Apprenez à programmer en C !

40 heures Moyenne Licence

Mis à jour le 28/04/2021

Quiz 2

Bravo ! Vous avez réussi cet exercice !

Compétences évaluées

- Utiliser des pointeurs
- Écrire des commandes de préprocesseur
- Effectuer des allocations dynamiques

Question 1

Qu'est-ce qu'une variable globale ?

- [x] Une variable accessible partout
- [ ] Une variable qui peut accepter n'importe quel type (int, double...)
- [ ] Une variable déclarée dans la fonction main

Une variable globale peut être utilisée partout dans le projet. Même si c'est possible de créer de telles variables, il faut éviter de le faire car cela peut poser des problèmes si vous avez une autre variable ayant le même nom ailleurs dans votre programme.

Question 2

Quand on inclut un header d'une bibliothèque standard, à quoi cela ressemble-t-il ?

- [x] `#include <time.h>`
- [ ] `#include "time.h"`
- [ ] `#include (time.h)`
- [ ] `#include [time.h]`

Les chevrons < > permettent d'inclure des fichiers headers des bibliothèques standard (situés dans le dossier de l'IDE) et les guillemets " " permettent d'inclure des headers situés dans le dossier du projet (à côté du main).

Question 3

Dans quel ordre s'effectue une compilation ?

- [x] Préprocesseur - Compilateur - Linker
- [ ] Compilateur - Linker - Préprocesseur
- [ ] Linker - Préprocesseur - Compilateur
- [ ] Préprocesseur - Linker - Compilateur

Retournez voir les schémas de compilation si vous les avez oubliés.

Question 4

Si je tape `&bidule` , qu'est-ce que j'obtiens ?

- [x] L'adresse de bidule
- [ ] La valeur de bidule
- [ ] La valeur de la variable sur laquelle pointe bidule

Le & permet d'obtenir l'adresse.

Si on ne met pas le &, on obtient la valeur de bidule.

*Enfin, si on met une étoile * devant bidule, on obtient la valeur de la variable située à l'adresse que contient bidule.*

Question 5

Par quelle valeur doit-on initialiser un pointeur ?

- [ ] NOTHING
- [ ] 1
- [x] NULL
- [ ] MAIN
- [ ] 0_ADDRESS

Si, lorsque vous créez votre pointeur, vous ne savez pas quelle valeur il va prendre, initialisez-le à NULL. NULL est une sorte de constante qui sert à indiquer "Pas d'adresse".

Que vaut NULL ? Ça dépend de votre OS. Souvent, NULL vaut 0. Toutefois, il vaut toujours mieux initialiser à NULL plutôt qu'à 0, comme ça en lisant le programme vous verrez de suite qu'il s'agit d'un pointeur. NULL est à réserver aux pointeurs donc.

Question 6

Soit le code suivant :

```
int nombre = 8;
int *pointeur = &nombre;
```

On suppose que nombre se trouve à l'adresse 5000, et pointeur à l'adresse 2500. Si dans la suite de mon programme je demande à afficher *pointeur, quelle valeur cela affichera-t-il ?

- [ ] 5000
- [ ] 2500
- [x] 8
- [ ] Impossible à prédire

*Comme le pointeur vaut l'adresse de nombre, si on écrit *pointeur on obtient la valeur de nombre, soit 8.*

Question 7

À quel indice commence un tableau ?

- [x] 0
- [ ] 1
- [ ] -1

Un tableau commence toujours à l'indice 0, c'est-à-dire à `tableau[0]`.
N'oubliez jamais ceci, on a naturellement tendance à penser qu'un tableau commence à l'indice 1 alors que ce n'est pas du tout le cas.

Question 8

Laquelle de ces lignes crée un tableau de 10 double ?

- [ ] `double* tableau[10];`
- [ ] `double tableau[10];`
- [x] `double tableau[10];`
- [ ] `double tableau[9];`

Il faut indiquer la taille du tableau entre crochets (ici 10).

Question 9

Ce programme a un défaut. Mais lequel ?

```
int main(int argc, char *argv[])
{
 char ville[100];

 printf("Dans quelle ville habites-vous ? ");
 scanf("%s", &ville);
 printf("Vous habitez %s, je connais bien cette ville !", ville);

 return 0;
}
```

- [ ] Il manque un & devant la variable "ville" dans le printf
- [ ] Il manque une * devant la variable "ville" dans la déclaration de la variable.
- [x] Il y a un & en trop devant "ville" dans le scanf.

Attention aux confusions. Comme je vous l'ai expliqué, un tableau peut être considéré comme un pointeur. Si on écrit juste "ville", on écrit donc un pointeur, c'est-à-dire l'adresse du tableau.
Or, la fonction scanf attend justement une adresse pour savoir où écrire le nom de la ville en mémoire. Vous devez donc lui donner "ville" (qui est l'adresse du tableau), et non "&ville".

Question 10

Quelle directive de préprocesseur indique la fin d'un #if ?

- [ ] #elif
- [x] #endif
- [ ] #ifdef
- [ ] #ifndef

#elif = "else if"
#endif = "Fin du if"
#ifdef = "si la constante a été définie"
#ifndef = "si la constante n'a pas été définie"

Question 11

Une structure peut-elle contenir des tableaux ?

- [x] Oui
- [ ] Non
- [ ] Seulement si tous ces tableaux sont de même type

Une structure peut contenir des tableaux sans problème. Ils n'ont pas besoin d'être de même type. Une structure peut en effet être composée de plusieurs types de variable différents, comme je vous l'ai dit au début du chapitre.

Question 12

Pourquoi est-il préférable d'utiliser une fonction pour initialiser ses structures ?

- [x] Cela permet d'éviter de changer toutes les initialisations de variables dans le programme si la structure change de forme
- [ ] C'est plus rapide pour l'ordinateur
- [ ] Ça transforme mon code source en C++

L'intérêt d'utiliser des fonctions pour initialiser est justement de "centraliser" les initialisations, au cas où la structure change de forme dans le futur. Si la structure comporte des éléments en plus (ou en moins), il n'y aura qu'à changer la fonction pour initialiser correctement toutes les variables du programme.

Cette technique n'est pas plus rapide pour l'ordinateur (et elle n'est pas plus lente non plus). Elle permet juste une meilleure organisation.

Question 13

Que faut-il toujours faire *juste après* l'ouverture d'un fichier ?

- [ ] Le fermer
- [x] Tester la validité du pointeur de fichier
- [ ] Le vider de son contenu
- [ ] Placer le curseur à la position 0

Rien ne vous garantit que le fichier ait été ouvert correctement avec la fonction fopen. Le fichier n'existe peut-être plus, ou il a été renommé, ou bien encore il est utilisé par un autre programme au moment de l'ouverture.
Pour cette raison, vous devez toujours tester la validité du pointeur de fichier après l'ouverture. Si l'ouverture a réussi, le pointeur est différent de NULL. Si elle a échoué eh bien... le pointeur vaut NULL.

Question 14

Que se passe-t-il si je fais l'opération suivante ?

```
malloc(sizeof(int) * 25);
```

- [ ] Cela réserve de la mémoire pour un int de 25 octets
- [x] Cela réserve de la mémoire pour un tableau d'int de 25 cases
- [ ] Cela réserve de la mémoire pour un float de 25 octets
- [ ] Cela réserve de la mémoire pour un tableau de float de 25 cases

L'argument que vous devez envoyer à malloc, c'est la taille de l'espace mémoire que vous voulez réserver.

`sizeof(int)` indique "Nombre d'octets que prend UN int en mémoire".
Multiplier ça par 25, et vous obtenez "25 fois l'espace mémoire d'un int".

Or, 25 int d'affilée c'est... un tableau de 25 int !

Question 15

Qu'est-ce qu'un buffer overflow ?

- [x] Un dépassement de la capacité prévue pour stocker une chaîne
- [ ] Une boucle infinie lors de la lecture du buffer
- [ ] Une fuite de mémoire lorsqu'on perd le pointeur sur une chaîne

Un buffer overflow est un dépassement de mémoire. Les données en trop vont "écraser" en mémoire d'autres données importantes, pouvant amener à un plantage de votre programme ou, plus grave, à une faille de sécurité.

APPRENEZ À EXPLIQUER LES POINTEURS

INSTALLATION DE LA SDL

Le professeur

Mathieu Nebra
Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms ou)

Découvrez aussi ce cours en...

Livre PDF