# Atelier 1

## Exercice 1

Déclarations : char c = '\x01' (valeur 1), short int p = 10.

1. p + 3

   promotions : short → int  type : int  valeur : 10 + 3 = 13

2. c + 1

   char → int  type : int  valeur : 1 + 1 = 2

3. p + c

   les deux promus en int  type : int  valeur : 10 + 1 = 11

4. 3 * p + 5 * c

   tout en int  type : int  valeur : 3*10 + 5*1 = 30 + 5 = 35

## Exercice 2

Déclarations : char c = '\x05' (5), int n = 5, long p = 1000, float x = 1.25, double z = 5.5.

1. n + c + p
   - n + c → int 5 + 5 = 10 puis conversion vers long pour + p
   - type : long
   - valeur : 1000 + 10 = 1010L
2. 2 * x + c
   - 2 → float 2.0; 2.0 * 1.25 = 2.5
   - c converti en float 5.0
   - résultat float 2.5 + 5.0 = 7.5
   - type : float, valeur 7.5
3. (char) n + c
   - (char)n vaut 5 (puis promu à int dans l'expression)
   - 5 + 5 = 10
   - type : int, valeur 10
4. (float) z + n / 2
   - n / 2 fait une division entière : 5 / 2 = 2
   - (float)z = 5.5f → 5.5 + 2 = 7.5
   - type : float, valeur 7.5

## Exercice 3

Déclarations : int n = 5, p = 9; int q; float x;

1. q = n < p ;
   - 5 < 9 vrai → 1
   - q = 1

2. q = n == p ;
    - 5 == 9 faux → 0
    - q = 0

3. q = p % n + p > n ;
    - attention à la priorité : % puis + puis >
    - p % n = 9 % 5 = 4
    - 4 + p = 4 + 9 = 13
    - 13 > n → 13 > 5 vrai → 1
    - q = 1

4. x = p / n ;
    - division entière 9 / 5 = 1 → convertie en float
    - x = 1.0

5. x = (float) p / n ;
    - (float)p = 9.0 → 9.0 / 5 = 1.8
    - x = 1.8

6. x = (p + 0.5) / n ;
    - p + 0.5 = 9.5 (double) → division flottante 9.5 / 5 = 1.9
    - x = 1.9

7. x = (int) (p + 0.5) / n ;
    - (p + 0.5) = 9.5 → cast (int) donne 9
    - 9 / 5 entier = 1 → x = 1.0

8. q = n * (p > n ? n : p) ;
    - p > n ? n : p → 9 > 5 vrai → choix n (=5)
    - q = 5 * 5 = 25

9. q = n * (p < n ? n : p) ;
    - p < n ? n : p → 9 < 5 faux → choix p (=9)
    - q = 5 * 9 = 45

## Exercice 4 — sortie du programme

A : i = 1 n = 0

B : i = 11 n = 11

C : i = 21 j = 6 n = 120

D : i = 18 n = 18

E : i = 12 j = 4 n = 12

```
A : i = 1 n = 0
B : i = 11 n = 11
C : i = 21 j = 6 n = 120
D : i = 18 n = 18
E : i = 12 j = 4 n = 12


...Program finished with exit code 0
Press ENTER to exit console.
```

## Exercice 5

```
#include <iostream>
using namespace std;

void remplir(int* arr, int n) {
    for (int i = 0; i < n; ++i) {
        cout << "arr[" << i << "] = ";
        cin >> *(arr + i);          // arithmétique de pointeurs
    }
}

void afficher(const int* arr, int n) {
    cout << "[";
    for (int i = 0; i < n; ++i) {
        cout << *(arr + i);
        if (i + 1 < n) cout << ", ";
    }
    cout << "]\n";
}

int& trouverMax(int* arr, int n) {
    // suppose n >= 1
    int* maxp = arr;
    for (int i = 1; i < n; ++i) {
        if (*(arr + i) > *maxp) {
            maxp = arr + i;
        }
    }
    return *maxp; // retourne une référence vers un élément du tableau
}

void inverser_inplace(int* arr, int n) {
    int* left = arr;
    int* right = arr + n - 1;
    while (left < right) {
        int tmp = *left;
        *left = *right;
        *right = tmp;
        ++left;
        --right;
    }
}

void inverser_avec_buffer(int* arr, int n) {
    int* buffer = new int[n];
    for (int i = 0; i < n; ++i) {
        buffer[i] = arr[n - 1 - i];
```

```
    }
    for (int i = 0; i < n; ++i) {
        arr[i] = buffer[i];
    }
    delete[] buffer;
}

int main() {
    cout << "Donner la taille du tableau n (>0) : ";
    int n;
    if (!(cin >> n) || n <= 0) {
        cout << "Taille invalide.\n";
        return 1;
    }

    int* arr = new int[n]; // allocation dynamique

    cout << "Remplissage du tableau :\n";
    remplir(arr, n);

    cout << "Tableau original : ";
    afficher(arr, n);

    // trouverMax et modification via référence
    int& maxRef = trouverMax(arr, n);
    cout << "Maximum actuel = " << maxRef << "\n";

    cout << "Donner une nouvelle valeur pour le maximum (sera affectée directement) : ";
    int newVal;
    cin >> newVal;
    maxRef = newVal; // modifie directement l'élément max dans arr

    cout << "Tableau après modification du maximum : ";
    afficher(arr, n);

    // inverser in-place
    inverser_inplace(arr, n);
    cout << "Tableau inversé (in-place) : ";
    afficher(arr, n);

    // (optionnel) inverser de nouveau en utilisant un buffer et afficher
    inverser_avec_buffer(arr, n);
    cout << "Tableau inversé (avec buffer) : ";
    afficher(arr, n);

    delete[] arr; // libération mémoire
    return 0;
}
```

```
Donner la taille du tableau n (>0) : 10
Remplissage du tableau :
arr[0] = 3
arr[1] = 5
arr[2] = 7
arr[3] = 8
arr[4] = 3
arr[5] = 5
arr[6] = 6
arr[7] = 7
arr[8] = 8
arr[9] = 8
Tableau original : [3, 5, 7, 8, 3, 5, 6, 7, 8, 8]
Maximum actuel = 8
Donner une nouvelle valeur pour le maximum (sera affectée directement) : 100
Tableau après modification du maximum : [3, 5, 7, 100, 3, 5, 6, 7, 8, 8]
Tableau inversé (in-place) : [8, 8, 7, 6, 5, 3, 100, 7, 5, 3]
Tableau inversé (avec buffer) : [3, 5, 7, 100, 3, 5, 6, 7, 8, 8]


...Program finished with exit code 0
Press ENTER to exit console.
```