TP Soutien : Bases de Programmation

Exercice 1 (Découverte de Premier Language)**.**
De exercices en ligne sont disponibles pour vous aider à réviser les bases. Le but de cet exercice est de vérifier que vous savez les utiliser (vous ferez ensuite ces exercices par vous-même en dehors des séances). Si vous savez déjà comment accéder aux exercies Premier Language et les utiliser, passez à l'exercice suivant. Sinon, voici les instructions :

Connectez-vous sur ecampus. Allez sur le cours *Introduction à l'informatique-L1 STS : portail MPI* puis allez dans la section « Révisions pour l'examen ». Testez rapidement les premiers exercices dans la fiche *Découverte de PremierLanguage*, puis passez à la suite du TP.

Exercice 2 (Serveur JupyterHub)**.**
Vous aurez besoin de programmer quelques heures par semaine en dehors des séances de TP. Pour cela, vous aurez besoin d'un ordinateur à votre disposition avec tous les outils appropriés, que ce soit un portable, un fixe chez vous, ou tout simplement une des machines en libre service de l'Université (salle 215, sauf si cours).

Pour pouvoir travailler depuis chez vous sans avoir à installer de logiciels spécifiques, vous pouvez utiliser le serveur `JupyterHub` de l'université. En salle de TP par contre, vous devez travailler en local pour ne pas surcharger le serveur.

Si vous savez déjà utiliser le serveur `JupyterHub` de l'université pour écrire et compiler des programmes .cpp, passez à l'exercice suivant. Sinon, voici les instructions :

Connectez-vous sur `https://jupytercloud.lal.in2p3.fr/`

Une fois connecté, vous pouvez cliquer à droite sur `New -> Terminal` qui ouvre un terminal dans un nouvel onglet. Depuis ce terminal, vous pourrez téléchargez les archives de TP avec la commande `wget` (ce qui permettra de récupérer les fichiers vierges pour recommencer un exercice depuis le début). Si vous voulez continuer chez vous un programme commencé en salle de TP, il faudra avoir pensé avant de quitter la salle de TP à charger sur le serveur le fichier correspondant à ce programme. Pour cela, depuis la page d'accueil, cliquer sur `Upload` et sélectionner le fichier auquel vous voulez pouvoir avoir accès en dehors des salles de TP, puis cliquer sur `Téléverser`.

Pour vous entraîner, faites-le avec l'un des programmes de la séance de soutien précédente, puis compilez et exécutez ce programme depuis le terminal du serveur JupyterHub (si vous ne vous souvenez plus des commandes pour compiler et exécuter, vous pouvez les trouver dans l'énoncé du TP de soutien précédent).

Enfin il faut se déconnecter proprement du serveur. Pour cela, ne pas utiliser le bouton `Logout` mais utilisez `Control Panel -> Stop My Server`, puis fermez la page.

Exercice 3.
Lancer un terminal (raccourci en bas à gauche), et dans ce terminal allez dans votre dossier Soutien à l'aide de la commande `cd`. Si vous n'avez pas encore de dossier Soutien, commencez-par en créer un avec `mkdir`. Une fois dans votre dossier Soutien, vous pouvez créer un fichier et l'ouvrir avec `jedit` en une seule ligne de commande, en tapant `jedit programme.cpp &`

Déclarer une variable `n` de type entier. On veut lire une valeur donnée par l'utilisateur à l'aide de `cin` et stocker le résultat dans `n`. On veut ensuite diviser n par 2 s'il est pair et lui enlever 1 s'il est impair, et stocker le résultat dans une variable `resultat`. On affichera ensuite le résultat obtenu à l'aide de `cout`. Écrire le programme correspondant et le tester.

Exercice 4.

(1) Écrire un programme qui prend en entrée l'âge d'une personne et qui affiche `true` si elle est majeure et `false` sinon.

(2) Modifier votre programme pour que l'affichage soit une chaîne de caractères (type `string`) contenant le tarif SNCF appliqué à cette personne : `"enfant"` (moins de 11 ans), `"jeune"` (12-27 ans), `"senior"` (60 ans ou plus) ou `"plein tarif"` sinon.

(3) Tester votre programme avec plusieurs valeurs pertinentes.

Exercice 5 (logarithme entier en base 2)**.**

(1) Écrire un programme dont l'entrée est un entier positif `n` et qui calcule le plus petit i de la forme 2^k tel que $n \leq i$.
Indication : partir de `i=1` et le multiplier par 2 tant que nécessaire.

(2) Modifier votre programme pour que la sortie soit `k` tel que $i = 2^k$ (où i est l'entier défini à la question précédente).

Exercice 6.

Lors de la fin d'un semestre dans une Université X, les enseignants sont amenés à calculer la moyenne générale des notes de physique et de mathématiques selon une règle précise : la meilleure note des trois épreuves de mathématiques est comptée coefficient 3, et la meilleure note des deux épreuves de physique est comptée coefficient 2 ; les autres notes ne comptent pas.

Un enseignant d'informatique est chargé de concevoir un algorithme prenant en entrée les trois notes de mathématiques et les deux notes de physique, et donnant la moyenne générale suivant la règle énoncée ci-dessus.

(1) Spécifier et écrire une fonction `max` qui prend deux entiers `a` et `b` en paramètres et retourne le plus grand nombre des deux nombres.

(2) De même, étant donnés trois entiers, écrire une fonction calculant le plus grand des trois. On pourra utiliser la fonction `max(a,b)`.

(3) Ecrire une fonction qui prend en entrée trois notes de mathématiques, puis deux notes de physique, et calcule la moyenne selon la règle spécifiée.

Exercice 7.

On considère une machine qui distribue des sucreries. Le problème consiste à écrire le programme qu'elle exécute pour rendre la monnaie sur une somme, à l'aide de pièces de 50 centimes, 20 centimes, 10 centimes et 5 centimes d'euro, de façon à minimiser le nombre de pièces rendues sachant que l'on connaît le prix et la somme donnée par le client. On suppose que les sommes sont données en centimes d'euro, qu'il n'y a pas de risque de pénurie de pièces de monnaie, et que les prix sont un multiple de 5 centimes.
Par exemple si le prix à payer est de 110 centimes et que le client a donné 200 centimes, il faut lui rendre 1 pièce de 50 centimes et 2 pièces de 20 centimes.

(1) Quelles sont les entrées / sorties du problème ?

(2) Écrire un programme pour résoudre le problème. Les variables `prix` et `somme` contiennent respectivement le prix et la somme et seront lues à l'aide de `cin`. À la fin du programme, le programme devra afficher le nombre de pièces de 50 centimes, le nombre de pièces de 20 centimes, le nombre de pièces de 10 centimes et le nombre de pièces de 5 centimes à rendre.

Exercice 8.
Voici une fonction qui calcule la surface d'un rectangle :

```
float surfaceRectangle(float longueur, float largeur) {
    return longueur * largeur;
}
```

(1) Implanter une **fonction** `surfaceDisque` qui calcule la surface d'un disque de rayon donné. On prendra $\pi = 3.1415926$. Appeler cette fonction dans la fonction main, en lui donnant comme paramètre un rayon lu à l'aide de `cin`.

(2) Implanter une **fonction** `surfaceTriangle` qui calcule la surface d'un triangle de base b et de hauteur h. Appeler cette fonction dans la fonction main, en lui donnant comme paramètres une base et une hauteur lues à l'aide de `cin`.

Instructions itératives avec compteur : boucles `while` et `for`

Exercice 9.
Voici des exemples de programmes pour afficher les nombres de 1 à 10 avec respectivement une boucle `while` et une boucle `for` :

```
    int i = 1;
    while ( i <= 10 ) {
        cout << i << endl;   // Affiche la valeur de i
        i = i + 1;
    }
```

```
    for ( int i = 1; i <= 10; i = i + 1 ) {
        cout << i << endl;   // Affiche la valeur de i
    }
```

(1) Adapter le premier exemple pour affiche les nombres pairs inférieurs ou égaux à 10, en commençant par 0. De même avec le deuxième exemple.

(2) Même chose pour afficher les nombres de 10 à 1 ;

(3) Même chose pour afficher les nombres entiers de carré inférieur à 10 ;

(4) Même chose pour calculer la valeur de la somme $1^2 + 2^2 + \cdots + 10^2$.

À votre avis, dans chacun des cas ci-dessus, laquelle des deux formes est la plus naturelle ?

Exercice 10 (Nombres premiers)**.**

(1) Écrire une fonction qui prend en argument (entrée) un entier n et *teste* si n est un nombre premier (c'est-à-dire renvoie `true` si n est premier et `false` sinon).

(2) Écrire une fonction qui prend en argument un entier n et affiche tous les nombres premiers entre 1 et n.

(3) Écrire une fonction qui affiche les n premiers nombres premiers.