Focus sur le I du modèle PRIMM

Le modèle des blocs

Nathalie Maïer & Nathalie Weibel ,
PCD NSI - Académie de Normandie

12 février (Caen) et 13 février (Rouen) 2024

1. PRIMM

Que signifie PRIMM ?

PRIMM signifie : **Predict** **Run** **Investigate** **Modify** **Make**

que l'on peut traduire par : *Prédire, exécuter, enquêter, modifier, créer*.

Ce modèle, élaboré par Sue Sentance, cherche à aider les enseignants à concevoir des situations d'enseignement dans lesquelles les élèves s'engagent de manière progressive dans la compréhension d'un programme.

Les cinq étapes de PRIMM

1. **Predict** (Prédire)

Les élèves commencent avec le code d'un programme qui fonctionne, et sont invités à l'examiner et à prédire ce qu'il réalise. Ils peuvent dessiner ou écrire le résultat qu'ils anticipent.

2. **Run** (Exécuter)

Les élèves exécutent le code, fourni par l'enseignant et observent ce que fait le programme. Ils testent ainsi leurs prédictions et s'interrogent sur les éventuels écarts constatés.

Les cinq étapes de PRIMM

3. **Investigate** (Enquêter)

Les élèves entrent ensuite dans les détails du programme et l'examinent ligne par ligne. Ils explorent comment le programme réalise ce qu'ils ont prédit, puis observé lorsqu'ils l'ont exécuté. L'enseignant propose une série d'activités pour explorer la structure du code ; cela comprend des activités telles que l'exécution pas à pas (tracing), l'explication, l'annotation, la correction d'erreurs, mais toujours en s'appuyant sur un programme complet, fourni par l'enseignant.

Les cinq étapes de PRIMM

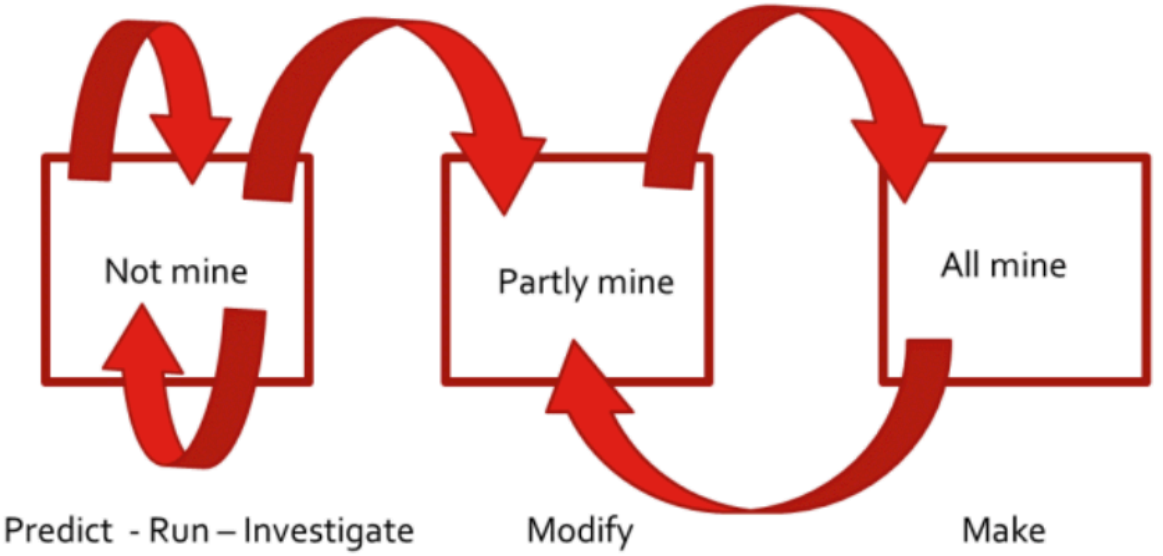
4. **Modify** (Modifier)

Les élèves modifient le programme pour changer sa fonctionnalité à travers une série d'exercices de plus en plus complexes ; le transfert de propriété passe du code "qui n'est pas le mien" à "en partie le mien", à mesure que les élèves gagnent en confiance en étendant les fonctionnalités du code.

5. **Make** (Créer)

En utilisant les concepts du programme qu'ils ont maintenant compris et adaptés, les élèves conçoivent un nouveau programme qui utilise les mêmes structures mais qui résout un nouveau problème.

Les cinq étapes de PRIMM



2. Exemple

Exercice

On définit la fonction suivante, qui prend en paramètre un tableau d'entiers non vide :

```
def mystere(tableau):
    resultat = 0
    for element in tableau:
        resultat = resultat + element
    return resultat
```

1.a [P] Quelle est la valeur de `mystere([7, 3, 8, 12])` ?

1.b [R] Exécuter le programme pour vérifier.

Visualiser l'exécution sur Python tutor

2. Quelles questions poser pour la partie Investigation ?

- Comment appelle-t-on le type de boucle présente dans la fonction `mystere` ?
- Comment appelle-t-on le type de parcours réalisé dans cette boucle ?
- Comment appelle-t-on le type de l'instruction présente à la ligne 2 ?
- Que se passerait-il si on supprimait la ligne 2, lorsqu'on appelle la fonction ?
- Parmi les variables présentes dans la fonction, l'une est qualifiée de *variable de cumul* ou *accumulateur*. Laquelle ? Pourquoi ?
- Que représente la valeur de `mystere(tableau)` lorsque tableau contient des entiers ?
- Donner un nom adapté à cette fonction

Suite de l'exercice :

3.a [Mo] Écrire une fonction `somme` qui prend en paramètre un tableau d'entiers non vide et qui renvoie la somme de ces entiers, en réalisant un parcours par indice du tableau.

On pourra s'aider de ce puzzle en ligne.

3.b [Mo] Écrire une fonction qui prend en paramètre un tableau d'entiers non vide et qui renvoie la moyenne de ces entiers.

3. Block Model

Le modèle des blocs

Le modèle des blocs a été créé par Carsten Schulte en 2008. Il s'agit d'un outil pour comprendre et catégoriser, selon deux axes, différents aspects de la compréhension d'un programme.

Ce modèle est très utile pour enrichir le questionnement de la phase I de PRIMM (Investigation). Il permet notamment de vérifier que le questionnement ne se limite pas à quelques aspects, mais développe bien chez l'élève une compréhension du programme dans ses différentes dimensions.

Premier axe : quatre niveaux

[A] Les atomes : les mots-clés, les symboles, la syntaxe, ou une seule ligne de code.

[B] Les blocs : des petits morceaux de code qui exécutent une tâche, par exemple des lignes simples, des boucles ou des fonctions.

[R] Les relations : les liens entre les différents blocs, comme les appels de fonction, les valeurs de retour, l'ordre d'exécution.

[M] La macro-structure : le programme dans son ensemble.

Deuxième axe : trois dimensions

[T] Le texte statique, avec sa syntaxe, sa structure.

[P] Le programme, vu comme objet dynamique, avec des comportements qui peuvent différer lors de son exécution, en fonction des entrées.

[F] La fonction du programme : l'objectif du programme (ou d'un élément) et l'interprétation que l'on peut en faire.

Douze zones

Quatre niveaux, analysables chacun dans chacune des trois dimensions :

le modèle des blocs comprend donc **12 zones** permettant à l'enseignant de cibler différentes activités visant la **compréhension** d'un programme.

	Texte (T)	**Programme (P)**	**Fonction (F)**
Atomes (A)			
Blocs (B)			
Relations (R)			
Macro-structure (M)			

Exemples : quelle classification pour les questions posées lors de l'exercice précédent ?

- [B-T] Comment appelle-t-on le type de boucle présente dans la fonction `mystere` ?
- [A-T] Comment appelle-t-on le type de parcours réalisé dans cette boucle ?
- [A-T] Comment appelle-t-on le type de l'instruction présente à la ligne 2 ?
- Que se passerait-il si on supprimait la ligne 2 ?
- [R-P] Parmi les variables présentes dans la fonction, l'une est qualifiée de *variable de cumul* ou *accumulateur*. Laquelle ? Pourquoi ?
- [M-F] Que représente la valeur de `mystere(tableau)` lorsque tableau contient des entiers ?
- [M-F] Donner un nom adapté à cette fonction

Quelques questions proposées par les différents groupes

- [A-T] Donner le type des différentes variables.
- [B-P] Quelles sont les différentes valeurs prises par la variable `element` lors de l'exécution de `mystere([7, 3, 8, 12])`?
- [A-T] Quel est le type de l'instruction ligne 2 ?
- [R-P] Que se passe-t-il si on supprime la ligne 2 lors de l'appel de la fonction ?
- [A-F] Renommer la variable résultat
- [A-T] Que se passe-t-il si on indente la ligne 5 à droite à gauche ?
- [R ou M-P] Que se passe-t-il si on indente la ligne 5 à gauche lors de l'exécution de `mystere([7, 3, 8, 12])`?
- [M-P] Que se passerait-il avec un tableau vide ?
- [M-P] Que se passerait-il avec un tableau de chaîne de caractères ?

- [B-P] Quel est le nombre d'itérations dans la boucle pour l'exécution de `mystere([7, 3, 8, 12])`?
- [M-F] A quoi sert la fonction `mystere`?
- [M-F] Donner une spécification de cette fonction.

Exercices

Exercice 1. On considère le programme suivant :

```
def mystere1(tab):
    resultat = {}
    for element in tab:
        if element in resultat:
            resultat[element] = resultat[element] + 1
        else:
            resultat[element] = 1
    return resultat

def gagnants(tab):
    resultat = mystere1(tab)
    maxi = 0
    for cle in resultat:
        if ... > maxi:
            maxi = ...
    liste_finale = [cle for cle in resultat if ... == maxi]
    return liste_finale

votes = ['A', 'A', 'A', 'B', 'C', 'B', 'C', 'B', 'C', 'B']
print(gagnants(votes))
```

d'après l'épreuve pratique n°1, 2023, exercice 2.

Quelles questions poser sur ce programme, dans l'esprit de la partie Investigation du modèle PRIMM ?

Quelques questions proposées par les différents groupes

- [A-T] Quel est le type de la variable `resultat` ?
- [A-T] Combien a-t-on défini de fonctions ?
- [A-T] ou [R-T] ligne 4, que représente `element` pour `resultat` ?
- [A-P] Quel est le type de la variable `cle`, ligne 13, lors de l'exécution du programme ?
- [A-P] Peut-on initialiser la variable `maxi` avec autre chose que 0 ?
- [A-F] A quoi sert la ligne 7 ?
- [B-T] Quel est le type de construction ligne 16 ?
- [B-T] À quelles lignes commence et se termine la définition de `mystere1` ?
- [B-P] Quelle doit être la propriété de `element` ligne 4 pour vérifier la condition ?

- [B-P] Trouver une valeur de `tab` pour que l'appel `mystere1(tab)` n'exécute jamais la ligne 5 ?
- [B-P] Quelle est la valeur de `mystere1(['x', 'y', 'x'])` ?
- [B-P] Quelle est la valeur de `mystere1(['A', 'A', 'A'])` ?
- [B-F] Quelle est la finalité des lignes 12 à 15 ? Les compléter.
- [B-F] Renommer la fonction `mystere1` avec un nom explicite.
- [R-T] Quel type de parcours est utilisé ligne 13 ?
- [R-T] Quel est le type de la variable `resultat` ligne 11 ?
- [R-P] Quelle est l'utilité de l'appel `mystere` ligne 12 ?
- [R-P] Quel est le type de la variable `resultat` ligne 13 ?
- [M-B] Que vaut la variable `resultat`, ligne 11, lors de l'appel ligne 20 ?
- [M-P] Quelle est la valeur de la variable `maxi` ligne 16 lors de l'exécution du programme ?

- [M-P] Quel est le type de `gagnants(votes)` ?
- [M-F] Quelle est l'utilité de la fonction `gagnants` ?
- [M-F] Écrire une spécification de la fonction `gagnants`.

Quelques questions possibles

1. [A-T] Identifier le type de la variable `resultat` dans `mystere1`.

2.a [A-P] Quelle est la valeur de `mystere1(['A', 'A', 'B', 'C', 'B'])`?

2.b [A-P] Suivre l'état des variables lors de l'évaluation de l'expression `mystere1(['A', 'A', 'B', 'C', 'B'])`

nom de la variable	valeur de la variable
`tab`	`['A', 'A', 'B', 'C', 'B']`
`resultat`	... à compléter
`element`	`'A'` ... à compléter

3. [A-F] Que représente l'expression `resultat[element]` ligne 5 et quel est le rôle de l'instruction de cette ligne 5 ?

4. [B-F] Écrire la spécification (docstring) de la fonction `mystere1`

5. [R-F] Choisir un nom explicite pour la fonction `mystere1`

6. [B-T] Y a-t-il une boucle dans le corps de la fonction `gagnants` ? Si oui, préciser son type et indiquer les numéros des lignes du code la composant.

7. [B-P] A quoi sert le bloc composé des lignes 14 et 15 ? Les compléter.

8.a [R-F] Que représente la variable `maxi` ligne 16 ?

8.b [R-P] Quelle est la valeur de la variable `maxi` ligne 16, suite à l'appel de la fonction `gagnants` ligne 20 ?

8.c [R-T] Identifier tous les appels de fonctions lorsqu'on exécute le programme complet.

9. [R-P] Identifier des entrées permettant de tester toutes les branches des deux fonctions `mystere1` et `gagnants`. Réaliser les tests.

10. [M-F] Décrire ce que réalise ce programme.

Exercice 2 : extrait de l'exercice 3 sujet 0 A 2024

On donne le programme `p1`

```
tab_itineraires = []
def cherche_itineraires(G, start, end, chaine=[]):
    chaine = chaine + [start]
    if start == end:
        return chaine
    for u in G[start]:
        if u not in chaine:
            nchemin = cherche_itineraires(G, u, end, chaine)
            if len(nchemin) != 0:
                tab_itineraires.append(nchemin)
    return []

def itineraires_court(G, dep, arr):
    cherche_itineraires(G, dep, arr)
    tab_court = ...
    mini = float('inf')
    for v in tab_itineraires:
        if len(v) <= ... :
            mini = ...
    for v in tab_itineraires:
        if len(v) == mini:
            tab_court.append(...)
    return tab_court
```

La fonction `itineraires_court` prend en paramètre un graphe `G`, un sommet de départ `dep` et un sommet d'arrivée `arr`. Cette fonction renvoie une liste Python contenant tous les itinéraires pour aller de `dep` à `arr` en passant par le moins de villes possible.

1. Expliquer pourquoi la fonction `cherche_itineraires` peut être qualifiée de fonction récursive.
2. Expliquer le rôle de la fonction `cherche_itineraires` dans le programme p1.
3. Compléter la fonction `itineraires_court`.

Voici les résultats obtenus en testant dans la console la fonction `itineraires_court` deux fois de suite (sans exécuter le programme entre les deux appels à la fonction `itineraires_court`) :

exécution du programme `p1`

```
>>> itineraires_court(G2, 'A', 'E')
[['A', 'C', 'E']]
>>> itineraires_court(G2, 'A', 'F')
[['A', 'C', 'E']]
```

alors que dans le cas où le programme `p1` est de nouveau exécuté entre les 2 appels à la fonction `itineraires_court`, on obtient des résultats corrects :

exécution du programme `p1`

```
>>> itineraires_court(G2, 'A', 'E')
[['A', 'C', 'E']]
```

exécution du programme `p1`

```
>>> itineraires_court(G2, 'A', 'F')
[['A', 'B', 'I', 'F'], ['A', 'H', 'G', 'F'], ['A', 'H', 'I', 'F']]
```

4. Donner une explication au problème décrit ci-dessus. Vous pourrez vous appuyer sur les tests donnés précédemment.

Références

Références

- Sentance, S., Waite, J., & Kallia, M. (2019) **Teaching computer programming with PRIMM: a sociocultural perspective**. Computer Science Education. 29 (2–3), 136–176.
- Sentance S. (2020), **The I in PRIMM**, p. 50, Hello World 14
- Waite J., Sentance S. (2021), **Teaching programming in schools: A review of approaches and strategies**, Raspberry Pi Foundation

Pedagogy Quick Reads

- Using PRIMM to structure programming lessons
- Understanding program comprehension using the Block Model
- Improving program comprehension through Parson's Problems

Fin