

INGÉNIEUR EN INFORMATIQUE – ICE4

EXAMEN ÉCRIT – SESSION PRINCIPALE

Module	Programmation sécurisée en C++ moderne	Date	10/01/2026
Enseignant	C.A. ABID	Durée	1h30
Documents	Non autorisés	Nbre de Pages	3

Exercice 1. (8 pts)

1) Pour chacun des programmes suivants, donnez le résultat affiché :

a)

```
const int& mymax(const int &a,const int &b) {
    return (a>=b)?a:b;
}
const int& mymin(const int &a,const int &b) {
    return (a<=b)?a:b;
}
int main() {
    int x{10},y{10};
    const int &max {mymax(x, y)};
    const int &min {mymmin(x, y)};
    x = 11; y = 9;
    std::cout << max << min;
    return 0;
}
```

b)

```
struct show_id {
    ~show_id() {
        std::cout << id;
    }
    int id;
};
int main() {
    delete[] new show_id[3]{ {0}, {1}, {2} };
    return 0;
}
```

2) Modifiez le moins possible les codes suivants pour obtenir exactement le comportement (affichage) attendu :

a) Affichage attendu :

Valeur 0 dans la case 1

Valeur 1 dans la case 2

Valeur 2 dans la case 3

Code à corriger :

```
int T[] {0,1,2};
int i{};
while (i<3) {
    printf("Valeur %d dans la case %d\n",i[T],++i);
}
```

b) Affichage attendu :

c1 est supérieur à c2

Code à corriger :

```
char c1{200},c2{50};
if (c1>c2) {
    std::cout<<"c1 est supérieur à c2\n";
}
else {
    std::cout<<"c2 est supérieur à c1\n";
}
```

Exercice 2. (12 pts)

On désire implémenter une classe template **Optional<T>** modélisant une valeur optionnelle, c'est-à-dire une valeur qui peut être présente ou absente.

Une instance de **Optional<T>** contient soit :

- un objet valide de type T, alloué dynamiquement,
- soit aucune valeur.

La classe gère elle-même la durée de vie de l'objet contenu.

On donne la déclaration suivante :

```
template <typename T>
struct Optional {
    Optional();
    Optional(const T& value);
    ~Optional();

    Optional(const Optional& other) = delete;
    Optional& operator=(const Optional& other) = delete;

    Optional(Optional&& other);
    Optional<T>& operator=(Optional&& other);

    // Accès (comportement indéfini si vide - aucune vérification)
    T& operator*();
    const T& operator*() const;

    bool has_value() const;
    explicit operator bool() const;
    void reset();
private:
    T *m_storage;
};
```

1) Implémenter le constructeur par défaut, le constructeur prenant une valeur et le destructeur.

2) Implémenter la méthode **reset()**.

3) Implémenter les opérateurs **operator\*()** (**const** et non **const**).

4) Implémenter **has\_value()** et l'opérateur de conversion explicite en **bool**.

5) Implémenter le constructeur de déplacement et l'opérateur d'affectation par déplacement en respectant la sémantique de transfert de propriété.

On souhaite maintenant utiliser la classe **Optional<T>** pour implémenter une grille de jeu bidimensionnelle permettant de stocker des pièces (Tic-Tac-Toe, échecs, dames, etc.).

Chaque case de la grille peut être vide ou contenir une pièce de type **T**. La grille possède les objets qu'elle contient. Elle est donc responsable de leur destruction.

Voici la déclaration fournie :

```
template <typename T, std::size_t WIDTH, std::size_t HEIGHT>
struct Grid {
    Grid()=default;
    ~Grid();
    Grid(const Grid& ) = delete;
    Grid& operator=(const Grid&) = delete;
    Grid(Grid&& src);
    Grid& operator=(Grid&& rhs);
    Optional<T>& at(std::size_t x, std::size_t y);
    const Optional<T>& at(std::size_t x, std::size_t y) const;
    constexpr std::size_t getWidth() const { return WIDTH; }
    constexpr std::size_t getHeight() const { return HEIGHT; }
private:
    Optional<T> m_cells[WIDTH][HEIGHT];
};
```

- 1) Implémenter le destructeur de manière à détruire correctement toutes les pièces contenues dans la grille.
- 2) Implémenter le constructeur et l'opérateur de déplacement qui permettent de déplacer les pièces d'une grille à une autre.
- 3) Implémentez la méthode **at()** ainsi que sa version **const**.
- 4) Écrire un court programme illustrant l'utilisation de la classe **Grid<>** :
  - créer une grille de dimension **3x3** de type **std::string**
  - placer le jeton **"X"** dans la case **(1,1)** et le jeton **"O"** dans la case **(2,1)**
  - parcourir la grille et afficher le contenu de chaque case (afficher **"vide"** pour une case ne contenant pas de valeur).

*Bon travail*