P. CASTELAN / D. MARY

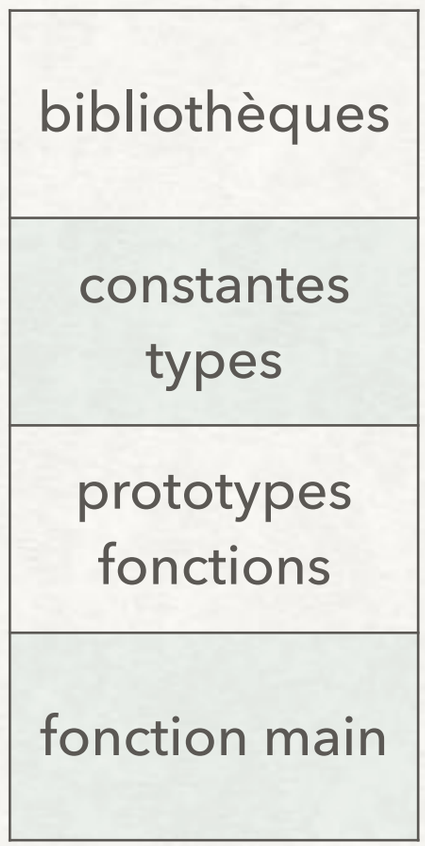
LANGAGE C

BASES

STRUCTURE D'UN PROGRAMME C

LANGAGE DECLARATIF

- #include : inclusion de bibliothèques
- #define : définition de constantes
- typedef : définition de types.
- Variables globales : connues de toutes les fonctions…
- Prototypes : déclarent les fonctions. Obligatoire.
- Fonctions :
  - C est un langage fonctionnel : collection de fonctions
  - C comporte a minima une fonction : main.
  - Le code d'un programme C est écrit au sein de fonctions.



Les variables globales sont interdites cette année !

FONCTION

TYPE NOM(LISTE DE PARAMÈTRES)

- Une fonction a un nom unique
  - Pas de caractères spéciaux…
- Une fonction retourne une valeur qui a un type avec return
  - Une fonction qui ne retourne rien, est de type void
- Une fonction reçoit des paramètres
  - Consignes données à la fonction
  - Pas de données à transmettre : remplacer la liste de paramètres par void
- Exemple : cos (x)
  - Retourne un réel, s'appelle cos, nécessite un réel
    => Prototype de cette fonction : float cos(float x) ;

Exemple de fonction

```
float carre( float x)
{    return x*x ;
}
```

FONCTION MAIN

int main (void)

- OBLIGATOIRE ! C'est le point de départ du programme…
- Formatée :
  - le nom doit être main
  - type int
  - pas de paramètres : donc void…
    - En fait il peut y en avoir mais cette année pas de paramètres
- Doit retourner 0

Exemple de fonction main

```
int main( void )
{   …
    …
    …
    return 0 ;
}
```

VARIABLES

NOTION DE TYPE

- Une variable est le nom que l'on donne à une zone de la mémoire
- Une variable est typée :
  - Entier : int
  - Réel : float, double
  - Caractère : char

Un ordinateur stocke des 0 et des 1.
Par le type, on précise comment interpréter cette suite de valeurs binaires.

VARIABLES

AFFECTATION

- Affectation "=" : attribue une **valeur** à la variable
  - Respect du type !
  - La variable reçoit la valeur, elle est à gauche du signe "="
- lvalue = valeur ;
  - Seule une variable peut se trouver à gauche du signe égal.
- A = 3 ; // affecte 3 à la variable A qui doit être int ou float.
- A = 3.14 ; // affecte 3.14 à la variable A qui doit être float.

VARIABLES

UTILISATION

- On utilise une variable comme une valeur.
  - Par son nom
  - Avec une combinaison d'opérateurs.
- Une variable a une valeur, un nom et une adresse

Utilisation	nom	&nom
Résultat	valeur contenue	adresse variable

```
int A ;
A = 3 ;
printf("%d", A) ;
printf("%d",&A) ;
```

Prog S9.c

OPERATEURS

POUR COMBINER DES VARIABLES

- Arithmétiques : + ; - ; * ; / ; % (modulo)
- Comparaison : > ; < ; >= ; <=
- Unaires :
  - Incréments : ++ ; --
    - A++ ou ++A : augmente A de 1
- Combinés : += ; -= ; *= ; /=
  - a += 3 ; // a = a + 3 ;
- Négation : !A
  - En C est Vrai tout ce qui est non nul.

```
A = 3 ;
printf("!3 = %d", !A ) ;
```

Prog S10.c

OPERATEURS
LOGIQUES

- Pas de booléen en C : toute expression entière peut être utilisée.

A	B	A&&B	A||B
0	0	0	0
0	1	0	1
1	0	0	1
1	-1	1	1

| est le caractère pipe

Attention !
Il faut 2 & ou 2 | pour traiter la variable, sinon on traite les bits de la variable…

0 : Faux, n'importe quoi d'autre : Vrai

SEQUENCE

LISTE D'INSTRUCTIONS

- Une séquence est une liste d'instructions permettant d'effectuer une action complexe.
- Chaque instruction est séparée des autres par un ; (point virgule).
- Une séquence débute par une {
- Une séquence se termine par une }
- break permet de quitter une séquence avant d'atteindre sa fin.
- Chaque fonction comporte au moins une séquence.
- On peut déclarer des variables dans une séquence, **elles sont locales à la séquence**.

De toutes façons pas le choix cette année, les variables globales étant interdites

SEPARATEURS

PONCTUATION DU C

- ; : sépare les instructions : A = 3 ; B = 4 ; ...
- , : sépare les éléments d'une liste : int toto(int A, int B, ...
- ' ' : indique un caractère : char A = 'a' ;
- " " : chaine de caractère : printf("coucou") ;
- { } : limites d'une séquence.
- . : partie décimale d'un réel : float a = 3.14 ;
- [ ] : sélection d'un élément de tableau.

VARIABLES DIMENSIONNEES

VECTEURS

- C'est un regroupement de variables du même type
- Accès aux données via un index.
- L'index débute à 0.

Déclaration directe

```
int a[ 3 ] ; // déclare un tableau a une dimension de 3 éléments
```

Déclaration avec type

```
typedef int vect[ 3 ] ;
...
vect a ; // déclare un tableau a une dimension de 3 éléments
```

Index : i	0	1	2
a[i]	...	...	...

VARIABLES DIMENSIONNEES

MATRICES

- Tableau a deux dimensions…
- Eléments du même type…
- Deux index : deux jeux de crochets !

- L'interprétation des index est arbitraire !
- Les index sont des entiers…

Déclaration directe

```
int a[ 3 ][ 3 ] ; // déclare un tableau a
une dimension de 3 éléments
```

Index : i et j	i = 0	i = 1	i = 2
j = 0	a[0][0]	a[1][0]	a[2][0]
j = 1	a[0][1]	a[1][1]	a[2][1]
j = 2	a[0][2]	a[1][2]	a[2][2]

Déclaration avec type

A

```
typedef int mat[ 3 ][ 3 ] ;
…
mat a ; // déclare un tableau a deux
dimensions de 9 éléments
```

B

```
typedef int vect[ 3 ] ;
typedef vect mat[ 3 ] ;
…
mat a ; // idem
```

VARIABLES

INITIALISATION

- On fixe la valeur initiale en l'affectant à la variable lors de sa création : float pi = 3.14159265358979 ;

- Variable dimensionnée ? On initialise avec une liste de valeurs !

  - int a[3] = { 1, 2, 3 } ;
  - int b[3] = { 0 } ;

index	0	1	2
a	1	2	3
b	0	0	0

- Deux dimensions ? Deux listes !

  - int m[3][3] = { {0} , { 1, 2, 3 }, {2} } ;

index	0	1	2
0	0	0	0
1	1	2	3
2	2	0	0

Une variable non initialisée contient ce qu'il y avait en mémoire a sa création : n'importe quoi !

#include

BIBLIOTHÈQUES

- Le noyau du C ne contient que très peu d'instructions.
- Extension via des bibliothèques que l'on inclue au besoin dans le programme.
  - Entrées/Sortie : stdio.h : standard input output
  - Fonction courantes : stdlib.h : standard library
  - Mathématiques : math.h : fonction mathématiques
- C'est une directive de compilation pas une instruction !
  - D'où le #
  - D'où pas de ; à la fin de la ligne

PROG C

bibliothèques
constantes types
prototypes fonctions
fonction main

#define

CHAINES DE REMPLACEMENT

- Il est possible de créer un dictionnaire en début de programme
- La chaine est remplacée dans tout le programme
- C'est une directive de compilation d'où…
  - Pas de ;
  - En début de programme
- Exemple :
  - #define n 100
  - #define begin {
  - #define end }

PROG C

#include #define
constantes types ;
prototypes ;
fonctions
fonction main

FONCTIONS

LE RETOUR…

- On déclare les fonctions à l'aide des prototypes
  - C'est la 1er ligne de la fonction suivie d'un ;
- Les fonctions sont étanches… pour les paramètres simples.
- int toto(int a , int b[3])
- La fonction s'appelle toto, elle retourne un entier et admet deux paramètres : un entier a, et b un vecteur de 3 éléments.
- Regardez le code S19.c

PROG C

bibliothèques
constantes types ;
prototypes ;
fonctions
fonction main

ENTREES SORTIES.

EN MODE TEXTE !

- Affichage à l'écran avec **printf**
- Saisie au clavier avec **scanf**
- Ce sont des fonctions standard d'entrée sortie, normalisées : stdio.h
- Elles permettent de formater les saisies/affichages : d'où le "f"
- Aller voir le site pcastelan…
  - Fiche Langage C -> printf puis scanf…
- Pourquoi le scanf veut des adresses ?

printf

AFFICHAGE FORMATÉ

- Syntaxe : printf("Chaine de format", liste de valeurs)
- Chaine de format ?
  - Du texte : printf("Coucou");
  - Des champs
    - Lieu ou les valeurs de la listes, interprétées sont affichées
    - printf("A = %d.", 3) affiche la chaine "A = 3."
    - %f = float ; %d = int ; %c = char ; %lf = double
    - %s = tableau de char

TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…

scanf

SAISIE FORMATÉE

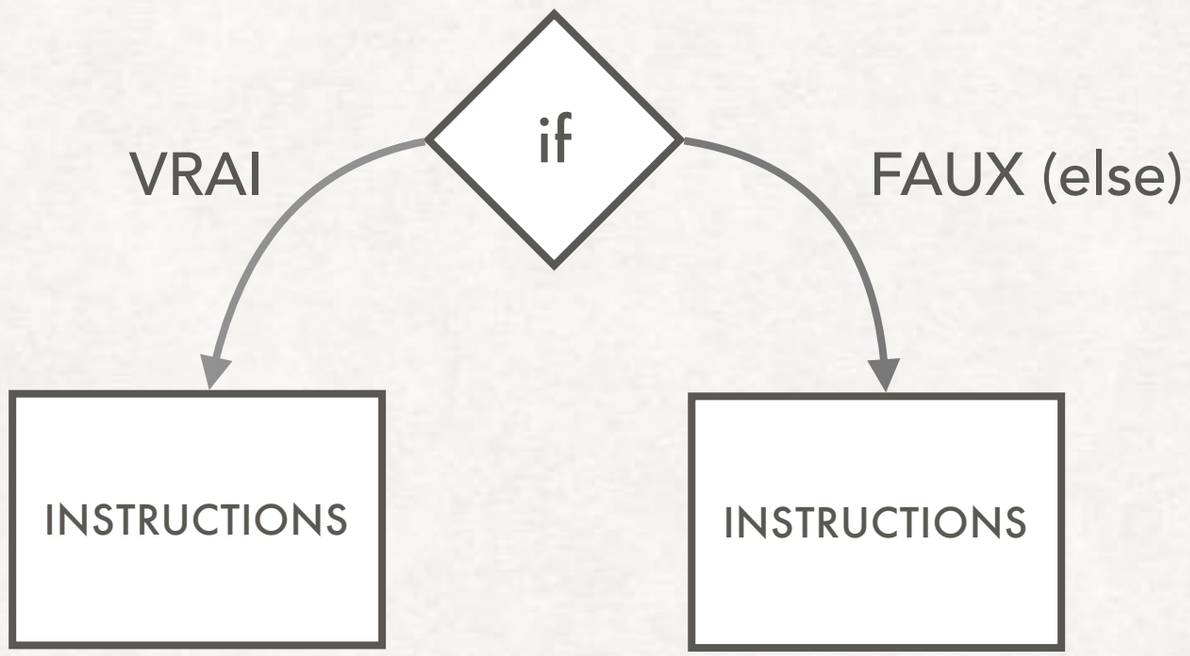
- Syntaxe : scanf("Chaine de format", liste d'adresses)
- Chaine de format ?
  - PAS DE TEXTE !
  - Des champs
    - Ici les champs expliquent comment interpréter les données lues au clavier.
    - scanf("%d", &a) ; provoque une saisie, stocke les données interprétées comme entier à l'adresse de la variable a : "dans a"
    - %f = float…
- Site pcastelan

TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…

INSTRUCTIONS DE CONTROLE DE FLUX

BRANCHEMENTS

- Permet d'orienter le flux des instructions
- Aller voir le site pcastelan
- if : branchement binaire



TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…

if

LA SUITE…

- Il n'est pas obligatoire de placer un else
  - Exemple : écrivez un code forçant la saisie d'un nombre impair en ajoutant 1 à la saisie si elle est paire
  - Corrigé prog S24.c
  - ATTENTION : Test d'égalité : ==
- La condition est entre parenthèses.
- Ce peut être un appel a une fonction : S24b.c

TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…

switch

CHOIX DU POINT D'ENTRÉE DANS UNE SÉQUENCE

- Permet d'orienter le flux des instructions
- Il faut pouvoir lister tous les cas possibles => variable énumérable
- Aller voir le site pcastelan

```
char choix ;
      …
switch( choix ) /* choix est une variable de type char
                 * les "cas" sont donc des char
                 */
{     case 't' : // pas d'instruction on passe à e=2
      case 'T' : e = 2 ;
                 d = 0 ;
                 break ;
      case 's' :
      case 'S' : e = 3 ;
                 d = 1 ;
                 break ;
      default :
         printf( "veuillez entrer 't', 'T', 's' ou 'S' ! " );
}
```

BOUCLES

RÉPÉTER DES INSTRUCTIONS

- while(condition) { … }
- do{…} while(condition);
- for(initialisations ; test d’arrêt ; incréments)
- Toutes ces boucles sont en fait des "tant que"
- La condition doit être mise à jour dans la boucle

while

while(condition){ ... }

- La condition doit être vérifiée pour entrer dans la boucle
  - Penser au besoin à l'initialiser
  - La séquence de la boucle peut ne pas être exécutée…
- La condition doit être mise à jour…
  - while (1) ;
- Site pcastelan

TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…

do {...} while

do {...} while(condition) ;

- La boucle est toujours parcourue au moins une fois.
  - On parle de test a posteriori.
- La condition doit être mise à jour…
- Site pcastelan
- while et do… while servent à gérer des boucles sans compteur a condition complexe
  - On peut bien sur y mettre des compteurs.

TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…

for

for (initialisationS ; testS ; incrémentationS)

- Boucle avec compteur
  - Le pas n'est pas forcément égal à 1
- Trois zones ! => 2 points virgules
  - Dans chaque zone il peut y avoir une liste…
  - Il est conseillé de n'avoir qu'un seul test d'arrêt, le premier atteint arrête la boucle.
- Site pcastelan

TRAVAILLEZ A PARTIR DE LA FICHE DU SITE : FAITES LA EVOLUER…