

Algorithmique et Structures de Données

Akkouche Samir :

E-mail : samir.akkouche@univ-lyon1.fr

Pronost Nicolas:

E-mail : nicolas.pronost@univ-lyon1.fr

Plan du cours

1. Rappels
2. Tableaux dynamiques
 - a. Introduction
 - b. Description du module Tab_Dynamique
 - c. Mise en œuvre et complexité des opérations
3. Listes chaînées
4. Arbres binaires
5. Piles et Files

Rappel sur les tableaux

```
#include <stdio.h>
#include <stdlib.h>
```

```
double somme (const double t[ ] )
{
    int j;
    double s = 0.0; int taille = sizeof(t) ;
    cout << "Dans somme, sizeof(t) = \n " << taille;
    for(j = 0; j < taille; j++) {s = s + t[j];}
    return s;
}

int main()
{
    int i;  double s1, s2;  double tab1[6];
    double *tab2;

    for( i = 0 ; i < 6 ; i++ ) {  tab1[i] = 1.0; }
    cout << "Dans main, sizeof(tab1) = \n " << sizeof(tab1);

    s1 = somme(tab1);

    tab2 = new double[6];

    for(i = 0; i < 6; i++) {tab2[i] = 1.0;}
    cout << "Dans main, sizeof(tab2) = \n " << sizeof(tab2);
    s2 = somme(tab2);
    delete [] tab2;
    return 0;
}
```

sizeof() est une fonction?

A quel moment est exécuté sizeof()?

Affichage à l'écran :

Dans main, sizeof(tab1) = ?

Dans somme, sizeof(t) = ?

Dans main, sizeof(tab2) = ?

Que faut-il modifier dans somme ?

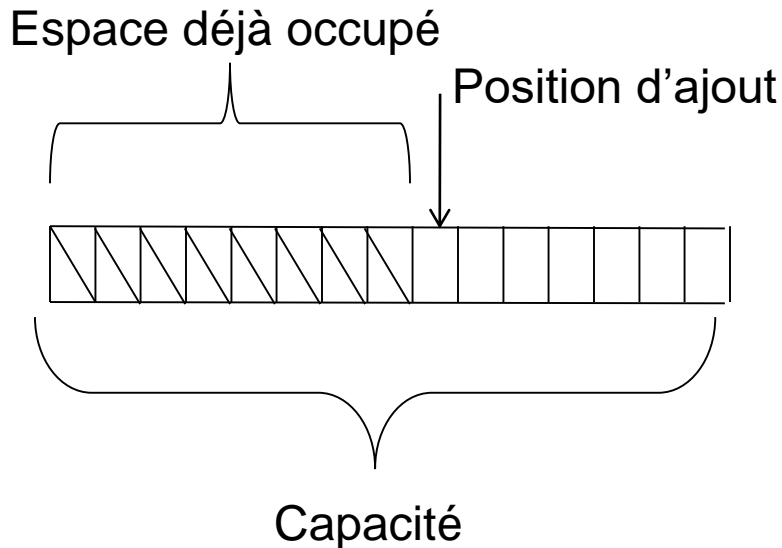
Caractéristiques d'un tableau

- Les cellules sont contigües en mémoire
- Accès direct
- Capacité fixée à l'avance :

```
float t[25]; //déclaration statique
```

```
float *t = new[25]; //déclaration dynamique
```

- En pratique, deux parties distinctes : Partie pleine et partie avec des éléments non significatifs (pour le problème traité)



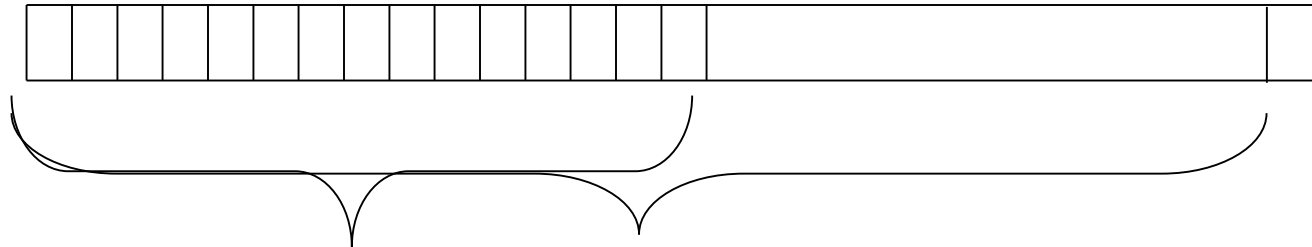
Tableaux Dynamiques

Tableau Dynamique

Type à accès direct dont on ne **S'OCCUPE PAS** de la taille.

- Il faut pouvoir augmenter la taille à loisir(et la réduire).
- Transparent pour l'utilisateur

Type TableauDynamique = structure
 adressePremierElt : pointeur sur Element
 capacite : entier
 tailleUtilisee : entier
Fin structure TableauDynamique



Capacité

module TableauDynamique

importer Element

exporter

type TableauDynamique

procedure initialiser (t : TableauDynamique)

Précondition : t non initialisé,

Postcondition : le tableau est créé avec une longueur égale à 1

Paramètre en mode donnée : aucun

Paramètre en mode résultat : t

procedure initialiser (t : TableauDynamique, taille : entier)

Précondition : t non initialisé,

Postcondition : le tableau est créé avec une longueur égale à taille

Paramètre en mode donnée : taille

Paramètre en mode résultat : t

procedure testament(t : TableauDynamique)

Précondition : t est bien initialisé,

Postcondition : la place réservée pour t est libérée

Paramètre en mode donnée-résultat : t

-----A suivre-----

implantation

finModule

module TableauDynamique (suite)

fonction tailleutilisee(t : TableauDynamique) : entier

Précondition : t bien initialisé,

Résultat : retourne le nombre effectif d'éléments dans le tableau

Paramètre en mode donnée : t

procedure ajouteElement(t : TableauDynamique, e : Element)

Précondition : t est bien initialisé

Postcondition : e est ajouté à la suite des éléments déjà présents

Paramètre en mode donnée : e

Paramètre en mode donnée-résultat : t

procedure insereElement(t : TableauDynamique, e: Element, i: entier)

Précondition : t est bien initialisé et $i < \text{tailleutilisee}$

Postcondition : e est inséré à la position i dans t

Paramètres en mode donnée : e, i

Paramètre en mode donnée-résultat : t

-----A suivre-----

implantation

finModule

module TableauDynamique (suite)

fonction accesElement_i(t : TableauDynamique, i : entier) : Element

Précondition : t est bien initialisé et $i < \text{tailleutilisée}$

Résultat : retourne la valeur à la position i dans t

Paramètres en mode donnée : i

Paramètre en mode donnée-résultat : t

procedure modifieElement_i(t : TableauDynamique, e: Element, i: entier)

Précondition : t est bien initialisé, $i < \text{tailleutilisée}$

Postcondition : modifie la valeur à la position i dans t

Paramètres en mode donnée : e, i

Paramètre en mode donnée-résultat : t

implantation

finModule

Mise en œuvre

```
#ifndef _ELEMENT_TD  
#define _ELEMENT_TD
```

```
typedef int ElementTD;
```

```
void afficheElementTD(ElementTD e);
```

```
/* Preconditions : aucune */
```

```
/* Post-conditions : affichage de e sur la sortie standard */
```

```
#endif
```

Tableau Dynamique : Mise en œuvre

Structure du fichier TableauDynamique.h

```
#ifndef __TABDYN
#define __TABDYN

#include "ElementTD.h"

class TableauDynamique {
public:
    ElementTD * adressePremierElt;
    int capacite;
    int tailleUtilisee;

    TableauDynamique ();
    TableauDynamique(int );

    ~TableauDynamique ();

    int  tailleUtilisee();
    void ajouteElement(ElementTD );
    void insereElement(ElementTD , int);

};

#endif
```

initialisation

On alloue de la place à l'initialisation

```
TableauDynamique :: TableauDynamique()
{
    capacite = 1;
    adressePremierElt= new ElementTD[1] ;
    tailleUtilisee = 0;
}
```

```
int main()
{
    TableauDynamique a;
    return 0;
}
```

pile

tailleUtilisee

0

3 962 837 630

capacite

1

3 962 837 626

adressePremierElt

87242

3 962 837 622

tas

ad[0]

87242

Méthode d'insertion dans un tableau dynamique : Doubler la taille du tableau quand plus de place

- Insérer la valeur 12



- Insérer la valeur 25



- Insérer la valeur 5



- Insérer la valeur 8



- Insérer la valeur 30



Pas de place pour insérer 25

Création d'un nouveau tableau

Recopie de l'ancien dans le nouveau

Insertion de la valeur 25

Ajout d'éléments

```
void TableauDynamique::ajouteElement(ElementTD e )
{
    int i; ElementTD *temp;
    if(tailleUtilisee == capacite)
    {
        temp = adressePremierElt;
        (adressePremierElt = new ElementTD[2*capacite];

        for( i=0;i< capacite;i++){ adressePremierElt[i]=temp[i];}

        capacite = 2*capacite;
        delete[] temp; /*On libère la place occupée par temp */
    }
    adressePremierElt[tailleUtilisee]=e;
    tailleUtilisee++;
}
```


Testament

```
TableauDynamique::~~TableauDynamique ( )  
{  
    if(adressePremierElt !=0)  
    {  
        delete[] adressePremierElt;  
        adressePremierElt = 0;  
        capacite = 0;  
        tailleUtilisee = 0;  
    }  
}
```

Etude des coûts

	Coût : tableau statique		Tableau dynamique
Ajout	Temps constant	$O(1)$?
Insertion	Temps linéaire	$O(n)$?
Suppression	Temps linéaire	$O(n)$	$O(n)$ ou ?
modification	Temps constant	$O(1)$	$O(1)$
recherche	$O(?)$		$O(?)$

Coût de l'insertion pour un tableau dynamique

- Stratégies d'insertion
 - Doubler la taille du tableau
 - Augmenter le tableau d'une taille constante

$2^0=1$ 

$2^1=2$ 

$2^2=4$ 

$2^3=8$ 

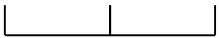
$2^4=16$ 

Etape $k \Rightarrow$ taille du tableau $= 2^k$


Coût de l'insertion pour un tableau dynamique

$$2^{p-1} < n \leq 2^p < 2n$$

$$2^0=1$$


$$2^1=2$$


$$2^2=4$$


$$2^3=8$$


$$2^4=16$$


$$\text{Coût total} = n + \sum_{k=0}^{p-1} 2^k < n + 2^p < 3n$$

- Combien de fois a-t-on doublé la taille du tableau?

Coût de l'insertion pour un tableau dynamique

$$\text{Coût amorti} = \frac{\text{Coût total}}{n} \leq \frac{3n}{n} = 3$$

Problème inverse

- Après avoir doublé la taille plusieurs fois et supprimé plusieurs éléments :
 - Le rapport $\text{tailleUtilisee} / \text{capacite}$ peut devenir petit \Rightarrow On occupe trop de places



- Solution : Si $\text{tailleUtilisee} / \text{capacite} < 1/3$ on divise la capacité par 2.
 - On alloue $\text{capacite}/2$ Elements dans le tas
 - On recopie les éléments du tableau dans la nouvelle place allouée
 - On libère l'ancienne place
- Exercice : Ecrire la procédure de suppression d'un élément