

# Programmation 1 – TD 1

La plupart des questions sont dues à Juliusz Chroboczek, que je remercie.

1. (Découverte de langages.) Pour chacun des programmes ou fragments de programmes suivants, indiquer : (a) ce qu'il fait (b) s'il s'agit de code en style impératif, fonctionnel, ou autre, et (c) dans quel langage de programmation il est écrit.

(a)           PROGRAM HELLO  
              WRITE (6,1)  
1            FORMAT 12HHELLO, WORLD  
              STOP  
              END

(b)           PROGRAM FACT  
              J=1  
              DO 1 I=1,10  
              J=J\*I  
1            CONTINUE  
              WRITE (6,2) J  
2            FORMAT I8  
              STOP  
              END

(c) 001 IDENTIFICATION DIVISION.  
002 PROGRAM-ID. 'HELLO'.  
003 ENVIRONMENT DIVISION.  
004 CONFIGURATION SECTION.  
005 SOURCE-COMPUTER. IBM-360.  
006 OBJECT-COMPUTER. IBM-360.  
0065 SPECIAL-NAMES.  
0066         CONSOLE IS CNSL.  
007 DATA DIVISION.  
008 WORKING-STORAGE SECTION.  
009 77 HELLO-CONST PIC X(12) VALUE 'HELLO,WORLD'.  
075 PROCEDURE DIVISION.  
090 000-DISPLAY.  
100         DISPLAY HELLO-CONST UPON CNSL.  
110         STOP RUN.

- (d) `10 J=1  
20 FOR I=1 TO 10  
30 J=J*I  
40 NEXT I  
50 PRINT J  
60 END`
- (e) `(defun fact (n)  
 (do* ((i 1 (+ i 1)) (j 1 (* j i)))  
 ((>= i n) j)))`
- (f) `(define (fact n)  
 (cond  
 ((<= n 1) 1)  
 (t (* n (fact (- n 1))))))`
- (g)  $\square \leftarrow */10$
- (h) `def factorial(n):  
 result = 1  
 for i in range(1, n+1):  
 result *= i  
 return result`
- (i) `function fact (n:integer):integer  
begin  
 var i, j : integer;  
 j:=1;  
 for i:=1 to n do  
 j := j*i;  
 fact := j  
end`
- (j) `int fact (int n)  
{  
 int i, j;  
  
 j = 1;  
 for (i=1; i<=n; i++)  
 j *= i;  
 return j;  
}`
- (k) `fact :: Int -> Int  
fact 1 = 1  
fact n = n * fact (n-1)`
- (l) `let rec fact n =  
 if n==1`

```

        then 1
        else n * fact (n-1);;
(m) fun fact n =
    if n=1
    then 1
    else n * fact (n-1);
(n) fact(1, 1).
    fact(N, M) :- N > 1, fact (N-1, M1), M=M1*N.
(o) counter=$1
    factorial=1
    while [ $counter -gt 0 ]
    do
        factorial=$(( $factorial * $counter ))
        counter=$(( $counter - 1 ))
    done
    echo $factorial
(p) : fact
    dup 1 = if
    else dup 1 - fact *
    endif ;
(q) /factorial {
    dup 1 eq {}{
    dup 1 sub factorial mul
    } ifelse
} def
(r) function Factorial (N : Positive) return Positive is
    Result : Positive := N;
    Counter : Natural := N - 1;
begin
    for I in reverse 1..Counter loop
        Result := Result * I;
    end loop;
    return Result;
end Factorial;

```

## 2. (Représentation des entiers.)

- (a) Combien de valeurs peut avoir un entier à 1 bit ? à 3 bits ? à  $n$  bits ?
- (b) Un grillage de 100 mètres a un poteau tous les mètres. Combien y a-t-il de poteaux ?
- (c) Quelles sont les valeurs possibles d'un entier naturel codé sur  $n$  bits ? d'un entier signé codé sur  $n$  bits ?
- (d) Effectuer l'addition sur 4 bits des couples d'entiers suivants :

- i. 0010 + 0110
  - ii. 0101 + 1010
  - iii. 1011 + 1101
  - iv. 1010 + 0110
  - v. 1111 + 1111
- (e) Donner les valeurs décimales correspondant aux opérations ci-dessus en décodant les entiers (a) de façon non signée d'abord et (b) en complément à deux ensuite.
- (f) La représentation en *complément à 1* code 0 comme 0000, 1 comme 0001, 2 comme 0010, etc., mais code les nombres négatifs comme la négation logique (le non) bit-à-bit de leur opposé : donc -1 est codé comme 1110, -2 comme 1101, -3 comme 1100, etc.
- i. Quel est le principal défaut de la représentation en complément à 1 ?
  - ii. A l'aide des exemples fournis à la question 2d, imaginer comment on doit implémenter l'addition en complément à 1. La question principale est : si les deux nombres à additionner engendrent une retenue à la fin du calcul, comment doit-on l'utiliser pour corriger le résultat final ?
  - iii. Pourquoi cette dernière opération de correction termine-t-elle ?
- (g) Quelles valeurs affiche le fragment de programme Java suivant ?
- ```
byte i = 101, j = 87, k = -101, l = -99;
byte m, n, o;
m = i+j; n = j+k; o = k+l;
System.out.println(m);
System.out.println(n);
System.out.println(o);
```

### 3. (Représentations des textes.)

- (a) Décoder la chaîne ASCII suivante (écrite en hexadécimal...c'est plus pratique !)
- ```
4a 65 61 6e 20 6e 69 65 20 6d 61 20 63 7a 61 74 75
```
- (b) L'ASCII n'est défini que pour les codes allant de 00 à 7f. Il y a des extensions la complétant de 80 à ff, mais elles dépendent de la *page*. La page dite Latin-1 (norme ISO 8859-1) en est une. Décoder la chaîne Latin-1 suivante :
- ```
55 6e 20 70 ea 63 68 65 75 72 20 e0 20 6c 61 20 61 69 67 6e 65
```
- (c) Un ami m'envoie un email, et il m'arrive comme suit. Que s'est-il passé ?
- ```
Je vais Ã_ SÃ"te cet Ã©tÃ©.
```
- (d) Unicode résout une grande partie des problèmes précédents... mais il y a plusieurs formats pour Unicode. Décoder la chaîne UTF-32 (UCS-4) suivante :
- ```
00 00 00 73 00 00 00 7a 00 00 00 63 00 00 00 7a
00 00 01 19 00 00 01 5b 00 00 00 63 00 00 00 69
00 00 00 65
```

Quelques caractères Unicode :

|        |                                        |
|--------|----------------------------------------|
| U+000A | LINE FEED (LF)                         |
| U+0020 | SPACE                                  |
| U+002C | COMMA                                  |
| U+0030 | DIGIT ZERO                             |
| U+0041 | LATIN CAPITAL LETTER A                 |
| U+0061 | LATIN SMALL LETTER A                   |
| U+00CA | LATIN CAPITAL LETTER E WITH CIRCUMFLEX |
| U+00EA | LATIN SMALL LETTER E WITH CIRCUMFLEX   |
| U+0105 | LATIN SMALL LETTER A WITH OGONEK       |
| U+0119 | LATIN SMALL LETTER E WITH OGONEK       |
| U+015B | LATIN SMALL LETTER S WITH ACUTE ACCENT |

(e) Quels sont les défauts du codage UTF-32 ?

(f) Le codage UTF-8 code les caractères Unicode comme suit :

- U+0000 à U+007F : 0xxxxxxx
- U+0080 à U+07FF : 110xxxxx 10xxxxxx
- U+0800 à U+FFFF : 1110xxxx 10xxxxxx 10xxxxxx
- U-10000 à U-1FFFFFF : 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Décoder la chaîne UTF-8 suivante :

6a 61 6b 20 7a 65 20 6d 6e c4 85 0a

(g) UTF-8 souffre-t-il des défauts d'UTF-32 trouvés plus haut ? Comment résout-il ceux qu'il résout ?

(h) On trouve sur la page des conférences de rentrée 2015 (<http://www.dptinfo.ens-cachan.fr/Conferences/conferences15.php>) le texte suivant :

Pierre-Alain Fouque

Aspects Algorithmiques de Cryptanalyse

Dans cette conférence, je présenterai quelques algorithmes importants en cryptanalyse au travers d'exemples issus de cryptographie symétrique et asymétrique comme par exemple : les attaques génériques sur les fonctions de hachage, les algorithmes de factorisation ou du logarithme discret, les schémas de signature construits sur la difficulté à résoudre des systèmes quadratiques en plusieurs variables ou la difficulté à résoudre des systèmes linéaires avec du bruit.

En coulisses, le serveur `projects.lsv.ens-cachan.fr` a envoyé à mon navigateur Web le texte suivant (extrait) :

```
<h2> <a name="Pierre-Alain_Fouque"
      href= "http://www.di.ens.fr/~fouque/" >
      Pierre-Alain Fouque </a>
</h2>
<p><big><b>
```

**<a> Aspects Algorithmiques de Cryptanalyse </a>  
</b> </big>**

</p>

<p>

Dans cette conf&eacute;rence, je pr&eacute;sent&eacute;rai quelques algorithmes importants en cryptanalyse au travers d'exemples issus de cryptographie sym&eacute;trique et asym&eacute;trique comme par exemple: les attaques g&eacute;n&eacute;riques sur les fonctions de hachage, les algorithmes de factorisation ou du logarithme discret, les sch&eacute;mas de signature construit sur la difficult&eacute; &agrave; r&eacute;soudre des syst&eacute;mes quadratiques en plusieurs variables ou la difficult&eacute; &agrave; r&eacute;soudre des syst&eacute;mes lin&eacute;aires avec du bruit.

</p>

- i. En quoi le langage utilis&eacute; fournit-il une autre solution aux probl&eacute;mes d'encodage de langages au-del&agrave; de ce que fournit l'ASCII ?
- ii. Comment s'appelle le langage utilis&eacute; ?

(i) Je vais sur la page Web <https://projects.lsv.ens-cachan.fr/topology/wp-admin/post.php?post=251&action=edit> et je vois :

Now remember that  $(x_i)_{i \in I}$  converges to  $x$  if and only if every open subset  $U$  that contains  $x$  is such that  $x_i$  is eventually in  $U$ . One obtains an equivalent definition by stating that every neighborhood  $A$  of  $x$  (i.e., in  $N_x$ ) is such that  $x_i$  is eventually in  $A$ . In other words, if and only if  $N_x$  is included in the convergence filter of the net.

En coulisses, le serveur [projects.lsv.ens-cachan.fr](https://projects.lsv.ens-cachan.fr) a envoy&eacute; &agrave; mon navigateur Web le texte suivant (extrait) :

Now remember that  $(x_i)_{i \in I}$  converges to  $x$  if and only if every open subset  $U$  that contains  $x$  is such that  $x_i$  is eventually in  $U$ .

One obtains an equivalent definition by stating that every neighborhood  $A$  of  $x$  (i.e., in  $N_x$ ) is such that  $x_i$  is eventually in  $A$ . In other words, if and only if  $N_x$  is included in the convergence filter of the net.

En quoi cela va-t-il au-delà des possibilités d'Unicode ?

4. Revenons à l'exemple de la conférence de Pierre-Alain Fouque citée plus haut. Le fichier complet commence par :

```
<?
$EXTRA_HEAD="antispam.html";
$ARG_BODY="onload=\"onLoad()\"";
SETLANG("fr");
STYLEDPTINFO();
HEAD("Conférences de rentrée 2015");
ADDTITLE("Conférences de rentrée 2015");
MKPAGEDPTINFO();
?>
```

Ce n'est pas du HTML. Qu'est-ce que c'est ? Peut-on imaginer ce que c'est censé faire ?