Le calendrier [dt09] - Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 21 mai 2018

Table des matières

Java - Le calendrier (Solution)

Mots-Clés Dates et Heures, Analyse descendante ■
Requis Structures de base, Structures conditionnelles, Algorithmes paramétrés, Structures répétitives, Schéma itératif ■
Difficulté ●●○ (3 h) ■

Objectif
Cet exercice affiche le calendrier d'un mois et année donnés.
En complément, il affiche l'ensemble des calendriers d'une année donnée.

1 L'algorithme Calendrier

Voici un exemple du résultat attendu :

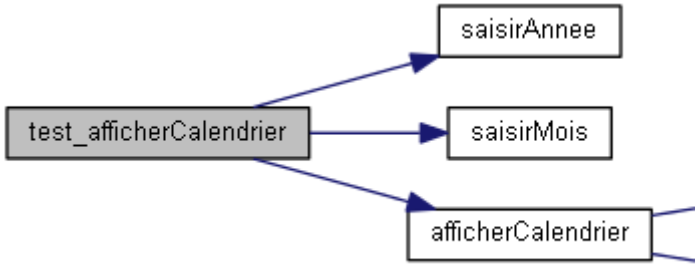
```
Une année (>1592)? 2010
Un mois ([1..12])? 4
Avril 2010
 Dim Lun Mar Mer Jeu Ven Sam
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30
```

Puisqu'il s'agit d'écrire un programme, un bon début est d'essayer de comprendre ce qu'il est censé faire.

En vous basant sur l'exemple, écrivez une procédure `test_afficherCalendrier` qui déclare deux variables entières, une pour le millésime de l'année et l'autre pour le numéro du mois.

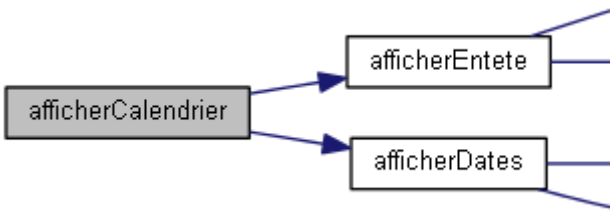
Complétez-la pour qu'ensuite elle appelle :

- Une fonction `saisirAnnee` pour la saisie du millésime de l'année.
- Une fonction `saisirMois` pour la saisie du numéro du mois.
- Une procédure `afficherCalendrier(mois,annee)` pour afficher le calendrier de `(mois,annee)`.



Écrivez ensuite la procédure `afficherCalendrier(mois,annee)` (paramètres d'entiers) qui appelle :

- Une procédure `afficherEntete(mois,annee)` pour afficher l'en-tête de `(mois,annee)`.
- Une procédure `afficherDates(mois,annee)` pour afficher les dates de `(mois,annee)`.



En considérant les deux cas de figures ci-dessous,
Trouvez les deux moyens de décomposer l'affichage des dates.

D	L	M	M	J	V	S
				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	

(a)

D	L	M	M	J	V	S
				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	

(b)

Solution simple
On peut afficher :

- Soit les jours
- Soit les semaines

Les dates seront affichées par boucle. Un des choix pour la boucle correspondant à la figure (a) affichera une seule date à chaque itération ; l'autre correspondant à la figure (b) affichera une semaine entière à chaque itération.

Quels sont les éléments principaux des deux possibilités ?
(c.-à-d. que faut-il déterminer/calculer/tester ?)

Solution simple
Il faut :

- Déterminer où commence le 1er du mois, c.-à-d. déterminer le premier jour semaine.
- Calculer le nombre de jours du mois de l'année.
- Tester quand on passe à la ligne suivante (cas (a)) : fin de semaine.
- Tester si on écrit le jour ou des espaces (cas (b)) : validité d'une date.

Écrivez la procédure `afficherDates(mois,annee)` (paramètres d'entiers) pour qu'elle calcule :

- Dans un entier `p` : le premier jour semaine du numéro de `mois` de l'`annee` par appel à une fonction `premierJSemaine(mois,annee)`.
- Dans un entier `n` : le nombre de jours par appel à la fonction `njours(mois,annee)`.

Affichez les calculs (où `[x]` désigne le contenu de `x`) :

```
Premier jsemaine vaut [p]
Nombre de jours vaut [n]
```

Écrivez la fonction `saisieAnnee` qui renvoie un entier représentant le millésime d'une année.
L'entier doit être saisi **jusqu'à** ce qu'il soit supérieur à 1592.
Affichez l'invite :

```
Une année (>1592)?
```

Écrivez la fonction `saisieMois` qui renvoie un entier représentant un numéro de mois.
L'entier doit être saisi **jusqu'à** ce qu'il soit valide, c.-à-d. compris dans [1..12].
Affichez l'invite :

```
Un mois ([1..12])?
```

Définissez la constante :

- `LGAFFICH=4` (longueur d'affichage des jours, espace séparateur compris)

Écrivez la procédure `afficherEntete(mois,annee)` qui affiche l'en-tête d'un calendrier d'un numéro de mois `mois` (entier) et de millésime `annee` (entier).
Pour l'exemple d'exécution, la procédure affiche :

```
Avril 2010
 Dim Lun Mar Mer Jeu Ven Sam
```

Aide méthodologique

Elle doit traduire un numéro de mois dans le nom du mois et afficher les `(LFAFFICH-1)` premières lettres des jours de la semaine.

Validez vos fonctions et procédures avec la solution.

Solution Java @[pgcalendrier.java]

```java
static int saisieAnnee(){
   Scanner cin = new Scanner(System.in);
   int annee;
   do{
       System.out.print("Millesime d'une annee (>="+UtilsDT.DEBUTGREGCALENDRIER+")? ");
       annee = cin.nextInt();
   } while (!(annee > UtilsDT.DEBUTGREGCALENDRIER));
   return annee;
}
```

```java
static int saisieMois(){
   Scanner cin = new Scanner(System.in);
   int mois;
   do{
       System.out.print("Un numero de mois ([1..12])? ");
       mois = cin.nextInt();
   } while (!(1 <= mois && mois <= 12));
   return mois;
}
```

```java
static void afficherEntete(int mois, int annee){
  System.out.println(UtilsDT.nomMois(mois)+" "+annee);
  for (int js = UtilsDT.PREMIERJSEM; js <= UtilsDT.DERNIERJSEM; ++js){
    System.out.print(" "+UtilsDT.nomJSemaine(js).substring(0,LGAFFICH-1));
  }
  System.out.println();
}
```

```java
static void afficherDates(int mois, int annee){
  int p = UtilsDT.premierJSemaine(mois,annee);
  int n = UtilsDT.njours(mois, annee);
  System.out.println("Premier jsemaine vaut "+p);
  System.out.println("Nombre de jours vaut "+n);
}
```

```java
static void afficherCalendrier(int mois, int annee){
  afficherEntete(mois,annee);
  afficherDates(mois,annee);
}
```

```java
static void test_afficherCalendrier(){
  int annee = saisieAnnee();
  int mois = saisieMois();
  afficherCalendrier(mois,annee);
}
```

2 Procédures et fonctions utilitaires

2.1 Fonction nomMois (nom d'un mois en toutes lettres)

Écrivez une fonction `nomMois(mm)` qui renvoie le nom d'un numéro de mois `mm` en toutes lettres, janvier étant le numéro 1 et décembre le numéro 12. L'entier `mm` est supposé compris dans [1..12].

Validez votre fonction avec la solution.

Solution Java @[UtilsDT.java]

```
/**
  Noms des mois
*/
final static String[] NOMSMOIS =
    {"","Janvier","Fevrier","Mars","Avril","Mai","Juin","Juillet","Aout","Septembre","Octobre","Novemb
```

```
/**
  Nom d'un mois en toutes lettres
  @param[in] mm - numero de mois dans (1=janvier, ..., 12=decembre)
  @return le nom d'un mois en toutes lettres
*/

public static String nomMois(int mm)
{
  return NOMSMOIS[mm];
}
```

2.2 Fonction nomJSemaine (jour de semaine en toutes lettres)

Écrivez une fonction `nomJSemaine(jsem)` qui calcule et renvoie le nom d'un numéro de jour de semaine `jsem` en toutes lettres, dimanche étant le numéro 0 et samedi le numéro 6. L'entier `jsem` est supposé compris dans [0..6].

Validez votre fonction avec la solution.

Solution Java @[UtilsDT.java]

```
/**
  Noms des jours
*/
final static String[] NOMSJOURS =
    {"Dimanche","Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi"};
```

```
/**
  Nom d'un jour de semaine en toutes lettres
  @param[in] jsem - numéro jour de semaine dans (0=dimanche,...,6=samedi)
  @return le nom de jsem en toutes lettres
*/

public static String nomJSemaine(int jsem)
{
  return NOMSJOURS[jsem];
}
```

2.3 Fonction bissextile (test de bissextilité)

Définition

Soit une année postérieure à 1582 (début du calendrier grégorien).
Elle est **bissextile** si et seulement si son millésime est :

- Divisible par 4 mais **non** divisible par 100.
- **Ou** divisible par 400.

Exemples

- 1986 : non (non divisible par 4)
- 1988 : oui (divisible par 4 et non divisible par 100)
- 1900 : non (divisible par 4 et par 100)
- 2000 : oui (divisible par 400)

Écrivez une fonction `bissextile(an)` qui teste et renvoie `Vrai` si le millésime d'une année `an` (entier supérieur à 1582) est bissextile, `Faux` sinon.

Solution Paramètres

Entrants : Un entier `an`

Résultat de la fonction : Un booléen

Solution simple

En tant que *prédicat* (fonction booléenne), le corps de la fonction est simplement constitué de l'**expression booléenne** qui traduit l'énoncé.

Validez votre fonction avec la solution.

Solution Java @[UtilsDT.java]

```java
/**
  Prédicat d'année bissextile
  @param[in] an - millésime d'une annee > DEBUTGREGCALENDRIER=1582
  @return vrai si an est bissextile
*/

public static boolean bissextile(int an)
{
  return (an % 4 == 0 && !(an % 100 == 0)) || (an % 400 == 0);
}
```

2.4 Fonction njours (nombre de jours d'un mois et année)

Propriété

Pour les années postérieures à 1582 (année du calendrier Grégorien) :

- Les mois numéros 4, 6, 9 et 11 ont 30 jours.
- Le mois numéro 2 a 29 jours si l'année est bissextile, 28 sinon.
- Sinon c'est 31 jours pour tous les autres mois 1, 3, 5, 7, 8, 10 et 12.

Écrivez une fonction `njours(mm,an)` qui calcule et renvoie le nombre de jours d'un numéro de mois `mm` (entier compris entre 1 (pour janvier) et 12 (pour décembre)) d'une année `an` (entier supérieur à 1582).

Aide méthodologique

Préférez une structure `Selon`.

Validez votre fonction avec la solution.

Solution Java @[UtilsDT.java]

```java
/**
  Nombre de jour d'un mois et année
  @param[in] mm - numéro de mois dans [1..12]
  @param[in] an - millésime d'une année > DEBUTGREGCALENDRIER
  @return le nombre de jours de (mm,an)
*/

public static int njours(int mm, int an)
{
  int rs = 0;
  switch (mm)
  {
    case 4: case 6: case 9: case 11:
      rs = 30;
      break;
    case 2:
      rs = (bissextile(an) ? 29 : 28);
      break;
    default:
      rs = 31;
  }
  return rs;
}
```

2.5 Fonction premierJSemaine (premier jour semaine)

Ce problème donne l'index du premier jour d'un mois et année donnés (0 signifie Dimanche, 1 Lundi, etc.).

Propriété
Le calcul du jour où tombe le *1er janvier d'une année* est donnée dans l'ouvrage « A Collection of Programming Problems and Techniques », H.A. Maurer and M.R. Williams, Prentice-Hall 1972 :

$$(36 + y + y \text{ div } 4 - y \text{ div } 100 + y \text{ div } 400) \text{ mod NJOURSSEM}$$

avec $y = annee - 1$ et NJOURSSEM$(= 7)$ le nombre de jours dans une semaine.

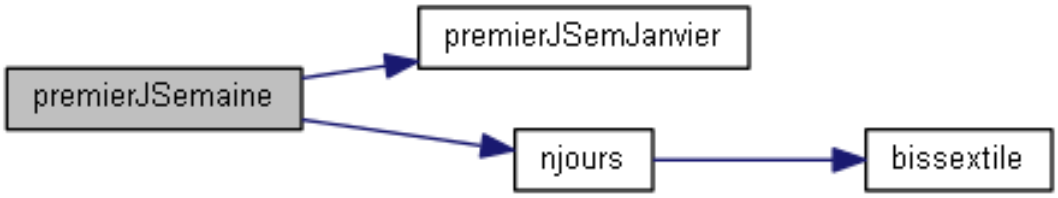
Écrivez une fonction `premierJSemJanvier(annee)` qui calcule et renvoie le premier jour semaine du 1er janvier d'une année `annee` (entier) donnée selon l'algorithme ci-dessus.

Connaissant l'index du premier jour d'une année `an`, comment en calculer le premier jour d'un mois `mm` ?

Aide simple
Admettons que Janvier commence un Mercredi en année `an`, c.-à-d. que `premierJSemJanvier(an)` renvoie 3(=`MER`). Pour calculer sa valeur en Février, on additionne le nombre de jours du mois de Janvier (ici 31). Nous obtenons $34 (= 3 + 31)$ et le reste de la division de 34 par `NJOURSSEM` (ici 7) donne 6 ce qui fait que Février commence un Samedi.

Solution simple
On cumule le nombre de jours des mois de 1 à `mm` **exclus** puis on élimine autant de semaines complètes que possible en appliquant le modulo `NJOURSSEM`.

Écrivez une fonction `premierJSemaine(mois,annee)` qui calcule et renvoie le premier jour semaine d'un mois `mois` (entier) et année `annee` (entier) donnés.



Validez vos fonctions avec la solution.

Solution Java @[UtilsDT.java]

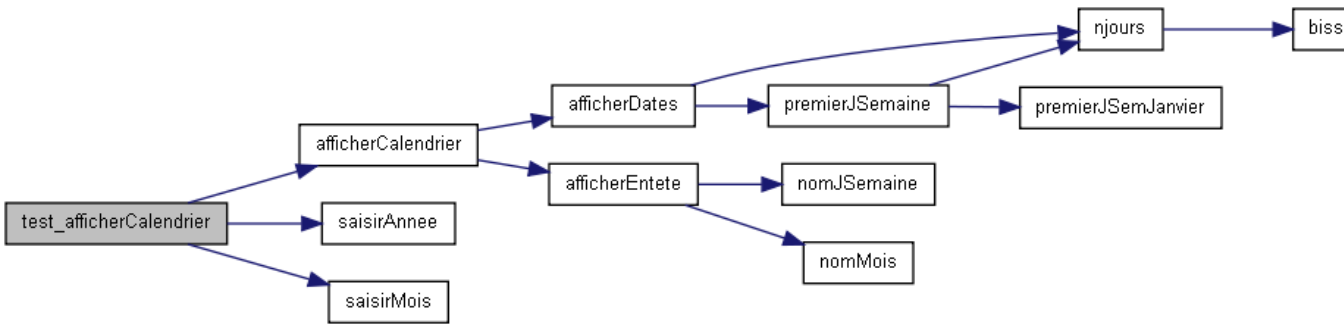
```java
/**
  Nombre de jours semaine
*/
final static int NJOURSSEM = 7;
```

```java
/**
  Premier jour semaine du 1er janvier d'une année
  @param[in] annee - millésime d'une année
  @return le premier jour semaine du 1er janvier de annee
*/

public static int premierJSemJanvier(int annee)
{
  int y = annee - 1;
  return (36 + y + y / 4 + (-y) / 100 + y / 400) % NJOURSSEM;
}
```

```java
/**
  Premier jour semaine d'un mois et année
  @param[in] mm - un numéro de mois
  @param[in] an - millésime d'une année
  @return le premier jour semaine de (mm,an)
*/

public static int premierJSemaine(int mm, int an)
{
  int n = premierJSemJanvier(an);
  int k = 1;
  while (k != mm)
  {
    n += njours(k,an);
    ++k;
  }
  return (n % NJOURSSEM);
}
```

2.6 Test de afficherCalendrier

Voici le graphe final des appels de la procédure de test.



Testez. Exemple d'exécution :

```
Une année (>1592)? 2010
Un mois ([1..12])? 4
Avril 2010
 Dim Lun Mar Mer Jeu Ven Sam
Premier jsemaine vaut 4
Nombre de jours vaut 30
```

Validez votre procédure avec la solution.

3 Les affichages

3.1 Affichage jour par jour

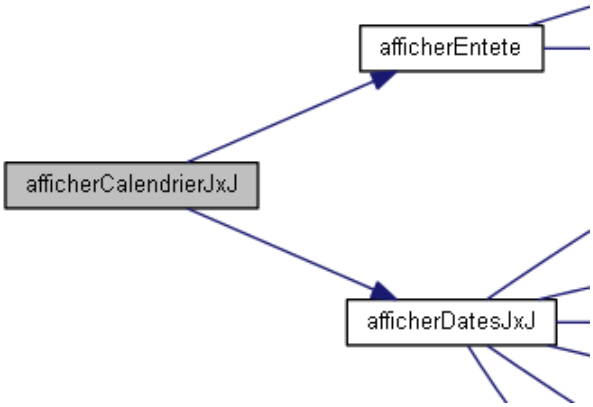
La figure montre l'**affichage jour par jour** des dates dans un mois.
Nous avons vu sa décomposition dans le problème @[L'algorithme Calendrier] :

1. Elle affiche d'abord les blancs au début de la semaine.
2. Une boucle affiche ensuite une seule date à chaque itération.
3. Enfin elle se positionne sur la ligne suivante (si elle n'y est pas déjà).

D	L	M	M	J	V	S
				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	

(a)

Faites un copier/coller de la procédure `afficherCalendrier` en la procédure `afficherCalendrierJxJ` et remplacez l'appel de la procédure `afficherDates` par `afficherDatesJxJ`.



De même, faites un copier/coller de la procédure `afficherDates` en la procédure `afficherDatesJxJ` puis supprimez (ou mettez en commentaires) les **affichages** (et non pas les calculs !) du premier jour semaine `p` et du nombre de jours `n`. L'objet du problème est de réaliser cette procédure.

(1.) Comment déterminer le nombre de blancs devant la première date ?

Solution simple

La valeur de `p` nous dit combien il y a de dates vides à afficher (0 pour Dimanche, 1 pour Lundi et ainsi de suite). Une date étant affichée sur `LGAFFICH` positions (deux chiffres et au moins un espace), il y a donc `LGAFFICH*p` blancs à afficher.

(2.) Que doit faire la boucle qui affiche une date à chaque itération ?

Solution simple

Elle doit :

1. Afficher le numéro de jour de la date.
2. Tester si `p` correspond à une fin de semaine auquel cas il faut se positionner sur la ligne suivante.
3. Incrémenter `p`.

(3.) Comment déterminer si on doit passer à la ligne suivante ?

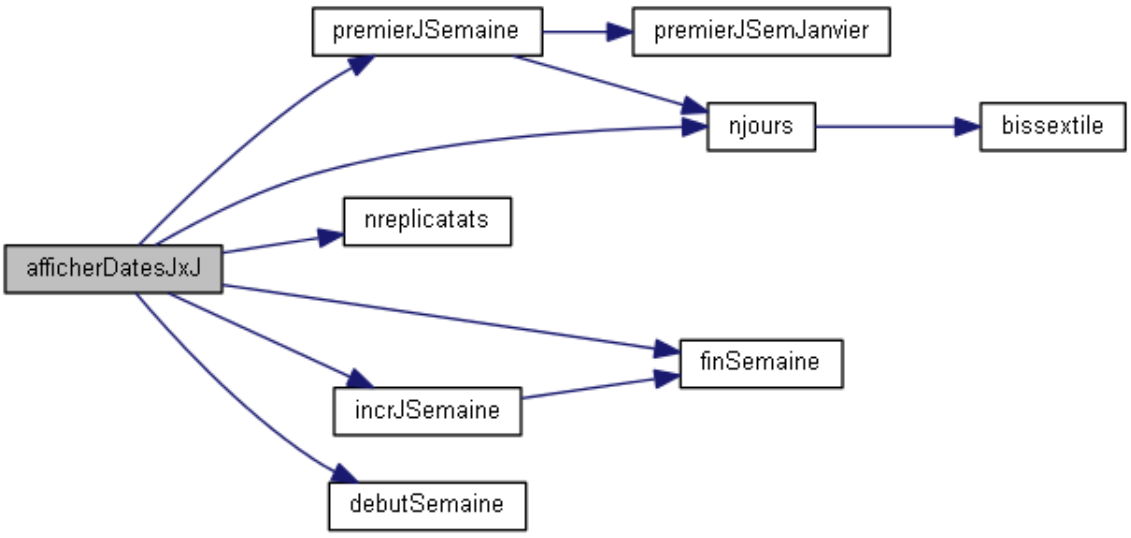
Solution simple

On teste si on est en début de semaine.

Complétez votre procédure `afficherDatesJxJ`. Elle fera appel à :

- (1.) Une fonction `nreplicatats(n,c)` qui renvoie une chaîne de `n` (entier) replicatats d'un caractère `c` pour afficher le nombre de blancs.

- (2.) Une fonction `finSemaine(jsem)` qui renvoie `Vrai` si `jsem` (entier) est une fin de semaine pour tester si `p` est une fin de semaine.
- (2.) Une fonction `suivJSemaine(jsem)` qui renvoie le jour semaine suivant de `jsem` (entier) pour passer au suivant de `p`.
- (3.) Une fonction `debutSemaine(jsem)` qui renvoie `Vrai` si `jsem` (entier) est un début de semaine pour tester si `p` est un début de semaine.



Écrivez la fonction `nreplicatats(n,c)` qui renvoie une chaîne de `n` (entier) replicatats d'un caractère `c`.

Écrivez la fonction `suivJSemaine(jsem)` qui renvoie le jour semaine suivant de `jsem` (entier). Rappel : un jour de semaine est un entier dans [0..6] avec 0=Dimanche, 1=Lundi, ..., 6=Samedi.

Écrivez la fonction `debutSemaine(jsem)` qui teste et renvoie `Vrai` si un jour de semaine `jsem` (entier) est un début de semaine, `Faux` sinon.

De même, écrivez la fonction `finSemaine(jsem)` qui teste et renvoie `Vrai` si `jsem` (entier) est une fin de semaine, `Faux` sinon.

Validez vos procédures et vos fonctions avec la solution.

Solution Java @[pgcalendrier.java]

```
static boolean debutSemaine(int jsem){
  return (jsem == UtilsDT.PREMIERJSEM);
}
```

```
static boolean finSemaine(int jsem){
  return (jsem == UtilsDT.DERNIERJSEM);
}
```

```
static int suivJSemaine(int jsem){
  return (finSemaine(jsem) ? UtilsDT.PREMIERJSEM : jsem + 1);
}
```

```
static void afficherDatesJxJ(int mois, int annee){
  int p = UtilsDT.premierJSemaine(mois, annee);
  System.out.print(nreplicatats(LGAFFICH*p, ' '));
  int n = UtilsDT.njours(mois, annee);
  for (int j = 1; j <= n; ++j){
    System.out.printf("%4d",j);
    if (finSemaine(p)){
      System.out.println();
    }
    p = suivJSemaine(p);
  }
  if (!debutSemaine(p)){
    System.out.println();
  }
  System.out.println();
}
```

```
static void afficherCalendrierJxJ(int mois, int annee){
  afficherEntete(mois, annee);
  afficherDatesJxJ(mois, annee);
}
```

Faites un copier/coller de la procédure `test_afficherCalendrier` en la procédure `test_afficherCalendrierJxJ` puis remplacez l'appel de l'affichage du calendrier (suffixe `JxJ`).

Testez. Exemple d'exécution :

```
Une année (>1592)? 2010
Un mois ([1..12])? 4
Avril 2010
 Dim Lun Mar Mer Jeu Ven Sam
                   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
```

Validez votre procédure avec la solution.

Solution Java @[pgcalendrier.java]

```
static void test_afficherCalendrierJxJ(){
  int annee = saisieAnnee();
  int mois = saisieMois();
  afficherCalendrierJxJ(mois,annee);
}
```

3.2 Affichage semaine par semaine

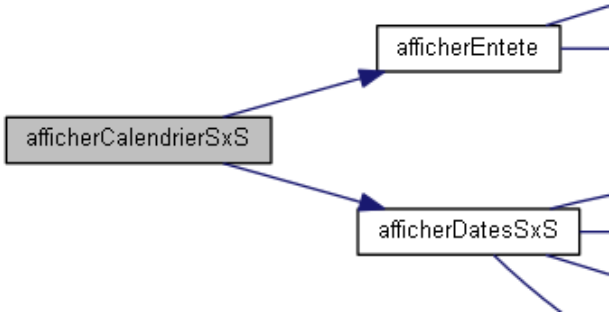
La figure montre l'**affichage semaine par semaine** des dates du mois.
De même, nous avons vu sa décomposition (cf. @[L'algorithme Calendrier]) :

1. Elle détermine d'abord combien de semaines doivent être affichées.
2. Puis elle utilise une boucle pour afficher les semaines.

D	L	M	M	J	V	S
				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	

(b)

Faites un copier/coller de la procédure `afficherCalendrier` en la procédure `afficherCalendrierSxS` et remplacez l'appel de l'affichage des dates.



De même, faites un copier/coller de la procédure `afficherDates` en la procédure `afficherDatesSxS` puis supprimez (ou mettez en commentaires) les **affichages** du premier jour semaine `p` et du nombre de jours `n`. L'objet du problème est de réaliser cette procédure.

(1.) Comment déterminer le nombre de semaines d'un mois et année donnés ?

Aide simple
Pour l'exemple, on a :
4 (pour `premierJSemaine`) + 30 (pour `njours`) = 34 soit 5(=34/7) semaines.

Solution simple

Le nombre de semaines dépend du nombre total de jours dans le mois (fonction `njours`) augmenté du nombre de dates vides au début de la première semaine (fonction `premierJSemaine`).

Pour trouver le nombre de semaines (groupes de `NJOURSSEM`) que compte le mois, nous devons diviser la somme des dates vides et du nombre de jours par `NJOURSSEM`.

(2.) L'affichage d'une semaine consiste à afficher systématiquement 7 jours.

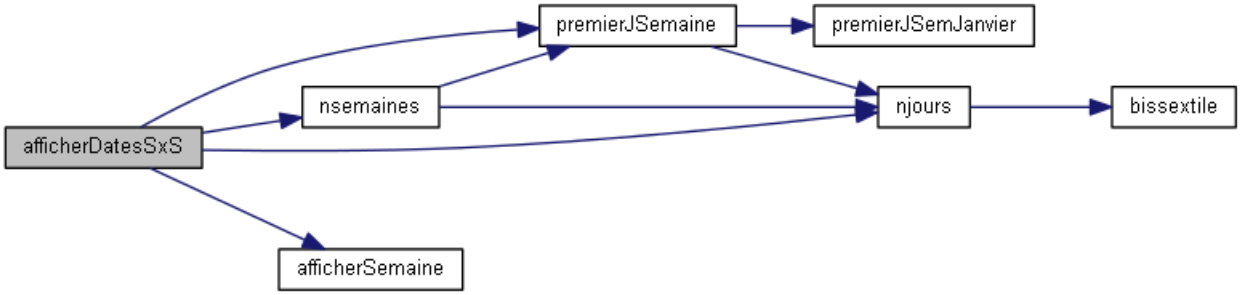
Mais comment déterminer s'il faut afficher le jour ou des blancs ?
Déduisez les paramètres utiles à la procédure.

Solution simple

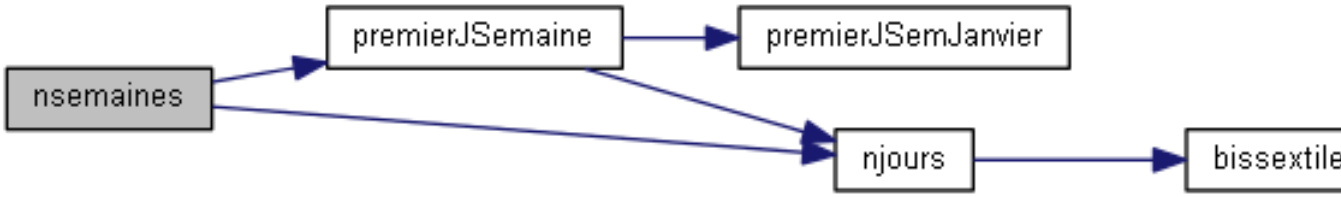
Il faut pouvoir tester si le jour est valide, c.-à-d. compris entre 1 et le nombre de jours du mois. Dans l'affirmative, on affiche le numéro du jour. Sinon on affiche des blancs.

Complétez votre procédure `afficherDatesSxS`. Elle fera appel à :

- (1.) Une fonction `nsemaines(mois,annee)` qui donne le nombre de semaines d'un `mois` (entier) et `annee` (entier) donnés pour calculer le nombre de semaines.
- (2.) Une procédure `afficherSemaine(jour,nj)` qui affiche une semaine à partir d'un numéro de `jour` (entier), le nombre de jours étant `nj` (entier) pour afficher une semaine.



Écrivez la fonction `nsemaines(mois,annee)` qui calcule et renvoie le nombre de semaines d'un numéro de mois `mois` (entier) et de millésime `annee` (entier).



Écrivez la procédure `afficherSemaine(jour,nj)` qui affiche une semaine complète, c.-à-d. `NJOURSSEM=7` jours, à partir d'un numéro de `jour` (entier), le nombre de jours du calendrier étant `nj` (entier).

Orientation

Testez si le `jour` est compris dans `[1..nj]`. Dans l'affirmative, affichez `jour` (sur `LGAFFICH` positions), sinon affichez `LGAFFICH` blancs.

Validez vos procédures et vos fonctions avec la solution.

Solution Java @[pgcalendrier.java]

```
static int nsemaines(int mois, int annee){
  int p = UtilsDT.premierJSemaine(mois, annee);
  int n = UtilsDT.njours(mois, annee);
  return (p+n+UtilsDT.NJOURSSEM-1)/UtilsDT.NJOURSSEM;
}
```

```
static void afficherSemaine(int jour, int nj){
  for (int k = 1; k <= UtilsDT.NJOURSSEM; ++k){
    if (1 <= jour && jour <= nj){
      System.out.printf("%4d",jour);
    }
    else{
      System.out.print(nreplicatats(LGAFFICH, ' '));
    }
    jour += 1;
  }
  System.out.println();
}
```

```
static void afficherDatesSxS(int mois, int annee){
  int p = UtilsDT.premierJSemaine(mois, annee);
  int n = UtilsDT.njours(mois, annee);
  int ns = nsemaines(mois, annee);
  int j = 1 - p;
  for (int s = 1; s <= ns; ++s){
    afficherSemaine(j, n);
    j += UtilsDT.NJOURSSEM;
  }
  System.out.println();
}
```

```
static void afficherCalendrierSxS(int mois, int annee){
  afficherEntete(mois, annee);
  afficherDatesSxS(mois, annee);
}
```

Faites un copier/coller de la procédure `test_afficherCalendrier` en la procédure `test_afficherCalendrierSxS` puis remplacez l'appel de l'affichage du calendrier (suffixe `SxS`).

Testez. Exemple d'exécution :

```
Une année (>1592)? 2010
Un mois ([1..12])? 4
Avril 2010
```

```
 Dim Lun Mar Mer Jeu Ven Sam
                   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
```

Validez votre procédure avec la solution.

Solution Java @[pgcalendrier.java]

```
static void test_afficherCalendrierSxS(){
  int annee = saisieAnnee();
  int mois = saisieMois();
  afficherCalendrierSxS(mois,annee);
}
```

3.3 Calendrier annuel

Écrivez une procédure `test_afficherDates` qui saisit le millésime d'une année puis affiche le calendrier des six premiers mois avec la procédure d'affichage du calendrier `JxJ` et ceux des six derniers mois avec celle `SxS`.

Testez.

Validez votre procédure avec la solution.

Solution Java @[pgcalendrier.java]

```
static void test_afficherCalendriers(){
  int annee = saisieAnnee();
  for (int mois = 1; mois <= 6; ++mois){
    afficherCalendrierJxJ(mois, annee);
  }
  for (int mois = 7; mois <= 12; ++mois){
    afficherCalendrierSxS(mois, annee);
  }
}
```

Comment comparer les deux versions `afficherCalendrier` ?

Solution simple

Les programmes peuvent être comparés de différentes manières. L'efficacité est un critère évident – quel programme s'exécute le plus vite ou utilise le moins de mémoire ? Cependant, l'efficacité a rarement autant d'importance que la lisibilité et la facilité de modification. Les programmes lisibles se comprennent facilement. Les programmes modifiables peuvent être mis au point ou modifiés par d'autres concepteurs.

La solution jour par jour `JxJ` est très lisible puisque sa décomposition reflète une manière bien naturelle de se représenter le calendrier d'un mois par un en-tête suivi d'une série de jours. Elle est aussi la plus courte des deux solutions.

La solution semaine par semaine `SxS` apparaît moins lisible. Elle utilise la notion de date négative. Sa décomposition d'un mois en une suite de semaines est à première vue quelque chose contre nature.

Mais regardons la facilité de modification :

- On peut vouloir afficher les Dimanches en caractères spéciaux.
- Les dates peuvent être affichées en format pavé.
- Le calendrier peut être affiché au centre de la page plutôt que contre la marge de gauche.
- Pour les mois où le 31 tombe un Dimanche, il peut afficher quelque chose comme 31/31.

Toutes ces améliorations seront relativement faciles à ajouter à la version semaine par semaine. Par contre, elles le sont nettement moins dans la version jour par jour.

On peut donc trouver des avantages à chacune des méthodes. Le choix dépend de l'utilisation prévue pour le programme.

4 Références générales

Comprend [Clancy-AL1 :c3] ■