Activité : Tracer l'exécution d'un Algorithme

Mise en situation

Lorsqu'un programme ne fonctionne pas correctement, on doit être capable de comprendre les étapes de son exécution pour trouver le problème. On peut bien sûr utiliser un débogueur symbolique, mais il est important de pouvoir le faire également « à la main ». Cela permet par exemple de faire la trace de l'exécution d'un programme écrit sur du papier ou d'analyser une petite portion de code d'un grand programme. Cela vous permet également de bien comprendre le fonctionnement des structures de contrôle et de l'affectation.

Consigne

Lisez attentivement le texte qui précède chacun des exercices. Chaque partie vous montre comment faire la trace de l'exécution d'un programme à l'aide d'un exemple qui met en œuvre l'une des structures de contrôle (séquence, choix, boucle définie, boucle indéfinie). Prenez le temps de bien étudier la trace d'exécution de chaque exemple avant de faire l'exercice correspondant.

Objectifs

À la fin de ce travail, vous devez :

> Être capable de tracer l'exécution d'un programme de quelques lignes composé d'affectation et de structures de contrôle (séquence, choix et boucles) et de présenter le résultat sous la forme d'un tableau.

Tracer l'exécution d'une séquence

Ce premier exemple vous montre comment faire la trace de l'exécution d'une séquence d'instruction. En l'occurrence, il s'agit essentiellement d'une séquence d'affectation.

```
1  Algorithme Exemple1
2  Variable a,b,c: entier;
3   Debut
4        b← 5;
5        a ← b * 3;
6        c ← a – b;
7        b ← c;
8        b ← b + 20;
9        c ← (b + a) * 2;
10       Écrire(c);
11 Fin
12
13
```

Fig. 1

La figure 1 montre le programme dont nous voulons faire la trace et la figure 2 montre la trace d'exécution de ce programme présentée sous la forme d'un tableau.

Pour parvenir à ce résultat, on commence par faire l'inventaire des variables qui seront utilisées en cherchant les **déclarations de variables**. On reporte le nom de chaque variable dans une des cases prévues à cet effet en haut à droit du tableau. Dans le programme Exemple1, il y a trois variables : a, b et c.

Trace de l'exécution d'un algorithme Exemple 1

N° étape	Instruction	Description / Remarques	État des variables locales (après exécution de l'instruction)				
			a	b	c		
1	Variable a,b,c :entier	Déclaration des variables	-	-	-		
2	b← **5**	Affectation de la valeur 5 à b		5			
3	a ← b * 3	Évalue l'expression b * 3 = 5 * 3=15 puis affecte la valeur à la variable a	15	5			

4	c ← a – b	Affecter c la valeur de l'expression a – b = 15 – 5 = 10	15	5	10		
5	b ← c	Évalue l'expression c qui est égale à 10 puis affecte la valeur à la variable b	15	10	10		
6	b ← b +20	Évalue l'expression b + 20 = 10 +20 = 30 puis affecte la valeur à la variable b	15	30	10		
7	c ← (b + a) * 2	Évalue l'expression (b + a) * 2 = (30 + 15) * 2 = 90 puis affecte la valeur à la variable c	15	30	90		
8	Écrire(c)	Affiche la valeur de c : 90	15	30	90		

On utilise en suite une ligne pour chaque étape de l'exécution du programme. Comme dans le programme Exemple1 il n'y a ni structure de choix ni structure de boucle, chaque instruction est une étape du programme.

Pour rappel, une affectation s'exécute en deux temps :

1. Évaluation de l'expression de la **partie droite**.
2. Affectation de la valeur de l'expression à la variable de la **partie gauche**.

Pour évaluer l'expression, il suffit de remplacer les variables, s'il y en a, par leur valeur à la fin de l'étape précédente, puis de faire le calcul en respectant les règles de priorité des opérations. Pour finir, on écrit la valeur de chaque variable dans la colonne correspondante en utilisant éventuellement une couleur différente pour indiquer la valeur qui a été modifiée par cette étape. Si la valeur d'une variable n'est pas définie, on l'indique à l'aide d'un tiret.

Exercice 1 : Faites la trace du programme suivant en utilisant le Tableau ci-dessous et expliquez ce que font les lignes 7, 8 et 9 de ce programme.

```
1 Algorithme Exercice1
2 Variable y,a,b,tmp;
3 Debut
4     x ← 5;
5     a ← 5;
6     b ← 10;
7     tmp ← a;
8     a ← b;
9     b ← tmp;
```

```
10      y ← a * x + b;
11      Écrire(y);
12  Fin
```

N° étape	Instruction	Description / Remarques	État des variables locales (après exécution de l'instruction)				
1							
2							
3							
4							

Tracer l'exécution d'une structure de choix

Pour faire la trace d'un programme, on doit parfois faire des suppositions. Dans l'exemple de la figure 3, on demande à l'utilisateur de saisir un nombre. Comme la trace de l'exécution dépend de la valeur saisie, on doit admettre une valeur et exécuter le programme avec cette valeur.

```
  Algorithme Exemple2
1 Variable val, res: reel
2 Debut
3     Lire(val)
4     Si (val >= 10 et val <= 20) alors
5             res ← -val;
6     sinon
7             res ← val * 2;
8     FinSi
9
10    res ← res + 20;
11
12    Écrire(res);
  Fin
```

Fig. 3 – Programme Exemple2

De plus, ce programme contient une structure de choix dont la condition du choix dépend de la valeur saisie par l'utilisateur. Il y a donc deux chemins d'exécution possibles. Pour illustrer cela, la figure 4 montre un organigramme de programmation(ou simplement organigramme) qui permet de bien visualiser les deux chemins que peut suivre le fil d'exécution dans une structure de choix.

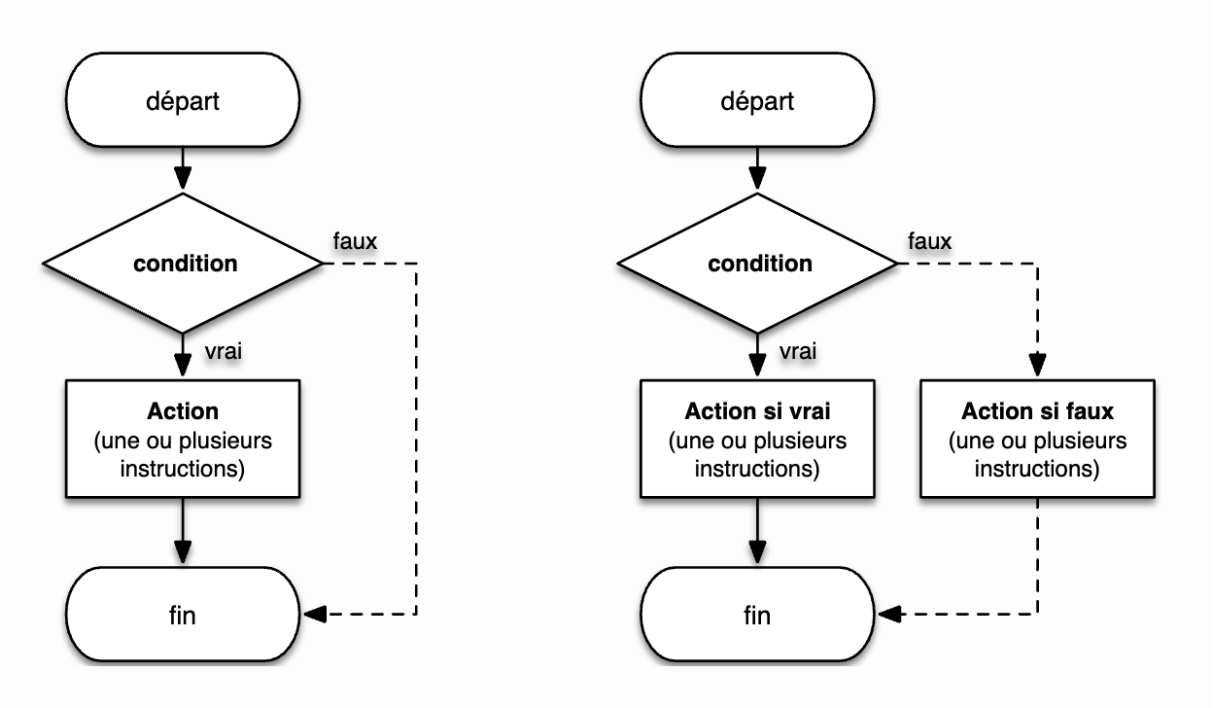

Fig. 4 – Organigramme d'une structure de choix

La trace de l'exécution pour une valeur donnée ne suivra qu'un seul de ces deux chemins. Si l'on veut analyser les deux chemins, on doit faire la trace une première fois avec une valeur qui remplit la condition, puis une seconde fois avec une valeur qui ne la remplit pas. La figure 5 montre deux traces d'exécution possible du programme Exemple2.

N° étape	Instruction	Description / Remarques	État des variables locales (après exécution de l'instruction)				
			val	res			
1	Variable val , res :reel	Déclaration des variables	-	-			
2	Lire(val)	Affecte la valeur saisie par l'utilisateur	15	-			
		L'utilisateur saisit la valeur 15					

3	**si** (val>=10 **et** val<=20)	Évalue l'expression 15>=10 et 15<=20 = vrai et vrai= vrai	15	-			
4	res ← val *2	Évalue l'expression 5*2=10 et affecte la valeur à la variable res	5	10			
5		Évalue l'expression 10 + 20 = 30 et affecte la valeur à la variable res	5	30			
6	Ecrire(res)	Affiche la valeur de res : 30	5	30			

Remarquez dans ces traces, comment on peut utiliser deux lignes du tableau pour une seule étape (étape 2), dans le cas où les espaces réservés à l'instruction ou comme ici à la description et aux remarques seraient insuffisants.

Exercice 2 : Faites les traces de trois exécutions du programme Exercice2 en admettant que la première fois l'utilisateur saisisse un 10, la deuxième fois un 20 et la troisième fois un 40. Est-ce que tous les chemins possibles ont été empruntés ? Décrivez à quels intervalles correspond chacune des deux conditions.

```
1  Algorithme Exercice2
2  Variable res, n :entier
3  Debut
4          Lire(n) ;
5
6          si (n > 10 et n <= 20) alors
7               res ← n * 3;
8           sinon si(n <= 10 ou n > 40) alors
9                       res ← n * 8;
10                  sinon
11                     res = n;
12                 Finsi
13         FinSi
14
15         Ecrire(res);
16 Fin
```

Tracer l'exécution d'une boucle POUR

Une boucle **pour** permet de répéter une ou plusieurs instructions un nombre de fois bien définie. Pour rappel, une boucle définie (ou boucle **pour**) dit que l'on doit répéter les instructions pour chaque valeur entière comprise entre une valeur initiale et une valeur finale de la variable de boucle (généralement i).