Rad sa stringovima i klasa string

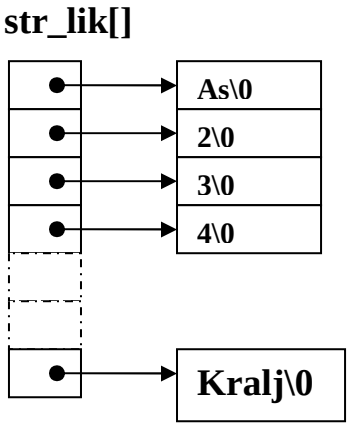
Dva su pristupa:

- **rad s ASCIIZ stringovima pomoću funkcija C-biblioteke**
- **korištenje klase string iz C++ biblioteke**

Ovaj drugi način je znatno jednostavniji.

Nekoliko primjera iz C jezika

Program: Miješanje karata



Slika 7.1 Niz pokazivača

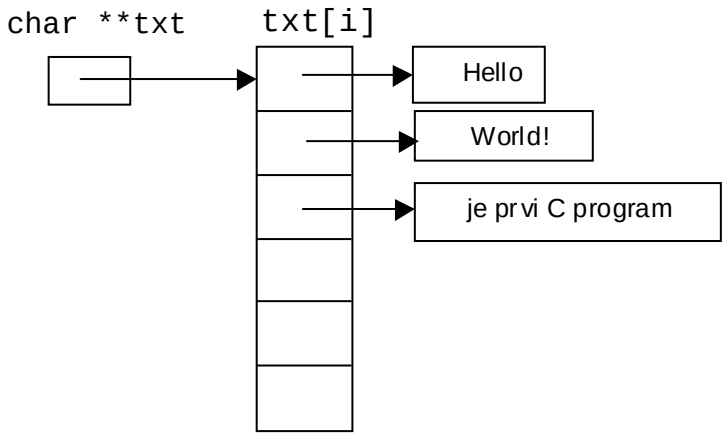
```
char *str_boja[4] = {"Srce", "Tref", "Karo", "Pik "};

char *str_lik[13]  = {"As", "2", "3", "4", "5", "6",  "7", "8", "9", "10",
                      "Jack", "Dama", "Kralj" };
```

```
// Datoteka: karte1.cpp – dijeljenje karata

char *str_boja[4] = {"Srce", "Tref", "Karo", "Pik "};
char *str_lik[13]  = {"As", "2", "3", "4", "5",
"6",  "7", "8", "9", "10", "Jack", "Dama", "Kralj" };

#define BROJ_KARATA 52

int main( void)
{
int i, karte[BROJ_KARATA];

   // poredaj po redu
   for (i = 0; i < BROJ_KARATA; i++)
        karte[i] = i;

   // izmijesaj zamjenom vrijednosti
   for (i = 0; i < BROJ_KARATA; i++)
   {
      int k = rand() % BROJ_KARATA;
      int tmp = karte[i];  // swap
      karte[i] = karte[k];
      karte[k] = tmp;
   }

   // ispisi
   for (i = 0; i < 52; i++)
   {
   cout << str_boja[karte[i]/13] << "-"
        << str_lik[karte[i]%13] << endl;
   }
   return 0;
}
```

```
c:> cards

Srce - As
Srce - Dama
Pik  - 5
Karo - Jack
Pik  - Kralj
Pik  - 2
Tref - 3
Karo - As
Pik  - 6
Tref - 2
Tref - 8
Pik  - As
Karo - Kralj
Tref - 10
Srce - 7
Tref – Kralj
.........
Tref – 4
Karo – 4
Srce – 5
Tref – 9
Pik  - 7
Tref – 5
```

Kontenjer za niz stringova promjenljive duljine

```
const int sizeincr=5;
typedef char * cstring;

class CStrArray
{
private :
    cstring *str_arr ;
    int numstrings;
    int maxsize;

public:
    CStrArray () : numstrings(0),
                   maxsize(sizeincr)
    {str_arr  = new cstring[sizeincr]; }

    ~CStrArray ();

// dopišite kopirni konstruktor i operator =

   int Numstrings () const
                   {return numstrings;}
   void AddString(cstring s) ;
    //...
cstring  operator [] (int index)
              { return str_arr [index];}
const cstring  operator [] (int index) const
              { return str_arr [index];}
};
```



Klasa CStrArray enkapsulira rad s nizom ASCIIZ stringova.

```
void CStrArray::AddString(cstring s)
{
    if(numstrings >= maxsize)
   {  // ako je numstrings veći od broja alociranih pok.
       //alociraj novi niz  uvećan za sizeincr
       cstring * p= new cstring[maxsize+sizeincr];

      //kopiraj stari u novi
      memcpy(p,str_arr , maxsize*sizeof(cstring ));

      delete [] str_arr ;             // dealociraj stari niz pok.
      str_arr = p;                    // novi pridijeli starome
      maxsize += sizeincr;
    }
     str_arr[numstrings]=s;
     numstrings++;
}


CStrArray::~CStrArray ()
{
     for(int i = 0; i < numstrings; i++ )
         delete [] str_arr[i];
     delete [] str_arr;
}


// Zadatak:
// dopišite kopirni konstruktor i operator =
```

```
int main( void )
{
     CStrArray txt;
     char str[256]={"\0"}; // radni string

   // Kraj unosa je ako se unese prazna linija.

     while(true)
     {
         cin.getline(str,255);        // dobavi string u str
         int len=strlen(str);         // odredi duljinu str
         if(len==0)                   // ako je prazan string prekini
             break;                   // inače, ako je unos ispravan
         char *s = new char[len+1];   // formiraj novi string
         strcpy(s, str);              // kopiraj radni string u njega
         txt.AddString(s);            // i pridjeli ga nizu stringova
      }
      cout << "Unos zavrsen ------------------------------" << endl;

   // Ispiši podatke

     for(int j = 0; j < txt.Numstrings(); j++ )
              cout << txt[j] <<endl ;

     return 0 ;
}
```

11.2 Argumenti komandle linije operativnog sustava

Funkcija "main" također može koristiti argumente. Ovoj funkciji argumente proslijeđuje operativni sustav s komandne linije. Primjerice, pri pozivu kompajlera

```
c:>cl /GX cards.cpp
```

iz komandne linije se kao argumenti u funkciju main() prenose dva stringa: `/GX` i `cards.cpp`.

Opći oblik deklaracije funkcije main, koja prima argumente s komandne linije, glasi:

```
int main( int argc, char *argv[])
```

Parametri su:

`argc` sadrži broj argumenata. Prvi argument (indeksa nula) je prema standardu uvijek ime samog programa, dakle broj argumenata je uvijek veći ili jednak 1.

`argv` je niz pokazivača na char, svaki pokazuje na početni znak stringova koji su u komandnoj liniji odvojeni razmakom.

Primjer 1: **program koji ispisuje sadržaj komandne linije operativnog sustava**

```
// Datoteka: cmdline.cpp


#include <iostream>
using namespace std;

int main( int argc, char *argv[])
{
int i;
    cout << "Ime programa je:"
         << argv[0] << endl;

    if (argc > 1)
    for (i = 1; i < argc; i++)
        cout << i << ". argument: "
             << argv[i] << endl;
    return 0;
}
```

```
c:> cmdline Hello World

Ime programa: C:\CMDLINE.EXE
1. argument: Hello
2. argument: World



Dakle :
argc = 3
argv[0] = "C:\CMDLINE.EXE"
argv[1] = "Hello"
argv[2] = "World"
```

11.3 String klasa

Rad s ASCIIZ stringovima ima mnoge nedostatke:

- **ASCIIZ stringovi nemaju karakter tipa podatka,**
- **sve radnje s njima mora obavljati sam programer koristeći funkcije biblioteke,**
- **te radnje se često zasnivaju na korištenju pokazivača i dinamičkog alociranja memorije, što je uzrok mnogim greškama.**

U C++ jeziku standardizirana je klasa `string` pomoću koje se može obaviti gotovo sve što je potrebno u radu s nizovima znakova. Detaljna specifikacija ove klase s primjerima primjene opisana je u Dodatku D.

Pokazat ćemo kako je realizirana klasa TString koja sadrži podskup operacija standardne klase string. Klasa TString je neznatno modificirana `apstring` klasa, koja je preporučena za nastavu od američkog College Boarda - Advance Placement Course of Computer Science.

Primjer upotrebe TString (string) klase	**Ispis:**

`#include <iostream>`
`#include "TString.h"`
`#include "TString.cpp"`
`using namespace std;`
 	
`int main()`
`{`
`  TString s = "Hello";`	

```
// dodaj znak ','
s += ',';

// ispiši string
cout << s << endl;                          Hello,

//dodaj literalni string;
s = s + " World";

// dodaj uskličnik
s +='!';
cout << s << endl;                          Hello, World!

//kopirni konstruktor –stvara s2
TString s2(s);

// pronadji substring
// na indeksu 7 duljine 5
TString s3 = s2.substr(7,5);                World
cout << s3 << endl;

// trazenje znakova i stringova
int idx = s.find('!');
cout << "indeks od !: " << idx << endl;     indeks od !: 12

// ako je indeks > -1 (npos)
if(idx != TString::npos)
       s[idx] = '?';
idx = s.find("World");
cout << "indeks od World: "
     <<  s.find("World")  << endl;          indeks od World: 7
```

```
    //zamijeni veliko u malo slovo
    if(idx != TString::npos)
         s[idx] = tolower(s[idx]);
    cout << s << endl;


    // mozemo koristiti i c-funkcije
    cout << "duljina stringa je:"
         << strlen(s.c_str()) <<endl;


    // ili koristimo svojstvo automatske
    // pretvorbe tipa string u char *

     cout << "duljina stringa je:"
          << strlen(s) <<endl;

    return 0;
}
```

```
Hello, world?


duljina stringa je:13



duljina stringa je:13
```

Specifikacija ADT- string:

- **String objekt može sadržavati sve znakove osim '\0'.**
- **Ako se u string objekt nenamjerno upiše '\0', onda se svi znakovi iza nul-znaka više ne smatraju dijelom stringa.**
- **Uz svaku funkciju specificiramo PRE i POST uvjete.**
- **Neispunjenje PRE uvjeta rezultira porukom o greški i prekidom programa.**
- **POST uvjeti određuju rezultat neke operacije**

```
class TString                // u std biblioteci ime: string
{

int m_length;                // duljina stringa (broj znakova)
int m_capacity;              // kapacitet stringa - u bajtima
char * m_cstr;               // pokativač na ASCIIZ string

public:

static const int npos;   // not a position (-1)

// konstruktori/destruktor

TString( );                 // konstruktor praznog stringa ""
TString( const char * s ); // konstruktor iz literalnog str.
TString( const TString & str );     // kopirni konstruktor
~TString( );                        // destruktor

// pridjela vrijednosti

const TString & operator = ( const TString & str );
const TString & operator = ( const char * s );           const TString &
operator = ( char ch );

// pristupnici

int length( )const {return m_length;}   // broj znakova
int size( )const {return m_length;}     // broj znakova
```

```
// traženje u stringu

//vraća indeks pojave stringa str u stringu this
int    find( const TString & str ) const;

//indeks pojave znaka ch
int    find( char ch ) const;

//substring duljine len od pozicije  pos
TString substr( int pos, int len ) const;

// pretvorba u  char *
const char* c_str( ) const {return m_cstr;}

// pretvorba u  char *
operator char* ( ) const {return m_cstr;}

// indeksirani pristup

char   operator[ ]( int k ) const;
char & operator[ ]( int k );

// mutatori - dopuna stringa  stringom str i znakom ch

const TString & operator += ( const TString & str );
const TString & operator += ( char ch );

};
```

```
// Nečlanske funkcije
//
// ulazni/izlazne operacije

ostream & operator << ( ostream & os, const TString & str );
istream & operator >> ( istream & is, TString & str );
istream & getline( istream & is, TString & str );

// relacijski operatori:

bool operator == (const TString & lhs, const TString & rhs);
bool operator != (const TString & lhs, const TString & rhs);
bool operator <  (const TString & lhs, const TString & rhs);
bool operator <= (const TString & lhs, const TString & rhs);
bool operator >  (const TString & lhs, const TString & rhs);
bool operator >= (const TString & lhs, const TString & rhs);

// operator + (dopuna stringa)

TString operator+ (const TString& lhs, const TString& rhs );
TString operator+ (char ch, const TString & str);
TString operator+ (const TString & str, char ch );
```

IMPLEMENTACIJA je dana u skripti

Prikažimo samo neke funkcije:

```
#include <string.h>
#include <assert.h>
#include <stdlib.h>
#include <ctype.h>
#include "TString.h"

const int TString::npos = -1;

TString::TString()
// POST: string je prazan
{
    m_length = 0;
    m_capacity = 1;
    m_cstr = new char[m_capacity];
    m_cstr[0] = '\0';           // cstring nulte duljine
}

TString::TString(const char * s)
//    Opis: konstruktor iz literalnog stringa, pr. "abcd"
//    PRE:  s je '\0'-terminirani ASCIIZ string
//    POST: formira se kopija od s
{
    assert (s != 0);          //  provjera: cstring not NULL?
    m_length = strlen(s);
    m_capacity = m_length + 1;      // dodaj bajt za '\0'
    m_cstr = new char[m_capacity];
    strcpy(m_cstr,s);
}

TString::~TString()
//    POST: dealocira se string
{     delete[] m_cstr;               // dealociraj memoriju
}
```

```
const TString& TString::operator =(const TString & rhs)
//    POST: pridjela kopiranjem string objekta
{
    if (this != &rhs)                    // provjeri da li je
    {                                    // s=s;
        if (m_capacity < rhs.length()+1) // procijeni da li treba
        {                                // povećati memoriju
            delete[] m_cstr;          // dealociraj stari string
            m_capacity = rhs.length() + 1;// dodaj 1 bajt za '\0'
            m_cstr = new char[m_capacity];
        }
        m_length = rhs.length();
        strcpy(m_cstr,rhs.m_cstr);
    }
    return *this;
}


char& TString::operator[](int k)
//    PRE:  0 <= k < length()
//    POST: vraća referencu k-tog znaka
//    Nota: ako se ova referenca koristi za upis '\0'
//          daljni resultati su nedefinirani
{
    if (k < 0 || m_length <= k)    {
        cerr << "Indeks izvan dozvoljenog intervala: "
             << k << " string: " << m_cstr  << endl;
        exit(1);
    }
    return m_cstr[k];
}
```

```
ostream& operator <<(ostream & os, const TString & str)
//    POST: str se šelje izlaznom toku os
{    return os << str.c_str();   }


istream& operator >>(istream & is, TString & str)
//    PRE:  ulazni tok je otvoren
//    POST: dobavlja se string iz ulaznog toka i sprema u str
{
    char ch;
    str = "";    // prazan string, dodavat ćemo znak po znak
    is >> ch;    // dobavi prvi ne-bijeli znak ch

    if (! is.fail()) {
        do{
            str += ch;
            is.get(ch);
        } while (! is.fail() && ! isspace(ch));

        if (isspace(ch))    // vrati razmak u stream
            is.putback(ch);
    }
    return is;
}


istream & getline(istream & is, TString & str)
//    Opis: dobavlja liniju teksta iz ulaznog toka u string str
//    PRE:  ulazni tok je otvoren
//    POST: više znakova zaključno s '\n' se očitava s ulaza
//          i sprema u str ('\n' se ne sprema u str)
{
    char ch;
```

```
    str = "";
    while (is.get(ch) && ch != '\n')  {
        str += ch;
    }
    return is;
}

const TString& TString::operator +=(const TString & str)
//    POST: dodaje kopiju str stringa na kraj this stringa
{
    TString copystring(str);         // napravi kopiju stringa

    int newLength = length() + str.length(); // procijeni duljinu
    int lastLocation = length();      // položaj  '\0' znaka

    // provjeri da li treba povećati memoriju
    if (newLength >= m_capacity)
    {
        m_capacity = newLength + 1;
        if (str.length() == 1)          {
            m_capacity *= 2;
        }
        char * newBuffer = new char[m_capacity];
        strcpy(newBuffer,m_cstr); // kopiraj u novi buffer
        delete [] m_cstr;         // dealociraj stari string
        m_cstr = newBuffer;
    }

    // sada dodaj str (copystring) na kraj od m_cstr
    strcpy(m_cstr+lastLocation, copystring.c_str() );
    m_length = newLength;    // upiši novu duljinu stringa

    return *this;
}
```

```
const TString & TString::operator += ( char ch )
//    POST: dodaje znak ch na kraj this stringa
{   TString temp;   // napravi string koj sadrži znak ch
    temp = ch;
    *this += temp;
    return *this;
}
TString operator +(const TString & lhs, const TString & rhs)
//    POST: vraća string koji se sastoji od lhs dopunjenog s rhs
{
    TString result(lhs);  // kopira lhs u result
    result += rhs;         // dopunjuje rhs
    return result;         // vraća result
}
TString operator + ( char ch, const TString & str )
//    POST: vraća string koji se sastoji znaka ch i stringa str
{   TString result; // napravi string koj sadrži znak ch
    result = ch;
    result += str;
    return result;
}

TString operator + ( const TString & str, char ch )
// POST: vraća string koji se sastoji str dopunjenog znakom ch
{
    TString result(str);
    result += ch;
    return result;
}
```

```
TString TString::substr(int pos, int len) const
//    Opis:   vraća substring duljine len počevši od indeksa pos
//    PRE:  this string sadrži c0, c1, ..., c(n-1), uz uvjet
//                0 <= pos <= pos + len - 1 < n.
//    POST: vraća string koji sadrži
//                c(pos), c(pos+1), ..., c(pos+len-1)
{
    if (pos < 0)            // start at front when pos < 0
        pos = 0;

    if (pos >= m_length) return "";// prazni string

    int lastIndex = pos + len - 1; // zadnji index koji kopiramo
    if (lastIndex >= m_length)     // budi unutar stringa?
    {
        lastIndex = m_length-1;
    }

    TString result(*this);   // alociraj dovoljno memorije
    int j,k;
    for(j=0,k=pos; k <= lastIndex; j++,k++)
    {
        result.m_cstr[j] = m_cstr[k];
    }
    result.m_cstr[j] = '\0';       // terminiraj cstring
    result.m_length = j;           // i zabilježi duljinu
    return result;
}
```

```
int TString::find(const TString & str)  const
//    Opis:   pronalazi pojavu stringa str u this stringu
//            i vraća indeks prvog znaka. Ako this string ne
//            sadrži string str, tada vraća npos.
//    PRE:  this string sadrži c0, c1, ..., c(n-1)
//          str sadrži s0, s1, ...,s(m-1)
//    POST: ako je s0 == ck0, s1 == ck1, ..., s(m-1) == ck(m-1)
//          i ako  ne postoji j < k0 takovi da je
//          s0 = cj, ...., sm == c(j+m-1),
//          tada vraća k0;
//          u suprotnonm slučaju vraća npos
{
    int len = str.length();
    int lastIndex = length() - len;
    for(int k=0; k <= lastIndex; k++)
        if (strncmp(m_cstr + k,str.c_str(),len) == 0) return k;
    return npos;
}


int TString::find( char ch ) const
//    Opis: pronalazi prvu pojavu znaka u this stringu
//          i vraća indeks
//          Ako znak ch nije u this stringu, tada vraća npos.
//    PRE:  this string sadrži c0, c1, ..., c(n-1)
//    POST: ako je ch == ck,
//          i ako ne postoji j < k takovi da je ch == cj
//          tada vraća k;
//          u suprotnonm slučaju vraća npos
{
    for(int k=0; k < m_length; k++)
        if (m_cstr[k] == ch)  return k;
    return npos;
}
```

```
bool operator == ( const TString & lhs, const TString & rhs )
{
    return strcmp(lhs.c_str(), rhs.c_str()) == 0;
}

bool operator != ( const TString & lhs, const TString & rhs )
{
    return ! (lhs == rhs);
}

bool operator <  ( const TString & lhs, const TString & rhs )
{
    return strcmp(lhs.c_str(), rhs.c_str()) < 0;
}

bool operator <= ( const TString & lhs, const TString & rhs )
{
    return !( rhs < lhs );
}

bool operator >  ( const TString & lhs, const TString & rhs )
{
    return rhs < lhs;
}

bool operator >= ( const TString & lhs, const TString & rhs )
{
    return ! ( lhs < rhs );
}
```

```
//Primjer programa karte2.cpp pomoću klase TString

#include <iostream>
#include "tstring.h"
#include "TString.cpp"
#include <cstdlib> // deklaracija rand()


TString boja[] = {"Srce", "Tref", "Karo", "Pik "};

TString lik[]  = {"As", "2", "3", "4", "5",
                  "6",  "7", "8", "9", "10",
                  "Jack", "Dama", "Kralj" };

// Sve ostalo prepiši iz programa karte.cpp u kojem smo
// koristili ASCIIZ stringove
```

Klasom string možemo u mnogo slučajeva zamijeniti ASCIIZ objekte.

Što time dobijamo?

- **Dobijamo na sigurnosti, jer se smanjuje mogućnost nedozvoljenog pristupa memoriji.**
- **Često dobijemo kraće i elegantnije napisane programe, iako s nešto duljim vremenom izvršenja.**

To ćemo demostrirati tako da pomoću klase string ponovo napišemo klasu za rad s nizom stringova CStrArray.

```
// program strarray2.cpp
#include <iostream>
#include <cassert>
#include <string>  // standardna klasa string
using namespace std;

const int sizeincr=5;
```

```
class CStrArray
{
private :
    string *str_arr ;     // niz stringova
    int numstrings;
    int maxsize;
public:
    CStrArray () : numstrings(0), maxsize(sizeincr)
                   {str_arr  = new string[sizeincr]; }
    ~CStrArray ();
    int Numstrings () const {return numstrings;}
    void AddString(string s) ;
    //...
    string&  operator [] (int index) { return str_arr [index];}
    const string&  operator [] (int index) const { return str_arr [index];}
};

void CStrArray::AddString(string s)
{
     if(numstrings >= maxsize)
      {    // ako je numstrings veći od broja alociranih pok.
           //allociraj novi niz pok.
         string * p= new string[maxsize+sizeincr];
         for(int i=0; i<maxsize; i++)
             p[i] = str_arr[i];
         delete [] str_arr ;  // dealociraj stari niz pok.
         str_arr = p;         // novi pridijeli starome
         maxsize += sizeincr;
    }
     str_arr[numstrings]=s;
     numstrings++;
}
```

```
CStrArray::~CStrArray ()
{
     delete [] str_arr;
}

int main( void )
{
     CStrArray txt;
     string str; // radni string

   // Kraj unosa je ako se unese prazna linija ili EOF
     while(getline(cin, str))
     {
        if(str.length()==0)    // ako je prazan string
           break;              // prekini unos teksta
                               // ako je unos ispravan
        txt.AddString(str);    // i pridjeli ga nizu
    }
     cout << "Unos zavrsen ----------------------" << endl;

// Ispiši podatke iz dinamički alociranog niza
     for(int j = 0; j < txt.Numstrings(); j++ )  {
         cout << txt[j] <<endl ;
     }
     return 0 ;
}
```

Razlike od verzije s ASIIZ stringom je u destruktoru i funkciji AddString.
U destruktoru se samo jednom koristi delete [] operator jer se njime automatski poziva i destruktor svih stringova koji su u nizu.
U funkciji AddString se ne koristi funkcije memcpy već se kopiranje niza stringova vrši petljom
To je nužno je se samo na taj način može formirati novi niz stringova. Jasno, ova operacija je znatno sporija od prostog kopiranja niza pokazivača. Da smo koristili prosto kopiranje pokazivača, tada bi nakon naredbe delete [] str_arr bili izbrisani i svi alocirani stringovi iz p niza.

11.4 Stringovni tokovi

U standardnoj `iostream` biblioteci implementirane su klase

`istringstream, ostringstream i stringstream.`

One omogućuje da se i objekti standardne klase `string` tretiraju izvor ili odredište ulazno/izlaznih tokova. Njih se može koristiti tako da se deklarira zaglavlje `<sstream>`.

11.4.1. Izlazni stringovni tok

```
ostringsteream ostr;
```

takovom objektu možemo pristupiti kao i bilo kojem drugom ostream toku:

```
int n = 2345;
ostr << "Broj iznosi: ";
ostr << hex <<n <<" Hex ili ";
ostr << dec << n << " Dec.";
cout << ostr.str() << endl;
```

bit će ispisano:

```
Broj iznosi: 929 Hex ili 2345 Dec.
```

Objekti `ostringstream` klase koriste (naslijeđuju) sve članske funkcije i manipulatore `ostream` klase, jedino oni još rapolažu s konstruktorom i funkcijom `str()`. Dva su tipa konstruktora;

```
ostringstream::ostringstream();
ostringstream::ostringstream (string const &s, ios::openmode mode);
string ostringstream::str() const
```

Prvi oblik konstruktora se koristi bez argumenata. Tada se podrazumijeva da je otvoren prazni izlazni tok. Drugi tip konstruktora ima dva argumenta. prvi je referenca stringa kojim se početno puni tok, a drugi argument mode koji uvijek iznosi ios:out | mode. Djelovanje ovog drugog tipa konstruktora nije ujednačeno implementirano kod većine današnjih kompajlera, pa se ne preporučuje njegova upotreba.

11.4.1. Ulazni stringovni tok

Ulazni stringovni tok se formira kao objekt klase `istringstream` koja naslijeđuje `istream`.
Članske funkcije su:

```
istringstream::istringstream();
istringstream::istringstream (string const &text);
void istringstream::str(string const &text);
```

Izlazni stringovni tok se uvijek prije upotrebe mora ispuniti nekim tekstualnim sadržajem.
Primjerice,

```
istringstream istr("55 678 921");
```

Kada je tok ispunjen, iz njega se može vršiti ekstrakcija numeričkih i stringovnih tipova pomoću operatora i funkcija ekstrakcije. Primjerice, iz ovog toka može se dobaviti vrijednost tri cjelobrojne varijable sa

```
int x, y, z;
istr >> x >> y >> z;
```

Uočite da će se petljom;

```
int x;
while ( istr >> x)
    cout << x << endl;
```

ispisati:

```
55
678
921
```

Može se koristiti klasa `stringstream` za formiranje i ulaznih i izlaznih tokova. Primjerice, sa

```
stringstream S(ios::in | ios::out);
S.str("123567");
```

formira se tok S koji je početno ispunjen sa "12345", a može se koristiti kao ulazni i izlazni tok, dok se deklaracijom;

```
stringstream S(ios::in);
```

formira tok S koji se može koristiti samo kao ulazni tok.