Structure de PILE

Nour-Eddine Oussous, Éric Wegrzynowski

Licence ST-A, USTL - API2

12 octobre 2009

Plan du cours

Objectif

- Présenter une structure de données très utilisée en informatique
  - pile système ;
  - pile d'un évaluateur.
- Montrer son fonctionnement.
- Donner une interface d'utilisation.
- Réaliser une implémentation.

Expressions arithmétiques

- On veut réaliser un programme d'évaluation d'expressions arithmétiques.
- On se limite à des expressions arithmétiques portant sur des nombres entiers avec les quatre opérations usuelles, comme 35, $12 + 3$, $12 + 3 \times 5$, ...

Définition d'une expression arithmétique

Une *expression arithmétique* est

- soit un nombre entier ;
- soit deux *expressions arithmétiques* reliées par l'un des quatre opérateurs $+$, $-$, $\times$ et $\div$.

Évaluation des expressions

Évaluer une expression = calculer sa valeur.

- pour les expressions simples $\text{valeur}(e) = e$
  Exemple : $\text{valeur}(12) = 12$
- pour les expressions composées
  $\text{valeur}(e_1 \text{ op } e_2) = \text{valeur}(e_1) \text{ op } \text{valeur}(e_2)$

  Exemple :

$$\text{valeur}(12 + 3) = \text{valeur}(12) + \text{valeur}(3) = 12 + 3 = 15$$

Simple, n'est-ce pas ? . . .

Pas si simple !

Comment évaluer l'expression $12 + 3 \times 5$?

- $\text{valeur}(12 + 3 \times 5) = 75$?
  C'est le cas si on applique les opérateurs dès qu'ils apparaissent sans tenir compte de leurs priorités usuelles ($\times$ prioritaire devant $+$).
- $\text{valeur}(12 + 3 \times 5) = 27$?
  C'est le résultat attendu en tenant compte des priorités.

On veut la seconde évaluation !

Écriture des expressions

Pour remédier (simplement) à ce problème, trois façons d'écrire les expressions :

- expressions *infixées complètement parenthésées*

$$(12 + 3),\ (12 + (3 \times 5))$$

- expressions *préfixées* : l'opérateur est placé avant les opérandes
  $+\,12\,3$, $+\,12\,\times\,3\,5$
- expressions *postfixées* : l'opérateur est placé après les opérandes
  $12\,3\,+$, $12\,3\,5\,\times\,+$

On choisit les expressions postfixées.

Expressions postfixées

Définition

Une *expression postfixée* est

- soit un nombre ;
- soit deux *expressions postfixées* suivies d'un opérateur binaire.

Exemples

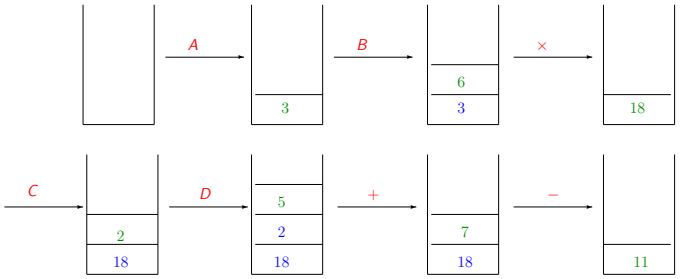
Expressions	écriture postfixée
12	12
12 + 3	$12\,3\,+$
12 + (3×5)	$12\,3\,5\,\times\,+$
$\frac{a+b}{c-d}$	$a\,b\,+\,c\,d\,-\,\div$

Intérêt de la notation postfixée

- Écriture non ambigüe ;
- sans parenthèse ;
- évaluation facile . . .
- . . . grâce à une structure nommée pile.

Exemple d'évaluation

Expression à évaluer : $A \sqcup B \sqcup \times \sqcup C \sqcup D \sqcup + \sqcup -$, avec $A = 3$, $B = 6$, $C = 2$ et $D = 5$.



Propriété invariante : *Le sommet de la pile est la valeur de la dernière expression lue*.

La structure de PILE

Définition

Une *pile* est une structure linéaire

- permettant de mémoriser un nombre variable de données (*empilement*) ;
- dont une seule est immédiatement accessible (*le sommet*) ;
- les autres ne le devenant que lorsqu'elles se retrouvent au sommet (*dépilement*).

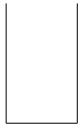
Pile = structure du dernier entré, premier sorti ou **LIFO** (**L**ast **I**n, **F**irst **O**ut).

États d'une pile

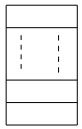
Une pile peut être

- vide : elle ne contient aucune donnée ;
- pleine : il n'est plus possible de lui ajouter un élément.

En théorie, une pile n'est jamais pleine. Mais toute implémentation des piles utilise de la mémoire, ressource disponible en quantité finie.



Pile vide.

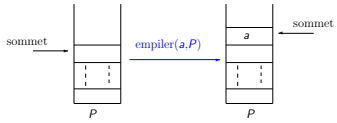


Pile pleine.

Transformations d'une pile

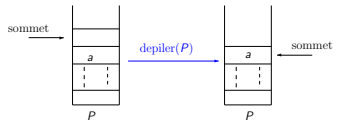
Au nombre de deux

- *empilement* : ajout d'un élément au sommet de la pile



Remarque : opération possible uniquement si la pile n'est pas pleine.

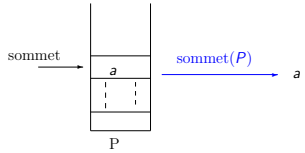
- *dépilement* : retrait de l'élément situé au sommet de la pile



Remarque : opération possible uniquement si la pile n'est pas vide.

Accès à un élément d'une pile

Seul élément accessible d'une pile : son *sommet*.



Pour accéder aux élements situés sous le sommet, nécessité de *dépiler* la pile.

Nommage des types

Dans la suite nous nommerons

- `T_PILE` les structures de type pile,
- et `T_ELEMENT` le type des données qu'elles contiennent.

Opérations primitives sur les piles

Voici les entêtes des procédures et fonctions de gestion des piles.

Listing

```
// estPileVide(P) ssi P est vide
function estPileVide(P : T_PILE) : BOOLEAN;

// sommet(P) = élément situé au sommet de P
// CU : P ne doit pas être vide
function sommet(P : T_PILE) : T_ELEMENT;

// empile x au sommet de la pile P
// CU : P ne doit pas être pleine
procedure empiler(const x : T_ELEMENT;
                  var P : T_PILE);

// dépile la pile P
// CU : P ne doit pas être vide
procedure depiler(var P : T_PILE);
```

Mais . . . le type `T_PILE` n'est pas défini.

Réalisation du type T_PILE

Représentation contigüe à l'aide

- d'un tableau `t[1..MAX]` pour stocker les élements ;
- et d'un indice `s` pour l'élément du sommet.

Lien avec les opérations primitives

- P est vide $\Leftrightarrow$ `s = 0` ;
- P est pleine $\Leftrightarrow$ `s = MAX` ;
- `sommet(P) = t[s]` ;
- `s` est incrémenté par toute opération d'empilement ;
- `s` est décrémenté par toute opération de dépilement.

Réalisation du type T_PILE(suite)

Déclaration du type `T_PILE`

Listing

```
const MAX = <à compléter>;   //taille maximale de la pile
type
  T_ELEMENT = <à compléter>; //type des éléments des piles
  T_PILE = record
     sommet : 0..MAX;
     contenu : array[1..MAX] of T_ELEMENT;
  end {T_PILE};
```

et de la constante `PILE_VIDE`

Listing

```
const
  PILE_VIDE : T_PILE = (sommet : 0);
```

Remarque : d'autres implémentations sont possibles...

Retour sur l'évaluation

Données : *e* une expression postfixée
But : évaluer *e*
Moyen : une pile *P*

```
1   tant que e non terminée faire
2     lire un élément op de e
3     si op est un nombre alors
4       empiler(valeur(op),P)
5     sinon // op est alors un opérateur
6       x := sommet(P)
7       depiler(P)
8       y := sommet(P)
9       depiler(P)
10      empiler(y op x,P)
11    fin si
12  fin tant que
13  {valeur(e) = sommet(P)}
```

Attention à l'ordre des opérandes dans l'expression passée en paramètre de la procédure `empiler` (ligne 10).