Algo et POO
TD1
Lic Info 3 – Univ Lumière Lyon 2

Exercice 1

Vous trouverez ci-dessous un exemple de fragment de code qui utilise plusieurs fonctionnalités présentées en CM. Les variables x, y et z sont toutes des variables à virgule flottante en double précision.

```
double x, y, z;
if ((x > y) || (x < 5.0))
{
  z = 4.0;
}
else
{
  z = 2.0;
}
```

1. Quelle valeur ce fragment de code attribuerait-il à la variable z lorsque les variables x et y prennent les valeurs suivantes :

   (a) x=10.0 et y=-1.0 ;
   (b) x=10.0 et y=20.0 ; et
   (c) x=0.0 et y=20.0

2. Modifier le code ci-dessus pour que la condition x > y soit remplacée par x ≥ y.

Exercice 2

Les exercices ci-dessous nécessitent tous une modification du code suivant. Dans tous les cas, utilisez une vérification appropriée pour vous assurer que votre code est correct.

```
#include <iostream>

int main(int argc, char[] argv*){
  double p, q, x, y;
  int j;

  return 0;
}
```

1. Donner à la variable x la valeur 5 si p est supérieur ou égal à q, ou si la variable j est différente de 10.
2. Fixer la variable x à la valeur 5 si à la fois y est supérieur ou égal à q, et la variable j est égale à 20. Si cette condition composée n'est pas remplie, on donne à x la même valeur que p.
3. Définir la variable x selon la règle suivante :

$$x = \begin{cases} 0, & p > q, \\ 1, & p \leq q, \text{ and } j = 10 \\ 2, & \text{otherwise} \end{cases}$$

Exercice 3

Dans cet exercice, on vous demande d'écrire et de tester un programme qui additionne une liste de nombres fournis par un utilisateur via std::cin.

1. Écrivez un programme qui calcule la somme d'une collection d'entiers positifs qui sont entrés par l'utilisateur à partir du clavier. Votre programme doit demander à l'utilisateur d'entrer chaque nombre entier suivi de la touche retour, et d'entrer "-1" à la fin de la liste des nombres entiers à ajouter. Notez qu'il n'est pas nécessaire de stocker la liste des entiers : vous pouvez garder la trace de la somme au fur et à mesure que l'utilisateur entre les valeurs.
2. Modifiez votre code de façon à ce qu'il se termine si la somme des entiers entrés jusqu'à ce point dépasse 100.
3. Modifiez votre code pour que, si l'utilisateur a saisi un nombre entier incorrect, il puisse entrer "-2" pour remettre la somme à zéro et recommencer à entrer des entiers.

Exercice 4

Cet exercice utilise les vecteurs et matrices suivants :

$$u=\begin{pmatrix}1\\2\\3\end{pmatrix};\quad v=\begin{pmatrix}6\\5\\4\end{pmatrix};\quad A=\begin{pmatrix}1&5&0\\7&1&2\\0&0&1\end{pmatrix};\quad B=\begin{pmatrix}-2&0&1\\1&0&0\\4&1&0\end{pmatrix}$$

En outre, le vecteur w satisfait à la relation w = u − 3v. Ces vecteurs et matrices sont stockés dans des tableaux à l'aide du programme suivant. Ce programme comprend un code permettant de calculer le vecteur w

```
#include <iostream>

int main(int argc, char* argv[])
{
   double u[3] = {1.0, 2.0, 3.0};
   double v[3] = {6.0, 5.0, 4.0};
   double A[3][3] = {{1.0, 5.0, 0.0},
                     {7.0, 1.0, 2.0},
                     {0.0, 0.0, 1.0}};
   double B[3][3] = {{-2.0, 0.0, 1.0},
                     {1.0, 0.0, 0.0},
                     {4.0, 1.0, 0.0}};

   double w[3];
   for (int i=0; i<3; i++)
   {
      w[i] = u[i] - 3.0*v[i];
   }

   return 0;
}
```

Nous définissons maintenant les vecteurs x, y et z, ainsi que les matrices C et D, tels que

$$x=u-v,\qquad y=Au,\qquad z=Au-v,\qquad C=4A-3B,\qquad D=AB.$$

Développez le programme ci-dessus pour calculer les vecteurs x, y et z et les matrices C et D, en utilisant des boucles lorsque cela est possible. Conseil : veillez à définir des tableaux de taille appropriée pour ces variables. Vérifiez votre réponse en imprimant les résultats et en les comparant avec le calcul direct.

Exercice 5

La méthode de Newton-Raphson est souvent utilisée pour résoudre des équations non linéaires de la forme f(x)=0. Il s'agit d'un algorithme itératif : entant donné une estimation initiale x_0, les itérations successives satisfont :

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0,1,2,3....$$

Cet algorithme peut se terminer lorsque $|x_k - x_{k-1}| < \varepsilon$ pour un certain ε prescrit par l'utilisateur. Dans cet exercice, nous allons appliquer l'algorithme de Newton-Raphson à la fonction $f(x) = e^x + x^3 - 5$, avec une estimation initiale $x_0 = 0$.

1. Écrivez (sur papier) l'itération de Newton-Raphson pour ce choix de f(x).
2. En utilisant une boucle for, et un tableau pour les itérations x_k, écrivez un programme qui implémente l'itération de Newton-Raphson pour k = 1,2,3,...,100. Imprimez la valeur de x_k à chaque itération, et confirmez que l'itération converge au fur et à mesure que k augmente. A ce stade, ne vous souciez pas de terminer l'itération lorsque ε est suffisamment petit.
3. Pensez à une vérification qui peut être effectuée sur les itères x_k, au fur et à mesure que k devient plus grand, qui vous permette d'être sûr que votre solution est correcte. Implémentez ce contrôle dans votre programme.
4. Il n'est pas nécessaire de stocker la valeur de x_k à chaque itération pour mettre en œuvre l'algorithme de Newton-Raphson. Tout ce dont vous avez besoin, c'est de l'itération précédente, x_{k-1}, et de l'itération actuelle x_k. Modifiez votre code pour que le tableau représentant x_k,i=1,2,...,100 soit remplacé par deux variables scalaires, x_{prev} et x_{next}.
5. Modifiez votre code de façon à ce que, à l'aide d'une instruction while, l'itération se termine lorsque $|x_{next} - x_{prev}| < \varepsilon$.