Devoir Ecrit

Janvier 2014

Durée : 2 heures

Sans calculatrice, sans documents

185

GAUTIER
Arthur

L1CPI1
2013

Avant de débuter le DE :

- Prenez le temps de bien lire les énoncés des exercices. Vous répondrez dans les espaces laissés libres à cet effet dans le sujet. Aucune autre copie ne sera prise en compte.
- N'oubliez pas de reporter vos nom, prénom, groupe et école dans les espace ci-dessous
- **La qualité de la rédaction de vos réponses entrera en compte pour la notation des exercices**

NOM GAUTIER

PRENOM Arthur

GROUPE A

Promotion L1 [X] CPI1 [ ]

21

Thème 1 : Éléments de base de l'algorithmique

Qu'est-ce qu'une variable ?

Une variable est un mot en algorithmique qui sert à stocker une valeur. Toute variable est définie par son type (la forme des éléments qu'elle peut stocker) et par son nom ainsi que par sa valeur

1

Quelle est l'utilité des tests et des boucles en algorithmique ?

En algorithmique, les tests et les boucles servent à ne pas multiplier les lignes en permettant d'effectuer des instructions répétées ou de vérifier des conditions de façon simple.

1

Comment, en algorithmique, peut-on tester simplement si une entier **a** est divisible par un entier **b** ?

```
entier a;
entier b;
entier reste;
reste <- 0;
afficher("Que vaut a ?");
saisir(a);
afficher("Que vaut b ?");
```

```
saisir(b);
reste <- a%b;
Si(reste = 0)
{
afficher(a" est divisible par "b);
}
```

? % !!

```
Sinon
{
afficher(a" n'est pas divisible par "b);
}
```

/

Un programme de calcul de factorielle (qui utilise un algorithme correct) vous indique que : `17! = -288 522 240`. Pourquoi la machine n'indique-t-elle pas qu'il s'agit d'une erreur ? Pour quelle raison la machine peut-elle calculer une factorielle négative ?

La machine ne trouve pas d'erreur dans l'algorithme, elle n'en affiche donc pas. Cependant, la factorielle est négative car le* calcul dépasse la capacité de calcul de la machine, il y a donc un dépassement de capacité (overflow)

1

*résultat du

Lorsque l'on définit un tableau statique, que doit-on écrire entre les crochets [ ] ?

On doit écrire la taille maximale du tableau (le nombre d'éléments qu'il contient) (0,5)

Lorsque l'on souhaite accéder à une valeur stockée dans un tableau, que peut-on écrire entre les crochets [ ] ?

On peut écrire une valeur numérique, une expression ou une variable. Cependant, tous ses éléments doivent être (des valeurs numériques positives représentant) (1)

L'ordinateur peut-il vérifier qu'un indice est valide lors d'un accès à une valeur stockée dans un tableau ? Justifiez votre réponse (*une réponse par un simple oui ou non vaudra 0 points*)

Non. En effet, l'ordinateur alloue un bloc mémoire au tableau mais a la capacité de regarder en dehors de ce bloc mémoire. Cependant, si l'on essaye d'accéder et de manipuler un élément n'appartenant pas au tableau, l'ordinateur risque de renvoyer une erreur de segmentation (0,5)

Thème 2 : Tableaux et boucles

Initialiser un tableau

Soit un tableau statique de réels dont la définition est la suivante :

```
reel tabval[50];
```

Ce tableau, bien entendu, aura une taille utile, qui sera nommée **util**.

Quel est le type de cette variable **util** ? Pourquoi doit-on choisir ce type ?

Le type de util est entier puisque cette variable indique l'indice de la prochaine case disponible (indices qui sont des entiers) (1)

Le tableau **tabval** n'étant pas initialisé, quelle doit être la valeur initiale de **util** ?

util doit être initialisée à 0. (0,5)

Quelle est la valeur maximale que peut prendre la variable **util** ?

La valeur maximale que peut prendre util est la taille maximale du tableau, ici, 50. (0,5)

On souhaite remplir le tableau **tabval** par des saisies de l'utilisateur : le principe est que l'utilisateur doit saisir au moins une valeur à stocker dans le tableau, et que le programme demande s'il souhaite continuer à saisir des valeurs.

Rédigez, en langage algorithmique, le programme correspondant – Rappel : ce programme a été intégralement étudié et rédigé en cours magistral.

```
entier ctrl = 0;
entier reponse = 0;
réel tabval[50];
réel valeur;
faire
{
  afficher("Quelle valeur réelle voulez-vous stocker à la case "ctrl" du tableau ?");
  Saisir(valeur);
  tabval[ctrl] ← valeur;
  faire ctrl ← ctrl + 1;
  afficher("Voulez-vous continuer la saisie ? 0=oui, 1=non");
  Saisir(reponse);
}
tant que (reponse ≠ 0 OU reponse ≠ 1);
}
tant que (reponse = 0);
```

(1)

+ tableau rempli ?

Faire une recherche par dichotomie

Lorsque l'on recherche une valeur précise dans un tableau, cette recherche peut être très efficace si le tableau est trié, en utilisant le principe de la dichotomie.

On considère, pour l'exercice à traiter, que les valeurs situées dans le tableau, nommé **tabrech**, sont triées par ordre croissant. On nommera **t_ut** la taille utile de ce tableau **tabrech**.

Ainsi, on peut affirmer que : $\forall$ i, j , $0 \leq i < t\_ut$, $0 \leq j < t\_ut$, $i < j \Rightarrow tabrech[i] \leq tabrech[j]$

Traduisez ce constat mathématique en français

Pour toute valeur positive i et j strictement inférieure à la taille utile, si i est strictement inférieure à j, cela implique que la valeur du tableau à l'indice i est inférieure ou égale à la valeur du tableau à l'indice j.

(1)

Soit **val** la valeur à rechercher dans ce tableau, et **ind** un indice qui indique la position où effectuer la recherche dans le tableau **tabrech**. Au début du programme, on fixe **ind** à **t_ut/2** (c'est une division entière, donc **ind** est bien un entier).

Première étape : on compare **val** et **tabrech[ind]**.

Compléter, en français, les cas suivants :

(1)

1. Si **val = tabrech[ind]**, alors la valeur recherchée se trouve à la case d'indice ind du tableau
2. Si **val < tabrech[ind]**, alors **val** est inférieur à
   la valeur de la case d'indice ind du tableau, il faut donc chercher avant dans le tableau
3. Si **val > tabrech[ind]**, alors **val** est supérieur à
   la valeur de la case d'indice ind du tableau, il faut donc chercher après dans le tableau

(2)

Après cette première étape, soit on a trouvé l'élément recherché, soit on doit le rechercher dans un tableau 2 fois plus court que le tableau original (c'est-à-dire que la taille utile de la zone dans laquelle on recherche est divisée par 2). Pour faire cette recherche, on utilise la même méthode.

Supposons que la taille utile `t_ut` de `tabrech` soit 1024 : au bout de combien d'étapes la zone de recherche atteint la taille de 1 (c'est-à-dire qu'on teste une seule case) ? Faites apparaître le calcul dans la réponse.

Pour passer d'une taille utile de 1024 à une taille utile de 1, il faut avoir diviser n fois par 2. On a $1024 = 2^{10}$. Donc, il y aura besoin de 10 étapes pour rechercher dans une seule case

(1)

Que pensez-vous de l'efficacité de cette méthode ?

Cette méthode n'est pas très efficace puisque potentiellement très longue. Plus la valeur de la taille utile augmente, plus le nombre d'étapes augmente

Comment ?

Thème 3 : les pointeurs

Soit le programme suivant

```
programme pointeurs_et_tableaux

caractere a[9] ← {12, 23, 34, 45, 56, 67, 78, 89, 90};
caractere *p;
p ← a;
```

Quelles valeurs ou adresses fournissent ces expressions ? On supposera que le tableau **a** est stocké à l'adresse 8000 en mémoire de l'ordinateur. Un caractère occupe un octet.

a)	`*p+2`	14
b)	`*(p+2)`	34
c)	`p+1`	8001
d)	`&a[4]-3`	~~8000~~ 0
e)	`a+3`	~~15~~
f)	`&a[7]-p`	~~6~~
g)	`p+(*p-10)`	8002
h)	`*(p+*(p+8)-a[7])`	23

15

Allocation dynamique

Avec une dimension

Ecrivez un programme (toujours en langage algorithmique) répondant à la séquence suivante :

1. Le programme utilise un tableau statique de caractère de grande taille
2. L'utilisateur saisit un texte dans ce tableau
3. Le programme fait une allocation dynamique pout un tableau de caractères dont la taille est exactement celle qui est nécessaire à stocker le texte saisi dans le tableau statique
4. Le programme recopie le texte du tableau statique dans le tableau dynamique

```
caractère tabs[1000000];
entier taille ← 0;
Caractere * tabmem;
afficher("Entrez votre texte");
lire (tabs);
taille ← longueur(tabs);
tabmem ← reservation (taille+1 caractere);
Copier (tabs tabmem);
```

(2)

Avec deux dimensions

Le triangle de Pascal est un tableau particulier qui stocke un ensemble de coefficients entiers que l'on appelle également : coefficient du binôme.

Dans sa ligne *i*, ce tableau stocke tous les coefficients de la forme développée de la formule : $(a+b)^i$

Exemples :

$(a+b)^0 = 1$: coefficient : 1

$(a+b)^1 = 1.a + 1.b$: coefficients 1, 1

$(a+b)^2 = 1.a^2+2.ab+1.b^2$: coefficients 1, 2, 1

En prenant les exemples des puissances 3 et 4, écrivez le triangle de Pascal et établissez une relation simple entre les coefficients permettant de passer d'une ligne à l'autre.

```
   :
3: 1 3 3 1
4: 1 4 6 4 1
   :
```

Soit i la ligne et j la colonne et coef le coefficient

coef[i][j] = coef[i-1][j-1] + coef[i-1][j]

(1)

Ecrivez le programme qui crée et remplir ce tableau pour un nombre de lignes N variable (traité en cours)

```
entier i<-0;
entier j<-0;
entier n<-0;
entier **pascal;
afficher("Combien de lignes voulez-vous ?");
Saisir(n);
*pascal<- reservation(n * entier);
pascal<-reservation(1 en
pour(i de 0 à n selon 1)
{
  *pascal[i]<- reservation(i+1 entier);
}pascal[ej][0]<- 1;
pour(i de  à n selon 1)
{ pascal[i][0]<- 1;
  pascal[i][i   ]<- 1;
}
```

(1)

(1,5)

(0,5)

```
pour(i de 1 à n selon 1)
{
  pour(j de 1 à i selon 1)
  {
    pascal[i][j] = pascal[i-1][j-1] + pascal[i-1][j];
  }
}
```

05