Apprentissage de l'algorithmique : le cas Choiseul

Description du thème

Propriétés	Description
Intitulé long	Découverte de l'algorithmique et de la programmation à travers une situation de gestion simple : l'évaluation des besoins en personnel d'entretien dans un hôtel
Formation concernée	Classes de première Sciences et technologies de la gestion (STG)
Matière	Information et gestion
Présentation	A travers l'étude et la réalisation de programmes Javascript dans des pages web, et de leur algorithme, l'élève est amené à découvrir et à mettre en œuvre toutes les notions du programme concernant la logique algorithmique.
Notions	2.4- La logique algorithmique
Transversalité	Programme de mathématiques, notion de fonction et de division entière
Pré-requis	Structure d'une page HTML simple
Outils	Editeur de texte et navigateur
Mots-clés	Programme, algorithme, Javascript
Durée	Sept heures
Auteur(es)	Olivier Capuozzo et Christine Gaubert-Macon
Version	v 2
Date de publication	15 Avril 2005

Apprentissage de l'algorithmique : le cas Choiseul

L'hôtel CHOISEUL dispose de 113 chambres dont l'entretien doit être assuré chaque jour. L'hôtelier, monsieur Viard, fait appel à une société spécialisée qui lui délègue chaque jour du personnel pour l'entretien des chambres. Monsieur Viard évalue ses besoins en heures d'entretien en fonction du nombre de chambres réservées. Il établit ses prévisions pour le jour suivant, et contacte chaque matin la société spécialisée pour demander une équipe pour le lendemain matin.

Afin d'évaluer ces besoins en personnel d'entretien, l'hôtelier dispose d'un programme réalisé à l'aide d'une page web accessible via un navigateur[1].

I – Étude de la première version du programme

Dans la première version du programme, l'hôtelier saisit le nombre de réservations et le programme lui affiche le nombre de vacations en personnel d'entretien nécessaires pour le jour suivant.
Ce calcul s'effectue en prenant en compte les éléments suivants :
- une personne d'entretien est embauchée pour une vacation de deux heures dans l'hôtel,
- chaque chambre nécessite un quart d'heure d'entretien.

Exemple :
L'hôtelier a 60 réservations pour le 15 juin 2005. Le 14 juin 2005 au matin, il lance son programme qui effectue le calcul suivant
*60 chambres * 15 minutes = 900 minutes de travail pour l'entretien des chambres.*

Une vacation d'entretien des chambres est de 120 minutes (2 heures). Donc, dans notre cas, le nombre de vacations se détermine ainsi :
900 minutes au total / 120 minutes pour une vacation soit 7.5 vacations sont nécessaires.
Le programme indiquera donc à l'hôtelier qu'il doit demander 7.5 vacations. Les besoins de l'hôtelier seront donc de 8 personnes d'entretien pour le 15 juin au matin.

I - 1) Analyse de l'algorithme

Le programme rend un service à l'utilisateur : l'aider à déterminer le nombre de vacations dont il a besoin le lendemain pour l'entretien de son hôtel.
Ce programme définit un traitement, en utilisant des données en entrée pour produire des résultats et en respectant un algorithme.

Dans le cas présent, la donnée en entrée est le nombre de réservations, et le résultat le nombre de vacations. Pour exploiter le nombre de réservations et le nombre de vacations dans le programme, on utilise des variables dont il faut définir le type. Pour ces deux variables, les valeurs sont des entiers, donc le type des variables est « entier ».

La grille d'analyse ci-dessous permet de définir l'algorithme.

Donnée à fournir en entrée de traitement

Donnée	*Nature et type*	*Valeur*	*Commentaire*
Nombre de réservations	Variable, Entier	Pas de valeur initiale	Renseignée par une valeur saisie par l'utilisateur

Traitement

Objectif	
Communiquer à l'utilisateur le nombre de vacations de personnel d'entretien nécessaire pour un nombre de chambres réservé.	
Action	***But***

[1] Le navigateur interprète la page, c'est-à-dire qu'il affiche à l'écran le résultat de son interprétation du contenu de la page.

Objectif **Communiquer à l'utilisateur le nombre de vacations de personnel d'entretien nécessaire pour un nombre de chambres réservé.**	
1. Récupérer la donnée saisie	Récupérer la valeur saisie par l'utilisateur désignant le nombre de réservations.
2. Calculer	Calculer le nombre de vacations en fonction du nombre de réservations, sachant qu'une chambre s'entretient en 15 minutes, et qu'une personne est obligatoirement embauchée pour une vacation de deux heures (120 minutes).
3. Afficher	Afficher le nombre de vacations calculé.

Résultat (donnée obtenue comme résultat du traitement)

Description	*Nature et type*	*Commentaire*
Nombre de vacations	Variable, réel	

I - 2) Étude de la page web

Fichier à utiliser : hotel1.html

Le programme est déclenché en demandant l'interprétation du fichier *hotel1.html* par un navigateur. Cette page web utilise deux langages : le langage HTML qui est utilisé pour construire l'apparence de la page et le langage JavaScript qui est utilisé pour réaliser l'algorithme que nous avons présenté ci-dessus.

I - 2.1- Rendu à l'écran de la page

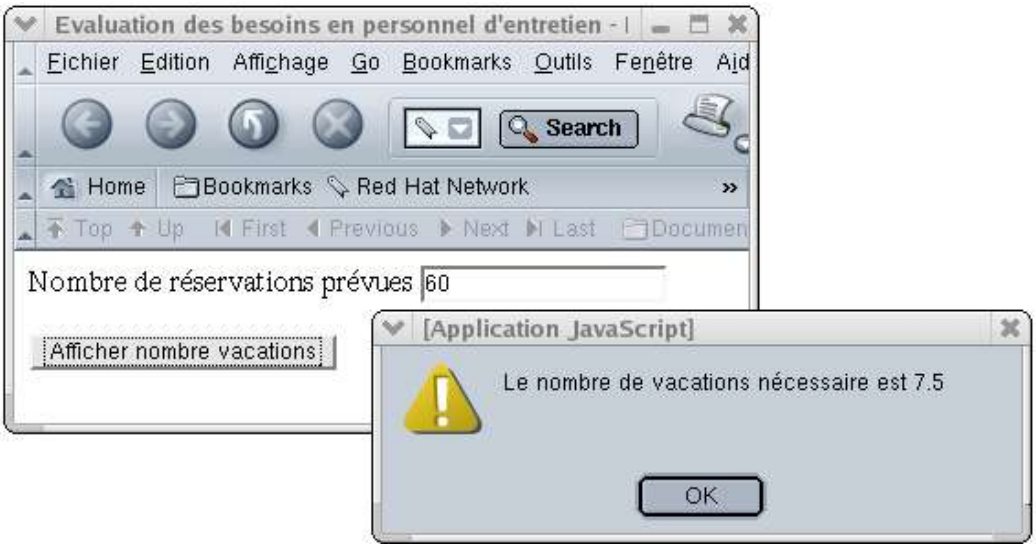
Fichier à utiliser : hotel1.html

Voici ce que l'utilisateur verra lorsqu'il exécute la page (en double-cliquant sur `hotel1.html` par exemple) :

Au lancement de l'application

Lorsqu'il aura saisi un nombre de réservations

Lorsqu'il aura cliqué sur le bouton « Afficher nombre vacations »



I - 2.2- Analyse du programme inclus dans la page web

Fichier à utiliser : hotel1.html

Voici ce qui a été programmé pour réaliser l'algorithme[2] :

Ce qui est en bleu correspond aux instructions en langage HTML pour la présentation de la page

En rouge la fonction en langage JavaScript qui réalise l'algorithme. Dans une page HTML, les fonctions en JavaScript sont écrites entre deux balises <script> et </script>.

```
<html>
<title> Evaluation des besoins en
<head>
<script>
function traitVac()
{
  var nbVac, nbResaS;

  nbResaS = document.formulaireResa.nbResa.value;

  nbVac = ((nbResaS * 15))/120;

  alert('le nombre de vacations nécessaire est  '+nbVac);
}
</script>
</head>
<body>

<form name="formulaireResa">
Nombre de réservations prévues <input type="text" name=nbResa><BR><BR>
<input type="button" value="Afficher nombre vacations" onclick="traitVac()"/>
</form>

</body>
</html>
```

Ici le début du formulaire qui est la partie de la page sur laquelle l'utilisateur va agir. Il contient la zone de saisie du nombre de réservations et un bouton pour lancer le programme de calcul.

nbResa est la zone de la page dans laquelle l'utilisateur saisit le nombre de réservations

Quand on clique sur le bouton, on lance l'exécution de la fonction traitVac

On voit que l'algorithme est réalisé grâce à des instructions JavaScript dans une fonction. Une instruction est un ordre élémentaire que comprend l'interpréteur du langage (le navigateur). Une fonction permet de représenter un traitement dans ce langage, et ce traitement sera exécuté à chaque appel de la fonction.

Cette fonction JavaScript est la partie du programme qui réalise l'algorithme d'évaluation du nombre de vacations de personnel d'entretien, détaillons ses instructions :

[2] Pour visualiser les instructions d'une page web, il faut aller dans le menu du navigateur (par exemple dans Internet Explorer menu « Affichage » et option « Source ») ou charger la page dans un éditeur de texte.

Programme JavaScript	Rôle des différentes instructions
`function traitVac()` `{`	*function* est un mot du langage qui permet de marquer le début de la fonction. Il est suivi du nom de la fonction puis d'un jeu de parenthèses. L'accolade ouvrante marque le début du programme.
`var nbVac, nbResaS;`	*var* est un mot du langage qui permet de définir (on dit *déclarer*) une donnée manipulée par le programme ou variable. Certaines variables ont une valeur au début du programme, d'autres pas.
`nbResaS = document.formulaireResa.nbResa.value;`	Le symbole « = » signifie que la variable prend la valeur indiquée à droite du signe « = » Il s'agit d'une instruction d'affectation (on affecte une valeur à une variable).
`nbVac=(nbResaS*15)/120;`	Là encore, c'est une instruction d'affectation, ici la variable nbVac va recevoir le résultat d'un calcul.
`alert('le nombre de vacations nécessaire est  '+nbVac);`	Cette instruction permet d'afficher un message à l'écran.
`}`	L'accolade fermante marque la fin du programme.

En JavaScript, chaque instruction se termine par un « ; ».

Détaillons l'instruction d'affectation
`nbResaS = document.formulaireResa.nbResa.value`
Dans cette instruction, `nbResaS` est le nom d'une variable déclarée et qui va contenir le nombre de réservations ; `document.formulaireResa.nbResa.value` désigne la valeur (*value*) détenue par le champ nommé `nbResa`, du formulaire nommé `formulaireResa`, là où l'utilisateur a saisi le nombre de réservations prévu.

Détaillons l'instruction d'affectation
`nbVac = ((nbResaS * 15))/120`
Dans cette instruction, on calcule le nombre de vacations et on affecte la valeur obtenue à la variable `nbVac`.

Détaillons l'instruction d'affichage
`alert('le nombre de vacations nécessaire est  '+nbVac);`
`alert` est une fonction qui permet d'afficher une boîte de dialogue à l'écran. Cette fonction est fournie par le langage JavaScript.
Entre parenthèses figure le message à afficher, il y a d'abord un bout de texte puis le contenu de la variable `nbVac`. Ici le signe + signifie qu'on met bout à bout (on concatène) la chaîne de caractères 'le nombre de vacations nécessaire est ' et le contenu de la variable `nbVac`.

Ce qui donne

De l'analyse de ce traitement on peut déduire la logique algorithmique suivante :

1. Représentation de données : déclaration de `nbResaS, nbVac`
2. Récupération de la saisie d'une valeur par l'utilisateur et affectation dans la variable `nbResaS`
3. Calcul du résultat à obtenir dans la variable `nbVac`
4. Affichage du résultat : valeur de la variable `nbVac`.

I - 2.3- Travail à faire

Fichier à utiliser : hotel1.html

La grille présentée au paragraphe I – 1 a permis de définir l'algorithme. Compléter cette grille pour montrer les éléments de la fonction JavaScript qui prennent en charge l'algorithme :

Donnée d'entrée				
Donnée	*Nature et type*	*Valeur*	*Commentaire*	*Nom dans le programme*
Nombre de réservations	Variable, Entier	Pas de valeur initiale	Variable renseignée par une valeur saisie par l'utilisateur	
Traitement				
Objectif Communiquer à l'utilisateur le nombre de vacations de personnel d'entretien nécessaire pour un nombre de chambres réservé.				
Action	*But*	*Instruction correspondante dans le programme*		
1. Récupérer la donnée saisie	Récupérer la valeur saisie par l'utilisateur désignant le nombre de réservations			
2. Calculer	Calculer le nombre de vacations en fonction du nombre de réservations, sachant qu'une chambre s'entretient en 15 minutes, et qu'une personne est obligatoirement embauchée pour une vacation de deux heures (120 minutes).			
3. Afficher	Afficher le nombre de vacations			
Résultat (donnée obtenue comme résultat du traitement)				
Description	*Nature et type*	*Nom dans le programme*		
Nombre de vacations	Variable, réel			

I – 3- En résumé

Nous avons présenté un programme pouvant s'exécuter dans un navigateur Web.

Ce programme est une page web qui est stockée dans un fichier suffixé `.html`. L'interaction avec l'utilisateur est programmée en langage HTML tandis que l'algorithme est réalisé grâce à une fonction définie par des instructions du langage JavaScript. Cette fonction se nomme `traitVac()`, elle est écrite au début la page web. Elle est appelée depuis un formulaire HTML. Cette fonction ne renvoie pas de valeur et n'attend pas d'argument.

Les données manipulées sont stockées dans des variables qui soit sont en entrée (nombre de réservations) soit permettent de stocker le résultat (nombre de vacations). Ces variables ont un type correspondant à la nature de la donnée stockée. Ici les deux variables sont de type « entier ». En JavaScript le type d'une variable ne se déclare pas explicitement, mais c'est au moment où la variable reçoit une valeur qu'elle devient typée.

L'opérateur d'affectation (=) permet de fournir une valeur (argument à droite) à une variable (argument à gauche).

Le calcul du nombre de vacations a été réalisé à l'aide des opérateurs * (multiplication) et / (division).

Le résultat est transmis à l'utilisateur grâce à une fonction fournie par le langage JavaScript (`alert`) qui provoque l'apparition d'une boîte de dialogue contenant un message personnalisé. Cette fonction attend un argument : le message à afficher.

Les instructions du programme s'exécutent en séquence (s'enchaînent) : le navigateur les interprète les unes après les autres pour afficher la page à l'utilisateur.

L'annexe 1 reprend les définitions des termes manipulés dans cette partie.

II – Évolution du programme

II – 1- Amélioration du service rendu à l'utilisateur

Il serait bon que l'application affiche un nombre entier de vacations.
C'est-à-dire que si le nombre de vacations est un réel, on approche à la valeur entière immédiatement supérieure la plus proche.

Exemple :
Dans l'exemple cité précédemment où l'utilisateur saisit 60 pour le nombre de réservations, on obtient 7.5 vacations, et l'hôtelier décide de demander la délégation de 8 personnes d'entretien.
Si on avait obtenu 7.4 vacations, l'hôtelier aurait également demandé la délégation de 8 personnes.
Si on avait obtenu 7.0 vacations, l'hôtelier aurait demandé la délégation de 7 personnes.

II – 1.1- Analyse de l'algorithme

Cette amélioration de l'application a une influence sur la partie traitement de l'algorithme. La grille d'analyse de l'algorithme devient (en rouge les modifications) :

Donnée à fournir en entrée de traitement

Donnée	*Nature et type*	*Valeur*	*Commentaire*
Nombre de réservations	Variable, Entier	Pas de valeur initiale	Renseignée par une valeur saisie par l'utilisateur

Traitement

Objectif **Communiquer à l'utilisateur le nombre entier de vacations de personnel d'entretien nécessaire pour un nombre de chambres réservé.**	
Étape	***But***
1. Récupérer la donnée saisie	Récupérer la valeur saisie par l'utilisateur désignant le nombre de réservations.
2. Calculer	Calculer la valeur entière immédiatement supérieure ou égale au nombre de vacations en fonction du nombre de réservations, sachant qu'une chambre s'entretient en 15 minutes, et qu'une personne est obligatoirement embauchée pour une vacation de deux heures (120 minutes).
3. Afficher	Afficher le nombre de vacations calculé.

Résultat (donnée obtenue comme résultat du traitement)

Description	*Nature et type*	*Commentaire*
Nombre de vacations	Variable, entier	

II – 1.2- Modification de la page web

JavaScript propose, dans sa bibliothèque[3] `Math`, une fonction nommée `ceil` qui correspond à notre besoin. La fonction `Math.ceil(x)` rend le plus petit entier supérieur ou égal à la valeur `x` donnée en argument. Exemple : `Math.ceil(2.01)` rend 3, `Math.ceil(1.99)` rend 2.

Dans notre cas, nous passerons en argument à la fonction `Math.ceil` le nombre de vacations (un réel).

II – 1.3- Travail à faire

Fichier à utiliser : hotel1.html

[3] Une bibliothèque est un ensemble de fonctions prédéfinies pour un langage et pouvant être utilisées par les programmeurs.

II - 1.3.1 Qu'est-ce qui différencie fondamentalement les fonctions `traitVac` et `Math.ceil` ?

II – 1.3.2 Copier le fichier `hotel1.html` et le renommer `hotel11.html`. Effectuer la modification sur le fichier `hotel11.html`. Tester (voir les consignes en annexe 2).

II - 2 - Contrôle de la donnée saisie par l'utilisateur

L'évaluation des vacations n'est possible que si l'hôtelier saisit un nombre de réservations fiable, c'est-à-dire une donnée numérique supérieure à zéro et inférieure ou égale au nombre total de chambres de l'hôtel.
Dans un premier temps, nous nous concentrons sur la qualité numérique ou non de la donnée fournie par l'utilisateur.

II – 2.1- Analyse de l'algorithme

Cette amélioration de l'application a une influence sur la partie traitement de l'algorithme. La grille d'analyse de l'algorithme devient (en rouge les modifications) :

Donnée à fournir en entrée de traitement

Donnée	*Nature et type*	*Valeur*	*Commentaire*
Nombre de réservations	Variable, Entier	Pas de valeur initiale	Renseignée par une valeur saisie par l'utilisateur

Traitement

Objectif **Communiquer à l'utilisateur le nombre entier de vacations de personnel d'entretien nécessaire pour un nombre de chambres réservé.**	
Action	***But***
1. Récupérer la donnée saisie	Récupérer la valeur saisie par l'utilisateur désignant le nombre de réservations.
2. Contrôler la donnée saisie	S'assurer d'un déroulement cohérent du programme : Si la donnée n'est pas numérique alors envoyer un message à l'utilisateur sinon continuer le déroulement de l'algorithme.
3. Calculer	Calculer la valeur entière immédiatement supérieure ou égale au nombre de vacations en fonction du nombre de réservations, sachant qu'une chambre s'entretient en 15 minutes, et qu'une personne est obligatoirement embauchée pour une vacation de deux heures.
4. Afficher	Afficher le nombre de vacations calculé ou un message d'erreur.

Résultat (donnée obtenue comme résultat du traitement)

Description	*Nature et type*	*Commentaire*
Nombre de vacations ou message d'erreur	Variable, entier ou chaîne_de_caractères	un message d'erreur est affiché si l'utilisateur n'a pas saisi un nombre.

L'action n° 2 de l'algorithme prend appui sur une alternative, c'est-à-dire une instruction qui permet de réaliser un groupe d'instructions si une condition est réalisée et un autre groupe d'instructions si la condition n'est pas réalisée.

II – 2.2- Modification de la page web

Il nous faut utiliser une fonction qui permette de vérifier si la valeur saisie est bien numérique (est un nombre).
En JavaScript, c'est la fonction `isNaN(x)` qui donne le résultat vrai (`true`) si x n'est pas un nombre, et faux (`false`) si c'est un nombre (isNaN est une abréviation de *is not a number*).

Exemple :
Supposons qu'une variable nommée `maVar` contienne « bonjour », alors `isNaN(maVar)` vaudra `true`.
Si `maVar` contient le nombre 56 alors `isNaN(maVar)` vaudra `false`.

Par ailleurs, il faut tester la valeur rendue par la fonction `isNaN`(x) et mettre en œuvre une alternative.
En Javascript (comme dans bon nombre de langages), l'instruction correspondante est :

```
if (condition_est_vraie)
{
   // On fait le traitement A adapté lorsque la condition est
      // vérifiée
}
   else
{
   //  On fait le traitement B adapté lorsque la
   // condition n'est pas vérifiée
}
```

Lors de l'exécution de cette instruction, le traitement réellement exécuté sera soit le A soit le B (mais jamais les deux et jamais aucun).
Lorsqu'on écrit la séquence d'instructions à l'intérieur de l'instruction `if`, on indente[4] les blocs d'instructions.

II – 2.3- Travail à faire

Fichiers à utiliser : hotel11.html et hotel2.html

II.2.3.1- En examinant la définition de `traitVac` du fichier `hotel11.html`, quelle variable mériterait que l'on vérifie sa qualité numérique ?

II.2.3.2- Ouvrir le fichier `hotel2.html`. Saisir une valeur non numérique. Que se passe-t-il ? Quelle est la partie du programme qui permet d'assurer le contrôle de la valeur saisie ?

II.2.3.3- Recopier le tableau d'analyse de l'algorithme ci-dessus et compléter par une colonne présentant les instructions correspondantes aux actions décrites.

II.2.3.4- Il est nécessaire de contrôler que la valeur numérique saisie par l'utilisateur correspond à un nombre de chambres.
a) Quel intervalle de valeurs peut prendre la valeur numérique saisie par l'utilisateur ?
b) Réaliser la grille d'analyse de l'algorithme qui permet de prendre en charge le contrôle de la validité de la valeur saisie par l'utilisateur.
c) Sachant que l'opérateur OU en JavaScript s'écrit « || », expliquer l'instruction suivante :

```
if ((nbResaS <= 0)||(nbResaS > 123))
{
  alert('Attention il faut saisir un nombre positif et inférieur à 123');
}
else
{
  nbVac=(nbResaS*15)/120;

  alert('Le nombre de vacations nécessaire est  '+nbVac)
}
```

d) Copier le fichier `hotel2.html` et le renommer `hotel21.html`. Effectuer la modification sur le fichier `hotel21.html`. Tester (voir les consignes en annexe 2).

[4] Il s'agit d'écrire les lignes avec un retrait de paragraphe.

e) Quel autre contrôle pourrait-on effectuer sur la valeur saisie ?

II - 3 – Programmation du calcul du nombre de vacations

L'instruction
```
nbVac = Math.ceil((nbResaS * 15)/120);
```
réalise la division `(nbResaS * 15)/120` puis affecte à la variable la valeur entière immédiatement supérieure ou égale au quotient de la division.
On a vu que si la variable nbResaS contient 60 alors à l'issue de cette instruction, la variable nbVac contient 8.

On souhaiterait créer une fonction qui réalise le même traitement sans utiliser la division mais de la façon suivante :
- calculer le quotient et le reste de la division entière de deux nombres en utilisant des **soustractions successives** ;
- renvoyer le quotient si le reste est nul, sinon renvoyer le quotient augmenté de 1.

Cette fonction est nommée arrondiSupDiv. Elle admet en arguments deux nombres.

Exemple : arrondiSupDiv (900, 120) rend 8.

De manière plus universelle, cette fonction peut s'écrire *arrondiSupDiv (x,y)*

II – 3.1- Analyse de l'algorithme

Comment cette fonction réalise-t-elle le traitement ? Elle effectue des soustractions successives tant que la différence est supérieure ou égale au diviseur. Le nombre de soustractions réalisées permet d'obtenir le quotient entier de la division des deux arguments (respectivement dividende et diviseur).

Terme 1	Terme 2 (diviseur)	Différence	Nombre de soustractions
(dividende) 900	120	780	1
780	120	660	2
660	120	540	3
540	120	420	4
420	120	300	5
300	120	180	6
180	120	**60**	7

On arrête la série de soustractions car la différence est devenue inférieure au diviseur.

Il a fallu réaliser 7 soustractions pour obtenir le quotient de la division de 900 par 120 est 7, et le reste est 60. Ce reste étant supérieur à 0, la valeur renvoyée par la fonction `arrondiSupDiv` sera 8.

Pour cette fonction, la grille d'analyse de l'algorithme est la suivante :

Données à fournir en entrée de traitement

Donnée	*Nature et type*	*Valeur*	*Commentaire*
x	Argument, Entier	Passée en argument	
y	Argument, Entier	Passée en argument	

Traitement

Objectif **Rendre la valeur entière immédiatement supérieure ou égale au résultat de la division de deux nombres.**	
Action	***But***
1. Initialiser le compteur de soustractions	Le compteur est nul au début du traitement.
2. Calculer le nombre de soustractions	Tant que la différence (x - y) est supérieure ou égale à y, soustraire à x la valeur de y et augmenter le compteur de 1.
3. Calculer la valeur qui sera retournée par la fonction	Si la différence est nulle alors la valeur est celle du compteur sinon celle du compteur augmenté de 1.

Résultat (donnée obtenue comme résultat du traitement)

Description	***Nature et type***	***Commentaire***
L'arrondi au supérieur de la division entière de x par y	Variable, entier	

L'action n° 3 de l'algorithme s'appuie sur une répétition, c'est-à-dire d'une instruction qui permet de répéter un groupe d'instructions selon une condition. On peut la représenter de la façon suivante :

En JavaScript, l'instruction

```
while (condition_est_vraie)
{
  // on réalise le traitement correspondant à la séquence
d'instructions
}
```

permet de commander l'exécution d'une séquence d'instructions tant que la condition est réalisée. Lorsqu'on écrit la séquence d'instructions à l'intérieur de l'instruction répétitive `while`, on indente[5] le bloc d'instructions.

[5] Il s'agit d'écrire les lignes avec un retrait de paragraphe.

II – 3.2- Étude de la page web

Fichier à utiliser : hotel22.html

II.3.2.1- Lancer l'exécution de la page et saisir un nombre de réservations. Est-ce que le résultat obtenu est différent de celui obtenu lors de l'exécution de la page `hotel11.html` ?

II.3.2.2- Dans le code de la page `hotel22.html`, repérer la définition de la fonction `arrondiSupDiv(x,y)` Recopier le tableau d'analyse de l'algorithme ci-dessus et compléter par une colonne présentant les instructions correspondantes aux actions décrites pour cette fonction.

II.3.2.3- Dans la page `hotel22.html`, repérer l'instruction où la fonction `arrondiSupDiv(x,y)` est appelée.

II - 4 – En résumé

Le programme a évolué pour rendre un meilleur service à l'utilisateur. Pour programmer cette évolution deux solutions ont été proposées :

- Un première version qui utilise une fonction prédéfinie par le langage provenant de la bibliothèque `Math` : `Math.ceil(x)` ;
- Une deuxième version qui a nécessité la mise au point d'une fonction nommée `arrondiSupDiv (x,y)`, et son appel dans la fonction `TraitVac()` ;
- Ces deux versions sont composées d'instructions permettant commander par des conditions la réalisation de groupes d'instructions
    - L'alternative qui permet de choisir en fonction d'une condition la réalisation d'un groupe d'instructions ou d'un autre ;
    - La répétitive qui permet de répéter un groupe d'instructions tant qu'une condition est vérifiée.

III – Utilisation du programme dans un autre hôtel

L'hôtelier est propriétaire d'un autre hôtel dans lequel il souhaite utiliser le même programme. Cet établissement de 89 chambres est d'une catégorie plus luxueuse et le temps d'entretien d'une chambre est évalué à 20 minutes. Le personnel d'entretien est toujours employé pour une vacation de deux heures.

III – 1- Analyse de l'algorithme

Fichier à utiliser : hotel21.html

Travail à faire

III . 1.1- Quelles lignes du programme doivent être modifiées pour que celui-ci puisse être utilisé dans le deuxième hôtel ?

Le nombre de chambres et la durée d'entretien d'une chambre sont des données qui varient en fonction de l'hôtel. On va définir leur valeur dans des constantes, c'est-à-dire des données qui sont fixées au début de l'algorithme et qui ne changent pas durant son déroulement. Ainsi lorsqu'on voudra utiliser ce programme dans un nouvel hôtel, il n'y aura que les valeurs des constantes à modifier et non à rechercher dans le programme les instructions à modifier.

La grille d'analyse de l'algorithme devient (en rouge les modifications) :

Données à fournir en entrée de traitement

Donnée	*Nature et type*	*Valeur*	*Commentaire*
Nombre de réservations	Variable, Entier	Pas de valeur initiale	Renseignée par une valeur saisie par l'utilisateur
Nombre de chambres	Constante, Entier	89	
Temps d'entretien d'une chambre	Constante, Entier	20	Exprimé en minutes

Traitement

Objectif **Communiquer à l'utilisateur le nombre entier de vacations de personnel d'entretien nécessaire pour un nombre de chambres réservé.**	
Étape	*But*
1. Récupérer la donnée saisie	Récupérer la valeur saisie par l'utilisateur désignant le nombre de réservations.
2. Contrôler la donnée saisie	Si la donnée n'est pas numérique alors envoyer un message à l'utilisateur. Si la donnée est numérique mais qu'elle n'est pas comprise entre 1 et le nombre de chambres, alors envoyer un message d'erreur à l'utilisateur. Dans les autres cas, continuer le déroulement de l'algorithme.
3. Calculer	Calculer la valeur entière immédiatement supérieure ou égale au nombre de vacations en fonction du nombre de réservations, du temps d'entretien d'une chambre, et qu'une personne est obligatoirement embauchée pour une vacation de deux heures.
4. Afficher	Afficher le nombre de vacations calculé ou un message d'erreur.

Résultat

Description	*Nature et type*	*Commentaire*
Nombre de vacations ou message d'erreur	Variable, entier ou chaîne_de_caractères	un message d'erreur est affiché si l'utilisateur n'a pas saisi une valeur correcte.

Fichier à utiliser : hotel3.html

Travail à faire

III. 1.2- Recopier le tableau d'analyse de l'algorithme ci-dessus et compléter par une colonne présentant les instructions correspondantes dans la page `hotel3.html`.

III. 1.3- Quelle autre valeur aurait mérité d'être déclarée comme constante ?

III. 1.4- Copier le fichier `hotel3.html` et le renommer `hotel31.html`. Effectuer la modification sur le fichier `hotel31.html` pour déclarer et utiliser cette constante. Tester (voir les consignes en annexe 2).

III – 2- En résumé

Lorsqu'une donnée utilisée dans un programme a une valeur fixe, on utilise une constante pour la représenter. Les constantes sont le plus souvent déclarées en début de programme. En JavaScript, une constante se déclare comme une variable.

IV – Test de connaissances

Travail à faire

Répondre aux questions suivantes en s'appuyant notamment sur les différentes versions du programme d'évaluation des besoins en personnel.

Quelle différence y-a-t-il entre un algorithme et un programme ?	
Que signifie « déclarer une variable » ?	
Une variable a-t-elle toujours une valeur définie lors de sa déclaration ?	
Quelle différence y-a-t-il entre une variable et une constante ?	
Qu'est-ce qu'une instruction ?	
Qu'est-ce qu'une page web ?	
Que signifie la phrase « le navigateur interprète la page. » ?	
Qu'est-ce qu'une fonction ?	
Quelle est la différence entre la définition d'une fonction et l'appel d'une fonction ?	
Qu'est-ce qu'un argument d'une fonction ?	
Donner un exemple de fonction fournie par le langage JavaScript.	
Donner un exemple de fonction écrite par le programmeur.	
Donner un exemple de séquence d'instructions.	
Donner un exemple d'alternative.	
Donner un exemple de répétitive.	

Annexe 1 : Vocabulaire

Programme informatique
« Ensemble organisé d'instructions, exécuté par un ordinateur, servant un à plusieurs objectifs utilisateur. »
Par exemple, le programme a pour utilisateur l'hôtelier et sert l'objectif utilisateur : « Déterminer le nombre de vacations de personnel d'entretien pour un nombre donné de réservations».

Un programme est un mécanisme de **service**. Il apporte une **plus-value** à son **utilisateur**.

L'utilisateur demande l'exécution d'un service, le service réalise un traitement qui produit un certain résultat (la plus-value).

En général la demande de service est accompagné de **données d'entrée**.
La mise en oeuvre du service est communément appelée **traitement**, et la plus-value **résultat**.

Nous avons alors le schéma ETR suivant :

Entrée -> Traitement -> Résultat

Ce schéma est observable à différents niveaux. Exemples :

Le programme de l'hôtelier :
E : le nombre de réservations.
T : calcul de détermination du nombre de personnes d'entretien.
R : le nombre de personnes nécessaire à l'entretien des chambres.

Le programme qui calcule les impôts d'un contribuable
E : données déduites de la déclaration du contribuable.
T : calcul des impôts.
R : le montant des impôts.

Algorithme
La définition organisée des données d'entrée, du traitement et du résultat constitue l'**algorithme**.

Le programme est une traduction de l'algorithme dans un langage de programmation.
Dans un algorithme, le traitement se découpe en actions, qui correspondent à des opérations élémentaires et qui se traduiront par des instructions dans les programmes.

Fonction
C'est un sous-programme, c'est-à-dire un ensemble d'instructions qui forme une unité fonctionnelle (rend un service) et qui sert un des objectifs du programme.
On distingue l'**utilisation** d'une fonction de sa **définition** (ses instructions correspondant à son algorithme).
Le programme qui utilise une fonction n'est pas tenu de connaître la façon dont elle opère (son algorithme), seule la nature du service rendu suffit.
Il existe donc une relation de confiance entre les concepteurs de fonctions (fournisseurs) et les programmeurs utilisateurs (clients). Exemple : j'estime que je peux utiliser `Math.ceil` de JavaScript en toute confiance (à condition de respecter les pré-conditions à son utilisation définies par son fournisseur, c'est-à-dire de fournir un nombre).

Si on reprend le schéma ETR, dans le cas d'une fonction, on peut dire :

E : argument(s) de la fonction. Ils sont passés à la fonction dans la zone délimitée par des parenthèses, juste après le nom de la fonction.
T : des opérations (cachées du programme utilisateur).
R : le plus souvent la valeur retournée par la fonction.

Exemple en JavaScript:
```
x = Math.sqrt(4)
// la valeur de x est 2
```

Type

Toute donnée manipulée dans un algorithme, et un programme, obéit à un type, ensemble de valeurs possibles pour cette donnée. Les types de base proposés généralement sont

- entier : pour représenter les nombres entiers positifs, nul et négatifs ;
- réel : pour représenter les nombres fractionnaires ;
- caractère : pour représenter les caractères d'un alphabet ;
- logique : pour représenter une des deux valeurs vrai ou faux.

Le type choisi pour une donnée conditionne les opérations possibles sur la donnée. Par exemple, il ne sera pas possible d'effectuer une soustraction sur des données de type caractère ou de type logique.

Variable
C'est un élément de l'algorithme, et du programme, qui permet de stocker et de restituer une valeur. Une variable ne peut stocker qu'une valeur conforme à son type (entier, réel, caractère, logique). Le type d'une variable est déterminé, selon les langages, soit au début du programme, soit lors de l'affectation.

Le contenu d'une variable peut évoluer au cours de l'exécution du programme. Lorsqu'une variable prend une première valeur lors de sa déclaration, on dit qu'elle est initialisée.

Constante
C'est un élément du programme qui identifie une valeur non modifiable. En opposition à variable, la valeur d'une constante ne change pas au cours de l'exécution du programme.

Séquence
Enchaînement d'instructions dans un programme.

Alternative
Instruction dans un programme, qui permet de commander le choix de séquences d'actions ou d'instructions à exécuter en fonction de l'évaluation d'une condition.

Répétitive
Instruction dans un programme, qui permet de commander la répétition (d'effectuer une boucle) d'une séquence d'actions ou d'instructions en fonction de l'évaluation d'une condition.

Annexe 2 : mise au point d'une page

Pour modifier le contenu d'un fichier web (qui porte l'extension .htm ou .html*)*, nous pouvons utiliser un simple éditeur de texte, par exemples le bloc note (Notepad) ou le mode édition de Mozilla. Il suffit alors d'ouvrir le fichier dans l'éditeur, faire les modifications et les enregistrer.

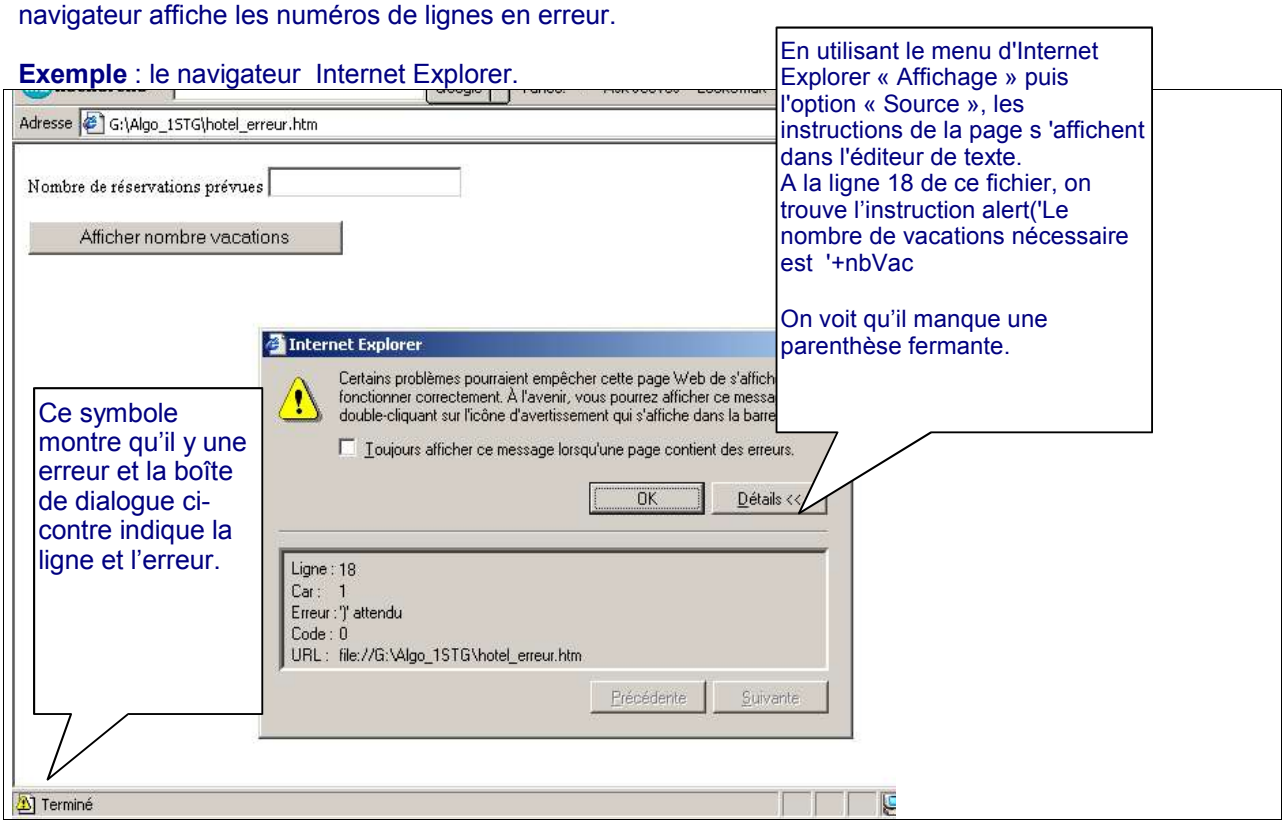
Pour tester la page après modification, il suffit de l'ouvrir dans un navigateur, ou encore de double-cliquer sur le nom du fichier suffixé `html` depuis un explorateur.

Attention si la page est déjà chargée dans le navigateur et qu'à l'aide de l'éditeur de texte, des modifications ont été effectuées, il faut **actualiser** la page dans le navigateur.

Il est aussi possible de visualiser le code d'une page web (instructions HTML et Javascript) en demandant l'affichage du code source dans le navigateur.

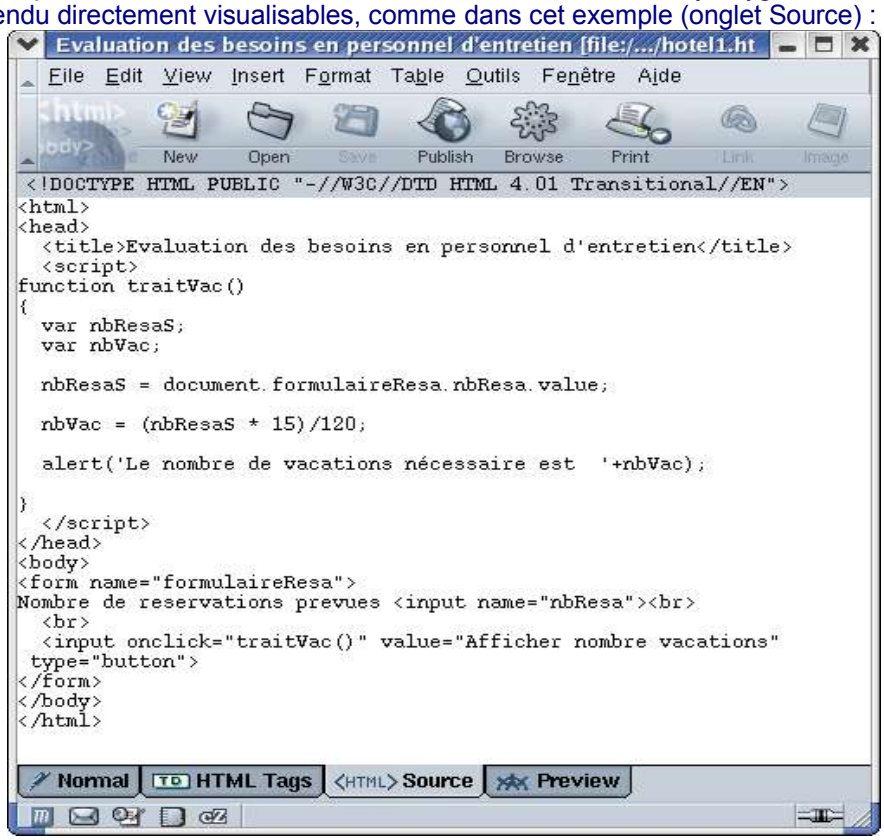
Sur certains navigateurs, si des instructions JavaScript comportent des erreurs, le navigateur affiche un symbole dans la barre d'état. En cliquant sur ce symbole et en demandant le détail des erreurs, le navigateur affiche les numéros de lignes en erreur.

Exemple : le navigateur Internet Explorer.



Utilisation de MOZILLA

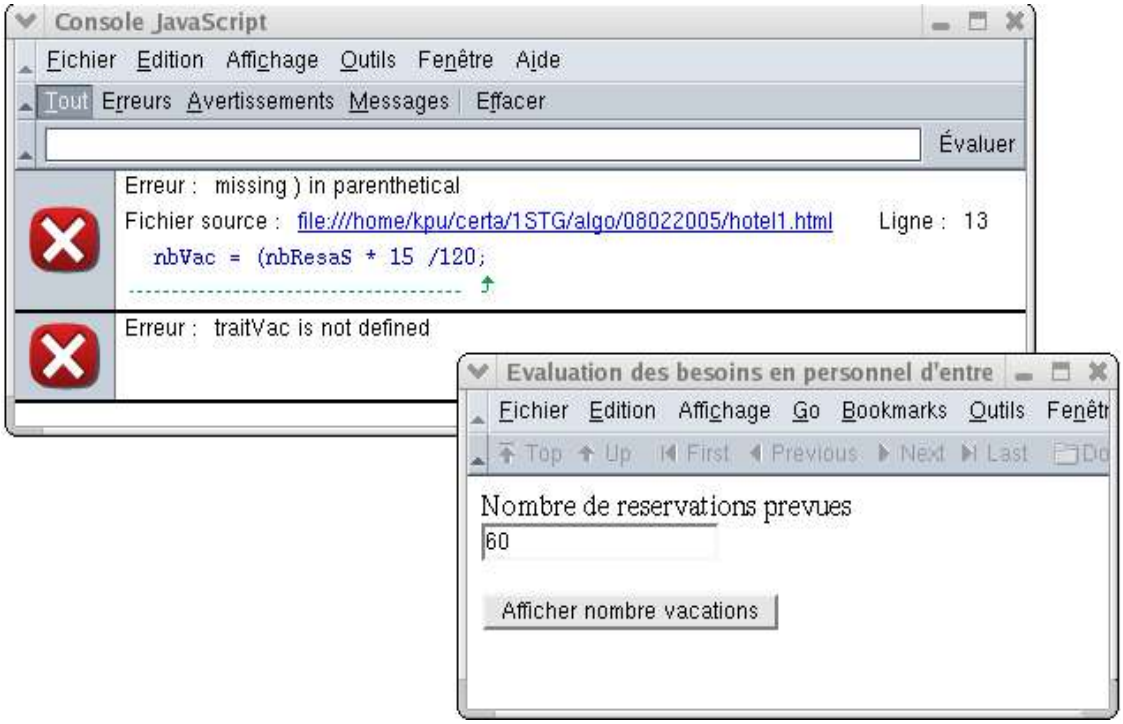
Mozilla, une communauté de développeurs et de testeurs, propose en accès libre un navigateur de qualité : http://www.mozilla.org/releases/#1.7.3. Une version française est disponible.
Pour éditer un document HTML, choisir : Fichier -> Éditer la page ou Ctrl+E
A partir de là on peut modifier, concevoir un document HTML en mode wysiwyg[6], mais les instructions sont bien entendu directement visualisables, comme dans cet exemple (onglet Source) :



Les erreurs JavaScript peuvent être consultées via :
Outils -> Développement Web -> Console Javascript

Dans l'exemple suivant, il manque une parenthèse fermante en ligne 13.

[6] What You See Is What You Get : le logiciel permet de visualiser immédiatement le résultat final obtenu.


Console JavaScript
Fichier Edition Affichage Outils Fenêtre Aide
Tout Erreurs Avertissements Messages Effacer
Évaluer
Erreur : missing) in parenthetical
Fichier source : file:///home/kpu/certa/1STG/algo/08022005/hotel1.html
Ligne : 13
nbVac = (nbResaS * 15 /120;
Erreur : traitVac is not defined
Evaluation des besoins en personnel d'entre
Fichier Edition Affichage Go Bookmarks Outils Fenêtr
Top Up First Previous Next Last
Nombre de reservations prevues
60
Afficher nombre vacations