Osnovi računarskih sistema

Milena Vujošević–Janičić

Čas 1

Vežbe — 10 05 2004

1.1 Operacije sa stringovima

Klasa string sadrži kolekciju funkcija za pretrazivanje, pri čemu je svaka imenovana kao varijanta funkcije *find*.

- Funkcija `find()` je najjednostavniji primerak: za zadatu nisku ona kao rezultat daje indeks mesta na kome je prvi znak pronađene podniske, ili posebnu vrednost `string::npos` koja označava da nije pronađena podniska.
- Funkcija `find_first_of()` kao rezultat daje indeks prvog znaka niske koji odgovara bilo kom znaku u niski navedenoj kao argument. Na primer:

  ```
  string odabrana_slova("abc");
  string ime("beograd");

  unsigned pozicija = ime.find_first_of(odabrana_slova);
  // pozicija dobija vrednost 0
  ```

  Ovoj funkciji moguće je dodeliti drugi argument koji označava indeks elementa niske od koga želimo da započnemo pretraživanje. Na primer:

  ```
  string odabrana_slova("abc");
  string ime("beograd");

  unsigned index = 1;
  unsigned pozicija = ime.find_first_of(odabrana_slova, index);
  // pozicija dobija vrednost 5
  ```

- Funkcija `find_first_not_of()` pronalazi prvi znak niske koji ne odgovara ni jednom elementu niske koja se zadaje kao argument.
- Funkcija `find_last_of()` pronalazi poslednji znak niske koji odgovara nekom od elemenata niske koja se zadaje kao argument.

- Funkcija `find_last_not_of()` pronalazi poslednji znak niske koji ne odgovara ni jednom od elemenata niske koja se zadaje kao argument.

Korisna funkcija prilikom rada sa stringovima je i

```
istream& getline (istream &is, string str, char delimiter)
```

koja iz ulaznog toka upisuje znakove, uključujući i razmake, sve dok ne naiđe na graničnik, ne dođe do kraja datoteke ili dok sekvenca znakova ne dostigne vrednost `max_size()` niske, kada operacija čitanja ne uspeva.

1.2 Pretraživanje teksta

Primer 1 *Napisati klasu* `Tekst` *koja omogućava učitavanje niza rečenica iz datoteke kao i njihovo ispisivanje. Ime datoteke zadaje se kao argument komandne linije, a ukoliko se ime datoteke ne zada izbaciti izuzetak.*

```
#include <iostream>
#include <fstream>
#include <vector>
#include <set>
#include <string>

using namespace std;

class Tekst
{
public:

    // Metod ucitava recenice iz datoteke.
    void CitajRecenice( istream& f )
        {
        string red;

        while( getline( f, red ) ){
            unsigned kraj = 0;
            while( kraj < red.size() - 1 ){

                // pronadjemo poziciju prvog karaktera reda koji je razlicit
                // od svakog od znakova niske " \t\r\n"
                unsigned pocetak = red.find_first_not_of( " \t\r\n", kraj );

                // Ako takav karakter nije pronadjen, prekidamo ucitavanje
                if( pocetak == string::npos )
                    break;
```

```
                // Pronalazimo poziciju kraja recenice
                kraj = red.find_first_of( ".?!", pocetak );

                // Ukoliko znak za kraj recenice nije nadjen
                // postavlja se kraj na poslednji znak razlicit od
                // " \t\r\n"
                if( kraj == string::npos )
                    kraj = red.find_last_not_of( " \t\r\n" );

                kraj++;
                // Izdvajamo recenicu i pamtimo je
                _Recenice.push_back( red.substr( pocetak, kraj-pocetak ) );
                }
            }
        }


    // Funkcija stampa redni broj recenice, recenicu, duzinu recenice kao i aski
    // kod prvog karaktera u recenici
    void IspisiRecenice( ostream& f ) const
        {
        for( unsigned i=0; i<_Recenice.size(); i++ )
            f << i << " - " << _Recenice[i]
              << " (" << _Recenice[i].length() << ")"
              << (int)_Recenice[i][0] << endl;
        }

private:
    vector<string>  _Recenice;
};

void pretrazivanjeTeksta( char* imedat )
{
    ifstream f(imedat);
    if( !f )
        throw string("Nije uspelo otvaranje datoteke!");

    Tekst t;
    t.CitajRecenice( f );
    t.IspisiRecenice( cout );
    f.close();
}
```

```
main( int argc, char** argv )
{
    try {
        if( argc < 2 )
            throw string( "Nedostaje naziv datoteke!" );
        pretrazivanjeTeksta( argv[1] );
        }
    catch( const string& s ){
        cerr << "GRESKA: " << s << endl;
        }
    return 0;
}
```

Primer 2 *Napisati program koji omogućava formiranje kataloga reči koje se pojavljuju u nekoj datoteci. Za svaku reč treba pamtiti redene brojeve rečenica u kojima se data reč pojavljuje.*

```
#include <iostream>
#include <fstream>
#include <vector>
#include <set>
#include <map>
#include <string>

using namespace std;

class Tekst
{
    // Definise se novi tip koji je skup neoznacenih brojeva.
    typedef set<unsigned> tPojavljivanja;

    // Definise se novi tip koji je katalog reci, pritom se za
    // svaku rec pamte redni brojevi recenica u kojima se rec pojavljuje.
    typedef map<string, tPojavljivanja> tKatalogReci;

public:

    // Konstruktor klase tekst na osnovu imena datoteke otvara datoteku,
    // ucitava recenice iz nje i formira katalog reci
    Tekst( char* imedat )
        {
        ifstream f(imedat);
        if( !f )
            throw string("Nije uspelo otvaranje datoteke!");

        CitajRecenice( f );
```

```
        PripremiKatalogReci();
        f.close();
        }

    // Ispisuje se svaka rec iz kataloga zajedno sa
    // rednim brojevima recenica u kojima se rec pojavljivala
    void IspisiKatalogReci( ostream& f ) const
        {
        tKatalogReci::const_iterator
            ri = _Reci.begin(),
            re = _Reci.end();
        for( ; ri!=re; ri++ ){
            f << ri->first << " - ";
            tPojavljivanja::const_iterator
                si = ri->second.begin(),
                se = ri->second.end();
            for( ; si!=se; si++ )
                f << *si << ' ';
            f << endl;
            }
        }

    void IspisiRecenice( ostream& f ) const
        {
        for( unsigned i=0; i<_Recenice.size(); i++ )
            f << i << " - " << _Recenice[i]
              << " (" << _Recenice[i].length() << ")"
              << (int)_Recenice[i][0] << endl;
        }

private:
    void CitajRecenice( istream& f )
        {
        string red;

        while( getline( f, red ) ){
            unsigned kraj = 0;
            while( kraj < red.size() - 1 ){
                // pronadjemo pocetak
                unsigned pocetak = red.find_first_not_of( " \t\r\n", kraj );
                if( pocetak == string::npos )
                    break;

                // pronadjemo kraj
```

```
            kraj = red.find_first_of( ".?!", pocetak );
            if( kraj == string::npos )
                kraj = red.find_last_not_of( " \t\r\n" );
            kraj++;

            // izdvojimo recenicu i zapamtimo je
            _Recenice.push_back( red.substr( pocetak, kraj-pocetak ) );
            }
        }
    }

void PripremiKatalogReci()
    {
    string slova = "abcdefghijklmnopqrstuvwxyz";
    for( unsigned i=0; i<_Recenice.size(); i++ )
        {
        string recenica = _Recenice[i];

        // Pretvaramo sva slova recenice u mala slova
        for( unsigned j=0; j<recenica.size(); j++ )
            if( recenica[j]>='A' && recenica[j]<='Z' )
                recenica[j] += 'a' - 'A';

        // Izdvajamo jednu po jednu rec i smestamo je u katalog
        unsigned kraj = 0;
        while( kraj < recenica.size() - 1 )
            {
            // pronadjemo pocetak reci
            unsigned pocetak = recenica.find_first_of( slova, kraj );
            if( pocetak == string::npos )
                break;

            // pronadjemo kraj reci
            kraj = recenica.find_first_not_of( slova, pocetak );
            if( kraj == string::npos )
                kraj = recenica.size();

            // izdvojimo rec i zapamtimo je, u katalog ubacijemo za ovu rec
            // redni broj tekuce recenice
            _Reci[recenica.substr( pocetak, kraj-pocetak )].insert(i);
            }
        }
    }
```

```
    vector<string>  _Recenice;
    tKatalogReci    _Reci;
};

void pretrazivanjeTeksta( char* imedat )
{
    Tekst t(imedat);
    t.IspisiRecenice( cout );
    t.IspisiKatalogReci( cout );
}

main( int argc, char** argv )
{
    try {
        if( argc < 2 )
            throw string( "Nedostaje naziv datoteke!" );
        pretrazivanjeTeksta( argv[1] );
        }
    catch( const string& s ){
        cerr << "GRESKA: " << s << endl;
        }
    return 0;
}
```

Primer 3 *Napisati program koji omogućava formiranje kataloga reči koje se pojavljuju u nekoj datoteci i štampanje odgovarajućeg izveštaja. Izveštaj se formira na osnovu uslova koji se zadaje kao niz reči ispred kojih opciono može da stoji znak + ili znak -. Reči ispred kojih je u uslovu znak + su* obavezne*, reči ispred kojih je znak - su* zabranjene *a reči bez znaka ispred su* tražene *reči. Izveštaj treba da sadrži sve rečenice iz datoteke u kojima se pojavljuju sve obavezne reči, ni jedna zabranjena i, opciono, neka tražena reč. Ukoliko nema obaveznih reči tada se u izveštaju nalaze samo rečenice koje sadrže bar jednu traženu reč a koje nemaju zabranjenih reči.*

```
#include <iostream>
#include <fstream>
#include <vector>
#include <set>
#include <map>
#include <string>

using namespace std;

class Tekst
{
public:
```

```
typedef set<unsigned> tPojavljivanja;
typedef map<string,tPojavljivanja> tKatalogReci;

Tekst( char* imedat )
    {
    ifstream f(imedat);
    if( !f )
        throw string("Nije uspelo otvaranje datoteke!");

    CitajRecenice( f );
    PripremiKatalogReci();
    f.close();
    }

// Metod pronalazi redne brojeve onih recenica koje zadovoljavaju uslov
void Pronadji( string uslov, tPojavljivanja& rezultat ) const
    {
    set<string> obavezne, trazene, zabranjene;

    // Na osnovu uslova pune se skupovi obaveznih, trazenih i
    // zabranjenih reci
    AnalizaUpita( uslov, obavezne, trazene, zabranjene );

    // Ako postji bar jedna obavezna rec
    if( obavezne.size() > 0 ){
        set<string>::iterator
            i = obavezne.begin(),
            e = obavezne.end();

        // U katalogu reci pronalazimo prvu obaveznu rec
        tKatalogReci::const_iterator f = _Reci.find(*i);

        // Ukoliko smo rec pronasli, u rezultat smestamo podatke
        // iz kataloga vezane za datu rec (redne brojeve recenica
        // u kojima se ta rec pojavljuje)
        if( f != _Reci.end() )
            rezultat = f->second;

        // Za svaku obaveznu rec odredjujemo da li se nalazi u katalogu reci.
        for( i++; i!=e; i++ ){
            tKatalogReci::const_iterator f = _Reci.find(*i);

            // Ako se rec nalazi u katalogu reci tada
            // pravimo presek prethodnog rezultata sa pojavljivanjima
```

```
            // ove obavezne reci.
            if( f != _Reci.end() ){
                tPojavljivanja presek;
                tPojavljivanja::const_iterator
                    pi = f->second.begin(),
                    pe = f->second.end();
                for( ; pi!=pe; pi++ )
                    // Funkcija count vraca 1 ukoliko je element pronadjen
                    // u rezultatu odnosno 0 ukoliko nije.
                    if( rezultat.count(*pi)>0 )
                        presek.insert(*pi);
                rezultat = presek;
                }
            }
        }

    // Ako ne postoje obavezne reci onda rezultat
    // formiramo kao uniju recenica u kojima se nalaze
    // trazene reci
    else{
        set<string>::iterator
            i = trazene.begin(),
            e = trazene.end();
        for( ; i!=e; i++ ){
            tKatalogReci::const_iterator f = _Reci.find(*i);
            if( f != _Reci.end() ){
                tPojavljivanja::const_iterator
                    pi = f->second.begin(),
                    pe = f->second.end();
                for( ; pi!=pe; pi++ )
                    rezultat.insert(*pi);
                }
            }
        }

    // Na kraju iz rezultata izbacujemo one recenice
    // u kojima se nalaze zabranjene reci
    set<string>::iterator
        i = zabranjene.begin(),
        e = zabranjene.end();
    for( ; i!=e; i++ ){
        tKatalogReci::const_iterator f = _Reci.find(*i);
        if( f != _Reci.end() ){
            tPojavljivanja razlika;
```

```
            tPojavljivanja::iterator
                pi = rezultat.begin(),
                pe = rezultat.end();
            for( ; pi!=pe; pi++ )
                if( !f->second.count(*pi)>0 )
                    razlika.insert(*pi);
            rezultat = razlika;
            }
        }
    }

void IspisiRezultat( ostream& f, tPojavljivanja s ) const
    {
    tPojavljivanja::const_iterator
        si = s.begin(),
        se = s.end();
    for( ; si!=se; si++ )
        f << *si << " : " << _Recenice[*si] << endl;
    }

void IspisiKatalogReci( ostream& f ) const
    {
    tKatalogReci::const_iterator
        ri = _Reci.begin(),
        re = _Reci.end();
    for( ; ri!=re; ri++ ){
        f << ri->first << " - ";
        tPojavljivanja::const_iterator
            si = ri->second.begin(),
            se = ri->second.end();
        for( ; si!=se; si++ )
            f << *si << ' ';
        f << endl;
        }
    }

void IspisiRecenice( ostream& f ) const
    {
    for( unsigned i=0; i<_Recenice.size(); i++ )
        f << i << " - " << _Recenice[i]
          << " (" << _Recenice[i].length() << ")"
          << (int)_Recenice[i][0] << endl;
    }
```

```
private:
    void CitajRecenice( istream& f )
        {
        string red;

        while( getline( f, red ) ){
            unsigned kraj = 0;
            while( kraj < red.size() - 1 ){
                // pronadjemo pocetak
                unsigned pocetak = red.find_first_not_of( " \t\r\n", kraj );
                if( pocetak == string::npos )
                    break;
                // pronadjemo kraj
                kraj = red.find_first_of( ".?!", pocetak );
                if( kraj == string::npos )
                    kraj = red.find_last_not_of( " \t\r\n" );
                kraj++;
                // izdvojimo recenicu i zapamtimo je
                _Recenice.push_back( red.substr( pocetak, kraj-pocetak ) );
                }
            }
        }

    void PripremiKatalogReci()
        {
        string slova = "abcdefghijklmnopqrstuvwxyz";
        for( unsigned i=0; i<_Recenice.size(); i++ ){
            string recenica = _Recenice[i];
            for( unsigned j=0; j<recenica.size(); j++ )
                if( recenica[j]>='A' && recenica[j]<='Z' )
                    recenica[j] += 'a' - 'A';
            unsigned kraj = 0;
            while( kraj < recenica.size() - 1 ){
                // pronadjemo pocetak
                unsigned pocetak = recenica.find_first_of( slova, kraj );
                if( pocetak == string::npos )
                    break;
                // pronadjemo kraj
                kraj = recenica.find_first_not_of( slova, pocetak );
                if( kraj == string::npos )
                    kraj = recenica.size();
                // izdvojimo rec i zapamtimo je
                _Reci[recenica.substr( pocetak, kraj-pocetak )].insert(i);
                }
```

```
            }
        }

    // Na osnovu uslova prave se skupovi obaveznih, trazenih i zabranjenih reci.
    void AnalizaUpita( string uslov, set<string>& obavezne,
                       set<string>& trazene, set<string>& zabranjene ) const
        {
        string slova = "abcdefghijklmnopqrstuvwxyz";

        // Pretvaraju se sva slova uslova u mala slova
        for( unsigned i=0; i<uslov.size(); i++ )
            if( uslov[i]>='A' && uslov[i]<='Z' )
                uslov[i] += 'a' - 'A';

        unsigned kraj = 0;

        // Izdvaja se jedna po jedna rec iz uslova i ona se smesta
        // u odgovarajuci skup
        while( kraj < uslov.size() - 1 ){
            // pronadjemo pocetak
            unsigned pocetak = uslov.find_first_of( slova, kraj );
            if( pocetak == string::npos )
                break;
            // pronadjemo kraj
            kraj = uslov.find_first_not_of( slova, pocetak );
            if( kraj == string::npos )
                kraj = uslov.size();
            // izdvojimo rec i zapamtimo je
            string rec = uslov.substr( pocetak, kraj-pocetak );

            // Smestamo rec u odgovarajuci skup
            if( pocetak>0 && uslov[pocetak-1] == '+' )
                obavezne.insert( rec );
            else if( pocetak>0 && uslov[pocetak-1] == '-' )
                zabranjene.insert( rec );
            else
                trazene.insert( rec );
            }
        }

    vector<string>  _Recenice;
    tKatalogReci    _Reci;
};
```

```
main( int argc, char** argv )
{
    try {
        if( argc < 2 )
            throw string( "Nedostaje naziv datoteke!" );

        Tekst t( argv[1] );
        //  t.IspisiRecenice( cout );
        //  t.IspisiKatalogReci( cout );
        string uslov;

        // Za svaki uneti uslov stampa se odgovarajuci
        // izvestaj
        while( getline( cin, uslov ) ){
            Tekst::tPojavljivanja rezultat;
            t.Pronadji( uslov, rezultat );
            t.IspisiRezultat( cout, rezultat );
            }
        }
    catch( const string& s ){
        cerr << "GRESKA: " << s << endl;
        }
    return 0;
}
```

Sadržaj