TD 13
Recherche du plus grand plateau

Algorithmique
Semestre 1

1 Recherche des occurences

1.1 En-têtes des sous-programmes

1.1.1

```
type Couple = enregistrement
  valeur <Entier>,
  nbValeur <Entier>; --nombre d'occurence

constante CMAX <Entier> = 100;

type TabCouples = tableau[1..CMAX] de <Couples>;
```

Listing 1 – Le type TabCouples

1.1.2

```
--recherche s'il existe le rang de l'occurence d'une valeur v dans un taleau
    tabOcc de n elements
-- s'il n'existe pas rang nous indique la position qu'elle devrait occuper
procedure rechercherOccurence (entree tabOccs <TabCouples>,
              entree n <Entier>,
              entree v <Entier>,
              sortie existe <Booleen>,
              sortie rang <Entier>);
```

Listing 2 – En-tête de rehercherOccurence

1.1.3

```
--ajoute au rang "rang" une nouvelle entree dans le tableau tabOccs avec un
    nombre d'occurence egal a un
procedure enregistrerOccurence (maj tabOccs <TabCouples>,
                maj n <Entier>,
                entree rang <Entier>,
                entree v <Entier>);
```

Listing 3 – En-tête de enregistrerOccurence

1.1.4

```
-- augmente d'une unite le nombre d'occurence du couple se trouvant au rang "rang"
-- dans un tableau de nbOccs elements
procedure ajouterOccurence (maj tabOccs <TabCouples>,
                entree nbOccs <Entier>, -- ne sert a rien ^_^
                entree rang <Entier>);
```

Listing 4 – En-tête de ajouterOccurence

1.2 Programme

```
importer entreeSortie;

type Couple = enregistrement
  valeur <Entier>,
  nbValeur <Entier>; --nombre d'occurence

constante CMAX <Entier> = 100;

type TabCouples = tableau[1..CMAX] de <Couples>;

programme suite

glossaire
  v <Entier>; -- valeur courante de la liste
  tabOcc <TabCouples>; -- tableau des couples
  nbOcc <Entier>; -- nb d'elements du tableau tabOccs
  trouve <Booleen>; -- indicateur d'existence de la valeur v
  rang <Entier>; -- positon effective ou eventuelle de v dans tabOccs

debut
  nbOccs <- 0;
  lire(v);

  tantque v /= 0 faire
    rechercherOccurence (tabOccs, nbOccs, v, trouve, rang);
    si trouve alors
      ajouterOccurence(tabOccs, nbOccs, rang);
    sinon
      enregistrerOccurence(tabOccs, nbOccs, rang, valeur);
    fin si;
    lire(v);
  fin tantque;
  ecrireCouples(tabOccs, nbOccs);
fin
```

Listing 5 – Programme

1.3 Corps des sous-programme

1.3.1

```
--recherche s'il existe le rang de l'occurence d'une valeur v dans un taleau tabOcc de n elements
-- s'il n'existe pas rang nous indique la position qu'elle devrait occuper
```

```
procedure rechercherOccurence (entree tabOccs <TabCouples>,
                entree n <Entier>,
                entree v <Entier>,
                sortie existe <Booleen>,
                sortie rang <Entier>)
glossaire
  i <Entier>;

debut
  tantque v > tab[i].valeur et i <= n faire
    i <- i + 1;
  fin tantque;
  si v = tab[i].valeur alors
    existe = VRAI;
    rang <- i;
  sinon
    existe <- FAUX;
  fin si;
fin
```

Listing 6 – Corps de rehercherOccurence

1.3.2

```
--ajoute au rang "rang" une nouvelle entree dans le tableau tabOccs avec un
    nombre d'occurence egal a un
procedure enregistrerOccurence (maj tabOccs <TabCouples>,
                maj n <Entier>,
                entree rang <Entier>,
                entree v <Entier>)
glossaire
  i <Entier>;

debut
  n <- n + 1;
  i <- n;

  tantque i > rang faire
    tabOccs[i].valeur <- tabOccs[i-1].valeur;
    tabOccs[i].nbVal <- tabOccs[i-1].nbVal;
    i <- i - 1;
  fin tantque;
  tabOccs[i].valeur <- v;
  tabOccs[i].nbVal <- i;
fin
```

Listing 7 – Corps de enregistrerOccurence

2 Recherche des plateaux

2.1 Algorithme

```
--Determiner la longueur et la valeru de tous les plateaux d'une suite
lire la premiere valeur de la ligne;
creer un plateau de longueur un avec cette valeur;

```

```
tantque valeur /= valeurArret faire
  lire la valeur suivante de la suite;
  si valeur suivante = valeur alors
    comptabiliser la valeur suivante dans le plateau en cours;
  sinon
    Ecrire la longueur du plateau en cours;
    Ecrire la valeur du plateau en cours;
    Creer un nouveau plateau de longueur un avec la valeur suivante;

  fin si;
fin tantque;
```

2.2 Programme

```
importer entreeSortie;
Programme rechercherPlateau
glossaire
  valeur <Entier>;
  valeurPrecedente <Entier>;
  lgPlateau <Entier>;
  valPlateau <Entier>;

debut
  lire(valeur);
  lgPlateau <- 1;
  valPlateau <- v;

  tantque valeur /= 0 faire
    lgPlateau <- 0;
    valPlateau <- valeur;
    valeurPrecedente <- valeur;
    tantque valeur = valeurPrecedente faire
      lgPlateau <- lgPlateau + 1;
      lire(valeur);
    fin tantque;
    ecrire(lgPlateau);
    ecrire(valPlateau);
  fin tantque;
fin
```

3 Recherche du plus long plateau

3.1 Programme

```
importer entreeSortie;
Programme rechercherPlusLongPlateau
glossaire
  valeur <Entier>; --valeur d'un element de la suite
  valeurPrecedente <Entier>;
  lgPlateau <Entier>;--longueur du plateau courant
  valPlateau <Entier>;-- valeur du plateau courant
  lgMaxPlateau <Entier>; -- longueur du plus grand plateau
  valMaxPlateau <Entier>; -- valeur du plateau le plus long
debut
  lgMaxPlateau <- 0;
```

```
12    lire(valeur);
13
14    tantque valeur /= 0 faire
15      lgPlateau <- 0;
16      valPlateau <- valeur;
17      valeurPrecedente <- valeur;
18      tantque valeur = valeurPrecedente faire
19        lgPlateau <- lgPlateau + 1;
20        lire(valeur);
21      fin tantque;
22      si lgPlateau > lgMaxPlateau alors
23        lgMaxPlateau <- lgPlateau;
24        valMaxPlateau <- valPlateau;
25      fin si;
26    fin tantque;
27    si lgMaxPlateau = 0 alors
28      ecrire("Pas d'element dans la liste");
29    sinon
30      ecrire(lgMaxPlateau);
31      ecrire(valMaxPlateau);
32    fin si;
33  fin
```

Listing 8 – Programme rechercherPlusLongPlateau

3.2 Du programme au sous-programme

3.2.1

```
1  constante MAX_ELEM <Entier> = 100;
2  type TabEleme = tableau[1 a MAX_ELEM] de <Entier>;
3  type TabSuite = enregistrement
4    elem <TabElem>,
5    nbElem <Entier>;
6  type Couple = enregistrement
7    valeur <Entier>,
8    nbValeur <Entier>;
```

Listing 9 – Type TabSuite

3.2.2

```
1  -- evalue la longueur et la valeur du plus long plateau d'une suite
2  procedure rechercherPlusLongPlateau(entree suite <TabSuite>,
3                    sortie plateau <Couple>);
```

Listing 10 – Entête de la procédure rechercherPlusLongPlateau

3.2.3

```
1  -- evalue la longueur et la valeur du plus long plateau d'une suite
2  procedure rechercherPlusLongPlateau(entree suite <TabSuite>, Sortie plateau
       <Couple>)
3
4  glossaire
5    valPrecedente <Entier>;
6    valeur <Entier>; --valeur d'un element de la suite
```

```
  i <Entier>; --rang de la valeur dans la liste
  lgMaxPlateau <Entier>; -- longueur du plus grand plateau
  valMaxPlateau <Entier>; -- valeur du plateau le plus long
  lgPlateau <Entier>;
  valPlateau <Entier>;

debut
  lgMaxPlateau <- 0;
  si suite.nbElem /= 0 alors
    i <- 1;
    valeur <- suite.Elem[i];
    tantque i <= suite.nbElem faire
      lgPlateau <- 0 ;
      valPrecedente <- valeur;
      valPlateau <- valeur;
      tantque valeur = valPrecedente faire
        lgPlateau <- lgPlateau + 1;
        i <- i + 1;
        valeur <- suite.Elem[i];
      fin tantque;
      si lgPlateau > lgMaxPlateau alors
        lgMaxPlateau <- lgPlateau;
        valMaxPlateau <- valPlateau;
      fin si;
    fin tantque;
    plateau.valeur <- valMaxPlateau;
  fin si;
  plateau.nbValeur <- lgMaxPlateau;
fin
fin
```

Listing 11 – Corps de la procédure rechercherPlusLongPlateau

3.2.4

```
--SOLUTION 1

procedure lireSuite (sortie suite <TabSuite>)

glossaire
  valeur <Entier>; -- valeur courante
  i <Entier>; -- indice de parcous

debut
  i <- 0;
  lire(valeur);

  tantque valeur /= 0 faire
    i <- i + 1 ;
    suite.Elem[i] <- valeur;
    lire(valeur);
  fin tantque;
  suite.nbElem <- i;
fin
```

Listing 12 – Corps de la procédure lireSuite Solution 1

_bis.algo