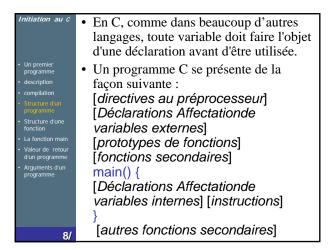
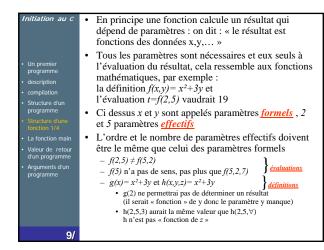
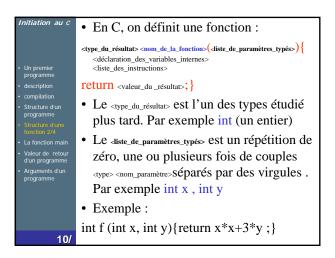


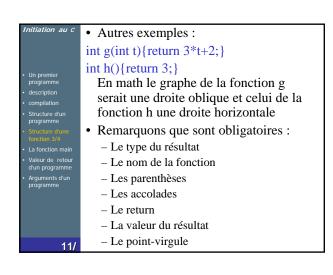
Compilation en 4 phases (-E -S -c):

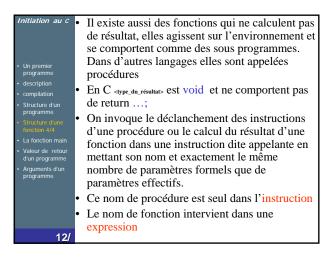
- Le traitement par le preprocesseur : il effectue des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers, etc.)
- La compilation : le fichier engendré par le pré-processeur est traduit en assembleur
- L'assemblage : transforme le code assembleur en un fichier objet (instructions compréhensibles par le processeur)
- L'édition de liens: Une fois le code source assemblé, il faut lier entre eux les différents fichiers objets (bibliothèques,

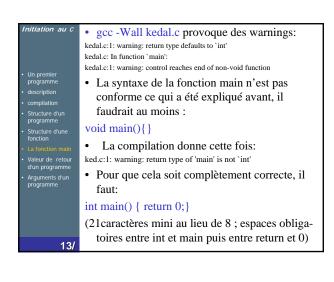


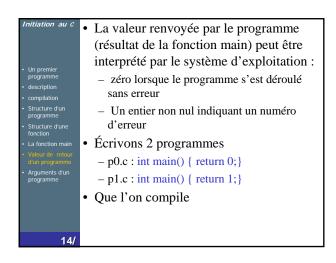


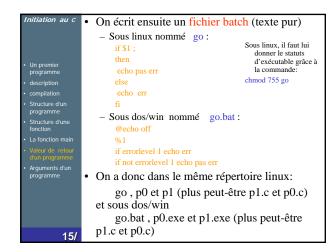


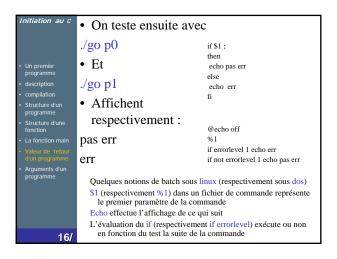


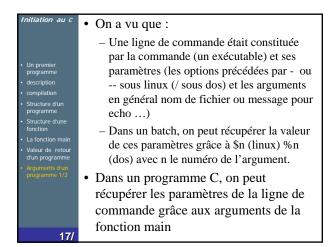


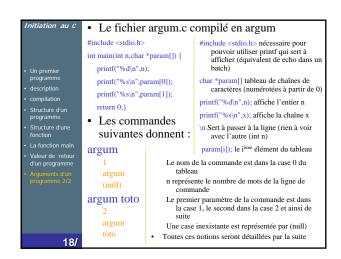


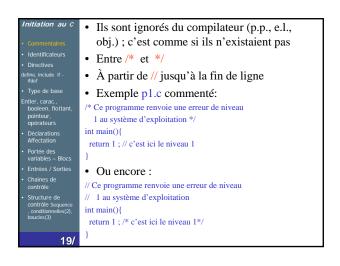


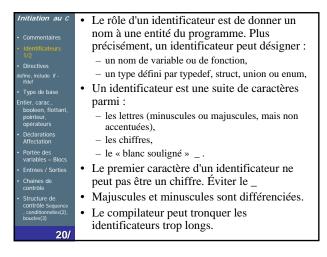












Commentaires

Identificateurs

ZZ

Directives

Identificateurs

ZZ

Directives

Identificateurs

Type de base

Entier, carac.,

booleen, flottant,

pointeur,

poerateurs

Déclaration

Affectation

Portée des

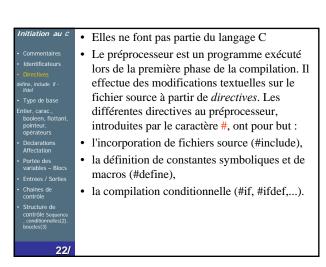
variables – Blocs

Entrées / Sorties

Chaines de

• Comme le S.E. linux Gcc est différent de gcc

- Exemples: var1, tab_23 ou _deb sont des identificateurs valides; par contre, 2i et i:j ne le sont pas
- Un certain nombre de mots, appelés *mots-clefs*, sont réservés pour le langage lui-même et ne peuvent pas être utilisés comme identificateurs. L'ANSI C compte 32 mots clefs (en minuscule):
- auto, const, double, float, int, short, struct, unsigned, break, continue, else, for, long, signed, switch, void, case, default, enum, goto, register, sizeof, typedef, volatile, char, do, extern, if, return, static, union, while



• La directive #define permet de définir : - des constantes symboliques · #define nom reste-de-la-ligne demande au préprocesseur de substituer toute occurence de nom par la chaîne de caractères reste-de-la-ligne dans la suite du fichier source. Son utilité principale est de donner un nom parlant à une constante, qui pourra être aisément modifiée. Par exemple : #define NB_LIGNES 10 #define NB_COLONNES 33 #define TAILLE_MATRICE NB_LIGNES * **NB COLONNES** • Il n'y a toutefois aucune contrainte sur la chaîne de caractères reste-de-la-ligne. On peut écrire #define BEGIN { #define END } des macros avec paramètres. DANGER #define CARRE(a) a * a le préprocesseur remplacera CARRE(a + b) par a + b * a + b et non par (a + b) * (a + b)#define CARRE (a) a * a la chaîne de caractères CARRE(2) sera remplacée par (a) a * a (2) espace entre CARRE et

• On peut définir des constantes

- commentaires

- identificateurs

- Directives

- entières (décimales, octales ou hexadécimales)

1234, 0243, 0x4D

- Réelles (double précision par défaut)

12.34, 34.5e-2

- Caractères (entre apostrophes)

- Octaire des variables - Blocs

- Entrées / Sorties

- Chaînes de caractères (entre guillemets)

""c'est"" une \n chaine\""

• Exemple

#define toto "\"c'est"

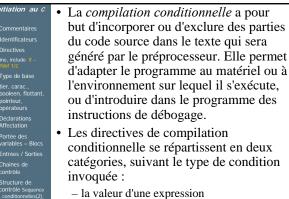
#define tutu " une \n chaine\""

#define tit tutu toto"xxx"

#define nombre 0243+0x4D // 163 + 77



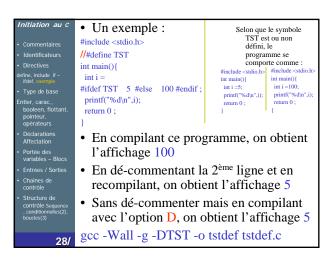
- Elle permet d'incorporer dans le fichier source le texte figurant dans un autre fichier(recopié par le préprocesseur). Ce dernier peut être un fichier en-tête de la librairie standard (stdio.h, math.h,...) ou n'importe quel autre fichier.
- La directive #include possède deux syntaxes voisines :
 - #include <nom-de-fichier> recherche le fichier mentionné dans un ou plusieurs répertoires systèmes définis par l'implémentation (par exemple, /usr/include/);
 - #include "nom-de-fichier" recherche le fichier dans le répertoire courant (celui où se trouve le fichier source). On peut spécifier d'autres répertoires à l'aide de l'option -I du compilateur.
- La première syntaxe est généralement utilisée pour les fichiers en-tête de la librairie standard, tandis que la seconde est plutôt destinée aux fichiers créés par l'utilisateur.

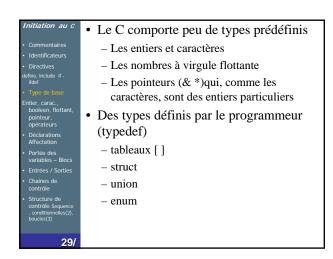


- l'avistance au l'inevistan
 - l'existence ou l'inexistence de symboles.

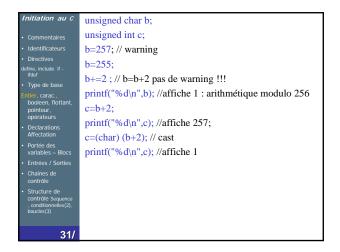


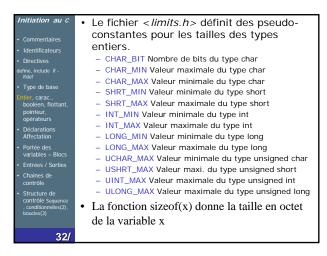
qui est compilée

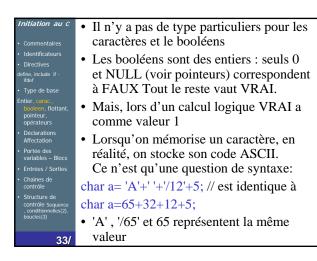


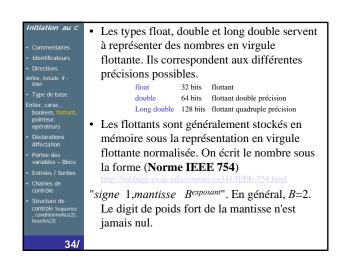


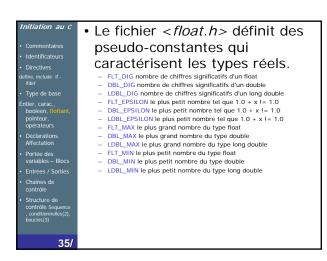
Initiation au C	Déclaration d'entier				
CommentairesIdentificateursDirectives	Туре	nom	Taille en bits	unsigned	signed
define, include if - ifdef Type de base	char	caractère	8	0255	-128127
Entier, carac., booleen, flottant, pointeur,	short	entier court	16	065535	-32768 32767
opérateurs • Déclarations Affectation	int	entier	32	0 4 294 967 296	-2 147 483 648 2 147 483 647
Portée des variables – Blocs	long	entier long	32	0 4 294 967 296	-2 147 483 648 2 147 483 647
 Entrées / Sorties Chaines de contrôle 	long long	entier non ansi	64	0 18 446 744 073 709 551 616	- 9 223 372 036 854 775 808 9 223 372 036 854 775 807
Structure de contrôle Sequence , conditionnelles(2), boucles(3)	• Exemple: unsigned long k=123456789; // déclarations				
30/	unsigned char i=123; // et initialisations				

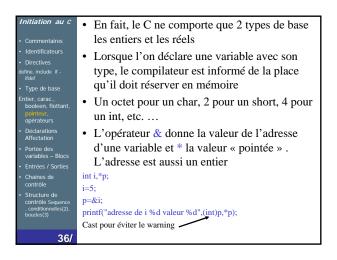












```
int main(){
unsigned int a,*n;
                                                                            Tailles: char->1 short->2 int->4 long->4 long long->8
                unsigned short *p,*q;
                                                                            12345678=305419896
                  unsigned char *r. *s
                  a=0x12345678
                                                                            5678
                                                                            1234
                 p=(short *)n;
                 q=p+1;
r=(char*)q;
                                                                            12
                                                                            12ab5678
                 s=r+1;
                    printf("Tailles: char->%d short->%d int->%d long->%d long long->%d \n", sizeof(char),sizeof(short),sizeof(int),sizeof(long),sizeof(long long));
printf("%x=%d\n",a,a);
                     printf("%x\n",*p);
                     printf("%x\n",*q);
printf("%x\n",*q);
printf("%x\n",*r);
printf("%x\n",*s);
                                                                            240ffa0
                                                                            240ffa2
                                                                            240ffa2
                      *r=0xAB:
                     printf("%x\n",a);
printf("%x\n",(unsigned int)p);
                     printf("%x\n",(unsigned int)q);
                                                                          Représentation de la mémoire ?
                     printf("%x\n",(unsigned int)r);
printf("%x\n",(unsigned int)s);
                                                                           → little indian (inversion
                                                                           d'octets)
37/
```

```
#include <stdio.h>
int main(){
    int x=3, *y,*z;
    y=&x;
    idefine, include if -
    ided
    ifded
    ifded
    int y=3, *y,*z;
    y=&x;
    z=y+1;
    printf("z-y=%d\n",(int)(z-y));
    printf("z-y=%d\n",(int)z-(int)y);
    return 0;
}

Afficiation
Portée des
    variables – Blocs
Entires / Sorties
Obcident flottant,
poliritatir,
poferateurs
Declarations
Afficiation
Portée des
variables – Blocs
Chaines de
controle sequence,
conditionnelles(2),
boucdes(3)

* Affiche

z-y=1

z-y=4

* Les pointeurs sont, en général, typés

* Il est possible de définir des pointeurs non
typés

* void *p;
```

```
Initiation au c

Commentaires
Identificateurs
Identificateurs
Int i=5,*p;
char *q;
void *r;
fddr
It gelfider
It g
```

```
#include <stdio.h>
                int main(){
                  int x=3:
                  int *y;
Directives
                  printf("&x=%p, *y=%p\n",&x,y);
                  printf("x=%d, *y=%d\n",x,*y); // Ne pas utiliser *y non affecté
                  y=&x;
                  printf("x=%d, *y=%d\n",x,*y);
                   *v=2:
                  printf("x=%d , *y=%d\n",*&x,*y);
                return 0;
                                                  · Affiche sous Linux :
                  Affiche sous dos:
               &x=0240FFA0, *y=004012C6
                                                  &x=0xbfffe3d4, *y=(nil)
                                                  Segmentation fault
               x=3, *y=-1979949685
                x=3, *y=3
                                                  Arrêt sur erreur
                x=2, *y=2
         40/
```

```
nitiation au C
               • Utilisation normale des pointeurs :
               #include <stdio.h>
               #include <stdlib.h>
               int main(){
                 int *y;
                 y= (int *) malloc(sizeof(int));
                 *v=5;
                 printf("&y=%x , y=%x , *y=%d",(int)&y,(int)y,*y);
                 return 0;
               • Affiche:
               &y=bffff2d4 , y=804a008 , *y=5
                 La valeur de y dépend de l'emplacement trouvé par
                 le système d'exploitation elle est susceptible de
                 changer à chaque exécution.
               · « géographie de la mémoire »
```

```
nitiation au C
                     Le tableau suivant classe les opérateurs par ordres de
                     priorité décroissants. Les opérateurs placés sur une
même ligne ont même priorité. Si dans une expression
                     figurent plusieurs opérateurs de même priorité, l'ordre d'évaluation est de gauche à droite. On préférera
                     toutefois mettre des parenthèses en cas de doute...
                      − () ∏ ->
                       - sizeof &(adresse) *(indirection) (type) -(unaire) ++ ~!
                      - * / %
                      - + -(binaire)
                      - << >>
                      - < <= > >=
                      - == !=
                      - &(et bit-à-bit)
                      - &&
                      - ||
                      - >>= <<= |= ^= &= %= /= *= -= += =
```

```
    Commentaires
    Identificateurs
    Identificateurs
    Directives
    Identificateurs
    Directives
    Inge de base
    Entificateurs
    Dooleen, flottant, pointeur, operateurs
    Dooleen, flottant, pointeur, operateurs
    Portee des variables – Blocs
    Entrées / Sorties
    Chaines de controle
    Structure de controle
    Structure de controle Sequence conditionnelies(2)
    boudes(2)
    44/
    Chaines de controle Sequence conditionnelies(2)
    boudes(2)
    A4/
    Chaines de controle sequence conditionnelies(2)
    Entrées / Sorties
    Lorsque plusieurs variables sont de même type, ainsi que des pointeurs de variables de ce type, il est possible de les déclarer sur la même ligne
```

```
• Syntaxe :  
• Commentaires  
• Identificateurs  
• Directives  
• Composition  
• Type de base  
• Entier, carac, booleen, flottant, pointeur, operateurs  
• Doctarations  
• Portee des variables – Blocs  
• Entrées / Sorties  
• Chaines de contrôle Sequence conditionalies (2): bouldes (3)  

45/

• Syntaxe :  
• (type) < (nom) [= < valeur>] [, <nom> [= < valeur>] ]*;  
• Exemple  
• int a,b=5,*c,*d,**e,f=g=7;  
• e est un pointeur sur un pointeur d'entier : Il faut 2 allocations  
• e= (int **) malloc(sizeof(int *));  
* e= (int *) malloc(sizeof(int));  
* e= (int *) malloc(sizeof(int));  
* e= 12;  
printf("e=%d, *e=%d, **e=%d\n", (int))e, (int)*e, **e);
```

```
• Les variables globales sont déclarées et
   éventuellement initialisées avant
#include <stdio.h>
int f(int x){
  return x+1; // +y \rightarrow l'utilisation de y ici provoquerait une erreur
         // y serait déclaré trop tard
int v=6:
int main(){
  int x=3,z=5;
  { int x=9;
     printf("f(%d)=%d , x=%d , z=%d\n",y,f(y),x,z);
  printf("f(%d)=%d, x=%d, z=%d\n",y,f(y),x,z);
  return 0;
      • Affiche:
                              · La variable utilisée est
                                 celle qui est déclarée dans
       f(6)=7, x=9, z=5
       f(6)=7, x=3, z=5
                                 le bloc le plus interne
```

```
• Utiliser #include <stdio.h>
           • printf : Elle permet l'écriture formatée sur le flux
             standard de sortie stdout (l'écran par défaut).
int printf( const char *format [, arg [, arg]...]);

    puts : Elle permet d'écrire, à la position courante, la
chaîne de caractères pointée par s sur la sortie

                              int puts(const char *s);
             standard.
           • putchar (macro) : Elle permet d'écrire, à la position
             courante, le caractère c sur la sortie standard.
             putchar(int c);
• Entrées de caractères

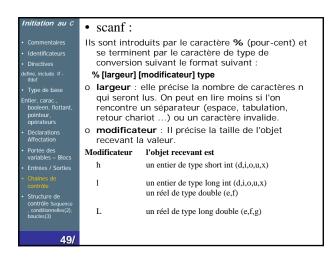
    scanf int scanf( const char *format [, pointer[, / ]

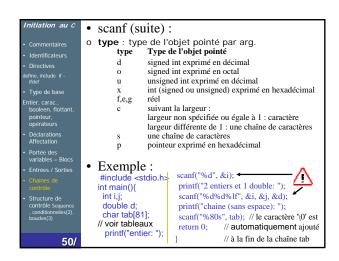
             pointer]...]);
          • gets
                                char *gets(char *s);

    getchar (macro)

                                          int getchar();
            sscanf et sprintf : sont analogue à scanf (printf), sauf qu'à la place de lire l'entrée (sortie) standard, elle lit
             (écrit) la chaîne d'adresse s.
          int sscanf(char *s, char *format, pointer ...);
          int sprintf(char *s, char *format, arg ...);
```

```
· puts et gets:
#include <stdio.h>
int main{
 char ligne[10]; // voir tableaux
 puts("Votre texte 1 ? "):
 gets(ligne);
 return 0:
/* résultat de l'exécution ----
Votre texte 1 ? qqqqqqqqqqqqqqqqqqqqqqqqqqqqq
   Segmentation fault (core dumped)
· putchar et getchar:
#include <stdio.h>
int main(){
char c
 while ((c= getchar())!=EOF) putchar(c); // voir Structure de contrôle
 return 0;
putgetchar chaque ligne tapée est dupliquée(immédiatement) dernière ligne
```





Initiation au C	• printf:	
Commentaires	o Valeur retournée :le nombre d'octets effectivement	
 Identificateurs 	écrits ou la constante EOF (-1) en cas d'erreur.	
Directives	o Spécificateurs de format :	
define, include if -	% [drapeaux] [largeur] [.precision] [modificateur] type	
Type de base	o drape	aux :
Entier, carac	Drapeaux	Signification
booleen, flottant,	rien	justifié à droite et complété à gauche par des espaces
pointeur,	-	justifié à gauche et complété à droite par des espaces
opérateurs	+	les résultats commencent toujours par le signe + ou -
 Déclarations Affectation 	espace	le signe n'est affiché que pour les valeurs négatives
Portée des	#	forme alternative. Si le type de conversion est :
variables – Blocs		c,s,d,i,u : sans effet
 Entrées / Sorties 		o : un 0 sera placé devant la valeur
Chaines de		x, X : 0x ou 0X sera placé devant la valeur
contrôle		e, E, f : le point décimal sera toujours affiché
		g, G : même chose que e ou E, mais sans
contrôle Sequence , conditionnelles(2), boucles(3)		supprimer les zéros à droite
Doucles(3)		
51/		

Initiation au C	• printf (suite):		
Commentaires Identificateurs Directives define, include if -	O largeur: elle précise la nombre de caractères n qui seront affichés. Si la valeur à afficher dépasse la taille de la fenêtre ainsi définie, C utilise quand même la place nécessaire.		
Type de base	Largeur Effet sur l'affichage		
Entier, carac., booleen, flottant,	n affiche n caractères, complété éventuellement par des espaces		
pointeur, opérateurs	On affiche n caractères, complété éventuellement à gauche par des 0		
 Déclarations Affectation 	* l'argument suivant de la liste fournit la largeur		
 Portée des variables – Blocs Entrées / Sorties Chaines de 	O precision: elle précise pour: un entier, le nombre de chiffres à afficher un réel, le nombre de chiffres de la partie décimale à afficher (avec arrondi) les chaînes, le nombre maximum de caractères à afficher.		
Structure de contrôle Sequence , conditionnelles(2), boucles(3)			
52/	0		

Initiation au C	• printf (suite):		
Commentaires Identificateurs	o precision :		
Directives define, include if -	Precision Rien	Effet sur l'affichage précision par défaut :	
ifdef Type de base		d,i,o,u,x : 1 chiffre e, E, f : 6 chiffres pour la partie décimale.	
Entier, carac., booleen, flottant, pointeur.	0	d,i,o,u,x : précision par défaut e, E, f : pas de point décimal	
opérateurs	n	n caractères au plus	
Déclarations Affectation	*	l'argument suivant de la liste contient la précision	
 Portée des variables – Blocs 	o modificateur : II précise comment sera interprété		
Entrées / Sorties	l'argument		
	Modificateur	interprétation comme	
	h	un entier de type short (d,i,o,u,x,X)	
Structure de contrôle Sequence	I	un entier de type long (d,i,o,u,x,X)	
, conditionnelles(2), boucles(3)	L	un réel de type long double (e,E,f,g,G)	
53/			

Initiation au C	• printf (suite) :		
Commentaires	o type: type de conversion de l'argument		
 Identificateurs 	Type	Format de la sortie	
Directives	d ou i	entier décimal signé	
define, include if -	О	entier octal non signé	
Type de base	u	entier décimal non signé	
Entier, carac.,	X	entier hexadécimal non signé	
booleen, flottant,	Χ	entier hexadécimal non signé en majuscules	
opérateurs	f	réel de la forme [-]dddd.ddd	
 Déclarations 			
Affectation	e	réel de la forme [-]d.ddd e [+/-]ddd	
 Portée des variables – Blocs 	E	comme e mais l'exposant est la lettre	
Entrées / Sorties	g	format e ou f suivant la précision	
Chaines de	G	comme g mais l'exposant est la lettre E	
contrôle	С	caractère	
Structure de	S	affiche les caractères jusqu'au	
contrôle Sequence . conditionnelles(2).		caractère nul '\0' ou jusqu'à ce que	
, conditionnelles(2), boucles(3)		la précision soit atteinte	
	р	pointeur	
54/			

```
• printf (suite):
  Exemple d'utilisation des formats numériques :
   instruction C
                                         résultat
 printf("|%d|\n",12345);
                                 |12345|
 printf("|%+d|\n",12345);
                                         l+12345l
 printf("|%8d|\n",12345);
                                         | 12345|
 printf("|%8.6d|\n",12345);
                                          012345
 printf("|%x|\n",255);
                                          |ff|
 printf("|%X|\n",255);
                                 |FF|
printf("|%#x|\n",255);
                                 I0xffl
 printf("|%f|\n",1.23456789012345); |1.234568|
printf("|%10.4f|\n",1.23456789);
                                          11.2346
printf("Bonjour\n");
printf("Nombre %d prix %ld Total %9ld\n",nbre, prix, prix *
  nbre):
printf("%s est facile\n", chaine);
printf("%8.2Lf \n", result);
printf("%*.*Lf \n", 8, 2, result); /* equivalent a %8.2Lf */
```

```
Structure de contrôle:

Ce sont elles qui déterminent le déroulement d'un programme, en C, il y en a 6:

La séquence: les instructions se déroulent dans le bloc courant de haut en bas

Les conditionnelles:

Déclarations
Affectation

Portée des variables - Blocs
Entrées / Sorties
Chaines de contrôle

Structure de contrôle:

La séquence: les instructions se déroulent dans le bloc courant de haut en bas

Les conditionnelles:

Le si: if

Le cas: switch

Les boucles:

La boucle tant que: while et for contrairement à d'autres langages, il n'y a pas de boucle « pour » for est une boucle while déguisée

La boucle faire: do ... while
```

```
- Commentaires
- Identificateurs
- Directives
define, include if
didef
- Type de base
Entier, carac,
booleen, flothant,
pointeur,
operateurs
- Declarations
Affectation
- Portée des
variables - Blocs
- Entrées / Sorties
- Chaines de
contrôle
- Structure de
contrôle
- Sequence
- Seq
```

```
Tests imbriqués
     if (a > b && (a < 5 || b < 10))
     z = a;
     else z = b;
     • Rappel: arret de l'évaluation dès que :
         - condition fausse pour: &&
         - condition vraie pour: ||

    Booléens

      • pas de type booléen en C => simulation :
      #define False 0
      #define True 1
      #define Boolean int
      Boolean ok, fini, status;
      if (ok) /*( ok != 0)*/
      status = True:
     if (! fini) /*( fini == 0)*/
       status = False;
58/
```

```
    Commentaires
    Identificateurs
    Directives
    doffine, include if-
iffider
    Type de base
    Entier, carac,
    pooleen, frotont,
    pointeur,
    opérateurs
    Portée des
    variables – Blocs
    Entirees / Sorties
    Chaines de
    controle
    Structure de
    controle
    Temps    Temps
```

```
Cas particulier: else- if cascadés

Commentaires
Identificateurs
If (test1)
action1;
else if (test2)
action2;
else if (test3)
action2;
else if (test3)
action3;

Poctarations
Affectation
Porte des
variables - Blocs
Entres / Sortes
Chaines de controle Sequences
Controle Sequences
Controle Sequences
Coules(3)

Cas particulier: else- if cascadés
if (test1)
action1;
else if (test2)
action2;
else if (test3)
action3;
....
else action_ par- défaut;
Structure de tests en "rateau"

Ne pas mettre des { } inutiles
Aternative: le switch
```

```
• Switch

• Commentaires
• Identificateurs
• Directives
define, include if riddif
• Type de base
Entier, carac., booleen, flottant, pointeur, operateurs
• Declarations
Affectation
• Portée des variables – Blocs
• Entires / Sorties
• Chaines de controle
• Structure de controle Sequence
• boudes(3)

• Switch

switch (expression){

case const1:actions1;

break;

case const2: actions2;

break;

...

default:actionsdef;

}
• const1, const2 .... sont des constantes à valeur entière

• break ou return pour sortir du switch
• (sinon on continue jusqu'à la fin du switch)
• plusieurs actions possibles séparées par ;
```

```
• Boucles while et for

• Commentaires
• Identificateurs
• Directives
define, include if -
ifidef
• Type de base
Entler, carac., booleen, flottant, pointeur, operateurs
• Declarations
Affectation
• Portee des variables - Blocs
• Entrees / Sorties
• Chaines de controle
• Structure de controle
• ornorios esquence conditionnelies(2), boudles(3)
• Règle : utiliser for pour les boucles pour
```

