

GIF-1003
PROGRAMMATION AVANCÉE EN C++

Laboratoire #1

Introduction au C++ et à l'environnement de programmation eclipse

Exercice #1. L'environnement Eclipse. Démarrer un projet C++ et utiliser le débogueur gdb.

1. Démarrez votre Interface de Développement (IDE).
2. Créez un nouveau projet C++ puis ajoutez y le fichier *puissance.cpp* présent dans l'archive qui contient cet énoncé.
3. Dans le menu *Project*, sélectionnez la commande *Build Project* ...
4. Placez votre curseur sur la ligne : « cout << "Ce programme calcule x a la puissance y. << endl; », appuyez sur Ctrl+Shift+B (menu Run, commande *Toggle breakpoint*).
5. Dans le menu *Run*, sélectionnez l'item *Debug* et ensuite la commande *Start Debug* (F11).
6. Poursuivez l'exécution du programme en appuyant sur F6 à chaque ligne. **Prenez soin de bien voir les valeurs des variables être modifiées** au fur et à mesure de l'exécution du programme.
7. Une fois rendu à la ligne : « } », la fin du programme, sortez du mode *Debug* et enlevez le *breakpoint*.

Exercice #2. Indiquez si les énoncés suivants sont vrais ou faux. S'ils sont faux, expliquez pourquoi.

1. Les commentaires demandent à l'ordinateur d'afficher le texte situé, après le // lors de l'exécution du programme.
2. Lorsque la séquence de changement de code, \n est produite par cout à la sortie, le curseur est positionné au début de la ligne suivante.
3. Toutes les variables doivent être déclarées avant de pouvoir être utilisées.
4. On doit affecter un type, à toutes les variables lors de leur déclaration.
5. Le C++ considère les variables **nombre** et **NoMbRe** comme identiques.
6. Des déclarations peuvent apparaître n'importe où dans le corps d'une fonction C++.
7. On ne peut utiliser l'opérateur modulo (%) qu'avec des opérandes d'entiers.
8. Les opérateurs arithmétiques *, /, %, + et - possèdent tous, le même niveau de préséance.

Exercice #3. Déterminez ce qui s'affiche lorsque chacune des instructions en C++ suivantes est exécutée. Si aucun affichage ne se produit, répondez « rien ». Supposez que x = 2 et que y = 3

1. cout << x;
2. cout << x+y;
3. cout << "x=" ;
4. cout << "x=" << x;
5. cout << x + y << " = " << y + x;
6. z = x = y;
7. cin >> x >> y;
8. // cout << "x + y = " << x + y;
9. cout << " \n";

Exercice #4. Voir les projets Bug_1 à Bug_5. Compilez les programmes et exécutez-les s'ils compilent correctement. Sinon, corrigez les erreurs. Si vous ne trouvez pas l'erreur, utilisez le débogueur.

Exercice #5 Boucle infinie

Reprendre le fichier puissance.cpp utilisé au premier exercice, enlever l'incrémentation `i++` dans le corps de la boucle dans le programme précédent, recompiler et exécuter ce programme sans débogueur :

Que remarquez-vous?

Exécuter ensuite le programme avec le débogueur pour voir la valeur des variables impliquées au niveau dans la boucle lors de l'exécution (faites CTRL C pour arrêter l'exécution d'un programme).

Exercice #6 Débordement

Considérer le programme suivant :

```
#include <iostream>
using namespace std;
int main(void)
{
    int x = 100;
    int y = 6;
    int i = 0;
    int puissance;
    puissance = x;
    while (i < y-1)
    {
        puissance = puissance * x;
        i++;
    }
    return 0;
}
```

Compilez et déboguez ce programme.

Que constatez-vous?

Pourquoi est-ce ainsi?

Expliquez et proposez une solution pour ne plus avoir de problème.

Exercice #7 Conversion implicite, perte de précision

Soient les déclarations suivantes :

```
int n = 5, p = 9 ;
```

```
int q ;
```

```
float x ;
```

Quelle est la valeur affectée aux différentes variables concernée par chacune des instructions suivantes?

```
q = n < p;
```

```
q = n == p;
```

```
q = n % n + p > n;
```

```
x = p / n;
```

```
x = (float) p / n;
```

```
x = (p + 0.5) / n;
```

```
x = (int) (p + 0.5) / n;
```

```
q = n * (p > n ? n : p);
```

```
q = n * (p < n ? n : p);
```

Exercice #8 Priorité des opérateurs

Considérons l'expression $1.0 / 3 * 3 - 1.0$. Le résultat mathématique exact est 0.

Voici un programme calculant le résultat de cette expression en C++ :

```
#include <iostream>
using namespace std;
int main()
{
    float a = 1.0;
    int b = 3;
    float c = a/b;
    float d = c*b-1.0;
    cout << d << endl;
    return 0;
}
```

Compilez et exécutez ce programme. Que pouvez-vous en conclure sur l'importance de l'ordre des opérateurs en C++ ?

Exercice #9 Perte de précision

Soit l'équation du second degré suivant : $x^2 - 4.0000000x + 3.9999999 = 0$, ses deux racines exactes sont

$r_1 = 2.000316228$ and $r_2 = 1.999683772$

Voici un programme calculant les deux racines en utilisant respectivement des variables de type float et double :

```
#include<iostream>
using namespace std;
int main(void)
{
    float fa = 1.0f;
    float fb = -4.0000000f;
    float fc = 3.9999999f;
    double da = 1.0;
    double db = -4.0000000;
    double dc = 3.9999999;

    // résolution de l'équation en utilisant des variables de type float
    float d1 = fb*fb - 4.0f*fa*fc;
    float sd1 = sqrtf(d1);
    float rf1 = (-fb + sd1) / (2.0f*fa);
    float rf2 = (-fb - sd1) / (2.0f*fa);
    cout.precision(5);
    cout << rf1 << '\t' << rf2 << endl;
    // résolution de l'équation en utilisant des variables de type double
    double d2 = db*db - 4.0*da*dc;
    double sd2 = sqrt(d2);
    double rd1 = (-db + sd2) / (2.0*da);
    double rd2 = (-db - sd2) / (2.0*da);
    cout.precision(5);
    cout << rd1 << '\t' << rd2 << endl;
    return 0;
}
```

Compilez et exécutez les deux programmes.

Que remarquez-vous et que pouvez-vous en déduire quant à ce qui différencie les types float et double?