Les programmes que vous avez écrits jusqu'à présent effectuaient toujours les mêmes actions et fournissaient toujours les mêmes résultats à chaque exécution. Le seul moyen de changer leur comportement était de modifier leur code.

Un programme peut cependant également recevoir des informations pour décider ce qu'il doit faire. C'est ce que nous verrons dans ce chapitre.

Dans l'invite de commandes ci-dessous, entrez `start` pour démarrer un programme qui vous demandera des informations. N'hésitez pas à le relancer plusieurs fois pour tester son fonctionnement, y compris avec des valeurs invalides pour l'année.

```
Tapez `start` puis appuyez sur Entrée pour démarrer.
>>
```

L'exécution du programme précédent est dite *interactive* car il y a un dialogue entre le programme et l'utilisateur (c'est-à-dire vous). La sortie du programme (en blanc) et la saisie des données d'entrée (en jaune) sont ainsi alternées. C'est la situation qui se présentera si vous exécutez un programme sur votre propre ordinateur.

Par contre, dans nos exercices, ce n'est pas vous qui allez donner ses entrées au programme mais le système d'évaluation automatisée. C'est-à-dire que les données seront déjà préparées, et que l'évaluateur s'en servira pour tester votre programme.

Ci-dessous, vous pouvez simuler une exécution automatisée sur un autre programme. À gauche, vous pouvez définir l'entrée du programme, puis démarrer la simulation en cliquant sur le bouton. La fenêtre sur la droite présente le déroulement tel qu'on pourrait se l'imaginer avec un utilisateur qui écrit les données.

Entrée :

```
Loïc
1986
```

Lancer la simulation

Sortie :

```
Loïc avait 14 ans en l'an 2000.
```

```
Tapez `start` puis appuyez sur Entrée pour démarrer.
>> start
 Loïc
 1986
 Loïc avait 14 ans en l'an 2000.
>>
```

Dans la suite, vous écrirez donc des programmes, qui seront testés automatiquement avec des données que nous avons préparées. À chaque test, l'évaluateur lira la sortie de votre programme pour vérifier qu'il répond bien au problème.

Votre programme devra bien entendu comporter des instructions pour récupérer les données, afin de calculer le résultat correspondant. Le pseudo-code ci-dessous correspond au second programme, uniquement pour les gens nés avant 2000 (nous ne verrons qu'au prochain chapitre comment gérer des cas distincts).

```
nom <- lireLigne()
année <- lireEntier()

Afficher nom, ", vous aviez ", 2000 - année, " ans en l'an 2000 !"
```

On lit donc les deux données et on les enregistre dans des variables, puis on affiche la phrase en les utilisant.

Notez que l'algorithme suivant est également valide et équivalent pour notre système d'évaluation.

```
nom <- lireLigne()
Afficher nom, ", vous aviez " (sans retour à la ligne)

année <- lireEntier()
Afficher 2000 - année, " ans en l'an 2000 !"
```

La sortie n'est lue qu'une fois l'exécution terminée. On peut ainsi afficher une partie de la sortie tout de suite ou attendre un peu, **c'est la même chose pour le système d'évaluation** : l'important est de lire les données dans l'ordre et d'écrire les bons résultats dans l'ordre.

Remarquez qu'on pourrait aussi ne pas utiliser de variable du tout (en mettant `lireLigne()` et `lireEntier()` directement dans les instructions d'affichage), mais cela rendrait le programme moins clair.