TD1 – Standard Template Library

L'objectif de la séance est de s'approprier la syntaxe des patrons de fonctions et de savoir utiliser et manipuler différents conteneurs.

1. Patron de fonction

Soit la déclaration de patron de fonction suivante :

```
template <typename T, typename U> T fct (T a, U b, T c) ;
```

avec les déclarations suivantes :

```
int n, p, q ;
float x ;
char t[20] ;
char c ;
```

```
fct(n,x,q);
p = fct(x,x,x);
c = fct(c,t,c);
fct(x,x,c);
```

```
p = fct(n,p,q);
fct(c,t,q);
t = fct(c,t,c);
```

- Donnez la déclaration correspondant aux appels des fonctions ?
- Sont-ils conformes à la déclaration de la fonction générique ? Justifiez votre réponse.

2. Patron de fonction min()

- Donnez la définition de la fonction générique `min()` qui renvoie la plus petite de 2 variables quelconques en paramètre.
- Proposez un programme de test pour des variables entières et des chaines de caractères.

3. Fils d'exécution

- Représentez l'évolution du conteneur et donnez l'affichage produit par l'exécution du programme suivant :

```
1  #include <iostream>
2  #include <list>
3  #include <algorithm>
4  int main()
5  {
6      std::list<int> laL;
7      laL.push_back(5);
8      laL.push_back(3);
9      std::reverse(laL.begin(), laL.end());
10     for (int leElement :  laL)
11     {
12         std::cout << leElement << std::endl;
13     }
14 }
```

- Représentez l'évolution du conteneur et donnez l'affichage produit par l'exécution du programme suivant :

```
1  void affiche(std::list<int> * uneL)
2  {
3      for(int lElement : *uneL)
4      {
5          std::cout << lElement << "␣";
6      }
7      std::cout << std::endl;
8  }
9
10 int main()
11 {
12     std::list<int> * laL = new std::list<int>();
13     laL->push_back(5);
14     laL->push_front(2);
15     laL->push_back(1);
16     laL->push_back(3);
17     affiche(laL);
18
19     std::list<int >::iterator it;
20     it = std::min_element(laL->begin(), laL->end());
21     std::reverse(laL->begin(), it);
22     affiche(laL);
23
24     std::swap_ranges(laL->begin(),it,it);
25     affiche(laL);
26
27     return 0;
28 }
```

4. Pile ou face

On souhaite écrire un programme de calcul d'expressions arithmétiques simples sur la base de leur description postfixée. Par exemple pour l'expression (12+3)*4, on aura "12" "3" "+" "4" "*". Cette expression est stockée dans un conteneur `<vector>`.

L'algorithme d'évaluation procède de la manière suivante :
1. On empile tout nombre (après conversion) dans la `pile` d'évaluation `<stack>`.
2. On dépile deux nombres (opérandes) de la `pile` pour calcul lorsqu'on a une opération : le résultat est empilé en tête de la `pile`.
3. Le résultat final correspond à la dernière valeur sur la tête de `pile`.
4. On vide les conteneurs à la fin du calcul.

- Donnez la déclaration C++ du tableau conteneur <vector>.

- Donnez la définition d'une fonction de saisie de l'expression postfixée par l'utilisateur avec arrêt de la saisie après appui de la touche « ! ».
- Donnez la définition d'une fonction qui permet de renvoyer le résultat du calcul de l'expression passée en paramètre conformément à l'algorithme décrit ci-dessus.

5. Entraînement course 200 mètres

Un champion sprinteur effectue une course d'entraînement typique de 200 mètres dans les temps suivants : 26.1, 25.6, 25.7, 25.2, 24.7 et 25.0 secondes.

On souhaite connaître le temps minimal, le temps maximal, le temps moyen et le temps médian du champion.

A partir d'un conteneur `<vector>` contenant les différents temps du champion, calculez les temps significatifs demandés* et stockez les différents temps dans une table associative `<map>` puis affichez la médiane.

Vous écrirez la fonction générique pour calculer la somme des éléments du vecteur.

Vous écrirez la fonction qui renvoie la moyenne des temps.

Vous écrirez la fonction qui renvoie la médiane des temps.

*vous utiliserez autant que possible les itérateurs `begin` et `end`.

Calcul de la médiane

Si les observations d'une variable sont ordonnées par valeur, la valeur médiane correspond à l'observation qui se trouve au point milieu de cette liste ordonnée. Elle correspond plus précisément à un pourcentage cumulé de 50 % (c'est-à-dire que 50 % des valeurs sont supérieures à la médiane et 50 % lui sont inférieures). La position de la médiane est la valeur,

$\frac{n+1}{2}$ le n désignant le nombre de valeurs dans un ensemble de données.

Pour calculer la médiane, il faut d'abord classer les données (les trier dans l'ordre ascendant). La médiane est le nombre qui se situe au point milieu. Si le nombre de valeurs est pair il faudra alors faire la moyenne des 2 valeurs contiguës au point milieu.

Annexes

Méthodes de la classe std::stack

`void pop()` : supprime l'élément en tête de stack
`void push(T& value)` : ajoute value en tête de stack
`T& top()` : renvoie l'élément en tête de stack
`unsigned int size()` : donne le nombre d'éléments
`bool empty()` : renvoie true si le stack est vide

Modifiers std ::vector

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_back C++11	Construct and insert element at the end (public member function)

Algorithmes

function template

std::sort <algorithm>

```
default (1)  template <class RandomAccessIterator>  void sort (RandomAccessIterator first, RandomAccessIterator last);
custom (2)   template <class RandomAccessIterator, class Compare>  void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);
```

Sort elements in range

Sorts the elements in the range `[first,last)` into ascending order.

The elements are compared using `operator<` for the first version, and *comp* for the second.

Equivalent elements are not guaranteed to keep their original relative order (see `stable_sort`).

function template

std::stoi C++11 <string>

```
int stoi (const string&  str, size_t* idx = 0, int base = 10);
int stoi (const wstring& str, size_t* idx = 0, int base = 10);
```

Convert string to integer

Parses *str* interpreting its content as an integral number of the specified *base*, which is returned as an `int` value.

If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.

The function uses strtol (or wcstol) to perform the conversion (see strtol for more details on the process).