

Cours n°8

Standard Template Library III

Sommaire

1. Algorithmes
2. Classe string

1 ALGORITHMES

Caractéristiques

Fonctions génériques implémentant les algorithmes les plus connus

Réelle qualité d'optimisation,

Application à tous les conteneurs d'objets

Nécessité de surcharge de certains opérateurs

== pour les algorithmes de recherche

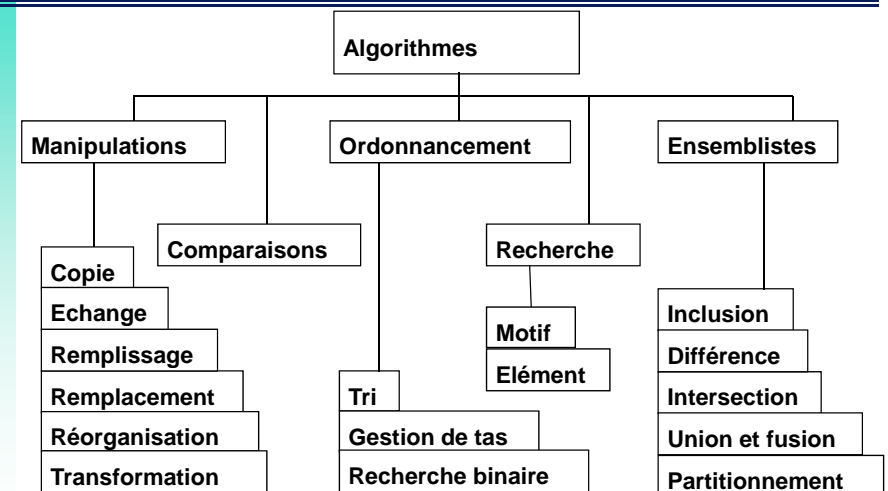
< (ou un autre opérateur de comparaison) pour les algorithmes de tri

<< et >> pour les algorithmes opérant sur les flots

opérateurs arithmétiques pour l'utilisation de foncteurs arithmétiques

1 ALGORITHMES

Schéma



Glossaire

- (UF) UnaryFunction** Foncteur unaire
- (BF) BinaryFunction** Foncteur binaire
- (P) Predicate** Foncteur unaire à valeur booléenne
- (BP) BinaryPredicate** Foncteur binaire à valeur booléenne
- (II) InputIterator** Itérateur unidirectionnel avec accès autorisé en lecture
Itérateur sur un flot d'entrée
- (OI) OutputIterator** Itérateur unidirectionnel avec accès autorisé en écriture
Itérateur sur un flot de sortie
- (FI) ForwardIterator** Itérateur unidirectionnel avec accès autorisé en lecture
et en écriture
- (BI) BidirectionalIterator** Itérateur bidirectionnel avec accès autorisé en
lecture et en écriture

Prototypes

- UF for_each**(II first, II last, UF f)
Application itérative d'un foncteur unaire
- OI transform**(II first, II last, OI result, UF f)
- OI transform**(II first1, II last1, II first2, OI result, BF f)
Application itérative d'un foncteur (unaire ou binaire)
+ sauvegarde du résultat (en place ou dans un conteneur)

testFor_each.cpp

Application itérative du foncteur « Ecrire »

```
template<typename T> class Ecrire {
private:
    int count; ostream& os;
public:
    écrire(ostream& out) : os(out), count(0) { }
    void operator() (T x) { os << x << ' '; ++count; }
    int getcount() { return count; }
};

int main() {
    deque<Date> dd; Date today, dfin;

    cin >> today; cin >> dfin;
    do { ++today; dd.push_front(today); } while (today < dfin);

    Ecrire<Date> ED = for_each(dd.begin(), dd.end(), Ecrire<Date>(cout));
    cout << endl << ED.getcount() << " objets écrits." << endl; }
};
```

```
13 7 2006
14 7 2006
64 objets écrits.
```

Prototypes (1/2)

Recherche d'un élément ou d'un prédicat dans un conteneur non trié

- FI find**(II first, II last, const T& value)
- FI find_if**(II first, II last, P p)
- II find_first_of**(II first1, II last1, FI first2, FI last2, [BP p])
- FI find_end**(FI first1, FI last1, FI first2, FI last2, [BP p])

Recherche d'une séquence d'éléments ou de prédicats dans un conteneur non trié

- FI search**(FI first1, FI last1, FI first2, FI last2, [BP p])
- FI search_n**(FI first, FI last, int count, const T& value, [BP p])

Nombre d'occurrences d'un élément ou de prédicat dans un conteneur non trié

- int count**(II first, II last, const T& value)
- int count_if**(II first, II last, P p)

Recherche d'un élément

```

include "../cours03/Animal.h"
#include <iostream>
#include <list>
#include <functional>

using namespace std;

int main() {
    list<Animal> la;
    la.push_front(*new Animal("oiseau", 2));
    la.push_front(*new Animal("araignee", 8));
    la.push_front(*new Animal("insecte", 6));
    la.push_front(*new Animal("poisson", 0));
    la.push_front(*new Animal("mammifere", 4));

    Animal a(Animal("mammifere", 2));
    // l'opérateur == a du être redéfini pour Animal
    if (find(la.begin(), la.end(), a) == la.end())
        cout << "Je ne connais pas cet animal" << endl;
    else cout << "Je connais cet animal" << endl;
}

```

Je ne connais pas cet animal

Prototypes (2/2)

Recherche d'un élément ou d'un prédicat dans un conteneur ordonné

FI lower_bound(**FI** first, **FI** last, const **T**& value, [**BP** p])
FI upper_bound(**FI** first, **FI** last, const **T**& value, [**BP** p]))
pair<**FI**, **FI**> equal_range(**FI** first, **FI** last, const **T**& value, [**BP** p]))
bool binary_search(**FI** first, **FI** last, const **T**& value, [**BP** p]))

Utilisation par défaut du foncteur less

redéfinition de l'opérateur <

Recherche rapide d'un élément

```

#include "../cours03/Animal.h"
#include <iostream>
#include <set>
#include <functional>

using namespace std;

int main() {
    set<Animal, less<Animal> > sa;
    // l'opérateur < a du être redéfini pour Animal
    sa.insert(*new Animal("oiseau", 2));
    sa.insert(*new Animal("araignee", 8));
    sa.insert(*new Animal("insecte", 6));
    sa.insert(*new Animal("poisson", 0));
    sa.insert(*new Animal("mammifere", 4));

    Animal a(Animal("mammifere", 4));
    if (binary_search(sa.begin(), sa.end(), a) == false)
        cout << "Je ne connais pas cet animal" << endl;
    else cout << "Je connais cet animal" << endl;
}

```

Je connais cet animal

Prototypes (1/2)

Copie d'une séquence d'éléments

OI copy(**OI** first, **OI** last, **OI** result)
OI copy_n(**OI** first, **int** count, **OI** result)
BI copy_backward(**BI** first, **BI** last, **BI** result)

Echange d'éléments ou de séquence d'éléments

void swap(**T**& a, **T**& b)
FI swap_ranges(**FI** first1, **FI** last1, **FI** first2)

Recherche et remplacement d'un élément ou d'un ensemble d'éléments

void replace(**FI** first, **FI** last, const **T**& old_value, const **T**& new_value)
void replace_if(**FI** first, **FI** last, **P** p, const **T**& new_value)
OI replace_copy(**OI** first, **OI** last, **OI** result, const **T**& old_value, const **T**& new_value);
OI replace_copy_if(**OI** first, **OI** last, **OI** result, **P** p, const **T**& new_value)

Copie d'une liste dans un fichier

```
#include "../tp03/Date.h"

using namespace std;
int main() {

    list<Date> ld;
    ld.push_front(*new Date(19, 11, 1703));
    ld.push_front(*new Date(15, 1, 1992));
    ld.push_front(*new Date(5, 5, 1821));
    ld.push_front(*new Date(9, 3, 1796));
    ld.push_front(*new Date(11, 2, 660));
    ld.push_front(*new Date(29, 8, 1842));
    ld.push_front(*new Date(24, 4, 1617));

    // écriture d'une liste dans un fichier
    ofstream floutOut("Cours 08/Date.txt");
    ostream_iterator<Date> out_it(floutOut);
    copy(ld.begin(), ld.end(), out_it);
    floutOut.close();
}
```

Création du fichier Date.txt

Prototypes (2/2)

Recherche et suppression d'un élément ou d'un ensemble d'éléments

FI remove(FI first, FI last, const T& value)

FI remove_if(FI first, FI last, P pred)

OI remove_copy(II first, II last, OI result, const T& value)

OI replace_copy_if(II first, II last, OI result, P p, const T& new_value)

Suppression d'éléments consécutifs

FI unique(FI first, FI last, [BP p])

OI unique_copy(II first, II last, OI result, [BP p])

Mirroir d'une séquence d'éléments

void reverse(BI first, BI last)

OI reverse_copy(BI first, BI last, OI result)

Suppression des noms propres

```
template<class T>class NomPropre {
public:
    bool operator() (const T& s) {
        if ((s.at(0) >= 'A') && (s.at(0) <= 'Z')) return true; else return
        false; }
};

list<string> ls;
// lecture des mots de la phrase
ifstream fEntree("cours08/Phrase.txt");
istream_iterator<string> in_it(fEntree);
copy(in_it, istream_iterator<string>(), back_inserter(ls));
fEntree.close();

list<string>::iterator it, itf;
for (it = ls.begin(); it != ls.end(); it++) cout << *it << " ";
cout << endl;
itf = remove_if(ls.begin(), ls.end(), NomPropre<string>());
for (it = ls.begin(); it != itf; it++) cout << *it << " ";
```

une opération à Madrid , Barcelone , Valladolid et Vigo

une opération à , , et

Prototypes

void sort(RI first, RI last, [BP p])

void stable_sort(RI first, RI last, [BP p])

void partial_sort(RI first, RI middle, RI last, [BP p])

void partial_sort_copy(II first, II last, RI result_first, RI result_last, [BP p])

bool is_sorted(FI first, FI last, [BP p])

Tri d'un fichier d'éléments

```
list<Date> ld;

// lecture des dates
ifstream fEntree("Cours 08/Date.txt");
istream_iterator<Date> in_it(fEntree);
copy(in_it, istream_iterator<Date>(), back_inserter(ld));
fEntree.close();

ld.sort();// tri

for (list<Date>::iterator it = ld.begin(); it != ld.end(); it++)
    cout << *it;
}
```

```
11 2 660
24 4 1617
19 11 1703
9 3 1796
5 5 1821
29 8 1842
15 1 1992
```

Prototypes

Opérations ensemblistes sur les conteneurs ordonnés

bool includes(**II** first1, **II** last1, **II** first2, **II** last2, [**BP** p])

test d'inclusion

OI **set_union**(**II** first1, **II** last1, **II** first2, **II** last2, **OI** result, [**BP** p])

union de deux ensembles

OI **set_intersection**(**II** first1, **II** last1, **II** first2, **II** last2, **OI** result, [**BP** p])

intersection de deux ensembles

OI **set_difference**(**II** first1, **II** last1, **II** first2, **II** last2, **OI** result, [**BP** p])

OI **set_symmetric_difference**(**II** first1, **II** last1, **II** first2, **II** last2, **OI** result, [**BP** p])

Généralités

Utilisation d'un codage ASCII étendue (ISO Latin 1)

Programmes de conversion nécessaire avec le standard Unicode

UTF-8 <-> ISO Latin 1

UTF-16 <-> ISO Latin 1

UTF-32 <-> ISO Latin 1

Algorithme de conversion UTF-8 -> ISO Latin 1

Caractère unicode représentée par un octet (x)

y (Caractère ISO Latin 1) = x

Caractère unicode représentée par deux octets (x1 et x2)

y (Caractère ISO Latin 1) = ((x1 & 0x1F) << 6) + (x1 & 0x3F);

Caractère unicode représentée par trois octets (x1, x2 et x3)

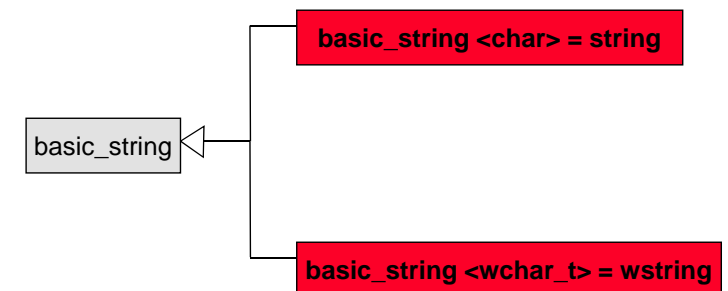
Il ne s'agit pas d'un caractère ISO Latin 1

Exemple

caractère 'ç' de code UTF-8 (C3 A7) a pour code ISO Latin 1 (E7)

Composants

Deux spécialisations du patron de classes « basic_string »



Classe string – Propriétés (1/2)

Constructeurs

Construction d'une chaîne de nb caractères 'car'
string (int nb, char car)

Construction à partir de la chaîne de caractères "s"
string (string s)

Construction à partir d'une partie de la chaîne de caractères "s"
string (string s, int début, int taille)

Redéfinitions d'opérateur

Concaténation

+ **+=**

Ecriture sur un flot de sortie, lecture à partir d'un flot d'entrée

<< **>>**

Comparaisons

< **<=** **>** **>=** **==** **!=**

21

Master Langue et informatique – Programmation générique et conception objet – Claude Montacié

Classe string – Propriétés (2/3)

Méthodes de recherche

Recherche de la première occurrence d'une chaîne ou d'un caractère

int find(string s) **int find(char car)**
int find(string s, int i) **int find(char car, int i)**

Recherche de la dernière occurrence d'une chaîne ou d'un caractère

int rfind(string s) **int rfind(char car)**
int rfind(string s, int i) **int rfind(char car, int i)**

Recherche de la première occurrence d'un des caractères de la chaîne s

int find_first_of(string s) **int find_first_of(string s, int i)**

Recherche de la première occurrence d'un des caractères n'appartenant pas à s

int find_first_not_of(string s) **int find_first_not_of(string s, int i)**

Recherche de la dernière occurrence d'un des caractères de la chaîne s

int find_last_of(string s) **int find_last_of(string s, int i)**

Recherche de la dernière occurrence d'un des caractères n'appartenant pas à s

int find_last_not_of(string s) **int find_last_not_of(string s, int i)**

22

Master Langue et informatique – Programmation générique et conception objet – Claude Montacié

Classe string – Propriétés (3/3)

Méthodes d'insertion, suppression et remplacement

Insertion d'une chaîne "s" ou de nb caractères 'car' à la position i
iterator insert(string s, int i) **iterator insert(int i, int nb, char car)**

Suppression d'une partie de la chaîne
iterator erase(int i, int taille) **iterator erase(int i)**

Remplacement d'une partie de la chaîne par la chaîne "s"
iterator replace(int i, int taille, string s)

Remplacement d'une partie de la chaîne par nb caractères 'car'
iterator replace(int i, int taille, int nb, char car)

23

Master Langue et informatique – Programmation générique et conception objet – Claude Montacié

testString.cpp

Classe string – Exemple

```
string s1(10, 'a'), s2(" la fille du roi zoulou. ");
cout << s1 << s2 << endl;

string s3(s2), s4(s2, 1, 4); cout << s3 << s4 << endl;

cout << s3.find("roi") << endl;

cout << s3.find_first_of("aeiou") << endl;

s2.insert(s3.find("fille"), "grande "); cout << s2 << endl;

s3.replace(1, 8, "le fils"); cout << s3 << endl;
```

```
aaaaaaaaa la fille du roi zoulou.
 la fille du roi zoulou. la f
13
2
 la grande fille du roi zoulou.
 le fils du roi zoulou.
```

24

Master Langue et informatique – Programmation générique et conception objet – Claude Montacié