Les chaines de caractères

Les chaines de caractères en programmation

Une chaine de caractères est une suite de caractères traitée comme un tout. Il existe des instructions qui nous permettent de combiner, comparer, trier, manipuler les chaines de caractères de différentes façons.

Dans Scratch

Nous créons une variable « nom » et nous y enregistrons une chaine de caractères.

Les caractères de la chaine de caractères sont enregistrés en séquence.

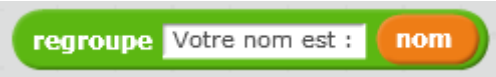
nom					
M	a	r	t	i	n

Dans la variable nom, nous avons 6 caractères auxquels nous pouvons accéder individuellement.

renvoie « i », cinquième lettre de Martin.

Scratch numérote les caractères à partir de 1.

renvoie 6, longueur de la chaine de caractères contenue dans la variable « nom ».



concatène (met bout à bout) les deux chaines de caractères « Votre nom est : » et le contenu de la variable « nom ».

Dans Algobox

Une variable qui doit contenir une chaine de caractères, doit être déclarée comme une variable de type « chaine ».

```
1   VARIABLES
2     nom EST_DU_TYPE CHAINE
```

Pour affecter une chaine de caractères à une variable de type « chaine », il faut mettre la chaine entre guillemets.

```
7     //Affecter une chaine de caractères à une variable de type chaine
8     nom PREND_LA_VALEUR "Martin"
```

La longueur d'une chaine de caractères est renvoyée par la fonction « length » appliquée à la variable contenant cette chaine.

```
3     longueur EST_DU_TYPE NOMBRE
```

```
10    //Longeur d'une chaine de caractères
11    longueur PREND_LA_VALEUR nom.length
```

La fonction « substr » appliquée à une variable contenant une chaine de caractères, renvoie un ou des caractères extraits de cette chaine.

La fonction « substr » possède deux paramètres : le numéro du premier caractère à extraire, le nombre de caractères à extraire.

Contrairement à Scratch, Algobox numérote les caractères à partir de 0.

Pour extraire le cinquième caractère de la variable « nom » :

```
4     extrait EST_DU_TYPE CHAINE

13    //Extraire une sous-chaine d'une chaine de cararactères
14    extrait PREND_LA_VALEUR nom.substr(4,1)
```

Pour concaténer deux chaines de caractères, il suffit de les lier avec le signe +.

```
5     phrase EST_DU_TYPE CHAINE

16    ////Concaténation de deux chaines de caractères
17    phrase PREND_LA_VALEUR "Votre nom est : " + nom
```

Travail sur les chaines de caractères

Exemple1 : Compter les lettres

L'utilisateur fournit une phrase et une lettre. Notre programme doit indiquer combien il y a d'occurrence de cette lettre dans la phrase.

L'algorithme Albobox : nous devons parcourir la chaine fournie par l'utilisateur, caractère par caractère, et comparer chacun des caractères avec la lettre fournie. Dès qu'un caractère de la chaine est identique au caractère fournie, nous ajoutons 1 à un compteur d'occurrence.

Nous avons besoin de 5 variables.

Une variable de type chaine dans laquelle nous allons enregistrer la phrase fournie par l'utilisateur : « **phrase** »

Une variable de type chaine dans laquelle nous allons enregistrer la lettre fournie par l'utilisateur : « **lettre** »

Une variable de type nombre qui mémorisera le nombre de fois où l'on rencontre la lettre lorsqu'on parcourt la chaine : « **occurrence** »

Une variable de type nombre qui mémorisera la longueur de la chaine fournie, de façon à savoir quand nous avons fini le parcours de la chaine : « **longueur** »

Une variable de type nombre qui nous servira d'index pour parcourir la chaine. Cette variable désignera le numéro du caractère de la chaine à traiter : « **i** »

```
1   VARIABLES
2     phrase EST_DU_TYPE CHAINE
3     lettre EST_DU_TYPE CHAINE
4     occurence EST_DU_TYPE NOMBRE
5     longueur EST_DU_TYPE NOMBRE
6     i EST_DU_TYPE NOMBRE
7   DEBUT_ALGORITHME
8     LIRE phrase
9     LIRE lettre
10    longueur PREND_LA_VALEUR phrase.length
11    POUR i ALLANT_DE 0 A longueur-1
12      DEBUT_POUR
13      SI (phrase.substr(i,1)==lettre.substr(0,1)) ALORS
14        DEBUT_SI
15        occurence PREND_LA_VALEUR occurence + 1
16        FIN_SI
17      FIN_POUR
```

- Dans la boucle « Pour », l'index i part de 0 (Albobox numérote les caractères d'une chaine à partir de 0) et s'arrêtera lorsqu'il aura dépassé la valeur longueur – 1.

  Si ma chaine a 6 caractères, avec Algobox, le dernier caractère aura le numéro 5.

- Dans l'instruction

```
SI (phrase.substr(i,1)==lettre.substr(0,1)) ALORS
```

  nous extrayons le ième caractère de la chaine « phrase » et nous le comparons au seul caractère de la chaine « lettre ».

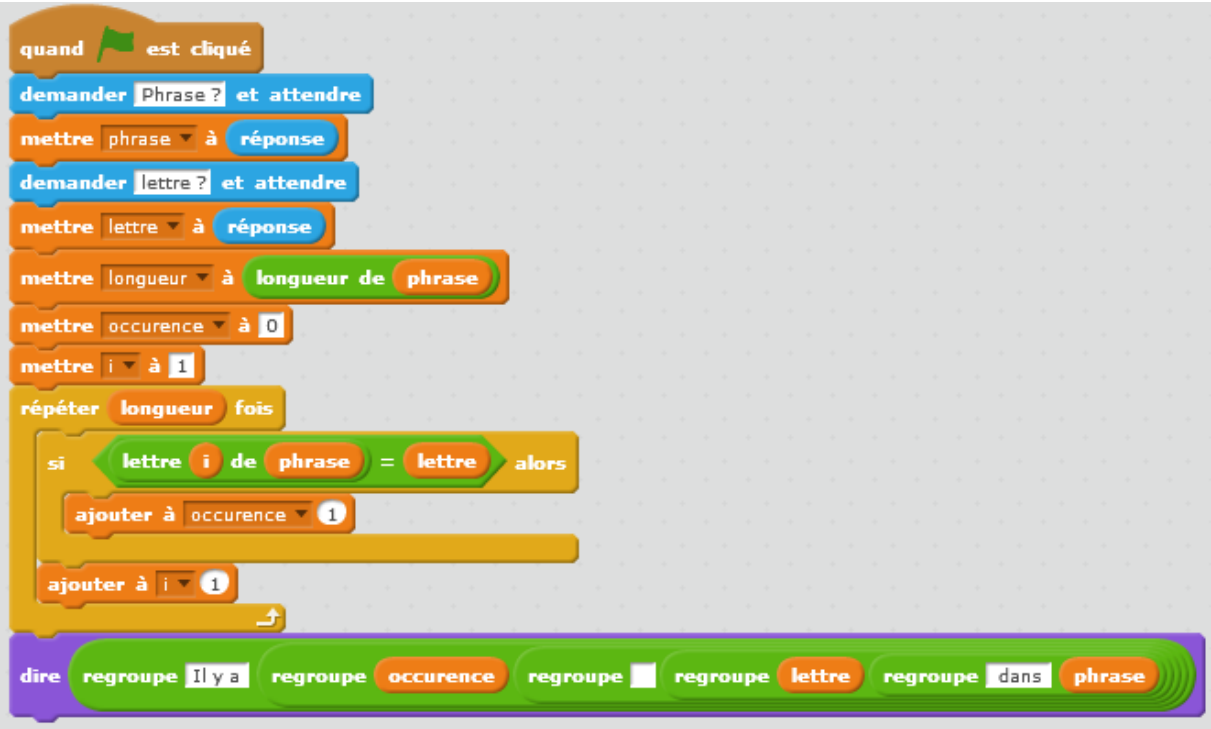
  S'il y a égalité, nous ajoutons 1 au compteur d'occurrence.

- Arrivé à la fin de la boucle « Pour », l'index i est incrémenté de 1 et l'exécution remonte sur le début de la boucle.
- Si i n'a pas dépassé la valeur longueur-1, on traite le caractère suivant dans la phrase.
- La fin de l'algorithme après la boucle « Pour » est l'affichage du résultat.

Avec Scratch :

Nous déclarons les mêmes variables et nous demandons à l'utilisateur de fournir la phrase et la lettre.

Nous initialisons les variables à leur valeur de départ : « **longueur** » à la longueur de « **phrase** », « **occurrence** » à 0 et « **i** » à 1, car Scratch numérote les caractères à partir de 1.



Scratch n'ayant pas de boucle « Pour » proprement dite, nous pouvons utiliser « répéter … fois », en utilisant la valeur contenue dans « longueur ».

Nous aurions aussi pu utiliser une structure « répéter jusqu'à … » en comparant la valeur de l'index « i » avec la valeur de « longueur ».

Nous exécutons la boucle jusqu'à ce que « i » devienne supérieur à « longueur »

Dans la boucle, nous comparons le ième caractère de « phrase » avec le caractère de « lettre ».

S'il y a égalité, nous augmentons « occurrence » de 1.

Avec Scratch, il ne faut pas oublier d'incrémenter « i » de 1.

Exemple 2 : Conjuguer un verbe du premier groupe.

L'utilisateur fournit un verbe du premier groupe et le programme affiche la conjugaison de ce verbe au présent de l'indicatif.

Algorithme Algobox :

Un verbe du premier groupe est composé d'un radical, suivi d'une terminaison « er ». Lorsqu'on conjugue, nous gardons le radical et nous le faisons suivre des terminaisons du présent de l'indicatif : e, es, e, ons, ez, ent.

Notre programme doit donc extraire le radical du verbe fourni par l'utilisateur, puis il doit le combiner avec les différents pronoms et terminaisons.

Nous avons besoins des variables suivantes :

Variable « **verbe** » de type chaine de caractères pour mémoriser le verbe à conjuguer.

Variable « **radical** » de type chaine de caractères pour mémoriser le radical du verbe à conjuguer.

Variable « **longueur** » de type nombre pour mémoriser la longueur du radical du verbe à conjuguer.

Variable « **verbeC** » de type chaine, dans laquelle nous mémoriserons la ligne de conjugaison à afficher.

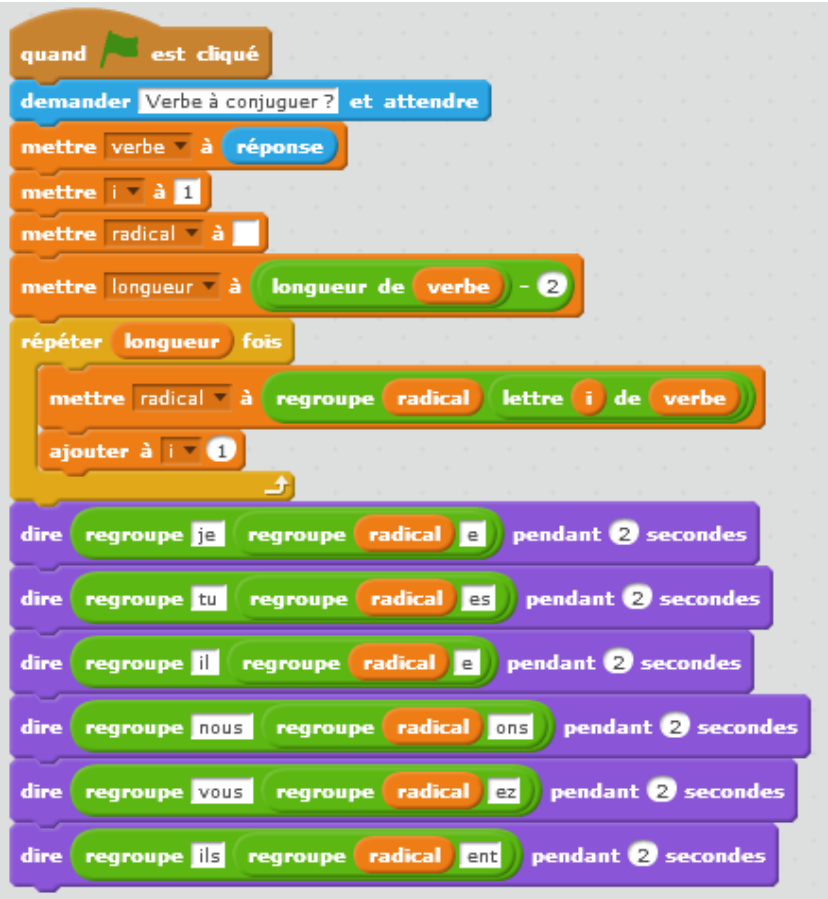
```
1   VARIABLES
2     verbe EST_DU_TYPE CHAINE
3     radical EST_DU_TYPE CHAINE
4     longueur EST_DU_TYPE NOMBRE
5     verbeC EST_DU_TYPE CHAINE
6   DEBUT_ALGORITHME
7     LIRE verbe
8     longueur PREND_LA_VALEUR verbe.length - 2
9     radical PREND_LA_VALEUR verbe.substr(0,longueur)
10
11    ////affichage de la conjugaison
12    verbeC PREND_LA_VALEUR "je " + radical +"e"
13    AFFICHER verbeC
14    verbeC PREND_LA_VALEUR "tu " + radical +"es"
15    AFFICHER verbeC
16    verbeC PREND_LA_VALEUR "il " + radical +"e"
17    AFFICHER verbeC
18    verbeC PREND_LA_VALEUR "nous " + radical +"ons"
19    AFFICHER verbeC
20    verbeC PREND_LA_VALEUR "vous " + radical +"ez"
21    AFFICHER verbeC
22    verbeC PREND_LA_VALEUR "ils " + radical +"ent"
23    AFFICHER verbeC
24  FIN_ALGORITHME
```

À la ligne 9 de l'algorithme nous extrayons le radical de la variable verbe en une seule instruction.

Avec Scratch

Nous nous passerons de la variable verbC, qui alourdit le programme.

Par contre, l'extraction du radical, à partir du verbe demande une nouvelle variable index « **i** ». En effet, nous sommes obligés de reconstituer ce radical, lettre par lettre, car les deux seules instructions à notre disposition sont : « lettre … de … » qui nous permet de récupérer une lettre de la chaine de caractères et « regroupe … … » qui permet de concaténer deux chaines de caractères.



Nous déclarons toutes les variables, comme locales au lutin. Seul ce lutin pourra les utiliser.

La variable « **i** » est initialisée à la valeur 1.

La variable « **radical** » est initialisée à une chaine vide.

La variable « **longueur** » prend une valeur égale à la longueur du verbe à conjuguer – les deux caractères de la terminaison.

Nous tournons dans la boucle « répéter … fois » un nombre de fois égal à la longueur du radical.

Dans cette boucle nous extrayons le ième caractère du verbe et nous le mettons au bout du radical et nous mettons le radical à jour.

Exemple : verbe sauter.

radical = «»

i = 1

longueur = 4.

Nous devons parcourir la boucle 4 fois.

<u>Première fois</u>

lettre 1 de verbe = s

radical ← s

i = 2

Deuxième fois

lettre 2 de verbe = a

radical ← sa (Surligné rose = le contenu de « radical » au moment de la concaténation)

i = 3

Troisième fois

lettre 3 de verbe = u

radical ← sau

i = 4

Quatrième fois

lettre 4 de verbe = t

radical ← saut

i = 5

Exercice 1 : Lire une chaine de caractères et l'écrire en sens inverse.

Écrire l'algorithme dans Algobox et le traduire dans Scratch

Exercice 2 : Lire une chaine de caractères et dire si cette chaine est un palindrome. Un palindrome est une chaine de caractère qui peut se lire dans les deux sens.

Quelque palindromes : elle, kayak, ici, radar, ressasser, sagas, rotor, ara, non, serres, sexes

Écrire l'algorithme dans Algobox et le traduire dans Scratch

Exercice 3 : Convertisseur Binaire → Décimal

Voir la fiche du projet, puis la vidéo consacrée à ce projet.

Il est important de bien lire et comprendre la fiche du projet, avant de regarder la vidéo.