Hledání v sekvencích

Termín odevzdání:	**21.04.2013 23:59:59**
Pozdní odevzdání s penalizací:	**12.05.2013 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**5.2800**
Max. hodnocení:	**4.0000** (bez bonusů)
Odevzdaná řešení:	6 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	1 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je vytvořit šablonu třídy `CSearch`. Tato šablona bude sloužit pro vyhledávání sekvence dat uvnitř jiné sekvence. Bude použitelná např. pro vyhledání řetězce uvnitř jiného řetězce. Navíc bude umět vyhledávat množinu sekvencí (např. řetězců) v zadané prohledávané sekvenci (např. v prohledávaném řetězci).

Aby byla třída `CSearch` co nejobecnější, bude to šablona, která bude parametrizovaná dvěma generickými parametry:

- Prvním parametrem bude datový typ sekvence `_Type`. Může jím být např. zmíněný řetězec `string`, ale stejně dobře i jiný kontejner z STL, konkrétně specializovaný `vector` nebo `list`. Tedy půjde např. vyhledávat v řetězcích, posloupnosti čísel (`vector<int>` nebo `list<int>`), posloupnosti řetězců (`vector<string>` nebo `list<string>`), ... Pozor, `list` omezuje rozhraní pouze na dopředný iterátor.

  Vlastní hodnoty ukládané v sekvencích (prvky kontejneru) jsou samozřejmě také parametrizované. Pro `string` je to datový typ `char`, pro `vector` a `list` pak libovolný primitivní nebo složený datový typ. Obecně o prvku sekvence můžete předpokládat, že má k dispozici kopírující konstruktor, destruktor, operátor přiřazení a operátory pro porovnání na shodu a neshodu (==, !=). Tyto operace jsou buď poskytované kompilátorem (primitivní datové typy), nebo jsou automaticky vygenerované/naprogramované pro složené datové typy. Žádné další rozhraní nemusí být k dispozici. Pozor, obecně není k dispozici ani implicitní konstruktor.

- Druhým nepovinným generickým parametrem šablony je datový typ porovnávače - komparátor. Jedná se buď o funktor, nebo o funkci. Pokud není parametr zadán, předpokládá se standardní komparátor `std::equal_to` pro datový typ prvku sekvence.

Vlastní třída bude mít následující rozhraní:

- Konstruktor s nepovinným parametrem komparátoru. Vytvoří prázdnou instanci třídy vyhledávače.
- Destruktor, pokud bude potřeba
- Metoda `Add (id, needle)`. Metoda přidá další hledanou sekvenci (`needle`) do seznamu vyhledávaných sekvencí. Sekvence je identifikovaná celočíselnou konstantou `id`.
- Metoda `Search(hayHeap)` zkusí v sekvenci `hayHeap` vyhledat dříve zadané sekvence vložené pomocí metody `Add`. Metoda vrátí množinu všech dříve zadaných sekvencí, které byly v prohledávané sekvenci `hayHeap` nalezené. Vrácená množina bude obsahovat identifikace (`id`) nalezených sekvencí.
- Kopírující konstruktor a přetížený operátor = není testovaný, tedy není potřeba je implementovat. Je vhodné je ale deklarovat jako `private` části rozhraní aby jejich případné použití vyvolalo chybu při překladu.

Odevzdávejte soubor, který obsahuje implementovanou šablonu třídy `CSearch` a další Vaše podpůrné třídy. Třída musí splňovat veřejné rozhraní podle ukázky - pokud Vámi odevzdané řešení nebude obsahovat popsané rozhraní, dojde k chybě při kompilaci. Do třídy si ale můžete doplnit další metody (veřejné nebo i privátní) a členské proměnné. Odevzdávaný soubor musí obsahovat jak deklaraci třídy (popis rozhraní) tak i definice metod, konstruktoru a destruktoru. Je jedno, zda jsou metody implementované inline nebo odděleně. Odevzdávaný soubor nesmí obsahovat vkládání hlavičkových souborů a funkci `main`. Funkce `main` a vkládání hlavičkových souborů může zůstat, ale pouze obalené direktivami podmíněného překladu jako v ukázce níže.

Při řešení úlohy využijte STL. Nepoužívejte ale syntaxi C++0x (`auto`, uzávěry, ...), kompilátor tuto syntaxi ještě nezvládá.

Úloha obsahuje povinné a bonusové testy. V bonusovém testu je třída testovaná pro velké množství hledaných sekvencí, které jsou vyhledávané v dlouhém vstupu. Naivní algoritmus bude velmi pomalý. Pro získání bonusu je potřeba implementovat algoritmus rychlejší, například algoritmus Aho-Corasicková.

Požadované rozhraní třídy:

```
#ifndef __PROGTEST__
#include <cctype>
#include <iostream>
```

```
#include <iomanip>
#include <set>
#include <list>
#include <map>
#include <vector>
#include <queue>
#include <string>
using namespace std;
#endif /* __PROGTEST__ */

template <typename _Type, typename _Comparator = .... >
class CSearch
 {
   public:
    // default constructor
    // constructor with comparator parameter
    // destructor (if needed)
    void     Add     ( int           id,
                       const _Type & needle );
    set<int> Search  ( const _Type & hayHeap ) const;
   private:
    // empty copy constructor
    // empty operator =
 };
```

Ukázka použití:

```
class CharComparator
 {
   public:
              CharComparator ( bool caseSensitive = true )
               : m_CaseSensitive ( caseSensitive ) { }
    bool      operator () ( const char & a, const char & b ) const
               { return m_CaseSensitive ? a == b : toupper (a) == toupper (b); }
   private:
    bool      m_CaseSensitive;
 };

bool upperCaseCompare ( const char & a, const char & b )
 {
   return toupper ( a ) == toupper ( b );
 }

void printSet ( const set<int> & s )
 {
   for ( set<int>::const_iterator it = s . begin (); it != s . end (); ++it )
    cout << ( it == s . begin () ? "" : ", " ) << *it;
   cout << endl;
 }

template <typename _T, int _CNT>
vector<_T> makeVector ( const _T (&data)[_CNT] )
 {
   return vector<_T> ( data, data + _CNT );
 }

...

CSearch <string> test1;
test1 . Add    ( 0, "hello" );
test1 . Add    ( 1, "world" );
test1 . Add    ( 2, "rld" );
test1 . Add    ( 3, "ell" );
test1 . Add    ( 4, "hell" );
printSet ( test1 . Search ( "hello world!" ) );
// 0, 1, 2, 3, 4
printSet ( test1 . Search ( "hEllo world!" ) );
// 1, 2

CSearch <string, bool (*)(const char &, const char &)> test2 ( upperCaseCompare );
```

```
test2 . Add    ( 0, "hello" );
test2 . Add    ( 1, "world" );
test2 . Add    ( 2, "rld" );
test2 . Add    ( 3, "ell" );
test2 . Add    ( 4, "hell" );
printSet ( test2 . Search ( "HeLlO WoRlD!" ) );
// 0, 1, 2, 3, 4
printSet ( test2 . Search ( "hallo world!" ) );
// 1, 2

CSearch <string, CharComparator> test3 ( CharComparator ( false ) );
test3 . Add    ( 0, "hello" );
test3 . Add    ( 1, "world" );
test3 . Add    ( 2, "rld" );
test3 . Add    ( 3, "ell" );
test3 . Add    ( 4, "hell" );
printSet ( test3 . Search ( "heLLo world!" ) );
// 0, 1, 2, 3, 4
printSet ( test3 . Search ( "Well, templates are hell" ) );
// 3, 4

CSearch <vector<int> > test4;
static const int needleA [] = { 1, 6, 1, 6, 9, 12 };
static const int needleB [] = { 9, 12, 7 };
static const int hayHeap [] = { 1, 6, 1, 6, 1, 6, 9, 12, 8 };
test4 . Add    ( 0, makeVector ( needleA ) );
test4 . Add    ( 1, makeVector ( needleB ) );
printSet ( test4 . Search ( makeVector ( hayHeap ) ) );
// 0

CSearch <vector<string> > test5;
static const string needleX  [] = { "Prague", "Bern", "Rome" };
static const string needleY  [] = { "London", "Prague", "Bern" };
static const string needleZ  [] = { "London", "Rome" };
static const string cityHeap [] = { "Berlin", "London", "Prague", "Bern", "Rome", "Moscow" };
test5 . Add    ( 0, makeVector ( needleX ) );
test5 . Add    ( 1, makeVector ( needleY ) );
test5 . Add    ( 2, makeVector ( needleZ ) );
printSet ( test5 . Search ( makeVector ( cityHeap ) ) );
// 0, 1
```

Referenční řešení

6	**08.04.2013 13:35:16**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	5.2800	

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test s parametry podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.001 s (limit: 6.000 s)
    - Využití paměti: 12840 KiB (limit: 18243 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.505 s (limit: 5.999 s)
    - Využití paměti: 15084 KiB (limit: 18243 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Efektivita - rychlost': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 2.103 s (limit: 8.000 s)
    - Využití paměti: 16604 KiB (limit: 25079 KiB)
    - Úspěch v bonusovém testu, hodnocení: 120.00 %
  - Celkové hodnocení: 120.00 % (= 1.00 * 1.00 * 1.20)
- Použité nápovědy: 1

- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 120.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 1.20 * (4.00 + 0.40) = 5.28

5	**07.04.2013 11:27:09**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - Chyba při základní kompilaci **[Zpřístupnit nápovědu (1409 B)]**
- Použité nápovědy: 1
- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 0.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 0.00 * (4.00 + 0.40) = 0.00

4	**07.04.2013 11:22:27**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - Chyba při základní kompilaci **[Zpřístupnit nápovědu (1409 B)]**
- Použité nápovědy: 1
- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 0.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 0.00 * (4.00 + 0.40) = 0.00

3	**07.04.2013 11:20:10**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test s parametry podle ukazky': Neúspěch
    - Dosaženo: 0.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 6.000 s)
    - Využití paměti: 12728 KiB (limit: 18243 KiB)
    - Neúspěch v závazném testu, hodnocení: 0.00 %
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (65 B)]**
    - Nesprávný výstup **[Zpřístupnit nápovědu (57 B)]**
  - Celkové hodnocení: 0.00 %
- Použité nápovědy: 1
- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 0.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 0.00 * (4.00 + 0.40) = 0.00

2	**06.04.2013 20:35:57**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - Chyba při základní kompilaci **[Zpřístupnit nápovědu (1393 B)]**
- Použité nápovědy: 1
- Penalizace za vyčerpané nápovědy: Není (1 <= 2 limit)
- Celkové procentní hodnocení: 0.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 0.00 * (4.00 + 0.40) = 0.00

1	**06.04.2013 20:15:49**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - ☐ Chyba při základní kompilaci
- Celkové procentní hodnocení: 0.00 %
- Bonus za včasné odevzdání: 0.40
- Celkem bodů: 0.00 * (4.00 + 0.40) = 0.00