température est précédée de l'année d'observation, puis du mois et enfin du jour. Examinez bien le fichier TXT avant de rédiger le programme qui construira votre table.

L'exercice 2.7 ne pose pas de difficultés réelles mais doit vous faire comprendre que l'utilisation de @@ ne se limite pas aux fichiers de données ne contenant qu'une seule ligne. @@ ne pourra cependant pas gérer les données présentées dans le résultat 2.4 pour obtenir la table ci-contre.

ultat 2.4 (données)

```
1 . . . 24 26.5 21 19.8
2 19.8 17.6 20.5 23.2 25.8 24.6 23.1
3 22.4 21.9 20.3 18.8 19.1 17.8 17.6
4 20.4 22.3 19 18 17.2 17.3 19
5 18.7 21.7 21.3 18.5 19.2 22.8 .
```

Résultat 2.5 (extrait)

Obs	jour_sem	sem	temp
1	jeudi	sem1	24.0
2	vendredi	sem1	26.5
3	samedi	sem1	21.0
4	dimanche	sem1	19.8
5	lundi	sem2	19.8

Vous aurez besoin de l'option @ d'INPUT pour gérer ces données. Cette option permet d'interrompre momentanément la lecture de l'enregistrement présent dans l'INPUT BUFFER et d'en reprendre la lecture. Les programmes 2.19 et 2.20 doivent vous permettre de comprendre l'intérêt que peut présenter cette option.

gramme 2.19

```
A test;
INPUT X1;
INPUT X2;
INPUT X3;
DS;
 3 4
 7 8
 2 1
```

Résultat 2.6

Obs	X1	X2	X3
1	1	5	4

Programme 2.20

```
DATA test;
   INPUT X1 @;
   INPUT X2 @;
   INPUT X3 @;
CARDS;
1 2 3 4
5 6 7 8
4 3 2 1
;
```

Résultat 2.7

Obs	X1	X2	X3
1	1	2	3
2	5	6	7
3	4	3	2

Dans les programmes 2.19 et 2.20, nous multiplions les instructions INPUT avec pour objectif de vous en préciser le fonctionnement. Lorsque l'instruction INPUT est lue, SAS charge un enregistrement et attribue des modalités aux variables listées ensuite. Après cette attribution, SAS retire l'enregistrement de l'INPUT BUFFER. À l'instruction INPUT suivante, SAS charge alors dans l'INPUT BUFFER un nouvel enregistrement. Vous comprenez maintenant pourquoi la table créée (voir résultat 2.6) ne contient qu'une seule observation : les trois instructions INPUT successives ont mobilisé trois enregistrements et n'ont construit qu'une observation.

L'option @ (placée en fin de ligne de l'instruction INPUT) demande à SAS de suspendre la lecture de l'enregistrement présent dans l'INPUT BUFFER et autorise une reprise de cette lecture ensuite. La première instruction INPUT donne donc à X1 sa modalité lue dans le premier enregistrement, et la seconde instruction INPUT continue la lecture de ce même enregistrement. Vous devez aussi comprendre, à travers le programme 2.20, qu'il n'y aura chargement d'un nouvel enregistrement que lorsque le programme sera intégralement exécuté sur le premier enregistrement. Nous avons introduit un @ inutile dans la troisième instruction INPUT, mais comme il s'agit de la dernière instruction du programme, les quatrièmes champs qui apparaissent dans les enregistrements ne sont jamais lus.

Reprenons le traitement des données présentées plus haut (voir résultat 2.4). Elles indiquent les températures moyennes journalières observées sur Orléans entre le 1er et le 31 juillet 2010. Pour chaque semaine, on dispose de sept températures. La première correspond au lundi qui ouvre la semaine et la dernière correspond donc au dimanche. En 2010, le 1er juillet était un jeudi – cela explique la présence des trois valeurs manquantes dans la première ligne.

Afin de construire la table présentée dans le résultat 2.5, vous pouvez utiliser le programme 2.21.

Programme 2.21

```
DATA temp_jul;
   LENGTH jour_sem $ 8;
   INPUT sem $ @;
   DO jour_sem='lundi','mardi','mercredi','jeudi','vendredi','samedi','dimanche';
      INPUT temp @;
      IF temp NE . THEN OUTPUT;
   END;
CARDS;
sem1 . . . 24 26.5 21 19.8
sem2 19.8 17.6 20.5 23.2 25.8 24.6 23.1
sem3 22.4 21.9 20.3 18.8 19.1 17.8 17.6
sem4 20.4 22.3 19 18 17.2 17.3 19
sem5 18.7 21.7 21.3 18.5 19.2 22.8 .
;
```

Ce programme nécessite l'emploi d'outils de programmation que nous traiterons au chapitre 3[1]. Si c'est votre première lecture, vous pouvez passer les explications qui suivent.

Lorsque SAS entame le traitement d'un enregistrement, il doit saisir dans une variable SEM le premier champ qu'il rencontre (soit « sem1 » pour le premier enregistrement) ; @ interrompt la lecture, mais cet enregistrement subsiste dans l'INPUT BUFFER. Nous introduisons ensuite une boucle qui « tournera » sept fois (puisqu'une semaine compte sept jours).

Au premier tour de la boucle, JOUR_SEM vaut « lundi ». Nous demandons *via* INPUT de reprendre la lecture de l'enregistrement contenu dans l'INPUT BUFFER et de lire le champ suivant (qui correspond normalement à une température moyenne observée un lundi), puis *via* @ d'interrompre une nouvelle fois la lecture des données. L'instruction suivante demande le versement du PDV dans la table à créer, uniquement si TEMP n'est pas valeur manquante. Le END clôt la boucle – on remonte alors en haut de la boucle : JOUR_SEM est maintenant égal à « mardi » et l'instruction INPUT est à nouveau exécutée pour lire la seconde température qui correspond bien à un mardi. Au septième tour de la boucle, les sept températures ont été lues, elles ont toutes été associées à la même valeur de SEM (puisqu'une fois qu'elle est saisie, notre programme ne la modifie pas), et chaque température se trouve associée à son jour.

On sort alors de la boucle ; le programme étant terminé pour le premier enregistrement, on remonte dans le programme et un nouvel enregistrement est chargé dans l'INPUT BUFFER. La figure 2.4 illustre le fonctionnement de ce programme pour le second enregistrement. La partie « boucle 0 » de la figure présente le contenu de l'INPUT BUFFER et du PDV juste avant que ne démarre la boucle DO.

1. Voir les sections 3.1.2 pour l'instruction LENGTH, 3.5.2 pour la boucle DO, 3.1.5 et 3.3.2 pour le IF / THEN OUTPUT.