Alain Troesch
Cours d'informatique, MPSI 4
Lycée Louis-Le-Grand (Paris)
Année scolaire 2014/2015

Informatique – Chapitre 3

Variables informatiques

I. Notion de variable informatique

- Concept fondamental en programmation (impérative)

I. Notion de variable informatique

- Concept fondamental en programmation (impérative)

Définition 1.1 (Variable informatique)

Une variable est la donnée de :

I. Notion de variable informatique

- Concept fondamental en programmation (impérative)

Définition 1.1 (Variable informatique)

Une variable est la donnée de :

- une localisation en mémoire, représentée par son adresse (identifiant de la variable)

I. Notion de variable informatique

- Concept fondamental en programmation (impérative)

Définition 1.1 (Variable informatique)

Une variable est la donnée de :

- une localisation en mémoire, représentée par son adresse (identifiant de la variable)
- un nom donné à la variable pour la commodité d'utilisation (appels, affectations)

I. Notion de variable informatique

- Concept fondamental en programmation (impérative)

Définition 1.1 (Variable informatique)

Une variable est la donnée de :

- une localisation en mémoire, représentée par son adresse (identifiant de la variable)
- un nom donné à la variable pour la commodité d'utilisation (appels, affectations)

- Ainsi, une variable est à voir comme une correspondance entre un nom et un emplacement en mémoire.

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.
- Déchiffrage selon le type

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.
- Déchiffrage selon le type

Définition 1.3 (Affectation)

L'affectation est l'action de définir le contenu d'une variable.

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.
- Déchiffrage selon le type

Définition 1.3 (Affectation)

L'affectation est l'action de définir le contenu d'une variable.

- Syntaxe de l'affectation en Python : `x = 3`

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.
- Déchiffrage selon le type

Définition 1.3 (Affectation)

L'affectation est l'action de définir le contenu d'une variable.

- Syntaxe de l'affectation en Python : `x = 3`
- Non commutativité de l'égalité d'affectation

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.
- Déchiffrage selon le type

Définition 1.3 (Affectation)

L'affectation est l'action de définir le contenu d'une variable.

- Syntaxe de l'affectation en Python : `x = 3`
- Non commutativité de l'égalité d'affectation
- Ne pas confondre avec l'égalité de comparaison

Définition 1.2 (Contenu d'une variable)

- Le contenu d'une variable est l'objet stockée à l'emplacement mémoire réservé.
- Déchiffrage selon le type

Définition 1.3 (Affectation)

L'affectation est l'action de définir le contenu d'une variable.

- Syntaxe de l'affectation en Python : `x = 3`
- Non commutativité de l'égalité d'affectation
- Ne pas confondre avec l'égalité de comparaison

Définition 1.4 (Lecture, ou appel)

La lecture, ou l'appel d'une variable est le fait d'aller récuperer en mémoire le contenu d'une variable.

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.6 (Type d'une variable)

Le type représente la nature du contenu

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.6 (Type d'une variable)

Le type représente la nature du contenu

- Le type détermine la taille à réserver en mémoire

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.6 (Type d'une variable)

Le type représente la nature du contenu

- Le type détermine la taille à réserver en mémoire
- Le type détermine le codage.

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.6 (Type d'une variable)

Le type représente la nature du contenu

- Le type détermine la taille à réserver en mémoire
- Le type détermine le codage.

Remarque 1.7 (Déclaration de variables)

- Certains langages imposent de déclarer les variables : allocation d'une place en mémoire.

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.6 (Type d'une variable)

Le type représente la nature du contenu

- Le type détermine la taille à réserver en mémoire
- Le type détermine le codage.

Remarque 1.7 (Déclaration de variables)

- Certains langages imposent de déclarer les variables : allocation d'une place en mémoire.
- Certains langages dispensent l'utilisateur de cette tâche.

Définition 1.5 (État d'une variable)

L'état momentanée d'une variable est la donnée de cette variable (donc son nom et son adresse) et de son contenu.

Définition 1.6 (Type d'une variable)

Le type représente la nature du contenu

- Le type détermine la taille à réserver en mémoire
- Le type détermine le codage.

Remarque 1.7 (Déclaration de variables)

- Certains langages imposent de déclarer les variables : allocation d'une place en mémoire.
- Certains langages dispensent l'utilisateur de cette tâche.
- Simulations en Python.

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée
- Typage dynamique : déclaration automatique.

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée
- Typage dynamique : déclaration automatique.

- Dans certains langages, le type d'une variable peut changer.

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée
- Typage dynamique : déclaration automatique.

- Dans certains langages, le type d'une variable peut changer.
- Simulations en Python.

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée
- Typage dynamique : déclaration automatique.

- Dans certains langages, le type d'une variable peut changer.
- Simulations en Python.

Définition 1.9 (Typage faible)

Typage faible (langage faiblement typé) : une variable peut changer de type au cours de son existence.

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée
- Typage dynamique : déclaration automatique.

- Dans certains langages, le type d'une variable peut changer.
- Simulations en Python.

Définition 1.9 (Typage faible)

Typage faible (langage faiblement typé) : une variable peut changer de type au cours de son existence.

- Python est faiblement typé, à typage dynamique.

Définition 1.8 (Typage dynamique, typage statique)

- Typage statique : déclaration imposée
- Typage dynamique : déclaration automatique.

- Dans certains langages, le type d'une variable peut changer.
- Simulations en Python.

Définition 1.9 (Typage faible)

Typage faible (langage faiblement typé) : une variable peut changer de type au cours de son existence.

- Python est faiblement typé, à typage dynamique.
- Le changement de type se fait avec changement d'adresse, au cours d'une affectation (simulation)

II. Structures de données

Définition 2.1 (Structure de données, attribut)

- Structure de données : organisation de la mémoire en vue de stocker un ensemble de données sous une certaine forme

II. Structures de données

Définition 2.1 (Structure de données, attribut)

- Structure de données : organisation de la mémoire en vue de stocker un ensemble de données sous une certaine forme
- Les données sont appelées attributs

II. Structures de données

Définition 2.1 (Structure de données, attribut)

- Structure de données : organisation de la mémoire en vue de stocker un ensemble de données sous une certaine forme
- Les données sont appelées attributs

- La structure de données détermine la façon dont les attributs sont rangés les uns par rapport aux autres, et la façon d'y accéder.

II. Structures de données

Définition 2.1 (Structure de données, attribut)

- Structure de données : organisation de la mémoire en vue de stocker un ensemble de données sous une certaine forme
- Les données sont appelées attributs

- La structure de données détermine la façon dont les attributs sont rangés les uns par rapport aux autres, et la façon d'y accéder.
- Une structure de donnée est définie par une *classe*, ayant un type et des méthodes.

II. Structures de données

Définition 2.1 (Structure de données, attribut)

- Structure de données : organisation de la mémoire en vue de stocker un ensemble de données sous une certaine forme
- Les données sont appelées attributs

- La structure de données détermine la façon dont les attributs sont rangés les uns par rapport aux autres, et la façon d'y accéder.
- Une structure de donnée est définie par une *classe*, ayant un type et des méthodes.

Définition 2.2 (Instantiation)

Une instantiation d'une classe est la création d'une variable dont le type est celui de la classe.

Définition 2.3 (Structure de tableau)

- Un tableau (statique) informatique est une structure de données séquentielle.

Définition 2.3 (Structure de tableau)

- Un tableau (statique) informatique est une structure de données séquentielle.
- Les attributs doivent être de même type.

Définition 2.3 (Structure de tableau)

- Un tableau (statique) informatique est une structure de données séquentielle.
- Les attributs doivent être de même type.
- Taille fixe.

Définition 2.3 (Structure de tableau)

- Un tableau (statique) informatique est une structure de données séquentielle.
- Les attributs doivent être de même type.
- Taille fixe.

- Stockage contiguë en mémoire.

Définition 2.3 (Structure de tableau)

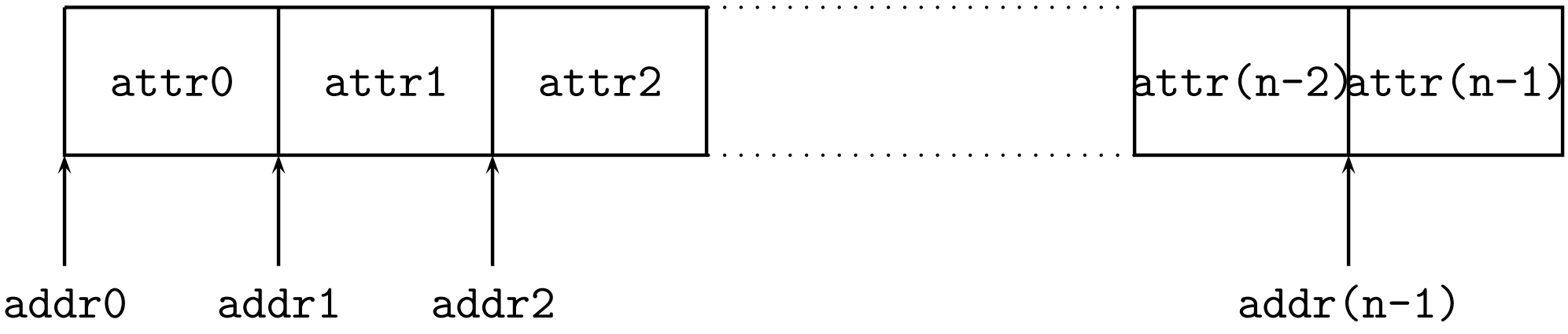
- Un tableau (statique) informatique est une structure de données séquentielle.
- Les attributs doivent être de même type.
- Taille fixe.

- Stockage contiguë en mémoire.
- Taille fixe pour chaque attribut, permettant un accès direct.

Définition 2.3 (Structure de tableau)

- Un tableau (statique) informatique est une structure de données séquentielle.
- Les attributs doivent être de même type.
- Taille fixe.

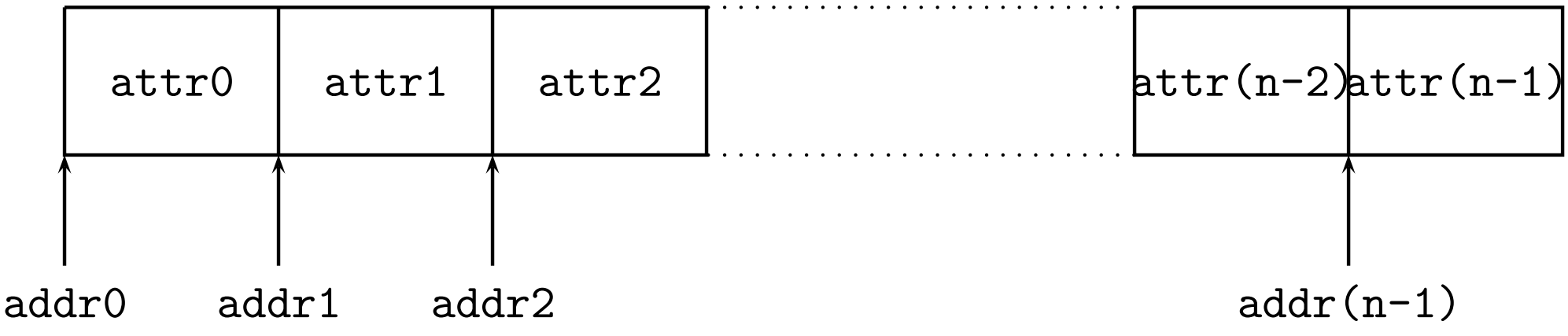
- Stockage contiguë en mémoire.
- Taille fixe pour chaque attribut, permettant un accès direct.



Définition 2.3 (Structure de tableau)

- Un tableau (statique) informatique est une structure de données séquentielle.
- Les attributs doivent être de même type.
- Taille fixe.

- Stockage contiguë en mémoire.
- Taille fixe pour chaque attribut, permettant un accès direct.



addr(i)= addr(0) + Ni

Proposition 2.4 (Complexité des op. élém. sur un tableau)

(i) Accès à un élément : temps constant $O(1)$

Proposition 2.4 (Complexité des op. élém. sur un tableau)

(i) Accès à un élément : temps constant $O(1)$

(ii) Recherche d'un élément : $O(n)$ (temps au plus linéaire par rapport à la taille du tableau)

Proposition 2.4 (Complexité des op. élém. sur un tableau)

(i) Accès à un élément : temps constant $O(1)$

(ii) Recherche d'un élément : $O(n)$ (temps au plus linéaire par rapport à la taille du tableau)

(iii) Insertion d'un élément : $O(n)$

Proposition 2.4 (Complexité des op. élém. sur un tableau)

(i) Accès à un élément : temps constant $O(1)$

(ii) Recherche d'un élément : $O(n)$ (temps au plus linéaire par rapport à la taille du tableau)

(iii) Insertion d'un élément : $O(n)$

(iv) Suppression d'un élément : $O(n)$

Proposition 2.4 (Complexité des op. élém. sur un tableau)

(i) Accès à un élément : temps constant $O(1)$

(ii) Recherche d'un élément : $O(n)$ (temps au plus linéaire par rapport à la taille du tableau)

(iii) Insertion d'un élément : $O(n)$

(iv) Suppression d'un élément : $O(n)$

(v) Recherche de la longueur : temps constant $O(1)$

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable
- Données de même type.

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable
- Données de même type.

Définition 2.6 (Capacité, taille)

- Capacité d'un tableau : nombre de cases momentanément disponibles en mémoire pour le tableau

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable
- Données de même type.

Définition 2.6 (Capacité, taille)

- Capacité d'un tableau : nombre de cases momentanément disponibles en mémoire pour le tableau
- Taille : nombre de cases réellement utilisées.

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable
- Données de même type.

Définition 2.6 (Capacité, taille)

- Capacité d'un tableau : nombre de cases momentanément disponibles en mémoire pour le tableau
- Taille : nombre de cases réellement utilisées.

- La capacité double en cas de besoin.

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable
- Données de même type.

Définition 2.6 (Capacité, taille)

- Capacité d'un tableau : nombre de cases momentanément disponibles en mémoire pour le tableau
- Taille : nombre de cases réellement utilisées.

- La capacité double en cas de besoin.
- Ceci nécessite copie ailleurs (besoin de cases contiguës)

Définition 2.5 (Tableaux dynamiques)

- Tableau dynamique : structure de données séquentielle de taille variable
- Données de même type.

Définition 2.6 (Capacité, taille)

- Capacité d'un tableau : nombre de cases momentanément disponibles en mémoire pour le tableau
- Taille : nombre de cases réellement utilisées.

- La capacité double en cas de besoin.
- Ceci nécessite copie ailleurs (besoin de cases contiguës)
- On attribue plus de cases que nécessaire pour éviter les copies trop fréquentes.

Proposition 2.7 (Complexité des op. élém. sur un tableau dynamique)

(i) Accès à un élément : $O(1)$

Proposition 2.7 (Complexité des op. élém. sur un tableau dynamique)

(i) Accès à un élément : $O(1)$

(ii) Recherche d'un élément : $O(n)$

Proposition 2.7 (Complexité des op. élém. sur un tableau dynamique)

(i) Accès à un élément : $O(1)$

(ii) Recherche d'un élément : $O(n)$

(iii) Insertion d'un élément : $O(n)$

Proposition 2.7 (Complexité des op. élém. sur un tableau dynamique)

(i) Accès à un élément : $O(1)$

(ii) Recherche d'un élément : $O(n)$

(iii) Insertion d'un élément : $O(n)$ (sauf lorsque cela induit un dépassement de capacité)

Proposition 2.7 (Complexité des op. élém. sur un tableau dynamique)

(i) Accès à un élément : $O(1)$

(ii) Recherche d'un élément : $O(n)$

(iii) Insertion d'un élément : $O(n)$ (sauf lorsque cela induit un dépassement de capacité)

(iv) Suppression d'un élément : $O(n)$

Proposition 2.7 (Complexité des op. élém. sur un tableau dynamique)

(i) Accès à un élément : $O(1)$

(ii) Recherche d'un élément : $O(n)$

(iii) Insertion d'un élément : $O(n)$ (sauf lorsque cela induit un dépassement de capacité)

(iv) Suppression d'un élément : $O(n)$

(v) Recherche de la longueur : $O(1)$ (donnée stockée)

Définition 2.8 (Listes Python)

- Une liste en Python est assimilable à un tableau dynamique.

Définition 2.8 (Listes Python)

- Une liste en Python est assimilable à un tableau dynamique.
- les attributs sont des adresses (dont de type homogène)

Définition 2.8 (Listes Python)

- Une liste en Python est assimilable à un tableau dynamique.
- les attributs sont des adresses (dont de type homogène)
- les données elles-mêmes sont stockées à ces adresses

Définition 2.8 (Listes Python)

- Une liste en Python est assimilable à un tableau dynamique.
- les attributs sont des adresses (dont de type homogène)
- les données elles-mêmes sont stockées à ces adresses
- elles peuvent être de type quelconque.

Définition 2.8 (Listes Python)

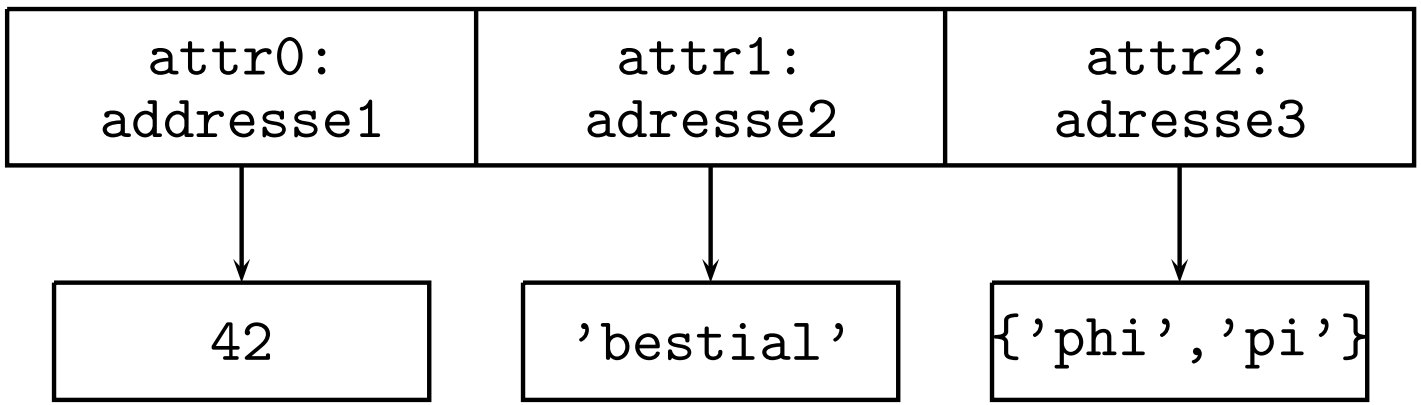
- Une liste en Python est assimilable à un tableau dynamique.
- les attributs sont des adresses (dont de type homogène)
- les données elles-mêmes sont stockées à ces adresses
- elles peuvent être de type quelconque.

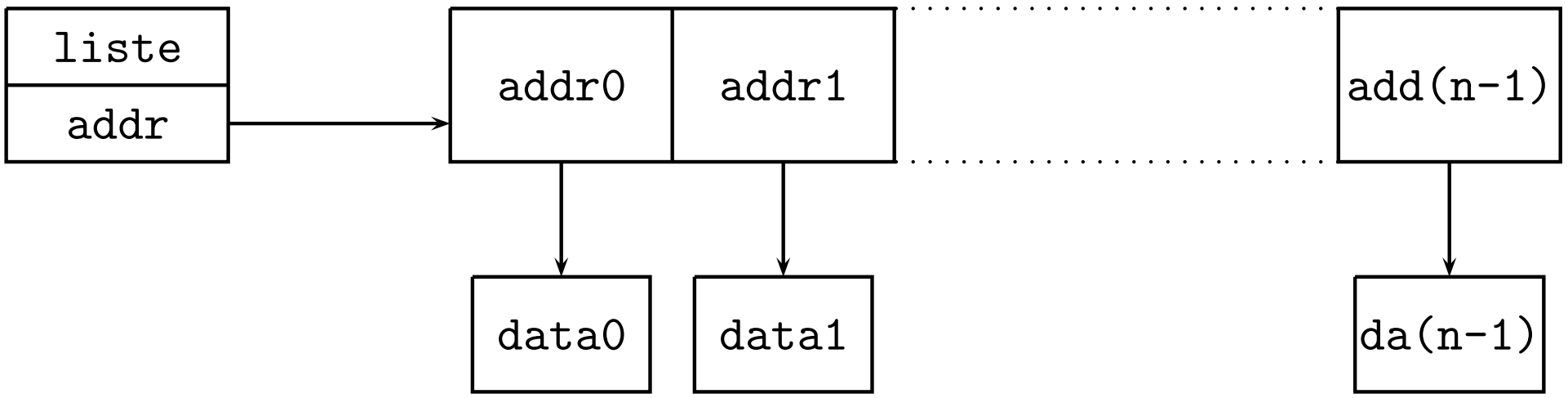
- Exemple : `liste = [42,'bestial','phi','pi']`.
  Voir identifiants et types

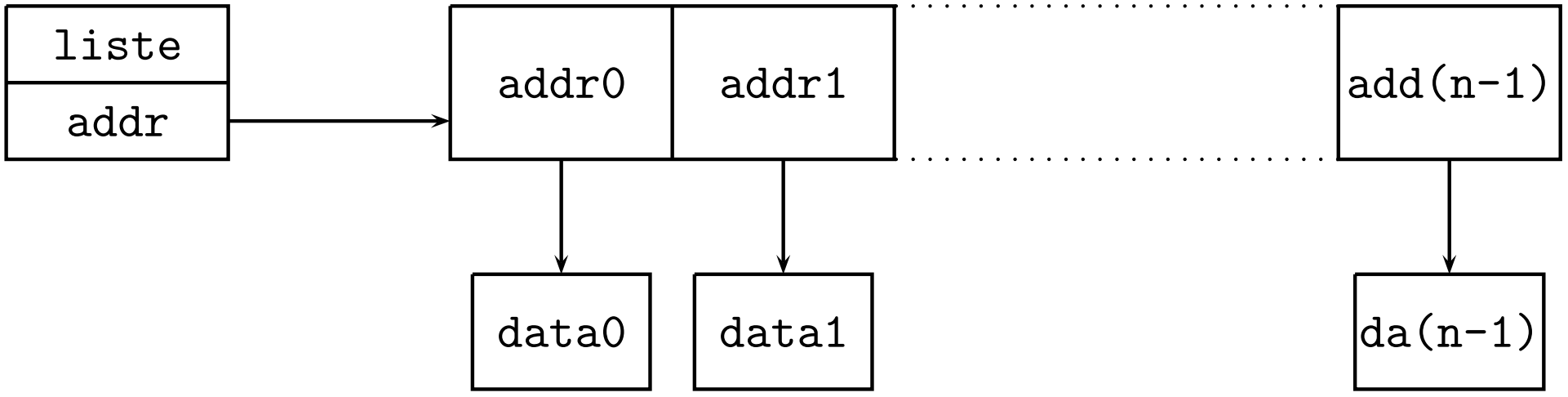
Définition 2.8 (Listes Python)

- Une liste en Python est assimilable à un tableau dynamique.
- les attributs sont des adresses (dont de type homogène)
- les données elles-mêmes sont stockées à ces adresses
- elles peuvent être de type quelconque.

- Exemple : `liste = [42,'bestial','phi','pi']`.
  Voir identifiants et types

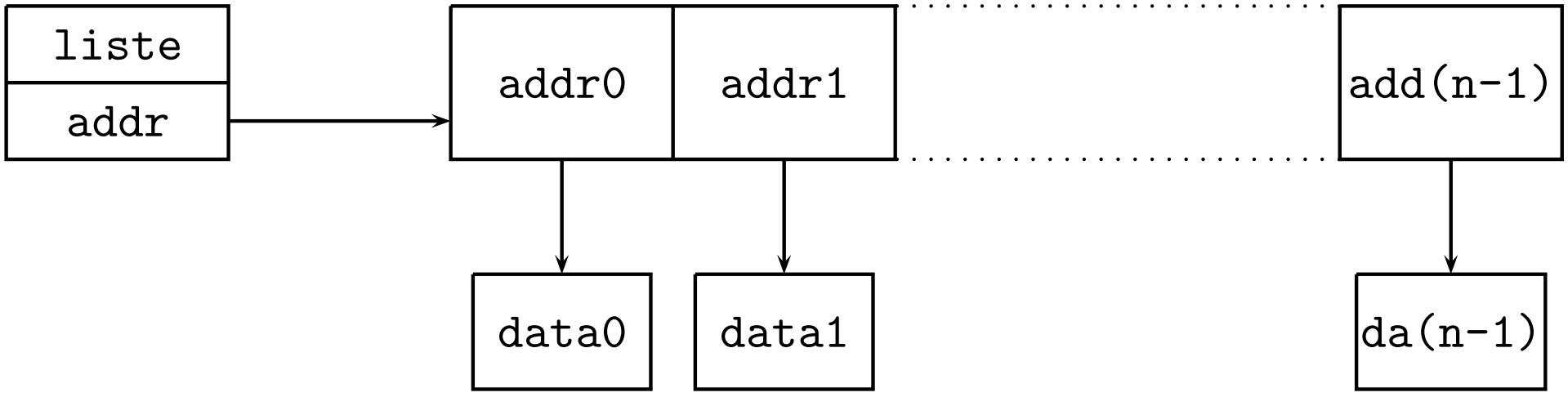



liste
addr
addr0
addr1
add(n-1)
data0
data1
da(n-1)



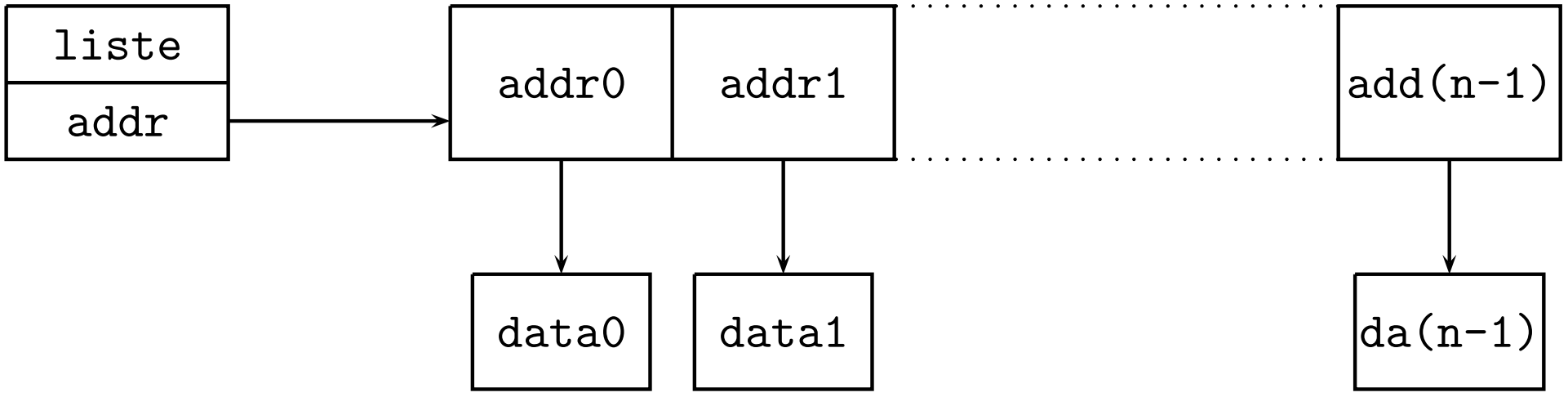
Remarque 2.9

- Ne correspond par à la notion usuelle de liste (liste chaînée)



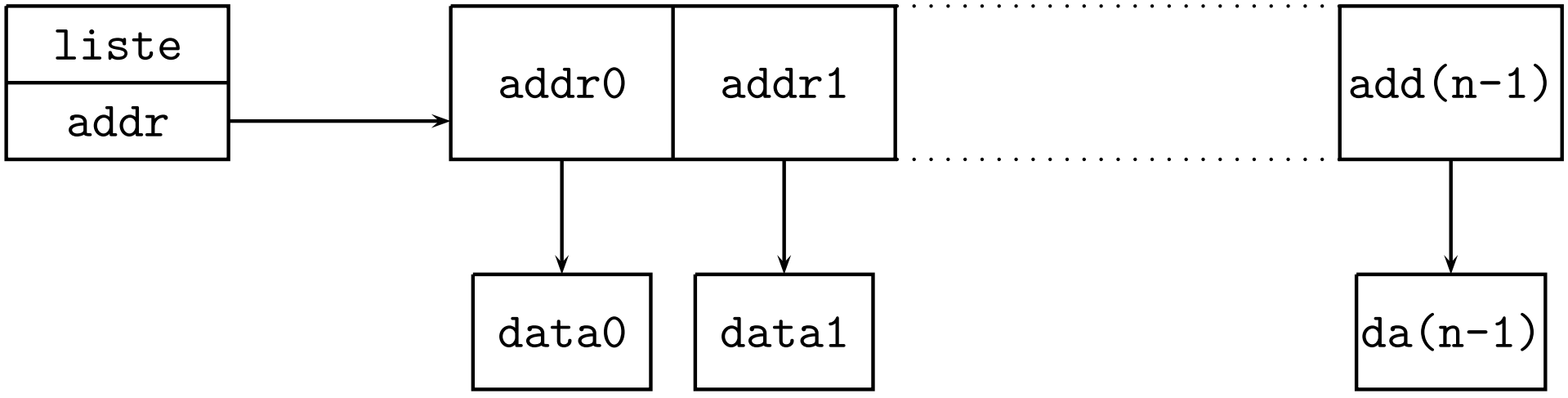
Remarque 2.9

- Ne correspond par à la notion usuelle de liste (liste chaînée)
- Listes chaînées : stockage non contiguë.



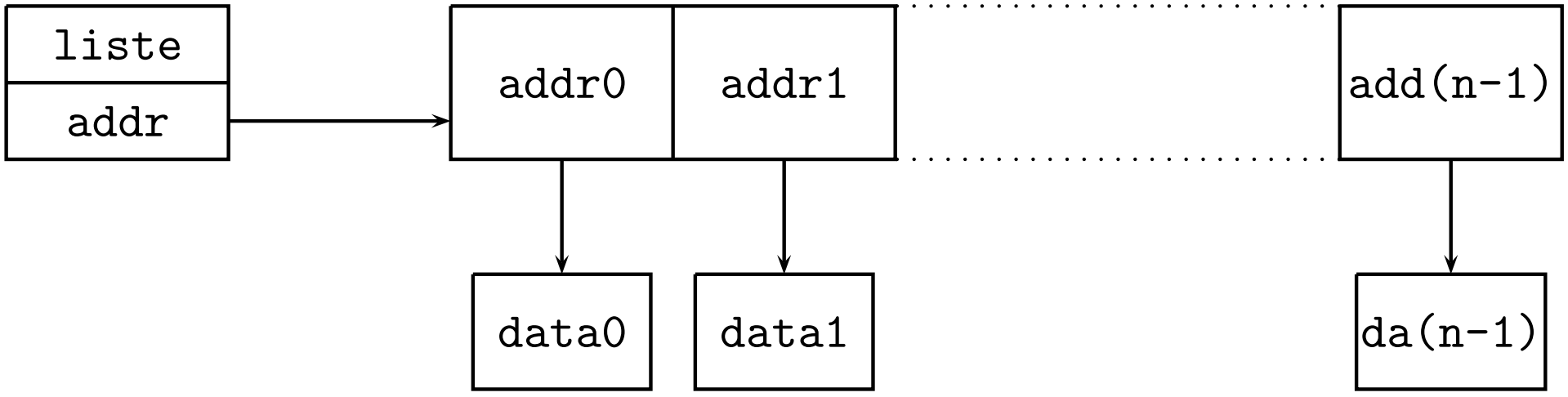
Remarque 2.9

- Ne correspond par à la notion usuelle de liste (liste chaînée)
- Listes chaînées : stockage non contiguë.
- L'adresse du suivant (et précédent) stocké avec la donnée.



Remarque 2.9

- Ne correspond par à la notion usuelle de liste (liste chaînée)
- Listes chaînées : stockage non contiguë.
- L'adresse du suivant (et précédent) stocké avec la donnée.
- Accès à un élément : Suivre les pointeurs, donc $O(n)$



Remarque 2.9

- Ne correspond par à la notion usuelle de liste (liste chaînée)
- Listes chaînées : stockage non contiguë.
- L'adresse du suivant (et précédent) stocké avec la donnée.
- Accès à un élément : Suivre les pointeurs, donc $O(n)$
- Insertion : temps constant

Définition 2.10 (Structure d'ensemble, set)

Ensemble informatique : les éléments sont stockés à une certains place en fonction de leur valeur.

Définition 2.10 (Structure d'ensemble, set)

Ensemble informatique : les éléments sont stockés à une certains place en fonction de leur valeur.

- Rangement thématique versus rangement par ordre d'arrivée.

Définition 2.10 (Structure d'ensemble, set)

Ensemble informatique : les éléments sont stockés à une certains place en fonction de leur valeur.

- Rangement thématique versus rangement par ordre d'arrivée.
- Pas d'accès à un élément donné.

Définition 2.10 (Structure d'ensemble, set)

Ensemble informatique : les éléments sont stockés à une certains place en fonction de leur valeur.

- Rangement thématique versus rangement par ordre d'arrivée.
- Pas d'accès à un élément donné.
- Recherche d'un élément (appartenance) : $O(1)$

Définition 2.10 (Structure d'ensemble, set)

Ensemble informatique : les éléments sont stockés à une certains place en fonction de leur valeur.

- Rangement thématique versus rangement par ordre d'arrivée.
- Pas d'accès à un élément donné.
- Recherche d'un élément (appartenance) : $O(1)$
- Ajout suppression : $O(1)$.

Définition 2.10 (Structure d'ensemble, set)

Ensemble informatique : les éléments sont stockés à une certains place en fonction de leur valeur.

- Rangement thématique versus rangement par ordre d'arrivée.
- Pas d'accès à un élément donné.
- Recherche d'un élément (appartenance) : $O(1)$
- Ajout suppression : $O(1)$.

En Python, type `set`, exemple `{1,2,3}`

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.
- Il est dit non mutable ou immuable sinon.

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.
- Il est dit non mutable ou immuable sinon.

Exemple 3.2 (Mutabilité des principales classes en Python)

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.
- Il est dit non mutable ou immuable sinon.

Exemple 3.2 (Mutabilité des principales classes en Python)

- Les `list` et les `set` sont mutables.

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.
- Il est dit non mutable ou immuable sinon.

Exemple 3.2 (Mutabilité des principales classes en Python)

- Les `list` et les `set` sont mutables.
- Les types numériques (`int`, `float`, `boolean`, `complex`...), les `tuples`, les `str` (chaînes de caractères) ne sont pas mutables.

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.
- Il est dit non mutable ou immuable sinon.

Exemple 3.2 (Mutabilité des principales classes en Python)

- Les `list` et les `set` sont mutables.
- Les types numériques (`int`, `float`, `boolean`, `complex`...), les `tuples`, les `str` (chaînes de caractères) ne sont pas mutables.

- Simulations à faire

III. Mutabilité ; cas des listes

Définition 3.1 (Mutabilité)

- Un type d'objets est dit mutable si les objets de ce type sont modifiables sans changement d'adresse.
- Il est dit non mutable ou immuable sinon.

Exemple 3.2 (Mutabilité des principales classes en Python)

- Les `list` et les `set` sont mutables.
- Les types numériques (`int`, `float`, `boolean`, `complex`...), les `tuples`, les `str` (chaînes de caractères) ne sont pas mutables.

- Simulations à faire
- Voir l'effet sur les adresses des attributs.

Schéma d'une modification avec changement d'adresse d'un attribut :

Schéma d'une modification avec changement d'adresse d'un attribut :

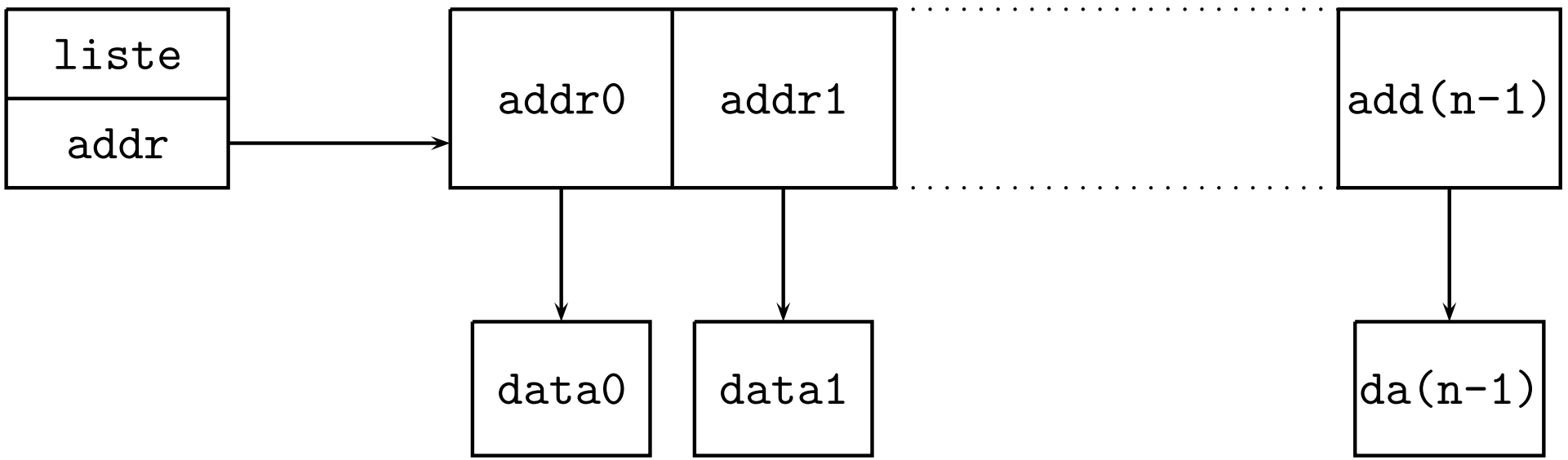
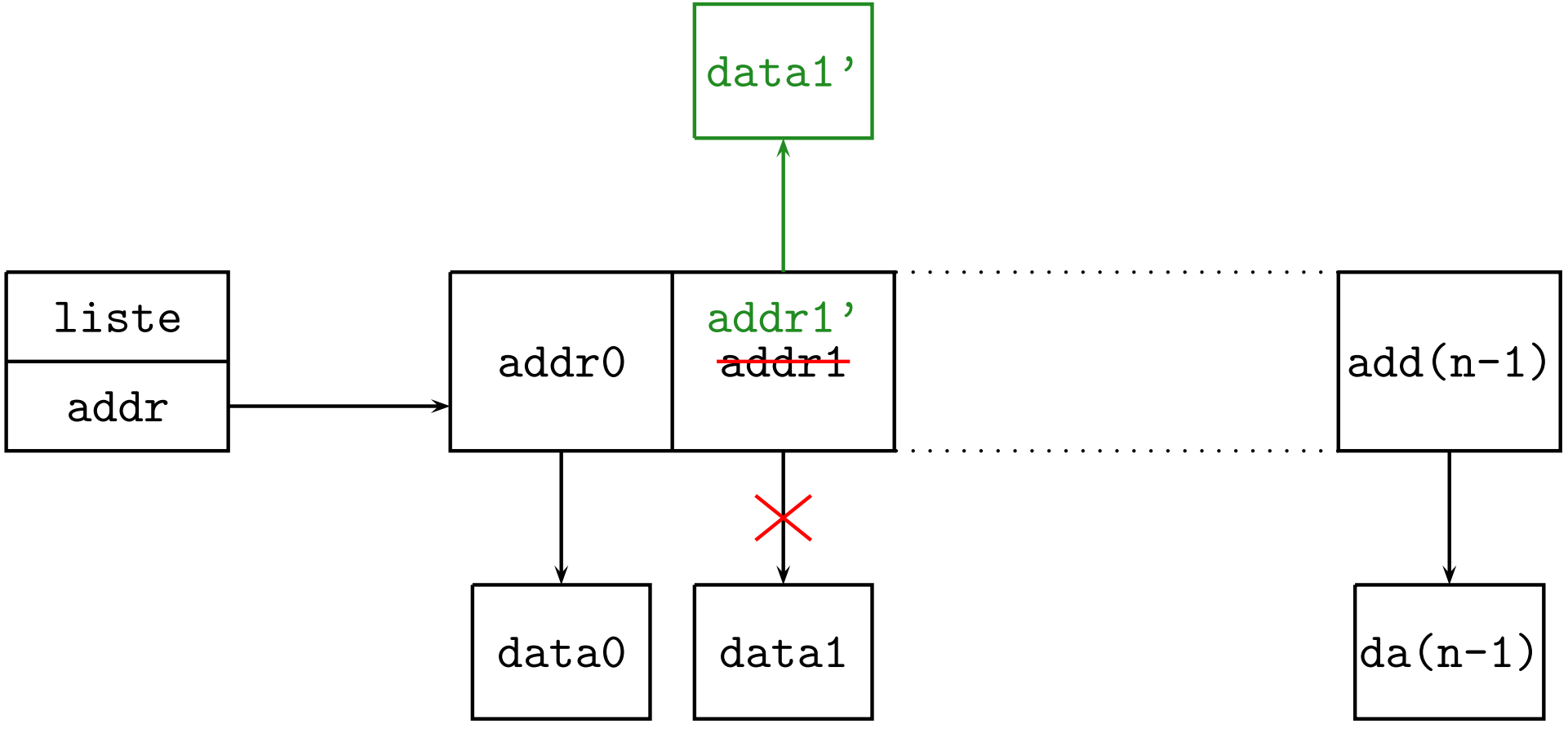


Schéma d'une modification avec changement d'adresse d'un attribut :



Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Effet sur une copie de la modification d'un type numérique

Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Effet sur une copie de la modification d'un type numérique

- Simulation à faire, avec vérification des adresses

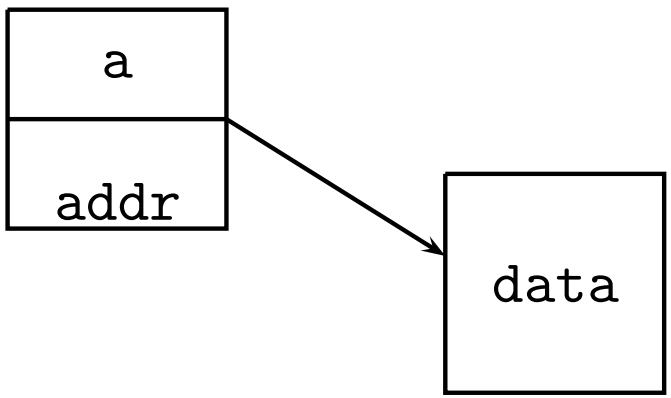
Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Effet sur une copie de la modification d'un type numérique

- Simulation à faire, avec vérification des adresses

Schéma en mémoire :



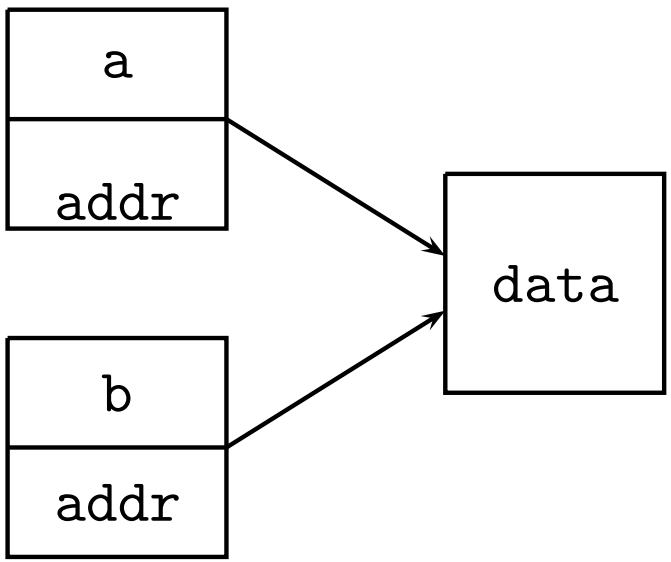
Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Effet sur une copie de la modification d'un type numérique

- Simulation à faire, avec vérification des adresses

Schéma en mémoire :



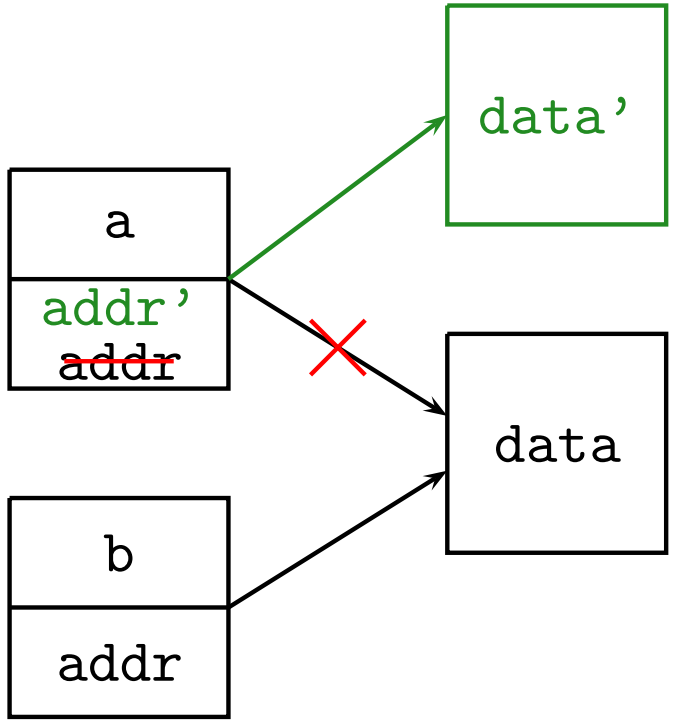
Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Effet sur une copie de la modification d'un type numérique

- Simulation à faire, avec vérification des adresses

Schéma en mémoire :



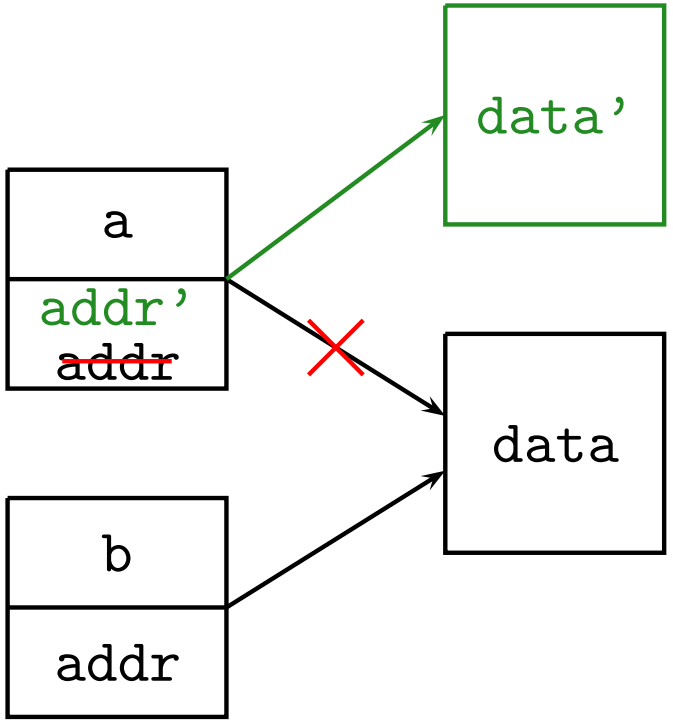
Problèmes de copies de liste

Dans un premier temps, intéressons nous à :

Effet sur une copie de la modification d'un type numérique

- Simulation à faire, avec vérification des adresses

Schéma en mémoire :



Situation générale pour les variables non mutables

Cas des copies de listes

- Simulation : modification d'un attribut :

.

Cas des copies de listes

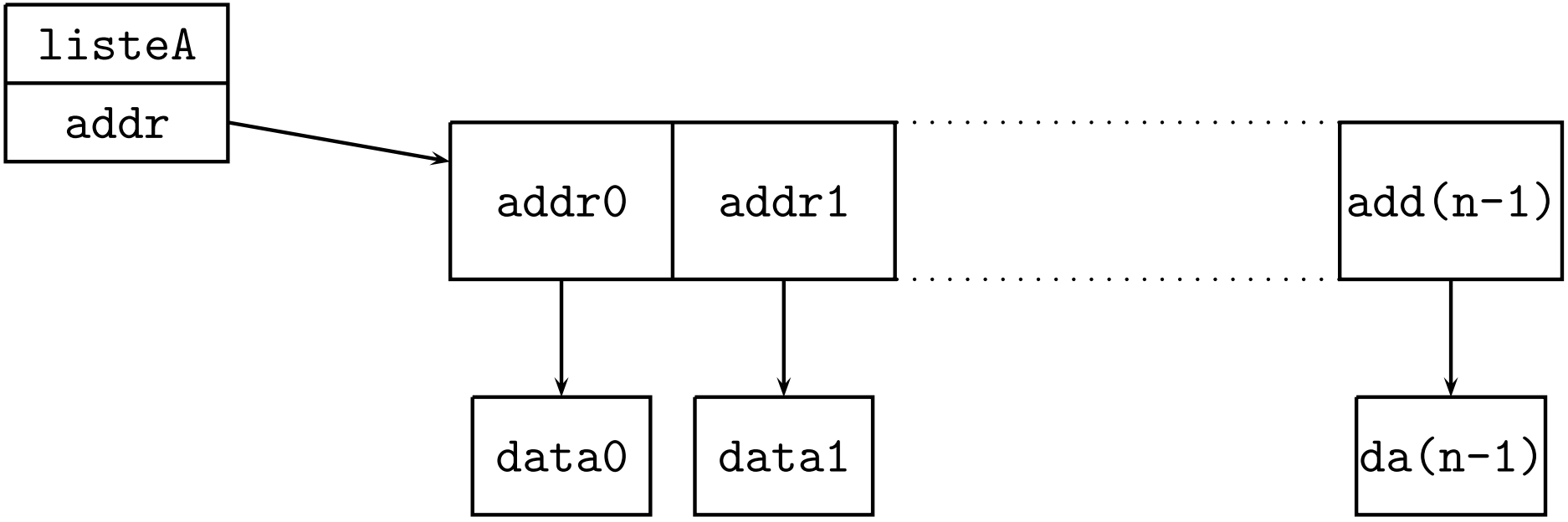
- Simulation : modification d'un attribut : cette modification est visible sur une copie.

Cas des copies de listes

- Simulation : modification d'un attribut : cette modification est visible sur une copie.
- La copie n'est pas du tout indépendante de l'original !

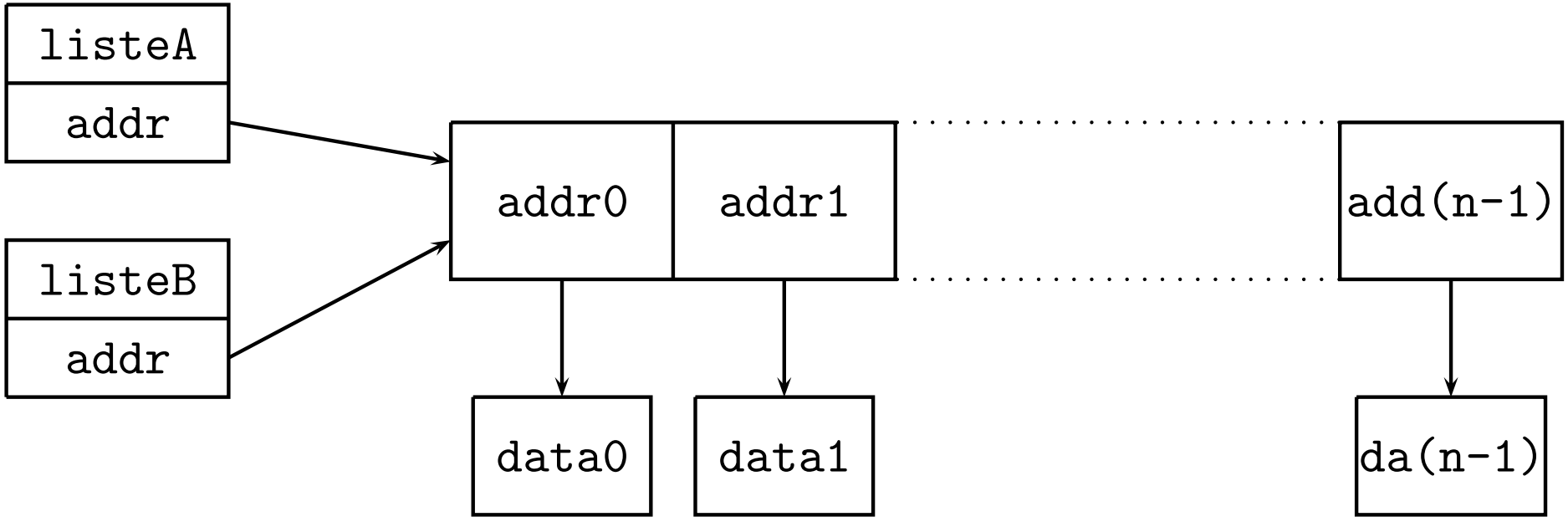
Cas des copies de listes

- Simulation : modification d'un attribut : cette modification est visible sur une copie.
- La copie n'est pas du tout indépendante de l'original !



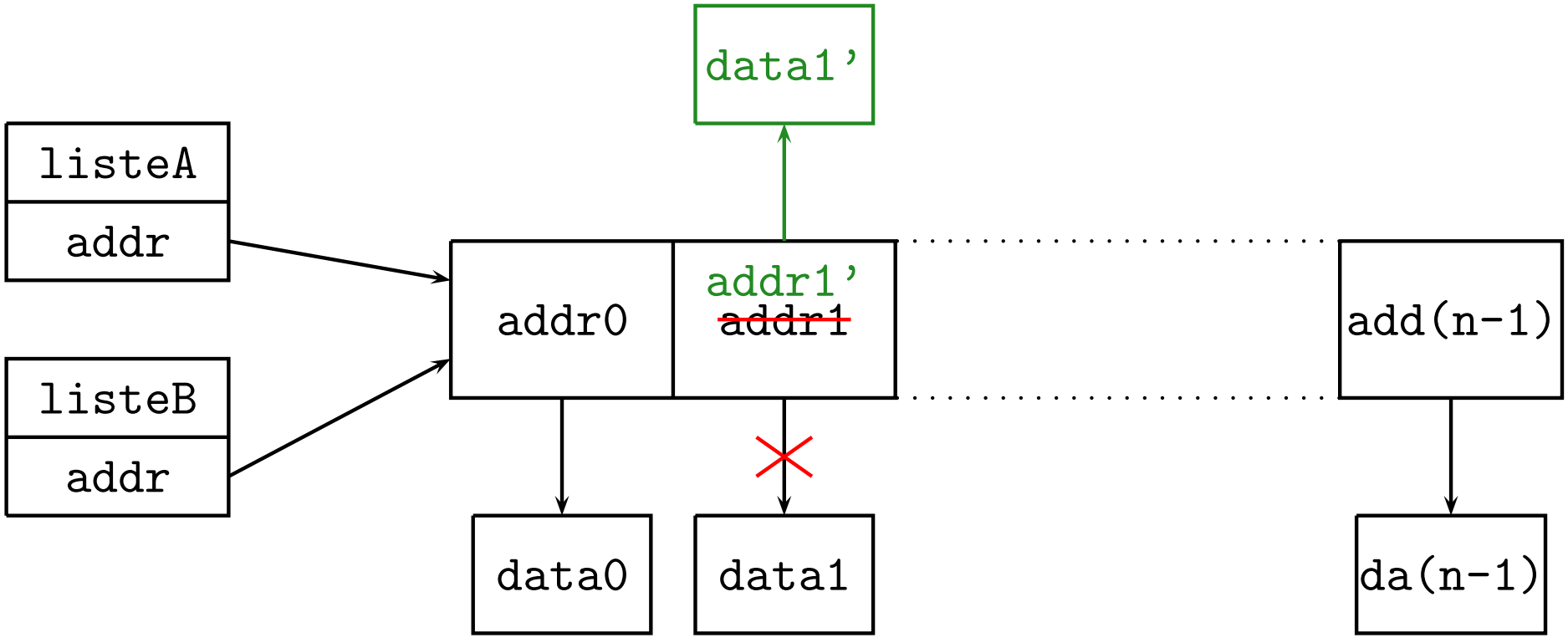
Cas des copies de listes

- Simulation : modification d'un attribut : cette modification est visible sur une copie.
- La copie n'est pas du tout indépendante de l'original !



Cas des copies de listes

- Simulation : modification d'un attribut : cette modification est visible sur une copie.
- La copie n'est pas du tout indépendante de l'original !

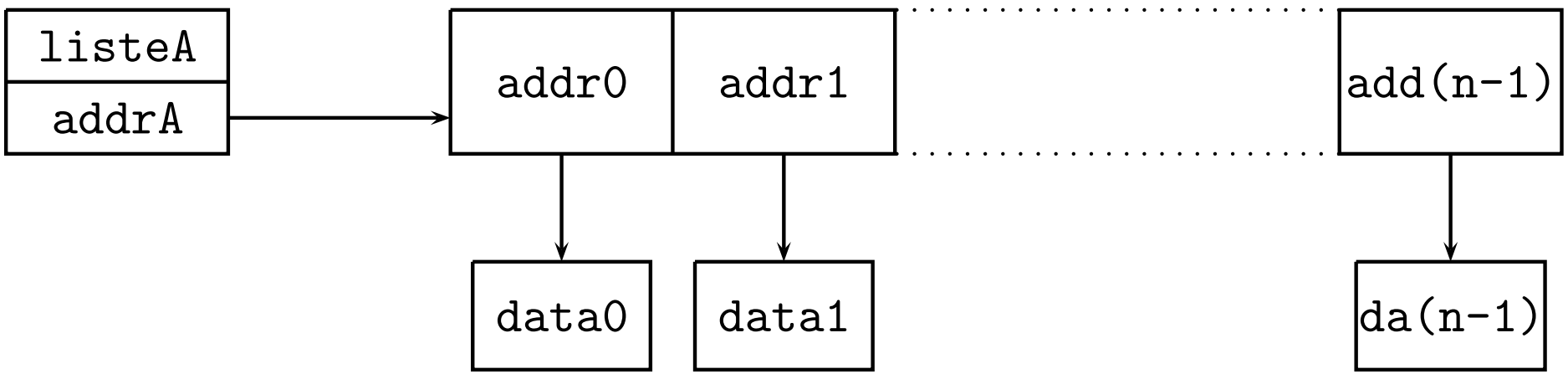


- Pour gagner en indépendance : recopier les adresses

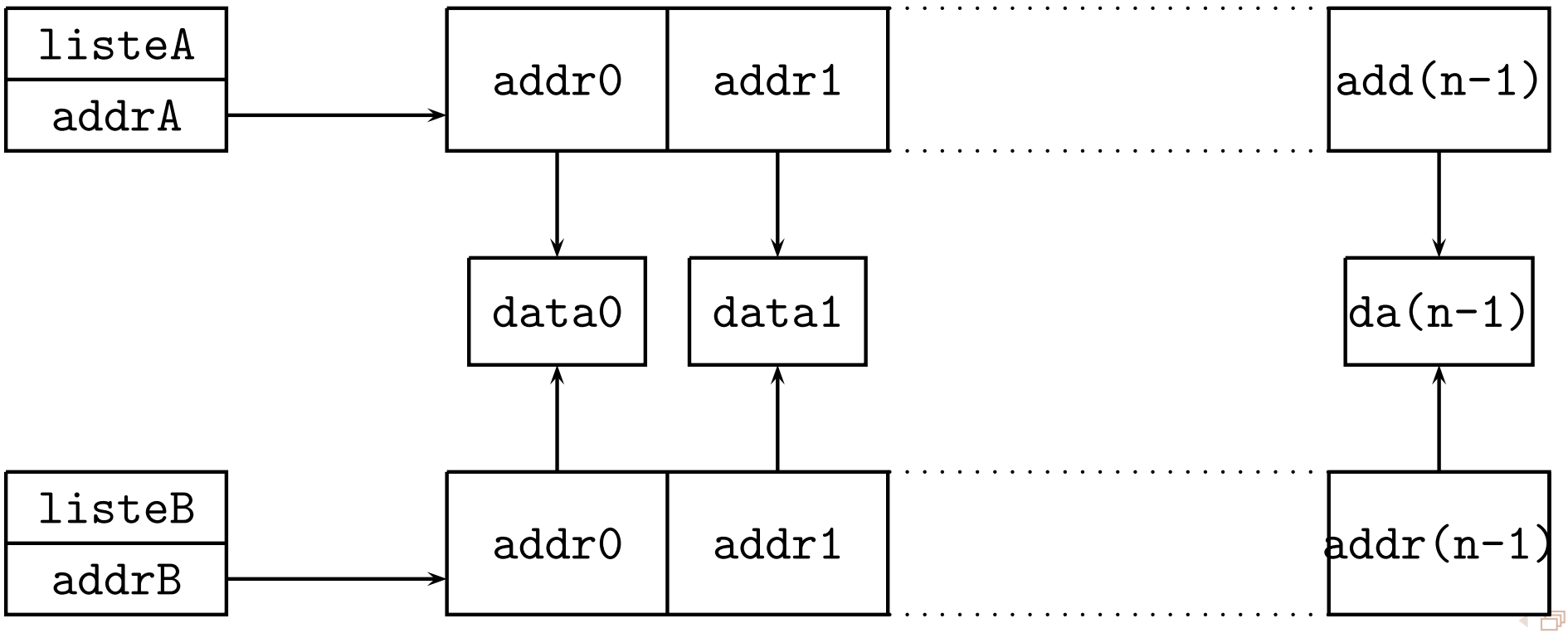
- Pour gagner en indépendance : recopier les adresses
- Cela peut se faire par un slicing.

- Pour gagner en indépendance : recopier les adresses
- Cela peut se faire par un slicing.
- Simulation. Vérification des identifications. Modification.

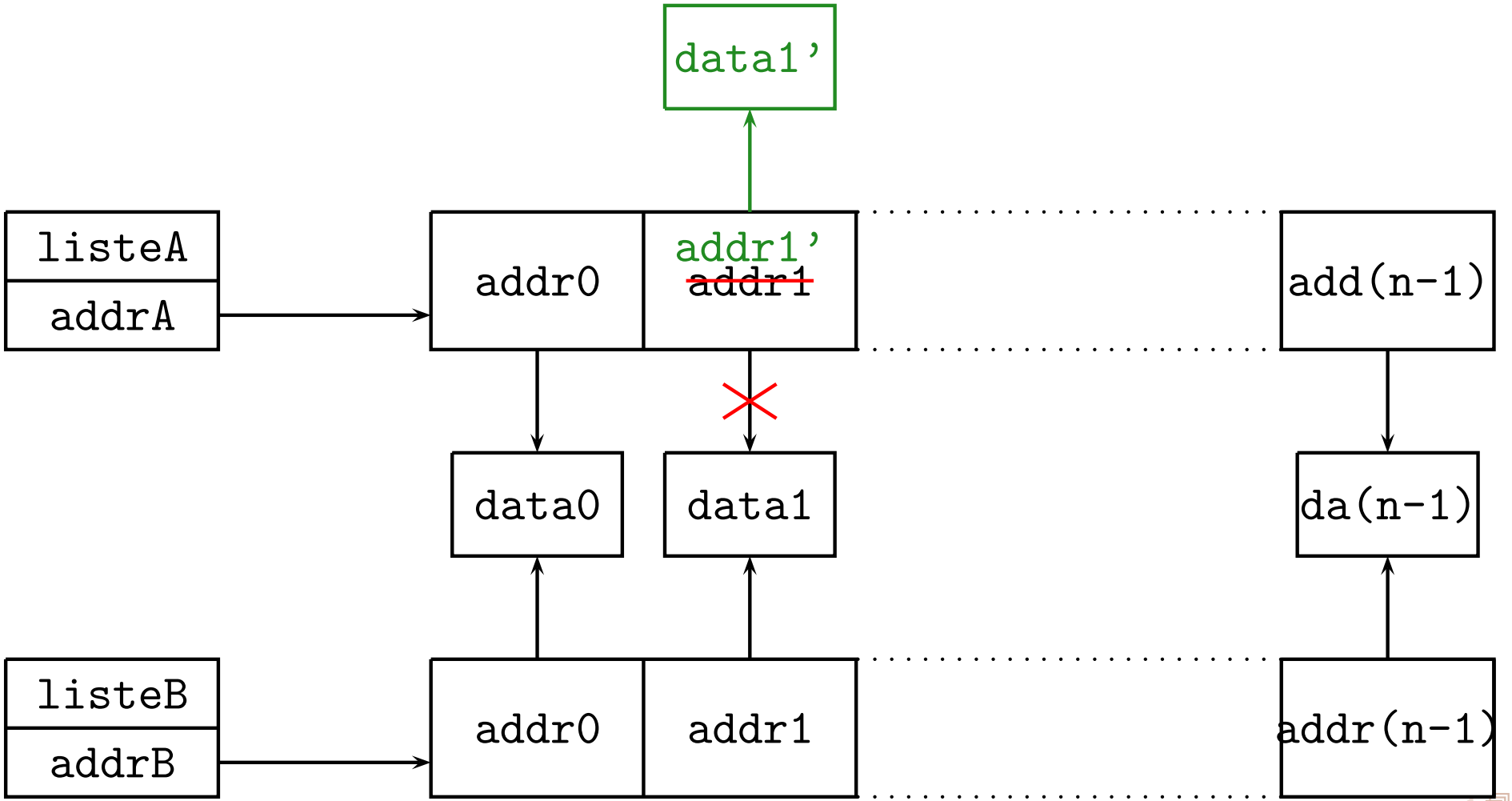
- Pour gagner en indépendance : recopier les adresses
- Cela peut se faire par un slicing.
- Simulation. Vérification des identifications. Modification.



- Pour gagner en indépendance : recopier les adresses
- Cela peut se faire par un slicing.
- Simulation. Vérification des identifications. Modification.



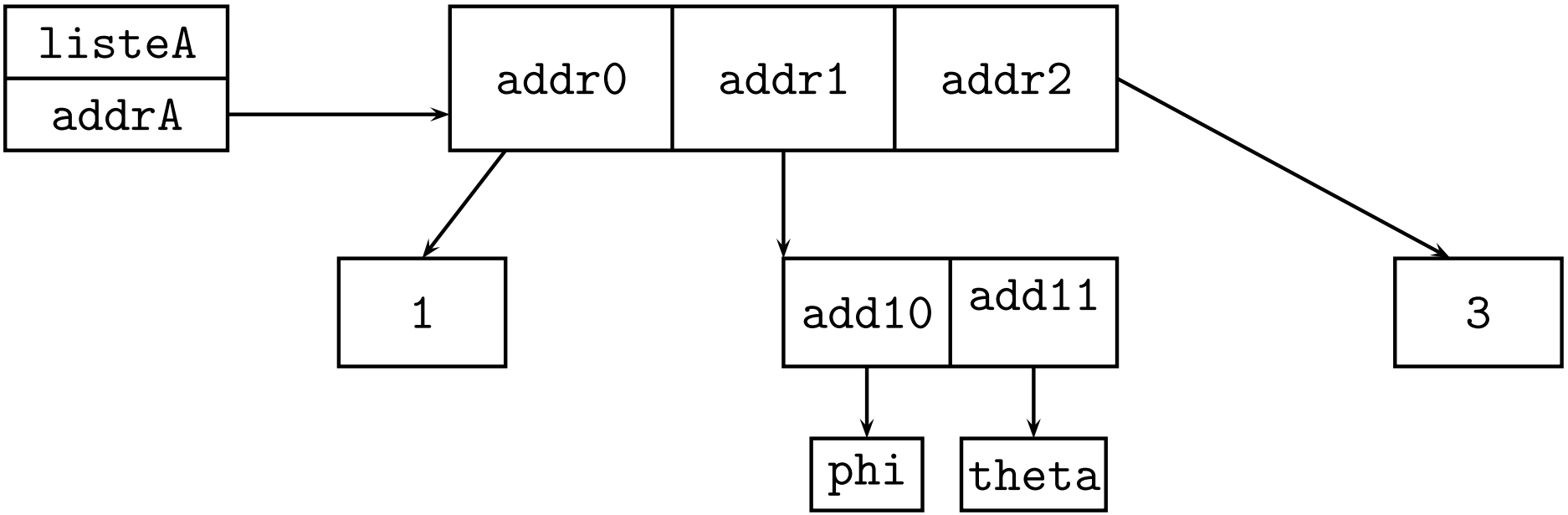
- Pour gagner en indépendance : recopier les adresses
- Cela peut se faire par un slicing.
- Simulation. Vérification des identifications. Modification.



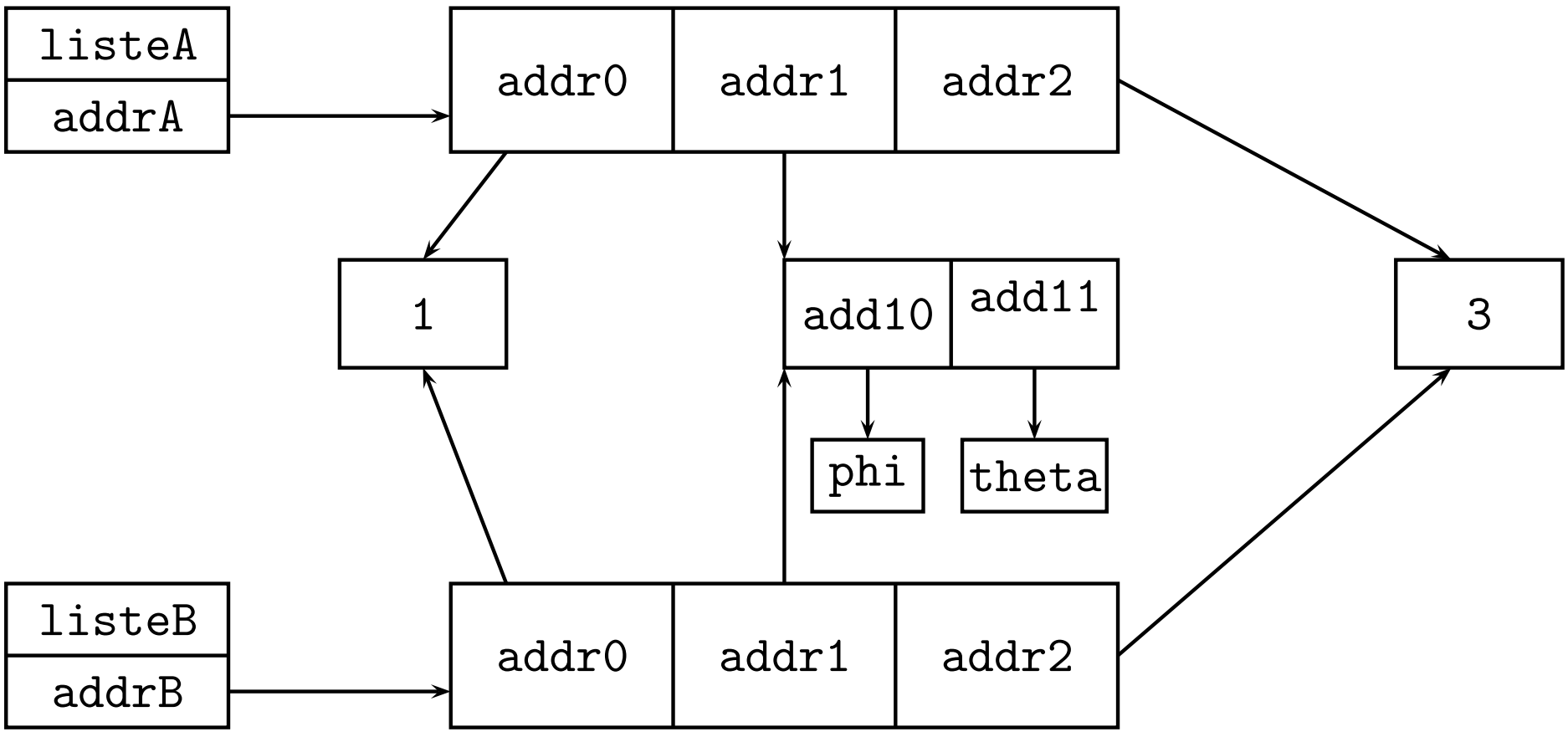
- Le problème subsiste si les modifications sont faites sans changement d'adresse, par exemple sur un objet mutable (liste de liste)

- Le problème subsiste si les modifications sont faites sans changement d'adresse, par exemple sur un objet mutable (liste de liste)
- Exemple à développer.

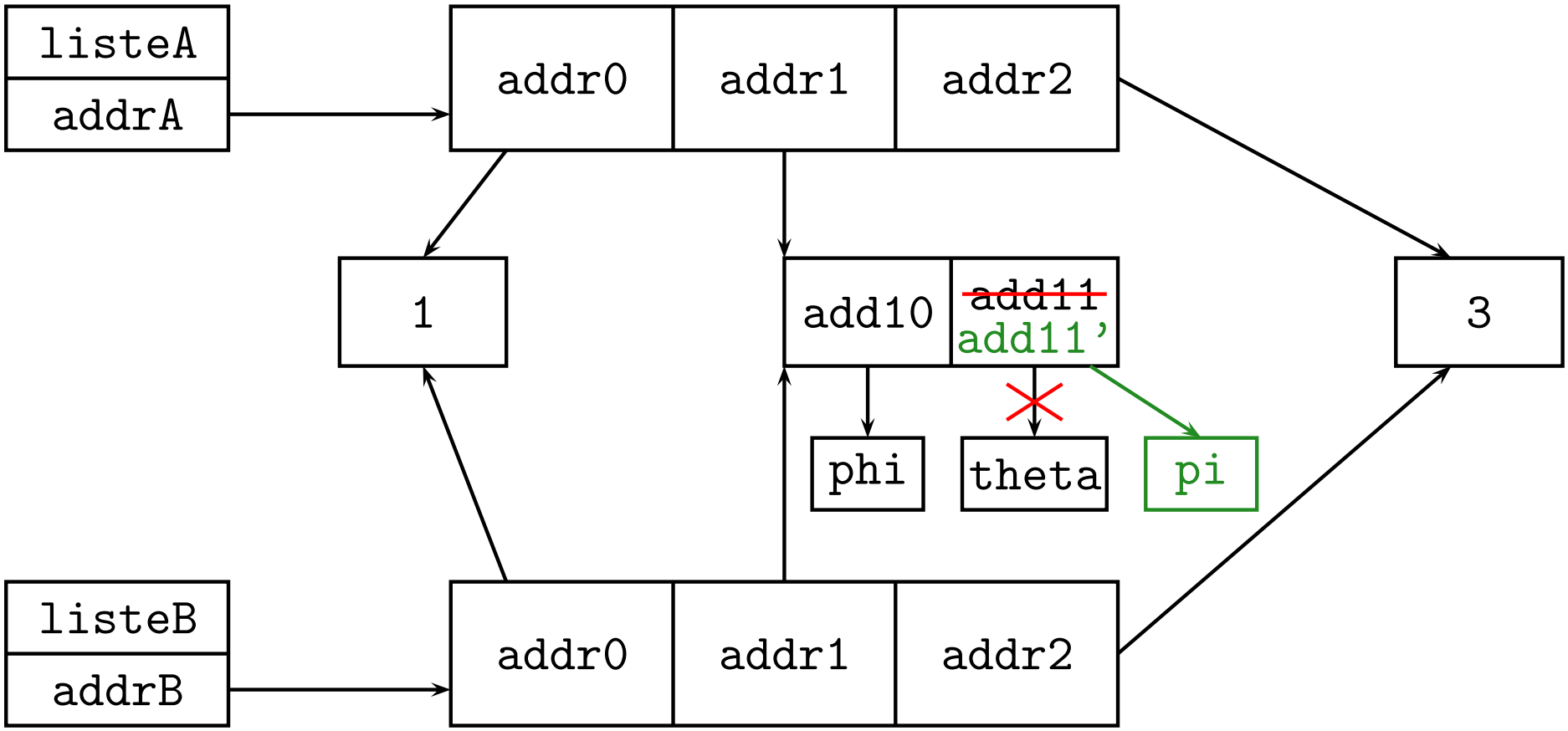
- Le problème subsiste si les modifications sont faites sans changement d'adresse, par exemple sur un objet mutable (liste de liste)
- Exemple à développer.



- Le problème subsiste si les modifications sont faites sans changement d'adresse, par exemple sur un objet mutable (liste de liste)
- Exemple à développer.



- Le problème subsiste si les modifications sont faites sans changement d'adresse, par exemple sur un objet mutable (liste de liste)
- Exemple à développer.



- Pour rendre la copie complètement indépendante : `deepcopy`

- Pour rendre la copie complètement indépendante : `deepcopy`
- Copie récursive des adresses.

- Pour rendre la copie complètement indépendante : `deepcopy`
- Copie récursive des adresses.
- Essai à faire.