TEKSTOVNE DATOTEKE

1

DELO S (TEKSTOVNIMI) DATOTEKAMI
BRANJE IZ DATOTEKE
PISANJE V DATOTEKO

Datotečni podatkovni toki

2

- viri, ki predstavljajo fizično datoteko na disku
- tekstovna datoteka je zbirka elementov tipa char
  - RAZLIKA GLEDE NA POLJA
    - tekstovna datoteka je shranjena v trajnem pomnilniku, polja v delovnem pomnilniku
    - tekstovne datoteke omogočajo zaporedni dostop, polja direktni dostop
- razredi za datotečne podatkovne toke so definirani v
  - knjižnico `fstream`
  - imenskem prostoru `std`

Delo z datotekami

3

- pri delu z datotekami poznamo tri osnovne korake
  1. ODPIRANJE DATOTEKE
     - pripravimo objekt, ki predstavlja datoteko, za delo
  2. DELO Z DATOTEKO
     - če je objekt pripravljen, lahko z njim delamo
     - izvedi operacijo, ki jo želiš
     - preveri, ali je bila operacija uspešno izvedena
  3. ZAPIRANJE DATOTEKE
     - ko je delo z datoteko zaključeno, jo zapremo

Razredi za delo z datotekami

4

- razredi za datotečne podatkovne toke so definirani v
  - knjižnico `fstream`
  - imenskem prostoru `std`
- razredi
  - ifstream
    - vhodna datoteka
  - ofstream
    - izhodna datoteka
  - fstream
    - vhodno/izhodna datoteka

Delo z datotekami (1)

5

- z vsako datoteko so povezane tri zastavice
  - eofbit
    - ta zastavica se postavi, kadar smo z zadnjo operacijo dosegli konec datoteke
  - failbit
    - se postavi, kadar zadnja operacija ni bila uspešna
  - badbit
    - se postavi, kadar naletimo na napako, ki ni vključena v nobeni od prejšnjih
- zastavica – spremenljivka tipa bool

Delo z datotekami (2)

6

- zastavice lahko preverjamo z naslednjimi funkcijami
  - bool eof();
    - vrne true, če je postavljena zastavica eofbit, in false, sicer
  - bool fail();
    - vrne true, če je postavljena zastavica failbit, in false, sicer
  - bool bad();
    - vrne true, če je postavljena katera izmed zastavic badbit ali failbit, in false, sicer
  - bool good();
    - vrne true, če ni postavljena nobena od treh zastavic, in false, sicer
- zapiranje datoteke izvedemo s funkcijo close()

Branje iz tekstovne datoteke (1)

7

- za branje podatkov potrebujemo vhodni datotečni tok
- definiran je z razredom ifstream
- konstruktorji
  - privzeti konstruktor
    - ifstream dat;
    - v tem primeru se samo pripravi objekt za delo z datoteko, ki ni povezan z nobeno fizično datoteko na disku. Zato ga je potrebno z želeno datoteko povezati s funkcijo open():
      - dat.open("ime_datoteke") → ime_datoteke – c-niz
    - za podrobnosti funkcije open glej naslednji konstruktor

Branje iz tekstovne datoteke (2)

8

- konstruktorji
  - `ifstream(const char * filename, ios_base::openmode mode = ios_base::in)`
    - parametri so identični parametrom funkcije open()
    - prvi parameter je ime datoteke, s katero želimo povezati objekt
      - mora biti podan kot C-niz
    - drugi parameter (ima privzeto vrednost) pove način odpiranja datoteke

Načini odpiranja datotek

9

- funkcija open()
  - `open(const char * filename, ios_base::openmode mode = ios_base::in | ios_base::out)`
    - prvi parameter je ime datoteke, s katero želimo povezati objekt
      - mora biti podan kot C-niz
    - drugi parameter (ima privzeto vrednost) pove način odpiranja datoteke
- načini odpiranja datotek

vrednost	pomen
ios_base::app	APPEND – dodaj novi podatki se dodajajo na konec datoteke
ios_base::ate	AT THE END – ob odpiranju datoteke se postavi na konec
ios_base::in	vhodna datoteka*1
ios_base::out	izhodna datoteka*2
Ios_base::trunc	TRUNCATE – odreži stara vsebina se izbriše iz datoteke

* pri razredih `ifstream`[1] in `ofstream`[2] sta to privzeti vrednosti, pri `fstream` pa sta privzeta oba načina

Branje iz tekstovne datoteke (3)

10

- iz tekstovne datoteke lahko beremo naslednje tipe podatkov
  - int
  - double
  - char
  - char*
  - string
    - razred ima definiran operator >>
  - razrede, ki imajo definiran operator >> za vhodne podatkovne toke

Branje iz tekstovne datoteke (4)

11

- branje osnovnih podatkovnih tipov
  - uporabimo operator >>
  - PRIMER: dat >> x;
- operator >> nad nizi (in C-nizi)
  - bere samo do prvega nevidnega znaka
- branje objektov tipa string
  - če želimo brati tudi nevidne znake, uporabimo že znano funkcijo getline()

Primer 5 – branje iz tekstovne datoteke

12

```
/* napiši program, ki prebere vsa števila v
   tekstovni datoteki in na koncu izpiše vsoto in
   povprečno vrednost prebranih števil. */

#include <fstream>
#include <string>
#include <iostream>
using namespace std;

void obdelajDatoteko (string aIme){
   int vs=0;
   int n=0;
   // odpri datoteko
   ifstream dat (aIme.c_str());
   if (dat.fail()){
     cout << "Napaka z datoteko.";
     return;
   }

   while (!dat.eof()) {
     int x;
     dat >> x;
     if (dat.fail())
       break;
     vs+=x;
     n++;
   }
   if (n==0)
     cout << "Nismo prebrali nobenega stevila.";
   else{
     cout << "Vsota je " << vs << endl;
     cout << "Povprecna vrednost je" <<
             double (vs)/n << endl;
   }
   dat.close();
}

int main () {
   string ime;
   cout << "Vnesi ime datoteke: ";
   getline (cin, ime);
   obdelajDatoteko (ime);
   return 0;
}
```

Pisanje v tekstovno datoteko (1)

13

- za pisanje podatkov potrebujemo izhodni datotečni tok
- definiran je z razredom ofstream
- konstruktorji
  - privzeti konstruktor
    - ofstream dat;
    - v tem primeru se samo pripravi objekt za delo z datoteko, ki ni povezan z nobeno fizično datoteko na disku. Zato ga je potrebno z želeno datoteko povezati s funkcijo open():
      - dat.open("ime_datoteke") → ime_datoteke – c-niz
    - za podrobnosti funkcije open glej naslednji konstruktor

Pisanje v tekstovno datoteko (2)

14

- konstruktorji
  - `ofstream(const char * filename, ios_base::openmode mode = ios_base::out)`
    - parametri so identični parametrom funkcije open()
    - prvi parameter je ime datoteke, s katero želimo povezati objekt
      - mora biti podan kot C-niz
    - drugi parameter (ima privzeto vrednost) pove način odpiranja datoteke
    - za podrobnosti funkcije open glej opis pri branju iz datoteke

Pisanje v tekstovno datoteko (3)

15

- v tekstovno datoteko lahko pišemo naslednje tipe podatkov
  - int
  - double
  - char
  - char*
  - string
    - razred ima definiran operator <<
  - razrede, ki imajo definiran operator << za izhodne podatkovne toke

Pisanje v tekstovno datoteko (4)

16

- za pisanje prej naštetih podatkovnih tipov v datoteko
  - uporabimo operator <<
  - PRIMER: dat << x;
- operator << zapiše vse znake v nizu (C-nizu) v datoteko
  - spomnimo: operator >> prebere nize (C-nize) le do prvega nevidnega znaka

Primer 6 – pisanje v tekstovno datoteko

17

```
/* napiši program, ki prebere deset
   celih števil preko tipkovnice in
   jih shrani v tekstovno datoteko.
   Vsako število naj bo v svoji
   vrstici. */

#include <fstream>
#include <string>
#include <iostream>
using namespace std;

void preberiVDatoteko (string aIme){
   // odpri datoteko
   ofstream dat (aIme.c_str());

   if (dat.fail()){
     cout << "Napaka z datoteko.";
     return;
   }
   for(int i=0; i<10; i++) {
      int x;
      cout << "Vnesi celo stevilo: ";
      cin >> x;
      // shrani št. v datoteko
      dat << x << endl;
   }
   dat.close();
}

int main () {
   string ime;
   cout << "Vnesi ime datoteke: ";
   getline (cin, ime);
   obdelajDatoteko (ime);
   return 0;
}
```

Primer 7 – še ena naloga z datotekami (1)

18

```
/* Napišite podprogram, ki prebere tekstovno datoteko z besedilom. V
   novo datoteko vpiše vrstice, ki ne vsebujejo besede stop. Na koncu
   izpiše število vrstic, ki vsebujejo besedo stop. */

int vpis (string aIme){
   ifstream dat (aIme.c_str());  // iz te dat. beremo
   ofstream dat2("Test.txt");  // v to datoteko pišemo

   int stevec = -1; // šteje število vrstic,
                    // ki vsebujejo besedo stop

   // sta obe datoteki pripravljeni za delo?
   if (!dat.fail() && !dat2.fail()) {
        stevec = 0;
        string niz;
```

Primer 7 – še ena naloga z datotekami (2)

19

```
        while (!dat.eof()){
                getline (dat, niz);        // preberi niz
                if (dat.fail())            // če branje ni bilo uspešno
                        break;

                 // ali niz vsebuje besedo "stop"?
                int indeks = niz.find ("stop");
                if (indeks == string::npos)  // če ne najdemo besede "stop"
                        dat2<< niz << endl; // shrani v datoteko
                else
                        stevec ++;
        }//konec while

        dat.close ();
        dat2.close();
        return stevec;
   } // konec if
   else
        cout << "Napaka pri odpiranju ene od datotek.";
   return stevec;
}
```

Primer 7 – še ena naloga z datotekami (3)

20

```
int main () {
  string imeDat;
  cout << "Vnesi ime datoteke: ";
  getline (cin, imeDat);

  int vrstice = vpis (imeDat);

  if (vrstice == -1)
        cout << "Nobena vrstica ni vsebovala besede stop." << endl;
  else {
        cout << "Število vrstic, ki vsebujejo besedo stop: ";
        cout << vrstice << endl;
  }
  return 0;
}
```