SAS – Langage de Programmation C

d 01

Résumé: Ce document est le sujet du module d 01 de SAS - Langage C de YouCode.

Table des matières

Chapitre I

Préambule

Ce chapitre présente plusieurs concepts de base, notamment les directives de préprocesseur, les fonctions, les variables et les instructions, dont nous aurons besoin pour écrire les programmes les plus simples.

Pour commencer, nous verrons un petit programme en C et nous décrirons comment le compiler et le lier. Ensuite, nous verrons comment généraliser ce programme, puis comment ajouter des remarques explicatives, appelées commentaires. La section suivante présente les variables, qui stockent des données pouvant changer pendant l'exécution d'un programme, et montre comment utiliser la fonction.

`scanf` pour lire des données dans ces variables. Les constantes — des données qui ne changeront pas pendant l'exécution du programme — peuvent recevoir des noms. Enfin, nous examinerons les règles de C pour la création de noms (identifiants) et les règles de mise en page d'un programme.

Pour ce projet, vous devez trouver des solutions par vous-même pour atteindre les objectifs d'apprentissage fixés.

Vous ne pourrez pas atteindre les objectifs de ce projet, ni des suivants, en copiant-collant le travail de quelqu'un d'autre.

Chapitre II

Faire une recherche sur les aspects

Q 01: Que signifie GCC ?

Q 02: OK, alors que signifie GNU ?

Q 03: C'est quoi le problème avec GCC, au fait ?

Q 04: Comment GCC est-il bon pour trouver des erreurs dans les programmes ?

Q 05: Pourquoi C est-il si laconique ? J'ai l'impression que les programmes seraient plus lisibles si C utilisait `begin` et `end` au lieu de `{` et `}`, `integer` au lieu de `int`, et ainsi de suite.

Q 06: Dans certains livres sur le C, la fonction `main` se termine par `exit(0)` au lieu de `return 0`. Est-ce la même chose ?

Q 07: Que se passe-t-il si un programme atteint la fin de la fonction `main` sans exécuter un `return` ?

Q 08: Le compilateur supprime-t-il un commentaire entièrement ou le remplace-t-il par un espace blanc ?

Q 09: Comment puis-je savoir si mon programme a un commentaire non terminé ?

Q 10: Est-il légal d'imbriquer un commentaire à l'intérieur d'un autre ?

Q 11: D'où vient le nom du type `float` ?

Q 12: Pourquoi les constantes à virgule flottante doivent-elles se terminer par la lettre `f` ?

Chapitre III

Exercices

Section 1.1

1. Créez et exécutez, le célèbre programme "hello, world" de Kernighan et Ritchie :

```
#include <stdio.h>
int main(void)
{
        printf("hello, world\n");
        return 0;
}
```

Question : Obtenez-vous un message d'avertissement du compilateur ? Si oui, que faut-il faire pour le faire disparaître ?

Section 1.2

2. Considérez le programme suivant :

```
#include <stdio.h>
int main(void)
{
        printf("Parkinson's Law:\nWork expands so as to ");
        printf("fill the time\n");
        printf("available for its completion.\n");
        return 0;
}
```

Questions :

(a) Identifiez les directives et les instructions dans ce programme.

(b) Quelle est la sortie du programme ?

Section 1.4

3. Condensez le programme dweight.c en remplaçant les affectations de height, length, et width par des initialisations et (2) en supprimant la variable weight, en calculant plutôt (volume + 165) / 166 dans le dernier printf.

3. Écrivez un programme qui déclare plusieurs variables int et float — sans les initialiser — puis affiche leurs valeurs. Y a-t-il un schéma dans les valeurs ? (Il n'y en a généralement pas.)

```
/* Computes the dimensional weight of a 12"x10"x8" box */
#include <stdio.h>

int main(void) {
        int height, length, width, volume, weight;

        height = 8;
        length = 12;
        width = 10;

        volume = height * length * width;
        weight = (volume + 165) / 166;

        printf("Dimensions: %dx%dx%d\n", length, width, height);
        printf("Volume (cubic inches): %d\n", volume);
        printf("Dimensional weight (pounds): %d\n", weight);
        return 0;
}
```

Section 1.7

5. Lesquels des éléments suivants ne sont pas des identifiants C légaux ?

(a) 100_bottles

(b) _100_bottles

(c) one__hundred__bottles

(d) bottles_by_the_hundred_

6. Pourquoi n'est-il pas judicieux pour un identifiant de contenir plus d'un trait de soulignement adjacent (comme dans current___balance, par exemple) ?

7. Lequels des éléments suivants sont des mots-clés en C ?

(a) for

(b) If

(c) main

(d) printf

(e) while

Section 1.8

8. Combien de jetons y a-t-il dans l'instruction suivante ?

answer=(3*q-p*p)/3;

9. Insérez des espaces entre les jetons de l'exercice 8 pour rendre l'instruction plus facile à lire.

10. Dans le programme dweight.c (Section 1.4), quels espaces sont essentiels ?

Chapitre IV

Projets de programmation

1. Écrivez un programme qui utilise `printf` pour afficher l'image suivante à l'écran :

```
1.          *
2.         *
3.        *
4.   *    *
5.    * *
6.     *
```

2. Écrivez un programme qui calcule le volume d'une sphère de 10 mètres de rayon, en utilisant la formule v = 4/3πr3 . Écrivez la fraction 4/3 comme `4.0f/3.0f`. (Essayez de l'écrire comme `4/3`. Que se passe-t-il ?)
   Indice : C n'a pas d'opérateur d'exponentiation, vous devrez donc multiplier r par lui-même deux fois pour calculer r^3.
3. Modifiez le programme du projet 2 pour qu'il demande à l'utilisateur d'entrer le rayon de la sphère.
4. Écrivez un programme qui demande à l'utilisateur d'entrer un montant en dollars et en cents, puis affiche le montant avec une taxe de 5% ajoutée :

```
Entrez un montant : 100.00
Avec taxe ajoutée : $105.00
```

5. Écrivez un programme qui demande à l'utilisateur d'entrer une valeur pour x, puis affiche la valeur du polynôme suivant :

   $3x^5+2x^4–5x^3–x^2+7x–6$

   Indice : C n'a pas d'opérateur d'exponentiation, vous devrez donc multiplier x par lui-même de manière répétée pour calculer les puissances de x. (Par exemple, x * x * x est x au cube).

6. Modifiez le programme du projet 5 pour que le polynôme soit évalué en utilisant la formule suivante :

   ((((3x+2)x–5)x–1)x+7)x–6

   Notez que le programme modifié effectue moins de multiplications. Cette technique d'évaluation des polynômes est connue sous le nom de règle de Horner.

7. Écrivez un programme qui demande à l'utilisateur d'entrer un montant en dollars américains, puis montre comment payer ce montant en utilisant le plus petit nombre de billets de 20$, 10$, 5$ et 1$:

```
Entrez un montant en dollars : 93
Billets de 20$ : 4
Billets de 10$ : 1
Billets de 5$ : 0
Billets de 1$ : 3
```

Indice : Divisez le montant par 20 pour déterminer le nombre de billets de 20$ nécessaires, puis réduisez le montant de la valeur totale des billets de 20$. Répétez l'opération pour les autres montants de billets. Assurez-vous d'utiliser des valeurs entières, pas des nombres à virgule flottante.

8. Écrivez un programme qui calcule le solde restant d'un prêt après le premier, le deuxième et le troisième paiement mensuel :

```
Entrez le montant du prêt : 20000.00
Entrez le taux d'intérêt : 6.0
Entrez le paiement mensuel : 386.66
Solde restant après le premier paiement : $19713.34
Solde restant après le deuxième paiement : $19425.25
Solde restant après le troisième paiement : $19135.71
```

Affichez chaque solde avec deux chiffres après la virgule.

Indice : Chaque mois, le solde est diminué du montant du paiement, mais augmenté du solde multiplié par le taux d'intérêt mensuel. Pour trouver le taux d'intérêt mensuel, convertissez le taux d'intérêt entré par l'utilisateur en un pourcentage et divisez-le par 12.