" Obtenir une donne d'un tableau à partir de son index

```
Employee Company::getEmployeeByName(string name) const
{
    for (int i = 0; i < nbEmployees_; i++) {
        if (employees_[i].getName() == name) {
            return employees_[i];
        }
    }

    /* Return an unknown employee because we didn't
     * find the one we searched for
     */
    return Employee();
}
```

Augmenter la taille d'une tableau dynamique

```
void Company::addEmployee(Employee &employee)
{
    if (nbEmployees_ >= capacityEmployees_) {
        // Create a new array that double the current capacity, or
        // that is 10 if we are just starting
        int newCapacity = (capacityEmployees_ > 0) ? capacityEmployees_ * 2 : 10;
        Employee* newArray = new Employee[newCapacity];

        // Copy all the previous employees
        for (int i = 0; i < nbEmployees_; i++)
            newArray[i] = employees_[i];

        // Free the previous array
        delete [] employees_;

        // Assign the new one
        employees_ = newArray;
        capacityEmployees_ = newCapacity;
    }

    // Insert new employee
    employees_[nbEmployees_++] = employee;
}
```

Une nouvelle syntaxe pour les if & else

```
int newCapacity;
If (capacityEmployees_>0) {
    newCapacity = capacityEmployees * 2;
}
else{
    newCapacity = 10;
}
```

Effacer une donnée d'un tableau si l'ordre est important

```
void Company::delEmployee(Employee &employee)
{
    for (int i = 0; i < nbEmployees_; i++) {
        if (employees_[i].getName() == employee.getName()) {
            // If the order isn't important
            //employees_[i] = employees_[nbEmployees_-1];

            // If the order is important
            for (int j = i; j < nbEmployees_ - 1; j++) {
                employees_[j] = employees_[j + 1];
            }

            nbEmployees_--;
            break;
        }
    }
}
```