Piles et files

1 Piles

1.1 Définition

Les primitives à vérifier sont :

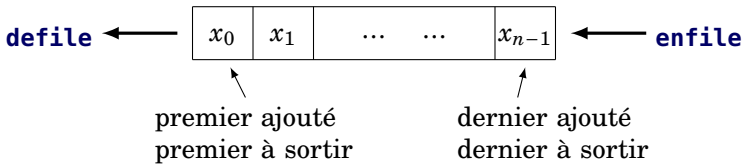
- `creer : () -> pile` créer une pile vide,
- `est_vide : pile->bool` vérifier si la pile est vide,
- `empiler : pile->element->pile/()` ajouter un élément,
- `depiler : pile-> element & pile/()` retirer l'élément le plus récent,
- `detruire : pile->()` désallouer la mémoire

1.2 Implémentation avec des maillons chaînés

1.3 Une application : textes bien parenthésés

2 Files

2.1 Définition

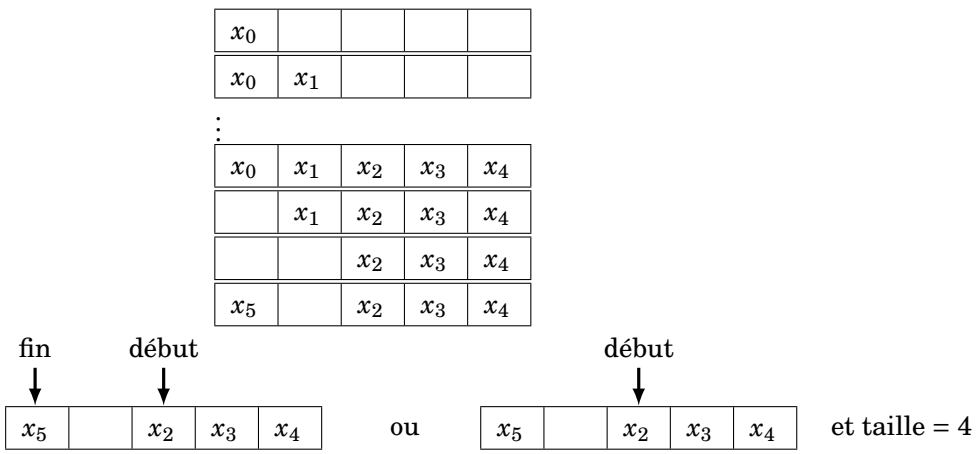


Les primitives à vérifier sont :

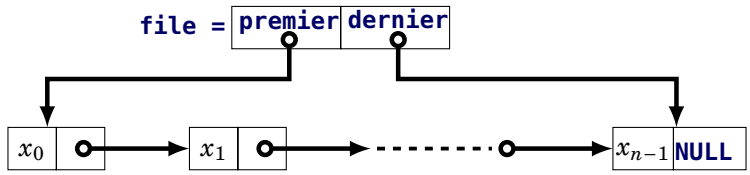
- `creer : () -> file` créer une file vide,
- `est_vide : file->bool` vérifier si la file est vide,
- `enfiler : file->element->file/()` ajouter un élément,
- `defiler : file->element & file/()` retirer l'élément le plus ancien,
- `detruire : file->()` désallouer la mémoire.

2.2 Implémentation avec un tableau (file de taille bornée)

On utilise le tableau de manière circulaire. Et on garde en mémoire où se trouvent le plus ancien élément et le plus récent.



2.3 Implémentation avec des maillons chaînés.



La file est dite vide si **premier** et **dernier** valent **NULL**.

C

```c
typedef struct file {maillon* premier;  maillon* dernier;} file;

file* creer_file() {
  file *f = malloc(sizeof(file));
  f->premier =
  f->dernier =
  return f;}

bool est_vide_file(file* f) {
  return (f->premier==NULL && f->dernier==NULL);}

void enfile(file* f, int x) {
  //Création du nouveau maillon
  maillon *p =
  if (est_vide_file(f)) {
    //Si le premier élément et le dernier élément sont le même
    f->premier =
    f->dernier =                                          }
  else{
    f->dernier->suivant =
    f->dernier =                                             }}

int defile(file *f) {
  assert(!est_vide_file(f));
  //On retire le premier maillon, on garde en mémoire la valeur qu'il contient
  maillon* p = f->premier;
  f->premier =
  int v =
  free(p);

  //Cas ou la file est désormais vide, il faut aussi modifier le pointeur dernier.
  if (f->premier==NULL) {f->dernier =
  return v;}
```

2.4 Implémentation avec deux piles