TD d'Arthur Blaise

Arthur Blaise

nov. 08, 2019

Contenu:

1 Introduction à l'algorithmique **3**
- 1.1 TD Eléments de base – Structures . . . 3
- 1.2 TD Structures - Les classes . . . 6
- 1.3 TD Récursivité . . . 8
- 1.4 TD Unix - Commandes de base . . . 11
- 1.5 TD Binaire - Complexité . . . 13
- 1.6 Maths Appliquées . . . 14

2 Pascal **15**
- 2.1 Premier TP . . . 15
- 2.2 TP Sup . . . 21
- 2.3 Encore plus loin . . . 27
- 2.4 Les tableaux dynamiques . . . 28
- 2.5 Les pointeurs . . . 32

3 Suite du TP Unix **39**
- 3.1 TP 2 . . . 39

4 Introduction au C **43**
- 4.1 TD 1 : Entrées, sorties . . . 43
- 4.2 TD 4 : Vecteurs 2D . . . 44

Moi c'est Arthur Blaise. Et là, c'est l'endroit où je stocke une partie des travaux à rendre.

Vous pourrez donc trouver ici certains TD ou autres exercices divers et variés créés pendant ma scolarité. Une version pdf de ce site est également téléchargeable, ainsi qu'une version Epub (utilisable par exemple dans les applications de liseuses, telle qu'iBook chez Apple).

CHAPITRE 1

Introduction à l'algorithmique

1.1 TD Eléments de base – Structures

1.1.1 Exercices sur les pièces (#13)

Consigne : *Entrer des valeurs de pièces (0.1€, 0.2€, 0.5€, 1€, 2€). Contrôler la saisie et arrêter quand la valeur de la pièce est nulle. Puis, constituer des rouleaux de 10 pièces de même valeur et des sacs de 20 rouleaux. Afficher alors, pour chaque type de pièces, le nombre de rouleaux, de sacs et de pièces non emballées.*

```
PROGRAMME tic-tac-boum
VARIABLES
    n1,n2,n5,n10,n20 : ENTIER  //nombre de pièces par montant
    r1,r2,r5,r10,r20 : ENTIER  //nombre de rouleaux
    s1,s2,s5,s10,s20 : ENTIER  //nombre de sacs
DEBUT
    n2,n5,n10,n20 <- 0,0,0,0,0
    //on récupère le nombre de pièces de 0.1e
    ECRIRE("Nombre de pièces de 0.1€")
    LIRE(n1)
    SI (n1<>0) ALORS :
        //idem pour les pièce de 0.2e
        ECRIRE("Nombre de pièces de 0.2€")
        LIRE(n2)
        SI (n2<>0) ALORS :
            //idem pour 0.5e
            ECRIRE("Nombre de pièces de 0.5€")
            LIRE(n5)
            SI (n5<>0) ALORS :
                //idem pour 1e
                ECRIRE("Nombre de pièces de 1€")
                LIRE(n10)
                SI (n10<>0) :
                    //idem pour 2e
```

(suite sur la page suivante)

(suite de la page précédente)

```
                    ECRIRE("Nombre de pièces de 2€")
                    LIRE(n20)
                FIN SI
            FIN SI
        FIN SI
    FIN SI

    //r1 récupère le nombre de rouleaux complets de 0.1 et n1 le reste,
    // le tout grâce à une division euclidienne
    r1,n1 <- n1 DIV 10, n1 MOD 10
    //s1 récupère le nombre de sacs complets de 0.1e et r1 tout les rouleaux restants
    s1,r1 <- n1 DIV 10, n1 MOD 20
    //la suite des instructions est sur la même base, seuls les valeurs changent
    r2,n2 <- n2 DIV 10, n2 MOD 10
    s2,r2 <- r2 DIV 10, r2 MOD 20
    r5,n5 <- n5 DIV 10, n5 MOD 10
    r5,n5 <- r5 DIV 10, r5 MOD 20
    r10,n10 <- n10 DIV 10, n10 MOD 10
    s10,r10 <- r10 DIV 10, r10 MOD 20
    r20,n20 <- n20 DIV 10, n20 MOD 10
    s10,r20 <- r20 DIV 10, r20 MOD 20

    //puis on affiche le résultat dans un texte sur plusieurs lignes
    ECRIRE("Il y a ",s1," sacs ",r1," rouleaux ",p1," pièces de 0.1€, \n",
                    s2," sacs ",r2," rouleaux ",p2," pièces de 0.2€, \n",
                    s5," sacs ",r5," rouleaux ",p5," pièces de 0.5€, \n",
                    s10," sacs ",r10," rouleaux ",p10," pièces de 1€, et \n",
                    s20," sacs ",r20," rouleaux ",p20," pièces de 2€")
FIN
```

Plusieurs améliorations sont possible, comme par exemple créer une fonction qui retourne le nombre de rouleaux et de sacs possibles à partir du nombre de pièces, ou utiliser une boucle POUR afin de réduire la taille du premier bloc de code, comme ceci :

```
[...]
DEBUT
    n2,n5,n10,n20 <- 0,0,0,0,0
    POUR i DANS [(n1,0.1),(n2,0.2),(n3,0.5),(n4,1),(n5,2)] FAIRE :
        ECRIRE("Nombre de pièces de ",i[1],"€")
        LIRE(i[0])
        SI (i[0]=0) ALORS:
            BREAK
    [...]
```

1.1.2 Exercice de l'horloge (#14)

Consigne : *Créer un compte à rebours (heures, minutes, secondes) qui affichera «boum» à 00 :00 :00*

```
PROGRAMME tic-tac-boum
VARIABLES
    h,m,s : ENTIER  //heures, minutes et secondes restantes
DEBUT
    ECRIRE("Saisissez le temps restant (heures, minutes et secondes) : ")
    LIRE(h,m,s)
    TANT QUE (h<>0 OU m<>0 OU s<>0) :  //tant qu'il reste du temps
```

(suite sur la page suivante)

(suite de la page précédente)

```
        s <- s-1
        SI (s<0 ET (m>0 OU h>0) ALORS :  //si la minute est terminée
            m <- m-1
            s <- 59
            SI (m<0 ET h>0) ALORS :  //si l'heure est terminée
                h <- h-1
                m <- 59
            FIN SI
        FIN SI
        ECRIRE(h,":",m,":",s)
    FIN TANT QUE  //fin du décompte
    ECRIRE("BOUM !")
FIN
```

On pourra éventuellement ajouter avant la boucle TANT QUE une sécurité permettant d'empêcher un nombre supérieur à 59 de minutes ou de secondes :

```
TANT QUE (s>59) :
    s <- s-60
    m <- m+1
FIN TANT QUE
TANT QUE (m>59) :
    m <- m-60
    h <- h+1
FIN TANT QUE
```

Enfin, pour plus de réalisme, il est possible d'ajouter une instruction demandant au code d'attendre une seconde, dans la boucle TANT QUE. Cela permettra d'attendre une seconde entre deux décomptes, au lieu de tout afficher quasi-instantanément.

1.1.3 PGCD

Consigne : *Ecrire une fonction permettant de calculer le plus grand diviseur commun de deux entiers naturels.*

```
FONCTION PGCD(a:INT, b:INT) :INT
VAR I,m : INT
DEBUT
    m = 0
    POUR I DE 1 A a:
        SI (a MOD I = 0) ET (b MOD I = 0) ALORS:
            m <- I
            FIN SI
    FIN POUR
    RETOURNER(m)
FIN


PROGRAMME pgcd
VAR x,y,r : ENTIER
DEBUT
    ECRIRE("Saisissez les deux entiers positifs")
    LIRE(x,y)
    r <- PGCD(x,y)
    ECRIRE("Le PGCD de ",x," et ",y," est ",r)
FIN
```

1.2 TD Structures - Les classes

1.2.1 Résolution du second degré

Consigne : *Définir en algorithmique, une structure nommée Eq2D qui permet de définir l'ensemble des solutions d'une équation du second degré.*

```
TYPE
STRUCTURE Eq2D
    r_one:float //première racine
    r_two:float //deuxième racine
    delta:float

    PROCEDURE calc (self,a:float, b:float, c:float)
    DEBUT
        self.delta <- b**2 - 4*a*c
        SI self.delta < 0 ALORS
            self.r_one,self.r_two = None
        SINON SI self.delta > 0 ALORS
            self.r_one <- (-b-sqrt(self.delta))/(2*a)
            self.r_two <- (-b+sqrt(self.delta))/(2*a)
        SINON
            self.r_one,self.r_two <- -b/(2*a)
        FIN SI
    FIN

PROGRAMME superSoluce
VAR a,b,c:float
        s:Eq2D
DEBUT
    ECRIRE("Entrez les coefficients")
    SAISIR(a,b,c)
    s.calc(a,b,c)
    ECIRE("Les deux solutions sont",s.r_one,"et",s.r_two)
    FIN
```

On peut aussi définir une fonction qui va calculer ces racines, au lieu d'un programme :

```
FONCTION superSoluce (a:float, b:float, c:float) : Eq2D
VAR s:Eq2D
DEBUT
    s.calc(a,b,c)
        RETOURNER(s)
FIN
```

1.2.2 Opérations complexes

Consigne : *Ecrire deux fonctions qui permettent de passer un nombre complexe de sa forme algébrique à sa forme exponentielle, puis d'additionner deux complexes sous leur forme algébrique.*

```
TYPE
    STRUCTURE algComplex
        reel:float
        img:float
```

(suite sur la page suivante)

(suite de la page précédente)

```
    STRUCTURE expComplex
        arg:float
        modul:float


FONCTION switch(a:algComplex) : expComplex //passer de la forme algébrique à la forme␣
↪exponentielle
VAR x:expComplex
    acos:float  //angle trouvé par cosinus
        asin:float //angle trouvé par sinus
DEBUT
    x.modul <- sqrt(a.reel**2 + a.img**2)
    t_one <- arccos(a.reel/x.modul)/pi
    t_two <- arcsin(a.img/x.modul)/pi
    SI -t_one <- t_two ALORS:  //si l'angle trouvé par cosinus est l'opposé de l
↪'angle trouvé par sinus
        x.arg <- t_two  //l'angle final prend la valeur du sinus
    SINON:
        x.arg <- arccos(cos(-t_one))  //l'ange final prend la valeur de l'angle dont␣
↪le cosinus est l'opposé du sinus
    FIN SI
    RETOURNER(x)
FIN

FONCTION switch2(a:expComplex) : algComplex //passer de la forme exponentielle à la␣
↪forme algébrique
VAR x:algComplex
DEBUT
    x.reel = a.modul*cos(a.arg)
        x.img = a.modul*sin(a.arg)
        RETOURNER(x)
FIN

FONCTION algAdd(a:algComplex, b:algComplex) : algComplex //additionner deux formes␣
↪algébriques
VAR x:algComplex
DEBUT
    x.reel <- a.reel + b.reel  //on additionne les réels
    x.imaginaire <- a.imaginaire + b.imaginaire //on additionne les imaginaires
    RETOURNER(x)
FIN

FONCTION expAdd(a:expComplex, b:expComplex) : expComplex //additionner deux formes␣
↪exponentielles
VAR alga:algComplex //a, sous forme algébrique
    algb:algComplex //idem pour b
    algsomme:algComplex //la somme sous forme algébrique
DEBUT
    alga <- switch2(a)
        algb <- switch2(b)
        algsomme <- algAdd(alga,algb)
        RETOURNER(switch(aglsomme))
FIN
```

Je ne suis pas sûr de la formule pour calculer l'argument d'un complexe à partir de sa valeur algébrique, mais après tout… c'est un cours d'informatique, pas de maths, n'est-ce pas ?

1.3 TD Récursivité

1.3.1 Factorielle itérative

Consigne : *Créer une fonction qui calcule la factorielle d'un nombre en utilisant une itération.*

```
FONCTION fact (n:int) : int
VAR i:int //itération
    s:int //total
DEBUT
    s <- 1
    POUR i de 1 à n:
        s <- s*i
    FIN POUR
        RETOURNER(s)
FIN
```

1.3.2 Division euclidienne

Consigne : *Écrire deux fonctions récursives qui calculent le quotient et le reste de la division entière de deux nombres strictement positifs.*

```
FONCTION quotient(a,b,c,i:int) : int
VAR r:int //résultat
DEBUT
    SI a<i OU b<i ALORS
        RETOURNER(c)
    SINON SI a*i<=b OU b*i<=a ALORS
        c <- i
    FIN SI
    r <- Quotient(a,b,c,i+1)
    RETOURNER(r)
FIN

FONCTION reste(a,b,c,i:int) : int
VAR r:int //résultat
DEBUT
    SI a<i OU b<i ALORS
        RETOURNER(c)
    SINON SI a*i<=b OU b*i<=a ALORS
        c <- ABS(MIN(a*i-b , b*i-a))
    FIN SI
    r <- Reste(a,b,c,i+1)
    RETOURNER(r)
FIN
```

1.3.3 Palindromes

Consigne : *Écrire un algorithme récursif qui permet de vérifier si une chaîne de caractères est un palindrome ou non.*

```
FONCTION palindrome(text:string, n:int) : bool
DEBUT
    text <- REPLACE(text," ","")
```

(suite sur la page suivante)

(suite de la page précédente)

```
    SI n > len(text)/2 ALORS
        RETOURNER(Vrai)
    SINON SI text[n]=text[-n-1] ALORS
        RETOURNER(Palindrome(text,n+1)
    SINON
        RETOURNER(Faux)
    FIN SI
FIN
```

On suppose que la fonction REPLACE prend en paramètre trois chaines de caractères et retourne une copie de la chaîne de caractères dans laquelle les occurrences de la deuxième ont été remplacées par la troisième.

1.3.4 Miroir

Consigne : *Écrire un algorithme récursif qui permet de réaliser cette fonction, et puis proposer une version itérative équivalente.*

Fonction récursive :

```
FONCTION miroir(text:string, n:int, rep:string) : string
DEBUT
    SI n = len(text) ALORS
        RETOURNER(rep)
    FIN SI
    rep <- text[n]+rep
    RETOURNER(Miroir(text,n+1,rep))
FIN
```

Fonction itérative :

```
FONCTION miroir2(text:string) : string
VAR i:int //itération
    rep:string //résultat
DEBUT
    POUR i DANS text:
        rep <- i+rep
    FIN POUR
    RETOURNER(rep)
FIN
```

1.3.5 Power

Consigne : *Écrire la fonction récursive puissance qui calcule n^k de façon naïve en considérant que n^k= n*n(k-1) et que n^0 = 1*

```
FONCTION naive(n,k,i : int) : int
DEBUT
    SI k=0 ALORS
        RETOURNER(1)
    SINON SI i<k-1 ALORS
        RETOURNER(Naive(n,k,i+1)*n
    SINON
        RETOURNER(n)
```

(suite sur la page suivante)

(suite de la page précédente)

```
    FIN SI
FIN
```

Puis la même fonction, en récursion terminale.

```
FONCTION term(n,k,acc : int) : int
DEBUT
    SI k=0 ALORS
        RETOURNER(acc)
    SINON
        RETOURNER(Term(n,k-1,acc*n))
    FIN SI
FIN
```

Puis par dichotomie.

```
FONCTION puissance-dic(n,k,i : int) : int
DEBUT
    SI k=0 ALORS
        RETOURNER 1
    SINON SI k MOD 2 = 0 ALORS
        SI i*2 = k ALORS
            RETOURNER((n*n)^(1/2))
        SINON
            RETOURNER(puissance-dic(n,k,i+1)^2)
        FIN SI
    SINON
        SI i = k-1 ALORS
            RETOURNER(n)
        SINON
            RETOURNER(puissance-dic(n,k,i+1)*n)
        FIN SI
    FIN SI
FIN
```

1.3.6 Parité

Consigne : *Un nombre n est pair si n-1 est impair, et un nombre n est impair si n-1 est pair. Partant de cette définition, créez deux fonctions récursives pair(n) et impair(n) qui permettent ensemble de savoir si un nombre entré est pair ou impair.*

```
FONCTION pair(n,i=0 : int) : bool
DEBUT
    SI i=n
        RETOURNER(True)
    SINON
        RETOURNER(impair(n,i+1))
    FIN SI
FIN

FONCTION impair(n,i=0 : int) : bool
DEBUT
    SI i=n
        RETOURNER(False)
    SINON
```

(suite sur la page suivante)

(suite de la page précédente)

```
        RETOURNER(pair(n,i+1))
    FIN SI
FIN
```

Une autre méthode permet cependant d'écrire ce code en une seule fonction :

```
FONCTION pair(n,i=0 : int ,c=True : bool) : bool
DEBUT
    SI i=n ALORS
        RETOURNER(c)
    SINON
        RETOURNER(n,i-1,not c)
    FIN SI
FIN
```

1.4 TD Unix - Commandes de base

1.4.1 Exercice 1

a.

```
    $ echo $PATH $PS1
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\[\033[01;
↪32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$

    $ export PS1="linux on en veux encore # > "

    #On peut même changer la couleur !
    $ export PS1="\e[0;36m\! - linux on en veux encore # > \e[m"

    $ export PS1="\e[0;36m\!\e[m - \e[0;32m\w # \e[m"
```

c. La commande `man ls` fonctionne, et affiche un beau pavé de texte explicatif.

```
    $ cal 10 2018
    Octobre 2018
di lu ma me je ve sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

    $ who
blaisearth tty2         2018-10-24 10:40 (:1)
    $ whoami
blaisearth
```

f. La commande `stty -echo` permet de rentrer du texte sans l'afficher à l'écran, comme par exemple un mot de passe.

g. On peut confirmer, `stty echo` nous permet de voir à nouveau les commandes entrées. C'est plus partique pour continuer le TP...

```
    $ xterm -hc blue -bg cyan -fg brown

    $ which python3
/usr/bin/python3

    $ alias hop=ls`

    $ alias rs="rm -i"

    $touch test.txt
    $rm test.txt
rm : supprimer fichier vide 'test.txt' ? y
```

1.4.2 Exercice 2

```
   $ emacs
   #appui sur les touches Ctrl-Z
   $ bg

   $ sleep 60
   #Cette commande demande au processus en cours d'attendre 60 secondes sans rien␣
↪faire.
   #Pour l'interrompre, il suffit d'appuyer sur les touches Ctrl-C

   $ ps
 PID TTY          TIME CMD
5092 pts/0    00:00:00 bash
5159 pts/0    00:00:00 emacs
5202 pts/0    00:00:00 ps
   $ kill 5159
```

1.4.3 Exercice 3

```
    $ pwd
/cergy/homee/b/blaisearth
    $ cd /
    $ ls
bin    dev   initrd.img      lib64       mnt   root  srv  usr      vmlinuz.old
boot   etc   initrd.img.old  lost+found  opt   run   sys  var      vms
cergy  home  lib             media       proc  sbin  tmp  vmlinuz
    $ cd ~
    $ mkdir Prepal
    $ mkdir Prepal/Info
    $ cd Prepal/info
    $ mkdir tmp
    $ cd tmp
    $ touch toto
    $ $ find ../.. | sed 's/[^/]*\//| /g;s/| *\([^| ]\)/+--- \1/'
+--- ..
| +--- Info
```

(suite sur la page suivante)

(suite de la page précédente)

```
| | +--- tmp
| | | +--- toto
    $ cd ..
    $ rmdir tmp
rmdir: impossible de de supprimer 'tmp': Le dossier n est pas vide
    $ rm -r tmp

    $ head -n 15 lorem.txt
[...]
    $ head -n 9 lorem.txt >temp.txt
    $ tail -n 5 temp.txt >temp2.txt
    $ head -n 13 lorem.txt >temp.txt
    $ tail -n 1 temp.txt >temp3.txt
    $ cat temp2.txt temp3.txt >final.txt

    $ cat final.txt
    $ more final.txt
    #Aucune différence visible

    $ mkdir tmp
    $ cp final.txt tmp/

    $ cd tmp
    $ cp final.txt "lorem en ipsum".txt
    $ mv final.txt ~/.local/share/Trash/files #Corbeille de l ordinateur
    $ mv "lorem en ipsum.txt" ~/
    $ ls ~/
Bureau     Images              Modèles  Prepal  Téléchargements  VirtualBox VMs
Documents  lorem en ipsum.txt  Musique  Public  Vidéos

    $ ln -s "/cergy/homee/b/blaisearth/lorem_en_ipsum.txt" ipsum
    $ ls -A --author -F -h -H -i -s
    $ ls -l
total 1
lrwxrwxrwx 1 blaisearth users 44 oct.  24 12:28 ipsum -> /cergy/homee/b/blaisearth/
↪lorem_en_ipsum.txt
    $ rm ~/lorem_en_ipsum.txt
    $ ls -l
    #La direction de l'alias est colorée en rouge, annoncant que le lien est rompu
```

1.5 TD Binaire - Complexité

Quel serait l'intervalle réel représentable par un codage hypothétique en « quadruple précision », c'est à dire en 128 bits ?

Sur 128 bits, selon la norme IEEE 754, l'exponentielle est codée sur 15 bits et la mantisse sur 113 bits. Pour éviter d'avoir des nombres trop longs, nous travaillerons avec la base 16.

Les valeurs minimales et maximales de l'exponentielle sont donc celles que nous pouvons coder le mieux sur 15 bits, soient $0001_{(16)}3FFF_{(16)} = 16382(10)$ et $7FFE_{(16)}3FFF_{(16)} = 16383_{(10)}$.

De là, pour obtenir la plus petite valeur positive, il suffit de calculer 2^{16382} (car on considère la mantisse à 0), soit environ 3.3610^{4932}. La valeur maximale est calculée avec une exponentielle à 16383, et une mantisse à 2^{112}, ce qui donne $2^{16383}(22^{112}) = 1.1910^{4932}$.

Les bornes des réels représentables par un codage en quadruple précision, aussi appelé binary128, sont donc approximativement 3.3610^{4932} et 1.1910^{4932} pour la partie positive. Pour la partie négative, il suffit de les multiplier par -1.

1.5.1 Comparaisons et affectations

Voici une fonction pour vérifier si une liste est triée (peu importe l'ordre), et une procédure pour insérer une valeur dans une liste triée en ordre croissant.

```
function check(A:intArray) : boolean;
var i:integer;
begin
    check := True;
    for i:=0 to high(A)-2 do
        check := check and ((A[i]-A[i+1]) * (A[i+1]-A[i+2]) >= 0);
end;

procedure Insert(var A:intArray;val:integer);
var i:integer;
    temp:intArray;
begin
    i := 0;
    SetLength(temp,length(A)+1);
    while A[i] < val do begin
        temp[i] := A[i];
        i := i+1;
    end;
    temp[i] := val;
    while i < high(A)+1 do begin
        temp[i+1] := A[i];
        i := i+1;
    end;
    A := temp;
end;
```

1.6 Maths Appliquées

1.6.1 Cryptage César

Voilà comment crypter un message via le code César, en python. Pour le décrypter, il suffit de rentrer l'opposé de la clé (-3 pour Cicéron par exemple).

```
alpha = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' # Clé de cryptage

def crypte(sent,key=10): # fonction à appeler
    answer = ''
    for i in sent.upper(): # on mélange majuscules et minuscules
        if i in alpha: # si le caractère est dans la clé
            answer += alpha[(alpha.index(i)+key)%26]
        else: # sinon (comme un espace ou une ponctuation)
            answer += i
    return answer
```

CHAPITRE 2

Pascal

Voici la page où je mettrai quelques travaux codés en Pascal. Ni plus ni moins.

2.1 Premier TP

```
(*
 ------------------------------------------------------------------------------------
 -- Fichier           : TP1.pas
 -- Auteur            : Florent Devin <fd@eisti.eu>
 -- Date de creation  : Mon Nov 13 13:20:06 2017
 --
 -- But               : Premier TP
 -- Remarques         : Aucune
 -- Compilation       : fpc
 -- Edition des liens : fpc
 -- Execution         : shell
 ------------------------------------------------------------------------------------
 *)
PROGRAM Tp1;

(*
 ------------------------------------------------------------------------------------
 -- Fonction          : LireEntier() : Integer
 -- Auteur            :
 -- Date de creation  :
 --
 -- But               : Lire un entier
 -- Remarques         : Aucune
 -- Pré conditions    : Aucune
 -- Post conditions   : Lire un entier
 ------------------------------------------------------------------------------------
↪*)
FUNCTION LireEntier() : Integer;
```

(suite sur la page suivante)

(suite de la page précédente)

```
VAR
   resultat : Integer;
BEGIN
    write('>>> ');
    read(resultat);
    LireEntier := resultat;
END;

(*
 -----------------------------------------------------------------------------------------
 -- Fonction          : Max4(nb1, nb2, nb3, nb4 : Integer) : Integer
 -- Auteur            :
 -- Date de creation  :
 --
 -- But               : Retourne le maximum de 4 nombres
 -- Remarques         : Aucune
 -- Pré conditions    : Aucune
 -- Post conditions   : retourne le plus grand des nombres passés en paramètre
 -----------------------------------------------------------------------------------------
↪*)
FUNCTION Max4(nb1, nb2, nb3, nb4 : Integer) : Integer;
BEGIN
    if (nb1>=nb2) and (nb1>=nb3) and (nb1>=nb4) then
    BEGIN
        Max4 := nb1
    END;
    if (nb2>=nb1) and (nb2>=nb3) and (nb2>=nb4) THEN
    BEGIN
        Max4 := nb2
    End;
    if (nb3>=nb1) and (nb3>=nb2) and (nb3>=nb4) then
    BEGIN
        Max4 := nb3
    END;
    if (nb4>=nb1) and (nb4>=nb2) and (nb4>=nb3) then
    BEGIN
        Max4 := nb4
    END;
END;

(*
 -----------------------------------------------------------------------------------------
 -- Procedure         : AfficheMax4()
 -- Auteur            : Florent Devin <fd@eisti.eu>
 -- Date de creation  : Mon Nov 13 14:12:09 2017
 --
 -- But               : Afficher le maximum de 4 nombre
 -- Remarques         : Aucune
 -- Pré conditions    : Aucune
 -- Post conditions   : Afficher le maximum de 4 nombre
 -----------------------------------------------------------------------------------------
↪*)
PROCEDURE AfficheMax4();
VAR
   nb1, nb2, nb3, nb4 : Integer;
BEGIN
   nb1 := LireEntier();
```

(suite sur la page suivante)

(suite de la page précédente)

```
   nb2 := LireEntier();
   nb3 := LireEntier();
   nb4 := LireEntier();
   writeln('Le maximum de ', nb1, ', ', nb2, ', ', nb3, ' et ', nb4, ' est : ',␣
↪Max4(nb1, nb2, nb3, nb4));
END;

(*
 -----------------------------------------------------------------------------------------
 -- Procedure          : AfficheBissextile()
 -- Auteur             : Florent Devin <fd@eisti.eu>
 -- Date de creation   : Mon Nov 13 14:25:52 2017
 --
 -- But                : Affiche si une année est bissextile
 -- Remarques          : Aucune
 -- Pré conditions     : Aucune
 -- Post conditions    : Affiche si une année est bissextile
 -----------------------------------------------------------------------------------------
↪*)
PROCEDURE AfficheBissextile();
VAR
    year :Integer;
BEGIN
    year := LireEntier();
    if ((year MOD 4 = 0) and (year MOD 100 <> 0)) or (year MOD 400 = 0) THEN
        writeln(year,' est bissextile !')
    else
        writeln(year,' n''est pas bissextile')
END;

(*
 -----------------------------------------------------------------------------------------
 -- Procedure          : AffichePGCD()
 -- Auteur             : Florent Devin <fd@eisti.eu>
 -- Date de creation   : Mon Nov 13 14:26:28 2017
 --
 -- But                : Calcule et affiche le pgcd de deux nombres
 -- Remarques          : Aucune
 -- Pré conditions     : Aucune
 -- Post conditions    : Calcule et affiche le pgcd de deux nombres
 -----------------------------------------------------------------------------------------
↪*)
PROCEDURE AffichePGCD();
VAR
    a,b,i,m : Integer;
BEGIN
    m := 1;
    a := LireEntier();
    b := LireEntier();
    for i:=1 TO a DO
    BEGIN
        if (a MOD i = 0) and (b MOD i = 0) then
            if (i>m) then
                m := i
        ;
    END;
    writeln('Le PGCD de ',a,' et ',b,' est ',m);
```

(suite sur la page suivante)

(suite de la page précédente)

```
END;

(*
 ----------------------------------------------------------------------------------------
 -- Fonction         : Power()
 -- Auteur           : Moi
 -- Date de creation  : Wed Nov 7 11:33:05 2018
 --
 -- But              : Retourne a élevé à la puissance b
 -- Remarques        : Aucune
 -- Pré conditions    : Aucune
 -- Post conditions   : ¯\_()_/¯
 ----------------------------------------------------------------------------------------
↪*)
FUNCTION Power(a,b:Integer) : Integer;
VAR
    n : Integer;
    sum : Integer;
BEGIN
    sum := 1;
    for n:=1 to b do
        sum := sum*a;
    Power := sum;
END;

PROCEDURE AfficherPower();
VAR
    s:Integer;
BEGIN
    s := Power(LireEntier(),LireEntier());
    writeln('Résultat : ',s);
END;

(*
 ----------------------------------------------------------------------------------------
 -- Procedure        : AffichePiEuler()
 -- Auteur           : Florent Devin <fd@eisti.eu>
 -- Date de creation  : Mon Nov 13 14:27:05 2017
 --
 -- But              : Affiche et calcule pi par la méthode d'Euler
 -- Remarques        : Aucune
 -- Pré conditions    : Aucune
 -- Post conditions   : Affiche et calcule pi par la méthode d'Euler
 ----------------------------------------------------------------------------------------
↪*)
PROCEDURE AffichePiEuler();
VAR
    p : Real;
    n,x,a : Integer;
BEGIN
    p := 0;
    writeln('Entrez le degré de précision');
    n := LireEntier();
    for x := 1 to n do
    BEGIN
        a := Power(x,2);
        p := p+ 1/a;
```

(suite sur la page suivante)

(suite de la page précédente)

```
    END;
    writeln('check');
    writeln(sqrt(p*6))
END;

(*
 -----------------------------------------------------------------------------------------
 -- Procedure          : AffichePiLeibniz()
 -- Auteur             : Florent Devin <fd@eisti.eu>
 -- Date de creation   : Mon Nov 13 14:27:42 2017
 --
 -- But                : Affiche et calcule Pi par la méthode de Leibniz
 -- Remarques          : Aucune
 -- Pré conditions     : Aucune
 -- Post conditions    : Affiche et calcule Pi par la méthode de Leibniz
 -----------------------------------------------------------------------------------------
↪*)
PROCEDURE AffichePiLeibniz();
VAR
    n,x : Integer;
    temp : Real;
    cond : boolean;
BEGIN
    cond := True;
    writeln('Entrez le degré de précision');
    n := LireEntier();
    temp := 0;
    for x:=1 to n do
        BEGIN
            IF (x MOD 2 = 1) THEN
            BEGIN
            IF cond THEN
                temp := temp + 1/x
            ELSE
                temp := temp - 1/x
            ;
            cond := not cond;
            END;
        END;
    writeln(temp*4)
END;

(*
 -----------------------------------------------------------------------------------------
 -- Procedure          : AfficheMenu()
 -- Auteur             : Florent Devin <fd@eisti.eu>
 -- Date de creation   : Mon Nov 13 13:25:58 2017
 --
 -- But                : Affiche le menu du TP
 -- Remarques          : Aucune
 -- Pré conditions     : Aucune
 -- Post conditions    : Affiche le menu du TP
 -----------------------------------------------------------------------------------------
↪*)
PROCEDURE AfficheMenu();
BEGIN
   writeln('1 : afficher le maximum de 4 nombres');
```

(suite sur la page suivante)

(suite de la page précédente)

```
   writeln('2 : affiche si une année est bissextile');
   writeln('3 : affiche le PGCD de deux nombres');
   writeln('4 : calcul de PI par la méthode d''Euler');
   writeln('5 : calcul de PI par la méthode de Leibniz');
   writeln('6 : calcul d''une puissance');
   writeln('');
   writeln('0 : Quitter');
    END;

(*
 -----------------------------------------------------------------------------------
 -- Procedure          : effectueActionMenu(choix : Integer)
 -- Auteur             : Florent Devin <fd@eisti.eu>
 -- Date de creation   : Mon Nov 13 14:20:19 2017
 --
 -- But                : Lance les actions correspondantes par rapport au choix de l
↪'utilisateur
 -- Remarques          : Si choix vaut 0 alors on affiche un message de sortie
 -- Pré conditions     : 0 <= choix < 6
 -- Post conditions    : Lance les actions correspondantes par rapport au choix de l
↪'utilisateur
 -----------------------------------------------------------------------------------
↪*)
PROCEDURE effectueActionMenu(choix : Integer);
BEGIN
   CASE choix OF
     1 : AfficheMax4();
     2 : AfficheBissextile();
     3 : AffichePGCD();
     4 : AffichePiEuler();
     5 : AffichePiLeibniz();
     6 : AfficherPower();
   ELSE
        writeln('');
        writeln('Bye');
   END;
END;

(*
 -----------------------------------------------------------------------------------
 -- Fonction           : SaisirChoix() : Integer
 -- Auteur             : Florent Devin <fd@eisti.eu>
 -- Date de creation   : Mon Nov 13 13:32:14 2017
 --
 -- But                : Permet la saisie d'un choix pour le menu
 -- Remarques          : Le nombre retourné est compris : 0 <= x < 6
 -- Pré conditions     : Aucune
 -- Post conditions    : Permet la saisie d'un choix pour le menu
 -----------------------------------------------------------------------------------
↪*)
FUNCTION SaisirChoix() : Integer;
VAR
   choix : Integer;
BEGIN
   REPEAT
      AfficheMenu();
      writeln('');
```

(suite sur la page suivante)

(suite de la page précédente)

```
      writeln('Entrez votre choix : ');
      readln(choix);
   UNTIL ((choix >= 0) and (choix < 7));
   SaisirChoix:=choix;
END;

VAR
   choix : Integer;
(*Début du programme principal*)
BEGIN
   REPEAT
      choix:=SaisirChoix();
      effectueActionMenu(choix);
      (* Ou aussi effectueActionMenu(SaisirChoix()) *)
      writeln('')
   UNTIL (choix = 0);
END.
(*feat Loann*)
```

2.2 TP Sup

```
(*
-----------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise / Loann Pottier
-- Date de creation  : Sun Nov 11 2018
--
-- But               : TP de géométrie en Pascal
-- Remarques         : Aucune
-- Compilation       : fpc
-- Edition des liens : fpc
-- Execution         : shell
-----------------------------------------------------------------------------------
*)

PROGRAM Tp1;

(*
-----------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise
-- Date de creation  : Sun Nov 11 2018
--
-- But               : TP de géométrie - question 1
-- Remarques         : Aucune
-- Compilation       : fpc
-- Edition des liens : fpc
-- Execution         : shell
-----------------------------------------------------------------------------------
*)

PROCEDURE ligne(n : integer);
VAR
    x: integer;
```

(suite sur la page suivante)

(suite de la page précédente)

```
begin
    for x:=1 to n do
        writeln('*');
end;

PROCEDURE carre1(n : integer);
VAR
    x,y: integer;
begin
    for x:=1 to n do
        begin
        for y:=1 to n do
            write('* ');
        writeln('')
        end;
end;

PROCEDURE triangle1(n : integer);
VAR
    x,y: integer;
begin
    for x:=1 to n do
        begin
            for y:=1 to x do
                write('* ');
            writeln('');
        end;
end ;

PROCEDURE triangle2(n : integer);
VAR
    x, y: integer;
begin
    for x:=1 to n do
        begin
            for y:=1 to n-x do
                write('  ');
            for y:=1 to x do
                write('* ');
        writeln(' ');
        end;
end;

PROCEDURE carre2(n : integer);
VAR x,y: integer;
begin
    for x:=1 to n do
        begin
        if (x=1) or (x=n) then
            begin
            for y:=1 to n do
                write('* ');
            end
        else
            begin
            write('* ');
            for y:=1 to n-2 do
```

(suite sur la page suivante)

(suite de la page précédente)

```
                write('  ');
            write('* ');
            end;
        writeln('');
        end;
end;

PROCEDURE croix(n : integer);
VAR x,y: integer;
begin
    for x:=1 to n do
        begin
        for y:=1 to n do
            begin
            if (y=x) or (y=n-x+1) then
                write('* ')
            else
                write('  ')
            ;
            end;
        writeln('');
        end;
end;

PROCEDURE q1();
VAR
    choix,n : integer;
begin
    writeln('Choisissez la fonction à lancer');
    writeln('1 - Ligne');
    writeln('2 - Carré1');
    writeln('3 - Triangle1');
    writeln('4 - Triangle2');
    writeln('5 - Carré2');
    writeln('6 - Croix');
    write('> ');read(choix);
    if (choix>6) or (choix<1) then
        begin
            writeln('Choix invalide')
        end
    else
        begin
            writeln('');
            write('Entrez l''entier n : ');read(n);
            case choix of
                1 : ligne(n);
                2 : carre1(n);
                3 : triangle1(n);
                4 : triangle2(n);
                5 : carre2(n);
                6 : croix(n);
            else
                writeln('Nothing');
            end;
        end;
    writeln('')
end;
```

(suite sur la page suivante)

(suite de la page précédente)

```
(*
-------------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise
-- Date de creation  : Sun Nov 42
--
-- But               : TP de géométrie - question 2
-- Remarques         : Aucune
-- Compilation       : fpc
-- Edition des liens : fpc
-- Execution         : shell
-------------------------------------------------------------------------------------
*)

PROCEDURE q2();
VAR
    word : string;
    x : integer;
    a : char;
begin
    writeln('Entrez une chaine de caractères');
    write('> ');read(word);
    for x:=1 to length(word) do
        begin
        a := word[x];
        if a in ['a','e','i','o','u','y'] then
            write('?')
        else
            write(a)
        ;
        end;
    writeln('');
end;


(*
-------------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise
-- Date de creation  : Mon Nov 12 2018
--
-- But               : TP de géométrie - question 3
-- Remarques         : Aucune
-- Compilation       : fpc
 -- Edition des liens : fpc
-- Execution         : shell
-------------------------------------------------------------------------------------
*)

FUNCTION q3f() : integer;
VAR word : string;
    x,a : integer;
begin
    writeln('Entrez une chaine de caractères');
    write('> ');read(word);
```

(suite sur la page suivante)

(suite de la page précédente)

```
    a := 1;
    for x:=1 to length(word) do
        if (word[x] = ' ') then
            a := a+1
        ;
    q3f := a;
end;

PROCEDURE q3();
VAR a:integer;
begin
    a := q3f();
    writeln(a);
end;


(*
-------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise
-- Date de creation  : Tue Nov 13 2018
--
-- But               : TP de géométrie - question 4
-- Remarques         : Aucune
-- Compilation       : fpc
-- Edition des liens : fpc
-- Execution         : shell
-------------------------------------------------------------------------------
*)

FUNCTION q4f() : string;
VAR sent,temp,rep:string;
    a:char;
    x,y:integer;
begin
    write('Saisissez une chaine de caractères :');
    read(sent);
    if sent='' then
        begin
        sent := '     lol mdr     ';
        write(sent)
        end;
    a := ' ';
    x := 0;
    temp := '';
    rep := '';
    while a=' ' do
        begin
        x := x+1;
        a := sent[x];
        end;
    for y:=x to length(sent) do
        temp := temp+sent[y];

    a := ' ';
    x := length(temp)+1;
    while a=' ' do
```

(suite sur la page suivante)

(suite de la page précédente)

```
        begin
        x := x-1;
        a := temp[x];
        end;
    for y:=x downto 1 do
        rep := temp[y]+rep;
    writeln('');
    q4f := rep;
end;

PROCEDURE q4();
VAR s:string;
begin
    s := q4f();
    writeln('-',s,'-')
end;

(*
-------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise
-- Date de creation  : Tue Nov 13 2018
--
-- But               : TP de géométrie - question 5
-- Remarques         : Aucune
-- Compilation       : fpc
-- Edition des liens : fpc
-- Execution         : shell
-------------------------------------------------------------------------------
*)

FUNCTION lireInt() :integer;
VAR x:integer;
begin
    writeln('');
    write('> ');
    read(x);
    lireInt := x;
end;

PROCEDURE q5funct(VAR x,y:real);
VAR a,b,c,d,e,f:integer;
    temp1,temp2:real;
begin
    writeln('Entrez les valeurs de a, b, c, d, e et f');
    a := lireInt();
    b := lireInt();
    c := lireInt();
    d := lireInt();
    e := lireInt();
    f := lireInt();
    temp1 := f-(d*c)/a;
    temp2 := e+b/a;
    y := -temp1/temp2;
    x := (c-b*y)/a;
end;
```

(suite sur la page suivante)

(suite de la page précédente)

```
PROCEDURE q5();
VAR x,y:real;
begin
    x := 0.0;
    y := 0.0;
    q5funct(x,y);
    writeln('Les solutions sont x=',x,' et y=',y);
end;

(*
-------------------------------------------------------------------------------
-- Fichier           : tp2.pas
-- Auteur            : Arthur Blaise
-- Date de creation  : Sun Nov 11 2018
--
-- But               : TP de géométrie - procédure prinicpale
-- Remarques         : Aucune
-- Compilation       : fpc
-- Edition des liens : fpc
-- Execution         : shell
-------------------------------------------------------------------------------
*)

VAR
    choix : integer;
begin
    write('Question n°');
    read(choix);
    if (choix>5) or (choix<0) then
        writeln('Choix invalide')
    else
        begin
            writeln('');
            case choix of
                1 : q1();
                2 : q2();
                3 : q3();
                4 : q4();
                5 : q5();
            else
                writeln('Nothing');
            end;
        end;
    writeln('')
end.
```

2.3 Encore plus loin

```
PROGRAM minMax;

CONST imin=1; imax=8;
Type tabstat = array[imin..imax] of real;

PROCEDURE q1(t:tabstat; var min, max:real);
```

(suite sur la page suivante)

(suite de la page précédente)

```
VAR i:integer;
begin
min := t[1]; max := t[1];
FOR i:=1 to length(t) do
    begin
    if t[i]<min then
        min := t[i]
    ;
    if t[i]>max then
        max := t[i]
    end;
end;

VAR t:tabstat;
i:integer;
min,max:real;
Begin
FOR i:=imin to imax do
begin
    write('> ');read(t[i]);
end;
writeln('');
q1(t,min,max);
writeln('Minimum : ',min,' | Maximum : ',max);
writeln(' ');
end.
```

```
PROCEDURE q2(t1,t2:tabstat, var stroumpf:real);
begin
FOR i:=1 to length(t2) do
    for j:=1 to length(t) do
        stroumpf := stroumpf + t[j]*t2[i];
end;
```

```
PROCEDURE q1(t:tabstat; var t2:tabstat);
VAR i,j:integer;
begin
j := 1;
t2[1] := t[1];
for i:=2 to length(t) do
    if t[i]<>t[i-1] then
    begin
        j := j+1;
        t2[j] := t[i];
    end;
end;
```

2.4 Les tableaux dynamiques

2.4.1 Le carré magique

```
function izmagic(tableau:carre):boolean;
var j,s,exs:integer; // incrément, somme, et deuxième somme
    i:array of integer; // ligne du carré
begin
    s := 0;
    exs := -1;
    izmagic := True;
    for i in tableau do begin // on commence par les lignes
        for j in i do
            s := s+j;
        if exs>-1 then
            izmagic := izmagic and (s=exs)
        else
            exs := s;
        s:=0;
        end;
    exs := -1;
    for j:=0 to high(tableau[0]) do begin // puis les colonnes
        for i in tableau do
        s := s+i[j];
        if exs>-1 then
            izmagic := izmagic and (s=exs)
        else
            exs := s;
        s := 0;
        end;
    exs := 0;
    for j:=0 to high(tableau) do begin // enfin les deux diagonales
        exs := exs + tableau[j,j]; // x=y
        s := s + tableau[j,high(tableau)-j]; // n-x=y
        end;
    izmagic := izmagic and (s=exs);
end;
```

2.4.2 Horloge digitale

```
Uses Dos,sysutils,Crt;
Type number=array[0..4] of string;

Const zero:number= ('#####',
                    '#   #',
                    '#   #',
                    '#   #',
                    '#####');
Const one:number = ('    #',
                    '    #',
                    '    #',
                    '    #',
                    '    #');
Const two:number = ('#####',
                    '    #',
                    '#####',
                    '#    ',
                    '#####');
Const three:number=('#####',
```

(suite sur la page suivante)

(suite de la page précédente)

```
                    '    #',
                    ' ####',
                    '    #',
                    '#####');
Const four:number= ('#   #',
                    '#   #',
                    '#####',
                    '    #',
                    '    #');
Const five:number= ('#####',
                    '#    ',
                    '#####',
                    '    #',
                    '#####');
Const six:number = ('#####',
                    '#    ',
                    '#####',
                    '#   #',
                    '#####');
Const seven:number=('#####',
                    '    #',
                    '    #',
                    '    #',
                    '    #');
Const eight:number=('#####',
                    '#   #',
                    '#####',
                    '#   #',
                    '#####');
Const nine:number= ('#####',
                    '#   #',
                    '#####',
                    '    #',
                    '#####');
Const point:number =('   ',
                    ' # ',
                    '   ',
                    ' # ',
                    '   ');

procedure displayNumb(wo:string);
var c:char;
    i:integer;
    t:string;
begin
    t := '';
    for i:=0 to 4 do begin
        for c in wo do
            CASE c OF
                '0': t := t+zero[i]+' ';
                '1': t := t+one[i]+' ';
                '2': t := t+two[i]+' ';
                '3': t := t+three[i]+' ';
                '4': t := t+four[i]+' ';
                '5': t := t+five[i]+' ';
                '6': t := t+six[i]+' ';
                '7': t := t+seven[i];
```

(suite sur la page suivante)

(suite de la page précédente)

```
                '8': t := t+eight[i]+' ';
                '9': t := t+nine[i]+' ';
                ':': t := t+point[i];
            end;
        writeln(t);
        t := '';
    end;
end;

procedure q2;
var Hour,Min,Sec,HSec:word;
    exSec:word; //backup des secondes
    H,M,S:string;
begin
    exSec := 0;
    while True do begin
        GetTime(Hour,Min,Sec,HSec);
        if Sec<>exSec then begin
            clrscr;
            if Hour<10 then H := '0'+IntToStr(Hour) else H := IntToStr(Hour);
            if Min<10 then M := '0'+IntToStr(Min) else M := IntToStr(Min);
            if Sec<10 then S := '0'+IntToStr(Sec) else S := IntToStr(Sec);
                displayNumb(H+':'+M+':'+S);
                exSec := Sec;
                end;
            end;
    end;
```

2.4.3 Manipulation de listes dynamiques

Ici le but est de créer une fonction invert(array) qui permet d'inverser l'ordre des valeurs d'une liste, et une deuxième, push(array,integer) qui décale chaque valeur d'un certain nombre de rang vers la droite. Evidemment les fonctions doivent pouvoir s'adapter à la longueur de la liste.

Parce que le modulo natif en Pascal m'a occasionné pas mal de bugs, j'ai préféré créer moi-même une fonction de modulo. Avec si peu de lignes pour tellement de problèmes en moins, je n'avais rien à perdre !

```
function modulo(a, b: integer): integer;
begin
  modulo:= a - b * Round(a / b);
  if modulo<0 then modulo := modulo+b
end;

function invert(table:array):array
var i:integer;
Begin
    SetLength(invert,length(table));
    for i:=0 to high(table) do
        invert[high(table)-i] := table[i];
end;

function push(table:array; n:integer):array;
var i,j:integer;
Begin
    SetLength(push,length(table));
```

(suite sur la page suivante)

(suite de la page précédente)

```
    for i:=0 to high(table) do begin
        j := modulo(n+i,length(table));
        push[j] := table[i];
        end;
end;
```

2.5 Les pointeurs

Ici nous utilisons les pointeurs pour créer notre propre type de liste, et en implémentant sur ces listes différentes fonctions telles que l'insertion d'une valeur ou la suppression d'un index. Voici donc ces fonctions, ainsi que le menu qui permet de les tester.

```
PROGRAM do_it_urself;

{$mode objfpc}

uses math;

TYPE
    ptr_noeud = ^noeud;
    noeud = RECORD
        valeur : INTEGER;
        suivant : ptr_noeud;
    end;


function creerNoeud(val:INTEGER;suivant:ptr_noeud=Nil):ptr_noeud;
var nv:ptr_noeud;
begin
    new(nv);
    nv^.valeur := val;
    nv^.suivant := suivant;
    creerNoeud := nv
end;

function len(tete:ptr_noeud):integer;
var tmp:ptr_noeud;
begin
    len := 0;
    tmp := tete;
    while (tmp <> Nil) do begin
        len := len+1;
        tmp := tmp^.suivant;
    end;
end;

procedure display(tete:ptr_noeud;text:string='');
var i:integer;
begin
    write(text,'[');
    for i:=1 to len(tete)-1 do begin
        write(tete^.valeur,';');
        tete := tete^.suivant;
        end;
```

(suite sur la page suivante)

(suite de la page précédente)

```
    writeln(tete^.valeur,']');
end;


function insert_b(tete:ptr_noeud;val:integer):ptr_noeud;
begin
    insert_b :=  creerNoeud(val,tete);
end;

procedure insert_e(tete:ptr_noeud;val:integer);
var n:ptr_noeud;
begin
    while tete^.suivant<>Nil do
        tete := tete^.suivant;
    n := creerNoeud(val);
    tete^.suivant := n;
end;

procedure insert_m(tete:ptr_noeud;val,pos:integer);
var n:ptr_noeud;
    i:integer;
begin
    i := 0;
    pos := min(pos,len(tete));
    while i<pos-1 do begin
        i := i+1;
        tete := tete^.suivant;
        end;
    n := creerNoeud(val,tete^.suivant);
    tete^.suivant := n;
end;

function del_b(tete:ptr_noeud):ptr_noeud;
begin
    del_b := tete^.suivant;
    dispose(tete);
end;

procedure del_e(tete:ptr_noeud);
begin
    while tete^.suivant^.suivant <> Nil do
        tete := tete^.suivant;
    dispose(tete^.suivant);
    tete^.suivant := Nil;
end;

procedure del_m(tete:ptr_noeud;pos:integer);
var i:integer;
begin
    i := 0;
    pos := min(pos,len(tete)-1);
    while i<pos-1 do begin
        i += 1;
        tete := tete^.suivant;
        end;
    tete^.suivant := tete^.suivant^.suivant
end;
```

(suite sur la page suivante)

(suite de la page précédente)

```
function search(tete:ptr_noeud;val:integer):integer;
var i:integer;
begin
    i := 0;
    while tete<>Nil do begin
        if tete^.valeur=val then Exit(i);
        i := i+1;
        tete := tete^.suivant;
        end;
    exit(-1);
end;

function test:boolean;
var tete,v2:ptr_noeud;
begin
    v2 := creerNoeud(14);
    tete := creerNoeud(12,v2);
    insert_e(tete,20);
    display(tete,'Liste de départ: ');
    writeln;
    tete := insert_b(tete,-5);
    insert_e(tete,42);
    display(tete,'Insertion de -5 au début et 42 à la fin: ');
    writeln('Longueur de la liste: ',len(tete));
    insert_m(tete,34,3);
    display(tete,'Insertion de 34 à la case 3: ');
    writeln;
    del_e(tete);
    display(tete,'Suppression de la dernière case: ');
    del_m(tete,2);
    display(tete,'Suppression de la case 2: ');
    writeln('Index de la valeur 34: ',search(tete,34));
    Exit(True);
end;


function menu_insert(tete:ptr_noeud):ptr_noeud;
var pos,val:integer;
begin
    write('Entrez la valeur à insérer',#10,'> ');readln(val);
    if len(tete)=0 then
        Exit(creerNoeud(val));
    write('Entrez la position à laquelle insérer la valeur',#10,'> ');readln(pos);
    if (pos<0) or (pos>len(tete)) then begin
        writeln('Position invalide');
        Exit(tete);
        end;
    if pos=0 then
        tete := insert_b(tete,val)
    else if pos=len(tete) then
        insert_e(tete,val)
        else insert_m(tete,val,pos);
    exit(tete);
end;

function menu_search(tete:ptr_noeud):ptr_noeud;
```

(suite sur la page suivante)

(suite de la page précédente)

```
var val:integer;
begin
    write('Entrez la valeur à rechercher',#10,'> ');readln(val);
    val := search(tete,val);
    if val=-1 then writeln('Valeur introuvable')
    else writeln('Cette valeur est à l''index ',val);
    exit(tete);
end;

function menu_del(tete:ptr_noeud):ptr_noeud;
var pos:integer;
begin
    write('Entrez l''index de la case à supprimer, entre 0 et ',len(tete),#10,'> ');
↪readln(pos);
    if (pos<0) or (pos>len(tete)) then begin
        writeln('Position invalide');
        Exit(tete);
        end;
    if pos=0 then
        tete := del_b(tete)
    else if pos=len(tete) then
        del_e(tete)
        else del_m(tete,pos);
    exit(tete);
end;

function menu_random(tete:ptr_noeud):ptr_noeud;
var i,n:integer;
begin
    write('Entrez le nombre de cases à ajouter',#10,'> ');readln(n);
    if n<1 then begin
        writeln('Impossible d''ajouter un nombre négatif de cases !');Exit(tete);end;
    if n>150 then begin
        writeln('Impossible d''ajouter plus de 150 cases !');Exit(tete);end;
    if len(tete)=0 then begin
        tete := creerNoeud(round(random(n*200)-n*100));
        n := n-1;
        end;
    for i:=1 to n do
        insert_e(tete,round(random(n*200)-n*100));
    Exit(tete);
end;

function menu_del_2(tete:ptr_noeud):ptr_noeud;
var val,pos:integer;
begin
    write('Entrez la valeur à effacer du tableau',#10,'> ');readln(val);
    pos := search(tete,val);
    if pos=-1 then begin
        writeln('Valeur introuvable');
        Exit(tete);
        end;
    if pos=0 then
        tete := del_b(tete)
    else if pos=len(tete) then
        del_e(tete)
        else del_m(tete,pos);
```

(suite sur la page suivante)

(suite de la page précédente)

```
    Exit(tete);
end;

function menu_del_3(tete:ptr_noeud):ptr_noeud;
var val,pos:integer;
    b:boolean;
begin
    write('Entrez la valeur à effacer du tableau',#10,'> ');readln(val);
    b := True;
    while b do begin
        pos := search(tete,val);
        // writeln('  #',pos,' trouvé');
        // display(tete,'   ');
        if pos=-1 then b:=False
        else if pos=0 then
            tete := del_b(tete)
        else if pos=len(tete) then
            del_e(tete)
            else del_m(tete,pos);
    end;
    Exit(tete);
end;

function menu_duplicate(tete:ptr_noeud):ptr_noeud;
var i:integer;
    node:ptr_noeud;
begin
    node := tete;
    for i:=1 to len(tete) do begin
        insert_e(tete,node^.valeur);
        node := node^.suivant;
        end;
    Exit(tete)
end;

function menu_reverse(tete:ptr_noeud):ptr_noeud;
var l:array of integer;
    node:ptr_noeud;
    i:integer;
begin
    SetLength(l,len(tete));
    i := 0;
    node := tete;
    while node<>Nil do begin
        l[i] := node^.valeur;
        node := node^.suivant;
        i := i+1
        end;
    node := tete;
    i := i-1;
    while node<>Nil do begin
        node^.valeur := l[i];
        node := node^.suivant;
        i := i-1
        end;
    Exit(tete)
end;
```

(suite sur la page suivante)

(suite de la page précédente)

```
function menu_clear(tete:ptr_noeud):ptr_noeud;
var node:ptr_noeud;
    i,pos:integer;
begin
    if len(tete)<2 then Exit(tete);
    node := tete^.suivant;
    i := 0;
    while node<>Nil do begin
        if node^.suivant<>Nil then // si on n'est pas à la fin de la liste
            pos := search(node^.suivant,node^.valeur) // on détecte si la valeur est␣
↪représentée plus loin dans la liste
        else break;
        if (pos=-1) and (node^.valeur <> tete^.valeur) then begin // Si non, on passe
            node := node^.suivant;
            i := i+1;
            continue
            end;
        node := node^.suivant;
        del_m(tete,i)
        end;
    Exit(tete)
end;

procedure menu;
var tete:ptr_noeud;
    choice:integer;
    useless_b:boolean;
begin
    tete := Nil;
    while True do begin
    writeln('Choisissez l''action à effectuer :',#10,'  1 - Insérer une valeur',#10,'␣
↪ 2 - Rechercher une valeur',#10,'  3 - Supprimer une case',#10,'  4 - Supprimer une␣
↪valeur',#10,'  5 - Supprimer toutes les occurences d''une valeur',#10,'  6 -␣
↪Dupliquer la liste',#10,'  7 - Inverser la liste',#10,'  8 - Supprimer les doublons
↪',#10,'  9 - Ajouter des cases aléatoires',#10,'  10 - Afficher le programme de test
↪',#10,'  0 - Quitter');
    write('> ');readln(choice);
    if (choice<0) or (choice>10) then begin
        writeln('Saisie invalide',#10);
        continue;
        end
    else if choice=0 then break;
    if (choice>1) and (choice<8) and (len(tete)=0) then begin
        writeln('Impossible de rechercher ou de supprimer une valeur dans un tableau␣
↪vide !');
        continue;
        end;
    case choice of
        1: tete := menu_insert(tete);
        2: tete := menu_search(tete);
        3: tete := menu_del(tete);
        4: tete := menu_del_2(tete);
        5: tete := menu_del_3(tete);
        6: tete := menu_duplicate(tete);
        7: tete := menu_reverse(tete);
        8: tete := menu_clear(tete);
```

(suite sur la page suivante)

(suite de la page précédente)

```
        9: tete := menu_random(tete);
        10: useless_b := test;
    end;
    if len(tete)>0 then display(tete,#10+'Valeur actuelle: ');
    writeln;
    end;
end;

begin
    randomize;
    menu;
    writeln(#10,'--- Fin du programme ---',#10)
end.
```

CHAPITRE 3

Suite du TP Unix

3.1 TP 2

3.1.1 Exercice 1

```
  $ pwd
/cergy/homee/b/blaisearth

  $ cd /
  $ ls
bin    dev   initrd.img      lib64       mnt   root  srv  usr      vmlinuz.old
boot   etc   initrd.img.old  lost+found  opt   run   sys  var      vms
cergy  home  lib             media       proc  sbin  tmp  vmlinuz

  $ cd ~ 'ou' cd /cergy/homee/b/blaisearth 'ou' cd

  $ mkdir Info

  $ mkdir Info/SGF

  $ cd Info/SGF

  $ mkdir tmp

  $ cd tmp

  $ touch toto 'ou' > toto

  $ find ../.. | sed 's/[^/]*\//| /g;s/| *\([^| ]\)/+--- \1/'
1/'
+--- ..
| +--- SGF
| | +--- tmp
```

(suite sur la page suivante)

(suite de la page précédente)

```
| | | +--- toto
  $ ls -lR ..
..:
total 1
drwxr-xr-x 2 blaisearth users 3 nov.  19 14:28 tmp

../tmp:
total 1
-rw-r--r-- 1 blaisearth users 0 nov.  19 14:28 toto

  $ cd ..

  $ rmdir tmp
rmdir: impossible de de supprimer 'tmp': Le dossier n'est pas vide
  $ rm -r tmp 'ou' rmdir --ignore-fail-on-non-empty tmp

  $ head ~/2050.txt -n 15

  $ more -9 ~/2050.txt

  $ head -n 9 ~/2050.txt | tail -n 5 > tmp1.txt
  $ head -n 13 ~/2050.txt | tail -n 2 > tmp2.txt
  $ cat tmp1.txt tmp2.txt > ~/data.txt

  $ cat ~/2050.txt
  $ more ~/2050.txt
  #La commande cat affiche le texte d'un coup, alors que more le fait défiler ligne␣
↪par ligne

  $ more +/vautours ~/2050.txt
  $ more +/FFME ~/2050.txt

  $ mkdir tmp

  $ cp ~/2050.txt ./tmp

  $ cd tmp
  $ mv 2050.txt Escalade2050.txt

  $ mv Escalade2050.txt ~/

  $ ls ~/

  $ ln -s "/cergy/homee/b/blaisearth/Escalade2050.txt" Escalade2050

  $ ls -A --author -F -h -H -i -s
total 512
83135077 512 Escalade2050@
```

3.1.2 Exercice 2

```
$ head -n 5 2050.txt

$ head -n 5 2050.txt > toto
```

(suite sur la page suivante)

(suite de la page précédente)

```
$ tail -n 5 2050.txt >> toto

$ sort -b toto

$ head -c 20 toto #affiche les 20 premiers caractères du fichier

$ cp toto tata

$ cat toto >> tata

$ sort -u tata > tata2
$ cat tata2 > tata
$ rm tata2

$ sed -i 's/ //g' tata

$ find -mtime -1

$ find -mtime -1 -name *les*
```

3.1.3 Exercice 3

```
  $ rm ~/Escalade2050.txt

  $ ls ./Info/SGF/tmp
Escalade2050 #(en rouge)
#La couleur rouge indique que le lien n'est plus valide, la source a disparu
```

CHAPITRE 4

Introduction au C

4.1 TD 1 : Entrées, sorties

4.1.1 Exercice 9 : Minimum et maximum

Consigne : *Écrire une méthode permettant de saisir une liste de n réels et de calculer et d'afficher les valeurs minimale et maximale de cette liste.*

```
/*!
\fn void ex9(void)
\brief Code de l'exercice 9
\return rien
*/
void ex9() {
    int int_min, int_max;   // minimum et maximum
    int int_n;              // nombre de cases
    int i;                  // iterateur
    printf("Entrez le nombre de cases à remplir, entre 1 et 50: ");
    scanf("%d",&int_n);
    if ((int_n < 1) || (int_n > 50)) { // si N dépasse les bornes
        printf("Nombre invalide");
        return;
    }
    int liste[int_n];
    for (i = 0; i < int_n; ++i) { // on construit la liste avec les entrees de l
↪'utilisateur
        scanf("%d",&liste[i]);
        if (i==0) { // initialisation du min et du max avec une valeur de la liste
            int_min = liste[i];
            int_max = liste[i];
        }
        else { // recherche du min et du max dans le reste de la liste
            if (int_min > liste[i]) {
```

(suite sur la page suivante)

(suite de la page précédente)

```
            int_min = liste[i];
        };
        if (int_max < liste[i]) {
            int_max = liste[i];
        };
        }
    };
    printf("Minimum : %d - Maximum : %d", int_min, int_max);
}
```

4.2 TD 4 : Vecteurs 2D

4.2.1 Exercice : Puissance 4

Consigne : *Ben... créer un puissance 4 avec une matrice (donc un vecteur de vecteurs). La taille est définie dans une constante symbolique notée N. Le plateau de jeu est noté* `ttint_plateau[i][i]` *, avec i pour le numéro de colonne et j le numéro de ligne.*

Extrait de code 1 : vérifier l'état actuel du jeu, si un joueur a gagné ou si ex-aequo (toutes les cases sont occupées), ou si on continue de jouer. J'ai défini les fonctions `checkDiag1`, `checkDiag2` etc. à côté, elles renvoient chacune d'entre elles le nombre de pions alignés io en fonction de la coordonnée donnée.

```
int aGagne(int PLATEAU) {
    if (plateauRempli(ttint_plateau)) {
        return(0);
    }
    for (int int_joueur=1; int_joueur<3; int_joueur++) {
        for (int int_increment=0; int_increment<N; int_increment++) {
            if (checkDiag1(ttint_plateau, int_increment, int_joueur) > 3) {
                return(int_joueur);
            }
            if (checkDiag2(ttint_plateau, int_increment, int_joueur) > 3) {
                return(int_joueur);
            }
            if (checkColonne(ttint_plateau, int_increment ,int_joueur) > 3) {
                return(int_joueur);
            }
            if (checkLigne(ttint_plateau, int_increment, int_joueur) > 3) {
                return(int_joueur);
            }
        }
    }
    return(-1);
}
```

Extrait de code 2 : l'affichage du plateau. En vrai il n'y a pas grand chose de compliqué, il suffit de ne pas s'emmêler avec les indices et les boucles. Mais c'est joli donc woala.

```
void affichage(int PLATEAU) {
    int i;
    int j;
    printf("+");
    for (j=0;j<N;j++) {
```

(suite sur la page suivante)

(suite de la page précédente)

```
        printf("---+");
    }
    for (j=0;j<N;j++) {
        printf("\n|");
        for (i=0;i<N;i++) {
            switch (ttint_plateau[i][j]) {
                case 1:
                    printf(" O |");
                    break;
                case 2:
                    printf(" X |");
                    break;
                default:
                    printf("   |");
                    break;
            }
        }
        printf("\n+");
        for (i=0;i<N;i++) {
            printf("---+");
        }
    }
    printf("\n+");
    for (i=0;i<N;i++) {
        printf(" %d +",i+1);
    }
    printf("\n");
}
```

Extrait de code 3 : le retournement du plateau à 90° (pi/2 pour les intimes). Le plus compliqué est le retournement en lui-même, surtout si l'on souhaite une fonction modulable et courte. Mais j'ai réussi en 30 lignes, je suis content. La base de d'algo vient d'"ici.

```
void gravite(int PLATEAU) {
    int i,j,k;
    for (i=0; i<N; i++) {
        k = N-1;
        for (j=N-1;j>=0;j--) {
            if (ttint_plateau[i][j] > -1) {
                if (ttint_plateau[i][k] == -1) {
                ttint_plateau[i][k] = ttint_plateau[i][j];
                ttint_plateau[i][j] = -1;
                }
                k--;
            }
        }
    }
}

void rotationPlateau(int PLATEAU) {
    int i,j;
    for (j=0; j<N-2; j++)
    {
        for (i=j; i <N-j-1; i++)
        {
            int temp = ttint_plateau[i][j];
```

(suite sur la page suivante)

(suite de la page précédente)

```
            ttint_plateau[i][j] = ttint_plateau[j][N-i-1];
            ttint_plateau[j][N-i-1] = ttint_plateau[N-i-1][N-j-1];
            ttint_plateau[N-i-1][N-j-1] = ttint_plateau[N-j-1][i];
            ttint_plateau[N-j-1][i] = temp;
        }
    }
    gravite(ttint_plateau);
}
```