C Certified Senior Programmer – CLS

Objectifs

- À l'issue de la formation, le participant sera en mesure de :
  - Maîtriser la chaîne de production d'un programme écrit en langage C
  - Mettre en œuvre les opérateurs, les expressions et les structures de contrôle du langage C
  - Manipuler des structures de données, des tableaux, des pointeurs et des chaînes de caractères
  - Organiser le code d'un programme à l'aide de fonctions
  - Exploiter les principales bibliothèques standard du langage C

Prérequis

- Connaissances de base en programmation.

Programme

Premiers pas en C

- Présentation du langage C, ses atouts.
- Le C++ par rapport au C. Normes C++11 et C11.
- Les fichiers sources (.c, .h).
- Structure générale d'un programme.
- La syntaxe de base du langage.
- Les types de données et les constantes de base.
- Variables globales et locales.
- Stockage et passage de paramètres.
- Entrées/sorties formatées.
- Les commentaires.
- Utilisation élémentaire de la chaîne de production.
- Les environnements d'édition, de compilation et d'exécution.
- Exécution d'un premier programme.

Opérateurs et expressions

- Opérateurs arithmétiques.
- Mécanismes d'évaluation des expressions.
- Post et pré-incrémentation de décrémentation.
- Précédence et associativité des opérateurs.
- Opérateurs d'affectation.
- Mécanismes de fonctionnement des expressions logiques.
- Expressions logiques dans les instructions while, if...
- Opérateurs de comparaison : <, >, ==, !=...
- Opérateurs logiques : ET, OU, négation.
- Les types numériques composés. Règle de conversion dans les expressions mixtes. Conversions implicites/explicites.
- Initialisation des variables.

- Arithmétique sur les adresses.
- Formats d'entrée/sortie associés aux types numériques.
- Opérateurs bit à bit : ET, OU, OU exclusif, complément à 1, négation. Opérateurs de décalage : >>, <<.
- Expression conditionnelle avec l'opérateur ternaire.

Travaux pratiques
Mise en œuvre des opérateurs et expressions.

Structures de contrôle

- Notion de blocs.
- Les structures de boucles : while, for.
- Instructions de contrôle de boucles : break, continue.
- Structures de choix : if, else, else if.
- Structure de choix multiple : switch.

Travaux pratiques
Mise en œuvre des structures de contrôle.

Tableaux, pointeurs et chaînes de caractères

- Définition, initialisation et accès aux éléments d'un tableau.
- Définition d'un pointeur. Récupérer l'adresse mémoire d'un objet. Accéder au contenu d'un pointeur.
- Equivalences pointeurs/tableaux.
- Calculs sur les pointeurs.
- Chaînes de caractères.
- Exemples de manipulation de chaînes de caractères.
- Les chaînes de caractères Unicode de C11.

Travaux pratiques
Manipulation de tableaux, de pointeurs et des chaînes de caractères.

Les structures

- Intérêts des structures.
- Déclarer, initialiser et accéder aux champs d'une structure.
- Utiliser des structures imbriquées.
- Créer de nouveaux types en utilisant Typedef.
- Les champs de bits.
- Les unions.
- Les énumérations.
- Les structures et énumérations anonymes de C11.
- Définir des pointeurs sur structures.

Travaux pratiques
Implémentation de nouvelles structures de données.

Les fonctions

- Définition d'une fonction.
- Appel d'une fonction.
- Passage de paramètres : par valeur ou par référence.
- Code retour d'une fonction. Les types de retour.
- La fonction "main".

Travaux pratiques
Découper son code à l'aide de fonctions. Gérer les appels de fonctions.

Compilation séparée, classe d'allocation

- Mécanisme de fonctionnement de la chaîne de production.
- Utilisation de bibliothèque de sources.
- Notion de Makefile.
- Configuration mémoire d'un programme C (pile, tas...).
- Classes d'allocation des variables (auto, register, static, extern).
- Différents cas de figure de la compilation séparée.
- Notion d'objet externe.
- Cas des données globales et statiques.
- Cas des données locales.
- Règle de visibilité.
- Compléments sur les fonctions et les initialisations.

Le préprocesseur

- Utilisation des macros prédéfinies (constantes symboliques). Définir ses propres macros avec #define.
- Définir des macros comme des fonctions. Utilisation des marqueurs # et ##.
- Annuler la définition de constante avec #undef.
- La compilation conditionnelle : #if, #ifdef, #ifndef, #elif, #endif.
- Inclure des ressources avec #include.

Travaux pratiques
Utilisation des directives du préprocesseur. Mise en place de la compilation conditionnelle.

Les bibliothèques standard

- Les fonctions de calcul mathématique (sqrt, sin...).
- Les fonctions d'entrées/sorties (fprintf, fscanf...).
- Les fonctions d'accès aux fichiers (fread, fwrite...).
- Les fonctions de manipulation de chaînes de caractères (strlen, strcat...).
- Les fonctions de gestion de la mémoire (malloc, free...).
- Mise en place de structures chaînées (listes chaînées, arbres n-aire...).
- Les fonctions "sécurisées" de la librairie standard C11 (strcat_s, strlen_s, ...).