NIZI ZNAKOV

1

DELO Z BESEDILOM
RAZRED string
C-nizi

Razred *string*

2

- besedilo oz. tekst imenujemo *znakovni nizi*
- razred **string** omogoča, da nize znakov obravnavamo kot osnovni podatkovni tip
- definiran je v knjižnici string in v imenskem prostoru std
- za uporabo razreda *string* moramo v program vključiti naslednje vrstice:

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

Deklaracija spremenljivke tipa string

3

- spremenljivko deklariramo kot smo vajeni

```
string niz1;            // prazen niz
```

- imamo pa še naslednje možnosti

```
string niz2(niz1);      // kopija niza niz1
string niz3(5, '*');    // "*****"
```

Izpis na ekran

4

- ta 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.
```

Prirejanje nizov

5

- 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

6

- 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.
```

Branje nizov (preko tipkovnice)

7

- 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)

8

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)

9

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)

10

- 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

Branje nizov (preko tipkovnice)

- kaj pa, če želimo prebrati stavke, torej besedilo, ki vsebuje presledke (ali druge nevidne znake)
- funkcija `getline()`
  - definirana v knjižnici `string`
  - je globalna funkcija, ne razredna
- dve različici uporabe funkcije getline()
  - `getline(vhodniPodatkovniTok, niz);`
  - `getline(vhodniPodatkovniTok, niz, ločilniZnak);`

za nas bo to tipkovnica, torej objekt **cin**

Branje nizov (preko tipkovnice)

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)

13

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)

14

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)

15

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.
```

Dolžina niza

16

- dolžina niza je definirana, kot število znakov v nizu
- razredna funkcija `size()` vrne dolžino niza
  - razredna funkcija … 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

17

- 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

18

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

19

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

20

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
- razredna funkcija `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 funkcije

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

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

Iskanje podniza - primer


```
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


```
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

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

PRIMER:

indeks 12

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

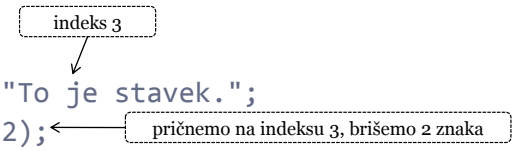
pričnemo na indeksu 12, kopiramo 3 znake

Brisanje podniza

25

- prav tako pogosta operacija nad nizi je brisanje niza ali vsaj dela niza
- razredna funkcija `erase(pol, dolzina)`
  - funkcija 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

26

- prav tako pogosta operacija nad nizi je vrivanje niza v obstoječi niz
- razredna funkcija `insert(pol, podniz)`
  - funkcija 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

27

- 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

28

```
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


```
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;
```

false

true

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.