Université de Reims Champagne-Ardenne
UFR Sciences Exactes et Naturelles
Licence Informatique
INFO0402: Programmation Orientée Objet

TD N°8

Exercice 4: Opérations sur une chaine de caractères

On considère une chaine de caractères contenant un texte en français. Les réponses proposées devront utiliser autant que possible les fonctionnalités de string et/ou les algorithmes de la STL.

1. Remplacer les lettres accentuées par les lettres non accentuées correspondantes.
2. Convertir cette chaine en majuscule.
3. Supprimer les espaces dans la chaine.
4. Écrire un code calculant le nombre d'occurrences de chaque lettre.
5. Afficher les valeurs de l'histogramme.
6. Afficher les valeurs de l'histogramme sous forme de barres de texte.
7. Récupérer la liste de tous les caractères à partir du conteneur contenant les fréquences.
8. Même question à partir de la chaîne de caractères obtenue après la question 3.
9. Vérifier si un caractère x fait partie de cette chaine.

Solution:

1. voir code suivant :

```
string str = "...";
map<char, char> cmap = { {'à','a'},{ 'é','e' },{ 'ê','e' } };
// version 1
for (auto &x : str) {
  auto it = cmap.find(x);
  if (it != cmap.end()) x = it->second;
}
// version 2
for (auto &x : cmap)
  replace(str.begin(), str.end(), x.first , x.second);
```

2. voir code suivant :

```
// version 1
transform(str.begin(), str.end(), str.begin(), ::toupper);
// version 2
for (auto &x : str) x = toupper(x);
```

3. voir code suivant :

```
// version 1: à corriger
remove(str.begin(), next(str.end()),' ');
// version 2: correct
str.erase(remove(str.begin(), next(str.end()),' '),str.end());
```

Correction : remove renvoie un caractère sur le dernier caractère remplacé. Comme on réécrit dans la même chaine, il faut supprimer le reste de la chaine (= mettre un '\0').

4. voir code suivant :

```
map<char, int>   hist;
// version 1
for (char &x : str) ++hist[x];
```

note : ceci fonctionne seulement parce lorsqu'une clef n'existe pas, la valeur mappée de type `A` est initialisée par valeur (i.e. utilise le constructeur `A{}` = constructeur par défaut s'il existe, et initialisation à 0 pour les bits).

5. voir code suivant :

```
// méthode 1
for(auto &x : hist) cout << x.first << ":" << x.second << "␣";
cout << endl;
```

méthode 2 : définir une surcharge de « pour la paire (pas de surcharge générique)

```
namespace std {
    ostream& operator<<(ostream &os, const pair<char, int> &val) {
      return os << val.first << ":" << val.second << "␣";
    }
}
```

Obligation de le définir dans le namespace std en raison de l'ADL (argument-dependent lookup) qui dit que si un argument est dans un certain namespace, alors la fonction appelée est recherchée dans ce namespace (ajouter l'ADL au cours).

Exemple :

```
namespace ADL {
  struct A {
    int a;
    A(int u) : a(u) {}
  };
  int funADL(A x) { return x.a + 1; }
}
int main() {
  ADL::A  a1(2);
  int x1 = funADL(a1);
  int x2 = funADL(ADL::A(2));
  // int x3 = funADL(1);  // introuvable
}
```

Dans le cas ci-dessus, la surcharge n'est recherchée que dans le namespace std. Conséquence : il est plus prudent de placer les surcharges pour les opérateurs ou les fonctions de la STL dans le namespace std (sinon, risque de ne pas être trouvée).

Le code obtenu :

```
for(auto &x : hist) cout << x;
// ou
copy(hist.begin(), hist.end(), ostream_iterator<int>(cout, ",␣"));
```

6. pour le max, voir code suivant :

```
// version 1
int vMax = 0;
for (auto &x : hist) if (x.second > vMax) vMax = x.second;
cout << vMax << endl;
```

```
// version 2
auto iMax = max_element(hist.begin(), hist.end(),
  [](const pair<char, int> &x, const pair<char, int> &y)
      { return x.second < y.second; });
cout << iMax->first << ":" << iMax->second << endl;
```

Pour l'histogramme :

```
// version 1
for (auto &x : hist) {
  cout << x.first << ": " << string(40 * x.second / vMax, '*') << endl;
}
```

```
// version 2
for (auto &x : hist) {
    cout << x.first << ": ";
    fill_n(ostream_iterator<char>(cout), 40 * x.second / vMax, '*');
    cout << endl;
}
```

7. voir code suivant :

```
// version: type 1
vector<char>  car(hist.size());
// version: type 2
string       car(hist.size(),0);
// commun
transform(hist.begin(), hist.end(), car.begin(),
          mem_fn(&pair<char, int>::first));
```

8. la liste étant triée, on peut utiliser une recherche par dichotomie. Voir code suivant :

```
if (binary_search(car.begin(),car.end(),x) cout << "found" << endl;
```