

Podaci znakovnog tipa

Tip podataka char omogućava deklaraciju promjenljive koja može poprimiti vrijednost jednog karaktera ili niza karaktera. Promjenljiva deklarirana kao char može poprimiti vrijednost bilo kojeg ASCII karaktera.

```
char slovo;  
cout <<"Unesite slovo: ";  
cin >> slovo;
```

Niz karaktera možemo deklarirati na dva načina:

```
char prezime[80];  
string ime;
```

U prvoj naredbi deklarirali smo niz prezime od maksimalno 80 elemenata koji su karakteri. Prezime je niz karaktera. U drugoj naredbi deklarirali smo string promjenljivu ime u koju možemo pohraniti niz karaktera. Ime je objekat klase string.

U ovoj oblasti upoznat ćemo kako možemo koristiti promjenljive koje čuvaju karaktere, kako ih možemo učitavati i manipulirati njima, te ćemo upoznati funkcije za rad sa stringovima.

String zapravo nije tip podataka, nego klasa koja je definisana u skladu sa objektno orijentisanom metodom programiranja. Klasa string zapravo omogućava da deklariramo objekat ove klase, te da se koristimo metodama klase. Mi ćemo umjesto pojma objekat i dalje koristiti **pojam promjenljiva**, a umjesto pojma metod koristiti ćemo **pojam funkcija**. Tako kažemo da je ime promjenljiva tipa string (ispravno bi bilo: ime je objekat klase string).

Promjenljiva tipa string deklarira se kao i druge, s tim što se kao tip podataka navodi string. Npr.

```
string naslov = "Zdravo svijete!";  
string odgovorDa = "Da";  
string pitanje = " ";
```

Da bi ovakva deklaracija bila ispravna, potrebna je direktiva kojom se u program uključuje biblioteka koja će omogućiti deklaraciju i rad sa string promjenljivim:

```
#include <string>
```

Broj karaktera koje možemo pohraniti u promjenljivu tipa string možemo dobiti pomoću funkcije `type_size()`, kao u primjeru:

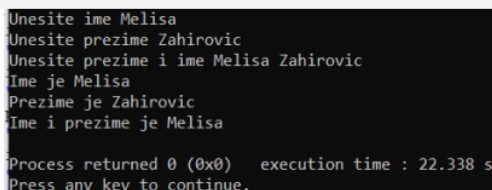
```
string ime;  
cout <<ime.max_size();
```

```
2147483647
```

Učitavanje i ispis stringa

Do sada smo podatke sa standardne ulazne jedinice, tastature, učitali pomoću **cin** naredbe. Istu naredbu smo koristili i za unos brojeva, i za unos karaktera. U slijedećem kodu od korisnika zahtijevamo da upiše tri podatka tipa string

```
int main()
{string ime, prezime, imePrezime;
  cout <<"Unesite ime ";
  cin >> ime;
  cout <<"Unesite prezime ";
  cin >> prezime;
  cout <<"Unesite prezime i ime ";
  cin >> imePrezime;
  cout <<"Ime je " << ime<<endl;
  cout <<"Prezime je " << prezime<<endl;
  cout <<"Ime i prezime je " << imePrezime<<endl;
  return 0;
}
```



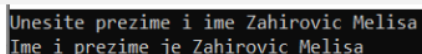
```
Unesite ime Melisa
Unesite prezime Zahirovic
Unesite prezime i ime Melisa Zahirovic
Ime je Melisa
Prezime je Zahirovic
Ime i prezime je Melisa

Process returned 0 (0x0)   execution time : 22.338 s
Press any key to continue.
```

Kada izvršimo ovaj kod, vidimo da se string unesen u promjenljivu imePrezime nije ispravno učitao. To je zato što operator >> naredbe cin prestaje da učitava podatke kada naiđe na prazan karakter, tj. space karakter (" "). Naime ovaj karakter ima specijalno značenje. On označava, između ostalog, kraj unosa. Stoga cin ignoriše sve karaktere koje upišemo poslije praznine. Isto se dešava kada se, prilikom unosa podataka, pritisnu tasteri Enter ili Tab. U sva tri ova slučaja, unos se prekida.

Klasa string obezbjeđuje naredbu za učitavanje podataka sa tastature koji prihvata sve karaktere, uključujući i karakter praznina. To je naredba **getline**. Pogledajmo modifikovani kod koji umjesto cin koristi getline i rezultat izvršavanja

```
cout <<"Unesite prezime i ime ";
getline(cin,imePrezime);
cout <<"Ime i prezime je " << imePrezime<<endl;
```



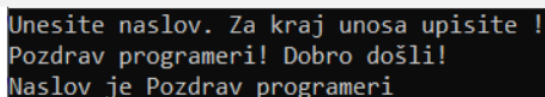
```
Unesite prezime i ime Zahirovic Melisa
Ime i prezime je Zahirovic Melisa
```

Naredba getline prihvata sve unesene karaktere, pohranjuje ih u string imePrezime, i zaustavlja unos kada korisnik pritisne taster **Enter**.

Pored ovog oblika, naredba getline može se koristiti na još nekoliko načina. Jedan od njih omogućava određivanje karaktera kojim se unos prekida. Npr.

```
int main()
{string naslov;
  cout <<"Unesite naslov. Za kraj unosa upisite ! "<<endl;
  getline(cin,naslov, '!');
  cout <<"Naslov je " << naslov<<endl;

  return 0;
}
```



```
Unesite naslov. Za kraj unosa upisite !
Pozdrav programeri!
Naslov je Pozdrav programeri
```

U ovom primjeru, naredba `getline` sadrži **terminator** `“!”`, kojim se određuje kada treba zaustaviti unos karaktera. Stoga je u testu ovog koda u stringu naslov upisan samo tekst „Pozdrav programeri“.

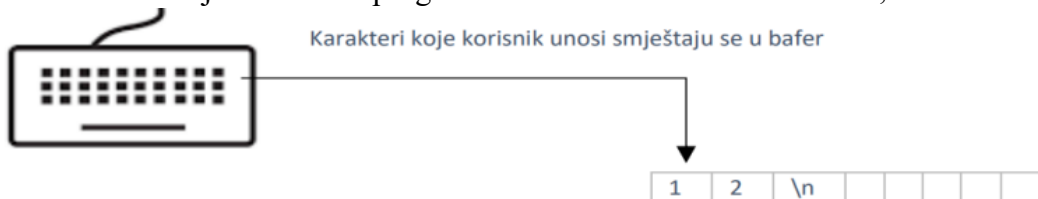
Učitavanje stringa	
string stringPromjenljiva; cin >> stringPromjenljiva; Sa standardne ulazne jedinice, tastature, učitavaju se karakteri u promjenljivu <code>stringPromjenljiva</code> . Učitavanje se prekida kada se naiđe na karakter praznina ili kada korisnik pritisne tastere <code>Enter</code> ili <code>Tab</code> .	
getline(cin, stringPromjenljiva); Sa standardne ulazne jedinice, tastature, učitavaju se karakteri u promjenljivu <code>stringPromjenljiva</code> . Unos se prekida kada korisnik pritisne taster <code>Enter</code> . getline(cin, stringPromjenljiva, terminator); Sa standardne ulazne jedinice, tastature, učitavaju se karakteri u promjenljivu <code>stringPromjenljiva</code> . Unos se prekida kada korisnik unese karakter <code>terminator</code> .	
Primjeri	
cin >> prezime;	U string <code>prezime</code> učitat će se karakteri sve dok korisnik ne unese prazninu ili pritisne tastere <code>Enter</code> ili <code>Tab</code> .
getline(cin, imePrezime);	U string <code>imePrezime</code> učitat će se karakteri sve dok korisnik ne pritisne tastere <code>Enter</code> .
getline(cin, naslov, '!');	U string <code>naslov</code> učitat će se karakteri sve dok korisnik ne unese karakter <code>!</code> .

Treba biti oprezan pri korištenju ova dva načina unosa (cin i getline) u istom programu. Posmatrajmo primjer :

```
int main()
{int broj;
 string naslov;
 cout << "Upisite cio broj ";
 cin >> broj;
 cout << "Upisali ste broj "<< broj << endl;
 cout << "Upiste naslov ";
 getline(cin, naslov);
 cout << "Upisali ste naslov " << naslov << endl;
 return 0;
}
```

```
Upisite cio broj 12
Upisali ste broj 12
Upiste naslov Upisali ste naslov
```

Kada se izvršava program, najprije se od korisnika traži da upiše cio broj. Nakon unosa broja, korisnik pritisne taster Enter. Na gotovo svim računarima, podaci se najprije upisuju u privremenu memoriju, koju nazivamo buffer, prije nego što se upišu u memoriju računara. Nakon unosa broja 12 iz testa programa u baferu se nalaze karakteri 1, 2 i \n.



Kada se naiđe na naredbu getline, iz bafera se uzima kod tastera Enter (\n), što je terminator getline naredbe. Stoga se prekida unos podataka i naredba getline ne učitava ništa.

Da bismo izbjegli ovakve probleme, moramo paziti da u istom programu ne koristimo obje naredbe: cin i getline. To uvijek nije moguće, te ćemo upoznati način kako ovaj problem možemo riješiti. Funkcija **ignore** omogućava da se pročita, a zatim ignoriše jedan ili više karaktera koji su u baferu. Ako modifikujemo prethodni primjer, tako što nakon naredbe cin, a prije naredbe getline navedemo naredbu cin.ignore(), riješit ćemo prethodno opisan problem:

```
int main(){
int broj;
 string naslov;
 cout << "Upisite cio broj ";
 cin >> broj;
 cout << "Upisali ste broj "<< broj << endl;
 cout << "Upiste naslov ";
 cin.ignore();
 getline(cin, naslov);
 cout << "Upisali ste naslov " << naslov << endl;
 return 0;
}
```

```
Upisite cio broj 12
Upisali ste broj 12
Upiste naslov Tesla
Upisali ste naslov Tesla
```

Funkcija ignore

cin.ignore();

Prihvata i ignoriše jedan karakter.

cin.ignore(1);

Isto što i cin.ignore().

cin.ignore(duzina);

Prihvata, a zatim ignoriše **duzina** karaktera.

cin.ignore(karakterTerminator);

Prihvata karaktere sve dok ne naiđe na karakter **karakterTerminator** . Odbacuje preostale (ignoriše).

Slijedi....

Funkcije za rad sa stringovima.....