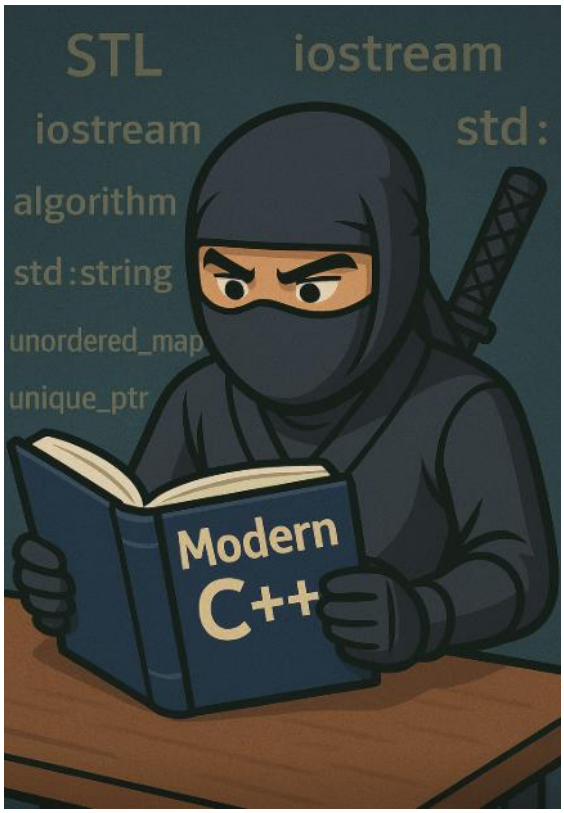
Standard Template Library

Azi ninja-ul nostru a decis că este suficient de capabil să facă pasul cel mare și să intre în rând cu toți programatorii lumii moderne. NullPointer dorește să ia contact cu STL-ul pentru a aplica absolut toate conceptele învățate într-o manieră modernă.



Fiind un băiat călător în timp a decis că vrea să învețe cum să lucreze cu angajații unei companii și să învețe să citească și din fișiere date despre acepe care mai apoi să le stocheze în structuri de date corespunzătoare pe care să aplice diverși algoritmi.

Clasele cu care vom lucra azi sunt Angajat și Departament. Structura clasei Angajat se poate observa mai jos.

```
class Angajat
{
    int varsta;

    double salariu;
    double comision;

    std::string nume;
    std::string prenume;
    std::string functie;

    const int id;
    const int idDepartament;

public:

    Angajat();
```

```
    Angajat(const int& id, const int& idDepartament, const int& varsta, const
double& salariu,
        const double& comision, const std::string& nume, const std::string&
prenume, const std::string& functie);

    Angajat& operator=(const Angajat& angajat);

    const int getIdDepartament() const;
    const std::string getFunctie() const;
    const std::string getNumeComplet() const;

    double getSalariuDeBaza() const;
    double getSalariuComplet() const;

    bool operator>(const Angajat& angajat) const;
    friend std::ostream& operator<<(std::ostream& out, const Angajat& angajat);
};
```

Datele vor fi preluate din fișiere csv care pot fi descărcate chiar de aici. Un departament are un id și o denumire pentru a fi ușor de recunoscut. În general un departament are o listă de angajați.

Cerințe

1) Să se scrie funcții utilitare care pot prelua datele despre departamente și angajați din fișierele puse la dispoziție. (1p)

2) Să se definească clasa Departament și să se pună la dispoziție toate metodele necesare pentru rezolvarea cerințelor următoare. (2p)

3) Să se afișeze în consolă datele colectate din fișiere. (1p)

4) Să se găsească departamentul cu cei mai puțin angajați și respectiv departamentul cu cel mai mare număr de angajați și să se afișeze denumirea fiecărui departament și numărul de angajați. Se va folosi funcția generică **minmax_element** pentru a putea returna cele 2 departamente care să respecte cerința. (1p)

5) Să se ordoneze invers alfabetic angajații din fiecare departament al companiei și să se afișeze întreaga structură a companiei. Se va folosi metoda **sort** care poate sorta o listă de angajați. (1p)

6) Să se calculeze salariul mediu de bază din companie. Se va folosi funcția **accumulate** din fișierul **numeric** pentru a face suma tuturor salariilor din companie. (1p)

7) Să se găsească pentru fiecare departament angajatul salariul maxim complet (cu tot cu bonusuri). Fiecare angajat care respectă această regulă trebuie stocat într-un vector care va fi sortat descrescător dupa salariul maxim complet. Se va folosi funcția **max_element** care va găsi angajatul cu salariul maxim per departament și ulterior îl va stoca într-un vector care va fi sortat după acest salariu. (1p)

8) Să se extragă primii 3 programatori din companie cu cel mai mare salariu de bazaă ordonați alfabetic dupa numele complet și să se afișeze în consolă. (2p)

- se caută departamentul cu denumirea "IT" în companie folosind funcția **find_if**
- se extrage lista de angajați din departamentul IT
- se filtrează pentru a obține doar programatorii folosind **remove_if**
- se sorteaza descrescător dupa salariul de bază (lista de angajați filtrați)
- se extrag primii 3 programatori într-un vector
- se sorteaza alfabetic dupa numele complet respectivul vector