C - PARTIE 1 - EXPRESSION - INSTRUCTIONS — Le 19/09/05

DECLARATION STATIQUES

I/ Introduction.

C : Langage introduit dans les années 70 par Kernigan - C'est un langage impératif.
Il existe 3 types de langage :
- langage fonctionnel. (OCaml ; lisp ; haskell...)
- langage déclaratif (Prolog ; Réécriture MAUD)
- langage impératif (C, Pascal, Cobol, Ada, Fortran...)

1.1) Rappel élémentaire sur les grammaires.

une grammaire : formalisme pour décrire les langages informatique, la syntaxe

- On distingue 3 élts :
- les Terminaux → les mots que l'on écrit. "il" "elle"
- les non-terminaux → des mots qui représente des structures de phrases. pronom s'écrit <pronom>
- les Règles → ce sont des règles qui permettent de remplacer les noms terminaux par des terminaux ou d'autres structures.

Pour décrire ces règles, on utilise les opérations suivantes (expression rationnelle) :
- le choix 'il' | 'elle'
- séquences <pronom> <verbe> <CO>
- l'option <pronom> <verbe> [<CO>]
- répétition 'aïe'*, {'aïe'}

La structure d'une règle est non terminal ::= exp. rationnelle.
ex: <pronom> ::= 'il' | 'elle' | 'ils' | 'elles'
il existe un non terminal spécial appelé axiome qui débute la grammaire.
exemple de grammaire (phrase simple)
<P> ::= <S> <V> <CO>
<V> ::= 'voit' | 'suit'
<S> ::= <nom> | <pronom>

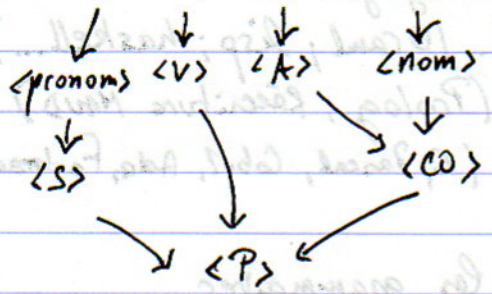
<pronom> ::= 'il' | 'elle' <P> est l'axiome.

<nom> ::= 'chien' | 'chat'

<A> ::= 'le' | 'un'

<CO> ::= <A> <nom>

ex : il voit le chien (reconnaissance de droite → gauche)



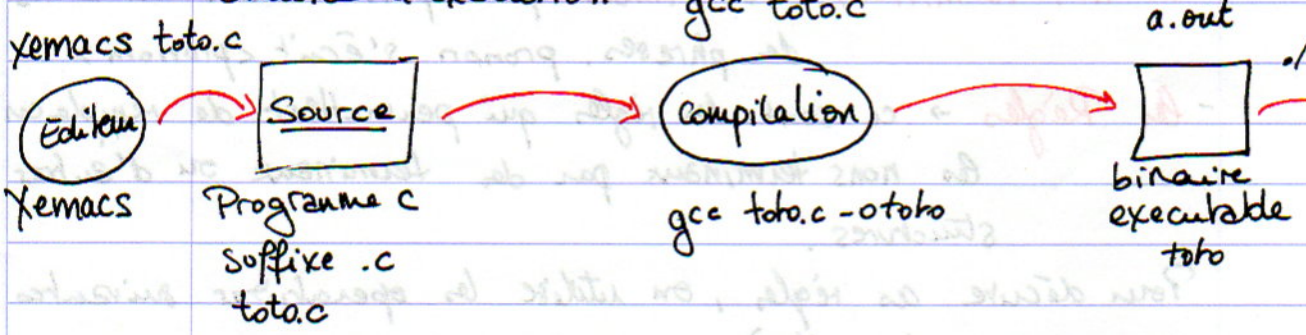
il voit (hors langage car non réductible à l'axiome)

<pronom> <V>

1.2/ Environnement du C.

- le C est un langage compilé, ce qui signifie qu'il est necessaire d'avoir un logiciel (compilateur) pour traduire le programme en langage c en binaire executable (ou non).

Chaîne d'execution

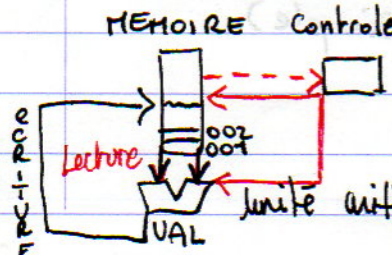


1.3/ Premier Programme C : hello.c #include <stdio.h> librairie.

```
#include <stdio.h>
main () {
    printf ("hello world ! \n");
}
```

```
> gcc hello.c -o hello
> ./hello
```

1.4/ Architecture des ordinateurs (Rappel)



→ chemin des données
→ chemin de traitement.
la memoire est divisée en 2 zones et elle possède des adresses de mémoire.

1.5/ Programme simple.

Memoire : - unité de stockage des données, des programmes.
- elle se "découpe en octet" (généralement)

unité élémentaire de stockage. Binary digit :

Bit : groupement de 8 bits = octect.

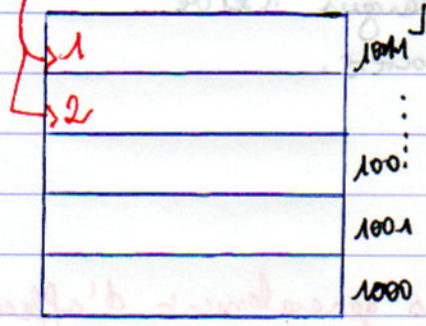
groupement d'octets = Mots.

Chaque adresse mémoire correspond à 1 octet.

La mémoire s'accède octet par octet.

II/ Variable, Valeur, Adresse.

2.1/ valeur = contenue de la mémoire. adresse : nb permettant de designer une ¢ mémoire.



en C, "l'abstraction" permettant de designer une ¢ mémoire est la variable

```
{ int a ;
  int b ;   ] déclaration des variables a et b.
  a = 1 ;
  b = 2 ;   ] operation d'affectation.
}
```

printf ("%d", a) ; écrit la valeur de a.

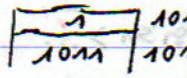
b = a ; écrire dans la variable b la valeur de a.

printf ("%d", &a) ;

écrit l'adresse de a à l'écran.

& <variable> donne l'adresse

b = &a ;



2.2/ Programme Simple grammaire.

un programme simple est un bloc d'instructions et d'expressions.

<bloc> ::= '{'
  [<declaration>]
  [<instructions>]
'}'

: optionnel.

Dans la zone de déclaration on décrit les variables que l'on utilisera. Cette déclaration s'écrit sous la forme
<type> <variable> ;
(64 bits) (4 octets) (Lettre)
Il y a 3 types simples 'float', 'int', 'char';

* exemple :

{ int a ; (4 octets) } Le chargeur réserve
  float b ; (8 octets) } ces octets.
  int c ; (4 octets)
}

Instructions : Suite de calculs generalement d'affectation.
l'execution du prog. se fait en séquence. (cad une instruction après l'autre dans le sens de la lecture classique).

```
{ int a,b ;          { int a,b ;
  a = 1 ;              a = 1 ;
  b = 2 ;              b = 2 ;
  b = a ;              a = b ;
  a = b ;              b = a ;
}                    }
```

valeur de a=1 // b=1 — valeur de a=2 // b=2

Instruction & Expression

<instruction> ::= <expression> ';'
<expression> ::= <variable> '=' <expression> (affectation)
<expression> ::= <expression arithmétique> |
<expression relationnelle> |
<expression logique>

Instruction & Expression. (suite)

expression arithmétique
description des calculs
ex : a+2 ;
Les Operateurs : +, -, /, *, %
les expressions arithmétiques sont valables sur les réels, les entiers, et les char.

Incrémentation

L'incrémentation appliquée aux variables permet de modifier la valeur de la variable sans affectation (effet de bord).

```
{ int b ;
  b = 1 ;
  b++ ;   post incrémentation ajoute 1 à b
}  valeur de b 2.
```

La post-incrémentation correspond à l'affectation
b = b+1 ; ≡ b++ ;

```
{ int a ;          b++ signifie : prendre la valeur de b
  int b ;          pour le calcul, puis incrémenter.
  a = 0 ;
  b = 1 ;
  a = 1+ b++ ;
}  b valeur 2.
   a valeur 2.
```

La pré-incrémentation incremente d'abord puis effectue le calcul de l'expression.

```
{ int a ;                    { b = 1 ;
  int b ;                      a = ++b + b++ ; }   ⚠ réponse imprévisible.
  a = 0 ;                    a = 3
  b = 1 ;                                          gcc toto → b = 2
  a = 1+ ++b ;  valeur de b 2.                     gcc -O2 toto → b = 3
}               valeur de a 3.
```