Table des matières

1. Introduction

Pour chaque semaine, un QCM d'algorithmique est suivi de QCM pour le langage informatique étudié.

- Pour commencer un QCM on doit cliquer sur le bouton **Début**.
- Pour le terminer, on clique sur **Fin**.
- Le score s'affiche.

Vous pouvez recommencer le QCM en cliquant de nouveau sur Début ou voir les bonnes réponses en cliquant sur **Réponses**.

Un clic sur les cases vertes permet de voir une petite explication sur la bonne réponse.

Quand une réponse litérale est à fournir et que l'on a demandé la correction, un clic sur **Ans** fournit la réponse dans la case en bas du questionnaire. Un appui sur SIFT+clic donne la réponse complète si elle existe.

2. QCM Semaine 3

2.1. algorithmique

Début Répondre aux 7 questions.

1. Un paramètre formel d'un sous-programme est caractérisé par

☐ Mode de passage Type ☐ Nom Mode de passage ☐ Nom Mode de passage Type

2. Une fonction peut avoir comme mode de passage des paramètres formels :

☐ IN ☐ IN, INOUT ☐ IN, OUT

3. Une procédure a obligatoirement au moins un paramètre formel

☐ Vrai ☐ Faux

4. Pour une procédure, son nom et la liste des paramètres formels se nomme :

5. Un sous-programme peut changer la valeur d'un paramètre effectif si son mode de passage est

☐ IN ☐ INOUT ☐ OUT

6. Une procédure peut en appeler une autre

☐ Vrai ☐ Faux

7. Une procédure peut appeler une fonction

☐ Vrai ☐ Faux

8. Utilisation des variables globales

☐ C'est conseillé pour diminuer le nombre de parammètres ☐ C'est conseillé pour diminuer la taille du programme ☐ C'est déconseillé car cela nuit à la lisibilité du code

Fin Score : Réponses

Réponse :

2.2. Language C

Début Répondre aux questions suivantes.

1. Indiquer quel est l'appel correct pour la signature suivante :

```
void sp(int *i, double j);
int a;
double b;
```

☐ sp(a, b); ☐ sp(a, &b); ☐ sp(&a, &b); ☐ sp(&a, b); ☐ sp(*a, b); ☐ sp(a, *b);

2. Que donne l'exécution du programme suivant :

```
void f(int a, char *b) {
    printf("(%c%d)", *b, a);
    a++;
    *b = *b + 1;
    printf("(%c%d)", *b, a);
}
```

```
void main() {
    int i = 0;
    char c = 'A';
    f(i, &c);
    printf("(%c%d)", c, i);
}
```

☐ (A0)(B1)(A0) ☐ (A0)(B1)(B0) ☐ (A0)(B1)(B1) ☐ (A0)(B1)(A1)

3. Pour traduire le mode de passage en entrée (**in**) du langage algorithmique, on fait

☐ un passage par valeur ☐ un passage par adresse

4. Pour traduire le mode de passage en sortie (out) du langage algorithmique, on fait

☐ un passage par valeur ☐ un passage par adresse

5. Pour traduire le mode de passage en entrée/sortie (in out) du langage algorithmique, on fait

☐ un passage par valeur ☐ un passage par adresse

6. Qu'est ce qui indique qu'un sous-programme est une procédure et non une fonction ?

☐ le nom du sous-programme qui est verbe
☐ le type de retour **void**
☐ le nom du sous-programme qui est un nom commun
☐ le mot-clé function

7. En c, pour indiquer la valeur retournée par une fonction, on utilise :

☐ la variable prédéfinie **result** ☐ le nom de la fonction utilisée comme variable ☐ le mot-clé **return**

Fin Score : Réponses

Les réponses aux questionnaires

Réponse : Un paramètre formel est caractérisé par son nom, son mode de passage et son type. Retour au questionnaire.

Réponse : Une fonction ne peut avoir que des paramètres IN sinon, on utilise une procédure. Retour au questionnaire.

Réponse : Une procédure peut ne pas avoir de paramètre. Pour une fonction, ce serait bizarre car elle deviendrait une constante !

Retour au questionnaire.

Page d'accueil
Page de Titre
Sommaire
◀◀ ▶▶
◀ ▶
Page 7 de 4
Retour
Plein Ecran
Fermer
Quitter

Réponse : Il s'agit de la signature de la procédure.

Retour au questionnaire.

Réponse : Il peut modifier la valeur si le mode de passage est OUT ou INOUT.

Retour au questionnaire.

Page d'accueil
Page de Titre
Sommaire
◀◀ ▶▶
◀ ▶
Page 9 de 4
Retour
Plein Ecran
Fermer
Quitter

Réponse : Une procédure peut en appeler un autre. Toutefois, une fonction appelle rarement une procédure.

Retour au questionnaire.

Réponse : Une procédure peut appeler une fonction.

Retour au questionnaire.

Page d'accueil
Page de Titre
Sommaire
◀◀ ▶▶
◀ ▶
Page 11 de 4
Retour
Plein Ecran
Fermer
Quitter

Réponse : C'est déconseillé. cela nuit non seulemnt à la lisibilité du code mais également à la correction des erreurs.

Retour au questionnaire.

Réponse : Le paramètre i est par adresse, il faut donc passer en paramètre l'adresse de a : &a. La paramètre j est passé par valeur, il faut donc passer b. La bonne solution est donc : sp(&a, b).

Notons que ∗a et ∗b n'ont pas de sens puisque ni a, ni b ne sont du type pointeur !

Retour au questionnaire.

Réponse : Le sous-programme f est appelé avec les paramètres effectifs i (qui vaut 0) et l'adresse de j (qui vaut 'A'). Ainsi, dans f, a est initialisé avec la valeur de i et b contient l'adresse de j.

Le premier affichage (printf) affiche donc (A0).

Le sous-programme incrémente la valeur de a qui passe dont à 1 et ∗b qui passe donc du caractère 'A' au caractère 'B'.

Le deuxième affichage écrit donc (B1).

La paramètre a est passé par valeur (en entrée), les modifications qui lui sont apportées dans le sous-programme (l'incrémentation) ne seront pas visibles du programme appelant. En revanche, le paramètre b est passé par adresse (en entrée/sortie). En conséquence, la modification apportée dans le sous-programme sera vue du programme appelant.

Aussi, le dernier affichage (dans le programme principal) affiche (B0).

Notons que ∗b = ∗b + 1 aurait pu être noté (∗b)++. Mais attention dans ce cas à ne pas oublier les parenthèses !

Retour au questionnaire.

Réponse : Le mode de passage **in** signifie que la valeur du paramètre ne peut pas être changée ou, en tout cas, ces changements ne seront pas visibles du programme appelant. En conséquence, le mode **in** se traduit par un passage par valeur. Le paramètre formel est alors équivalent à une variable locale du sous-programme, variable initialisée avec la valeur du paramètre effectif.

Retour au questionnaire.

Réponse : Le mode de passage **out** signifie que la valeur du paramètre ne peut pas être utilisée par le sous-programme et que le sous-programme doit nécessairement lui donner une valeur qui sera vue du programme appelant. En conséquence, il faut faire un passage par adresse.

Retour au questionnaire.

Réponse : Le mode de passage **in out** signifie que la valeur du paramètre peut être utilisée par le sous-programme et que tout changement réalisé sur ce paramètre sera visible du programme appelant. En conséquence, il faut faire un passage par adresse.

Retour au questionnaire.

Réponse : C'est le « type de retour » **void** qui indique qu'un sous-programme est une procédure.

Même si généralement, le nom d'une procédure est un verbe, il ne s'agit pas d'une condition suffisante. Lorsque le nom du sous-programme est un nom commun, il s'agit généralement d'une fonction mais ce n'est pas non plus une garantie.

Enfin, le mot-clé **function** n'existe pas en C.

Retour au questionnaire.

Réponse : En C, c'est le mot-clé **return** suivi de la valeur de retour. Il n'y a pas de variable prédéfinie **result** qui pourrait être l'équivalent du **Résultat** du langage algorithmique. Le nom de la fonction ne peut pas être utilisée comme variable en C (contrairement à un langage comme Pascal).

Retour au questionnaire.