Université Bordeaux 1 LST Mention Informatique Semestre 3 2010-2011

Algorithmique 1 : Devoir Surveillé 2

Pile et files

Durée : 60mn
Sans documents

1 Pile

Exercice 2.1 *Implémentation d'une pile de caractères par listeSC_car.*
Proposer une implémentation d'une pile de caractères par `listeSC_car`, une liste simplement chaînée de caractères.
Ecrire les primitives `creerPile`, `valeur` et `empiler` dans cette implémentation.

Exercice 2.2 *Application*
Soit le type `pile de caractere` implémenté par `listeSC_car`.
Soit la fonction `g(ref Tab: tableau [1..N] de caractere): vide`
avec la définition suivante :

```
fonction g(ref Tab: tableau [1..N] de caractere): vide
var i, j: entier;
var P: pile de caractere;
debut
  creerPile(P);
  j=1;
  i=1;
  tant que Tab[i] != EOF faire
    si( Tab[i] == '#') alors
      empiler(P, '#');
    sinon
      si( pileVide(P) == vrai) alors
        Tab[j]= Tab[i];
        j++;
      sinon
        depiler(P);
      fsi
    fsi
    i++;
   fintq
   Tab[j]= EOF;
   afficherTabCar(Tab);
fin
```

On fait appel à `g` avec
`Tab={'#','#','#','a','b','a','b','a','b','a','b','#','c','d','e','f','#',EOF}`.
Le caractère `EOF` est un caractère spécial et dénote le dernier élément du tableau.
Quel est le résultat affiché par `g` ?

La fonction `afficherTabCar(val T[1..N] tableau de caractere): vide`
affiche la suite des caractères contenus dans `T`. Le caractère `EOF` n'est pas affiché.
Par exemple `afficherTabCar(Tab)` affiche `# # # a b a b a b a b # c d e f #`.

2 File

Exercice 2.3 *Implémentation d'une file d'entiers par un tableau*
Proposer une implémentation d'une file d'entiers par un tableau.
Ecrire les primitives `creerFile` et `enfiler` dans cette implémentation.

Exercice 2.4 *Application*
Soit le type `file de objet` implémenté dans un tableau.
Soit la fonction `f(val Tab: tableau de entier): entier`
avec la définition suivante :

```
fonction f(val Tab: tableau[1..N] de entier): entier;
var i, j, M: entier;
var F: file de entier;
debut
  i=1;
  j=0;
  M= 10;
  creerFile(F);
  tant que i <= N faire
    si ( fileVide(F)== vrai ) alors
      enfiler(F, Tab[i]);
      i++;
    sinon
      si ( Tab[i] > valeur(F) + M ) alors
        defiler(F);
      sinon
        si ( Tab[i] == valeur(F) + M ) alors
          afficher(valeur(F), Tab[i]);
          j++;
        fsi
        enfiler(F, Tab[i]);
        i++;
      fsi
    fsi
  ftq
  retourner j;
fin
```

La fonction `afficher(val x: entier, val y: entier): vide`
affiche les valeurs des deux entiers passés en paramètre.
On fait appel à `f` avec
`Tab={0, 2, 3, 4, 6, 7, 9, 10, 12, 13, 14, 16, 20}`.
Quel est le résultat affiché et renvoyé par `f` ?

3 File à double entrée

Alors qu'une pile n'autorise l'insertion et la suppression des éléments qu'à une extrémité, et qu'une file autorise l'insertion à une extrémité et la suppression à l'autre extrémité, une **file à double entrée** autorise l'insertion et la suppression à chaque bout. On définit le type abstrait **file_a_dbl_entree**, file à double entrée de objet, par les primitives :

```
fonction creerFile(ref  F: file_a_dbl_entree): vide;
fonction detruireFile(ref F: file_a_dbl_entree): vide;
fonction fileVide(val F: file_a_dbl_entree): booleen;

fonction premier(val F: file_a_dbl_entree): objet;
fonction enfiler_premier(ref F: file_a_dbl_entree, val X: objet): vide;
fonction defiler_premier(ref F: file_a_dbl_entree): vide;

fonction dernier(val F: file_a_dbl_entree): objet;
fonction enfiler_dernier(ref F: file_a_dbl_entree, val X: objet): vide;
fonction defiler_dernier(ref F: file_a_dbl_entree): vide;
```

Proposer une implémentation du type **file à double entrée de objets** et écrire les primitives correspondantes dans l'implémentation choisie.

ANNEXE A **Type abstrait** *pile de objet*

```
fonction valeur(val P:pile de objet):objet;
fonction pileVide(val P:pile de objet):booleen;
fonction creerPile(ref P:pile de objet): vide;
fonction empiler(ref P:pile de objet, val x:objet):vide;
fonction depiler (ref P:pile de objet):vide;
fonction detruirePile(ref P:pile de objet):vide;
```

ANNEXE B **Type abstrait** *file de objet*

```
fonction valeur(val F:file de objet):objet;
fonction fileVide(val F:file de objet):booleen;
fonction creerFile(ref F:file de objet): vide;
fonction enfiler(ref F:file de objet, val v:objet):vide;
fonction defiler(ref F:file de objet):vide;
fonction detruireFile(ref F:file de objet):vide;
```