Université Paris Cité

UE Programmation Impérative

Langage C

F. CLOPPET

2024-2025

SOMMAIRE

- Informations pratiques
- Introduction
- Eléments de base
  - Programmer en Langage C – Compilation
  - Structure d'un programme / Règles d'écritures
  - Types de base
  - Constantes/Variables
  - Opérateurs
  - Instructions de contrôle
  - Pointeurs
  - Tableaux
- Fonctions
- Chaînes de caractères
- Pointeurs- Tableaux-Fonctions
- Types Construits
- Entrées – Sorties sur Fichiers
- Compilation séparée
- Implémentation de Types Abstraits de Données

Informations Pratiques

- Equipe pédagogique
- Planning
- Modalités de Contrôle des connaissances
- Descriptif de l'UE
  - Objectifs
  - Support de Cours
  - Bibliographie

Equipe pédagogique

Florence Cloppet

Neilze Dorta

Alla Jammine

prenom.nom@u-paris.fr

Planning

			Horaire	Salle
Cours		Lundi	08h00-09h30	Amphi Fourier
TP Gr1	A. Jammine	Mercredi	13h30-16h30	Fourier D-529
TP Gr2	F. Cloppet	Lundi	15h00-18h00	Cordier-523 B
TP Gr3	A. Jammine	Mardi	14h00-17h00	Fourier A-526
TP Gr4	N. Dorta	Vendredi	09h00-12h00	Cordier A- 523
TP Gr5	A. Jammine	Mardi	09h45-12h45	Fourier A- 526
TP Gr6	N. Dorta	Mercredi	08h00-11h00	Fourier A-526

Modalités de contrôle des Connaissances

- Contrôle continu (CC) *intégral avec 2eme chance intégrée*
  - CC1: Epreuve sur ordinateur (semaine des CC, semaine du 11 novembre)
  - CC2: QCM sur moodle (décembre),
  - CC3: Contrôle terminal
    - *épreuve écrite sur copie de 1h30 (janvier)*

- Note = max(*CC3 ; (2*CC1 + 1*CC2 + 3*CC3) / 6)*

- **Aucun document autorisé lors des évaluations**

Objectifs de l'UE

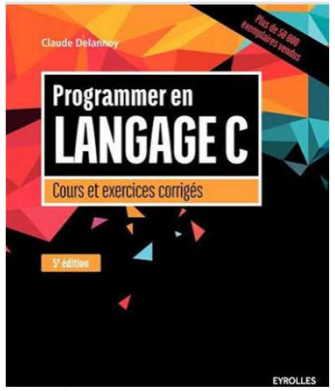
- Consolider et développer les compétences acquises en Initiation à la programmation
  - Langage de programmation = langage C

- **Compétences acquises**
  - Savoir traduire un algorithme en langage C
  - Savoir compiler et exécuter des programmes avancés en langage C

Bibliographie non exhaustive

- Le plus complet ….
  - Le langage C Norme ANSI , ***Brian W. Kernighan et Dennis M. Ritchie***, Dunod 2ème Edition (2022)

- Un peu plus pédagogique
  - Programmer en langage C, ***Claude Delannoy***, Eyrolles (5e édition 2016)



Support de Cours

- Poly disponible sur **moodle**

ou temporairement sur **http://www.mi.parisdescartes.fr/~cloppet/ProgImperative/**

- Tout n'est pas dans le poly …..

Ce poly n'est là que pour vous aider à la prise de notes et ne vous dispense en aucun cas d'assister au cours !!!!

- Les TDs sont corrigés en séance interactivement avec les étudiants

- Pas de corrigé donné sur les tps mais votre enseignant.e est là pour évaluer avec vous pendant la séance les programmes que vous aurez écrits

SOMMAIRE

- Informations pratiques
- Introduction
- Eléments de base
  - Programmer en Langage C – Compilation
  - Structure d'un programme / Règles d'écritures
  - Types de base
  - Constantes/Variables
  - Opérateurs
  - Instructions de contrôle
  - Pointeurs
  - Tableaux
- Fonctions
- Chaînes de caractères
- Pointeurs- Tableaux-Fonctions
- Types Construits
- Entrées – Sorties sur Fichiers
- Compilation séparée
- Implémentation de Types Abstraits de Données

Introduction

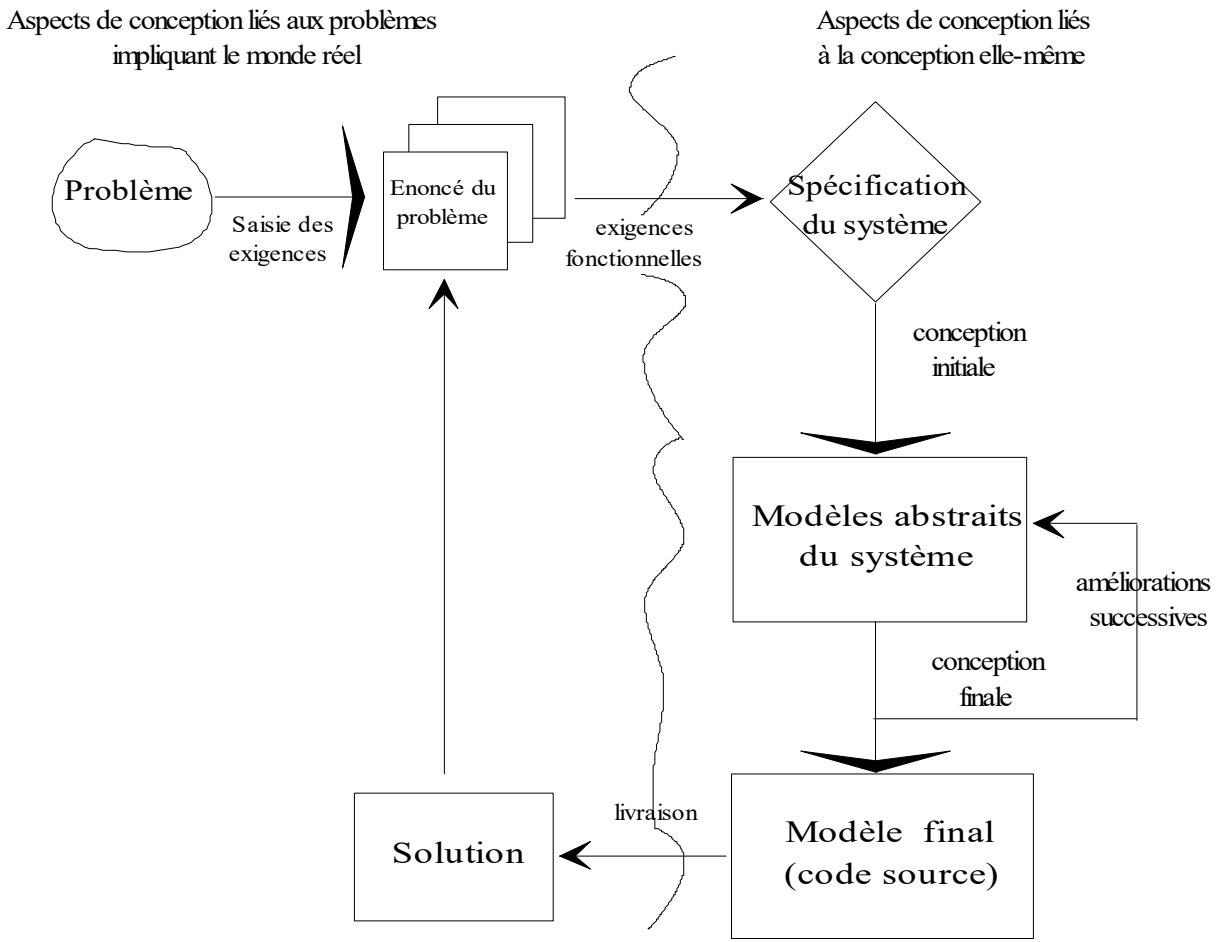
- Notions générales sur la programmation
  - Activité de Programmation
  - Paradigmes de Programmation
  - Langages de Programmation
- Langage C
  - Programmer en Langage C – Compilation
  - Règles de base
  - Eléments de base sur les types, constantes, variables

Notions générales sur la programmation

- Activité de programmation
  - Étudier le problème à résoudre
    - solution programmable ?
  - Développer un programme optimal
    - programme fait ce qu'il est censé faire
    - programme prend en compte les situations anormales
    - programme facile à modifier et à étendre

    Ces 2 points sont liés avec **le cours d'algorithmique et structures de données**
  - Limiter les coûts de production
    - Appel aux techniques du génie logiciel
      - le développement d'un logiciel doit être divisé en plusieurs parties pour être contrôlable
      - modèles de développement de logiciels

Notions générales sur la programmation



Vue générale des activités de développement d'un système

Notions générales sur la programmation

- Paradigmes de Programmation
  - Programmation Impérative
    - Programme représenté par une machine à états qui représente les états successifs de la mémoire
      - mémoire centrale et des instructions qui modifient son état grâce à des affectations successives
    - Basic, Pascal, Python, Langage C ….
  - Programmation Fonctionnelle
    - Toute opération d'affectation interdite
    - Programme décrit par un emboitement de fonctions
      - Un ou plusieurs paramètres en entrée
      - Une seule sortie
    - Lisp, Common Lisp, Caml, Scheme …

Notions générales sur la programmation

- Langages de programmation
  - Langage binaire
    - Processeur ⇔ circuits électroniques
    - Système de communication binaire (0-1)

  - Langage de plus haut niveau
    - langage d'assemblage ou assembleur
      - Encore très proche de la machine
    - langages de haut niveau (langages compilés, interprétés)
      - Plus proche du langage naturel

```
RxCar   btfss   PIR1,RCIF
        return
        bcf     PIR1,RCIF
        clrf    TimeoutRx
        btfsc   RCSTA,OERR      ; over run error
        goto    RxCarEr
        Mov     Rx_Car,RCREG    ; lecture uart RCREG
        SiV8SupIm Rx_Car,128,RxCar2 ; <128->0
        MovIm   Rx_Car,0
RxCar2  ;MovPt8 Rx_Ptr,Rx_Car   ;  Caractere bien recu
        movf    Rx_Ptr,w
        movwf   FSR
        movf    Rx_Car,w
        movwf   INDF
        SiV8EgIm Rx_NumCar,(4-1),RxCar3
        incf    Rx_Ptr,f
        incf    Rx_NumCar,f
        goto    RxCarF
```

```
if (nombreEntree < nombreMystere)
{
    printf("c'est plus\n");
    coups++  ;
}
else if(nombreEntree > nombreMystere)
{
    printf("c'est moins\n") ;
    coups++    ;
}
```

Notions générales sur la programmation

- Langages de programmation impérative de haut niveau
  - Instructions impératives principales
    - l'assignation
      - Opération sur l'information en mémoire
    - le branchement conditionnel
      - Bloc d'instructions exécuté que si une condition prédéterminée est réalisée
    - le branchement sans condition
      - Séquence d'exécution transférée à un autre endroit du programme (appel de procédure ou fonction, appel de sous-programme, « goto » …)
    - le bouclage
      - Répétition d'une suite d'instructions un nombre prédéfini de fois ou jusqu'à ce qu'une certaine condition soit réalisée.

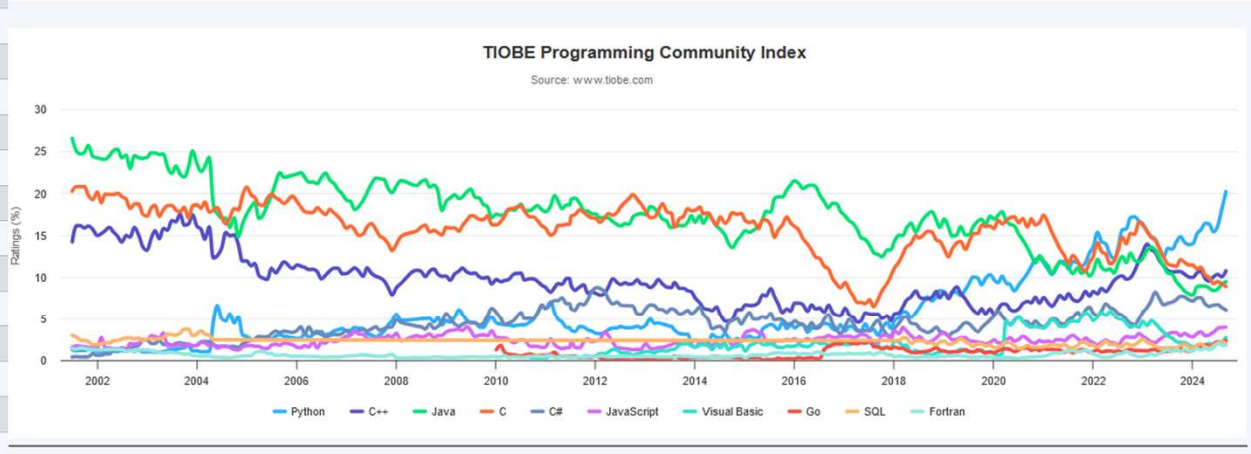
Langage C

- Conçu par B.W. Kernighan et D.M. Ritchie (années 1970)
  - ☞ pour développer le système d'exploitation Unix sur les stations de travail
- Langage structuré de haut niveau
  - fonctions, instructions impératives principales …
- Mais aussi un Langage de plus bas niveau
  - opérateurs permettant de travailler au niveau du codage machine
- Il s'étend des stations de travail aux micro-ordinateurs
- Norme C ANSI (American Standards National Institute)
  - ☞ donner une définition non ambiguë et indépendante de la machine
- C++ est une extension de C

Langage C

- Top 10 des langages de programmation
  - Classement TIOBE (Sept 2024)

Sep 2024	Sep 2023	Change	Programming Language	Ratings	Change
1	1		Python	20.17%	+6.01%
2	3	↑	C++	10.75%	+0.09%
3	4	↑	Java	9.45%	-0.04%
4	2	↓	C	8.89%	-2.38%
5	5		C#	6.08%	-1.22%
6	6		JavaScript	3.92%	+0.62%
7	7		Visual Basic	2.70%	+0.48%
8	12	⇈	Go	2.35%	+1.16%
9	10	↑	SQL	1.94%	+0.50%
10	11	↑	Fortran	1.78%	+0.49%
11	15	⇈	Delphi/Object Pascal	1.77%	+0.75%
12	13	↑	MATLAB	1.47%	+0.28%
13	8	⇊	PHP	1.46%	-0.09%
14	17	↑	Rust	1.32%	+0.35%
15	18	↑	R	1.20%	+0.23%
16	19	↑	Ruby	1.13%	+0.18%
17	14	↓	Scratch	1.11%	+0.03%
18	20	↑	Kotlin	1.10%	+0.20%
19	21	↑	COBOL	1.09%	+0.22%
20	16	⇊	Swift	1.08%	+0.09%



Langage C

- Top 10 des langages de programmation
  - Classement Institute of Electrical and Electronics Engineers (IEEE) - 2021

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	C		94.7
4	C++		92.4
5	JavaScript		88.1
6	C#		82.4
7	R		81.7
8	Go		77.7
9	HTML		75.4
10	Swift		70.4

Language Ranking: Jobs

Rank	Language	Type	Score
1	Python		100.0
2	C		96.0
3	Java		95.9
4	JavaScript		89.6
5	C++		88.3
6	Go		87.3
7	R		85.7
8	HTML		81.3
9	C#		79.8
10	SQL		71.9

Classement dans le cadre du développement d'applications mobiles

Rank	Language	Type	Score
1	Java		95.4
2	C		94.7
3	C++		92.4
4	C#		82.4
5	Swift		70.4
6	Dart		67.7
7	Kotlin		58.5
8	Scala		55.4
9	Objective-C		44.4
10	Delphi		37.8

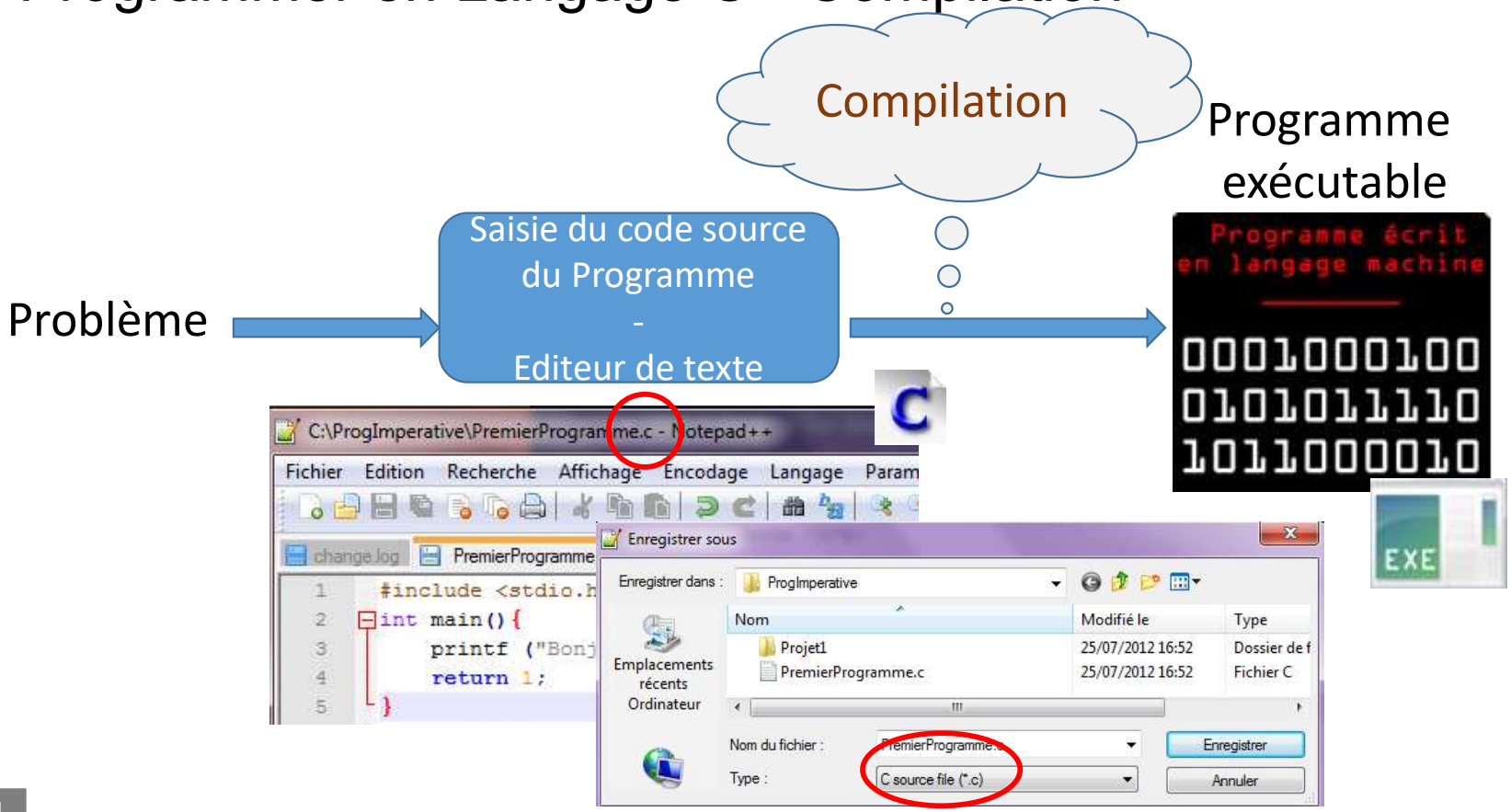
https://programmation.developpez.com/

SOMMAIRE

- Informations pratiques
- Introduction
- Eléments de base
  - Programmer en Langage C – Compilation
  - Structure d'un programme / Règles d'écritures
  - Types de base
  - Constantes/Variables
  - Opérateurs
  - Instructions de contrôle
  - Pointeurs
  - Tableaux
- Fonctions
- Chaînes de caractères
- Pointeurs- Tableaux-Fonctions
- Types Construits
- Entrées – Sorties sur Fichiers
- Compilation séparée
- Implémentation de Types Abstraits de Données

Eléments de base

- Programmer en Langage C - Compilation



Programmer en Langage C - Compilation

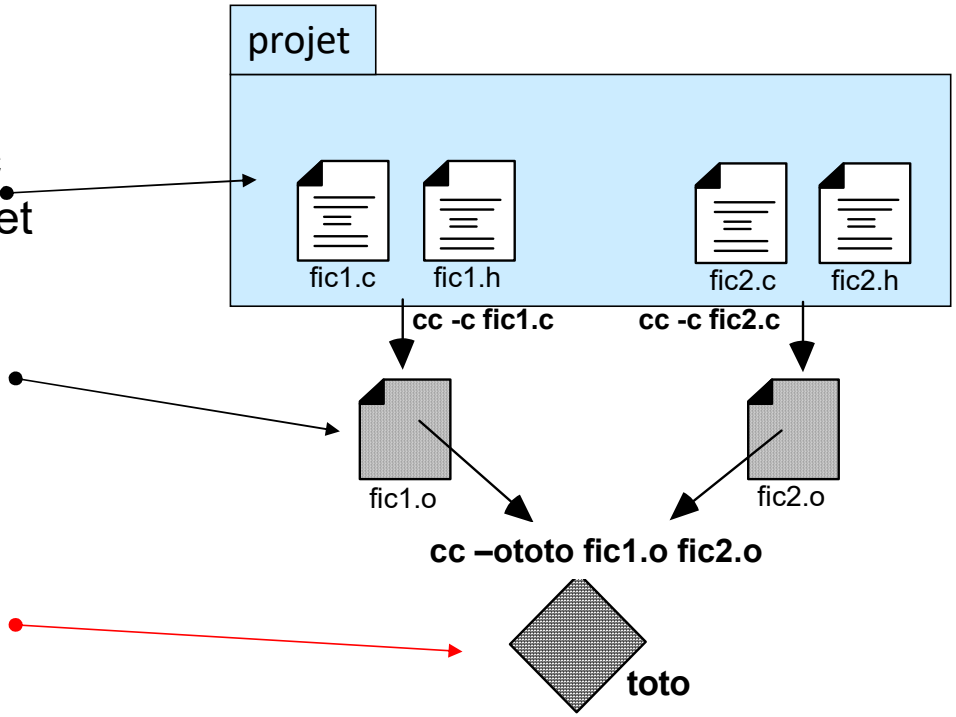
- Programmer en Langage C
  - Une application peut être conçue comme une collection de modules,
  - chaque module pouvant être
    - une collection de sous-programmes,
    - une collection de données partagées.
  - De tels modules peuvent constituer des unités de programmes en C chacune mémorisée dans un fichier.
  - Une unité de programme constitue une unité de compilation ou projet.

Programmer en Langage C - Compilation

- Compilation (3 étapes)
  - phase de pré-processing des fichiers *.c
    - inclusion des fichiers d'en-tête (*.h) et les remplacements
  - compilation des fichiers *.c
    - obtention des fichiers objets

    fichier objet => *.o
  - édition des liens entre les différentes librairies C et les fichiers objets
    - obtention du fichier exécutable

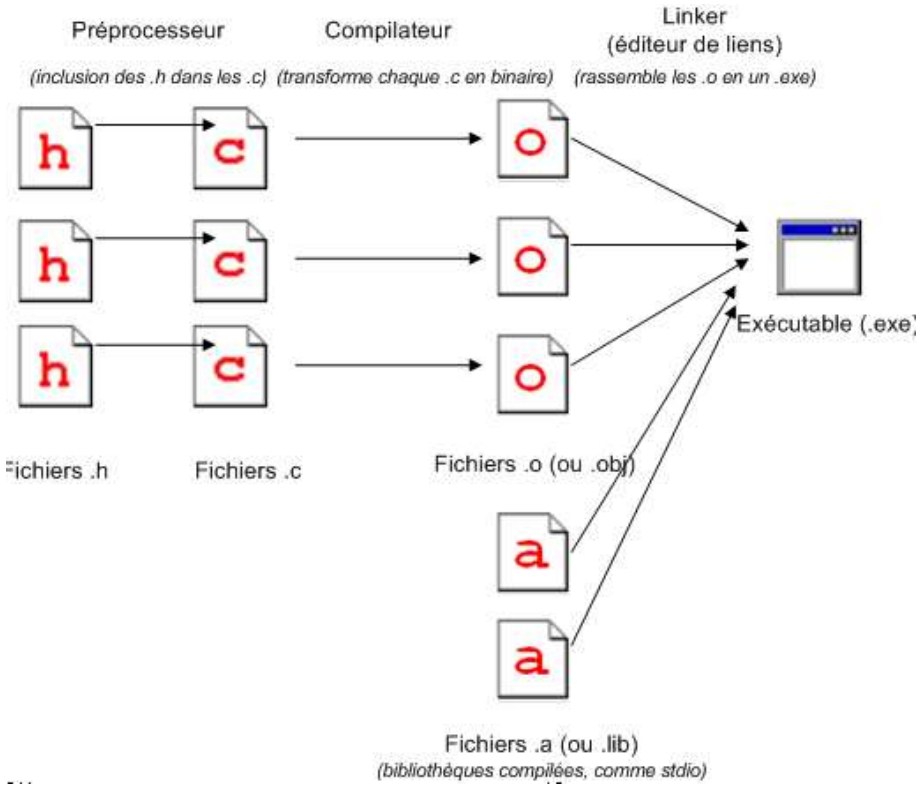
      .exe sous windows



Exemple des phases de compilation et d'édition de liens avec le compilateur cc sous UNIX.

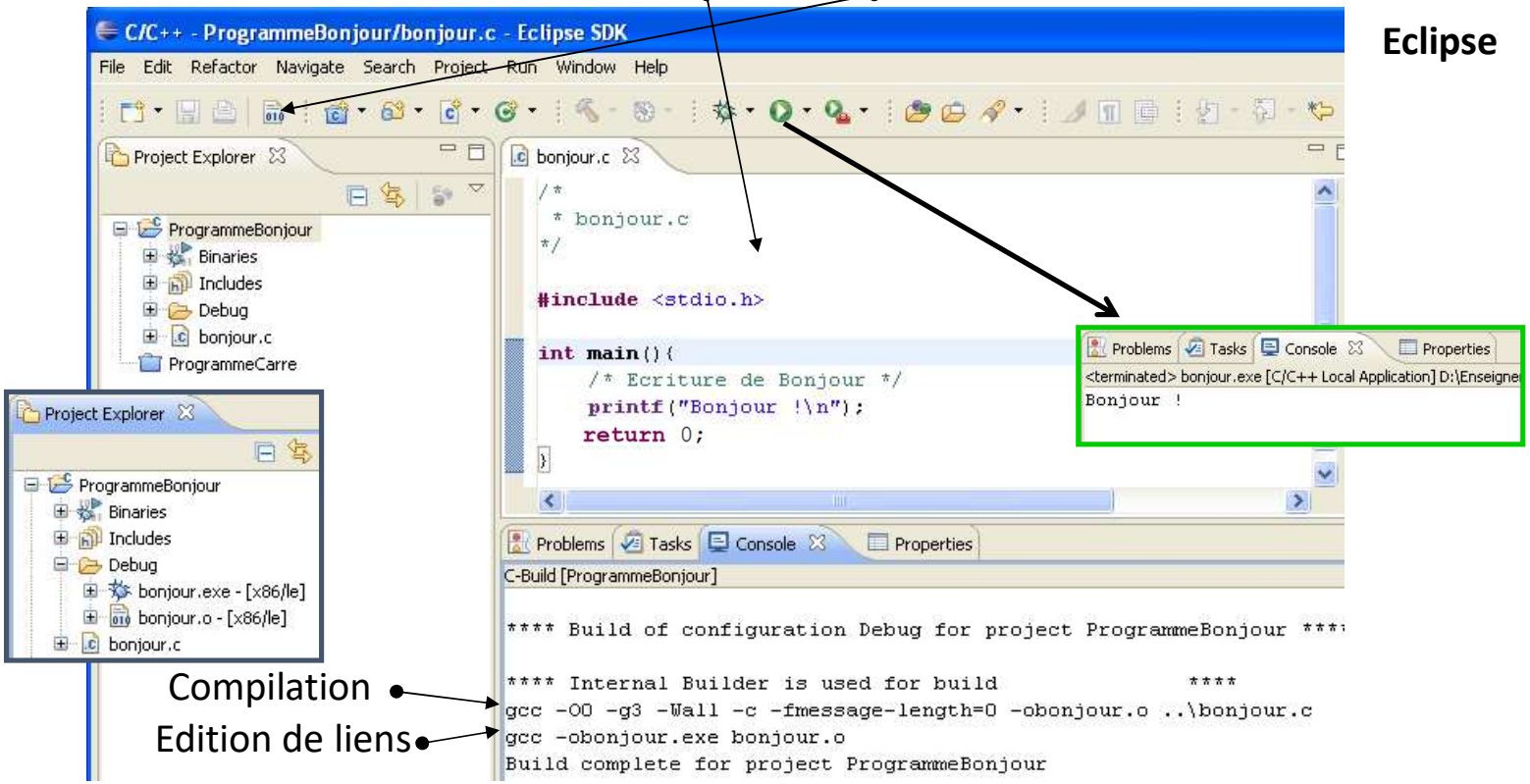
Programmer en Langage C - Compilation

- Pour être plus complet ….



Programmer en Langage C - Compilation

- Integrated Development Environment (IDE)
  - Programme 3 en 1 (éditeur de texte, compilateur–éditeur de liens, debugger)



Eclipse

Structure d'un programme en Langage C

monfichier.c

```
#include <fichier d'entête>
#define  constante   valeur
```

Directives de compilation du préprocesseur

```
typedef  struct duo {
                  int  a, b;
                }tduo;
```

Définitions des types

```
extern  int  VarGlobaleExt;
```

Importation des fonctions et variables externes

```
typefctl fonction1(int, char);
typefct2 fonction2(int, int);
```

Prototypage des fonctions internes

```
int   VarGlobaleInt;
```

Définitions des variables globales au fichier

```
typefctl fonction1(int a, char b)
{


  return(variable1);
}
```

Définitions des fonctions

```
typefct2 fonction2(int a, b)
{


    return(variable2);
}
```

Définitions des fonctions

```
int main()
{
  int   a, b;

  a = fonction1(2,'a');

  b = fonction2(2,3);
  return 0;
}
```

Définition de la fonction main et appel d'autres fonctions.

Règle de base

- Toute instruction se termine par un **;**
- Un bloc d'instruction commence par **{** et se termine par **}**
- Un commentaire commence par **/*** et se termine par ***/**
  - Peut être écrit sur plusieurs lignes entre /* et */
  - Commentaire sur une ligne **// si compilateur C/C++**
- Caractères autorisés pour les noms d'identificateurs
  - caractère ou caractère+chiffres mais **caractère en 1er**(taille < 31)
  - pas de caractères d'espacement, pas d'accents
    - le perimetre le_perimètre => le_perimetre
  - C fait la différence entre majuscule et Minuscule ⇔ sensible à la casse
    toto est différent de Toto , TOto, TOTO, toTO, tOto …..
  - Donner des noms qui indiquent le rôle de la variable ou de la fonction
  - Mots clés et opérateurs réservés ne peuvent être utilisés comme noms de variables, constantes, fonctions, ….

Règles d'écriture pour une bonne lisibilité

- Une et une seule instruction par ligne
- L'accolade fermante d'un bloc est alignée sur l'accolade ouvrante correspondante
- Les instructions d'un bloc sont décalées par rapport aux accolades marquant le début et la fin du bloc
- Deux styles de convention d'écriture autorisés

```
int  main( )
{
   printf("bonjour ") ;  /* affichage d'un
                            message à l'écran
                         */
    printf(" à tous \n") ;
   return 0 ;
}
```

```
int  main( ) {
   printf("bonjour ") ;  // affichage
   printf(" à tous \n") ;
   return 0 ;
}
```

Types de base

- Donnent l'étendue des valeurs du domaine concerné
  - Taille des types représentant des valeurs entières ou réelles est définie sur chaque machine
    - Pour types représentant un entier
      définies dans limits.h
    - Pour type représentant un réel
      définies dans float.h
  - Pas de type booléen en C

Types de base

- Valeurs entières

Type	Octets Architecture 16 bits Architecture 32 bits	Etendue de valeurs
char	1 1	-128 -> +127
short	2 2	-32768 -> +32767
int	2 4	-32768 -> +32767 -2 147 483 648 -> 2 147 483 647
long	4 4	-2 147 483 648 -> 2 147 483 647

Types de base

- Valeurs entières + mot clé unsigned

Type	Octets Architecture 16 bits Architecture 32 bits	Etendue de valeurs
char	1 1	0 -> 255
short	2 2	0 -> 65535
int	2 4	0 -> 65535 0 -> 4 294 967 295
long	4 4	0 -> 4 294 967 295

Types de base

- char ⇔ valeur entière ?
  - **Table de code ascii**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Types de base

- Valeurs réelles

Représentation d'un réel : <+/-> <mantisse> *10 <exposant> $1.458791 * 10^4 \Leftrightarrow 1\,4587.91$

mantisse: un décimal positif avec un seul chiffre devant la virgule

Exposant: un entier donnant l'ordre de grandeur

Type	Octets Architecture 32 bits	Etendue de valeurs
float format simple précision	4 (32 bits) signe: 1 bit exposant : 8 bits mantisse : 23 bits	-3.4×10^{38} -> 3.4×10^{38}
double format double précision	8 (64 bits) signe: 1 bit exposant : 11 bits mantisse : 52 bits	-1.7×10^{308} -> 1.7×10^{308}
long double format précision étendue	10 (80 bits) signe: 1 bit exposant : 15 bits mantisse : 64 bits	

Constante / Variables

- Constante
  - Ne change jamais de valeur pendant l'exécution d'un programme
  - Constante non typée
    - Macrodéfinition

```
#define NB 5
#define PI 3.14
```

⇨ pas d'allocation en mémoire

⇨ le pré-processeur remplace dans la suite du programme toute référence aux macros définies par leur définition

⇨ le pré-processeur se contente de remplacer les références à la macro par la valeur de celle-ci sans opérer de contrôle sur l 'utilisation de cette valeur

Constante / Variables

- Constante non typée : illustration

TestConst.c

```c
/*
 ============================================================
 Name        : TestConst.c
 ============================================================
 */

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

#define NB_VIES_INITIALES 5

int main(void) {
    int i;
    printf("Nombre de vies initiales: %d\n", NB_VIES_INITIALES);
    for(i=NB_VIES_INITIALES;i>0;i--)
        printf("Il vous reste %d vie(s)\n", i);
    return EXIT_SUCCESS;
}
```

Properties | Problems | Tasks | Console

<terminated> TestConst.exe [C/C++ Application] C:\User

```
Nombre de vies initiales: 5
Il vous reste 5 vie(s)
Il vous reste 4 vie(s)
Il vous reste 3 vie(s)
Il vous reste 2 vie(s)
Il vous reste 1 vie(s)
```

8

Properties | Problems | Tasks | Console

<terminated> TestConst.exe [C/C++ Application] C:\User

```
Nombre de vies initiales: 8
Il vous reste 8 vie(s)
Il vous reste 7 vie(s)
Il vous reste 6 vie(s)
Il vous reste 5 vie(s)
Il vous reste 4 vie(s)
Il vous reste 3 vie(s)
Il vous reste 2 vie(s)
Il vous reste 1 vie(s)
```

Constante / Variables

- Constante typée
  - mot clé const ⇨ const Type NOM_CONSTANTE = ValeurConstante ;

```
const   int     NOMBRE_DE_VIES_INITIALES = 5;
const   float   PI = 3.14;
```

  - Allocation en mémoire du nombre d'octets correspondant au type utilisé pour la déclaration

Constante / Variables

- Constante typée : illustration

TestConst.c

```
/*
 ============================================================
 Name        : TestConstTypee.c
 ============================================================
 */

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

int main(void) {
    const int NB_VIES_INITIALES=5;
    int i;
    printf("Nombre de vies initiales: %d\n", NB_VIES_INITIALES);
    for(i=NB_VIES_INITIALES;i>0;i--)
        printf("Il vous reste %d vie(s)\n", i);
    return EXIT_SUCCESS;
}
```

8

Properties | Problems | Tasks | Console

<terminated> TestConst.exe [C/C++ Application] C:\User

```
Nombre de vies initiales: 5
Il vous reste 5 vie(s)
Il vous reste 4 vie(s)
Il vous reste 3 vie(s)
Il vous reste 2 vie(s)
Il vous reste 1 vie(s)
```

Properties | Problems | Tasks | Console

<terminated> TestConst.exe [C/C++ Application] C:\User

```
Nombre de vies initiales: 8
Il vous reste 8 vie(s)
Il vous reste 7 vie(s)
Il vous reste 6 vie(s)
Il vous reste 5 vie(s)
Il vous reste 4 vie(s)
Il vous reste 3 vie(s)
Il vous reste 2 vie(s)
Il vous reste 1 vie(s)
```

Constante / Variables

- Constante typée : illustration

TestConst.c

```
/*
 ============================================================
 Name        : TestConstTypee.c
 ============================================================
 */

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

int main(void) {
    const float PI=3.14;

    printf("Surface d'un disque de 4 cm de rayon : %.2f\n", PI*4*4);
}
```

Properties | Problems | Tasks | Console

```
<terminated> TestConst.exe [C/C++ Application] C:\Users\fl
Surface d'un disque de 4 cm de rayon : 50.24
```

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

int main(void) {
    const float PI=3.14;

    printf("Surface d'un disque de 4 cm de rayon : %d\n", PI*4*4);

    return EXIT_SUCCESS;
}
```

format '%d' expects type 'int', but argument 2 has type 'double'

Constante / Variables

- Variables
  - Peut changer de valeur pendant l'exécution du programme
  - Définition de variable
    - Déclaration + réservation de l'espace mémoire
    - liée à
      - La définition du domaine de valeurs et des opérations autorisées

Constante / Variables

- Variables
  - Déclaration de la variable

  ⇔ association nom de la variable et du type
    - Type de la variable
      - char, int, short, float, double …
    - Classe mémoire
      - Globale/locale

      ⇔ Lieu de définition/déclaration
        - En dehors d'une fonction
        - Dans une fonction
      - Qualificatifs
        - register, static, extern
      - Association d'une durée de vie

Constante / Variables

- Variables
  - Initialisation de la variable

  ⇔ affectation d'une valeur initiale

    - Au moment de la définition
    - Dans une instruction séparée

Constante / Variables

- Où et quand définir/déclarer une variable?
  - **AVANT de l'utiliser**
  - En dehors du corps des fonctions

    ⇒ **variable globale**
    ⇒ **visible dans toutes les fonctions** du fichier où elle est déclarée
    ⇒ variable est **allouée en début de programme** et est **détruite à la fin du programme**
    ⇒ automatiquement **initialisée à 0**

  - Dans le corps d'une fonction ou au début d' un bloc

    ⇒ **variable locale** ou variable automatique
    ⇒ **visible uniquement dans le bloc ou la fonction** où elle est déclarée et **dans les sous-blocs de celui-ci ou de celle-ci**
    ⇒ variable est **allouée à chaque entrée** dans le bloc et elle est **détruite à la sortie du bloc**
    ⇒ perte de sa valeur à la sortie du bloc
    ⇒ Pas d'initialisation automatique
    valeur initiale doit être **explicitement définie par le programmeur**

☝ **Une variable locale qui a le même nom qu'une variable globale masque la variable globale dans la zone de visibilité de la variable locale**

Constante / Variables

- Domaine de visibilité d'une variable

```c
/*
 ============================================
 Name        : TestVariable.c
 ============================================
 */
#include <stdio.h>
#include <stdlib.h>

int nb;


int main(void) {
    printf("Nb avant affectation: %d\n", nb);
    nb=167;
    printf("Nb après affectation: %d\n", nb);
    return EXIT_SUCCESS;
}
```

Variable Globale

```c
/*
 ============================================
 Name        : TestVariable.c
 ============================================
 */
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int i;
    for(i=0;i<3;i++){
        float rayon=5.1 +i;
        printf("rayon  %.2f\n", rayon);
    }
    printf("i: %d",i);
    printf("rayon  %.2f\n", rayon);
    return EXIT_SUCCESS;
}
```

Variable Locale
Visible dans la fonction main

Variable Locale
Visible dans le bloc for

Multiple markers at this line
- each undeclared identifier is reported only once for each function it appears in
- 'rayon' undeclared (first use in this function)
- Symbol 'rayon' could not be resolved

```c
int main(void) {
    int i;
    for(i=0;i<3;i++){
        float rayon=5.1 +i;
        printf("rayon  %.2f\n", rayon);
    }
    printf("i: %d",i);
    return EXIT_SUCCESS;
}
```

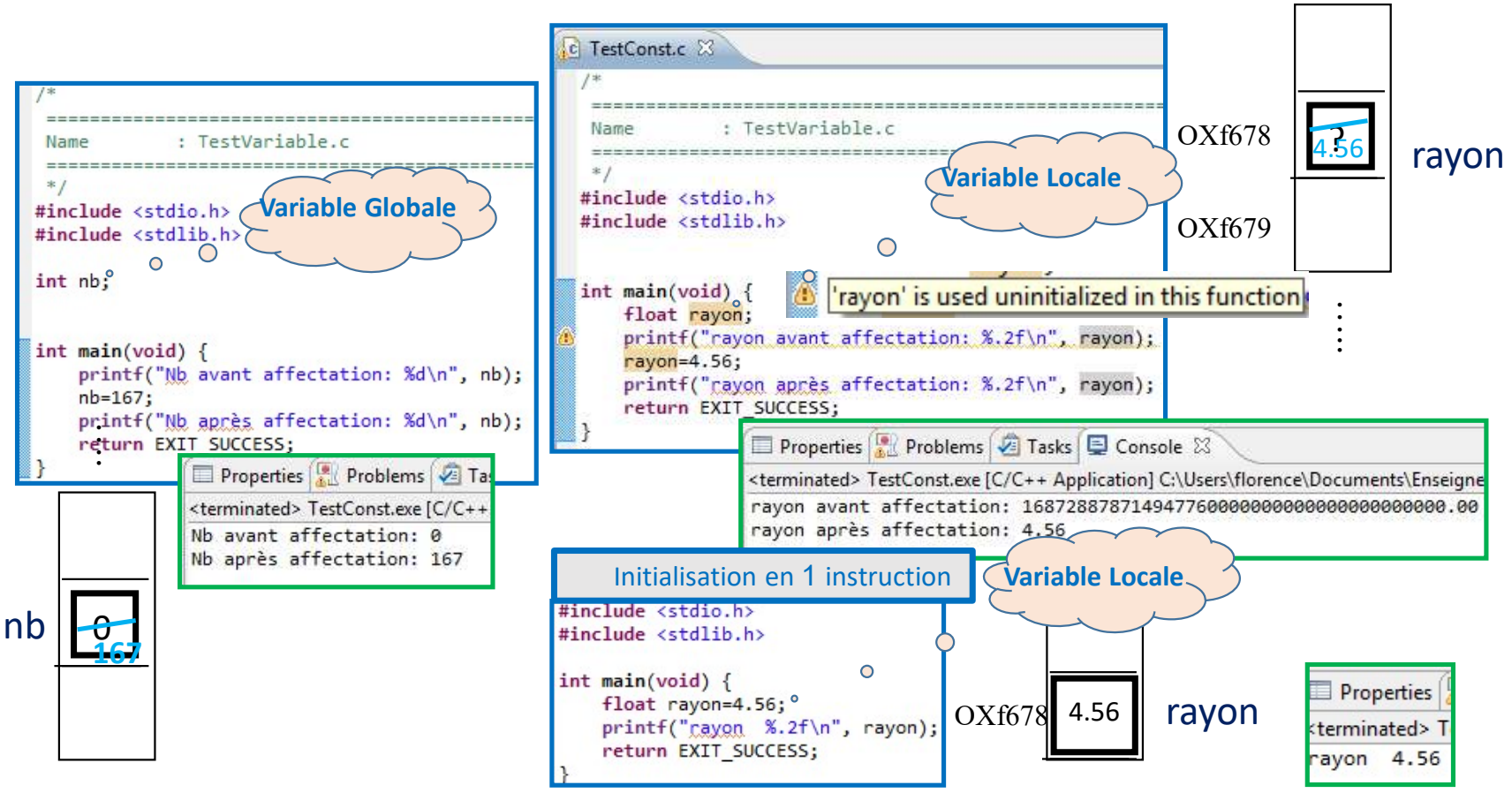
Properties

<terminated> T

```
rayon  5.10
rayon  6.10
rayon  7.10
i: 3
```

Constante / Variables

- Définition d'une variable / Initialisation



Constante / Variables

- Classes de variables
  - Mots clés ajoutés devant la définition/déclaration de la variable
    - **extern**
      - Mot clé utilisé pour des variables globales déclarées dans d'autres fichiers
    - **register**
      - variable de type registre est stockée dans un registre
        ⇒ mémoire à accès plus rapide que la RAM
        ⇒ accélération des traitements mais on ne peut pas accéder à l 'adresse mémoire de la variable
    - **static**
      - Appliqué à une variable globale
        ⇒ variable sera invisible dans les autres fichiers
      - Appliqué à une variable locale à une fonction
        ⇒ la variable (automatiquement initialisée à 0) n'est pas détruite à la fin de l 'appel de la fonction
        ⇒ elle conserve sa valeur entre 2 appels de fonctions
        ⇒ Variable rémanente

Constante / Variables

- Classes de variables

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

int somme(int a){
    int s=0;
    s=s+a;
    return s;
}
int main(){
    int i,x;

    for(i=0;i<4;i++){
        printf("Donnez la valeur de x\n");
        scanf("%d",&x);
        printf("La somme des nombres entrés est %d\n", somme(x));
    }
    return EXIT_SUCCESS;
}
```

Exemple d'exécution

i	valeur rentrée	valeur affichée
0	5	5
1	4	4
2	10	10
3	3	3

Constante / Variables

- Classes de variables

```c
#include <stdio.h>
#include <stdlib.h>
int somme(int a){
    int s=0;
    s=s+a;
    return s;
}
int main(){
    int i,x;

    for(i=0;i<4;i++){
        printf("Donnez la valeur de x\n");
        scanf("%d",&x);
        printf("La somme des nombres entrés est %d\n", somme(x));
    }
    return EXIT_SUCCESS;
}
```

```c
int somme(int a){
 static int s ;
  s=s+a ;
  return(s) ;
}
```

Exemple d'exécution

i	valeur rentrée	valeur affichée
0	5	5
1	4	9
2	10	19
3	3	22