Devoir Ecrit

Janvier 2014

Durée : 2 heures

Sans calculatrice, sans docu...

Charles 2013

13,5

BLOQUET Romain PL1 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 BLOQUET

PRENOM Romain

GROUPE PL1

Promotion L1 [X] CPI1 [ ]

15,5

Thème 1 : Éléments de base de l'algorithmique

Qu'est-ce qu'une variable ?

0,5

Une Variable est une boîte que l'on peut utiliser car elle stocke une valeur. Une Variable a un type, un nom et un contenu.

+ @

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

1

Un test en algorithmique est utile pour savoir si ce que l'on écrit est Vrai ou faux, donc deux solutions possibles 0 ou 1 (0 pour faux et 1 pour Vrai).
Une boucle est un test reitéré, elle sert à faire un test plusieurs fois selon les conditions données.

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

1

Pour savoir si un entier a est divisible par un entier b en algorithmique il faut faire le modulo. le modulo entre deux entiers donne le reste, si il est égale à zero alors a est divisible par b.

① entier a, b, c;
a ← 25

② b ← 5
c ← a % b

③ afficher (c)
c = 0

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 ?

1

La Machine ne nous indique pas qu'il y a une erreur car elle exécute ce qu'on lui demande de faire. Elle calcule une factorielle négative car la factorielle de 17 est trop importante, elle dépasse donc la capacité de stockage en mémoire. Il y overflow.

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

Lorsqu'on définit un tableau statique, on doit écrire la taille maximale entre les crochets.

0,5

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

Lorsqu'on souhaite accéder à une valeur stockée dans un tableau on peut écrire l'indice de la case souhaitée entre les crochet

0,5

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 l'ordinateur ne peut pas vérifier qu'un indice est valide lors d'un accès à une valeur stockée car il exécute ce qu'on lui demande, cela nous donnera une valeur erronée. Il faut faire une saisie sécurisé pour ce en utilisant la boucle faire... tant que.

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 ?

la variable util sera de type entier car par exemple dans un tableau on ne peut pas utiliser 2,75 cases.

1

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

la valeur initiale de util devra être 0 car le tableau tabval n'est pas initialisé.

0,5

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

la valeur maximale que peut prendre la variable util est la valeur maximale donné à l'initialisation du tableau moin un. exemple tabval[50] la taille util maximale sera 49

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.

Programme remplissage d'un tableau par saisie d'utilisateur :

```
reel tabval [50];
entier util;
charactere reponse;

util ← 0;    //initialisation de util.

faire
{
   Afficher ("Saisir une Valeur dans le tableau :");
   saisir (tabval [util]);
   util ← util + 1;
   Afficher ("souhaitez Vous Continuer ? saisir o pour
              oui, n pour non");
   saisir (reponse);          + taille tab.
} tant que (reponse = 'o');

// a la sortie de la boucle util aura la bonne valeur.
```

(1)

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 tout i, j, i est strictement inférieur à t_ut et supérieur ou égal à zéro, j est strictement inférieur à t_ut et supérieur ou égal à zéro et i strictement inférieur à j ce qui implique que tabrech[i] est inférieur ou égal à tabrech[j].

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. Si **val = tabrech[ind]**, alors la Valeur est présente.
2. Si **val < tabrech[ind]**, alors **val** est inférieur à

   la Valeur de la Case du tableau indiquée par l'indice ind. et donc.
3. Si **val > tabrech[ind]**, alors **val** est supérieur à

   la Valeur de la Case du tableau indiquée par l'indice ind.

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.

Quand toutes les cases on déjà été testées.

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

Cette méthode n'est pas efficace car si la valeur a chercher se trouve avant l'indice elle ne pourra pas être trouvée. ~~et si elle ne se trouve pas dans le tab~~

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, 25, 36, 47, 58, 69, 80, 91, 92};
b)	`*(p+2)`	{14, 25, 36, 47, 58, 69, 80, 91, 92};
c)	`p+1`	8001
d)	`&a[4]-3`	8002.
e)	`a+3`	{15, 26, 37, 48, 59, 70, 81, 92, 93};
f)	`&a[7]-p`	7
g)	`p+(*p-10)`	
h)	`*(p+*(p+8)-a[7])`	

0,5

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

```
# include <cstring>;
caractère tab [10000];
caractère * tab dynamique;
entier longueur_texte;
Afficher ("Veuillez saisir un texte");
lire (tab);
longueur_texte ← strlen (tab);
tab dynamique ← reservation (longueur_texte + 1 caractere);
copier (tab dynamique, tab);
```

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.

$(a+b)^3 = 1.a^3 + 3a^2b + 3ab^2 + 1$: Coefficient: 1, 3, 3, 1

$(a+b)^4 = 1.a^4 + 4.a^3b + 6.a^2b^2 + 4ab^3 + 1.b^4$. coefficient: 1, 4, 6, 4, 1

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

dans la ligne d'indice i par exemple pour trouver 2 il faut additionner la ligne d'au dessus (même colonne) avec la ligne d'au dessus avec une colonne en moin.

(1)

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

```
entier ** pascal;
entier lig, col;
entier indice, t_ut;
t_ut ← 0;
indice ← 0;
Afficher (" Veuillez saisir le nombre de lignes souhaitez:");
saisi (indice);
pascal ← reservation (indice, entier);
  de 0 à indice selon +1 Faire
{
  pascal [t_ut] ← pascal [t_ut +1];
}
  pascal [lig][col] ← 1;
de 0 à indice selon +1 Faire
{
```

(1)

(0,5)

2ème reservation ?

(0,5)

pascal [lig][col] ← pascal [lig-1][col] + [lig-1][col-1];