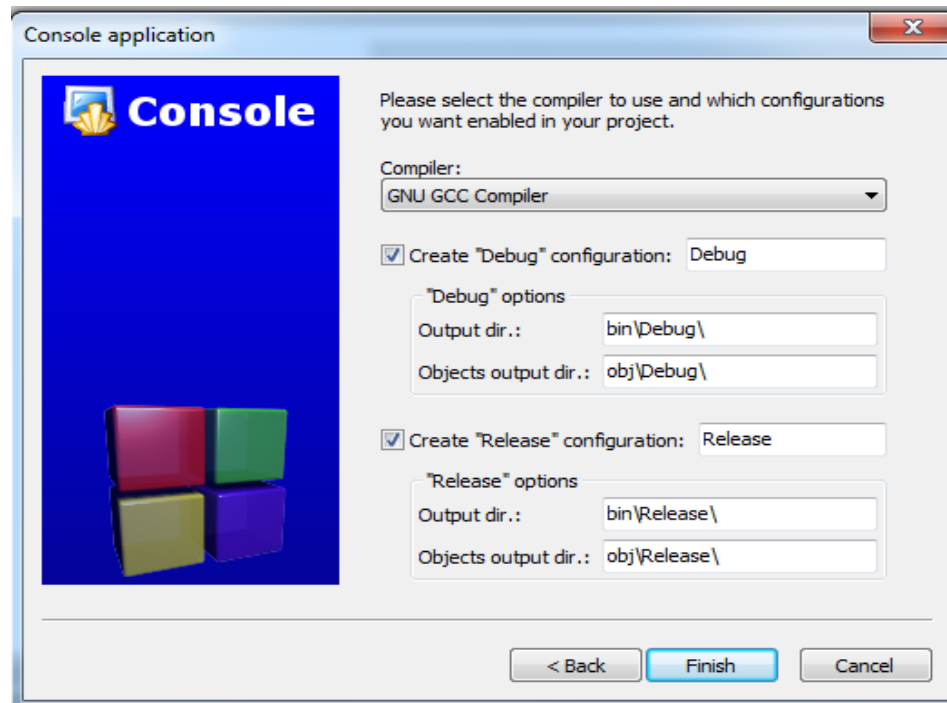
Premiers pas

- Créer un nouveau projet
 - ✓ Allez dans le menu File>New>Project.
 - ✓ Dans la fenêtre qui s'ouvre, choisissez Console application (figure suivante).



Premiers pas

- ✓ Cliquez sur GO, puis sur Next
- ✓ Choisissez un projet C et cliquez sur Next
- ✓ Donnez un nom et un emplacement pour votre projet
- ✓ Enfin cliquez sur Finish

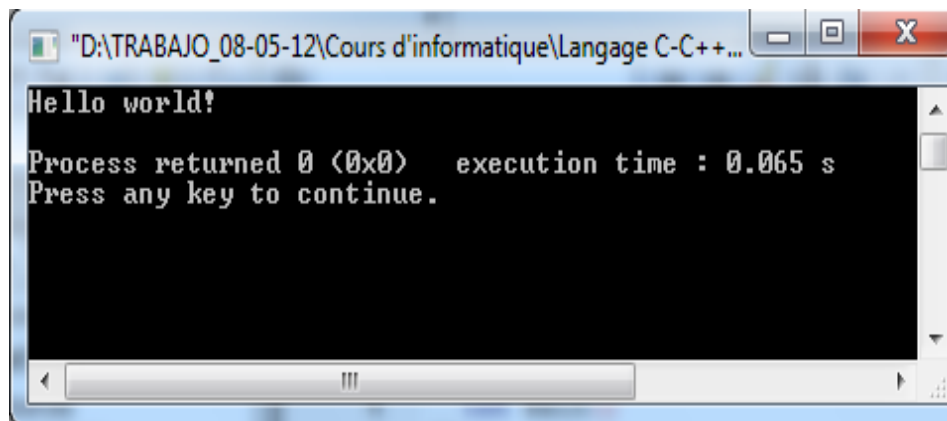


Premiers pas

- ✓ C'est bon! Code::Blocks vous crée un premier projet contenant déjà un tout petit peu de code source.
- ✓ Compilez votre programme.
- ✓ Exécuter votre programme



- ✓ Et voilà!!



```
"D:\TRABAJO_08-05-12\Cours d'informatique\Langage C-C++..."  
Hello world!  
Process returned 0 (0x0) execution time : 0.065 s  
Press any key to continue.
```

Présentation du Langage C par exemple

- Voici un exemple plus complet qui permet de calculer la surface d'un cercle.

```
# include <stdio.h>
# include <stdlib.h>
# define pi 3.14
int main(int argc, char *argv[ ])
{
    float R, S ;
    puts("veuillez entrer le rayon R :") ;
    scanf("%f", &R);
    S = pi*R*R;
    printf("la surface S est %f \n ",S) ;
    system("PAUSE");
    return 0;
}
```

Les directives à destination du préprocesseur

- Les trois premières lignes de notre 1er programme
 1. `# include <stdio.h>`
 2. `# include <stdlib.h>`
 3. `# define pi 3.14`
- Il s'agit de « directives » qui seront prises en compte avant la traduction (compilation) du programme.
- Ces directives, contrairement au reste du programme, doivent être écrites une par ligne et elles doivent obligatoirement commencer au début de la ligne.

Les directives à destination du préprocesseur

- D'une manière générale, il est préférable de les mettre au début du programme, comme nous l'avons fait.
- Les deux premières directives demandent en fait d'introduire (avant compilation) des instructions (en langage C) situées dans les fichiers `stdio.h` et `stdlib.h`.
- Notons que, dès lors que nous faisons appeler une fonction prédéfinie, il est nécessaire d'incorporer de tels fichiers, nommés «fichiers en-têtes», qui contiennent des déclarations appropriées concernant cette fonction.

Les directives à destination du préprocesseur

Exemple:

<code>stdio.h</code>	<code>stdlib.h</code>	<code>math.h</code>
<code>printf, scanf,</code> <code>getch, puts,</code> <code>putchar, gets,</code> <code>getchar,...</code>	<code>abs, exit, rand</code> <code>malloc, realloc,</code> <code>system...</code>	<code>sqrt, cos, sin, tan,</code> <code>exp, floor, log...</code>

- Notez qu'un même fichier en-tête contient des déclarations relatives à plusieurs fonctions.
- En général, il est indispensable d'incorporer `stdio.h`

Les directives à destination du préprocesseur

- La troisième directive demande simplement de remplacer systématiquement, dans toute la suite du programme, le symbole `pi` par `3.14`.
- Ici la directive `#define` servait à définir la valeur d'un symbole (constante).

Structure d'un programme en langage C

- La ligne `: int main(int argc, char *argv[])` C'est une en-tête de fonction.
- Dans ce cas, le programme ne possède qu'une seule fonction, la fonction principale `main`. Cette ligne est obligatoire en C: elle définit le « point d'entrée » du programme, c'est-à-dire l'endroit où débutera l'exécution du programme.
- Le mot `main` (principale en français) est imposé. Le programme principale vient à la suite de cet en-tête. Il est délimité par des accolades `{` et `}`.

Déclarations

- L'instruction `float R, S;` est une déclaration. Elle est sous la forme :
`type liste_de_variables ;`
- On définit le nom que l'on choisit pour chaque variable, ainsi que son type, ici `float`, c'est-à-dire réel (type dit à virgule flottante).
- Dans un programme C, on ne peut jamais utiliser une variable sans l'avoir déclaré auparavant.

Instructions

- Une instruction est un ordre élémentaire que l'on donne à la machine, qui manipulera les données (variables) du programme, ici soit par un appel d'une fonction (puts, scanf, printf) soit par une affectation (=).
- Toutes les instructions se terminent par un ;
- Notre programme comporte 4 instructions:

Instructions

- i. `puts("veuillez entrer le rayon R :") ;`
- ii. `scanf(" %f ", &R) ;`
- iii. `S = pi * R * R ;`
- iv. `printf("la surface S est %f \n ",S) ;`

Détaillons ces quatre instructions.

Instructions

- L'instruction: `puts("veuillez entrer le rayon R :") ;` appelle une fonction prédéfinie (fournie avec le langage C) nommé `puts`. Ici cette fonction reçoit un argument qui est une chaîne de caractères :
`veuillez entrer le rayon R :`
- Les guillemets servent à délimité cette chaîne de caractères.

Instructions

- L'instruction: `scanf(" %f ", &R);`
- `scanf` attend l'entrée d'une valeur au clavier, puis elle la met dans la mémoire (on préfère dire dans la variable),
- La fonction `scanf` ici reçoit deux arguments:

`%f` et `&R`

Instructions

- L'instruction : $S = \text{pi} * R * R;$ est une affectation.
- on multiplier pi fois R fois R et le résultat de ce calcul est stocké (affecté) dans la variable S .
- **Remarque:** Une affectation se fait toujours dans le même sens: on détermine (évalue) tout d'abord la valeur à droite du signe $=$, en faisant tous les calculs nécessaires, puis elle est transférée dans la mémoire dont le nom qui est indiqué à gauche du signe $=$.
- On peut donc placer une expression à droite de $=$, mais à sa gauche seul un nom de variable est possible (appelée *lvalue*).

Instructions

- L'instruction : `printf("la surface S est %f \n ",S) ;` appelle la fonction prédéfinie `printf`. Elle reçoit ici deux arguments
 - ✓ `la surface S est` : message fixe
 - ✓ `%f` : format associé au variable a afficher dans cet emplacement
 - ✓ `S` : Variable a afficher dont le format est `%f`
- Remarque:
 - ✓ `\n` : retour a la ligne
 - ✓ Autre exemple : `printf("Bonjour\n ") ;`

Instructions

- La ligne : `system("PAUSE");` Elle est nécessaire pour que le programme s'arrête.
- Il est recommandé de rajouter l'instruction `system("PAUSE")` juste avant l'instruction `return 0;`

Exercices

- Écrire un programme qui affiche bonjour
- Écrire un programme qui lit un réel et qui l'affiche
- Écrire un programme qui lit un réel a qui l'affiche et qui calcul son racine carré.
- Écrire un programme qui calcul la surface d'une pièce rectangulaire de longueur a et de largeur b.
- Écrire un programme qui calcul la somme de deux entiers.

Les identificateurs

- Les identificateurs servent à désigner les différents «objets» manipulés par le programme: variables, fonctions,... Comme dans la plus part des langages, ils sont formés d'une suite de caractères choisis parmi les lettres ou les chiffres, le premier d'entre eux étant nécessairement une lettre.
- le caractère « souligné » (`_`) est considéré comme une lettre. Il peut donc apparaître au début d'un identificateur. Voici quelques identificateurs corrects:
`lg_ligg` `valeur_5` `valeur77` `_2008`

Les identificateurs

- Les majuscules et les minuscules sont autorisées mais ne sont pas équivalentes (contrairement, par exemple, à ce qui se produit en Pascal).
- En ce qui concerne la longueur des identificateurs, la norme ANSI prévoit qu'au moins les 31 premiers caractères soient « significatifs »

Correct	Incorrect	Raison
Variable	Nom de variable	Comporte des espaces
Non_de_variable	52Non_de_variable	Commence par un chiffre
non_de_variable	h5@site.fr	Caractère spécial @
_variable_1	Non-de-variable	Signe – interdit
_2136457895421	float	Nom réservé (mot clé)

Les mots clés

- Certains « mots clés » sont réservés par le langage à un usage bien défini et ne peuvent pas être utilisés comme identificateurs.
- En voici la liste, classée par ordre alphabétique:

auto	default	float	register	struct	volatile
break	do	for	return	switch	while
case	double	goto	short	typedef	
char	else	if	signed	union	
const	enum	int	sizeof	unsigned	
continue	extern	long	static	void	

Les séparateurs

- Dans notre langue écrite, les différents mots sont séparés par un espace, un signe de ponctuation ou une fin de ligne.
- Il en va quasiment la même chose en langage C dans lequel les règles vont donc paraître naturelles. Ainsi, dans un programme, deux identificateurs successifs entre lesquels la syntaxe n'impose aucun signe particulier doivent impérativement séparés soit par un espace, soit par une fin de ligne.

Les séparateurs

- Ainsi, vous devrez impérativement écrire
`float x,y;`
Et non :
`foaltx,y;`
- En revanche, vous pourrez écrire indifféremment:
`int n,compte,total,p;`
Ou plus lisiblement :
`int n, compte, total, p ;`

Le format libre

- Le langage C autorise une « mise en page » parfaitement libre. Les fins de ligne jouent un rôle de séparateurs, comme un espace, sauf dans les constantes chaînes, où elle sont interdites.
 - i. Une ligne peut comporter plusieurs instructions.
 - ii. Une instruction peut s'étendre sur plusieurs lignes.
 - iii. Attention à la lisibilité du programme.

Les commentaires

- Les commentaires sont des textes explicatifs destinés aux lecteurs du programme et qui n'ont aucune incidence sur le compilateur.
- Il sont formés de quelques caractères placés entre les symboles `/*` et `*/`
- On les met en général à des emplacements propices à une bonne lisibilité du programme.
- Voici quelques exemples de commentaires

Exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
/* programme de calcul de racines carrées */
main()
{
    int a; /* déclaration de la variable a de type entier*/
    float rac ; /* déclaration de la var rac */
    puts("Tapez un entier positif: ") ;
    scanf("%d", &a);
    rac=sqrt(a); /* L'en-tête math.h est obligatoire
                  pour que la fonction sqrt() fonctionne */
    printf("La racine carée de %d est egal %f \n ",a,rac) ;
    system("PAUSE");
    return 0;
}
```