Anti-sèche pour la programmation en C

Voici les concepts du langage C structurés par thématique et avec des exemples pour chaque concept:

Inclusion de bibliothèques

Pour utiliser les fonctions de bibliothèques standard, il faut les inclure avec la directive #include.

```
#include <math.h>
#include <string.h>
```

Opérateurs en C

Les opérateurs sont utilisés pour effectuer des opérations sur des variables et des valeurs. Les opérateurs courants en C sont:

- Opérateurs arithmétiques : +, -, *, /, %
- Opérateurs de comparaison : ==, !=, >, <, >=, <=
- Opérateurs d'affectation : =, +=, -=, *=, /=, %= Exemple:

```
int a = 10;
int b = 20;
int somme = a + b;
int difference = a - b;
```

Fonctions en C

Une fonction est un bloc de logique réutilisable qui peut avoir un ensemble défini d'entrée et de sortie. Voici quelques concepts liés aux fonctions en C:

Fonctions intégrées en C

Le langage de programmation C est livré avec des fonctions intégrées de la bibliothèque standard, telles que:

- printf()
- scanf()
- rand()
- srand()

Appel de fonctions

En C, une fonction est appelée en mentionnant le nom de la fonction suivi de parenthèses. Une ou plusieurs valeurs d'argument peuvent être placées entre les parenthèses si la fonction nécessite des valeurs d'entrée.

Exemple:

```
int carre(int x) {
    return x * x;
}

int main() {
    int resultat = carre(5);
}
```

Signature de fonction

Une fonction définie par l'utilisateur est définie à l'aide d'une signature de fonction. Cette signature spécifie le type de retour et le nom de la fonction suivi des paramètres entre parenthèses.

Exemple:

```
int carre(int x) {
    return x * x;
}
```

Type de retour void

Une fonction qui ne retourne aucune valeur doit utiliser le mot-clé void comme type de retour dans la signature de la fonction.

Exemple:

```
void afficherBonjour() {
    printf("Bonjour !\n");
}
```

Valeur de retour de la fonction

Une fonction définie par l'utilisateur peut retourner une valeur avec le mot-clé return suivi de la valeur à retourner. Le type de la valeur retournée doit correspondre au type de retour spécifié dans la signature de la fonction.

Exemple:

```
int carre(int x) {
    return x * x;
}
```

Structures de contrôle

Les structures de contrôle en C permettent de gérer le flux d'exécution du programme. Les exemples courants sont les boucles for, while et les instructions conditionnelles if, else if, else.

Boucle for

La boucle for est utilisée pour répéter un bloc de code un certain nombre de fois. Elle est définie avec une initialisation, une condition et une mise à jour des variables de contrôle.

Exemple:

```
for (int i = 0; i < 10; i++) {
    printf("Iteration %d\n", i);
}
```

Boucle while

La boucle while permet de répéter un bloc de code tant qu'une condition donnée est vraie.

Exemple:

```
int i = 0;
while (i < 10) {
    printf("Iteration %d\n", i);
    i++;
}
```

Instruction if, else if, else

Les instructions if, else if et else permettent d'exécuter des blocs de code en fonction de conditions spécifiques.

Exemple:

```
int note = 75;

if (note >= 90) {
    printf("Excellente performance\n");
} else if (note >= 70) {
    printf("Bonne performance\n");
} else {
    printf("Performance insuffisante\n");
}
```

Instruction break

L'instruction break est utilisée pour sortir d'une boucle (for, while, do-while). Dans cet exemple, break est utilisé pour sortir de la boucle while lorsque la variable compteur atteint 5.

```
int compteur = 0;
while (1) {
    printf("Compteur : %d\n", compteur);
    compteur++;
    if (compteur == 5) {
        break;
    }
}
```

Gestion des entrées utilisateur

Pour gérer les entrées utilisateur en C, on utilise généralement la fonction scanf().

Exemple:

```
int age;
printf("Entrez votre age: ");
scanf("%d", &age);
```

Formatage des chaînes de caractères

La fonction printf() permet de formater des chaînes de caractères avec des spécificateurs de format pour afficher des variables. Les spécificateurs de format courants sont:

- %d : entier
- %f : flottant
- %lf : double
- %c : caractère
- %s : chaîne de caractères Exemple:

```
int age = 25;
printf("J'ai %d ans.\n", age);
```

Fonction srand

La fonction srand() est utilisée pour initialiser le générateur de nombres aléatoires. Elle prend un argument appelé graine (seed), qui est un nombre entier. En initialisant le générateur avec une graine spécifique, les nombres aléatoires générés seront les mêmes à chaque exécution du programme.

Cependant, pour générer des nombres aléatoires différents à chaque exécution du programme, il est courant d'utiliser la fonction time() pour fournir une graine unique basée sur l'horloge du système.

Exemple:

```
#include <stdlib.h>
#include <time.h>

int main() {
    // Initialiser le générateur de nombres aléatoires avec une graine basée sur le temps actuel
    srand(time(NULL));

    // Générer un nombre aléatoire entre 0 et 99
    int nombre_aleatoire = rand() % 100;

    printf("Nombre aléatoire : %d\n", nombre_aleatoire);
}
```

Dans cet exemple, le générateur de nombres aléatoires est initialisé avec srand(time(NULL)), ce qui signifie que la graine utilisée pour générer les nombres aléatoires est basée sur le temps actuel. Ainsi, les nombres aléatoires générés seront différents à chaque exécution du programme.