TP Introduction – Prise en main du langage C

Partie I : Créer et compiler un programme

I.1 quelques commandes système à connaître

Tapez votre login (nom d'utilisateur), puis votre mot de passe.

Faire apparaître un terminal.

Commandes système à connaître :

- cd (permet de changer de répertoire courant)
- ls
- pwd
- rm
- mkdir
- cp

Que signifient ces commandes ? Pour le savoir tapez :

man *nomCommande*

Exemple : **man** *rm*

Ensuite (toujours dans le terminal) :

- lister le contenu du répertoire.
- Afficher le nom du répertoire courant
- Créer un répertoire nommé algoProg, et un sous-répertoire nommé TP1
- Se placer dans ce sous-répertoire
- On va créer quelques fichiers (vides) ; pour cela, on peut utiliser la commande touch (exemples touch toto créé le fichier vide nommé « toto »
- Pour écrire un programme C, il faut taper le programme dans un éditeur de texte (kate, xemacs, nedit, edit...).

Dans l'éditeur, tapez le programme suivant, puis sauvegardez sous le nom minimal.c

```
int  main(void)
{
  printf("Un premier programme.\n");
  return 0 ;
}
```

I.2 Compiler et exécuter un programme

Compilez (appel au compilateur " gcc ") ce fichier en tapant : ***$ gcc -c minimal.c***
Quel fichier obtient-on ?

Créez un programme exécutable à partir du programme objet en tapant :
$ ***gcc minimal.o -o minimal***

*Utilisez la commande : " **$ gcc -E minimal.c > minimal.i** " afin de n'effectuer que la phase de précompilation.*

*Modifiez le programme source en y incluant en première instruction :**#include <stdio.h>**.*
*Exécutez à nouveau la commande « **$ gcc -E minimal.c > minimal_1.i** » et regardez les différences avec le fichier minimal.i.* Que constatez-vous ? Exécutez le programme.

Partie II : Mon premier programme en C

II.1 Déclarations de variables

On souhaite écrire un programme C de gestion des mouvements d'argent sur un compte bancaire. Chaque compte est identifié par un *numéro* à 6 chiffres. Il héberge une somme d'argent, appelée *solde*. Le solde ne peut être négatif que si le compte a une autorisation de *découvert*. Toute transaction correspond au transfert d'une *somme*. On distingue les *dépôts* et les *retraits*.

- Donnez l'ensemble des variables utilisées dans ce programme, correspondant aux concepts en italique dans le texte, en précisant leur type et écrivez l'instruction C qui permet de les déclarer.

II.2 Initialisation et affectations

On se limite pour l'instant aux variables correspondant au *solde*, à la *somme* transférée et au type de transfert (*dépôt* ou *retrait*).

1. Ecrivez un programme C complet dans lequel les variables ont été déclarées et initialisées avec les valeurs suivantes :
   - Le compte présente un solde positif de 120,5 euros.
   - Le transfert est un *retrait* de 12,0 euros.

   Le programme doit afficher le solde du compte.

2. Ajoutez une instruction qui modifie le type d'opération pour effectuer un dépôt.

3. Ajoutez une instruction qui modifie le montant de la transaction à 35,5 euros.

II.3 Expressions arithmétiques

Pour compléter notre programme, nous allons devoir écrire des expressions arithmétiques, c'est-à-dire des opérations sur les variables numériques. Nous utiliserons les opérateurs suivants du langage C :

- + pour l'opérateur d'addition, - pour l'opérateur de soustraction,
  Attention : l'opérateur - peut être, selon le contexte, unaire ou binaire. Unaire, il s'agit de l'opérateur qui rend l'opposé, par exemple -3 ou -*solde*. Binaire, il s'agit de la soustraction.
- * pour la multiplication, / pour la division.
  Attention : l'opérateur / correspond à une division entière si les deux expressions sont de type entier. Ainsi, 3/2 vaut 1 alors que 3/2.0 vaut 1.5.
- % pour le modulo (reste de la division entière).

1. Complétez le programme de l'exercice précédent de manière à effectuer le retrait de 12 euros, puis le dépôt de 35,5 euros. La valeur du solde devra être affichée après chaque opération.
2. À l'issue de ces deux opérations, le compte est rémunéré avec un intérêt de 1,5%. Ecrivez une instruction qui modifie la valeur du solde.

II.4 Expressions logiques

Nous allons avoir besoin d'écrire des expressions logiques, par exemple pour tester la validité d'une transaction. Nous utiliserons les opérateurs suivants du langage C :

- ! pour l'opérateur de négation,
- && pour la conjonction (l'opérateur ET),
- || pour la disjonction (l'opérateur OU),
- <, <=, > et >= pour les comparaisons,
- == pour le test d'égalité (à ne pas confondre avec = qui désigne l'affectation d'une valeur à une variable),
- != pour la différence.

1. Ecrivez l'expression logique qui prend la valeur *vrai* s'il s'agit d'un retrait et *faux* sinon.

2. Ecrivez l'expression logique qui prend la valeur *vrai* s'il s'agit d'un dépôt et *faux* sinon.

3. Ecrivez l'expression logique qui prend la valeur *vrai* si la somme transférée est positive ou nulle, et *faux* sinon.

4. Un retrait ne pourra être réalisé que si le solde du compte bancaire est suffisant. Déterminez l'expression qui prend la valeur *vrai* s'il s'agit d'un retrait que le solde du compte permet de réaliser et *faux* sinon.

Partie III : Exercices de programmation

Implémenter les en langage C les exercices de la feuille du TD1.