



Structures de données - Les tableaux Analyse descendante - Les fonctions Test unitaire

Distributeur automatique de boisson

1 Mise en situation

Vous êtes chargé de développer un logiciel permettant le fonctionnement d'un distributeur de boissons. La partie opérative sera remplacée par des interactions avec l'utilisateur à l'aide de son clavier et de son écran.

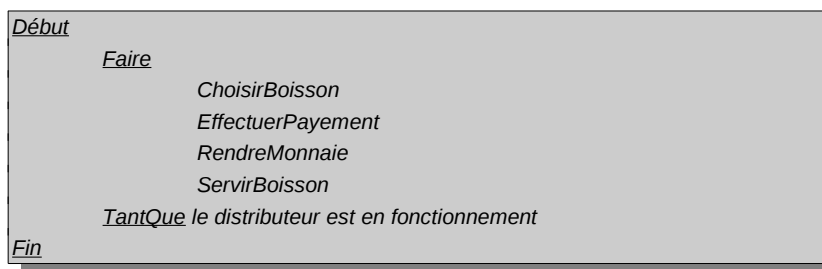
2 Fonctionnement de l'application

L'utilisateur voit apparaître un menu lui permettant de choisir une boisson. Lorsque ce choix est effectué, le prix et le nom de la boisson s'affichent, un nouveau menu apparaît pour la saisie des pièces à introduire. Au fur et à mesure du paiement le montant à payer diminue, jusqu'à atteindre le prix de la boisson ou une somme supérieure dans ce cas si cela est possible, le distributeur redonne la monnaie, ou demande de faire l'appoint. La boisson est servie, son montant réglé, le distributeur est prêt à proposer une autre boisson.

3 Décomposition fonctionnel

3.1 Fonction principale :

L'étude du fonctionnement de l'application fait apparaître un enchainement de plusieurs fonctionnalités. Un premier niveau d'analyse permet de réaliser la structure algorithmique de haut niveau suivante :



À ce niveau d'analyse, en déduire les principales structures de données nécessaires à la résolution du problème :

Nom de la structure	Description
NomBoissons	Tableau contenant le nom des différentes boissons disponibles sous forme de chaîne de caractères.
prixBoissons	Tableau contenant des réels
valeurPieces	Tableau contenant des réels exemple : 0,1 0,2 0,5 1 2
piecesIntroduites	Tableau contenant des entiers nombre de pièces par valeur
piecesRendues	Tableau contenant des entiers nombre de pièces par valeur
stockPieces	Tableau contenant des entiers nombre de pièces par valeur

Réalisez l'algorithme de cette fonction principale en utilisant le modèle proposé, vous-vous attacherez à rechercher les différents paramètres que les fonctions utiliseront par la suite.

Projet : Distributeur de Boissons		Auteur : Philippe CRUCHET	04/11/18
Fonction : Fonction principale			Page n° 1
Rôle : Assurer la distribution d'une boisson, collecter l'argent nécessaire et rendre la monnaie le cas échéant.			
Environnement :			
En entrée :	Rien		
En sortie :	Rien		
Paramètre d'entrée :	Aucun		
Paramètre de sortie :	Aucun		
Paramètre d'E/S :	Aucun		
Paramètre de retour :	Aucun		
Schéma algorithmique :		Lexiques :	
		Constantes : NB_BOISSONS 5 NB_PIECES 5	
		Variables locales : boisson <small>entier</small> nomBoissons[NB_BOISSONS] <small>chaîne de car.</small> prixBoissons[NB_BOISSONS] <small>réel</small> payement <small>booléen</small> valeurPieces[NB_PIECES] <small>réel</small> piecesIntroduites[NB_PIECES] <small>entier</small> montant <small>réel</small> stockPieces[NB_PIECES] <small>entier</small> rendu <small>booléen</small> piecesRendues[NB_PIECES] <small>entier</small>	
		Fonctions : <i>(hyperlien vers les fonctions)</i> ChoisirBoisson EffectuerPayement CalculerMontant RendreMonnaie ServirBoisson	
Début faire boisson ← ChoisirBoisson(nomBoissons,prixBoissons) si boisson ≠ -1 alors prix ← prixBoissons[boisson] payement ← EffectuerPayement(prix, valeurPieces, piecesIntroduites) si payement = vrai alors montant ← CalculerMontant(valeurPieces, piecesIntroduites) rendu ← RendreMonnaie(montant, valeurPieces, stockPieces,piecesRendues) si rendu = vrai alors ServirBoisson(boisson, valeurPieces, piecesRendues) FinSi FinSi Tant que (vrai) Fin			

3.2 Fonction ChoisirBoisson

Cette fonction a pour objet d'afficher la liste des boissons disponibles et le prix correspondant. L'utilisateur effectue un choix parmi les propositions qui lui sont faites ou annule sans choisir de boisson.

1 – Boisson 1 (prix 1)
2 – Boisson 2 (prix 2)
3 – Boisson 3 (prix 3)
4 – Boisson 4 (prix 4)
0 – Annuler
Votre choix :

Complétez la feuille d'algorithme suivante :

Projet : Distributeur de Boissons		Auteur : Philippe CRUCHET		04/11/18	
Fonction :		ChoisirBoisson			Page n° 2
Rôle :					
Environnement :					
En entrée :		Clavier			
En sortie :		Ecran			
Paramètre d'entrée :		nom[] chaîne de caractères prix[] réel			
Paramètre de sortie :		Aucun			
Paramètre d'E/S :		Aucun			
Paramètre de retour :		Entier indice de la boisson choisi ou -1 si annulation			
Schéma algorithmique : Début Pour indice allant de 1 à NB_BOISSON Ecrire : indice , nom[indice - 1] , prix[indice - 1] FinPour Ecrire : « 0 - Annuler » Ecrire : « votre choix » Faire lire : choix Tantque choix < 0 ou choix > NB_BOISSON retourner choix -1 Fin				Lexiques :	
				Constantes :	
				Variables locales :	
				indice entier choix entier	
				Fonctions :	

3.3 Fonction EffectuerPayement

Cette fonction a pour objet de récolter l'argent versé par l'utilisateur. La somme à verser est affichée et diminue au fur et à mesure qu'il entre des pièces dans le monnayeur. L'introduction des pièces est effectuée jusqu'à atteindre ou dépasser le montant de la boisson choisie. Il peut également annuler son choix. Prévoir le paramètre de retour en conséquence.

Proposez un visuel de l'interface sous la forme d'un menu :

Montant à régler : xxxx

1 – Pièce 2 €

2 – Pièce 1 €

3 – Pièce 50 cts

4 – Pièce 20 cts

5 – Pièce 10 cts

0 – pour Annuler

votre choix :

Complétez la feuille d'algorithme suivante :

Projet : Distributeur de Boissons		Auteur : Philippe CRUCHET		04/11/18	
Fonction : EffectuerPayement				Page n° 3	
Rôle :					
Environnement :					
En entrée :		Clavier			
En sortie :		Ecran			
Paramètre d'entrée :		montant réel valeurPieces[] réel			
Paramètre de sortie :		piecesInserees[] entier			
Paramètre d'E/S :					
Paramètre de retour :		Booléen vrai si paiement effectué faux si annuler			
Schéma algorithmique :				Lexiques :	
				Constantes :	
				Variables locales :	
				Fonctions :	

3.4 Fonction RendreMonnaie

L'objet de cette fonction est de calculer le nombre de pièces à rendre à l'utilisateur en fonction du prix de la boisson, la somme versée et la quantité de chaque pièce disponible dans le monnayeur.

Le principe retenu est le suivant : à partir du montant à rendre, le système essaye de donner des pièces de plus forte valeur, puis passe aux pièces de valeur inférieure. Il se peut que le monnayeur ne possède pas suffisamment de pièces de la valeur souhaitée, dans ce cas, il essaye avec la valeur inférieure. Si cela n'est pas possible, il ne redonne aucune pièce et retourne le fait qu'il n'a pas pu rendre la monnaie. Aucun affichage n'est prévu dans cette fonction.

Complétez la feuille d'algorithme suivante :

Projet : Distributeur de Boissons		Auteur : Philippe CRUCHET	04/11/18
Fonction :		RendreMonnaie	Page n° 4
Rôle :			
Environnement :			
En entrée :		Rien	
En sortie :		Rien	
Paramètre d'entrée :		montant ^{réel} valPiece[] ^{réel}	
Paramètre de sortie :		pieceRendu[] ^{entier}	
Paramètre d'E/S :		pieceEnStock[] ^{entier}	
Paramètre de retour :		booléen <i>vrai si opération effectuée, faux sinon</i>	

Schéma algorithmique :	Lexiques :
<p><u>Début</u></p> <p>Pour indice allant de 0 à NB_PIECES – 1 pieceRendue[indice] ← 0 <u>FinPour</u></p> <p>indice ← 0 <u>TantQue</u> montant > 0 <u>ET</u> indice < NB_PIECES <u>TantQue</u> montant > valPiece[indice] <u>ET</u> indice < NB_PIECES montant ← montant – valPiece[indice] pieceRendue[indice] ← pieceRendu[indice] + 1 pieceEnStock[indice] ← pieceEnStock[indice] – 1 <u>FinTantQue</u> indice ← indice +1 <u>FinTantQue</u> <u>Si</u> montant = 0 <u>Alors</u> retour ← vrai <u>Sinon</u> retour ← faux <u>FinSi</u> retourner retourne <u>Fin</u></p>	Constantes :
	Variables locales :
	<p>indice <small>entier</small> retour <small>booléen</small></p> <p><i>Fonctions :</i></p>

Afin de valider la fonction `RendreMonnaie`, il est nécessaire d'effectuer un test unitaire sur cette dernière, complétez le jeu de test permettant de vérifier son fonctionnement pour les différents cas de figure pouvant se présenter :

- Les pièces à rendre sont en nombre suffisant par rapport au montant à rendre. Plusieurs cas de figure sont à envisager pour vérifier le passage aux pièces de montant inférieur.
- Il n'y a pas les pièces nécessaires pour rendre la monnaie

On prévoit pour cela une interface permettant la saisie des pièces en stock pour chaque valeur, et la saisie du montant à rendre. Le programme affiche, si cela est possible, le nombre de pièces rendu pour chaque valeur.

Complétez la fiche de test si dessous :

[illegible]

3.5 Fonction ServirBoisson

La fonction **servirBoisson** a pour objet d'afficher le nom de la boisson et le rendu de monnaie.
Cette fonction ne présentant pas de difficulté particulière, on se contentera ici de compléter l'environnement de cette fonction.

Projet : Distributeur de Boissons	Auteur : Philippe CRUCHET	04/11/18
Fonction : ServirBoisson		Page n° 5
Rôle :		
Environnement :		
En entrée :		
En sortie :		
Paramètre d'entrée :		
Paramètre de sortie :		
Paramètre d'E/S :		
Paramètre de retour :		

4 Réalisation du projet

4.1 Codage de la fonction RendreMonnaie

Sous NetBeans, réalisez un nouveau projet nommé **Test_RendreMonnaie** de type Application en langage C.

Ajouter un fichier d'entête **distributeur.h** et un fichier source **distributeur.c**, ils contiendront respectivement les déclarations et les définitions des fonctions nécessaires au projet final.

À partir de votre algorithme (page n°4), codez la fonction RendreMonnaie dans les fichiers **distributeur** et le programme principal dans la fonction **main**. Ce dernier a pour rôle de tester la fonction.

Procédez au test unitaire de cette fonction et complétez le compte rendu de test dans le tableau ci-dessous.

Compte-rendu du Test F01	Date :	Version :
Opération x :	Le résultat obtenu n'est pas conforme aux attentes (description), la solution au problème est ...	
Etape 1 :	pieceRendues[] ne contient pas les bonnes valeurs, solution remettre le tableau à 0 dans la fonction RendreMonnaie.	

Lorsque tout est conforme aux attentes, passez à la suite de la programmation.

4.2 Codage des autres fonctions

Réalisez un nouveau projet nommé **DistributeurDeBoisson** de type Application en langage C. Copiez les fichiers **distributeur.h** et **distributeur.c** dans le répertoire de ce nouveau projet. Ajoutez les deux fichiers à votre projet et inclure le fichier **distributeur.h** dans le fichier **main.c**.

Procédez ensuite par étape :

- Définition des constantes dans le fichier **distributeur.h**
- codage d'une première fonction dans les fichiers **distributeur**
- Ajout des structures de données utiles à son fonctionnement dans la fonction **main**
- Appel de la fonction dans le **main** et vérification de son fonctionnement.

Procédez de manière similaire pour chacune des autres fonctions, les unes après les autres, pour au final obtenir le programme complet.

4.3 Complément pour le codage :

Fonction ChoisirBoisson

La constante à définir est **NB_BOISSONS** initialisée avec la valeur 5. Dans la fonction **main**, déclarez le tableau **nomBoissons** contenant des chaînes de caractères de la manière suivante :

```
char * nomBoissons [NB_BOISSONS] = { « Café sucré », « Café au lait », « Chocolat »,  
                                       « Coca cola », « Orangina » } ;
```

En correspondance d'indice, déclarez le tableau **prixBoissons** contenant des réels avec les prix 1,5 € pour les boissons chaudes et 1 € pour les boissons froides.

Fonctions relatives à la gestion des pièces

Pour le programme on prévoit que le monnayeur dispose de pièces de 10 cts, 20 cts, 50 cts, 1 € et 2 €. Les pièces en stock dans le monnayeur sont initialisées dans le programme principal par des valeurs constantes.

Pour les différents tableaux liés à la gestion des pièces, les indices doivent correspondre avec leur valeur.