


# TD 03 – Jeux d'adresses

## Exercice 1 Traduction C++ ↔ Français

## 1.1 Thème

 Traduisez les propositions suivantes du Français vers le C++ :

➡ Déclaration d'une variable « a » de type entier initialisée à 42.

[illegible]

➡ Déclaration d'une référence nommée « ra » de « a ».

A blank sheet of graph paper with a grid of squares. The grid consists of 20 columns and 10 rows of small squares. There are also larger squares formed by groups of four smaller squares (2x2). The paper has a light blue background and the grid lines are thin and grey.

➡ Déclaration d'une référence constante nommée « cra » de « a ».

[illegible]

➡ Déclaration d'une référence nommée « rra » de « ra ».

[illegible]

➡ Affectation de 17 à la référence « ra ».

[illegible]

➡ Affectation de 18 à la référence « rra ».

[illegible]

➡ Déclaration d'une chaîne de caractères nommée « texte » de type **std::string**.

[illegible]

➡ Déclaration d'une référence constante nommée « rTexte » de « texte ».

[illegible]

➡ Déclaration d'une fonction nommée « maFonction » retournant un double et prenant en paramètres une référence constante d'une chaîne de caractères de type **std::string**, une référence d'un tableau de **size\_t** de type **std::vector**, un entier par référence et un **double**.

[illegible]

➡ Déclaration d'un pointeur nommé « p » vers un entier, initialisé à **nullptr**.



➡ Affectation de l'adresse de « a » à « p ».

📄 Déclaration d'un pointeur nommé « pc » vers un entier constant, initialisé à l'adresse de « cra ».

[illegible]

➡ Déclaration d'un pointeur constant nommé « cp » vers un entier, initialisé à l'adresse de « rra ».

[illegible]

➡ Déclaration d'un pointeur constant nommé « cpc » vers un entier constant, initialisé à l'adresse de « ra ».

A blank sheet of graph paper with a grid pattern. The grid consists of 20 columns and 10 rows of small squares. The lines are light gray and evenly spaced. There are no margins or other markings on the page.

- ➡ Affectation de la valeur 28 à la donnée pointée par p.

- ➡ Déclaration d'un pointeur nommé « pp » vers un pointeur vers un entier, initialisé à l'adresse de p.

[illegible]

- ➡ Affectation de la valeur 412 à la donnée pointée par le pointeur pointé par « pp ».

[illegible]


-  En supposant que l'ensemble du code écrit dans cet exercice soit exécuté ligne à ligne, que vaut la variable « a » ?

## 1.2 Version

-  Avec une extrême rigueur, traduisez en Français le code C++ suivant :

```
1. double tab[4] = { 1, 2, 3, 4 };
2. double* p = tab;
3. *p += 3;
4. p += 2;
5. (*p)++;
6. ++p;
7. *p = 2;
```

[illegible]

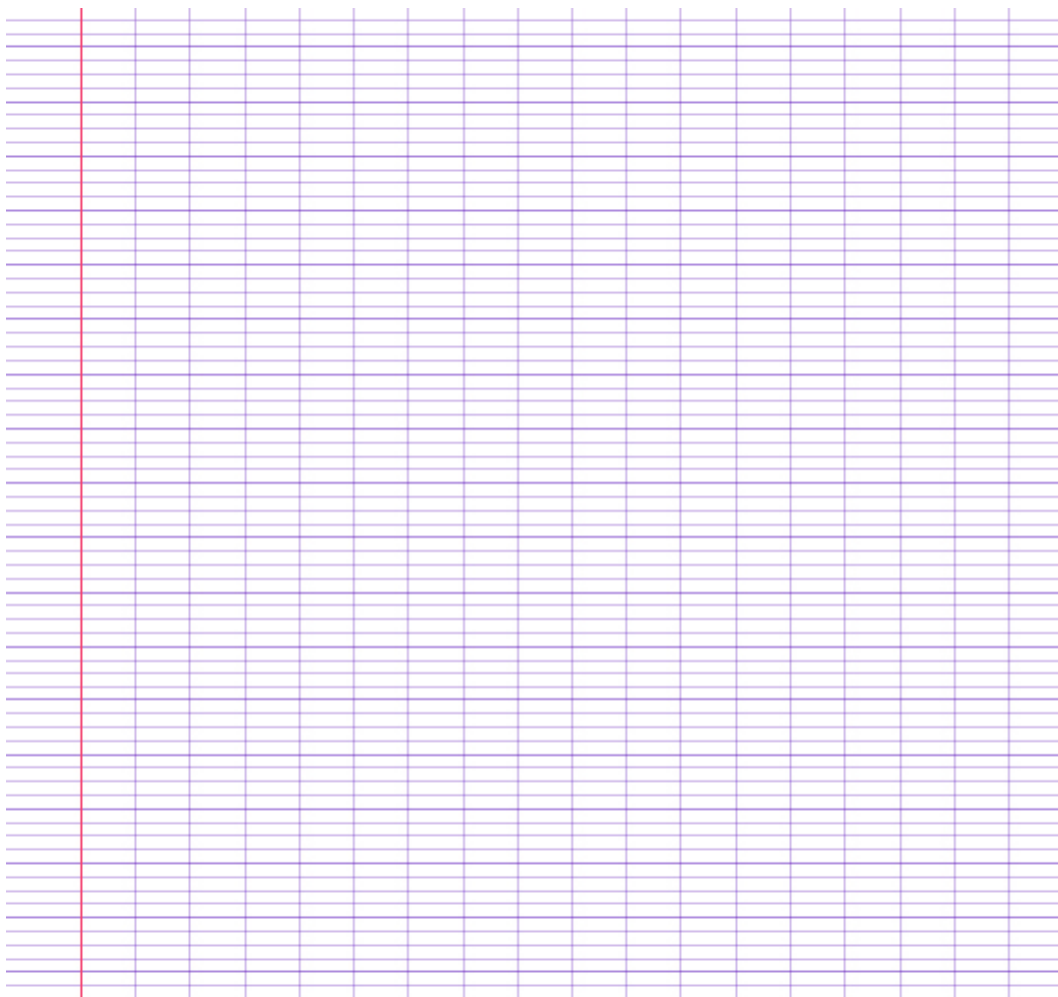
-  A la fin de l'exécution du code précédent, que vaut le tableau `tab` ?

## Exercice 2 Faites la trace des programmes suivants

Pour le programme suivant, on supposera chaque nouvelle variable stockée à la première adresse multiple de 0x1000 libre. C'est-à-dire que la variable **a** est à l'adresse 0x1000, la variable **b** à l'adresse 0x2000, etc... Pour rappel, la taille mémoire d'un **int** est de 4 octets. Le programme est compilé en mode 64-bit (taille des adresses).

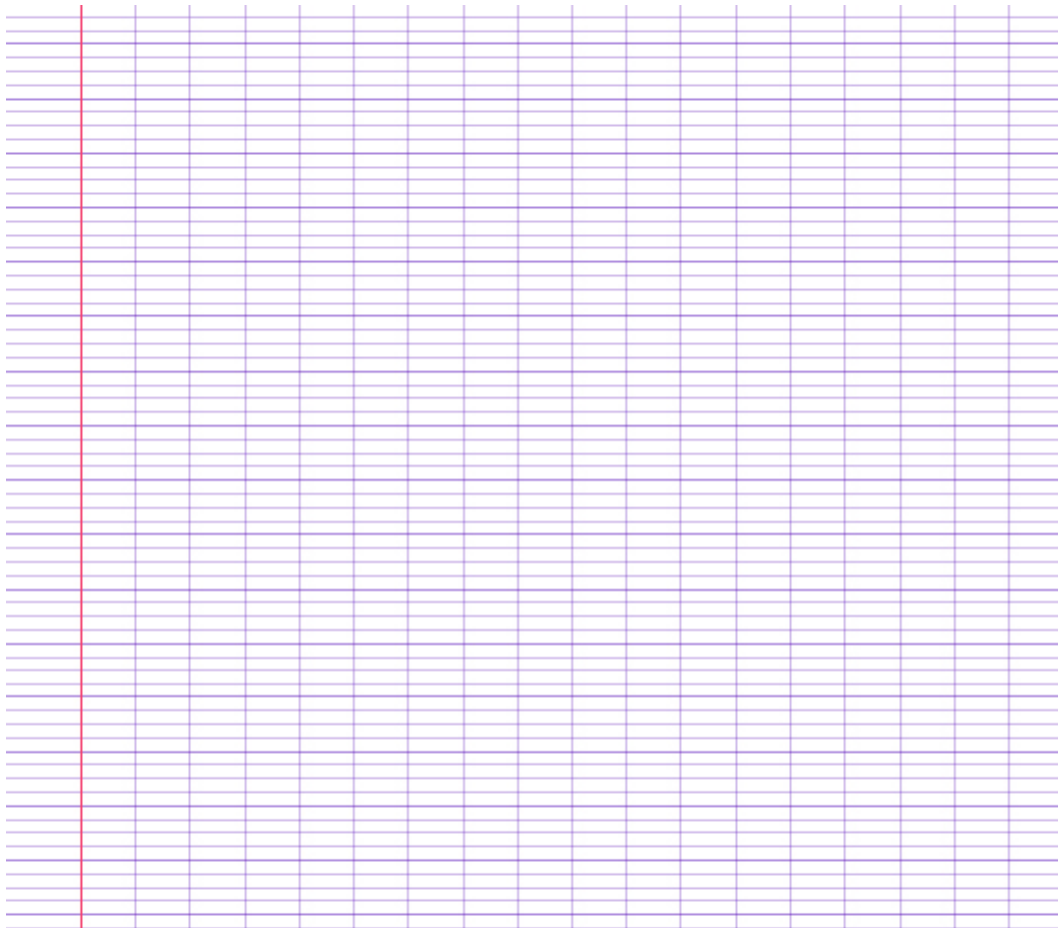
### 2.1

```
1.  int a = 42;
2.  int b;
3.  int c;
4.
5.  int& r1 = a;
6.  int& r2 = b;
7.  int& r3 = c;
8.
9.  int* p1 = &r1;
10. int* p2 = &r2;
11. int* p3 = &r3;
12.
13. int* tab[3] = { p1, p2, p3 };
14.
15. for (int** p = tab + 1; p < tab + std::size(tab); ++p)
16.     **p = **(p - 1) - 1;
```



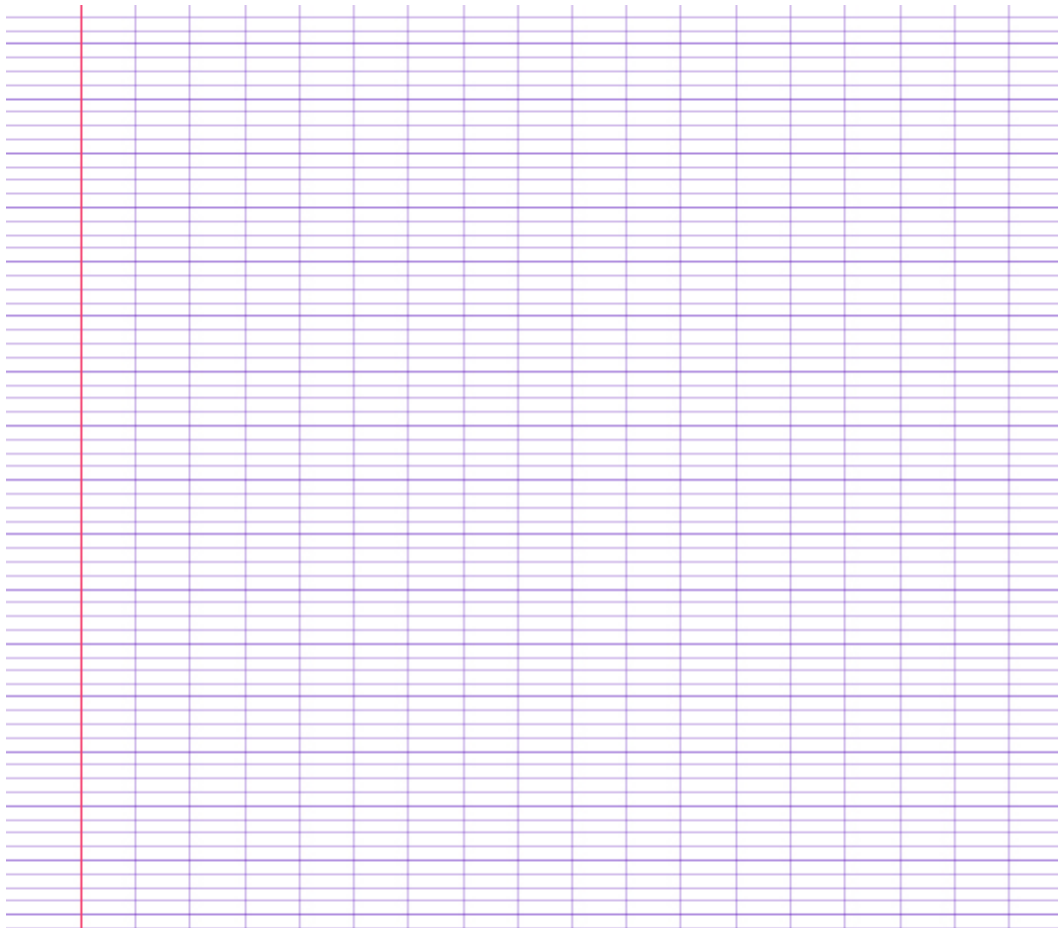
## 2.2

```
1.  bool fonction(int& a, int& b)
2.  {
3.      if (b < a)
4.      {
5.          int c = b; b = a; a = c;
6.          return true;
7.      }
8.      return false;
9.  }
10.
11. int main()
12. {
13.     int tab[4] = { 1, 3, 2, 1 };
14.     bool b = true;
15.     size_t n = 0;
16.     while (b)
17.     {
18.         b = false;
19.         for (size_t i = 1; i < std::size(tab) - n; i++)
20.         {
21.             b |= fonction(tab[i - 1], tab[i]);
22.         }
23.         ++n;
24.     }
25. }
```



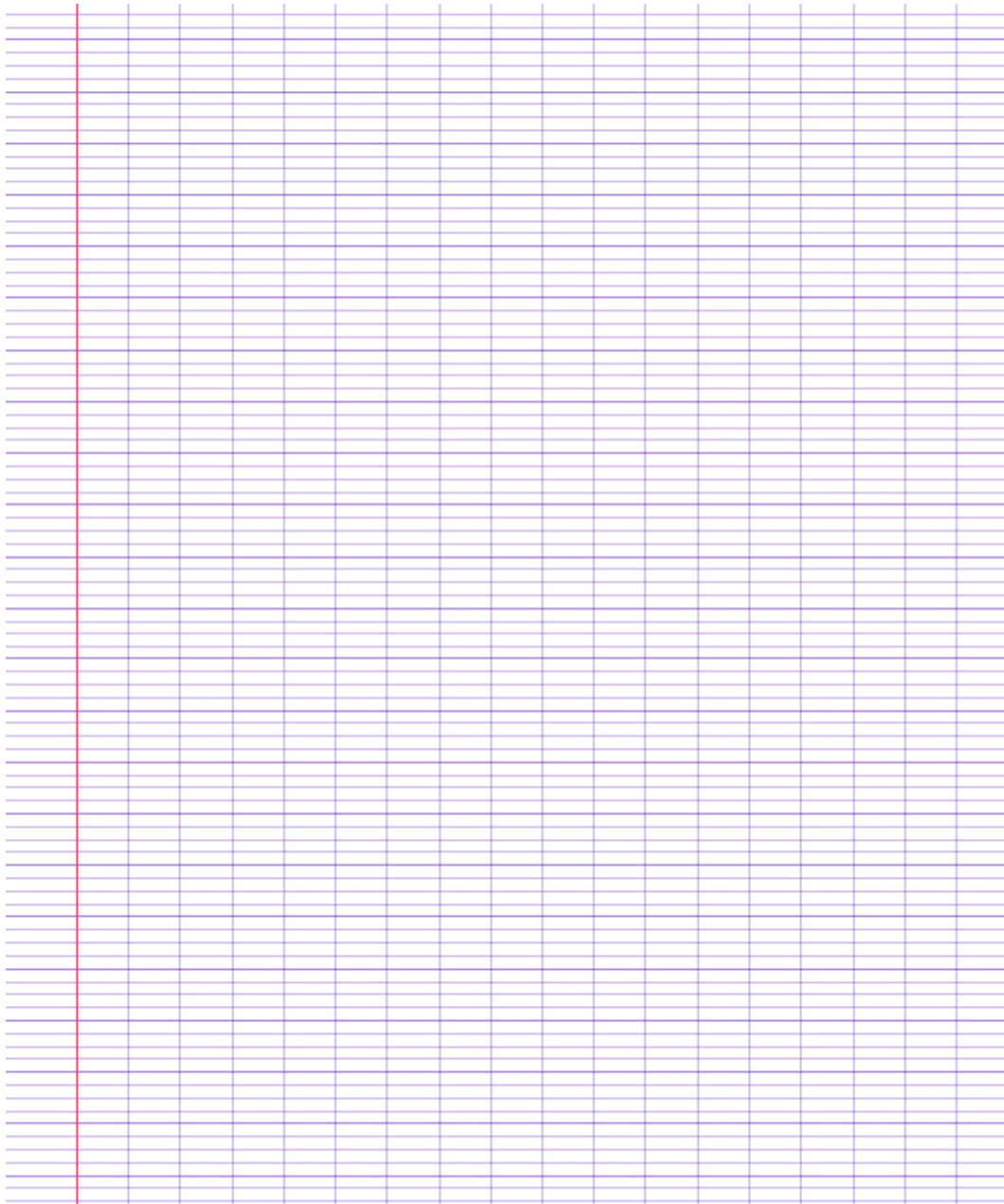
## 2.3

```
1.  bool fonction(int* p)
2.  {
3.      if (*(p + 1) < *p)
4.      {
5.          int c = *(p + 1); *(p + 1) = *p; *p = c;
6.          return true;
7.      }
8.      return false;
9.  }
10.
11. int main()
12. {
13.     int tab[4] = { 1, 3, 2, 1 };
14.     bool b = true;
15.     size_t n = 0;
16.     while (b)
17.     {
18.         b = false;
19.         for (int* p = tab; p != tab + std::size(tab) - n - 1; p++)
20.         {
21.             b |= fonction(p);
22.         }
23.         ++n;
24.     }
25. }
```



## 2.4

```
1. void fonction(char* t)
2. {
3.     while (*t != 0)
4.     {
5.         if (*t >= 'a' && *t <= 'z')
6.             *t += 'A' - 'a';
7.         ++t;
8.     }
9. }
10.
11. int main()
12. {
13.     char texte[] = "Hello !";
14.     fonction(texte);
15. }
```



## 2.5

```
1. void plus(double& a, double b) { a += b; }
2. void moins(double& a, double b) { a -= b; }
3. void fois(double& a, double b) { a *= b; }
4. void divise(double& a, double b) { a /= b; }
5. using PF = decltype(plus)*;
6.
7. int main()
8. {
9.     std::map<char, PF> op;
10.    op['+'] = plus; op['-'] = moins; op['*'] = fois; op['/'] = divise;
11.    std::stringstream ss("12+43*52-4/68");
12.    double res;
13.    ss >> res;
14.    while (!ss.eof())
15.    {
16.        char c; ss >> c;
17.        double o; ss >> o;
18.        op[c](res, o);
19.    }
20.    std::cout << res << '\n';
21. }
```

