

# CORRECTIONS DES EXERCICES DU CONCOURS CRITECH EPITA 2006

## **REMARQUE :**

*Les programmes en Pascal, C sont des exemples et peuvent être discutés en terme d'implémentation et de construction. Le choix qui a été fait, est celui d'une découpe fonctionnelle et procédurale importante.*

*Cela permet de mieux faire ressortir un algorithme principal.*

*Ensuite chaque tâche est détaillée dans la procédure ou fonction correspondante.*

*D'autre part, il n'y a pas forcément d'optimisation entre les programmes Pascal et C qui peuvent être de simple traduction les uns des autres.*

## **A) DIFFERENCE INCHANGEE ...**

Le principe est le suivant :

Pour effectuer le calcul, nous allons utiliser un tableau de quatre entiers qui va nous permettre de stocker chaque chiffre lu. Pour une séquence donnée, nous allons à chaque tour trier les chiffres en ordre croissant. Puis créer le nombre maximum et le nombre minimum. Pour ce faire, nous allons mettre en place une petite boucle qui fait à l'endroit (pour le min) et à l'envers (pour le max) la somme du chiffre avec le calcul précédent multiplié par 10. Par exemple la séquence 7 0 8 3 est triée en 0 3 7 8, ce qui donne pour le min :  $(10*(10*(10*0)+3)+7)+8$  et pour le max :  $(10*(10*(10*8)+7)+3)+0$ . Ensuite, nous calculons la différence et si celle-ci est égale à celle calculée au tour précédent, nous donnons la valeur de cette différence (inchangée) et nous passons à la séquence de chiffres suivante :

Pour chaque séquence est écrit à chaque tour, dans le fichier de sortie (DIFFINCH.OUT) la séquence de chiffres, le plus grand et le plus petit nombre que l'on puisse obtenir avec la séquence donnée ainsi que la valeur de la différence entre ceux-ci.

### **Programme Pascal (Delphi mode console)**

```
program diffinch;

{$APPTYPE CONSOLE}

uses
  SysUtils;

Var
  FichierEntree, FichierSortie : Text;

procedure OuvertureFichier;
begin
  Assign(FichierEntree, 'DIFFINCH.IN');
  Reset(FichierEntree);           (* Ouverture de DIFFINCH.IN
en lecture *)
  Assign(FichierSortie, 'DIFFINCH.OUT');
  Rewrite(FichierSortie);       (* Ouverture de DIFFINCH.OUT
en écriture *)
end;

Procedure LectureEtTraitement;
Var
  chiffre : array[1..4] of integer;
  i, j, k, nb, max, min, diff, diffprec : integer;
  change : boolean;

Begin
  readln(FichierEntree, nb);
  j:=1;
```

```

while (j <= nb) do
Begin
  writeln(FichierSortie,'Séquence ',j,' :');
  writeln(FichierSortie,'');
  for i:=1 to 3 do read(FichierEntree,chiffre[i]);
  readln(FichierEntree,chiffre[4]);
  diff:=0;
  repeat
    diffprec:=diff;
    write(FichierSortie,'Chiffres : ');
    for i:=1 to 4 do write(FichierSortie,chiffre[i],' ');
    writeln(FichierSortie,'');
    (* tri des chiffres en ordre croissant *)
    change:=true;
    while change do
      begin
        change:=false;
        for i:=1 to 3 do
          if chiffre[i]>chiffre[i+1] then
            begin
              k:=chiffre[i];
              chiffre[i]:=chiffre[i+1];
              chiffre[i+1]:=k;
              change:=true;
            end;
          end;
        max:=chiffre[4];
        for i:=3 downto 1 do max:=max*10+chiffre[i];
        writeln(FichierSortie,'Plus grand : ',max); (* écriture du plus grand *)
        min:=chiffre[1];
        for i:=2 to 4 do min:=min*10+chiffre[i];
        writeln(FichierSortie,'Plus petit : ',min); (* écriture du plus petit *)
        diff:=max-min;
        writeln(FichierSortie,'Différence : ',diff); (* écriture de la différence *)
        writeln(FichierSortie,'');
        (* remise ne place des nouveaux chiffres *)
        k:=diff;
        for i:=4 downto 1 do
          begin
            chiffre[i]:=k mod 10;
            k:= k div 10;
          end;
        until diff=diffprec;
        writeln(FichierSortie,'différence inchangée : ',diff); (* arrêt séquence *)
        j:=j+1;
        writeln(FichierSortie,'');
      End;
    End;

procedure FermetureFichier;
begin
  Close(FichierEntree);
  Close(FichierSortie);
end;

begin
  OuvertureFichier;
  LectureEtTraitement;
  FermetureFichier;
end.

```

### **Programme C (DevC++ 4)**

```

#include <string.h>
#include <stdio.h>

FILE          *FichierEntree; /* Fichier TEXT pour l'entrée */
FILE          *FichierSortie; /* Fichier TEXT pour la sortie */

void OuvertureFichier()

```

```

{
    if ((FichierEntree = fopen("DIFFFINCH.IN", "r")) == NULL)
    {
        perror("DIFFFINCH.IN");
        exit(1);
    }
    rewind(FichierEntree);
    /* Ouverture de DIFFFINCH.IN en lecture */
    FichierSortie = fopen("DIFFFINCH.OUT", "w");
    if (FichierSortie != NULL)
        rewind(FichierSortie);
    else
        FichierSortie = tmpfile();
    if (FichierSortie == NULL)
    {
        perror("FichierSortie");
        exit(1);
    }
    /* Ouverture de DIFFFINCH.OUT en écriture */
}

void FermetureFichier()
{
    if (FichierEntree != NULL)
        fclose(FichierEntree);
    FichierEntree = NULL;
    if (FichierSortie != NULL)
        fclose(FichierSortie);
    FichierSortie = NULL;
}

void LectureEtTraitement()
{
    unsigned int chiffre[4];
    unsigned int i, j, k, nb, max, min, diff, diffprec;
    int change; /*booléen */

    fscanf(FichierEntree, "%d", &nb);
    for(j=1; j<=nb; j++)
    {
        fprintf(FichierSortie, "Séquence : \n");
        fscanf(FichierEntree, "%d %d %d %d", &chiffre[0], &chiffre[1], &chiffre[2],
&chiffre[3]);
        diff=0;
        do
        {
            diffprec=diff;
            fprintf(FichierSortie, "Chiffres : ");
            for(i=0; i<4; i++) fprintf(FichierSortie, "%d ", chiffre[i]);
            fprintf(FichierSortie, "\n");
            /* tri des chiffres en ordre croissant */
            change=1;
            while (change)
            {
                change=0;
                for(i=0; i<3; i++)
                    if (chiffre[i]>chiffre[i+1])
                    {
                        k=chiffre[i];
                        chiffre[i]=chiffre[i+1];
                        chiffre[i+1]=k;
                        change=1;
                    }
            }
            max=chiffre[3];
            for(i=3; i>0; i--) max=max*10+chiffre[i-1];
            fprintf(FichierSortie, "Plus grand : %d\n", max);
            min=chiffre[0];
            for(i=1; i<4; i++) min=min*10+chiffre[i];
            fprintf(FichierSortie, "Plus petit : %d\n", min);
            diff=max-min;

```

```

        fprintf(FichierSortie,"Différence : %d\n",diff);
        /* remise ne place des nouveaux chiffres */
        k=diff;
        for(i=4;i>0;i--)
            {
                chiffre[i-1]=k % 10;
                k/=10;
            }
        while (diff!=diffprec);
        fprintf(FichierSortie,"Différence inchangée : %d\n",diff); /* arrêt
séquence */
    }
    fprintf(FichierSortie,"\n");
}

int main(int argc, char *argv[])
{
    OuvertureFichier();
    LectureEtTraitement();
    FermetureFichier();

    return 0;
}

```

## B) NOMBRES DÉFICIENTS, PARFAITS, ABONDANTS ET VILAINS...

Le principe est le suivant :

La première chose pour chaque nombre  $x$  lu est de déterminer tous ces diviseurs. Une méthode simple (celle que nous allons employer) consiste à faire une boucle itérative qui va de  $1$  jusqu'à la *racine carrée de  $x$*  avec un pas d'incrément de  $1$  et qui vérifie à chaque fois si nous sommes sur un diviseur de  $x$ . Si tel est le cas nous récupérons le quotient de cette division (l'autre diviseur de  $x$ ).

Pour savoir si le nombre  $x$  est déficient, parfait ou abondant, il faut connaître la somme des diviseurs de  $x$ . Il nous suffit pour cela de posséder une variable qui augmente au fur et à mesure des rencontres de nouveaux diviseurs de  $x$  et qui sera à la fin de la boucle comparée à  $x$  pour savoir à quel type de nombre nous avons affaire.

Pour l'aspect vilain de chaque nombre, nous allons utiliser l'astuce suivante : Nous allons nous servir de deux tableaux de booléens (l'un pour les sommes des diviseurs, l'autre pour les différences) correspondant à l'existence de telle ou telle valeur de somme ou de différence entre deux diviseurs de  $x$ . Au départ, toutes les valeurs des deux tableaux sont initialisés à *faux*. Ensuite (au cours du traitement) une case d'indice  $i$  du tableau des sommes bascule à *vrai* si  $i$  correspond au résultat de la somme des deux nouveaux diviseurs rencontrés, de même pour le tableau des différences.

Ce qui nous permet de regarder ensuite (à l'aide d'une simple opération booléenne) lorsque l'on trouve deux nouveaux diviseurs, si leur différence existe (est *vrai*) dans le tableau de somme et si leur somme existe (est *vrai*) dans le tableau de différence. Si c'est le cas, l'indicateur *vilain* mis initialement à *faux* passe à *vrai*.

La valeur maximale de  $x$  étant  $32500$ , nous pouvons limiter la taille des deux tableaux à  $32502$  ( de  $0$  à  $32501$ ).

A la fin, il suffit d'écrire pour chaque nombre le verdict des tests sur la somme des diviseurs et sur la valeur de *vilain*.

### Programme Pascal (Delphi mode console)

```

program dpav;

{$APPTYPE CONSOLE}

```

```

uses
  SysUtils;

Var
  FichierEntree, FichierSortie : Text;

procedure OuvertureFichier;
begin
  Assign(FichierEntree, 'DPAV.IN');
  Reset(FichierEntree); (* Ouverture de DPAV.IN en lecture *)
  Assign(FichierSortie, 'DPAV.OUT');
  Rewrite(FichierSortie); (* Ouverture de DPAV.OUT en écriture *)
end;

Procedure LectureEtTraitement;
Var
  ds, i, nb, diviseur, quotient, somdiv : integer;
  vilain : boolean;
  som, diff : array[0..32502] of boolean;
Begin
  readln(FichierEntree, ds);
  while (ds > 0) do
  Begin
    Readln(FichierEntree, nb);
    For i:=0 to 32502 do (* mise à faux des drapeaux *)
    Begin
      som[i]:=false;
      diff[i]:=false;
    end;
    vilain:=false;
    somdiv:=-nb; (* soustraction du nombre lui-même *)
    diviseur:=1;
    while (diviseur<=sqrt(nb)) do (* limite de diviseur: la racine carré *)
    begin
      if (nb mod diviseur = 0) then (* on passe les non-diviseurs *)
      begin
        somdiv:=somdiv+diviseur; (* somme des diviseurs *)
        quotient:=nb div diviseur;
        if quotient<>diviseur then (* pas 2 fois la racine carré *)
          somdiv:=somdiv+quotient;
          if (diff[quotient+diviseur] or som[abs(quotient-diviseur)]) then
          begin (* on en a trouvé un nombre vilain *)
            vilain:=true;
          end;
          som[quotient+diviseur]:=true; (* marquage de ces sommes et différences *)
          diff[abs(quotient-diviseur)]:=true;
        end;
        inc(diviseur);
      end;
    if somdiv=nb then
      write(FichierSortie, nb, ' est parfait ') (* nb est parfait *)
    else
      if somdiv<nb then
        write(FichierSortie, nb, ' est déficient ') (* nb est déficient *)
      else
        write(FichierSortie, nb, ' est abondant '); (* nb est abondant *)
    if vilain then
      writeln(FichierSortie, nb, ' et est vilain.') (* nb est vilain *)
    else
      writeln(FichierSortie, nb, ' et n''est pas vilain. '); (* nb ne l'est pas *)
    dec(ds);
  end;
End;

procedure FermetureFichier;
begin
  Close(FichierEntree);
  Close(FichierSortie);
end;

```

```

begin
    OuvertureFichier;
    LectureEtTraitement;
    FermetureFichier;
end.

```

### **Programme C (DevC++ 4)**

```

#include <string.h>
#include <stdio.h>

FILE      *FichierEntree; /* Fichier TEXT pour l'entrée */
FILE      *FichierSortie; /* Fichier TEXT pour la sortie */

void OuvertureFichier()
{
    if ((FichierEntree = fopen("DPAV.IN", "r")) == NULL)
    {
        perror("DPAV.IN");
        exit(1);
    }
    rewind(FichierEntree);
    /* Ouverture de DPAV.IN en lecture */
    FichierSortie = fopen("DPAV.OUT", "w");
    if (FichierSortie != NULL)
        rewind(FichierSortie);
    else
        FichierSortie = tmpfile();
    if (FichierSortie == NULL)
    {
        perror("FichierSortie");
        exit(1);
    }
    /* Ouverture de DPAV.OUT en écriture */
}

void FermetureFichier()
{
    if (FichierEntree != NULL)
        fclose(FichierEntree);
    FichierEntree = NULL;
    if (FichierSortie != NULL)
        fclose(FichierSortie);
    FichierSortie = NULL;
}

void LectureEtTraitement()
{
    int ds, i, nb, diviseur, quotient, somdiv;
    int vilain; /* booléen */
    int som[32502], diff[32502]; /* booléen */

    fscanf(FichierEntree, "%d", &ds);
    while (ds-->0) do
    {
        fscanf(FichierEntree, "%d", &nb);
        for(i=0; i<=32501; i++) /* mise à faux des drapeaux */
        {
            som[i]=0;
            diff[i]=0;
        }
        vilain=0;
        somdiv=-nb; /* soustraction du nombre lui-même */
        diviseur=1;
        while (diviseur<=sqrt(nb)) /* limite de diviseur: la racine carré */
        {
            if (!(nb % diviseur)) /* on passe les non-diviseurs */
            {
                somdiv+=diviseur; /* somme des diviseurs */
                quotient=nb/diviseur;
                if (quotient!=diviseur) /* pas 2 fois la racine carré */

```

```

        somdiv=somdiv+quotient;
    if ((diff[quotient+diviseur] || (som[abs(quotient-diviseur)]))
        vilain=1; /* on en a trouvé un nombre vilain */
        vilain=true;
        som[quotient+diviseur]=1; /* marquage des sommes et différences */
        diff[abs(quotient-diviseur)]=1;
    }
    diviseur++;
}

if (somdiv==nb) /* nb est parfait */
    fprintf(FichierSortie,"%d est parfait ",nb);
else
    if (somdiv<nb) /* nb est déficient */
        fprintf(FichierSortie,"%d est déficient ",nb);
    else /* nb est abondant */
        fprintf(FichierSortie,"%d est abondant ",nb);

if (vilain) /* nb est vilain */
    fprintf(FichierSortie,"et est vilain.\n");
else /* nb n'est pas vilain */
    fprintf(FichierSortie,"et n'est pas vilain.\n");
}
}

int main(int argc, char *argv[])
{
    OuvertureFichier();
    LectureEtTraitement();
    FermetureFichier();

    return 0;
}

```