المدرسة العليا للعلوم التطبيقية والتصرف
ÉCOLE SUPÉRIEURE PRIVÉE DES SCIENCES APPLIQUÉES ET DE MANAGEMENT

Programmation C

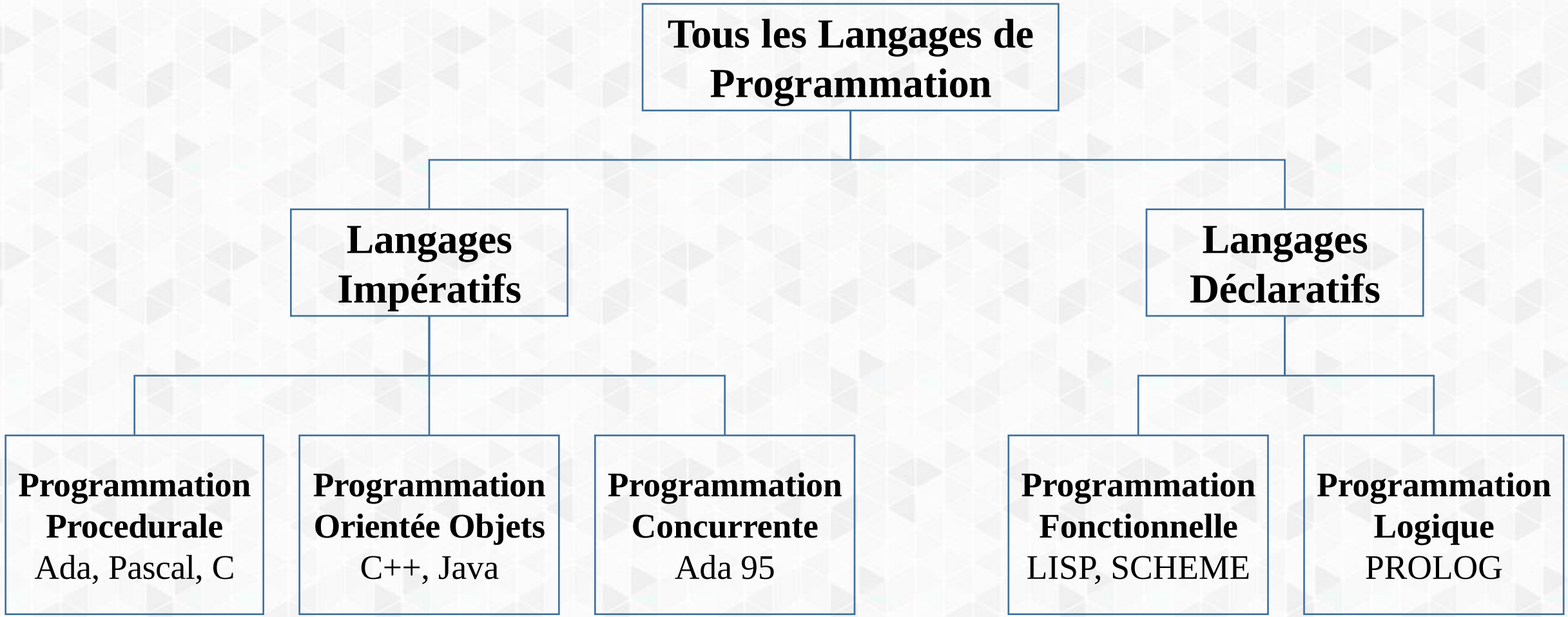
Chapitre 1:

Les notions de base

Nizar KHELIL & Fatma SOMAA & Fatma SOMAA

Année Universitaire 2017/2018

Introduction



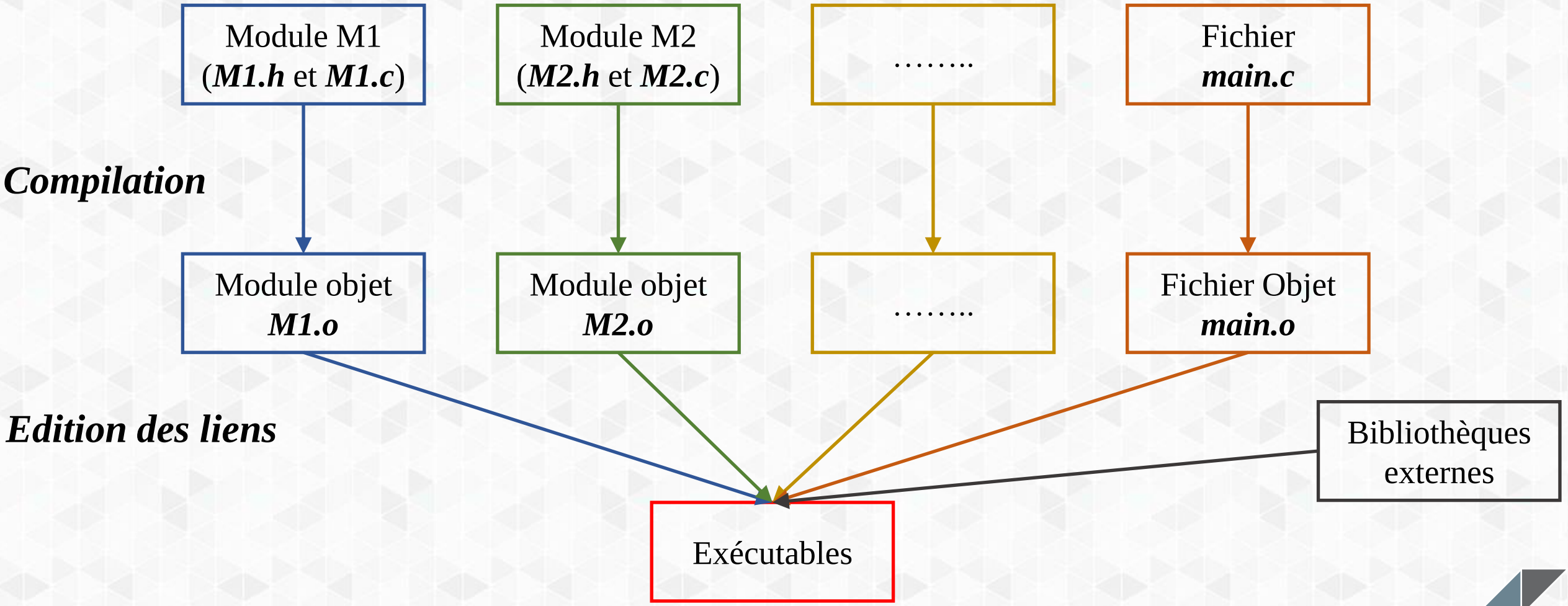
Plan

- Caractéristiques du C
- Génération d’un exécutable
- Anatomie
- Les directives de pré-compilation
- Notion du bloc
- La fonction main
- Les constantes
- Les variables
- Les opérateurs
- Les fonctions de base

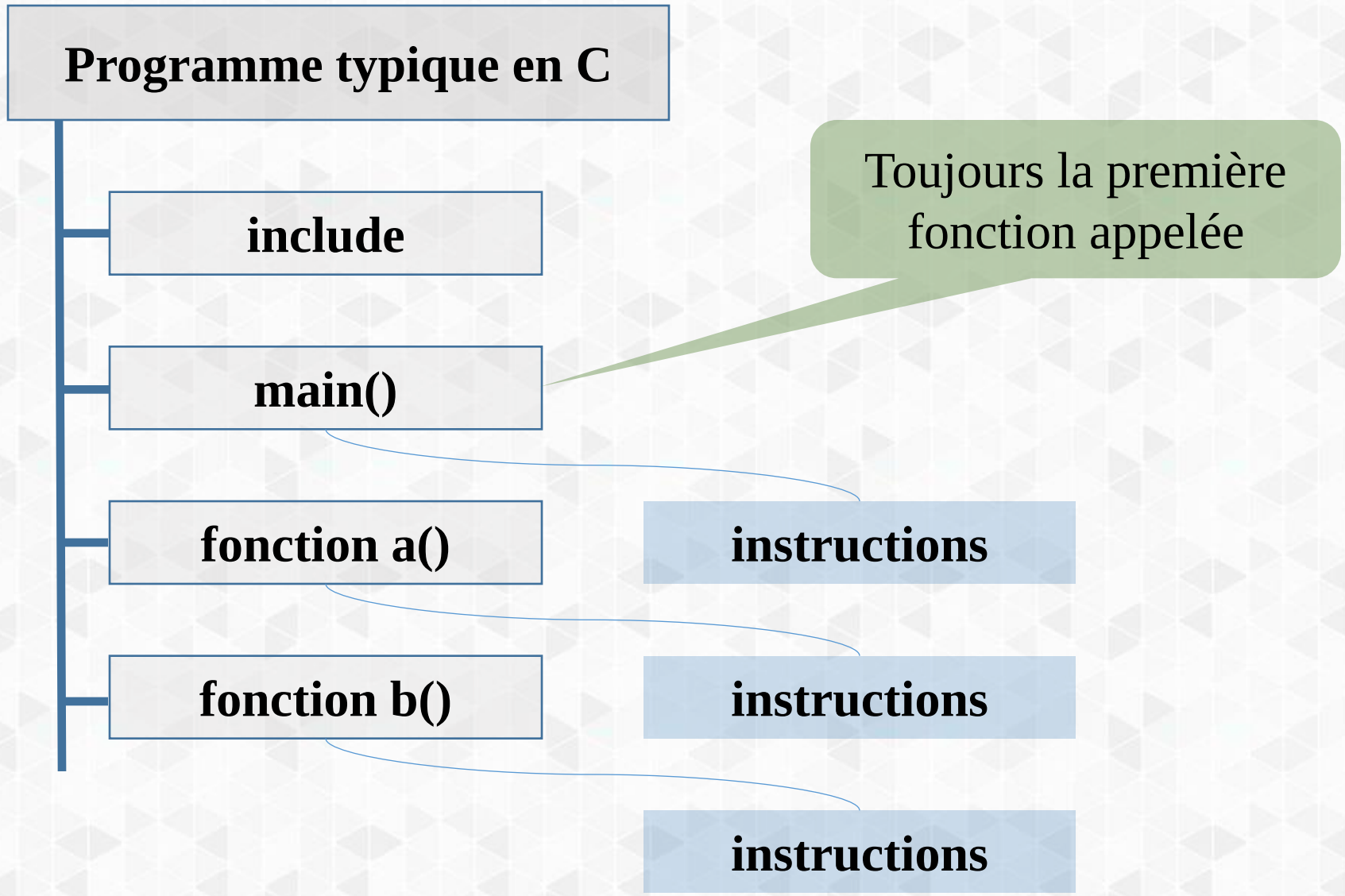
Caractéristiques du C

- **Structuré**
- **Modulaire:** peut être découpé en modules qui peuvent être compilés séparément
- **Universel:** n'est pas orienté vers un domaine d'application particulier
- **Typé:** tout objet C doit être déclaré avant d’être utilisé
- **Portable:** sur n'importe quel système en possession d'un compilateur C

Génération d'un exécutable



Anatomie


Programme typique en C
include
main()
fonction a()
fonction b()
Toujours la première fonction appelée
instructions
instructions
instructions

Les directives de pré-compilation

- Elles commencent toutes par un #.
- Exemples

#include <stdio.h> (permet d'utiliser les fonctions printf() et scanf())
#include <math.h> (permet d'utiliser les fonctions mathématiques)
#define PI 3.14159 (définit la constante PI)
#undef PI *(à partir de cet endroit, la constante PI n'est plus définie)*
#ifdef PI
    instructions 1 ...
    #else
    instructions 2 ...
    #endif *(si la constante PI est définie, on compile les instructions 1 sinon, les instructions 2)*

Notion du bloc

- Dans un bloc, les déclarations de données précèdent toujours les instructions.
- Une instruction simple est toujours terminée par un “ ;”.
- **Syntaxe générale**

{

Zone de déclaration de données

Zone des instructions

}

La fonction main

```
main()
        {
            int i ; /* déclaration des variables */
            instruction_1 ;
            instruction_2 ;
            …
            Return …;
        }
```

Les constantes

- En utilisant les directives:

#define nom contenu

- Exemple

```
#define  MAX 100
#define LANGAGE " java"
```

- En utilisant mot clé « const »

const type nom = valeur;

- Exemple

```
const int n=10;
const char ch = ‘a’;
```

- Constantes entières 1,2,3,...
- Constantes caractères ’a’,’A’,...
- Constantes chaînes de caractères "Bonjour"
- Pas de constantes logiques Pour faire des tests, on utilise un entier. 0 est équivalent a faux et tout ce qui est différent de 0 est vrai

Les variables : Noms des variables

- Le C fait la différence entres les MAJUSCULES et les minuscules.
- Par convention, on écrit les noms des variables en minuscule et on réserve
- les majuscules pour les constantes symboliques définies par un #define.
- Les noms doivent commencer par une lettre et ne contenir aucun blanc.
- Le seul caractère spécial admis est le soulignement (_).
- Il existe un certain nombre de noms réservés (while, if, case, ...) dont on
- ne doit pas se servir pour nommer les variables.
- on ne doit pas utiliser les noms des fonctions pour des variables.

Les variables : Déclaration des Variables

- Pour déclarer une variable, on fait précéder son nom par son type.
- Il existe 6 types de variables :

Type	Signification
char	caractère codé sur 1 octet (8 bits)
short	entier codé sur 1 octet
int	entier codé sur 4 octets
long	entier codé sur 8 octets
float	réel codé sur 4 octets
double	réel codé sur 8 octets

- On peut faire précéder chaque type par le préfixe « unsigned », ce qui force les variables à prendre des valeurs uniquement positives.

Les variables : Déclaration des Variables

- Exemples

Type	Signification
int a;	a est entier
int z=4;	z est entier et il vaut 4
unsigned int x ;	x est un entier positif (non signé)
float zx, zy ;	zx et zy sont de type réel
float zx=17.63 ;	zx est de type réel et vaut 17.63
char zz=’a’ ;	zz est un caractère et il vaut ’a’
double z ;	z est un réel en double précision

Les opérateurs

- L’opérateur d’affectation "=".
- Pour les opérations naturelles, on utilise les opérateurs « + », « - », « * », « / », « % ».
- % est l’opération modulo : 5%2 est le reste de la division de 5 par 2. 5%2 est donc égal à 1.
- Par ailleurs, l’opération ’a’+1 a un sens, elle a pour résultat le caractère suivant à a dans le code ASCII.

Les opérateurs

- En C, il existe un certain nombre d'opérateurs spécifiques tel que:
  - ++ incrémente la variable d'une unité.
  - - - décrémente la variable d'une unité.
- Ces 2 opérateurs ne s'utilisent pas avec des réels. Exemples d'utilisation :
  - i++ ; /* effectue i=i+1 */
  - i-- ; /* effectue i=i-1 */

Les opérateurs

- Quand l'opérateur ++ est placé avant une variable, l'incrémentation est effectuée en premier. L'incrémentation est faite en dernier quand ++ est placé après la variable.
- Exemples
  - int i=1 , j ;
  - j=i++ ; /* effectue d'abord j=i et ensuite i=i+ 1 */
    /* on a alors j=1 et i=2 */
  - j=++i ; /* effectue d'abord i=i+ 1 et ensuite j=i */
    /* on a alors j=2 et i=2 */
- D'autres opérateurs sont définis dans ce qui suit
  - i+=5 ; /* i=i+5 */
  - i-=3 ; /* i=i-3 */
  - i*=4 ; /* i=i*4 */
  - i/=2 ; /* i=i/2 */
  - i%=3 ; /* i=i%3 */

Les opérateurs

- Les opérateurs qui servent à comparer 2 variables sont :
  - == égal à
  - != différent de
  - < inférieur
  - <= inférieur ou égal
  - > supérieur
  - >= supérieur ou égal
  - && 'et' logique
  - || 'ou' logique

- D'autres opérateurs:
  - ! négation logique
  - ~ complément à un
  - & opérateur d'adresse appelé aussi de référencement

N.B: Ne pas confondre l'opérateur d'affectation = et l'opérateur de comparaison ==

Les opérateurs

- « sizeof » est un opérateur qui donne la taille en nombre d'octets du type dont le nom est entre parenthèses.

  sizeof(short) < sizeof(int) < sizeof(long)

  sizeof(float) < sizeof(double) < sizeof(longdouble)

- Il est possible de forcer la conversion d'une variable (ou d'une expression) dans un autre type avant de l'utiliser par une conversion implicite. Cette opération est appelée « cast ». Elle se réalise de la manière suivante :

  (type) expression

- exemple :

  i = (int) f + (int) d ;

Les opérateurs : Opérateurs ternaire

- L'opérateur ternaire met en jeu trois expressions (ex1,ex2,ex3) et il permet de construire une opération de test avec retour de valeur.
- Exemple1:

```
a == b ? c : d.
```

Cette expression retourne la valeur contenue dans la variable c si la valeur contenue dans la variable a est égale à celle de la variable b. Dans le cas contraire, l'expression retourne la valeur contenue dans la variable d.

- Exemple2:

```
a >= b ? a : b
```

si la valeur de a est supérieure ou égale à la valeur de b, alors l'expression donne la valeur de a ;sinon l'expression donne la valeur de b ;

- Exemple3:

```
max = a >= b ? a : b
```

permet de mémoriser dans la variable max la valeur maximum des deux valeurs contenues dans les variables a et b.

Les fonctions de base: La fonction printf()

- Elle sert à afficher à l'écran la chaîne de caractère donnée en argument.

Exemple: printf("Bonjour\n") ; affichera Bonjour à l'écran.

- Certains caractères ont un comportement spécial :

\n : retour à la ligne

\b : backspace

\r : retour chariot

\t : tabulation horizontale

\v : tabulation verticale

\f : saut de page

\a : alert (sonnerie, BEL)

\" : affichera "

\' : affichera '

\? : affichera ?

\! : affichera !

\\ : affichera \

Les fonctions de base: La fonction printf()

- printf() permet d'afficher à l'écran la valeur d'une variable.
- Le caractère % indique le format d'écriture à l'écran.

```
printf("n=%d, m=%d",n,m);
```

Les fonctions de base: La fonction printf()

Format	Type	Explication
%d	Integer	Entier(décimal)
%u	Unsigned	Entier non signé (positif)
%hd	Short	Entier court
%l	Long	Entier long
%f	Float	Réel, notation avec le point décimal (ex 123.15)
%e	Float	Réel, notation exponentielle (ex $0.12315^{E}03$)
%lf	Double	Réel en double précision, notation avec le point décimal
%le	Double	Réel en double précision , notation exponentielle
%c	Char	Caractère
%s	String	Chaine de caractère

Les fonctions de base: La fonction printf()

- Exemple:

```
main()
{
int n=3, m=4 ;
printf("%d",n) ;                    /* affiche la valeur de n au format décimal */
printf("n=%d",n) ;                  /* affiche 'n=3' */
printf("n=%d, m=%d",n,m);           /* affiche 'n=3, m=4' */
printf("n=%5d",n) ;                 /* affiche la valeur de n sur 5 caractères : 'n= 3'*/
}
```

Les fonctions de base: La fonction scanf()

- Elle permet de lire la valeur que l'utilisateur rentre au clavier et de la stocker dans la variable donnée en argument.
- *Exemple:*

```
main()
{ int a ;
scanf("%d",&a) ;
}
```

- On retrouve les formats de lecture précisés entre " " utilisés pour printf().
- « **&** » est indispensable pour le bon fonctionnement de la fonction. Il indique l'adresse mémoire de la variable.

Des questions

المدرسة العليا للعلوم التطبيقية والتصرف
ÉCOLE SUPÉRIEURE PRIVÉE DES SCIENCES APPLIQUÉES ET DE MANAGEMENT

Programmation C

Chapitre 2:

Les structures conditionnelles

Nizar KHELIL & Fatma SOMAA

Année Universitaire 2017/2018

If

- **Syntaxe:**

```
if (expression1)
    Instruction1
else if (expression2)
    Instruction2 ...
```

- Le else est facultatif

```
If (expression)
    instruction
```

Switch

- **Syntaxe:**

```
switch (expression)
{
case constante1 :
        liste d'instructions1;
break;
case constante2 :
        liste d'instructions2;
break;
... default : liste d'instructions n;
Break;
}
```

Des questions

المدرسة العليا للعلوم التطبيقية والتصرف
ÉCOLE SUPÉRIEURE PRIVÉE DES SCIENCES APPLIQUÉES ET DE MANAGEMENT

Programmation C

Chapitre 3:

Les structures itératives

Nizar KHELIL & Fatma SOMAA

Année Universitaire 2017/2018

While

- **Syntaxe:**

```
while (expression)
instruction;
```

- Tant que expression est non nulle, instruction est exécutée.
- Si expression est nulle instruction ne sera jamais exécutée

- **Exemple:**

```
I=1;
While (i < 10)
{
printf("\n i = %d",i);
I++;
}
```

Affiche les entiers de 1 à 9

Do --- while

- **Syntaxe:**

```
do
instruction;
while (expression);
```

- L'instruction est exécutée tant que expression est non nulle.
- Instruction est toujours exécutée au moins une fois.

- **Exemple:**

```
int a;
do
{
printf("\n Entrez un entier entre 1 et 10 : ");
scanf("%d",&a);
}
while ((a <=0) || (a > 10));
```

saisie au clavier un entier entre 1 et 10

For

- **Syntaxe**

```
for ( expr1; expr2; expr3)
instruction;
```

équivalent à

```
expr1;
while (expr2);
{
instruction;
expr3;
}
```

- **Exemple :**

```
for (i = 0; i < 10; i++)
printf("\n i = %d",i);
```

A la fin de la boucle i vaut 10

Instructions de branchement non conditionnelles

❑break :

- Elle permet d'interrompre le déroulement d'une boucle, et passe à la première instruction qui suit la boucle.

- **Exemple:**

```
main()
{ int i;
for (i = 0; i < 5; i++)
{ printf("i = %d\n ",i);
if (i==3)
break;
}
printf("valeur de i a la sortie de la boucle = %d\n,i);
}
```

Imprime → i= 0 jusqu'à i=3 et

→valeur de i a la sortie de la boucle = 3

Instructions de branchement non conditionnelles

❑continue:

- Elle permet de passer directement à l'itération suivante de la boucle sans exécuter les autres instructions de l'itération courante.

- **Exemple:**

```
main()
{ int i;
for (i = 0; i < 5; i++)
{
if (i==3)
continue;
printf("i = %d\n",i);
}
printf("valeur de i a la sortie de la boucle = %d\n,i);
}
```

→ *imprime i= 0, i=1, i=2 et i = 4*

→ *valeur de i à la sortie de la boucle = 5*

Des questions

المدرسة العليا للعلوم التطبيقية والتصرف
ÉCOLE SUPÉRIEURE PRIVÉE DES SCIENCES APPLIQUÉES ET DE MANAGEMENT

Programmation C

Chapitre 4:

Les tableaux

Nizar KHELIL & Fatma SOMAA

Année Universitaire 2017/2018

Déclaration d'un tableau

- **Syntaxe:**

  Déclaration d'un tableau de taille n :

  type nom_tableau [n] ;

  réservation de n cases contiguës en mémoire

- **Exemple:**

  int tb1[10] ;

Accès aux éléments d'un tableau

- Modification du contenu d'un élément :

  Exemple: `tb1[3] = 19 ;`

- Utilisation de la valeur d'un élément :

  Exemple : `x = tb1[3] + 1 ;`

- L'utilisation est toujours élément par élément

Initialisation au moment de la déclaration

- Initialisation au moment de la déclaration

  Exemple : int tb1[10] = { 21, 32, -4, 1, 37, 88, 9, -1, 0, 7} ;

- Initialisation d'une partie du tableau :

  Exemple : int tb1[10] = { 21, 32, -4, 1} ;

- Initialisation sans spécifier la taille :

  Exemple : int tb1[ ] = { 21, 32, -4, 1} ;

Opérateur &

- En langage C, l'adresse en mémoire d'une variable est donnée par &variable

- Même principe dans le cas des tableaux :

  &(tb1[i]) *correspond à l'adresse en mémoire de tb1[i]*

→ tb1 est l'adresse en mémoire du premier élément du tableau → &(tb1[0])

Tableau à plusieurs dimensions

- Déclaration d'un tableau de taille n:

  type nom_tableau [n][m] ;

  - *réservation de n.m cases contiguës en mémoire*
  - *n : le nombre de lignes*
  - *m le nombre de colonnes*

- Exemple:

  int tb1[2][4] ;

Des questions

المدرسة العليا للعلوم التطبيقية والتصرف
ÉCOLE SUPÉRIEURE PRIVÉE DES SCIENCES APPLIQUÉES ET DE MANAGEMENT

Programmation C

Chapitre 5:

Les chaines de caractères

Nizar KHELIL & Fatma SOMAA

Année Universitaire 2017/2018

Introduction

- il n’existe pas de type chaîne de caractères prédéfinit en C.

- Il existe deux façons pour déclarer une chaîne de caractères

    i. en utilisant un tableau de char dont la taille est fixée en avance,

    ii. en utilisant un pointeur sur des char dont la taille de la chaîne ne peut être connue d’avance.

Déclaration

- La déclaration est identique à un tableau normal

char nom[dimension_max] ;

- La représentation interne d'une chaîne de caractères est terminée par le symbole '\0' (NULL) ➔ pour un texte de n caractères, nous devons prévoir n+1 octets.

NB: le compilateur C ne contrôle pas si nous avons réservé un octet pour le symbole de fin de chaîne; l'erreur se fera seulement remarquer lors de l'exécution du programme

Initialisation

Exemples:

- char MACHAINE[ ] = “Hello”;
- char MACHAINE[6] = “Hello”;
- char MACHAINE[ ] = {‘H’ , ’e’ , ’l’ , ’l’ , ’o’ , ’\0’};
- char MACHAINE[8] = “Hello”;
- char MACHAINE[5] = “Hello”; ➔ donnera une erreur à l’exécution
- char MACHAINE[4] = “Hello”;➔ donnera une erreur à la compilation.

NB:

- ‘x’ est un caractère constant, qui a la valeur 120 dans le code ASCII➔ il est codé dans un octet
- ” x” est un tableau de caractères qui contient deux caractères: la lettre 'x' et le caractère NUL: '\0' ➔ "x” est codé dans deux octets

Fonctions prédéfinies

- Des fonctions de traitement des chaînes de caractères sont disponibles dans les bibliothèques standards:
- <stdio.h>:
    - scanf, printf en utilisant le format %s (scanf prend une adresse en argument (&x)),
    - puts(MACHAINE); est équivalent à printf
    - gets: gets(MACHAINE); lit une ligne jusqu'au retour chariot et remplace le '\n' par '\0' dans l'affectation de la chaîne.

- <string.h>
    - strlen(s) fournit la longueur de la chaîne sans compter le '\0'
    - strcpy(s,t) copie « s » vers « t »
    - strcat(s, t) ajoute « t » à la fin de « s »
    - strcmp(s,t) compare « s » et « t » et lexicographiquement et fournit un résultat:
        - négatif si « s » précède « t »
        - zéro si « s » est égal à « t »
        - positif si « s » suit « t »
    - strncpy(s, t, n) copie « n » caractères de « t» vers « s »
    - strncat(s, t, n) ajoute « n » caractères de « t» à la fin de « s»

- <stdlib.h> conversion chaîne -> nombre
- atoi(s) retourne la valeur numérique représentée par « s » comme int
- atol(s) retourne la valeur numérique représentée par « s » comme long
- atof(s) retourne la valeur numérique représentée par « s » comme double

- Règles générales pour la conversion:
  - Les espaces au début d'une chaîne sont ignorés
  - La conversion s'arrête au premier caractère non convertible
  - Pour une chaîne non convertible, les fonctions retournent zéro

Des questions

المدرسة العليا للعلوم التطبيقية والتصرف
ÉCOLE SUPÉRIEURE PRIVÉE DES SCIENCES APPLIQUÉES ET DE MANAGEMENT

Programmation C

Chapitre 6:

LES FONCTIONS

Fatma SOMAA

Année Universitaire 2017/2018

Définition d'une fonction

type nom-fonction (type-1 arg-1, …, type-n arg-n)
{
[déclarations de variables locales]
liste d'instructions
}

• type: désigne le type de la fonction ➔le type de la valeur qu'elle retourne
si la fonction ne renvoie pas de valeur elle est de type void.

Définition d'une fonction

- La fonction se termine par l'instruction « return »

  return(expression);

- expression est de même type que celui de la fonction

NB: si son type est void ➔ On peut ne pas mettre return

```
int puissance (int a, int n)
{
        if ( n == 0)
            return(1);
        return(a * puissance(a, n-1));
}
```

Appel & déclaration d'une fonction

- **L'appel** ce fait par:

nom-fonction(para-1,para-2, …, para-n);

- **Déclaration**

Le langage C n'autorise pas les fonctions imbriquées

➔On peut déclarer une fonction secondaire soit avant, soit après la fonction principale main: il est indispensable que le compilateur "connaisse" la fonction à son appel.

Elle doit impérativement être déclarée avant :

➔ type nom-fonction (type-1, …, type-n); • /* sans code */

➔Ou bien utiliser des <fichier,h> : Programmation modulaire

Déclaration d'une fonction

```
int puissance (int , int);

main()
{
 int x = 2, y = 5;
printf("%d\n",puissance(x,y));
}

 int puissance (int a, int n) {
 if (n == 0)
return(1);
return(a * puissance(a, n-1));
}
```

Durée de vie des variables

Les variables manipulées dans un programme C n'ont pas la même durée de vie. ➔2 catégories de variables :

- **variables permanentes (ou statiques):** Occupent une place mémoire durant toute l'exécution du programme.

- **variables temporaires** : Se voient allouer une place mémoire de façon dynamique, Elles ne sont pas initialisées. Leur place mémoire est libérée à la fin d'exécution de la fonction secondaire.

Exemple de variable globale

- Variable globale : variable déclarée en dehors des fonctions.

```
int n;
```

- void fonction();

```
void fonction()
{ n++;
printf("appel numero %d\n",n);
}
main() {
int i;
for (i = 0; i < 5; i++)
 fonction();
 }
```

Exemple de variable locale

- Variable locale : variable déclarée à l'intérieur d'une fonction (ou d'un bloc d'instruction).

```
 int n = 10;
 void fonction ();
void fonction ()
{
int n = 0;
n++;
printf("appel numero %d\n",n);
} main()
{
int i;
for (i = 0; i < 5; i++)
fonction();
}
```

Transmission des paramètres d'une fonction

- Les paramètres de fonction sont traités de la même manière que les variables locales.
- On dit que les paramètres d'une fonction sont transmis par valeurs. Exemple:

```
void echange (int , int);
void echange (int a, int b)
{ int t;
 printf("debut fonction :\n a = %d \t b = %d\n",a,b);
t = a;
a = b;
b = t;
printf("fin fonction : \n a = %d \t b = %d\n",a,b);
 return;
}
```

... suite

```
main()
{
int a = 2, b = 5;
printf("debut programme principal : \n a = %d \t b = %d\n",a,b); echange(a,b);
 printf("fin programme principal : \n a = %d \t b = %d\n",a,b);
 }
```

Affiche début programme principal
a = 2 b = 5
debut fonction:
a = 2 b = 5
fin fonction: a = 5 b = 2
fin programme principal: a = 2 b = 5

les paramètres par adresse

- Pour qu'une fonction modifie la valeur de ses arguments, il faut passer les paramètres par adresse :

```
void echange (int *adr_a, int *adr_b);
void echange (int *adr_a, int *adr_b) {
int t;
t = *adr_a; *adr_a = *adr_b; *adr_b = t;
}
main(){
 int a = 2, b = 5;
 printf("debut programme principal : \n a = %d \t b = %d\n",a,b);
echange(&a,&b);
printf("fin programme principal : \n a = %d \t b = %d\n",a,b);
}
```

Des questions