Увод у програмирање – Стрингови у Ц++

Уколико желимо да радимо са низовима карактера (нискама) користимо класу **string**. Данас ћемо се упознати и са уграђеним функцијама за рад са њима.

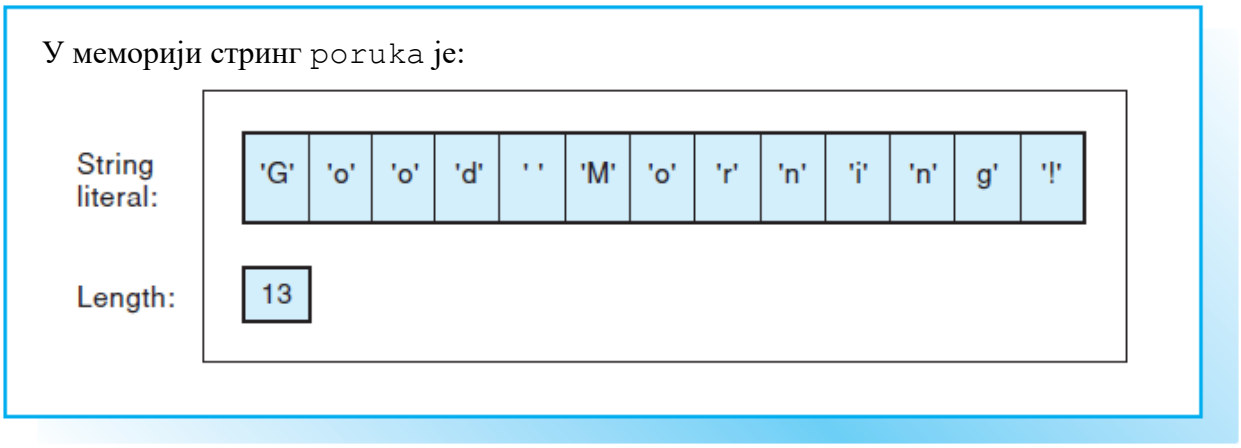
Дефиниција ниске и иницијализација

Уколико желимо да радимо са нискама морамо у коду да укључимо одговарајуће заглавље односно да наведемо претпроцесорску директиву:

```
#include <string>
```

Уколико откуцамо и извршимо следећу линију кода:

```
string poruka = "Good Morning!";
```



Ако не иницијализујемо стринг алоцираће се празан стринг дужине 0. Дужина ниске (the length) представља број карактера ниске и чува се интерно те јој се може приступити коришћењем метода length().

Задатак 1. Сунђер Боб и Патрик су одлучили да направе такмичење на дну и да победник освоји Кебину пљескавицу као прву награду. Како је Лигњослав једини био у Кеба Краби прогласили су га за судију. Договор је био, да у првој игри ко напише дужу реченицу побеђује. Како је Лигњослав увек мрзовољан и лењ Сунђер Боб и Патрик су се за помоћ обратили нама. Написати програм који ће за унету реченицу коју је написао Сунђер Боб, а потом ону коју је написао Патрик одредити победника у првој игри и на излаз написати чија је реченица дужа.

Улаз:
Ja sam sundjer.
Ja sam morska zvezda.

Излаз:
Pobednik je Patrik!
Улаз:
Ja radim u Keba Krabi.
Ja sam morska zvezda.
Излаз:
Pobednik je Sundjer Bob!
Решење:

```
#include <iostream>
#include <string>
using namespace std;
int main(){
   string bob, patrik;
   getline(cin, bob);
   getline(cin, patrik);
   if(bob.length()>patrik.length())
      cout << "Pobednik je Sundjer Bob!" << endl;
   else
      cout << "Pobednik je Patrik!" << endl;
   return 0;
}
```

Задатак 2. Друга игра такмичења на дну мора јесте да за унету реченицу обрнемо и испишемо на излаз како бисмо проверили ко је тачно обрнуо реченицу.
Улаз:
Ja sam Sundjer Bob.
Излаз:
.boB rejdnuS mas aJ
Решење:

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main(){
   string str;
   getline(cin, str);
   reverse(str.begin(), str.end());
   cout<<str<<endl;
   return 0;
}
```

Задатак 3. За задати стринг који чине велика слова и размаци проверити да ли је палиндром.

Шта је то палиндром?

Улаз:
ANA VOLI MILOVANA
Излаз:
DA

Улаз:
MILOVAN VOLI ANU
Излаз:
NE

Решење:

```
#include <iostream>
#include <string>
using namespace std;
int main(){
   string str;
   getline(cin, str);
   int i=0;
   int j=str.length()-1;
   bool ind=true;
   while(i<j){
      if(str.at(i)==' ') {i++; continue;}
      if(str.at(j)==' ') {j--; continue;}
      if(str.at(i)!=str.at(j)){ind=false; break;}
      i++; j--;
   }
   if(ind) cout<<"DA"<<endl;
   else cout<<"NE"<<endl;
   return 0;
}
```

Задатак 4. У програмима за табеларна израчунавања (Ексел) колоне су обележене словима и то као A, B, ..., Z, AA, AB, ..., AZ, BA, BB, ..., ZZ, AAA, ... Напиши програм који омогућава конверзију редног броја (од 1 па навише) у текстуалну ознаку колоне и обратно.

Улаз

На стандардном улазу налази се у једној линији низ карактера, а у другој број.

Излаз

На стандардном излазу исписати у једној линији број колоне која је репрезентована низом карактера из првог реда, а у другом реду низ карактера који би репрезентовао колону са тим бројем.

Примери:

Улаз	**Улаз**	**Улаз**
D	AB	ZZZZZ
18	100	1000000
Излаз	**Излаз**	**Излаз**
4	28	12356630
R	CV	BDWGN

Решење:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  int broj;
  string s;
  cin >> s;
  cin >> broj;
  int br = 0;

  for (int i=0;i<s.length();i++)
    br = br * 26 + (s.at(i) - 'A' + 1);
  string rez = "";
  do {
    broj--;
    rez = string(1, 'A' + broj % 26) + rez;
    broj /= 26;
  } while (broj > 0);
  cout << br << endl;
  cout << rez << endl;
}
```

Задаци за самостални рад:

Задатак 5. У стрингу који се уноси са стандардног улаза заменити свако појављивање бланко карактера са карактером * и исписати новонастали стринг, као и број извршених промена.

Задатак 6. На стандардном улазу у једној линији уноси се низ карактера који представља хексадецималан запис броја x, а у другој број чију хексадецималну репрезентацију треба исписати на излаз. Исписати редом у првој линији декадну репрезентацију броја чија је репрезентација дата низом карактера у првој линији, а у другој хексадекадну репрезентацију броја из друге линије стандардног улаза.

Примери:

Улаз	**Улаз**	**Улаз**
A5	F1	100
18	100	311
Излаз	**Излаз**	**Излаз**
165	241	256
12	64	137

Задатак 7. Два друга Матија и Василије желе да се дописију шифрујући своје поруке. Договорили су се да ће САМО да шифрују слова поруке и то свако слово поруке да запишу као следеће слово енглеске абецеде. Уколико је у питању карактер Z или z договор је да се он шифрује као A односно a.

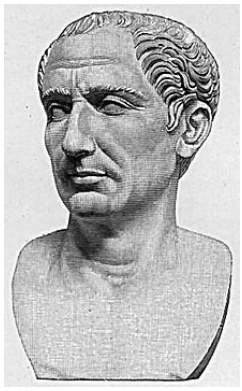
За дату реченицу која се уноси у једној линији са стандардног улаза и коју Василије жели да пошаље Матији исписати код који би требало да запише.

Улаз

Cas je zavrsen

Излаз

Dbt kf abwstfo



Овај начин шифровања познат је и као Цезаров код. У криптографији, **Цезарова шифра** је један од најпростијих и најраспрострањенијих начина шифровања. То је тип шифре замењивања у коме се свако слово отвореног текста мења одговарајућим словом азбуке, померeним за одређени број места. На пример, са помаком 3, A се замењује словом Г, Б са Д итд. Овај метод је добио име по Јулију Цезару, који га је користио за размену порука са својим генералима.

Ако је имао да каже нешто поверљиво, он је то писао шифровано тако што је мењао редослед слова у алфабету и на тај начин постигао да се ниједна реч није могла препознати. Ако би неко то желео да дешифрује и добије значење тога, морао би да замени четврто слово алфабета, дакле D *са* A *и тако даље за остала.* — Светоније, *Живот Јулија Цезара*

Sintaksa za koriščenje stringova i korisne funkcije:

```
string s, t; //deklaracija
cin >> s; //ucitavanje
t = "aaa"; //dodeljivanje vrednosti stringu t
cout << s.size() << endl; //ispisivanje duzine stringa s
string q = s + t; //konkatenacija (nadovezivanje)
q += s; //nadovezivanje s na kraj q
sort(q.begin(), q.end()); //sortiranje stringova, potrebna biblioteka
    algorithm
reverse(q.begin(), q.end()); //obrtanje stringa, potrebna biblioteka
    algorithm
cout << q[3] << endl; //ispisivanje karaktera na indeksu 3
cout << q.substr(3, 4) << endl; //ispisivanje podstringa koji pocinje od
    indeksa 3, duzine 4
```

Uvod u programiranje - stringovi

1. Šta smo naučili na prošlom času?
https://petlja.org/biblioteka/r/lekcije/prirucnik-cpp/stringovi1

```cpp
#include <iostream>
#include <string>

using namespace std;

int main()
{
string s;
    cin >> s;
    cout << s << endl;

    return 0;
}
```

Uvežbavamo stringove

https://petlja.org/biblioteka/r/lekcije/prirucnik-cpp/stringovi2

1. Napisati C++ program koji će ispisati ime i prezime Miki Maus u dva reda, dok u 3. redu će ispisati ime i prezime sa tri uzvičnika na kraju!!!
Ulaz: **nema.**
Izlaz: **tri reda na standardnom izlazu.**
Primer

Ulaz	Izlaz:
	Miki Maus
	Miki Maus
	Miki Maus!!!

```cpp
#include <iostream>
#include <string>

using namespace std;

int main(){
  string ime="Miki";
  string prezime("Maus");
  cout << ime << " " << prezime << endl;
  cout << ime +  " " + prezime << endl;
  string spojeno=ime + " " + prezime;
  spojeno=spojeno+'!';
  spojeno+="!!";
  cout << spojeno << endl;
    return 0;
}
```

U C++-u postoji tip podataka string, koji sadrži tekst. On se koristi kao i svi ostali tipovi, te se neka promenljiva tipa string deklariše na sledeći način:

```cpp
using namespace std;
string s = "Bubble bee string";
```

Ili samo

```cpp
std::string s = "Bubble bee string";
```

Možete da primetite da se navodnici koriste za dodelu teksta kao vrednosti stringu u primeru iznad. String ovde predstavlja nisku (niz) pojedinačnih znakova.

Dodela stringa

Takođe, možemo da dodelimo vrednost jednog stringa drugom:

```
string str1 = "Zdravo!";
string str2;
str2 = str1;
```

Možete i da inicijalizujete jedan string drugim kao u primeru:

```
string str1 = "Zdravo!";
string str2 = str1;
```

"Spajanje" dva stringa

Dva stringa (ili više njih) se mogu spojiti koristeći operaciju sabiranja "+" kao što smo radili sabirajući brojeve.

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string s = "Bubble";
    string p = "bee";
    s = s + p;
    cout << s << endl;
    return 0;
}
```

Nakon izvršavanja programa promenljiva `s` će sadržati string "`Bubblebee`".

Unos stringa sa tastature

Kao i ostali tipovi u C++-u, za učitivanje i ispisivanje stringa se koriste se globalni objekti `cin` i `cout`.

Primer unošenja stringa sa tastature:

```
std::string moj_string;
cin>>moj_string;
```

Operator >> će zaustaviti naš program i sačekati da korisnik nešto otkuca. Evo jednog primera unosa:

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

int main(){
  string ime;
  cout << "Unesite svoje ime: ";
  cin >> ime;
  cout << "Zdravo, " << ime << "!" << std::endl;
```

```
  return 0;
}
```

Objekat cin međutim prestaje da učitava string čim naiđe na razmak, tab (tabulator) ili nov red. Ukoliko želiš da uneseš celu liniju teksta možeš da koristiš funkciju *getline*:

```
getline(cin, moj_string);
```

Prvi argument funkcije getline je cin, koji kaže odakle nam dolazi tekst (u ovom slučaju sa konzole, to jest tastature). Drugi argument je ime string promenljive u koju želiš da smestiš uneti tekst. `getline` čita celu liniju sve dok korisnik ne pritisne Enter. Ovo je korisno kada ulazni string sadrži razmake. Sad zamisli da od korisnika tražiš da unese puno ime i prezime u jednoj liniji, pošto ćemo tako imati razmak u tekstu koristićemo `getline` .

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

int main(){
  string puno_ime;
  cout << "Unesite svoje puno ime i prezime: ";
  getline(cin, puno_ime);
    cout << "Zdravo " << puno_ime << "!" << std::endl;

  return 0;
}
```

Evo za nekoliko primera ulaznog teksta koje bi rezultate uporedo dali cin i getline:

"ana bane"
cin: "ana"
getline: "ana bane"

"ana bane
cane dejan"
cin: "ana"
getline: "ana bane"

"ana"
cin: "ana"
getline: "ana"

Dužina stringa

Slova, razmaci, znaci interpunkcije i svi ostali znaci koji se koriste u računaru nazivaju se karakteri. Da bismo dobili broj karaktera u nekom stringu, pišemo:

```
str1.length()
```

Zapis str1.length() se koristi kao (brojni) izraz, čija je vrednost upravo broj karaktera stringa str1, odnosno njegova dužina.

Evo jednog primera kratkog programa u kome ilustrujemo dodelu, spajanje i dužinu stringa:

```
#include <iostream>
#include <string>

using namespace std;

int main()

{
 string ime;
 string prezime;
 string puno_ime;
 int duzina_imena;

 cout << "Unesi svoje ime: ";
 getline(cin, ime);

 cout << endl;

 cout << "Unesi svoje prezime: ";
 getline(cin, prezime);
 puno_ime = ime + " " + prezime;
 duzina_imena = puno_ime.length() - 1; // Oduzimamo 1 zbog razmaka

 cout << endl <<endl;
 cout << "Zdravo, " << puno_ime << endl;
 cout << "Tvoje puno ime sadrzi " << duzina_imena << " karaktera." << endl;

    return 0;

}
```

Primer ispisa ovog programa:

Unesi svoje ime: Ibro
Unesi svoje prezime: Dirka
Zdravo, Ibro Dirka
Tvoje puno ime sadrzi 9 karaktera.

Poređenje stringova

Sledeći programski kod određuje da li je ulazni string jednak unapred zadatom stringu (stringu naše lozinke):

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

int main()
{
    string lozinka;
    getline(cin, lozinka);

    if(lozinka == "3astoL1$am0dabrao0vol1koDugackuLoz1nku")
    {
      cout<<"Tacna lozinka!";
    }
    else
```

```
    {
      cout<<"Netacna lozinka!";
    }

    return 0;
}
```

Dakle, operatori poređenja `>, <, ==, >=, <=, !=` se mogu primeniti i na stringove tako da možemo formirati izraze koji vraćaju vrednosttrue/false. Operatori <, >,<= i >=, upoređuju stringove leksikografski, tj. po abecedi, a ne po dužini stringa. Tako bi npr. string "akrobacije" bio "manji" od stringa "burek", iako je duži od drugog stringa, zato što se pre njega pojavljuje u rečniku.

Toliko za sad o stringovima.

ZADACI ZA VEŽBU

2. Šta je rezultat rada programa? Zašto?

```
#include <iostream>
#include <string>

using namespace std;

int main(){
  string niz1, niz2;  //prazni su
  string niz3="111";
  string niz4(3,'8'); // niz "888"

   niz1="123";
   niz2 += niz3+" 333 "+"_"+niz1;
   niz4 += niz2;
   cout << niz2+"!"+'\n'+niz4<< endl;
  if (niz1=="123" && niz4=="888"+niz2) cout << "jednake vrednosti" << endl;
  else cout << "nejednake vrednosti" << endl;
    return 0;
}
```

Izlaz
111 333 _123!
888111 333 _123
jednake vrednosti

3. // string::length

```
#include <iostream>
#include <string>

int main ()
{
  std::string str ("Mika Pera Laza");
  std::cout << "Velicina stringa iznosi " << str.length() << " karaktera.\n";
  return 0;
}
```

IZLAZ
Velicina stringa iznosi 14 karaktera.

4. **// string::size**

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

```
int main ()
{
  string str ("Pera Mika Laza");
  cout << "Velicina stringa isnosi " << str.size() << " bajtova-karaktera.\n";
  return 0;
}
```

IZLAZ
Velicina stringa isnosi 14 bajtova-karaktera.

5.
```
// string::find
  // trazenje po stringu

#include <iostream>       // std::cout
#include <string>         // std::string
```
using namespace std;

```
int main ()
{
  string str ("Milovana voli Ana Ana.");
  string str2 ("Ana");

  // razlicite verzije pretrage find, rfind, find_first_of, find_last_of
  cout << str.find(str2) << endl;
  cout << str.rfind(str2) << endl;
  cout << str.find_first_of("aeiou") << endl;
  cout << str.find('l') << endl;
  cout << str.find_last_of("aeiou") << endl;
cout << str.find_first_not_of("abc") << endl;
cout << str.find_last_not_of('l') << endl;

  return 0;
}
```

IZLAZ	OBJASNJENJE
14	//prva pojava Ane je od pozicije 14! Oprez, znak M je na poziciji 0
18	//druga pojava Ane je od pozicije 18
1	// prvi se pojavljuje znak a na poziciji 1
2	// znak l se pojavljuje na poziciji 2
20	// poslednja pojava znaka a je na poziciji 20
0	// znak M se razlikuje i od a, i od b i od c
21	//znak . je poslednji karakter (prvi otpozadi) koji se razlikuje od slova l

6. U stringu ab+cd=abcd nadji prvi znak koji ne pripada engleskoj abecedi.

```
// string::find_first_not_of
#include <iostream>       // std::cout
#include <string>         // std::string
```
using namespace std;

```
int main ()
{
```

```
  string str ("ab+cd=abcd");

  int nasao= str.find_first_not_of("abcdefghijklmnopqrstuvwxyz");

  if (nasao>=0)
  {
    cout << "Prvi znak koji ne pripada engleskoj abecedi je " << str[nasao];
    std::cout << " na poziciji " << nasao << '\n';
  }

  return 0;
}
```

IZLAZ
Prvi znak koji ne pripada engleskoj abecedi je + na poziciji 2

7. Napisati C++ program koji ce od stringa koji sadrzi sva velika slova engleske abecede napraviti string koji ne sadrzi prvih 5 slova.

```
#include <iostream>       // std::cout
#include <string>         // std::string
using namespace std;

int main ()
{
 string str ("ABCDEFGHIJKLMNOPQRSTUVWXYZ");

 cout << str << endl;
 str.erase (0,5);  //izbrisi karakter pocev od pozicije 0 i ostalih 5 karaktera
 cout << str << endl;

 return 0;
}
```

IZLAZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
FGHIJKLMNOPQRSTUVWXYZ

8. Napisati C++ program koji ce ukloniti beline sa kraja stringa „12345678 “.
SAVET: Koristite funkciju find_last_not_of i erase

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

int main ()
{
 std::string str ("12345678   \t");
 std::string beline (" \t\f\v\n\r");

 cout << '[' << str << "]\n";
 int nasao = str.find_last_not_of(beline);
 if (nasao >=0)
   str.erase(nasao+1);

 cout << '[' << str << "]\n";

 return 0;
```

}

IZLAZ
[12345678]
[12345678]

9. Prevodjenje malih slova stringa u velika slova

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
     string str1,str2;
     str1="!abcdefghijkl!";

//kopiranje stringova moze se obaviti i preklopljenim operatorom dodele vrednosti!
     str2=str1;

     cout<<"ovo je izvorni string: "<<str1<<endl;

     for (int i=0; str1 [i]; i++) str2 [i] = toupper (str1[i]);
     cout << str2 << endl;
     return 0;
}

IZLAZ
 ovo je izvorni string: !abcdefghijkl!
!ABCDEFGHIJKL!
```

10. Napisati program koji ucitava string i proverava da li je palindrom.

ULAZ IZLAZ
anavolimilovana Da
neven Da
oko Da
Ana Ne
12321 Da

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s,t;
    int i;
    cin >>s;
    t=""; // postavljanje pocetne vrednosti stringa na "0", odnosno na prazan string


  //s.length() vraca duzinu stringa s, samim tim pocetna vrednost brojaca i je duzina
// stringa -1 (zato sto indeksi znakova u stringu pocinju od 0)

    for (i=s.length()-1;i>=0;i--)
    {
       t=t+s[i];// pravimo string koji se dobija citanjem stringa s unazad
    }

    if (s==t) cout <<"Da!";
    else cout <<"Ne!";
    return 0;
```

}

Осмислите решење које не користи додатни стринг.

РАД СА РЕГУЛАРНИМ ИЗРАЗИМА - ПРЕПОЗБАВАЊЕ ШАБЛОНА (датуми, опис интервала,...)

Најједноставнији начин да се задатак реши је да се употребе регуларни изрази. Регуларним изразима се описују шаблони текста на основу којих можемо вршити претрагу и замену.

Основна врста регуларног израза је ниска. На пример, регуларни израз је `abc` и једина ниска која је тим изразом покривена је управо `abc`.

Регуларни изрази се комбинују следећим операторима:

`|` - овај оператор се чита "или". На пример, регуларни израз `abc|def` је шаблон у који се уклапају две могуће ниске: прва је `abc`, а друга `def`.

`*` - овај оператор се чита "нула или више пута". На пример, регуларни израз `a*` означава да се `a` јавља нула или више пута и обухвата празну ниску, затим `a`, `aa`, `aaa` и тако даље.

`+` - овај оператор се чита "један или више пута". На пример, регуларни израз `a+` означава да се `a` јавља један или више пута и обухвата ниске `a`, `aa`, `aaa` и тако даље.

`?` - овај оператор се чита "нула или један пут". На пример, регуларни израз `a?` описује или празну ниску или ниску `a`.

Када се на израз допише `{n}` то означава да се ниска описана изразом понавља тачно `n` пута. На пример, `a{5}` означава ниску `aaaaa`.

Регуларни изрази постављени један поред другог представљају дописивање ниски. На пример, `(ab)+(cd)?` означава да се `ab` јавља један или више пута, за чим може, а не мора да следи `cd`. Неке ниске које су описане овим изразом су `ab`, `abcd`, `abab`, `ababcd` итд.

- Изрази облика `[a-z]` су такозване карактерске класе и представљају један карактер из датог распона. Израз `[a-z]` описује ниске `a`, `b`, `c` итд., док израз `[0-9]` описује једну цифру.

- Израз `.` описује било који карактер, осим преласка у нови ред.

Постоји још велики број регуларних операција и начина да се регуларни изрази компактније запишу. Датуми првог облика се могу описати регуларним изразом `[0-9][0-9]?[.][0-9][0-9]?[.][0-9]{5}`, а другог облика изразом `[0-9]{2}/[0-9]{2}/[0-9]{4}`.

Рад са регуларним изразима у језику C++ постиже се коришћењем regex библиотеке. Методом `regex_search` се проверава да ли дата ниска садржи неко појављивање подниске која се уклапа у шаблон описан регуларним изразом.
Методом `regex_match` добија се објекат `regex` који садржи информације о препознатим нискама, односно одређује колико је пута шаблон препознат. Заграде у реглуарном изразу служе да промене подразумевани приортет оператора, али и више од тога.
Наиме, ниска која је препозната неким заграђеним делом регуларног израза се може одредити идексирањем објекта добијеног методом `match`.

```
#include <iostream>
#include <regex>

using namespace std;

string pad(string s) {
  return s.size() == 1 ? ("0" + s) : s;
}

int main() {
  string format1 = "([0-9][0-9]?)[.]([0-9][0-9]?)[.]([0-9][0-9][0-9][0-9])[.]";
  string format2 = "([0-9]{2})/([0-9]{2})/([0-9]{4})";
  regex datum("(" + format1 + "|" + format2 + ")");
  smatch sm;
  string linija;
  while (getline(cin, linija))
    if (regex_search(linija, sm, datum)) {
      string prepoznat_datum = sm[0];
      if (regex_match(prepoznat_datum, sm, regex(format1)))
        cout << pad(sm[1]) << "." << pad(sm[2]) << "." << sm[3] << "." << endl;
      else if (regex_match(prepoznat_datum, sm, regex(format2)))
        cout << sm[2] << "." << sm[1] << "." << sm[3] << "." << endl;
    }
  return 0;
}
```

Bibliotečka podrška za rad sa stringovima

https://arena.petlja.org/competition/stringovi2019

1 Upoznavanje

Stringovi su nizovi karaktera, u ovom slučaju posebno definisani u standardnoj biblioteci programskog jezika koji koristimo, C++. Da bismo koristili stringove, potrebno je da uključimo biblioteku *string* na početku našeg koda. Sintaksa za korišćenje stringova i korisne funkcije:

```
string s, t; //deklaracija
cin >> s; //ucitavanje
t = "aaa"; //dodeljivanje vrednosti stringu t
cout << s.size() << endl; //ispisivanje duzine stringa s
string q = s + t; //konkatenacija (nadovezivanje)
q += s; //nadovezivanje s na kraj q
sort(q.begin(), q.end()); //sortiranje stringova, potrebna biblioteka
    algorithm
reverse(q.begin(), q.end()); //obrtanje stringa, potrebna biblioteka
    algorithm
cout << q[3] << endl; //ispisivanje karaktera na indeksu 3
cout << q.substr(3, 4) << endl; //ispisivanje podstringa koji pocinje od
                                   indeksa 3, duzine 4
```

2 Zadaci

1. Za uneti string, koji se sastoji samo od malih slova engleskog alfabeta, odrediti da li je on palindrom.
   Ulaz: U prvom redu nalazi se string, dužine ne veće od 100000 karaktera, koji se sastoji samo od malih slova engleskog alfabeta.
   Izlaz: Ispisati *Da* ako jeste, *Ne* ako nije palindrom.

Ulaz	Izlaz	Ulaz	Izlaz
anavolimilovana	Da	abdcba	Ne

2. Napiši program koji od unete reči pravi novu reč sastavljenu iz delova početne reči. Na primer, za uneto

```
dabrakabmra

6 2
3 3
1 1
0 3
9 2
```

program gradi i ispisuje reč abrakadabra (prvo idu 2 karaktera krenuvši od pozicije 6 tj. ab, zatim 3 karaktera od pozicije 3 tj. rak, zatim 1 karakter od pozicije 1 tj. a, zatim tri karaktera od pozicije 0 tj. dab i na kraju dva karaktera od pozicije 9 tj. ra).
Ulaz: U prvoj liniji standardnog ulaza je string koji predstavlja datu reč (dužina mu je najviše 50000 karaktera) i sadrži samo slova engleske abecede. U narednim linijama (njih najviše 10000) se unose parovi: pozicija u datom stringu i dužina podstringa (pozicija i dužina ispravno zadaju podnisku).
Izlaz: Reč koja se dobija navedenim postupkom.

Ulaz	**Izlaz**
maranaakkabbikopa 13 3 16 1 8 3 3 3	kopakobana

3. Napisati program kojim se za dve date reči proverava da li je druga sadržana u prvoj, ako jeste odrediti prvu poziciju na kojoj se druga reč pojavljjuje u prvoj.
Ulaz: Na standardnom ulazu nalaze se dve reči svaka u posebnoj liniji. Svaka reč ima najviše 20 karaktera.
Izlaz: Na standarnom izlazu prikazati prvu poziciju (pozicije su numerisane od 0) na kojoj se nalazi druga reč u prvoj, ako se druga reč ne nalazi u prvoj prikazati -1.

Ulaz	**Izlaz**	**Ulaz**	**Izlaz**
banana ana	1	branka ana	-1

4. Napisati program u kojem se učitavaju dva stringa A i B, pa proverava da li je moguće zameniti mesta karakterima u stringu B, tako da dobijemo string A. **Ulaz:** Na standardnom ulazu nalaze se dva stringa, svaki dužine ne veće od 100000.
Izlaz: Na standardnom ispisati *Da* ako je moguće, *Ne* ako nije.

Ulaz	**Izlaz**	**Ulaz**	**Izlaz**
ananas nsnaaa	Da	ananas bnbnbs	Ne

5. Napisati u program u kojem se učitava string A, pa proverava da li je moguće zameniti mesta karakterima u stringu A, tako da dobijemo string koji je palindrom. **Ulaz:** Na standardnom ulazu nalazi se string A, dužine ne veće od 100000.
Izlaz: Na standardnom ispisati *Da* ako je moguće, *Ne* ako nije.

Ulaz	**Izlaz**	**Ulaz**	**Izlaz**
mnoaovlliavaian	Da	abcdab	Ne

3 Rešenja

1. C++:

```cpp
#include <iostream>
#include <algorithm>
#include <string>
using namespace std;
int main() {
    string s, rev;
    cin >> s;
    rev = s;
    reverse(rev.begin(), rev.end());
    if(rev == s) cout << "Da" << endl;
    else cout << "Ne" << endl;
    return 0;
}
```

2. C++:

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    string a, rez = "";
    cin >> a;
    int x, y;
    while(cin >> x >> y) {
        rez += a.substr(x, y);
    }
    cout << rez;
    return 0;
}
```

3. C++:

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    string a, b;
    cin >> a >> b;
    if(a.size() > b.size()) swap(a, b);
    for(int i = 0; i <= b.size()-a.size(); i++) {
        if(b.substr(i, a.size()) == a) {
            cout << i << endl;
            return 0;
        }
```

```
    }
    cout << -1 << endl;
    return 0;
}
```

4. C++:

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main() {
    string a, b;
    cin >> a >> b;
    sort(a.begin(), a.end());
    sort(b.begin(), b.end());
    if(a == b) cout << "Da" << endl;
    else cout << "Ne" << endl;
    return 0;
}
```

5. C++:

```
#include <iostream>
#include <string>
using namespace std;
int cnt[30];
int main() {
    string a;
    cin >> a;
    for(int i = 0; i < a.size(); i++) {
        cnt[a[i]-'a']++;
    }
    int neparnih = 0;
    for(int i = 0; i < 26; i++) {
        if(cnt[i]%2 == 1) neparnih++;
    }
    if(neparnih > 1) cout << "Ne" << endl;
    else cout << "Da" << endl;
    return 0;
}
```