SAÉ 1.01 - Implémentation d'un Besoin

1 Equipe

Fakhfakh Ahmed TD II – TP4

Garcia Angel TD II – TP4

2 Exercice traité : Pack 14

Jeu déplacer pièces

3 Rappel des spécifications du programme

3.1 Spécifications initiales

Scénario nominal : Le joueur joue et gagne.

Scénario alternatif : Le joueur abandonne.

Scénario alternatif : Le joueur se trompe dans la saisie

3.2 Spécifications complémentaires = extensions traitées

Le joueur peut décider s'il veut commencer à la case 1 pour finir à la case 7 ou commencer à la case 7 pour finir à la case 1

4 Algorithmes du programme (action principale et ses sous-actions)

4.1 Déplacer pièces

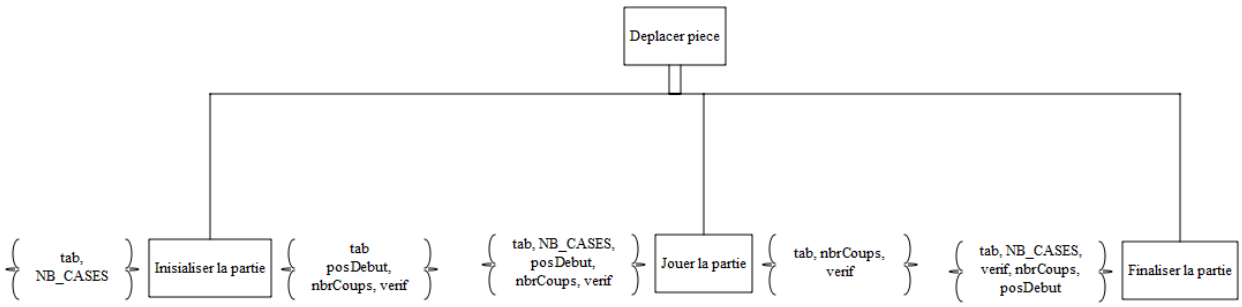
4.1.1 Initialiser la partie, jouer la partie et finaliser la partie

Ce sont les 3 grandes étapes du déroulement du jeu

4.1.2 Stratégie de l'algorithme mise en œuvre

On commence par initialiser la partie, puis on joue la partie et on affiche le résultat pendant la finalisation.

4.1.3 Algorithme



4.1.4 Dictionnaire des éléments associés à cet algorithme

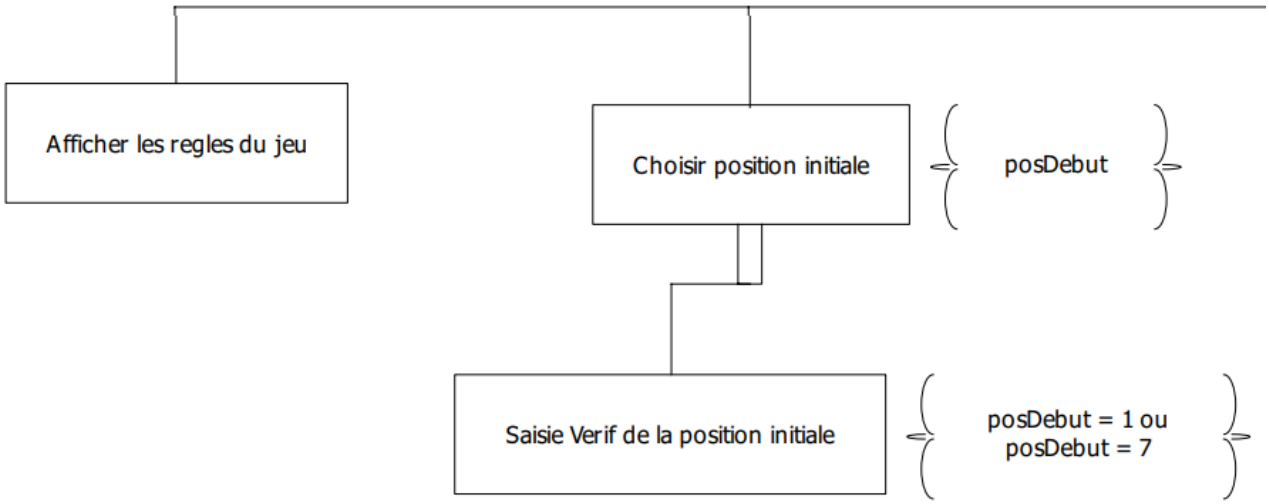
Nom	Type	Signification
NB_CASES	const unsigned short int	Taille du tableau
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
posDebut	unsigned short int	Position initiale que le joueur a choisie
nbrCoups	unsigned short int	Le nombre de coups jouer par le joueur
verif	bool	Variable qui contient vrai si le joueur à gagner

4.2 Initialiser la partie – Règles du jeu et choix position initiale

4.2.1 Afficher les règles du jeu puis demande à l'utilisateur s'il veut commencer à la position 1 ou à la position 7.

4.2.2 On affiche les règles puis grâce à l'algorithme de saisie-vérif on demande au joueur de rentrer s'il veut commencer à la position 1 ou 7.

4.2.3 Algorithme



4.2.4 Dictionnaire des éléments associés à cet algorithme

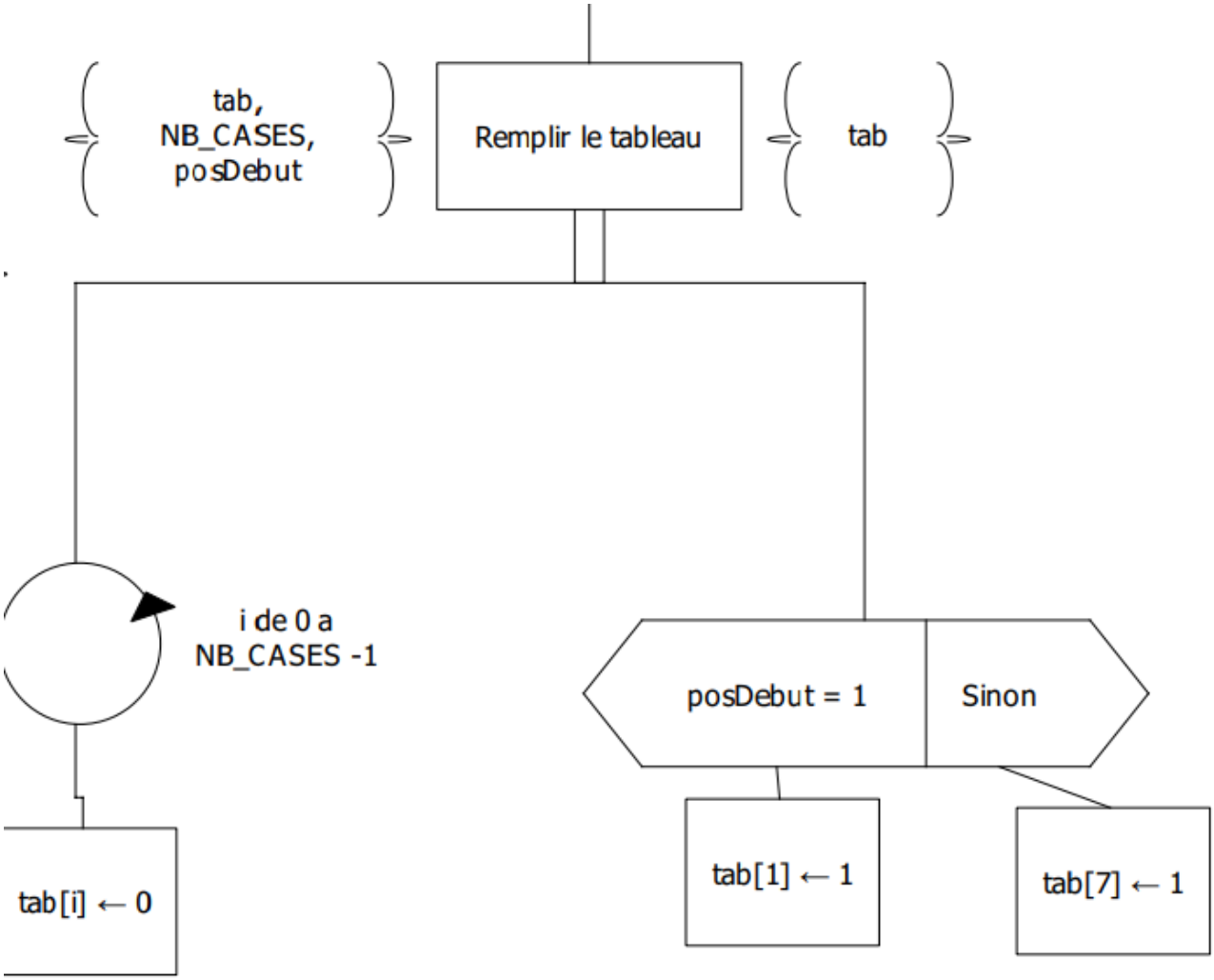
Nom	Type	Signification
posDebut	unsigned short int	Position initiale que le joueur a choisie

4.3 Initialiser la partie – Remplir le tableau

4.3.1 Remplir le tableau de 0 sauf à la position où le joueur commence.

4.3.2 On fait un parcours entier du tableau avec une boucle « for » en mettant un 0 à chaque case. Ensuite on met un 1 à la case où le joueur commence.

4.3.3 Algorithme



4.3.4 Dictionnaire des éléments associés à cet algorithme

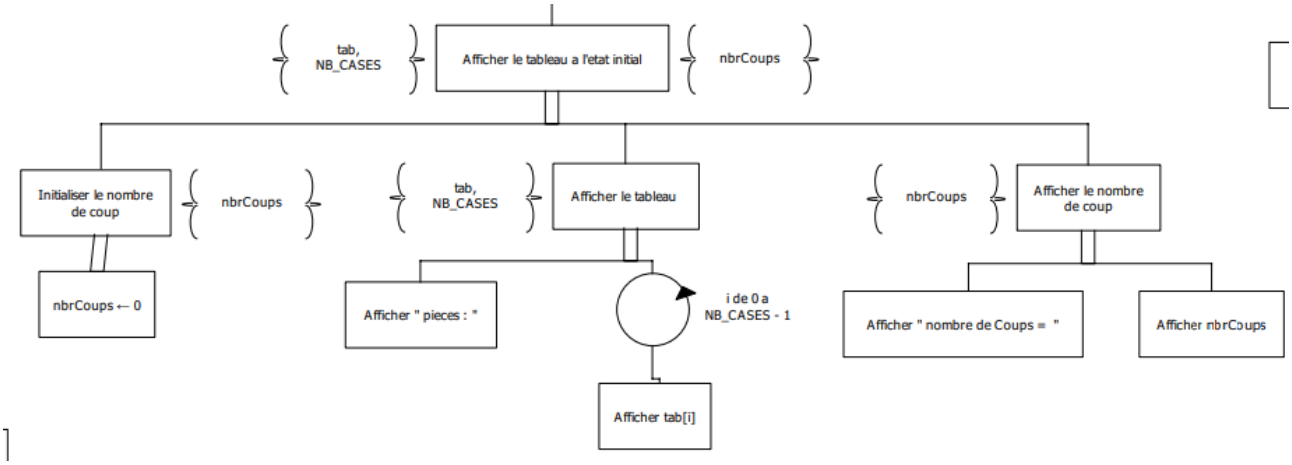
Nom	Type	Signification
NB_CASES	const unsigned short int	Taille du tableau
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
posDebut	unsigned short int	Position initiale que le joueur a choisie

4.4 Initialiser la partie – Afficher le tableau à l'état initial

4.4.1 Affiche le tableau a son état initial avant que le joueur ne commence à jouer et initialise la variable nbrCoup.

4.4.2 On initialise nbrCoup à 0 et on affiche le tableau en affichant tous les éléments du tableau en le parcourant entièrement. On affiche aussi le nombre de coup.

4.4.3 Algorithme



4.4.4 Dictionnaire des éléments associés à cet algorithme

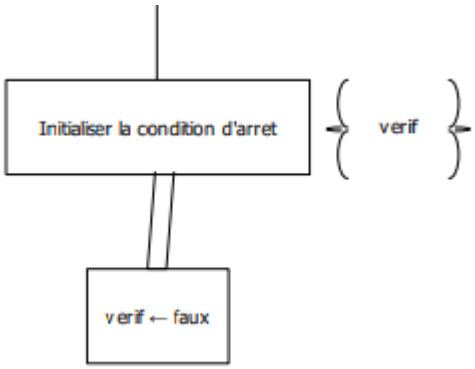
Nom	Type	Signification
NB_CASES	const unsigned short int	Taille du tableau
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
nbrCoups	unsigned short int	Le nombre de coups jouer par le joueur

4.5 Initialiser la partie – Initialiser la condition d'arrêt

4.5.1 Initialiser la condition d'arrêt quand le joueur gagne

4.5.2 On initialise la variable verif a faux

4.5.3 Algorithme



4.5.4 Dictionnaire des éléments associés à cet algorithme

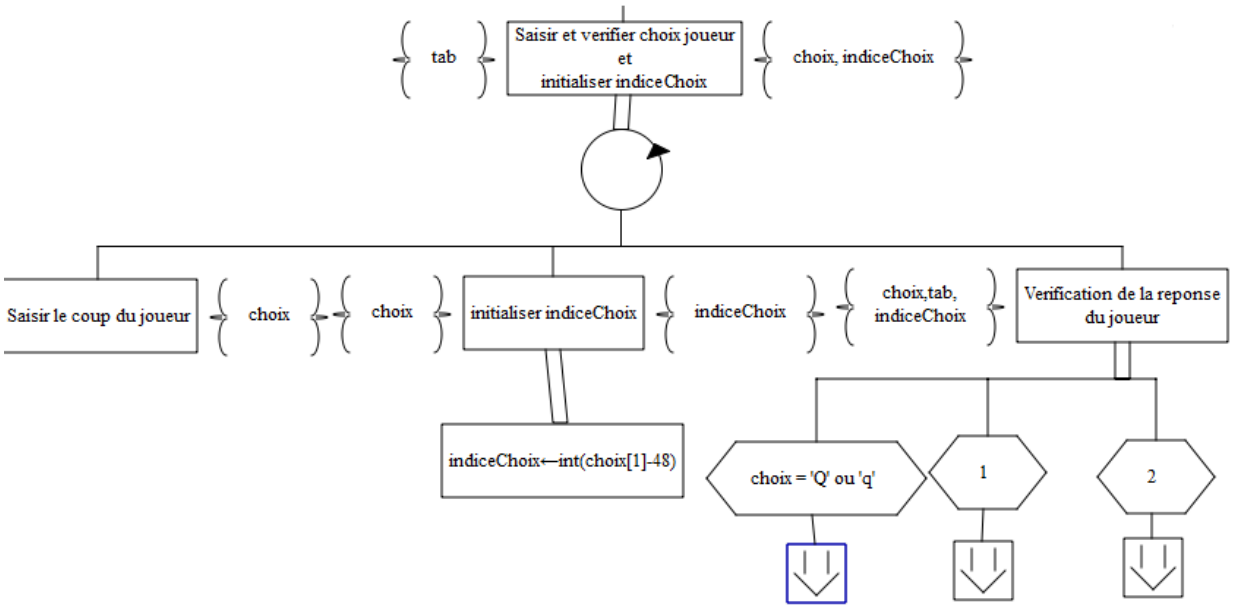
Nom	Type	Signification
verif	bool	Variable qui contient vrai si le joueur à gagner

4.6 Jouer la partie – Choix du coup

4.6.1 Le joueur choisit son coup

4.6.2 Le joueur saisis un coup, on initialise indice choix et on vérifie que la saisie soit bonne

4.6.3 Algorithme



1 : choix[0] = 'A' ou 'a' et
choix.size()=2 et
indiceChoix € [1,2,..,8] et
tab[indiceChoix]>=1

2 : choix[0] = 'S' ou 's' et
choix.size()=2 et
indiceChoix € [1,2,..,8] et
tab[indiceChoix -1]>=1 et
tab[indiceChoix+1]>=1

4.6.4 Dictionnaire des éléments associés à cet algorithme

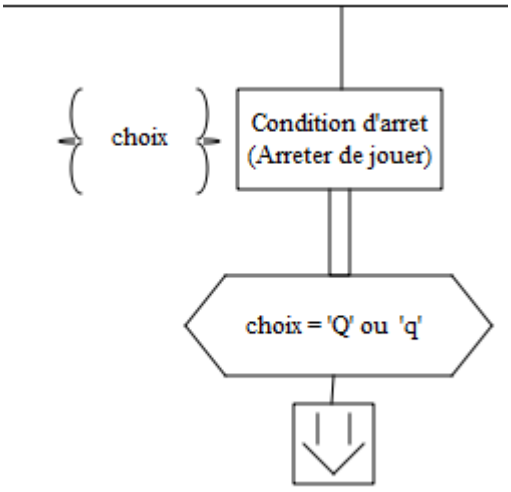
Nom	Type	Signification
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
indiceChoix	int	Indice choisi par l'utilisateur
choix	string	Coup que le joueur a saisi

4.7 Jouer la partie – Condition d'arrêt

4.7.1 Arrête le jeu si le joueur décide de quitter

4.7.2 On vérifie la saisie du joueur, si c'est égal a 'q' ou 'Q' alors on quitte la boucle principale.

4.7.3 Algorithme



4.7.4 Dictionnaire des éléments associés à cet algorithme

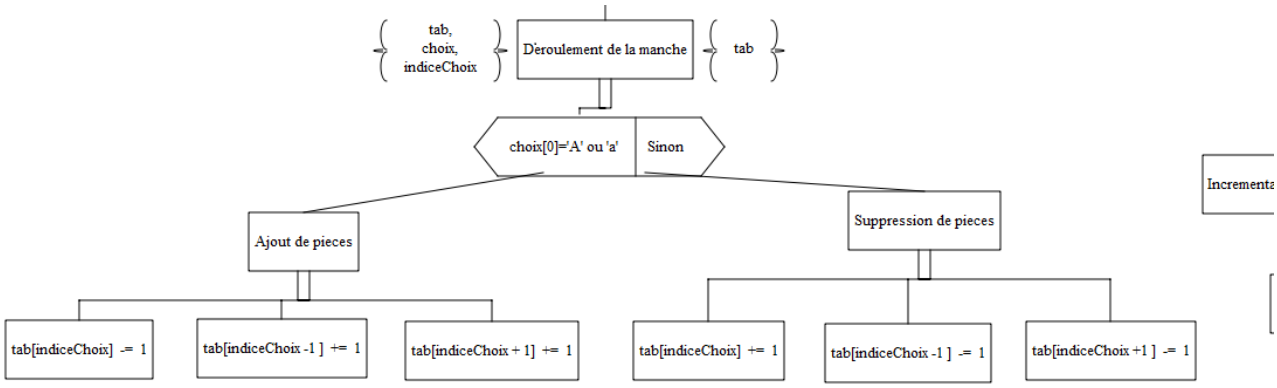
Nom	Type	Signification
choix	string	Coup que le joueur a saisi

4.8 Jouer la partie – Déroulement de la manche

4.8.1 Gère le déroulement de la manche, si on ajoute ou supprime des pièces

4.8.2 On ajoute les pièces nécessaires si le choix du joueur est d'ajouter et sinon on supprime.

4.8.3 Algorithme



4.8.4 Dictionnaire des éléments associés à cet algorithme

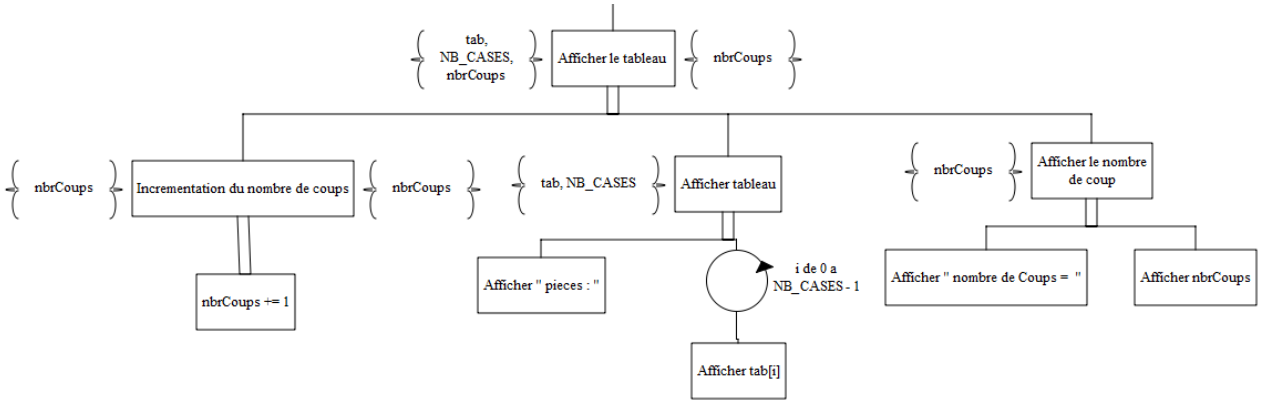
Nom	Type	Signification
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
indiceChoix	int	Indice choisi par l'utilisateur
choix	string	Coup que le joueur a saisi

4.9 Jouer la partie – Afficher le tableau

4.9.1 On affiche le tableau et le nombre de coup

4.9.2 On commence par incrémenter le nombre de coup de 1, on affiche ensuite le tableau en le parcourant entièrement et on finit par afficher le nombre de coup.

4.9.3 Algorithme



4.9.4 Dictionnaire des éléments associés à cet algorithme

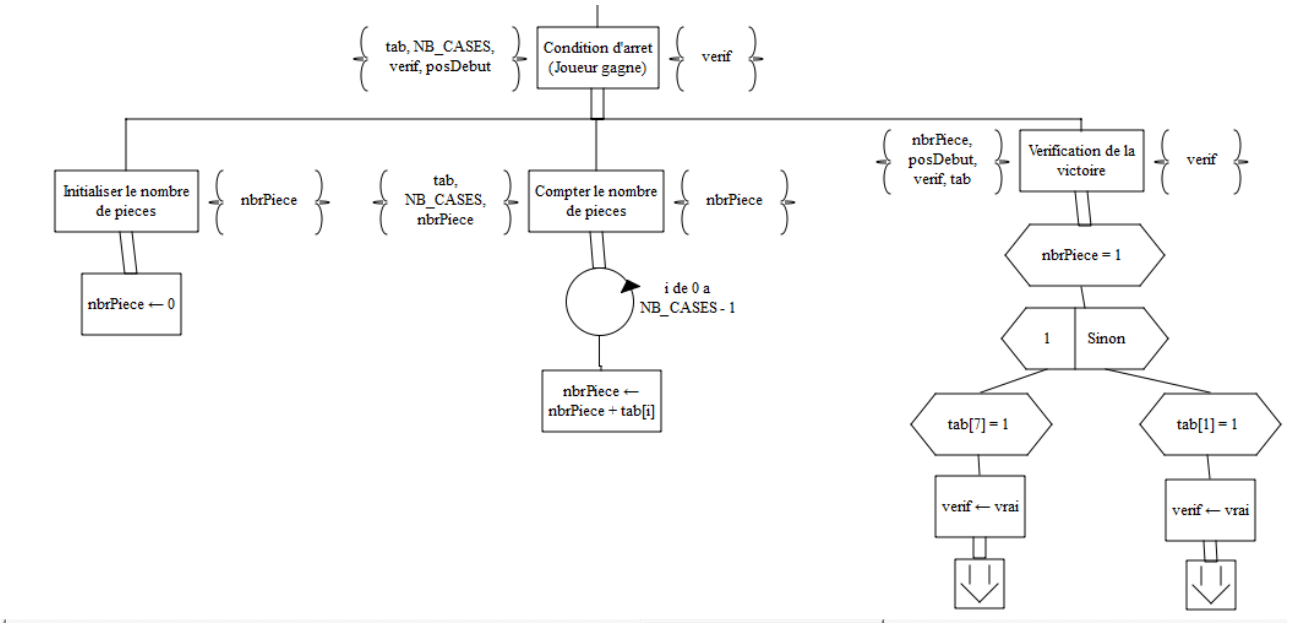
Nom	Type	Signification
NB_CASES	const unsigned short int	Taille du tableau
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
nbrCoups	unsigned short int	Le nombre de coups jouer par le joueur

4.10 Jouer la partie – Condition d'arrêt (joueur gagne)

4.10.1 Vérifier si le joueur à gagner

4.10.2 On met la valeur 0 dans la variable nbrPiece, ensuite on parcourt tout le tableau en incrémentant de 1 la variable nbrPiece quand il y a une pièce dans le tableau. S'il y a une seule pièce présente et qu'elle est placé à la bonne place alors on sort de la boucle principale.

4.10.3 Algorithme



4.10.4 Dictionnaire des éléments associés à cet algorithme

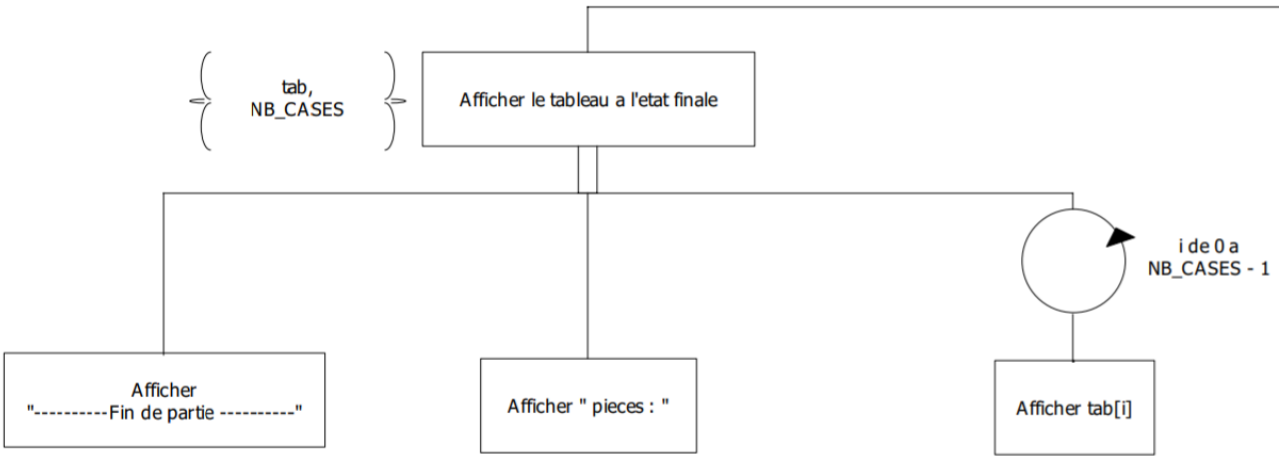
Nom	Type	Signification
NB_CASES	const unsigned short int	Taille du tableau
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.
posDebut	unsigned short int	Position initiale que le joueur a choisie
verif	bool	Variable qui contient vrai si le joueur à gagner

4.11 Finaliser la partie – Afficher le tableau à l'état final

4.11.1 Affiche le tableau à la fin de la partie

4.11.2 On parcourt le tableau en entier et on affiche chaque élément.

4.11.3 Algorithme



4.11.4 Dictionnaire des éléments associés à cet algorithme

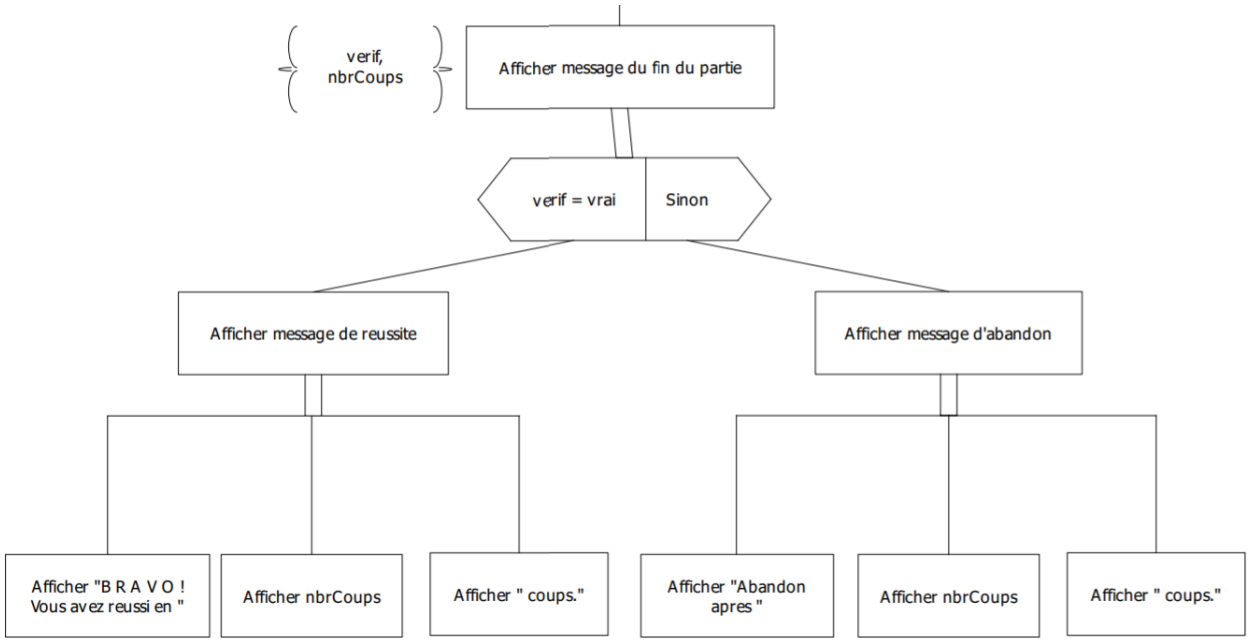
Nom	Type	Signification
NB_CASES	const unsigned short int	Taille du tableau
tab	unsigned short int	Tableau de taille NB_CASES contenant le nombre de pièces dans chaque case.

4.12 Finaliser la partie – Afficher message de fin de partie

4.12.1 Affiche un message de fin au joueur

4.12.2 On regarde si le joueur a gagné la partie ou non et on lui affiche le message correspondant.

4.12.3 Algorithme



4.12.4 Dictionnaire des éléments associés à cet algorithme

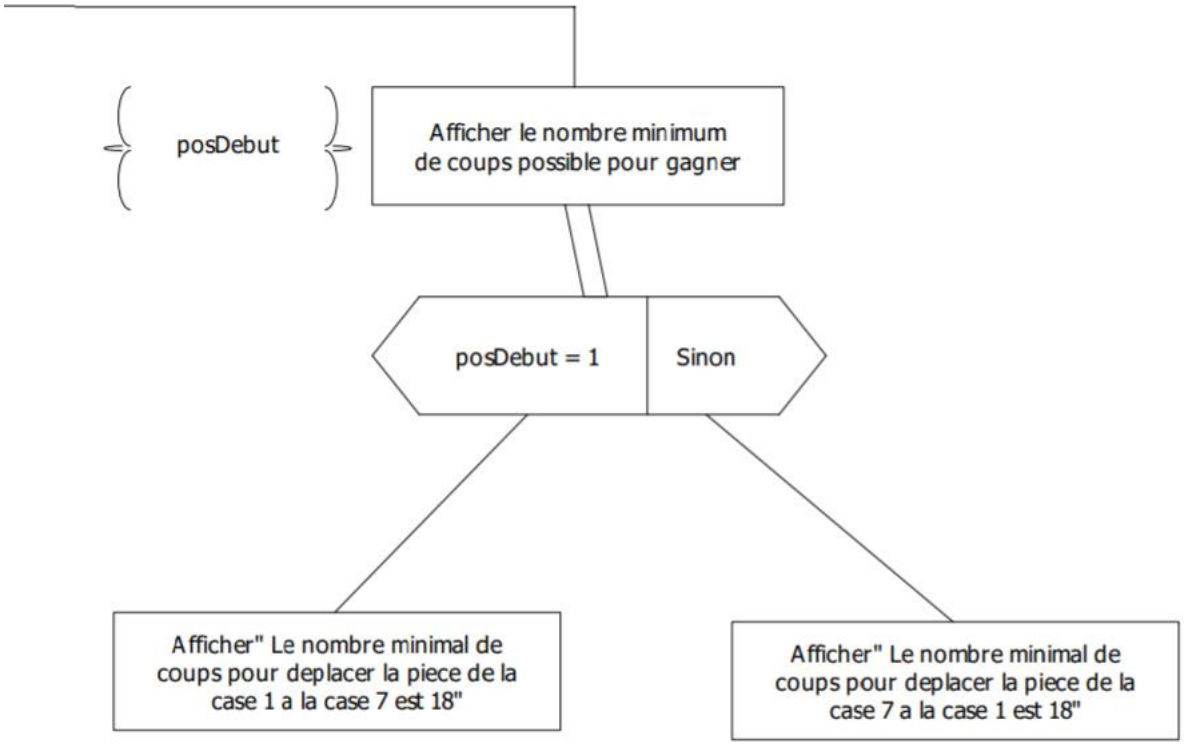
Nom	Type	Signification
nbrCoups	unsigned short int	Le nombre de coups jouer par le joueur
verif	bool	Variable qui contient vrai si le joueur à gagner

4.13 Finaliser la partie – Afficher le nombre minimum de coup possible

4.13.1 Afficher le nombre minimum de coup possible pour gagner la partie

4.13.2 On regarde où le joueur a commencé pour ajuster le message

4.13.3 Algorithme



4.13.4 Dictionnaire des éléments associés à cet algorithme

Nom	Type	Signification
posDebut	unsigned short int	Position initiale que le joueur a choisie

5 Traces d'exécution

- Le joueur commence à la case numéro 1 et abandonne.

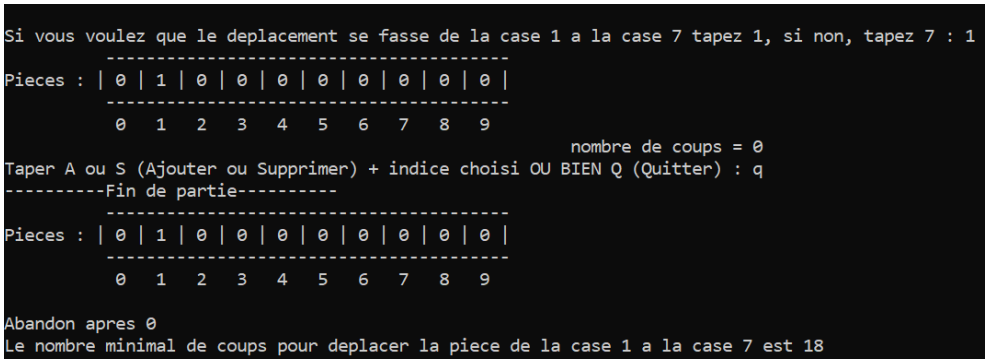


Figure 1 : Comportement lié au scénario de l'extension et au scénario d'abandon.

- Le joueur commence à la case numéro 7 et abandonne.

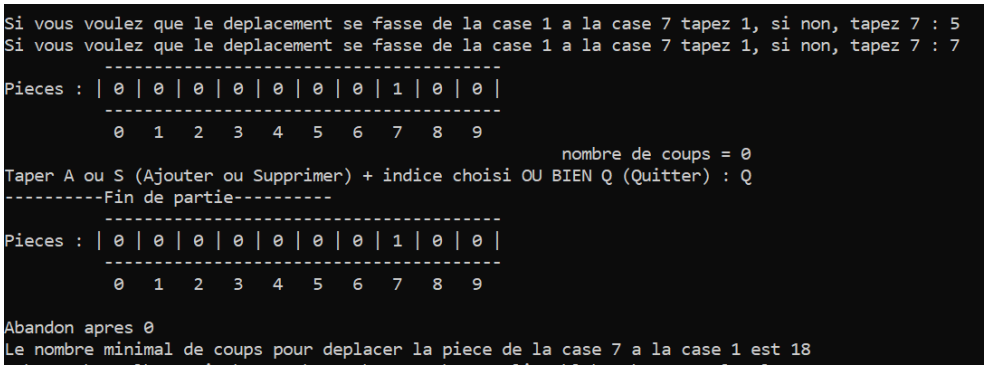


Figure 2 : Comportement lié au scénario de l'extension et au scénario d'abandon.

- Le joueur commence à la case numéro 1 et gagne, les erreurs de saisie sont gérées.

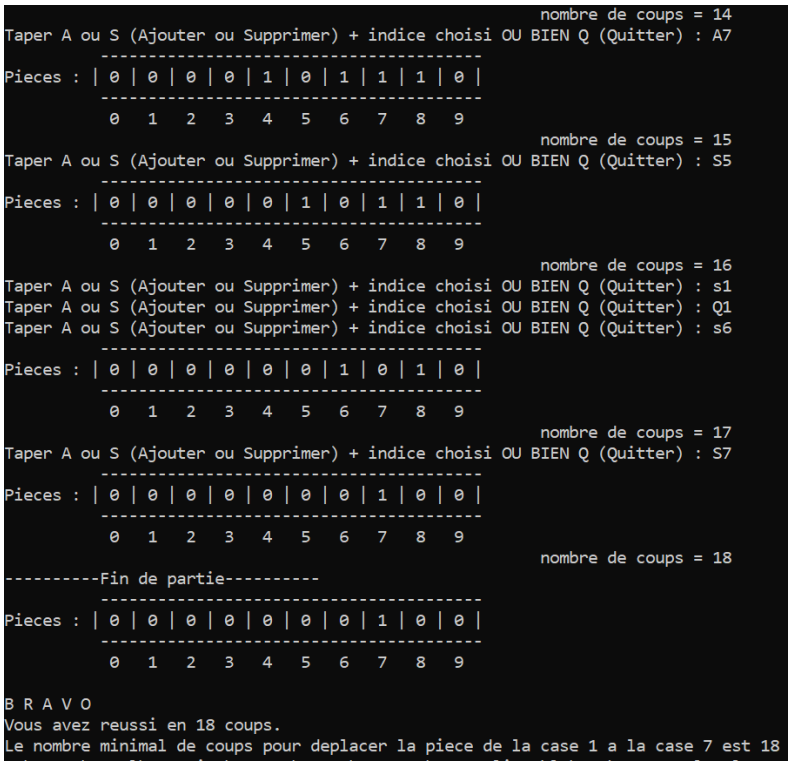


Figure 3 : Comportement lié au scénario nominal et le scénario où la saisie est fausse.

- Le joueur commence à la case numéro 7 et gagne, les erreurs de saisie sont gérées.

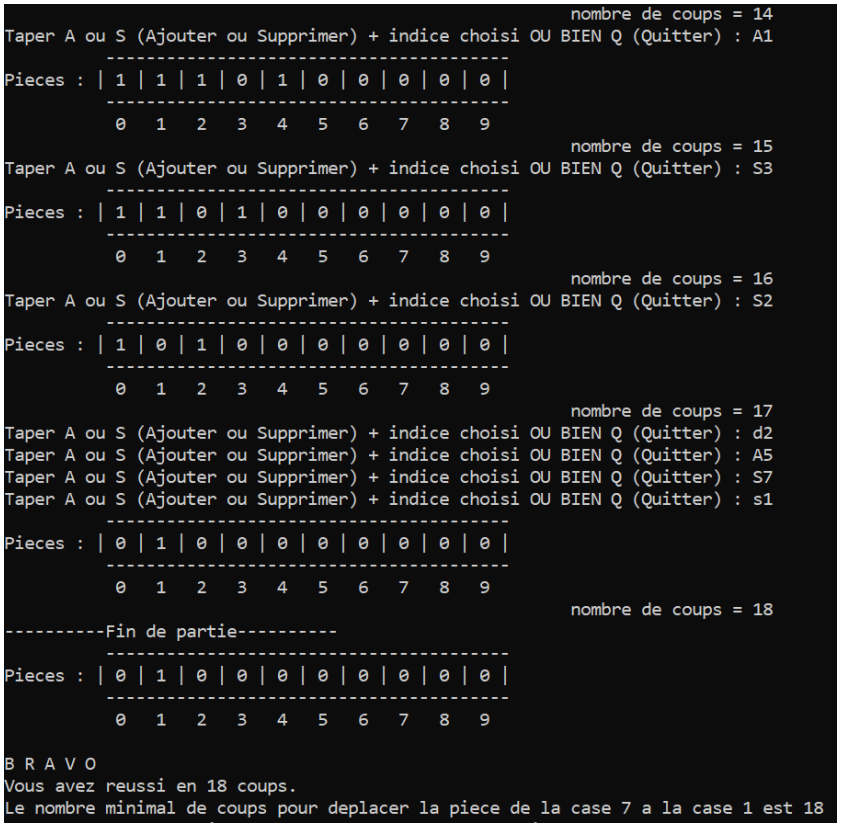


Figure 4 : Comportement lié au scénario de l'extension et où les erreurs de saisie sont prises en compte.

6 Remarques

Informations que les étudiants souhaitent communiquer aux enseignants au sujet de cette SAE :

- On a décidé de repartir le travail différemment que la répartition pour le jeu 1, ce n'était pas la meilleure idée mais on aura essayé et on a appris de nos erreurs.
- La propreté de l'algorithme n'a pas été un problème cette fois, nous avons directement réussis a produire un algorithme propre et bien décomposer afin de l'intégrer au mieux au livrable.
- Amélioration possible :
  Dans le 4.10 (Jouer la partie – Condition d'arrêt (joueur gagne)) on pourrait commencer à faire la vérification à partir du 18^{ième} coup et on pourrait arrêter la vérification quand 2 pièces sont détectées.
- Nous avons décidé cette fois ci d'intégrer directement les extensions ce qui a réduit notre charge de travail et nous permet de faire le jeu avec extensions cette fois ci.
- Nous avons remarqué que le premier saisie-vérif pour le choix de là où le joueur commence ne marchait que si on se trompait dans la saisie du chiffre mais pas si on y mettait une lettre par exemple. Nous n'avons pas réussi régler ce souci et nous en avons déduit le soucis venait du fait que ce que l'on saisissait été un entier et que ça bugger quand nous y mettions un caractère.

7 Code C++

Fichier main.cpp joint au dossier avec en-tête certifiant l'originalité du code produit.