

UEF 1 : Informatique & Programmation

Faculté des Sciences de Nice
DEUG 2001-2002

Jérôme DURAND-LOSE
Jean-Paul ROY

COURS 8

Itérations et Manipulation de texte

8-2

Tâches répétitives dont on connaît le nombre

À chacune des 35 personnes
envoyer un carton d'invitation

10 fois de suite
avancer de 50
tourner de 144 degrés

Pour les positions 1 à 10
faire ...

Ce n'est plus *tant qu'une condition est vérifiée*,
mais *pour un nombre de fois connu à l'avance*

8-3

Boucle for

La variable compteur n'existe
que dans la boucle. Elle n'a plus
aucun sens après

```
for ( int compteur = valeur initiale ;  
      compteur <= valeur finale ;  
      compteur augmente de 1 ) {  
    << instructions >>  
}
```

Application avec une tortue :
7 fois de suite
Avancer de 100
Tourner de $3 \times 360.0 / 7$

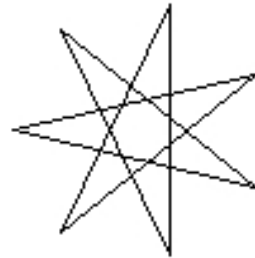
Pour faire 3 tours
en 7 étapes

8-4

Boucle for

```
Turtle etoile = new Turtle();  
  
for ( int i = 1 ; i <= 7 ; i ++ ) {  
    etoile.forward( 100 );  
    etoile.left( 3 * 360 / 7.0 );  
}
```

i vaut donc 1,
puis 2,
puis 3,
...
puis 7



8-5

Parcourir les caractères d'une chaîne

Une chaîne de caractères \Rightarrow
suite de caractères de longueur connue

Exemple :

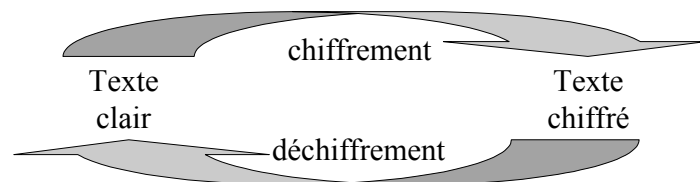
Connaître le nombre d'espace \Rightarrow parcourir la chaîne

Exemple plus compliqué :

chiffrer et déchiffrer avec un code simple

8-6

Chiffre de Cæsar



Clé : « H »

Les majuscules sont remplacées selon

A \rightarrow H, B \rightarrow I, C \rightarrow J ... S \rightarrow Z, T \rightarrow A, ... Z \rightarrow G

Les minuscules et les autres caractères ne sont pas codés

8-7

Exemple

« LE RIRE EST LE PROPRE DE L'HOMME »

« SL YPYL LZA SL WYVWYL KL S'OVTTL »

Algorithme de chiffrement

Variable

résultat : chaîne de caractères = chaîne vide

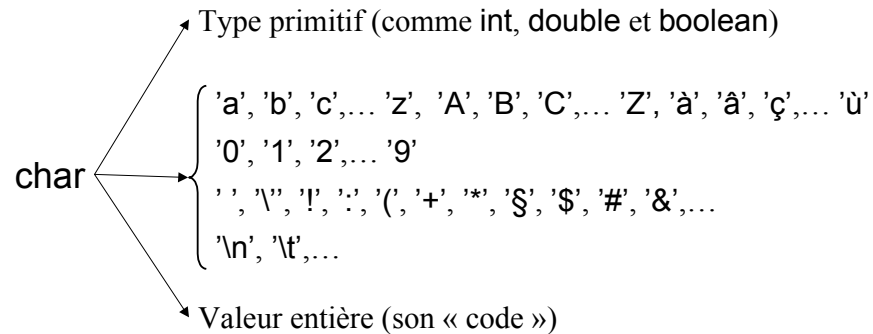
Algorithme

Prendre les caractères du mot à coder **du premier au dernier**
Coder le caractère et l'ajouter à la fin du résultat

8-8

Qu'est-ce qu'un caractère ?

"Alerte !" \longleftrightarrow (String) Chaîne de caractères (char)



Codage comme entier (ASCII sur 7 bits, unicode sur 2 octets,...)
Entièrement géré par l'ordinateur (heureusement !)

8-9

Utilisation des caractères

```
char c = 'A';
boolean b = ( c == 'a' );
char cPlusUn = (char) (c + 1);
System.out.println( "c=" + c + " b=" + b + " c+1=" + cPlusUn );
System.out.println( "code de A = " + (int) 'A' );
System.out.println( "Charmant".charAt ( 0 ) );
```

c=A b=false c+1=B
code de A = 64
C

Ordre sur les caractères : 'A' < 'Z' < 'a' < 'z' et '0' < '1' < '9'
(le codage n'est pas innocent)

8-10

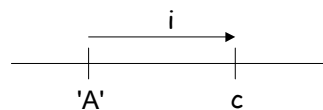
Majuscule vers 0..25

Soit **c** un caractère, une lettre majuscule

Décalage 'A' → 'H', 'B' → 'I', 'C' → 'J', ... 'S' → 'Z'
soudis pour 'T', 'U', ... et 'Z'
⇒ calcul sur 0..25 modulo 26

$c \equiv 'A' + i$ avec $0 \leq i \leq 25$

⇒ $i \equiv c - 'A'$



i représente l'écart entre les codes de **c** et de 'A'

8-11

Décalage

Soit *i* cette valeur entre 0 et 25

Décalage 'A' → 'H' \Leftrightarrow Ajouter 'H' - 'A' \equiv **decal**
 $i \leftarrow (i + \text{decal}) \text{ modulo } 26$

Pour un char **c**

$(\text{char}) ('A' + ((c - 'A') + \text{decal}) \% 26)$

L'addition est sur int

Les parenthèse sont-elles utiles ?

Oui, par sécurité, car il se méfier des priorités des opérateurs

8-12

Fonction de décalage et de chiffrement

```

/** Fonction retournant une lettre majuscule décalée circulairement. */
static char decale ( char lettre, char clef ) {
    if ( ( lettre < 'A' ) || ( lettre > 'Z' ) )
        return lettre;    // ce n'est pas une majuscule
    return (char) ( 'A' + ( ( lettre - 'A' ) + ( clef - 'A' ) ) % 26 );
}

/** Fonction retournant le texte chiffré selon le chiffre de César
seules les majuscules sont chiffrées. */
static String chiffre ( String texte, char clef ) {
    String texteChiffre = "";
    for ( int i = 0; i < texte.length(); i ++ )
        texteChiffre = texteChiffre + decale ( texte.charAt( i ), clef );
    return texteChiffre;
}

```

8-13

Table de codage

Pour chaque majuscule
Afficher en quoi elle est transformée

```

/** Procédure affichant une table
pour un (dé)chiffrement/vérification manuel */
static void afficheTable ( char clef ) {
    for ( char c = 'A'; c <= 'Z'; c ++ ) {
        System.out.println( c + " => " + chiffre ( c, clef ) );
    }
}

```

A => H
B => I
C => J
D => K
E => L
F => M
G => N
H => O
I => P
J => Q
K => R
L => S
M => T
N => U
O => V
P => W
Q => X
R => Y
S => Z
T => A
U => B
V => C
W => D
X => E
Y => F
Z => G

8-14

Complexité ?

1 test, 1 affectation, 1 multiplication... temps « constant »

Le codage et le décodage se font
en temps proportionnel à la longueur du message

Une chaîne de longueur 1000 fait
100 fois plus d'opérations
qu'une chaîne de longueur 10

On parle d'algorithme *linéaire*

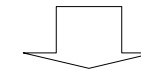
8-15

for plus généralement

```

for ( <init> ; <cond> ; <màj> ) {
    <instructions>
}

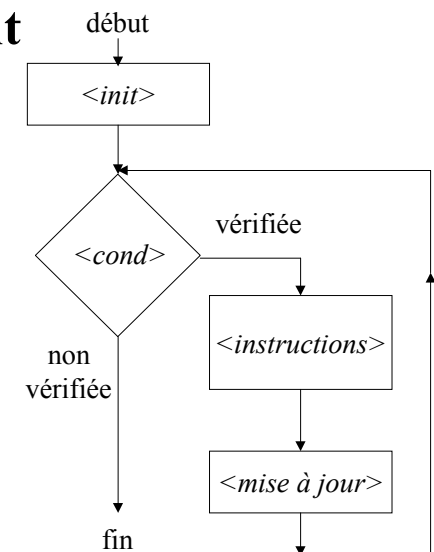
```



```

<init>
while ( <cond> ) {
    <instructions>
    <màj>
}

```



8-16