

CORRECTIONS DES EXERCICES DU CONCOURS D'ENTREE EPITA 2007

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) LA DIVISION ...

Le principe est le suivant :

Pour effectuer le calcul, nous allons utiliser deux nombres entiers (nomb et divi) lus dans le fichier d'entrée (GDIV.IN) nous permettant de stocker à chaque tour, respectivement, le dividende et le diviseur. La suite est assez simple, pour soustraire à gauche, nous allons multiplier le diviseur temporaire (divitemp) par 10 autant de fois que nécessaire pour l'amener à la même unité que le dividende (nomb).

Ensuite, nous soustrayons le diviseur (divitemp) du dividende (nomb) autant de fois que possible et nous mémorisons ce nombre de fois dans un quotient temporaire (quottemp), ce qui nous donne une nouvelle valeur de dividende (nomb). Le quotient (quot) initialisé à 0 est mis à jour par sa multiplication par 10 augmenté de la valeur du quotient temporaire (quottemp).

Après avoir divisé le diviseur temporaire (divitemp) par 10, nous recommençons la phase précédente jusqu'à ce que le diviseur (divi) devienne supérieur au dividende restant (nomb).

A ce stade du traitement, nous sommes en possession du quotient (quot) et du reste (nomb), il ne reste plus alors qu'à les écrire dans le fichier de sortie (GDIV.OUT) et de recommencer pour un nouveau couple de valeurs s'il en reste à traiter.

Programme Pascal (Delphi studio 7 mode console)

program gdiv;

```
{ $APPTYPE CONSOLE }

uses
  SysUtils;

Var
  FichierEntree, FichierSortie : Text;

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

Procedure LectureEtTraitement;
Var
  ds, nomb, divi, divitemp, quot, quottemp : integer;
```

```

Begin
  readln(FichierEntree, ds);
  while (ds > 0) do
  Begin
    Readln(FichierEntree, nomb);
    Readln(FichierEntree, divi);
    quot:=0;

    if divi>nomb then          (* le diviseur est + grand que le dividende *)
    begin
      writeln(FichierSortie,0);
      writeln(FichierSortie,nomb);
    end
    else
    begin
      divitemp:=divi;
      while (nomb>=divitemp) do divitemp:=divitemp*10;
      divitemp:=divitemp DIV 10;

      quottemp:=0;
      while (divi<=nomb) do
      begin
        while (divitemp<=nomb) do
        begin
          nomb:=nomb-divitemp;
          quottemp:=quottemp+1;
        end;
        quot:=10*quot+quottemp;
        divitemp:=divitemp DIV 10;
        quottemp:=0;
      end;

      writeln(FichierSortie, quot);
      writeln(FichierSortie, nomb);

    end;
    dec(ds);
  end;
End;

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

begin
  OuvertureFichier;
  LectureEtTraitement;
  FermetureFichier;
end.

```

Programme C (DevC++ 4.9.9.2)

```

#include <stdio.h>
#include <stdlib.h>

FILE          *FichierEntree;    /* Fichier TEXT pour l'entrée */
FILE          *FichierSortie;    /* Fichier TEXT pour la sortie */
int cval[256];

void OuvertureFichier()
{
  if ((FichierEntree = fopen("GDIV.IN", "r")) == NULL)
  {

```

```

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

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

void LectureEtTraitement()
{
    int ds, nomb, divi, divitemp, quot, quottemp;

    fscanf(FichierEntree,"%d", &ds);
    while (ds > 0)
    {
        fscanf(FichierEntree, "%d %d", &nomb, &divi);
        quot=0;

        if (divi>nomb)          /* le diviseur est + grand que le dividende */
        {
            fprintf(FichierSortie,"%d\n",0);
            fprintf(FichierSortie,"%d\n",nomb);
        }
        else
        {
            divitemp=divi;
            while (nomb>=divitemp) divitemp*=10;
            divitemp/=10;

            quottemp=0;
            while (divi<=nomb)
            {
                while (divitemp<=nomb)
                {
                    nomb-=divitemp;
                    quottemp++;
                }
                quot=10*quot+quottemp;
                divitemp/=10;
                quottemp=0;
            }

            fprintf(FichierSortie,"%d\n",quot);
            fprintf(FichierSortie,"%d\n",nomb);
        }
        ds--;
    }
}

```

```

    }
}

int main(int argc, char *argv[])
{
    OuvertureFichier();
    LectureEtTraitement();
    FermetureFichier();
    return 0;
}

```

B) LES PARTITIONS ...

Le principe est le suivant :

Pour chaque couple de nombre lus (k et i) dans le fichier d'entrée (PART.IN) nous allons déterminer la $i^{\text{ème}}$ partition du nombre k. Pour cela, nous utilisons un vecteur d'entiers p (la limite est fixée arbitrairement à 1024) dont nous initialisons les k premiers éléments à 1 (ce qui correspond à la première partition).

Pour déterminer la partition suivante, nous partons du dernier élément et nous le comparons avec son prédécesseur. Tant qu'ils sont égaux, nous remontons sur l'élément précédent et effectuons cette même comparaison jusqu'à en trouver deux de valeurs différentes. Nous appellerons cet élément le point de contraction.

Une fois déterminé, le dernier élément égal aux précédents est sommé avec la valeur qui le suit dans la partition courante et il est ajouté derrière autant de 1 que nécessaire pour que la somme de tous les éléments de la nouvelle partition donne k.

Si jamais nous sommes sur le dernier élément, c'est le premier qui est alors considéré comme point de contraction. Il se voit alors augmenté de 1 et l'on rajoute là aussi le nombre de 1 nécessaire à atteindre une somme égale à k.

Parallèlement, un compteur (partact) compte les partitions déterminées. L'arrêt du traitement intervient lorsque ce compteur (partact) est égal à i (la partition à déterminer) ou lorsque nous sommes sur la dernière partition possible (quand le premier élément de p est égal à k) et que le nombre de partitions déterminées est inférieur à celle demandée.

Dans le premier cas, la partition est écrite dans le fichier de sortie (PART.OUT), dans le deuxième, il est écrit dans le même fichier que le i est trop grand.

Programme Pascal (Delphi Studio 7 mode console)

```

program part;

{$APPTYPE CONSOLE}

uses
    SysUtils;

Var
    FichierEntree, FichierSortie : Text;

procedure OuvertureFichier;
begin
    Assign(FichierEntree, 'PART.IN');
    Reset(FichierEntree);
    (* Ouverture de PART.IN en lecture *)

```

```

Assign(FichierSortie,'PART.OUT');
Rewrite(FichierSortie);          (* Ouverture de PART.OUT en écriture *)
end;

Procedure LectureEtTraitement;
Var
  ds, k, kt, i, j, s, partact : integer;
  p : array[1..1024] of integer;
Begin
  readln(FichierEntree,ds);      (* nombre de partition à calculer *)
  while (ds > 0) do
  Begin
    Readln(FichierEntree, k, i); (* récupération de k et de i *)
    kt:=k;                       (* utilisation d'un k temporaire kt *)
    partact:=1;                   (* N° de la partition actuelle *)
    for j:=1 to k do p[j]:=1;     (* 1ere partition *)
    while (partact<i) and (p[1]<>k) do
    begin
      (* calcul de la partition suivante *)
      (* repérage du pt de contraction *)
      inc(partact);;
      j:=kt;
      s:=p[j];;
      while (j>1) and (p[j]=p[j-1]) do
      begin
        dec(j);;
        s:=s+p[j];;
      end;
      (* contraction *)
      if j<kt then
      begin
        p[j]:=p[j]+1;           (* augmentation de la valeur de 1 *)
        s:=s-p[j];             (* la somme est diminuée de cette valeur *)
        while (s>0) do        (* on comble de 1 le reste de la somme *)
        begin
          inc(j);
          p[j]:=1;
          dec(s);
        end;
        kt:=j;                 (* nouvelle limite *)
      end
      else                      (* si on est sur la dernière valeur *)
      begin
        p[1]:=p[1]+1;
        s:=k-p[1];            (* reprise à partir de la valeur k *)
        j:=1;
        while (s>0) do        (* on comble de 1 le reste de la somme *)
        begin
          inc(j);
          p[j]:=1;
          dec(s);
        end;
        kt:=j;
      end;
    end;
  end;
  if partact=i then
  begin
    write(FichierSortie,'(');
    j:=1;
    while (j<kt) do
    begin
      write(FichierSortie,p[j],',');
      inc(j);
    end;
    write(FichierSortie,p[j]);
    writeln(FichierSortie,')');
  end;

```

```

    end
    else writeln(FichierSortie,'i trop grand');

    dec(ds);
end;
End;

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

begin
    OuvertureFichier;
    LectureEtTraitement;
    FermetureFichier;
end.

```

Programme C (DevC++ 4.9.9.2)

```

#include <stdio.h>
#include <stdlib.h>

FILE          *FichierEntree; /* Fichier TEXT pour l'entrée */
FILE          *FichierSortie; /* Fichier TEXT pour la sortie */
int cval[256];

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

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

void LectureEtTraitement()
{
    int ds, k, kt, i, j, s, partact, p[1024];

    fscanf(FichierEntree,"%d", &ds);          /* nombre de partition à calculer */
    while (ds > 0)

```

```

{
fscanf(FichierEntree,"%d %d", &k, &i);      /* récupération de k et de i */
kt=k;                                       /* utilisation d'un k temporaire kt */
partact=1;                                  /* N° de la partition actuelle */
for(j=1;j<=k;j++) p[j]=1;                  /* 1ere partition */
while (partact<i && p[1]!=k)
{
/* calcul de la partition suivante */
/* repérage du pt de contraction */
partact++;
j=kt;
s=p[j];
while (j>1 && p[j]==p[j-1])
{
j--;
s+=p[j];
}
/* contraction */
if (j<kt)
{
p[j]++;                                     /* augmentation de la valeur de 1 */
s-=p[j];                                   /* la somme est diminuée de cette valeur */
while (s>0)                                /* on comble de 1 le reste de la somme */
{
j++;
p[j]=1;
s--;
}
kt=j;                                       /* nouvelle limite */
}
else                                        /* si on est sur la dernière valeur */
{
p[1]++;
s=k-p[1];                                  /* reprise à partir de la valeur k */
j=1;
while (s>0)                                /* on comble de 1 le reste de la somme */
{
j++;
p[j]=1;
s--;
}
kt=j;
}
}
if (partact=i)
{
fprintf(FichierSortie,"(");
j=1;
while (j<kt)
{
fprintf(FichierSortie,"%d,",p[j]);
j++;
}
fprintf(FichierSortie,"%d",p[j]);
fprintf(FichierSortie,")");
}
else fprintf(FichierSortie,"i trop grand");
ds--;
}
}

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

```

```
    return 0;  
}
```