Ecole supérieure de la Statistique et de
l'Analyse de l'Information de Tunis

Algorithmique et programmation II
1ère année Ingénieur
aicha.elgolli@essai.ucar.tn

Files et Piles

1) Introduction et définition

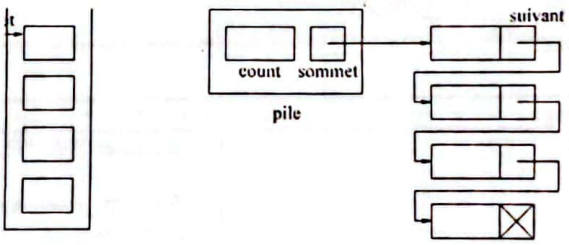
Une file est une liste linéaire pour laquelle l'insertion d'un élément se fait en fin de liste, et la suppression se fait en début de liste (first in-first out, FIFO). Par exemple, une file d'attente.

- File = collection (ensemble) d'éléments gérée en FIFO (First In First Out) - *queue*
- Pile = collection (ensemble) d'éléments gérée en LIFO (Last In First Out) - *stack*

• Utilisations

Files	Piles
systèmes d'exploitation	compilation
simulations	gestion des imbrications
autres ...	autres ...

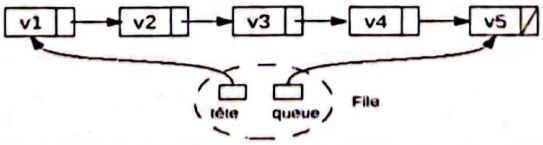
Une pile est une liste linéaire pour laquelle les opérations d'insertion et de suppression sont restreintes à une extrémité de la liste : un nouvel élément ne peut être inséré qu'en fin de liste, et l'élément supprimé ne peut être que le dernier élément inséré (last in-first out, LIFO). Par exemple, une pile d'assiettes.



Concept de Pile — Representation de la Pile

2) Opérations de base sur les files

Contrairement à une pile, on a besoin d'accéder à la fois au premier élément et au dernier élément de la file.



Algo	C
Type Elem =structure donnee : <type de données> ; suivant : pointeur sur Elem; **Fin ;**	***typedef struct element*** ***{ int donnee;*** ***struct element *suivant;*** ***} Elem;***
Type File= structure tete : pointeur sur Elem ; queue : pointeur sur Elem ; **Fin ;**	***Typedef struct file*** ***{ Elem*tete;*** ***Elem *queue;*** ***}File;***

Modèle des Files

1) **Procedure CreerFile(d/r F:File)**

initialise la file F à vide

2) **FileVide(F:File) : Booleen**
3) **Procedure enfiler(x:entier, d/r F:File)**

insère x en queue de la file F

4) **Fonction defiler(d/r F:File) : entier**

Retire et retourne l'élément en tête de la file F. **Conditions:** Défiler est défini si not FileVide.

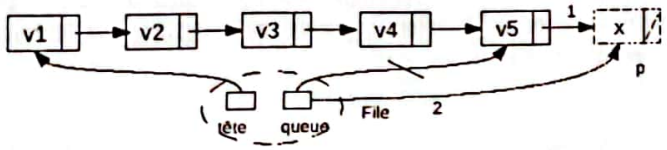
5) **procedure vider_file(d/r F : File)**

1) Création file

Algo	C
Procedure CreerFile(r F:File) **DEBUT** F.tete null ; F.queue null ; **FIN**	*void CreerFile(File* F)* *{* *F->tete=NULL ;* *F->queue=NULL ;* *}*

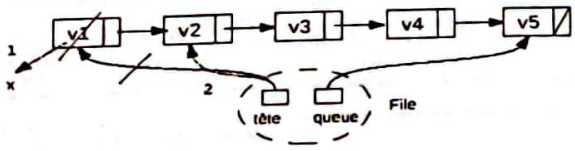
2) file vide

Algo	C
Fonction FileVide(F:File) : Booleen **DEBUT** retour(F.tete=NULL); **FIN**	*int FileVide(File F)* *{* *return(F.tete==NULL); }*



3) enfiler

Algo	C
Procedure enfiler(x:entier; d/r F:File) p: pointeur vers Elem; **DEBUT** allouer(p) ; p donnée x ; p suivant NULL ; si (FileVide(F))Alors F.tete=p ; sinon F.queue suivant=p ; FSI ; F.queue p; **FIN;**	*void enfiler(File *F, int donnee)* *{* *Elem *p_nouveau = malloc(sizeof *p_nouveau);* *if (p_nouveau != NULL)* *{ p_nouveau->suivant = NULL;* *p_nouveau->donnee = donnee;* *if (FileVide(*F))* *F->tete = p_nouveau;* *else* *F->queue->suivant=p_nouveau ;* *F->queue=p_nouveau ;* *}* *}*



4) defiler

Algo	C
Fonction defiler(d/r F:File): entier p: pointeur vers Elem; x :entier ; **DEBUT** x= -1 ; Si (not(FileVide(F))) alors	*int defiler(File *F)* *{* *int ret = -1;* */* On teste si la file n'est pas vide. */* *if (!FileVide(*F))//(F->tete != NULL)* *{*

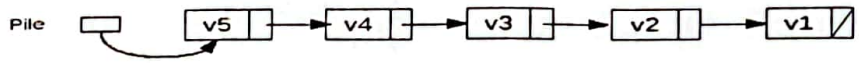
x F.tete donnée ; p F.tete ; F.tete F.tete suivant ; Libérer(p) ; Si (FileVide(F)) alors F.queue NULL ; Retour(x) ; FIN	*Elem *p_tmp = (F->tete)->suivant;* *ret = (F->tete)->donnee;* */* Effacement du premier élément. */* *free(F->tete), F->tete = NULL;* */* On fait pointer la file vers le deuxième élément. */* *F->tete = p_tmp;* *if(FileVide(*F)) F->queue=NULL;* *}* *return ret;* *}

5) vider file

Algo	C
Procedure vider_file (d/r F:File) **DEBUT** **Tant que** (not(FileVide(F))) **Faire** defiler(F) ; **FinTQ** **FIN**	**void vider_file(File *F)** **{ /* Tant que la file n'est pas vide. */** **while (! FileVide(*F))** **{ /* On défile */** **defiler(F);** **}** **}**

3) Opérations de base sur les piles

Algo	C
Type Pile =structure donnee : <type de données> ; suivant : pointeur sur Pile ; **Fin ;**	*typedef struct pile* *{ int donnee;* *struct pile*suivant;* *} Pile;*



Modèle des Piles

1) **fonction PileVide(P:Pile) : Booleen**

testent l'état de P (vide)

2) **Procedure CreerPile(d/r P:Pile)**

initialise la pile P à vide

3) **Empiler(x:entier; d/r P:Pile)**

insère x au sommet de la pile P

4) **fonction Depiler(d/r P:Pile) : entier**

retire et retourne l'élément du sommet de la pile P

Conditions: Dépiler est défini si not PileVide.

5) **procedure vider_pile(d/r P : Pile)**

Algo	C
Procedure CreerPile(d/r P:Pile) **DEBUT** P NULL ;**FIN**	*void CreerPile(Pile** P)* *{* **P=NULL ;* *}*
Algo	**C**
fonction Pile Vide(P:Pile) : Booleen debut retour(P=NULL); fin	*int PileVide(Pile* P)* *{* *return(P==NULL); }*

Algo	C
procedure Empiler(x:entier , d/r P:Pile) q: pointeur vers Pile; **DEBUT** allouer(q) ; q donnée x ; q suivant P; P q ; **FIN;**	*void empiler(Pile **P, int donnee)* *{* *Pile *p_nouveau = malloc(sizeof *p_nouveau);* *p_nouveau->suivant = *P;* *p_nouveau->donnee = donnee;* **P=p_nouveau ; }*

Algo	C
Fonction Depiler(d /r P:Pile): entier q: pointeur vers Pile; x : entier ; **DEBUT** **X -1 ;** **Si** (not Pilevide(P)) **alors** x P donnée ; q P ; P P suivant ; Libérer(q) ;FinSi Retour(x) ; **FIN**	*int depiler(Pile **P)* *{* *int ret = -1;* *if (!PileVide(*P))* *{* *Pile *temporaire = (*P)->suivant;* *ret = (*P)->donnee;* *free(P); *P = NULL;* **P= temporaire;* *}* *return ret;}*

Algo	C
Procedure vider_pile (d/r P:Pile) **DEBUT** **Tant que** (not(PileVide(P))) **Faire** depiler(F) ; **FinTQ** **FIN**	**void vider_pile(Pile **P)** **{ /* Tant que la pile n'est pas vide. */** **while (! PileVide(*P))** **{ /* On dépile */** **depiler(P);** **}** **}**

Exemple d'exécution

```
void affichage(File f){
if(FileVide(f)) printf("\nvotre file est vide!\n");
   else
   { printf("\nFile: < ");
     Elem* p=f.tete;
     while(p!=NULL){
     printf(" %d ", p->donnee);
     p=p->suivant;
     if(p!=NULL) printf(" -> ");
     }
     printf(" > \n");
   }
}
```

```
void affichagePile(Pile *p){
if(PileVide(p)) printf("\nvotre pile est vide!\n");
   else
   { printf("\nPile: < ");
     Pile *ptr=p;
     while(ptr!=NULL){
     printf(" %d ", ptr->donnee);
     ptr=ptr->suivant;
     if(ptr!=NULL) printf(" -> ");
     }
     printf(" > \n");
   }
}
```

```
void main(){
   File f;
   CreerFile(&f);
   enfiler(&f,12); enfiler(&f,11); enfiler(&f,101); enfiler(&f,78);
   printf("defilement de %d ",defiler(&f));
   affichage(f);
   vider_file(&f);   affichage(f);
   Pile *p;
   CreerPile(&p); empiler(&p,12); empiler(&p,11); empiler(&p,101); empiler(&p,78);
   printf("depilement de %d ",depiler(&p)); affichagePile(p); vider_pile(&p); affichagePile(p);}
```