Langage C++

- CIR2 -

ISEN ALL IS DIGITAL! NANTES | yncréa

TP7 : Template

Ayoub KARINE (ayoub.karine@isen-ouest.yncrea.fr)

Partie 1 : Point générique

1. Créer un nouveau projet dans lequel vous implémentez la classe Point du TP6 en utilisant, cette fois-ci, des types numériques génériques.
2. Proposer une fonction test dans le main.cpp pour avoir l'affichage ci-dessous :

```
Point(T,T) : 0x7ffffe214870
(6.9,2.4)
~Point() : 0x7ffffe214870
```

3. Remplacez UNIQUEMENT float par int dans le test déjà proposé. Que constatez-vous ?
4. Créer une fonction template pointTest permettant de ne pas mettre le type en dur dans l'instanciation du point
5. Tester cette fois-ci votre code de la manière suivante :

```
void pointTests(){

    pointTest<int>();
    pointTest<float>();
}
```

Partie 2 : STL

1. Implémentez la fonction printVector() afin que l'exécution de la fonction (utiliser les différentes méthodes vues pendant le cours) :

```
void printNumbers(){
    vector<int> numbers = {1,2,3};
    printVector(numbers);
}
```

Donne l'affichage :
[1,2,3]

2. Modifiez (si nécessaire) la fonction printVector() afin que l'exécution de la fonction :

```
void printVectorTests(){
    printNumbers();
    printLetters();
}
```

Donne l'affichage :
[1,2,3]
[A,B,C]
La fonction printLetters() étant définie par :

```
void printLetters(){
    vector<char> letters = {'A', 'B', 'C'};
    printVector(letters);
}
```

3. Implémentez la fonction isNumber() afin que l'exécution de la fonction :

```
#define boolToString(b) (b ? "true" : "false")

void isNumberTests(){
    cout << boolToString( isNumber("1") )<< endl;
    cout << boolToString( isNumber("42") ) << endl;
    cout << boolToString( isNumber("6PO") ) << endl;
    cout << boolToString( isNumber("R2D2") ) << endl;
}
```

Donne l'affichage :

true
true
false
false

Indice : regarder les fonctions de <cctype> ⇒ lien

4. Implémentez la fonction keepNumbersOnly() afin que l'exécution de la fonction (utiliser la fonction isNumber) :

```
void filterVectorTest(){
    vector<string> values = {"Z", "1", "A", "42", "E", "G", "0"};
    keepNumbersOnly(values);
    printVector(values);
}
```

Donne l'affichage :

[1,42,0]

5. Implémentez la fonction splitString() afin que l'exécution de la fonction :

```
void splitTests(){

    vector<string> operators = splitString("+ - / =");
    printVector(operators);

    vector<string> numbers = splitString("1,2,3,4,5", ",");
    printVector(numbers);

    vector<string> letters = splitString("A-B-C-D", "-");
    printVector(letters);
}
```

Donne l'affichage :

[+,-,/,=]

[1,2,3,4,5]

[A,B,C,D]

Indice : utiliser string::find() et string::substr() et string::push_back

6. Implémentez la fonction printMap() afin que l'exécution de la fonction :

```
void printAges(){
    map</*name*/string, /*age*/unsigned short> ages = {
            { "Emma", 30 },
            { "Manon", 19 },
            { "Sacha", 45 }
    };

    printMap(ages);
}
```

Donne l'affichage :

Emma -> 30

Manon -> 19

Sacha -> 45

7. Modifiez (si nécessaire) la fonction printMap() afin que l'exécution de la fonction :

```
void printMapTests(){
    printAges();
    cout << "---" << endl;
    printRights();
}
```

Donne l'affichage :

Emma -> 30

Manon -> 19

Sacha -> 45

4587 -> 644

58714 -> 777

254756 -> 666

La fonction printRights() étant définie par :

```
void printRights(){
    map</*userId*/int, /*rights*/int> userRights = {
            { 254756, 666 },
            { 4587, 644 },
            { 58714, 777 }
    };
    printMap(userRights);
}
```

8. Implémentez la fonction keepMalesOnly() afin que l'exécution de la fonction :

```
void filterMapTest(){
    map</*name*/string, /*male*/bool> genders = {
            { "Emma", false } ,
            { "Manon", false } ,
            { "Sacha", true } ,
            { "Ambre", false } ,
            { "Gabin", true } ,
            { "Paul", true }
    };
    keepMalesOnly(genders);
    printMap(genders);
}
```

Donne l'affichage :

Gabin -> 1

Paul -> 1

Sacha -> 1

9. Implémentez la fonction splitOnGender() afin que l'exécution de la fonction :

```
void splitMapTest(){
    map</*name*/string, /*male*/bool> genders = {
            { "Emma", false } ,
            { "Manon", false } ,
            { "Sacha", true } ,
            { "Ambre", false } ,
            { "Gabin", true } ,
```

```
            { "Paul", true }
    };
    vector<string> males;
    vector<string> females;
    splitOnGender(genders, males, females);
    printMap(genders);
    cout << "---" << endl;
    printVector(males);
    printVector(females);
}
```

Donne l'affichage :

Ambre -> 0

Emma -> 0

Gabin -> 1

Manon -> 0

Paul -> 1

Sacha -> 1

[Gabin,Paul,Sacha]

[Ambre,Emma,Manon]

10. Implémentez la fonction computeStatistics() afin que l'exécution de la fonction :

```
void statsTest(){
    vector<int> values = {1, 42, -3, 6, 12, 404};
    map<string, float> statistics = computeStatistics(values);
    printMap(statistics);
}
```

Donne l'affichage :

average -> 77

max -> 404

min -> -3

11. Réimplémenter votre fonction en partant du code fourni ci-dessous et en complétant les parties manquantes avec des fonctions définies dans <algorithm> et <numeric>

```
map<string, float> computeStatistics(const vector<int>&
values){
    return {
            { "min" , /*à compléter*/ },
            { "max" , /*à compléter*/ },
            { "average" , /*à compléter*/ },
    };
}
```

Comparez les deux versions en termes de simplicité de code et de performances.