Algorithmique procédurale

Les données complexes

Equipe pédagogique

CY Tech

Bibliographie - Sitographie

- Transparents Houcine Senoussi
- Transparents Jean-Paul Forest
- Wikipedia : `https://fr.wikipedia.org/wiki/Algorithme`
- Laurence Pilard (Université de Versailles)

Retour sur l'exercice Minimum et maximum

- Écrire un algorithme permettant de saisir une liste de n réels et de calculer et d'afficher les valeurs minimale et maximale de cette liste.

Retour sur l'exercice Minimum et maximum

- Écrire un algorithme permettant de saisir une liste de n réels et de calculer et d'afficher les valeurs minimale et maximale de cette liste.
- Deux lectures possibles ...

Retour sur l'exercice Minimum et maximum

- Écrire un algorithme permettant de saisir une liste de n réels et de calculer et d'afficher les valeurs minimale et maximale de cette liste.
- Deux lectures possibles ...
- On saisie un réel et on ne garde que le min et le max

Retour sur l'exercice Minimum et maximum

- Écrire un algorithme permettant de saisir une liste de n réels et de calculer et d'afficher les valeurs minimale et maximale de cette liste.
- Deux lectures possibles ...
- On saisie un réel et on ne garde que le min et le max
- On saisie la liste des réels et on calcule le min et le max

Retour sur l'exercice Minimum et maximum

Algorithme MinMax - v1

Postcond : Algorithme calculant le min et le max
Algorithme MinMax
Variables
r, n, min, max : Réel
i : Entier
Début
  Ecrire("Donner le nombre de réels dans la liste")
  Lire n
  i $\leftarrow$ 1
  Faire
    Ecrire("Donner un nombre réel")
    Lire r

Algorithme MinMax - v1

```
    Si i=1 Alors
    Début
      min ← r
      max ← r
    Fin
    Sinon
    Début
      Si min > r alors min ← r
      Si max < r alors max ← r
    Fin
    FinSi
    i ← i+1
  Tant que i<=n
Fin
```

Retour sur l'exercice Minimum et maximum

Algorithme MinMax - v2

Postcond : Algorithme calculant le min et le max
Algorithme MinMax
Variables
lr : chaîne ? liste ? de Réels
r, n, min, max : Réel
i : Entier
Début
  Ecrire("Donner le nombre de réels dans la liste")
  Lire n
  i ← 1
  Faire
    Ecrire("Donner le nombre réel [",i,"]")
    Lire r
    lr[i] ← r
    i ← i+1
  Tant que i<=n

Algorithme MinMax - v2

```
i ← 1
Tant que i<=n
 Si i=1 Alors
 Début
  min ← lr[1]
  max ← lr[1]
 Fin
 Sinon
 Début
  Si min > lr[i] alors min ← lr[i]
  Si max < lr[i] alors max ← lr[i]
 Fin
 FinSi
 i ← i+1
Fin Tant que
Fin
```

Introduction

- Dans l'algorithme précédent on ne sait pas comment nommer la variable lr : chaîne ? liste ? de Réels

Introduction

- Dans l'algorithmes précédent on ne sait pas comment nommer la variable lr : chaîne ? liste ? de Réels
- Les 4 types de base connus sont : **Entier**, **Caractère**, **Réel**, **Booléen**, et **Chaîne**.

Introduction

- Nous avons une opération [] pour accéder aux différents caractères d'une chaîne, pour les consulter ou les modifier.
- La fonction 'standard' **longueur()** nous donne le nombre de caractères d'une chaîne.
- Les indices des caractères dans une chaîne *ch* varient entre 1 et *longueur*(*ch*).
- La concaténation des chaînes sera notée &.

Introduction

- Nous avons tout pour parcourir une liste d'éléments

Introduction

- Nous avons tout pour parcourir une liste d'éléments
- Mais, dans notre pseudo langage, cela ne fonctionne QUE pour les éléments "caractères" ... pas les réels.

Introduction

- Nous avons tout pour parcourir une liste d'éléments
- Mais, dans notre pseudo langage, cela ne fonctionne QUE pour les éléments "caractères" ... pas les réels.
- Alors comment faire ?

Introduction

Introduction aux données complexes.

Introduction

- Dans une application de géométrie nous manipulons des **points** et des **vecteurs**.

Introduction

- Dans une application de géométrie nous manipulons des **points** et des **vecteurs**.
- Dans toutes les applications supposant la résolution de systèmes d'équations linéaires, nous manipulons des **vecteurs** et des **matrices**.

Introduction

- Dans une application de géométrie nous manipulons des **points** et des **vecteurs**.
- Dans toutes les applications supposant la résolution de systèmes d'équations linéaires, nous manipulons des **vecteurs** et des **matrices**.
- Dans une application de gestion de la scolarité nous manipulons des **étudiants** et des **matières**.

Introduction

- Dans une application de géométrie nous manipulons des **points** et des **vecteurs**.
- Dans toutes les applications supposant la résolution de systèmes d'équations linéaires, nous manipulons des **vecteurs** et des **matrices**.
- Dans une application de gestion de la scolarité nous manipulons des **étudiants** et des **matières**.
- Dans une application de gestion commerciale nous manipulons des **produits**, des **clients** et des **fournisseurs**.

Introduction

- Un étudiant est défini, par exemple, par un nom, un prénom et une année d'études.

Introduction

- Un étudiant est défini, par exemple, par un nom, un prénom et une année d'études.
- Un produit est défini, par exemple, par un code et un libellé.

Introduction

- Un étudiant est défini, par exemple, par un nom, un prénom et une année d'études.
- Un produit est défini, par exemple, par un code et un libellé.
- Un client (ou un fournisseur) est défini, par exemple, par un code, un nom et une adresse.

Introduction

- Le point commun entre ces données est que qu'elles sont "**complexes**"

Introduction

- Le point commun entre ces données est que qu'elles sont "**complexes**"
  - Par opposition aux données "**simples**" rappelées ci-dessus définies par une seule valeur.

Introduction

- Le point commun entre ces données est que qu'elles sont "**complexes**"
  - Par opposition aux données "**simples**" rappelées ci-dessus définies par une seule valeur.
- Un point p se définit par un couple (x, y) si on est dans le plan et par un n-uplet $(x_1, ..., x_n)$ si on est dans un espace de dimension n.

Introduction

- Le point commun entre ces données est que qu'elles sont "**complexes**"
  - Par opposition aux données "**simples**" rappelées ci-dessus définies par une seule valeur.
- Un point p se définit par un couple (x, y) si on est dans le plan et par un n-uplet $(x_1, ..., x_n)$ si on est dans un espace de dimension n.
- Un vecteur de R^n se définit par un n-uplet $(x_1, ..., x_n)$ de réels.

Introduction

- Le point commun entre ces données est que qu'elles sont "**complexes**"
  - Par opposition aux données "**simples**" rappelées ci-dessus définies par une seule valeur.
- Un point p se définit par un couple (x, y) si on est dans le plan et par un n-uplet $(x_1, ..., x_n)$ si on est dans un espace de dimension n.
- Un vecteur de R^n se définit par un n-uplet $(x_1, ..., x_n)$ de réels.
- Une matrice à n lignes et m colonnes se définit par m n-uplets $(x_{i1}, ..., x_{in})$ de réels.

Introduction

- Ces données ont en commun d'être "**composées**" d'autres données mais diffèrent de deux points de vue :

Introduction

- Ces données ont en commun d'être "**composées**" d'autres données mais diffèrent de deux points de vue :
  - Certaines sont "numérotées" (on accède à chaque élément à l'aide d'un indice) et homogènes (tous les éléments sont du même type) : c'est le cas des vecteurs et des matrices.

Introduction

- Ces données ont en commun d'être "**composées**" d'autres données mais diffèrent de deux points de vue :
  - Certaines sont "numérotées" (on accède à chaque élément à l'aide d'un indice) et homogènes (tous les éléments sont du même type) : c'est le cas des vecteurs et des matrices.
  - Certaines sont "hétérogènes" et non numérotées : c'est le cas des étudiants, des matières, des produits, des clients et des fournisseurs.

Introduction

- Ces données ont en commun d'être "**composées**" d'autres données mais diffèrent de deux points de vue :
  - Certaines sont "numérotées" (on accède à chaque élément à l'aide d'un indice) et homogènes (tous les éléments sont du même type) : c'est le cas des vecteurs et des matrices.
  - Certaines sont "hétérogènes" et non numérotées : c'est le cas des étudiants, des matières, des produits, des clients et des fournisseurs.
- Nous allons représenter la première catégorie à l'aide des **tableaux**

Introduction

- Ces données ont en commun d'être "**composées**" d'autres données mais diffèrent de deux points de vue :
  - Certaines sont "numérotées" (on accède à chaque élément à l'aide d'un indice) et homogènes (tous les éléments sont du même type) : c'est le cas des vecteurs et des matrices.
  - Certaines sont "hétérogènes" et non numérotées : c'est le cas des étudiants, des matières, des produits, des clients et des fournisseurs.
- Nous allons représenter la première catégorie à l'aide des **tableaux**
- et la seconde à l'aide des **enregistrements**.

Les tableaux

Définitions

- Nous avons souvent besoin de manipuler une suite finie de données de même type.

Définitions

- Nous avons souvent besoin de manipuler une suite finie de données de même type.
  - Exemple : les notes (ou les noms ou les adresses) d'un groupe d'étudiants.

Définitions

- Nous avons souvent besoin de manipuler une suite finie de données de même type.
  - Exemple : les notes (ou les noms ou les adresses) d'un groupe d'étudiants.
  - Exemple : une liste de nombres réels.

Définitions

- Nous avons souvent besoin de manipuler une suite finie de données de même type.
  - Exemple : les notes (ou les noms ou les adresses) d'un groupe d'étudiants.
  - Exemple : une liste de nombres réels.
- Une telle suite est représentée par un tableau (à une dimension).

Définitions

- Nous avons souvent besoin de manipuler une suite finie de données de même type.
  - Exemple : les notes (ou les noms ou les adresses) d'un groupe d'étudiants.
  - Exemple : une liste de nombres réels.
- Une telle suite est représentée par un tableau (à une dimension).
- Dans un tableau nous avons donc un ensemble d'éléments de même type et numérotés (accessible via un indice).

Les tableaux

Déclaration

nomdutableau : tableau[1..tailledutableau] de typededonnées

Les tableaux

Déclaration

nomdutableau : tableau[1..tailledutableau] de typededonnées

- *tailledutableau* doit être une constante (jamais de variable ! ! !).

Les tableaux

Déclaration

nomdutableau : tableau[1..tailledutableau] de typededonnées

- *tailledutableau* doit être une constante (jamais de variable ! ! !).
- *typededonnees* est un type de données quelconque (simple ou complexe ! ! !).

Les tableaux

Déclaration

nomdutableau : tableau[1..tailledutableau] de typededonnées

- *tailledutableau* doit être une constante (jamais de variable ! ! !).
- *typededonnees* est un type de données quelconque (simple ou complexe ! ! !).

Manipulation

- L'accès au i^{ieme} élément d'un tableau *tab* se fait par *tab*[*i*].

Les tableaux

- Exemples :

Les tableaux

- Exemples :
  - nomsEtudiants : tableau[1..150] de Chaîne

Les tableaux

- Exemples :
  - nomsEtudiants : tableau[1..150] de Chaîne
  - listeProduits : tableau[1..10] de Produit

Les tableaux

- Exemples :
  - nomsEtudiants : tableau[1..150] de Chaîne
  - listeProduits : tableau[1..10] de Produit
  - **lr : tableau[1..500] de Réel**

Retour sur l'exercice Minimum et maximum

Algorithme MinMax - v2

Postcond : Algorithme calculant le min et le max
Algorithme MinMax
Variables
lr : tableau[1..500] de Réel
r, n, min, max : Réel
i : Entier
Début
  Ecrire("Donner le nombre de réels dans la liste")
  Lire n
  i ← 1
  Faire
    Ecrire("Donner le nombre réel [",i,"]")
    Lire r
    lr[i] ← r
    i ← i+1
  Tant que i<=n

Les tableaux

- Nous pouvons correctement déclarer lr : tableau[1..500] de Réel

Les tableaux

- Nous pouvons correctement déclarer lr : tableau[1..500] de Réel
- Nous avons tout pour parcourir une liste d'éléments

Les tableaux

- Nous pouvons correctement déclarer lr : tableau[1..500] de Réel
- Nous avons tout pour parcourir une liste d'éléments
- Mais on doit poser une constante pour la taille du tableau

Les tableaux

- Nous pouvons correctement déclarer lr : tableau[1..500] de Réel
- Nous avons tout pour parcourir une liste d'éléments
- Mais on doit poser une constante pour la taille du tableau
- Y-a-t-il un autre moyen ?

Les tableaux dynamiques

- Ici la taille n'est pas connue d'avance.

Les tableaux dynamiques

- Ici la taille n'est pas connue d'avance.
- Le tableau sera alors déclaré de la manière suivante :

Les tableaux dynamiques

- Ici la taille n'est pas connue d'avance.
- Le tableau sera alors déclaré de la manière suivante :
  - nomdutableau : tableau de typededonnées

Les tableaux dynamiques

- Ici la taille n'est pas connue d'avance.
- Le tableau sera alors déclaré de la manière suivante :
  - nomdutableau : tableau de typededonnées
- Lorsque nous pourrons lui donner une taille précise, nous le ferons de la manière suivante :

Les tableaux dynamiques

- Ici la taille n'est pas connue d'avance.
- Le tableau sera alors déclaré de la manière suivante :
  - nomdutableau : tableau de typededonnées
- Lorsque nous pourrons lui donner une taille précise, nous le ferons de la manière suivante :
  - affecterLongueur(nomdutableau, tailledutableau)

Retour sur l'exercice Minimum et maximum

Algorithme MinMax - v2

Postcond : Algorithme calculant le min et le max
Algorithme MinMax
Variables
lr : tableau de Réel
r, n, min, max : Réel
i : Entier
Début
  Ecrire("Donner le nombre de réels dans la liste")
  Lire n
  affecterLongueur(lr, n)
  i $\leftarrow$ 1
  Faire
    Ecrire("Donner le nombre réel [",i,"]")
    Lire r
    lr[i] $\leftarrow$ r
    i $\leftarrow$ i+1

Algorithme inverserTableau

Les tableaux

Algorithme inverserTableau

variables
  i : Entier
  tab : tableau[1..10] de Entier
Début
//Lecture du tableau
Pour i **de** 1 **à** 10
  Ecrire("tab[",i, "]=")
  Lire(tab[i])
FinPour

Les tableaux

Suite

```
//Inversion du tableau
Pour i de 1 à 5
   tmp ← tab[i]
   tab[i] ← tab[11-i]
   tab[11-i] ← tmp
FinPour
//Affichage du tableau inversé
Ecrire("tab = ")
Pour i de 1 à 10
   Ecrire(tab[i]," ")
FinPour
```

Les tableaux multi-dimensionnels

- Les tableaux multi-dimensionnels (le plus souvent bi-dimensionnels) se manipulent de manière analogue à celle des tableaux à une dimension.

Les tableaux multi-dimensionnels

- Les tableaux multi-dimensionnels (le plus souvent bi-dimensionnels) se manipulent de manière analogue à celle des tableaux à une dimension.

Déclaration

nomdutableau : tableau[1..taille1, 1..taille2] de typededonnées

Les tableaux multi-dimensionnels

- Les tableaux multi-dimensionnels (le plus souvent bi-dimensionnels) se manipulent de manière analogue à celle des tableaux à une dimension.

Déclaration

nomdutableau : tableau[1..taille1, 1..taille2] de typededonnées

Manipulation

- L'accès aux éléments se fait grâce à un couple d'indices (i, j). Ces éléments sont notés $tab[i, j]$.

Les tableaux multi-dimensionnels

- Les tableaux multi-dimensionnels peuvent être dynamiques. Dans ce cas ils sont déclarés de la manière suivante :
  - nomdutableau : tableau de tableau typededonnées

Les tableaux multi-dimensionnels

- Les tableaux multi-dimensionnels peuvent être dynamiques. Dans ce cas ils sont déclarés de la manière suivante :
  - nomdutableau : tableau de tableau typededonnées
- L'affectation des tailles se fait alors de la manière suivante :
  - affecterLongueur(tab, t)
  - Pour i de 1 à t
    - affecterLongueur(tab[i], ti)
  - FinPour

Les tableaux multi-dimensionnels

- Les tableaux multi-dimensionnels peuvent être dynamiques. Dans ce cas ils sont déclarés de la manière suivante :
  - nomdutableau : tableau de tableau typededonnées
- L'affectation des tailles se fait alors de la manière suivante :
  - affecterLongueur(tab, t)
  - Pour i de 1 à t
    - affecterLongueur(tab[i], ti)
  - FinPour
- Remarque : Notez bien que dans ce cas les *ti* ne sont pas forcément tous égaux. On peut avoir un tableau de tableaux de taille différente.

Les enregistrements

Les enregistrements

- Les enregistrements nous permettent de représenter des données **composées** et éventuellement **hétérogènes**.

Les enregistrements

- Les enregistrements nous permettent de représenter des données **composées** et éventuellement **hétérogènes**.
- Exemple :
  - (Dans une application de gestion de la scolarité) Etudiant = nom+prénom+année d'études.
  - (Idem) Matière = intitulé+année d'enseignement+nombre d'heures.
  - (Dans une application de gestion commerciale) Produit = code+libellé.
  - (Dans une application mathématique) Nombre complexe = une partie réelle + une partie imaginaire.

Les enregistrements

Définition

Enregistrement nomenreg
  nomchamps1 : typechamps1
  ...
  nomchamps1 : typechamps1
Finenregistrement

Les enregistrements

Définition

Enregistrement nomenreg
  nomchamps1 : typechamps1
  ...
  nomchamps1 : typechamps1
Finenregistrement

Déclaration

nomvar : nomenreg

Les enregistrements

Définition

Enregistrement nomenreg
  nomchamps1 : typechamps1
  ...
  nomchamps1 : typechamps1
Finenregistrement

Déclaration

nomvar : nomenreg

Manipulation

- L'accès au champs *nomchamps* d'une variable *nomvar* se fait par *nomvar.nomchamps*.

Les enregistrements

Exemple

Enregistrement Produit
  code : Entier
  libelle : Chaîne
Finenregistrement
...
vprod : Produit
Ecrire("Donnez le code du produit : ")
Lire(vprod.code)
Ecrire("Donnez la désignation du produit : ")
Lire(vprod.libelle)
Ecrire("Produit saisi : code=",vprod.code," libellé =
",vprod.libelle)

Retour sur l'exercice Minimum et maximum

Algorithme MinMax - v2

Postcond : Algorithme calculant le min et le max
Algorithme MinMax
Enregistrement MiniMaxi
  min : Réel
  max : Réel
Finenregistrement
Variables
lr : tableau de Réel
m : MiniMaxi
r, n : Réel
i : Entier
Début
  Ecrire("Donner le nombre de réels dans la liste")
  Lire n
  affecterLongueur(lr, n)
  i $\leftarrow$ 1

Algorithme MinMax - v2

```
i ← 1
Tant que i<=n
 Si i=1 Alors
 Début
  m.min ← lr[1]
  m.max ← lr[1]
 Fin
 Sinon
 Début
  Si m.min > lr[i] alors m.min ← lr[i]
  Si m.max < lr[i] alors m.max ← lr[i]
 Fin
 FinSi
 i ← i+1
Fin Tant que
Fin
```

Les enregistrements

Exercice

Écrire un algorithme permettant de lire un nombre complexe saisi par l'utilisateur et afficher son module.
Rappel : le module d'un nombre complexe z est noté $|z|$. Si le complexe z s'exprime sous sa forme algébrique, $a + ib$, où i est l'unité imaginaire, a est la partie réelle de z et b sa partie imaginaire, ce module est la racine carrée de la somme des carrés de a et b.

Les enregistrements

Module nombre complexe

Précond : On suppose la fonction racine() existante
Postcond : Algorithme calculant le module d'un nombre complexe
Algorithme Module_Z
Enregistrement Complexe
  a_reel : Réel
  b_imag : Réel
Finenregistrement
Variables
nb : Complexe
Début
  Ecrire("Donner la partie réelle")
  Lire (nb.a_reel)
  Ecrire("Donner la partie imaginaire")
  Lire (nb.b_imag)
  Ecrire("Le module est ",racine($nb.a\_reel^2 + nb.b\_imag^2$))
Fin

Conclusion

- Données complexes : tableaux et enregistrements.
- Cours suivant : structures de données (pile, file et liste).