UNIVERSITE DE SHERBROOKE

Département de mathématiques

MAT 2584

Langages de programmation

Examen intra

Mardi, le 11 juillet 1978 Professeur: Guy Tiphane

de 13 h 30 à 15 h 30.

Note: Toute documentation permise.

Question 1

Supposons qu'on a deux langages de programmation *L1* et *L2*. Tous les deux offrent la possibilité de définir des tableaux à plusieurs dimensions. Cependant, chaque langage implante les tableaux en mémoire d'une façon particulière: pour *L1*, le dernier indice varie le plus rapidement, alors que pour *L2* c'est le premier, i.e.

pour *L1*: $(i_1, \ldots, i_n + 1)$ est situé après $(i_1, \ldots, i_n)$

pour *L2*: $(i_1 + 1, \ldots, i_n)$ est situé après $(i_1, \ldots, i_n)$

a) (3 points)

Si on définit un tableau T à 3 dimensions de la façon suivante:

.../2

Question 1 (suite)

T: *TABLEAU*[-1..1, 'A'..'E', 0..5]

Combien d'éléments contient le tableau T? (On suppose que les lettres se suivent dans notre jeu de caractères)

b) (10 points)

Etant donnée la même définition et en supposant que le tableau est implanté à l'adresse 0, quelle sera l'adresse de l'élément T[0, 'C', 2]

- pour un programme écrit en *L1*?
- pour un programme écrit en *L2*?

c) (5 points)

En supposant qu'on peut appeler une procédure écrite en *L1* à partir d'un programme écrit en *L2* et vice-versa, comment devront être les définitions et les utilisations, dans chaque sous-programme, de tableaux partagés entre les sous-programmes écrits en *L1* et ceux écrits en *L2*?

Question 2

Soit la grammaire suivante, exprimée en *BNF*:

(voir suite de la question sur page suivante)

Question 2 (suite)

<énoncé> ::= <identificateur> ← <expression>

<identificateur> ::= <lettre> | <lettre> <fin d'identificateur>

<fin d'identificateur> ::= <lettre> | <chiffre> |
<lettre> <fin d'identificateur> |
<chiffre> <fin d'indentificateur>

<expression> ::= (<expression> + <expression>) |
<identificateur> | <nombre>

<nombre> ::= <chiffre> | <chiffre> <nombre>

<lettre> ::= *A*|*B*|*C*...*Y*|*Z*

<chiffre> ::= 0|1|2... 8|9

a) (9 points)

Donner une grammaire équivalente exprimée sous forme de diagrammes syntaxiques.

b) (5 points)

Parmi les bouts de textes suivants, quels sont ceux qui répondent à la syntaxe d'un énoncé telle que définie ci-haut? Justifier les réponses négatives.

1. *DEBUT* ← (12**A* + 5)
2. *A* ← ((*A* + *B*) + (*Z* + 1))
3. 2*X*2 ← 4
4. *PI* ← 3.14159
5. *Z* ← *A* + *Y*

Question 3

Quels seront les contenus des variables du programme suivant à la fin de l'exécution

a) (6 points)

si les paramètres sont passés par valeur ?

b) (6 points)

s'ils sont passés par référence?

c) (6 points)

s'ils sont passés par nom?

```
programme passage;
var  i: entier;
     t: tableau [1..6] de entier;
 procédure bidon (n: entier);
    début
        i:= i + 1;
        n:= 3296
    fin;
début
      pour  i: 1 haut 6 faire
            t[i] := i;
      i:= 1;
      bidon (t[i]);
      bidon (i)
fin .
```

Question 4 (12 points)

Indiquer les identificateurs accessibles aux points (α) , (β) , (γ), (δ) dans le programme suivant:

```
programme toto;
const
        pi = 3.14159;
        max = 10;
        min = 0;
type
        complexe = struct
                        re, im : réel
                   fin;
        entier2 = entier;
var
        i, j: entier2;
        c: complexe;
        k: entier;
procédure première (A:réel);
var i:entier;
   procédure seconde (i: booléen);
       procédure troisième (k: réel);
       var première: booléen;
        début
             avec c faire
              début
                   (α)
              fin
        fin;
   début
        (β)
   fin;
début
     (γ)
fin;
début
     (δ)
fin.
```

Question 5

a) (12 points)

Que fait la procédure *MYSTERE* dans le programme suivant?

Expliquer par des schémas les opérations impliquées.

```
programme prog;
type ptrélément = ↑ élément;
     élément = struct
                    n: entier;
                    suiv: ptrélément;
               fin;
var
     ptr: ptrélément;
procédure MYSTERE (var p: ptrélément; val:entier);
var ptemp: ptrélément;
début
      nouveau (ptemp);
      avec ptemp ↑ faire
        début
             n:= val;
             suiv:= p
        fin;
      p:= ptemp
 fin;
 début (*du programme*)
      ptr:= nil;
      MYSTERE (ptr, 1);
      MYSTERE (ptr. 2);
      MYSTERE (ptr, 3);
 fin.
```

b) (6 points)

Exposer la situation (par un schéma) à la fin du programme ci-haut.

Question 6

Soit la fonction *ACK* ainsi définie:

```
fonction ACK (m, n: entier) : entier;
   début
        si  m = 0  alors  ACK := n + 1
        sinon
             si  n = 0  alors  ACK:=ACK(m-1, 1)
             sinon
                  ACK:=ACK (m-1, ACK(m, n-1))
   fin;
```

a) (15 points)

Etant donné un appel *ACK*(1,2), exposer toute la suite d'appels générés et la valeur obtenue après chaque appel.

b) (5 points)

Pourquoi une sous-routine *FORTRAN* ne peut-elle pas être récursive?

UNIVERSITE DE SHERBROOKE

Département de mathématiques

MAT 2584

Langages de programmation

Examen final

Vendredi, le 28 juillet 1978 Professeur: Guy Tiphane

de 9 heures à 12 heures.

Note: Toute documentation permise.

Question 1 (10 points)

Si on dit qu'un compilateur accepte deux représentations pour un même symbole du langage (par exemple, :=et.= représentent le même symbole), à quel(s) niveau(x) (lexical, syntaxique ou sémantique) le compilateur doit-il s'adapter à cette particularité? Pourquoi?

Question 2 (10 points)

Un sous-programme *FORTRAN* peu scrupuleux peut présumer que le contenu de ses variables locales ne change pas entre deux appels. Cela lui permet de connaître les valeurs laissées dans les variables locales à la fin de l'appel précédent. Cette pratique serait impossible et purement aléatoire avec un langage comme *PASCAL*. Pourquoi?

.../2

Question 3 (30 points)

Soit le programme *FORTRAN* suivant

```
      DIMENSION M(10,10)

      DO 100 I = 1,10
      DO 100 J = 1,10
  100 M(I,J) = 0

      CALL INIT (M,5,5)

      DO 200 I = 1,5
      DO 200 J = 1,5
      IF(M(I,J).NE.10) WRITE(6,150)I,J
  150 FORMAT(2I5)
  200 CONTINUE
      STOP
      END

      SUBROUTINE INIT(MAT,ILIG,ICOL)
      DIMENSION MAT(ILIG,ICOL)

      DO 1000 I = 1,ILIG
      DO 1000 J = 1,ICOL
 1000 MAT(I,J) = 10

      RETURN
      END
```

Bien que semblant initialiser une sous-matrice de *M*,*INIT* ne fait pas le travail attendu.

a) (10 points)

Pourquoi?

b) (10 points)

Quels seront les valeurs imprimées?

c) (10 points)

Quelle serait la solution au problème de *INIT*?

Question 4 (10 points)

a) Qu'est ce que le lien dynamique et à quoi sert-il?

b) Qu'est ce que le lien statique et à quoi sert-il?

Question 5 (10 points)

Dans un langage où on fait de l'allocation dynamique, quelles peuvent être les techniques permettant d'éviter un manque d'espace (récupération d'espace)?

Question 6 (10 points)

Que fait le programme *SNOBOL* suivant?

```
        N=1
ETIQ    OUTPUT = ' ' N ' ' INPUT   :F(END)
        N = N + 1                  :(ETIQ)
END
```

Question 7 (10 points)

a) Pourquoi utilise-t-on le passage de paramètres par valeur?

b) Par référence?

c) Quels sont les avantages et désavantages de chacun?

Question 8 (10 points)

Ecrivez une procédure qui lit une suite de nombres et les écrit dans l'ordre inverse, en utilisant la récursivité.

Par exemple 12 43 36 deviendra 36 43 12