11. Nizi znakov

1

11. 1. C-nizi

2

- podatkovni tip `char` predstavlja natanko en znak
- za predstavitev več znakov potrebujemo dodatni tip

- prvi način predstavitve nizov je s C-nizi
  - izhaja iz programskega jezika C (predhodnik C++)
  - nize si predstavimo kot polje znakov (elementi so tipa `char`)
  - vsak C-niz se zaključi z dodatnim t.i. null znakom `'\0'`

`PRIMER:`
`"niz"` je predstavljen kot `'n' 'i' 'z' '\0'`

Še o C-nizih

3

PRIMER:

```
// niz z največ 29 znaki, saj je zadnji '\0'
char niz[30];  // statično polje
char  *niz2 = new char[30]; // dinamično polje
```

- omenjamo jih iz zgodovinskih razlogov
- za delo z njimi je na voljo knjižnica `cstring`

Delo s C-nizi ter `cout` in `cin`

4

- z objektom `cout` lahko izpisujemo C-nize z operatorjem <<
  - kot smo to počeli do sedaj za osnovne podatkovne tipe
- z objektom `cin` lahko preberemo C-niz preko tipkovnice z operatorjem >>
  - POZOR: prebere do prvega nevidnega znaka
- za branje tudi nevidnih znakov uporabimo funkcijo `cin.getline(niz, kolicina, locilni_znak);`

11.2. Razred string

- za lažje delo z nizi uporabimo knjižnico string
- vsebuje razred string, ki predstavlja niz znakov
  - definiran je v imenskem prostoru std
- za uporabo razreda *string* moramo v program vključiti naslednje vrstice:

```
#include <string>
using namespace std;
```

- prednosti glede na C-nize:
  - za delo s pomnilnikom je samodejno poskrbljeno
  - ne rabimo v naprej poznati količine znakov v nizu
  - preprosto pretvorimo C-niz v string in obratno

string – ustvarjanje nizov

- od sedaj dalje bomo objektu tipa `string` rekli niz
- objekt tipa `string` ustvarimo med drugim tudi z enim od naslednjih konstruktorjev:
  - osnovni konstruktor – ustvari prazen niz

```
string n1;
```

  - ustvari kopijo obstoječega niza

```
string n2(n1);
```

  - ustvari kopijo obstoječega C-niza

```
string n3("Primer");
```

  - ustvari niz z *n* ponovitvami znaka *c*

```
string n4(5, '*');
```

Izpis na ekran

- za razred `string` je definiran operator <<, ki ga lahko uporabljamo s `cout`
- niz izpišemo na ekran

```
string niz("Kmalu bodo prazniki.");
cout << niz << endl;
```

- izpis na ekran

```
Kmalu bodo prazniki.
```

Branje nizov

- nize lahko beremo z vhodnih podatkovnih tokov na dva načina
- branje z operatorjem >>
  - prebere, kot pri C-nizih, le do prvega nevidnega znaka

PRIMER:

```
cin >> n1;
```

- za branje tudi nevidnih znakov je v knjižnici `string` definirana globalna funkcija `getline()`

Branje nizov (preko tipkovnice)

9

- operator >> in objekt *cin*
- niz lahko preberemo, kot smo to do sedaj počeli s števili

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
cin >> ime;
cout << "Pozdravljen(a), " << ime << "." << endl;
```

Branje nizov (preko tipkovnice)

10

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
cin >> ime;
cout << "Pozdravljen(a), " << ime << "." << endl;
```

IZPIS NA EKRAN:

```
Vnesi svoje ime: Andrej Taranenko
Pozdravljen(a), Andrej.
```

Branje nizov (preko tipkovnice)

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
cin >> ime;
cout << "Pozdravljen(a), " << ime << "." << endl;
```

IZPIS NA EKRAN:

```
Vnesi svoje ime: Andrej Taranenko
Pozdravljen(a), Andrej.
```

Zakaj ne izpiše naslednje?
Pozdravljen(a), Andrej Taranenko.

Branje nizov (preko tipkovnice)

- z operator >> preberemo zaporedje znakov do prvega nevidnega znaka (presledek, nova vrstica, ...)
- rečemo lahko, da na ta način beremo samo besede

```
string niz;
cin >> niz;
```

prebere do prvega nevidnega znaka

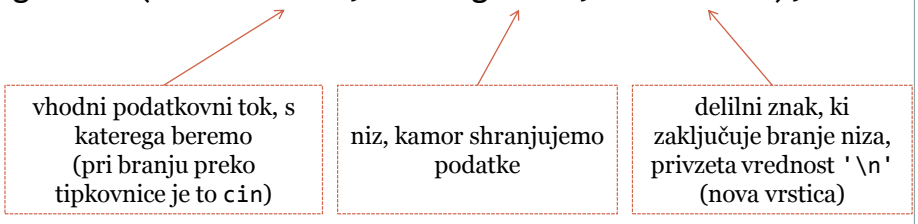
Funkcija getline()

13

- definirana je v knjižnici string
- uporabljamo jo za branje nizov (vključno z nevidnimi znaki) z vhodnega podatkovnega toka

PROTOTIP FUNKCIJE:

```
getline(istream &is, string &str, char znak);
```



Branje nizov (preko tipkovnice)

14

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
getline(cin, ime);
cout << "Pozdravljen(a), " << ime << "." << endl;
```

IZPIS NA EKRAN:

```
Vnesi svoje ime: Andrej Taranenko
Pozdravljen(a), Andrej Taranenko.
```

Branje nizov (preko tipkovnice)

15

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
getline(cin, ime);
cout << "Pozdravljen(a), " << ime << "." << endl;
```

prebere vse znake do konca vrstice, vključno s presledki

IZPIS NA EKRAN:

```
Vnesi svoje ime: Andrej Taranenko
Pozdravljen(a), Andrej Taranenko.
```

Branje nizov (preko tipkovnice)

16

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
getline(cin, ime, 'a');
cout << "Pozdravljen(a), " << ime << "." << endl;
```

IZPIS NA EKRAN:

```
Vnesi svoje ime: Andrej Taranenko
Pozdravljen(a), Andrej T.
```

Branje nizov (preko tipkovnice)

17

PRIMER:

```
string ime;
cout << "Vnesi svoje ime: ";
getline(cin, ime, 'a');
cout << "Pozdravljen(a), " << ime << "." << endl;
```

prebere vse znake do prve pojavitve znala 'a' ali konca vrstice, vključno s presledki

IZPIS NA EKRAN:

```
Vnesi svoje ime: Andrej Taranenko
Pozdravljen(a), Andrej T.
```

Razredne metode vs. klasične funkcije

18

- kako smo funkcije poznali do sedaj:
  - funkcijo smo izvedli nad nekimi objekti / podatki
    imeFunkcije(objekti);
- razredne metode uporabljamo drugače, kot "klasične" funkcije
  - izvedemo jih preko objekta: imeObjekta.imeMetode(...);
  - torej najprej povemo, nad katerim objektom izvajamo metodo, šele nato, katero metodo izvajamo

Metode za delo z nizi

19

- v razredu string do med drugimi definirane tudi naslednje metode za delo z nizi:
  - size() ... vrne velikost niza, število znakov v nizu
  - clear() ... izbriše vse znake v nizu, dobimo prazni niz
  - empty() ... vrne true, če je niz prazen, in false, sicer
  - at(i) ... dostop do znaka na indeksu i v nizu
  - [i] ... enako kot at(i)
  - insert(i, niz) ... na i-to mesto vrine niz
  - erase(i, n) ... od i-tega mesta naprej izbriše n znakov
  - c_str() ... vrne niz kot C-niz

Metode za delo z nizi

20

- v razredu string do med drugimi definirane tudi naslednje metode za delo z nizi:
  - substr(i, n) ... vrne podniz, ki se prične na indeksu i, vsega skupaj pa je n znakov
  - find(podniz) ... vrne indeks prve pojavitve podniza v nizu, nad katerim izvajamo funkcijo.
    - vrne vrednost string::npos, če iskanega podniza ne najde
  - operatorji <, <=, >, >=, ==, !=
    - leksikografska primerjava nizov
  - operator + ... lepljenje nizov
  - operator = ... prirejanje vrednosti

Prirejanje nizov

21

- prireditveni stavek deluje tudi nad nizi
- z operatorjem = shranimo vrednost v spremenljivko

```
string stavek1, stavek2;
...
stavek1 = "To je stavek";
stavek2 = stavek1; // kopira vrednost iz
                   // stavek1 v stavek2
```

Lepljenje nizov – konkatenacija

22

- nize lahko tudi "lepimo", staknemo skupaj
- operacija se imenuje konkatenacija nizov
- uporabimo operator +

PRIMER:

```
string niz = "To je stavek";
niz = niz + ".";
cout << niz << endl;
```

Izpis na ekran:

```
To je stavek.
```

Dolžina niza

- dolžina niza je definirana, kot število znakov v nizu
- metoda `size()` vrne dolžino niza
  - metoda ... pomeni uporabljamo preko objekta
  - vrne število znakov v nizu – celo število

PRIMER:

```
string niz = "To je stavek.";
int dolzina = niz.size();
cout << "Stevilo znakov v nizu je " << dolzina;
```

IZPIS NA EKRAN:

```
Stevilo znakov v nizu je 13
```

Dostop do posameznih znakov

- niz si lahko predstavljamo kot zaporedje znakov
- prvi znak je na mestu številka 0
  - pravimo, da ima indeks 0

PRIMER:

0	1	2	3	4	5	6	7	8	9	10	11	12
T	o		j	e		s	t	a	v	e	k	.

Dostop do posameznih znakov

25

PRIMER:

0	1	2	3	4	5	6	7	8	9	10	11	12
T	o		j	e		s	t	a	v	e	k	.

- do posameznega znaka dostopamo z operatorjem []
- v splošnem

```
imeNiza[indeksZnaka]
```

- indeks mora biti znotraj dovoljenih meja
- na to moramo paziti sami!!!

Dostop do posameznih znakov

26

PRIMER:

0	1	2	3	4	5	6	7	8	9	10	11	12
T	o		j	e		s	t	a	v	e	k	.

```
string niz = "Ta je stavek";
niz[1] = 'o';     // popravim 'a' v 'o'

// izpišimo prvi in zadnji znak v nizu
cout << "Prvi znak v nizu je " << niz[0] << endl;
int i = niz.size() - 1; // indeks zadnjega znaka
cout << "Zadnji znak v nizu je " << niz[i] << endl;
```

IZPIS NA EKRAN:

```
Prvi znak v nizu je T
Zadnji znak v nizu je .
```

Dostop do posameznih znakov

PRIMER (popravek za splošni primer):

```
// izpišimo prvi in zadnji znak v nizu
// storimo lahko, če znaki obstajajo

if (niz.size()>0) {
  cout << "Prvi znak v nizu je " << niz[0] << endl;
  int i = niz.size() - 1; // indeks zadnjega znaka
  cout << "Zadnji znak v nizu je " << niz[i] << endl;
}
```

Iskanje podniza

- včasih želimo vedeti, ali se neko besedilo pojavi znotraj drugega besedila
- metoda `find(iskaniNiz)`
  - vrne mesto iskanega niza, indeks, kjer se iskani niz začne
  - vrne mesto, kjer se iskani niz **prvič** pojavi, če se pojavi večkrat
  - vrne konstanto **string::npos**. če iskanega niza ne najde
- uporaba metode

```
int i = besedilo.find(iskaniNiz);
```

v nizu *besedilo* poišči *iskaniNiz* in mesto, kjer se prvič pojavi, shrani v spremenljivko *i*

Iskanje podniza - primer

29

```
string niz = "To je stavek.";
int i;

i = niz.find("je");
cout << "je se v nizu pojavi na indeksu " << i << endl;

i = niz.find("e");
cout << "e se v nizu pojavi na indeksu " << i << endl;
```

ne pozabi: vrne mesto, kjer se prvič pojavi

Izpis na ekran:

```
je se v nizu pojavi na indeksu 3
e se v nizu pojavi na indeksu 4
```

Iskanje podniza - primer

30

```
string niz = "To je stavek.";
int i;

i = niz.find("TO");
if (i != string::npos)
  cout << "je se v nizu pojavi na indeksu " << i << endl;
else
  cout << "Iskani niz se v nizu ne pojavi." << endl;
```

iščemo niz TO, ne niza To

Izpis na ekran:

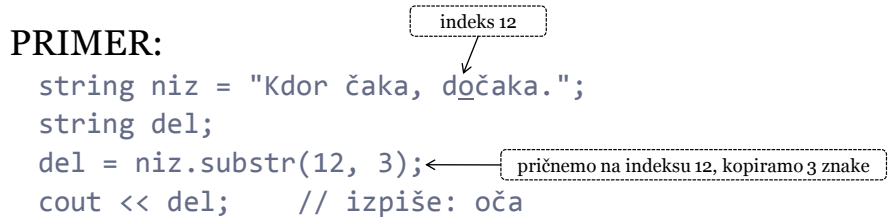
```
Iskani niz se v nizu ne pojavi.
```

Vračanje podniza – kopiranje (dela) niza

31

- pogosta operacija nad nizi je kopiranje niza ali vsaj dela niza
- metoda `substr(pol, dolzina)`
  - metoda vrne podniz obstoječega niza – vrne novi niz
  - pol … na katerem mestu v nizu pričnemo kopirati
  - dolzina … koliko znakov želimo skopirati

PRIMER:



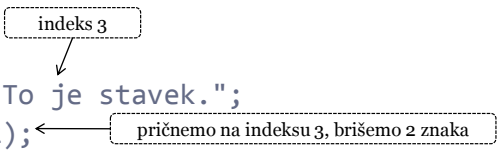

```
string niz = "Kdor čaka, dočaka.";
string del;
del = niz.substr(12, 3);
cout << del;    // izpiše: oča
```

Brisanje podniza

32

- prav tako pogosta operacija nad nizi je brisanje niza ali vsaj dela niza
- metoda `erase(pol, dolzina)`
  - metoda spremeni obstoječi niz
  - pol … na katerem mestu v nizu pričnemo brisati
  - dolzina … koliko znakov želimo brisati

PRIMER:



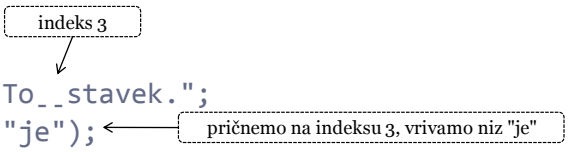

```
string niz = "To je stavek.";
niz.erase(3, 2);
cout << niz;    // izpiše: To__stavek.
                // ostaneta dva presledka
```

Vrivanje podniza

33

- prav tako pogosta operacija nad nizi je vrivanje niza v obstoječi niz
- metoda `insert(pol, podniz)`
  - metoda spremeni obstoječi niz
  - pol ... na katero mesto v nizu vrinemo novi niz
  - podniz ... kateri niz vrivamo v obstoječi niz

PRIMER:




```
string niz = "To__stavek.";
niz.insert(3, "je");
cout << niz;     // izpiše: To_je_stavek.
                 // znaki se zamaknejo
```

Operatorji primerjanja

34

- za nize so definirani tudi operatorji primerjanja

  `==` `!=` `<` `<=` `>` `>=`

- dva niza sta enaka natanko tedaj, ko sta enake dolžine in so vsi istoležni znaki enaki
- en niz je manjši od drugega natanko tedaj, ko je v leksikografski ureditvi pred drugim nizom
- leksikografska ureditev glede na ASCII tabelo

Operatorji primerjanja - primeri

35

```
string niz1 = "avto";
string niz2 = "bus";
string niz3 = "Bus";

if (niz2 == niz3)
  cout << "Niz " << niz2 << " je enak nizu " << niz3 << endl;
else
  cout << "Niza sta različna." << endl;

if (niz1 < niz2)
  cout << "Niz " << niz1 << " je leksikografsko pred " << niz2;
else
  cout << "Niz " << niz2 << " je leksikografsko pred " << niz1;
```

Operatorji primerjanja - primeri

36

```
string niz1 = "avto";
string niz2 = "bus";
string niz3 = "Bus";

if (niz2 == niz3)        // false
  cout << "Niz " << niz2 << " je enak nizu " << niz3 << endl;
else
  cout << "Niza sta različna." << endl;

if (niz1 < niz2)         // true
  cout << "Niz " << niz1 << " je leksikografsko pred " << niz2;
else
  cout << "Niz " << niz2 << " je leksikografsko pred " << niz1;
```

Primeri

1. Napišite program, ki prešteje, kolikokrat se v prebranem nizu pojavi presledek.
2. Napišite program, ki iz prebranega niza izbriše vse samoglasnike.
3. Napišite program, ki vse presledke v prebranem nizu spremeni v zvezdice.