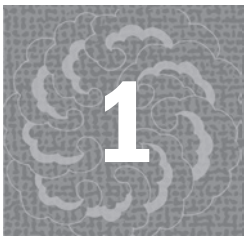


Vaši prvi C++ programi

Programiranje u C++-u nije ništa zastrašujuće – zaista! Kao i svaki programski jezik, on služi da se računaru daju logički precizna uputstva.

C++ može da se zakomplikuje koliko god želite, ali se učenje najlakše počinje rešavanjem osnovnih programerskih zadataka. To je moj pristup ovde.

U prvih nekoliko odeljaka, izlažem osnovne koncepte programiranja. Ako ste već programirali u bilo kojem jeziku, možda ćete da ih preskočite. Ali ako se zadržite, obećavam da neću da odugovlačim.

Razmišljajte programerski

Programiranje možda nije sasvim slično drugim aktivnostima kojima ste se bavili. U osnovi, vi samo dajete uputstva – ali to radite na logičan i sistematski način.

Računari rade samo ono što im naredite

Računari rade samo ono što im naredite: to je najvažnije pravilo u ovoj knjizi, pogotovo ako tek počinjete da programirate. Kad koristite programski jezik, kao što su C++, Visual Basic, Pascal, ili FORTRAN, vi računaru dajete spisak poslova; to je program.

Računaru su, naravno, potrebne informacije – to su programski *podaci*. Ali mora takođe da zna šta da radi sa tim podacima. Instrukcije koje mu nalažu šta da radi zovu se programski *kôd*.

Odredite šta će program da radi

Dakle, da bi se računar primorao da nešto radi, mora mu se reći tačno šta da radi.

Do sada ste računar verovatno koristili tako što ste izvršavali programe koje su za vas napisali drugi. Utoliko ste vi bili *krajnji korisnik* – ili skraćeno *korisnik*.

Kad sami napišete program, vi sebe promovišete na sledeći ešalon računarskog osoblja. Sada vi odlučujete šta će program da radi. Vi ćete da diktirate događanja.

Ali računar je – čak i više nego Dastin Hofman u filmu *Kišni čovek* – krajnji idiot naučnik. On nikada ne nasluti šta ste hteli. Nikada ne može da donese nezavisnu odluku. On krajnje bukvalno prihvata i izvršava tačno ono što ste rekli, bez obzira da li je to glupo. Prema tome, morate da pazite da tačno kažete ono što mislite.

Računaru čak ne možete da izdate naređenje koje bi čoveku izgledalo relativno jasno, kao što je „Konvertuj mi broj iz Celzijusa u Farenhajte". Čak i to je previše uopšteno. Morate da budete određeniji, morate da napišete postupne korake, ovako:

1. Štampati poruku „Uneti temperaturu u Celzijusovim stepenima:". 2 Prihvatiti broj sa tastature i spremiti ga u promenljivu ctemp.
3. Pretvoriti broj u Farenhajtove stepene po formuli ftemp = (ctemp * 1.8) + 32.
4. Štampati poruku „Temperatura u Farenhajtovim stepenima je:".
5. Štampati vrednost promenljive ftemp.

Zašto bi iko to radio ako je potrebna tolika muka za tako jednostavnu stvar? Odgovor je da program, kad ga jednom napišete, možete da izvršavate koliko god hoćete puta. Osim toga, mada je za pisanje programa potrebno vreme, oni se obično izvršavaju brzinom munje.

Napišite odgovarajuće C++ naredbe

Nakon što precizno odredite šta hoćete da program, korak po korak, radi, morate da napišete odgovarajuće C++ naredbe. Naredba je, grubo rečeno, C++ ekvivalent rečenice.

Na primer, recimo da želite da vaš program uradi sledeće:

1. Štampati poruku „Temperatura u Farenhajtovim stepenima je:".
2. Štampati vrednost promenljive ftemp.

Te rečenice biste preveli u C++ naredbe:

```
cout << "Temperatura u Farenhajtovim stepenima je:";
cout << ftemp;
```

Ne zaboravite, cilj programiranja je da se računar primora da obavi niz konkretnih zadataka. Ali računar razume samo svoj vlastiti maternji jezik – *mašinski kôd* – koji se sastoji od jedinica i nula. Pedesetih godina prošlog veka, programeri su zaista pisali naredbe mašinskim kodom, ali to je bilo teško i trošilo se mnogo vremena.

Da bi se posao olakšao, inženjeri su razvili programske jezike kao što su FORTRAN, Basic i C, koji omogućavaju da čovek piše programe koji imaju bar neke sličnosti sa engleskim jezikom.

Kad pišete program, možda ćete početi od *pseudokoda* – to je pristup koji ja često koristim u ovoj knjizi. Pseudokôd je sličan govornom jeziku, ali opisuje rad programa na sistematičan način koji odražava logički tok programa. Na primer, ovo je jednostavan program napisan pseudokodom:

```
Ako je a veće od b
      Štampati „a je veće od b.”
Inače
      Štampati „a nije veće od b.”
```

Kad jednom napišete pseudokod, niste daleko od C++ programa. Jedino treba da potražite odgovarajuću C++ naredbu za svaku radnju:

```
if (a > b)
      cout << "a je veće od b.";
else
      cout << "a nije veće od b.";
```

Prednost programskog jezika je to što se on vlada po pravilima koja ne dopuštaju dvosmislenost. C++ naredbe su toliko precizne da mogu da se prevedu u nule i jedinice mašinskog koda bez ikakvog nagađanja.

Ne bi trebalo da vas čudi to što programski jezici imaju stroga sintaksna pravila. Ta pravila su doslednija, i obično jednostavnija, od pravila za ljudske jezike. Povremeno u knjizi rezimiram ta pravila. Na primer, ovo je sintaksa za **if-else**:


```
if (uslov)
      naredba
else
      naredba
```

Reči koje su zacrnjene su *ključne reči*; njih morate u programu da napišete tačno kako stoje. Reči koje su iskošene su čuvari mesta; one predstavljaju stavke koje vi dodajete.

Aplikacija koja prevodi C++ naredbe u mašinski kôd se zove *kompajler* (tj. *prevodilac*). O kompajlerima će biti mnogo više reči u odeljku „Izgradnja C++ programa”. Međutim, najpre da pogledamo neke ključne definicije.

Intermeco

Koliko su računari stvarno „pametni”?

Kad sam pre više godina vodio jednu računarsku laboratoriju u Takomi (država Vašington), nailazio sam na neke zanimljive ličnosti, neke od njih pravo sa ulice. Jedan je bio debeljuškast čovečuljak u slamenom šeširu, rasparenoj odeći, starim cipelama i širokim osmehom, koji je nosio sa sobom časopis Daily Racing Form kao da je to Biblija.

Prilazio mi je svakih par dana sa tim časopisom o konjskim trkama. „Imam ideju,” stalno je ponavljao. „Zaradićemo milion. Treba samo da ubacimo ove informacije u računar, a on će da bira konje. Ja ću donositi *Daily Racing Form*, a ti napiši program. Bićemo bogati!” Osmehnuo sam se ali sam promumlao nešto o problemima izvodljivosti.

On je zaboravljao – ili verovatno uopšte nije shvatao – da u samom računaru ne postoji nikakvo pravo znanje. Unutra nije nikakvo čarobno biće koje odgovara na pitanja kao u brodskom računaru iz serije *Zvezdane staze*. Ubacivanjem sirovih podataka u računar ne postiže se ništa. Računaru prvo treba pravi *program*… niz naredbi koje mu naređuju da premesti podatke, kopira ih, sabira, oduzima, množi ili ih drugačije procenjuje.

Da bi računar pravilno birao konje koji će pobediti na trkama, pravi štos bi naravno bio u tome da se smisli pravi *algoritam*… a algoritam je sistematska tehnika kojom se korak po korak procenjuju podaci i dobija rezultat, to jest, prava formula. Slažem se da biste se obogatili kad biste smislili pravi algoritam za biranje konja. Ali u tom slučaju bi vam najmanji problem bio da pronađete računar. Svaki računar bi odgovarao, pod uslovom da može da primi potrebne podatke.

Što se tiče programera… ako biste uspeli da ga ubedite da stvarno imate pravi algoritam, jednostavno biste mu ponudili da podelite sa njim prvih 10 miliona dolara dobitaka. Poznajem dosta njih koji bi pristali.

Ili, još bolje, mogli biste da primenite tehnike programiranja opisane u ovoj knjizi, pa da sami napišete taj program.

Neke zaluđeničke definicije – pregled

Trudim se da izbegavam žargon. Zaista. Ali kad počnete da učite programiranje, vi ulazite u jedan svet koji zahteva novu terminologiju. Slede neke definicije potrebne za preživljavanje u tom svetu.

aplikacija

U suštini je isto što i program, ali gledano iz aspekta korisnika. Aplikacija je program koji korisnik izvršava da bi obavio neki zadatak. Program za obradu teksta (engl. *word processor*) je aplikacija; isto tako i veb pretraživač (engl. *Internet browser*) ili rukovalac baze podataka (engl. *database manager*). Čak je i *kompajler* (koji malo dalje definišem) takođe aplikacija, ali vrlo posebne vrste, zato što ga koriste programeri. Jednostavno rečeno, kada je program napisan, izgrađen i testiran, on postaje aplikacija.

kôd

Još jedan sinonim za „program", ali gledano iz aspekta programera. Izraz *kôd* potiče iz vremena kada su programeri pisali mašinski kôd. Svaka mašinska instrukcija se kodira jedinstvenom kombinacijom nula i jedinica, pa tako predstavlja računarev kôd za obavljanje određene akcije.

Programeri i dalje koriste izraz kôd čak i kada pišu u jezicima kao što su C++, Java, FORTRAN ili Visual Basic. (Pogledajte definiciju *izvornog koda*.) Osim toga, izraz *kôd* se koristi da bi se razgraničile pasivne informacije u programu (njegovi podaci) i deo programa koji izvršava akcije (njegov kôd).

kompajler

Prevodilac jezika koji kao ulaz prima C++ naredbe (to jest, C++ izvorni kôd), a kao izlaz proizvodi program u obliku mašinskog koda. To je neophodno, zato što sam računar – njegov procesor (CPU) – razume samo mašinski kôd.

podaci

Informacije nad kojima se izvršava program. Na najosnovnijem nivou, ove informacije se sastoje od reči i/ili brojeva.

mašinski kôd

Maternji jezik procesora, u kojem se svaka računarska instrukcija sastoji od jedinstvene kombinacije (ili *koda*) jedinica i nula. Programiranje u mašinskom kodu je još uvek moguće, ali je za to potrebno potražiti svaku instrukciju i treba detaljno poznavati arhitekturu procesora.

Jezici kao što je C++ omogućavaju da se pišu programi sličniji engleskom jeziku, ali su ipak dovoljno logički precizni da mogu da se prevedu u mašinski kôd.

program

Niz instrukcija koje računar treba da izvrši, zajedno sa početnim podacima. Kao što sam već spomenuo, za pisanje programa potrebno je vreme, ali kad ga jednom napišete, on se obično izvršava brzinom munje i možete da ga izvršavate koliko god hoćete puta.

izvorni kôd

Program napisan na jeziku višeg nivoa, kao što je C++. Izvorni kôd mora da se prevede u mašinski kôd da bi mogao da se izvrši na računaru. Mašinski kôd se obično predstavlja u heksadecimalnom obliku (osnova 16), pa izgleda ovako:

```
08 A7 C3 9E 58 6C 77 90
```

Ne vidi se šta se time postiže, zar ne? Takav program može da se razume tek kad se potraže svi kodovi instrukcija – pa zato skoro niko više ne piše mašinskim kodom. Naprotiv, izvorni kôd bar donekle liči na engleski. Sledeća C++ naredba znači „Ako je plata manja od 0, štampati poruku o grešci”:

```
if (plata < 0)
     štampanje_poruke();
```

naredba

Jednostavna naredba obično odgovara jednom redu C++ programa, a završava se znakom tačka i zarez (kao što se u običnom tekstu završava tačkom). Ali C++ podržava složene naredbe, kao što u tekstu postoje složene rečenice, pa se one protežu na više redova. Većina C++ naredbi vrši jednu akciju, mada neke vrše više akcija.

korisnik

Osoba koja izvršava program – to jest, osoba koja koristi računar da bi uradila nešto korisno, kao što je obrada tekstualnog fajla, čitanje elektronske pošte, istraživanje Interneta ili saldiranje tekućeg računa. Zvaničnije ime za korisnika je *krajnji korisnik*.

Dok sam radio u Microsoftu, korisnik je bila osoba koja je izazivala najviše problema na svetu, ali takođe i osoba koja je plaćala račune. Makar ta hipotetična osoba bila kompletan ”duduk” za tehniku, Microsoft ne bi postojao ako korisnici ne bi kupovali njegove proizvode. Prema tome, kad počnete da projektujete ozbiljne programe, važno je da pažljivo razmotrite potrebe korisnika.

Po čemu je C++ drugačiji?

Većina onoga što sam upravo rekao za C++ odnosi se i na druge programske jezike kao što su Pascal, Java, FORTRAN i Basic. To su sve *jezici višeg nivoa* , što znači da nisu tačna slika mašinskog koda, već koriste ključne reči (kao što su „if" i „while") koje imaju sličnosti sa engleskim jezikom.

Svaki jezik je razvijen za drugu svrhu. Basic je projektovan da bi se lako učio i koristio. Zbog toga on dozvoljava opušteniju sintaksu što, nažalost, može da dovede do loših navika u programiranju. Ipak, Microsoft je razvio Visual Basic u moćnu, ugodnu i brzu alatku za izradu aplikacija za Microsoft Windows i pri tom sredio nešto od te aljkavosti.

Pascal je razvijen da bi se koristio u akademskim okruženjima, da bi se predavali složeni programski koncepti. Pascal je razrađeniji od Basica, ali ne može da uradi sve što može C ili C++.

Legendarni računarski naučnik Denis Riči (Dennis Richie) je prvobitno projektovao C kao pomoć u pisanju operativnih sistema. To je uredan jezik koji podržava prečice i omogućava pisanje konciznijih programa. Njegova neposredna a ipak opsežna sintaksa dokazano je popularna među programerima celog sveta. Još jedna prednost C-a je to što postavlja manje ograničenja od drugih jezika.

A C++?

Glavna razlika među jezicima C i C++ je u tome što C++ dodatno omogućava *objektno-orijentisano programiranje*. To je pristup posebno dobro prilagođen radu sa kompleksnim sistemima, kao što su grafički korisnički interfejsi i mrežna okruženja. Kao objektno orijentisani programer, imaćete sledeća pitanja:

1. Koje su glavne vrste podataka (to jest, informacija) u problemu koji se rešava?
2. Koje operacije treba definisati za svaku vrstu podataka?
3. Kakva je interakcija među objektima podataka?

Dok sam učio objektno orijentisano programiranje, nalazio sam da je lakše ako se najpre savlada osnovna sintaksa naredbi. Prema tome, na objektnu orijentaciju se usredsređujem tek u poglavlju 11.

Ali neke objekte – komade podataka koji mogu da reaguju na operacije – uvodim već rano u knjizi. Na primer, u ovom poglavlju, koristim **cout**, objekat koji nije iz jezika C. U jeziku C biste informacije štampali pozivanjem funkcije (unapred definisanog skupa naredbi). Ali kada koristite **cout**, vi podatke šaljete objektu koji – stvarno – *zna kako* da prikaže informacije.

To se pokazuje kao savršeniji način da se obavi posao: **cout** zna kako da štampa mnoge vrste podataka. Što je još najbolje, možete vlastite, prilagođene tipove podataka (klase) da proširite tako da glatko sarađuju sa izlaznim objektima kakav je **cout**. To ne biste mogli (bar ne lako) u C-u.

Izgradnja C++ programa

Pisanje programa je samo prvi korak u pravljenju aplikacije. Najpre se unose programske naredbe.

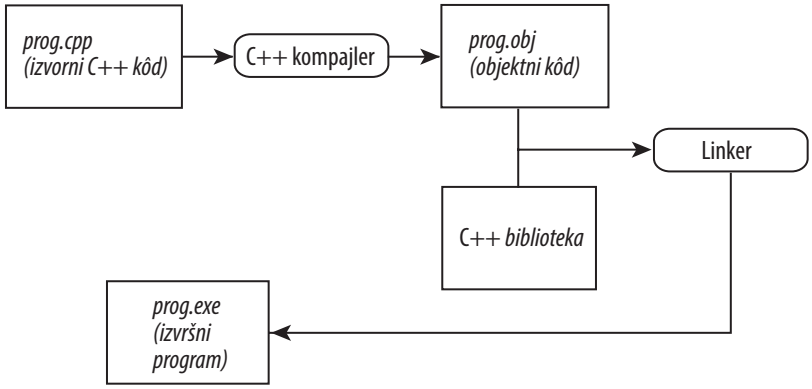
Unošenje programskih naredbi

Da bi se napisao C++ program, treba vam način da unesete programske naredbe:

- Možete da upotrebite tekst editor, kao što je Microsoft Word ili Notepad. Ako to radite, morate da sačuvate dokument kao običan tekstualni fajl/ (engl. *plain-text*).
- Možete da unosite tekst u integrisano okruženje za razvoj (IDE – integrated development environment). Okruženje za razvoj je tekst editor kombinovan sa drugim pomoćnim programskim alatima. Microsoft Visual Studio je okruženje za razvoj.

Izgradnja programa (prevođenje i povezivanje)

Izgradnja programa je proces pretvaranja vašeg izvornog koda (C++ naredbi) u aplikaciju. To se u razvojnim okruženjima obično svodi na to da pritisnete funkcijski taster. Postupak se obično sastoji od dve faze: prevođenja (engl. *compile*) i povezivanja (engl. *link*) (kojim se uključuju standardne funkcije napisane za vas). Ako te dve faze uspeju, program može da se izvršava.



Ako izgradnja uspe, možete sebe da potapšete po leđima: ni kompajler ni linker nisu naišli na greške. Da li to znači da ste gotovi? Ne. Kompajler hvata sintaksne greške, kao i greške programske strukture. Ali ima mnogo grešaka koje on ne može da pronađe. Pogledajmo jednu analogiju… uzmimo da imate sledeću rečenicu:

Mesec se sastoji zelenog sira. [pazi sad]

Ta rečenica nije gramatički ispravna. Da bi se ispravila gramatika, treba da se doda reč *od*:

Mesec se sastoji *od* zelenog sira.

Sada rečenica nema sintaksne greške. Ali, da li to samim tim znači da je rečenica istinita? Naravno da ne. U ovom slučaju, potrebno je dodati reč *ne*:

Mesec se *ne* sastoji od zelenog sira.

Programski jezici su slični. C++ kompajler utvrđuje da li je program sintaksno i gramatički ispravan, pa ako nije, izvestiće u kojem redu postoji greška. Ali šire pitanje… pitanje *da li se program pravilno ponaša u svim situacijama*… nije tako očigledno, što nas dovodi do sledeće faze.

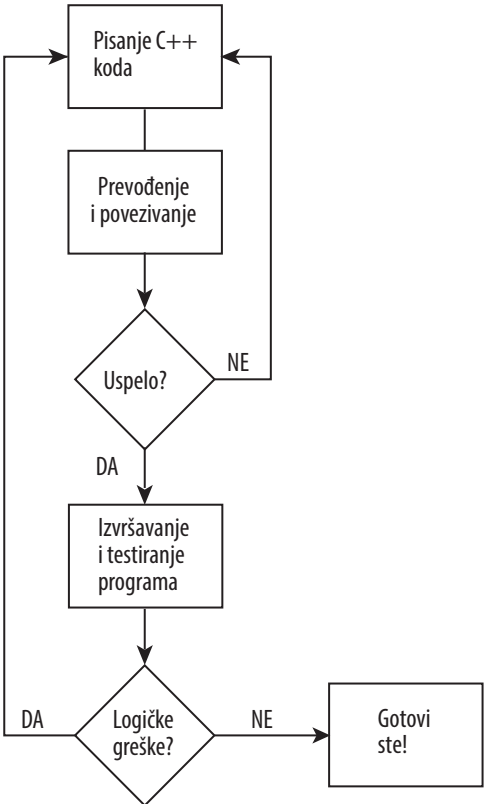
Testiranje programa

Nakon uspešne izgradnje programa, morate da ga izvršite kako biste proverili da li on radi ono što ste vi hteli. Ako je reč o ozbiljnom programu – programu koji se predaje ili prodaje drugim ljudima – možda ćete morati da ga testirate mnogo puta.

Greške koje se traže u ovoj fazi su *greške programske logike*. One mogu da budu daleko neuhvatljivije od sintaksnih grešaka. Uzmimo da program odštampa pogrešan broj ili prestane da se izvršava bez ikakvog vidljivog razloga. Koja naredba je dovela do greške? Proces testiranja i utvrđivanja izvora problema zove se *otklanjanje grešaka* (engl. *debugging*).

Prerada po potrebi

Ako se program pravilno izvršava, završili ste. Ali ako postoje greške programske logike, morate da utvrdite izvor greške, da izvršite izmene u programu i ponovo ga izgradite.



Ako je reč o složenom softveru, možda ćete ovaj ciklus morati da ponavljate više puta. Za takav program biće potrebno mnogo testiranja kako bi se proverilo da li se pravilno ponaša u svim slučajevima. Program nije završen sve dok se ne obavi takvo testiranje i doterivanje. Ali za jednostavne programe, obično je dovoljna umerena količina testiranja.

Instaliranje vlastitog C++ kompajlera

Veb sajt Pearson (*www.informit.com/title/ 9780132673266*) sadrži link sa kojeg možete da preuzmete besplatne i šerver kompajlere, kao i okruženje Dev-C++ (koje toplo preporučujem). Ono sadrži kompletno razvojno okruženje u kojem možete da pišete, prevodite i testirate svoje programe.

Primer 1.1 *Štampanje poruke*

Da biste počeli da programirate, otvorite nov izvorni fajl i unesite sledeći kôd:

print1.cpp

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

int main(){
      cout << "Bez brige, C++ je ovde!";
      system ("PAUSE");
      return 0;
}
```

Napomena ▶ Neke stvari ćete morati da uradite drugačije ako koristite Visual Studio ili Dev-C++, pa nemojte da očekujete da će kôd da se prevede bez greške dok ne pregledate taj deo.

Ne zaboravite da nije važno precizno razmicanje, ali je važno paziti na upotrebu velikih i malih slova.

Ako koristite Visual Studio ili okruženje Dev-C++, pojaviće se neki dodatni redovi koda; ostavite ih. Osim toga, moraćete da otvorite novi projekat.

Ako koristite okruženje Dev-C++, idite na meni File, izaberite New, otvorite nov projekat, kao tip projekta izaberite Console Application, pa odaberite neko ime projekta. Zatim u prozoru Project (na levoj strani ekrana) kliknite na izvorni fajl main.cpp. Dev-C++ će vam napraviti sledeći kôd:

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

int main() {
      system("PAUSE");
      return 0;
}
```

U ovom slučaju, dovoljno je da dodate naredbu koja počinje sa **cout**; ubacite je tačno ispod reda koji glasi `int main() {` a iznad reda `system("PAUSE");`. Ostale redove ne dirajte.

Pošto unesete program, sačuvajte ga kao print1, prevedite (compile) ga i izvršite (run). Ako se pravilno unese i izvrši, program će štampati sledeće:

```
Bez brige, C++ je ovde!
```

Ako koristite okruženje Dev-C++

Ako ste instalirali šerver verziju C++-a opisanu u prethodnom odeljku, evo kako da izvršite program:

1. Pritisnite F9 da bi se program izgradio i izvršio.
2. Ako ne primite nijednu poruku o grešci, program se uspešno preveo i povezao. Čestitke! Ako ste primili neku poruku, znači ili da instalacija kompajlera nije bila uspešna, ili da ste prilikom unošenja primera napravili neku grešku kucanja. Vratite se i proverite da li ste svako slovo – uključujući interpunkciju – kucali tačno kako je navedeno.

Ako koristite Microsoft Visual Studio

Ako za pisanje C++ programa koristite Microsoft Visual Studio, morate da uradite još neke dodatne stvari. Visual Studio je odlična alatka za pisanje programa, ali je pre svega projektovan za pravljenje ozbiljnih Windows aplikacija, a ne za jednostavne programe (što treba prvo da radite ako tek učite C++).

Da biste u okruženju Visual Studio napisali program, treba prvo da postavite pravu vrstu projekta. (U žargonu Visual Studija, *projekat* su fajlovi koji svi zajedno čine program.)

1. Na meniju File, izaberite podmeni File, a na njemu komandu New. Možete takođe da pritisnete dugme New Project ako je već prikazano na sredini ekrana.
2. Popunite okvir za dijalog pritiskom na Console Application kao tip projekta. Osim toga, unesite i ime programa – u ovom slučaju, print1 – pa pritisnite OK.
3. Ako se ne prikaže fajl print1.cpp, potražite ga na listi imena fajlova na levoj strani ekrana, pa ga pritisnite dva puta.

Pre nego što počnete da unosite kôd, najpre izbrišite sav kôd koji vidite u fajlu print1.cpp, osim sledećeg:

```
#include "stdafx.h"
```

Ova naredba mora da postoji u konzolnim aplikacijama (to jest, aplikacijama koje nisu Windows) koje se prave u Visual Studiju. Ako za praćenje ove knjige koristite Visual Studio, ne zaboravite da stavite taj red na početak svakog programa.

Prema tome kôd print1.cpp bi trebalo da izgleda ovako (dodatni red je zacrnjen):

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main(){
      cout << "Bez brige, C++ je ovde!";
      system ("PAUSE");
      return 0;
}
```

Da biste izgradili program, pritisnite F7. Time se pokreće i kompajler i linker.

Ako ste uspeli, čestitam! Krenuli ste. Ako se program nije uspešno izgradio, vratite se i proverite da li ste sve redove uneli doslovce.

Da biste izvršili program, pritisnite CTRL+F5. Mada postoje i drugi načini da se u Visual Studiju program izvrši, ovo je jedini način da se izbegne problem sa MS-DOS prozorom koji blesne na ekranu i odmah nestane. Kad pritisnete CTRL+F5 (Start without debugging – pokrenuti bez traženja grešaka) program se izvrši a zatim se pojavi korisna poruka „Press any key to continue" koja vam omogućava da vidite rezultat i sami ga uklonite pritiskom na bilo koji taster.

Napomena ▶ Naredba `system("PAUSE");` je potrebna ako koristite operativni sistem Windows kako konzolna aplikacija ne bi previše brzo nestala. Pošto nije sastavni deo C++-a, biće preskočena u svim narednim primerima u knjizi. Uvek je možete sami dodati iznad reda naredbe return 0; ako bude potrebno.

Kako ovo radi

Verovali ili ne, ovaj jednostavan program sadrži samo jednu pravu naredbu. Ostatak možete za sada da smatrate šablonom – nešto što mora da se stavi, ali može bez brige da se zanemari. (Ako vas interesuju detalji, u sledećem "Intermecu" se opisuje direktiva **#include**.)

U sledećoj sintaksi su zacrnjeni standardni, obavezni delovi. Za sada, ne brinite zašto su obavezni; samo ih upotrebite. Među vitičaste zagrade ({ }), stavićete redove vašeg programa – koji se u ovom slučaju sastoji samo od jedne važne naredbe.

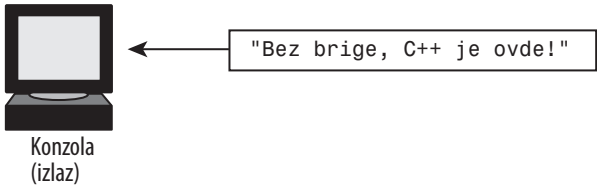
```
#include <iostream>
using namespace std;

int main(){
      Ovde_stavite_svoje_naredbe!
      return 0;
}
```

U ovom programu postoji samo jedna prava naredba (koju ubacujete u peti red prethodnog šablona). Nemojte da zaboravite znak tačka i zarez (;)!

```
cout << "Bez brige, C++ je ovde!";
```

Šta je to **cout**? To je jedan *objekat* – o tom konceptu će biti mnogo reči u drugoj polovini knjige. U međuvremenu, dovoljno je da znate da **cout** znači „izlaz na konzolu" (engl. *console output*). Drugim rečima – on predstavlja ekran računara. Kada pošaljete nešto na ekran, to se tamo štampa, kao što biste i očekivali.

U C++-u se izlaz štampa tako što se upotrebi **cout** i levi operator „toka" (<<) koji pokazuje tok podataka od vrednosti (u ovom slučaju, tekstualnog stringa „Bez brige, C++ je ovde!") ka konzoli. Zamislite slikovito:




```
cout    <<      "Bez brige, C++ je ovde!";
```

Nemojte zaboraviti znak tačka i zarez (;). Svaka C++ naredba, uz vrlo malo izuzetaka, mora da se završi znakom tačka i zarez.

Iz tehničkih razloga, **cout** mora uvek da se nalazi na levoj strani kad god se koristi. U ovom slučaju podaci teku ulevo. Primenite „strelice" ulevo, to su ustvari dva znaka manje od (<<), jedan do drugog.

U sledećoj tabeli su još neke jednostavne primene objekta **cout**:

NAREDBA	AKCIJA
`cout << "Volite li C++?";`	Štampa reči „Volite li C++?"
`cout << "Mislim,";`	Štampa reč „Mislim,"
`cout << "Dakle programiram.";`	Štampa reči „Dakle programiram."

VEŽBE

Vežba 1.1.1. Napišite program koji štampa poruku „Uključi se u program!" Ako želite, možete da radite na istom izvornom fajlu koji smo koristili kao primer i promenite ga prema potrebi. (Savet: izmenite jedino tekst u znacima navoda; zadržite sav preostali programski kôd.)

Vežba 1.1.2. Napišite program koji štampa vaše ime.

Vežba 1.1.3. Napišite program koji štampa „Volite li C++?"

Intermeco

Šta su to #include i using?

Rekao sam da je peti red u programu prva „prava" naredba u programu. Zataškao sam prvi red:

```
#include <iostream>
```

To je jedan primer *predprocesorske direktive* u C++-u, opšteg uputstva C++ kompajleru. Direktiva u vidu;

```
#include <imefajla>
```

učitava deklaracije i definicije za podršku dela standardne biblioteke C++-a (ili drugih biblioteka). Da nije bilo te direktive, ne bismo mogli da koristimo **cout.**

Ako ste koristili starije verzije jezika C ili C++, možda se pitate zašto nije naveden konkretan fajl (na primer, neki .h fajl). Ime fajla iostream je *virtuelni jnclude fajl* koji sadrži informacije u unapred prevedenom formatu.[1]

Ako tek učite C++, samo zapamtite da morate da koristite **#include** da bi se uključili određeni delovi standardne C++ biblioteke. Kasnije, kad počnemo da koristimo matematičke funkcije kao što je **sqrt** (square root – kvadratni koren), moraćete da uključite podršku za matematičku biblioteku:

```
#include <cmath>
```

Da li je to dodatni posao? Pomalo, jeste. Include fajlovi su nastali zbog razlika između jezika C i standardne izvršne biblioteke (engl. *standard runtime library*). (Profesionalni C i C++ programeri ponekad izbegnu standardnu biblioteku i radije koriste vlastitu.) Bibliotečke funkcije i objekti – mada su za početnike neophodni – tretiraju se isto kao i korisnički defi-

1 Po specifikaciji C++-a, zaglavlja standardne biblioteke nemaju nastavak '.h'. Nedostatak nastavka nema veze sa „virtuelnim zaglavljima" što možete i proveriti tako što ćete u instalaciji DevC++-a (koji autor preporučuje) naći taj fajl i otvoriti ga nekim tekst editorom. Unapred prevedena zaglavlja (PCH – precompiled headers) su tehnika koju moderni kompajleri koriste radi ubrzanja procesa prevođenja programa. *~Prim. rec.*

▼ *nastavlja se na sl. strani*