PREVOD DRUGOG IZDANJA

EFIKASNI C

UVOD U PROFESIONALNO PROGRAMIRANJE U JEZIKU C

ROBERT C. SIKORD

AŽURIRANO OBUHVATA VERZIJU C23

kompjuter biblioteka

no starch press

EFIKASNI C

UVOD U PROFESIONALNO PROGRAMIRANJE U JEZIKU C

ROBERT C. SIKORD

kompjuter biblioteka

no starch press

Izdavač:

kompjuter biblioteka

Obalskih radnika 4a
Beograd, Srbija

Tel: 011/2520272

e-pošta: kombib@gmail.com

veb-sajt: www.kombib.rs

Za izdavača:
Nemanja Lukić, direktor

Autor: Robert C. Sikord

Prevod: Mihailo Zoin
Recezent: Miroslav Ristić
Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf", Zemun
Tiraž: 500
Godina izdanja: 2024.

Broj knjige: 578
Izdanje: Prvo
ISBN: 978-86-7310-607-6

Naslov originala:

EFFECTIVE C, 2ND EDITION

ISBN: 978-1-7185-0412-7

No Starch Press, Inc.
245 8th Street, San Francisco, CA 94103
phone: 1.415.863.9900
www.nostarch.com

EFIKASNI C

Autorizovani prevod sa engleskog jezika.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004.432.2C

СИКОРД, Роберт Ц.-
Efikasni C : uvod u profesionalno programiranje u jeziku C / Robert C. Sikord; [prevod Mihailo Zoin]. - Izd. 1. - Beograd: Kompjuter Biblioteka, 2024 (Zemun: Pekograf). - XXII, 281 str.: ilustr.; 24 cm. - (Kompjuter biblioteka; br. knj. 578)

Prevod dela: Effective C / Robert C. Seacord. - Tiraž 500. - O autoru: str. III. - Bibliografija: str. 267-270. - Registar.

ISBN 978-86-7310-607-6

а) Програмски језик „C"

COBISS.SR-ID 159152393

O AUTORU

Robert C. Sikord (rcs@robertseacord.com) je vodeći stručnjak za standardizaciju u kompaniji Woven by Toyota, gde radi na unapređenju softverskog zanata. Prethodno je bio tehnički direktor u NCC Group, rukovodilac inicijative za bezbedno kodiranje na Institutu za softversko inženjerstvo Univerziteta Karnegi Melon, i vanredni profesor Škole za računarske nauke i Institutu za mrežno umrežavanje na istom univerzitetu. Robert je koordinator međunarodne radne grupe za standardizaciju programskog jezika C, ISO/IEC JTC1/SC22/WG14. Autor je i više knjiga, uključujući *The CERT® C Coding Standard*, drugo izdanje (Addison-Wesley, 2014); *Secure Coding in C and C++*, drugo izdanje (Addison-Wesley, 2013); i *Java Coding Guidelines*: *75 Recommendations for Reliable and Secure Programs* (Addison-Wesley, 2014). Takođe je objavio više od 50 radova na teme bezbednosti softvera, softverskog inženjeringa zasnovanog na komponentama, dizajna sistema zasnovanih na veb tehnologijama, modernizacije nasleđenih sistema, spremišta softverskih komponenti i pretraživača, kao i dizajna i razvoja korisničkih interfejsa.

O SARADNIKU

Aron Balman (aaron@aaronballman.com) je vodeći održavalac kompajlera Clang, popularnog kompajlera otvorenog koda za C, C++ i druge jezike. Aron je stručnjak u odborima za standardizaciju programskih jezika C (JTC1/SC22/WG14) i C++ (JTC1/SC22/WG21). Njegov profesionalni fokus je na pomaganju programerima da prepoznaju greške u svom kodu kroz bolje dizajniranje jezika, dijagnostiku i alate. Kada ne razmišlja o programiranju, voli da provodi mirne trenutke sa porodicom u šumama ruralnog Mejna.

O TEHNIČKOM RECENZENTU PRVOG IZDANJA

Martin Sebor je glavni softverski inženjer u GNU Toolchain timu u kompaniji Red Hat. Njegov rad se fokusira na GCC kompajler i oblasti detekcije, dijagnostikovanja i prevencije bezbednosnih problema u C i C++ programima, kao i implementaciju optimizacija algoritama zasnovanih na radovima sa niskama. Pre nego što se pridružio Red Hat-u 2015. godine, radio je kao inženjer za kompajlerske alate u kompaniji Cisco. Martin je član Odbora za standardizaciju jezika C++ od 1999. godine i član Odbora za jezik C od 2010. Živi sa suprugom u blizini malog grada Lajons u Koloradu.

O TEHNIČKOM RECENZENTU DRUGOG IZDANJA

Vincent Melol je viši inženjer za bezbednost proizvoda u kompaniji Woven by Toyota, gde dizajnira i razvija C softver za pouzdano izvršno okruženje elektronskih upravljačkih jedinica u automobilima. Član je kompanijskog odbora za kriptografiju i, zajedno sa Robertom, uspostavio je smernice za bezbedno i sigurno kodiranje unutar kompanije. Pre nego što se pridružio kompaniji Woven 2020. godine, bio je vođa tima za testiranje proboja u ESCRYPT Japan, podružnici Bosch Group, specijalizovanoj za bezbednost u automobilskoj industriji. U slobodno vreme, Vincent aktivno doprinosi i održava CAN podsistem Linux jezgra, poznat i kao SocketCAN.

O RECENZENTU SRPSKOG IZDANJA

Miroslav Ristić je redovni profesor na Prirodno-matematičkom fakultetu Univerziteta u Nišu, sa preko 25 godina iskustva u razvoju statističkog softvera. Posebno se ističe njegov rad na razvoju grafičkog korisničkog interfejsa R Commander za programski jezik R. Dugi niz godina recenzirao je značajan broj knjiga za izdavačku kuću Springer i časopis Journal of Applied Statistics. Od 2023. godine aktivno recenzira najaktuelnija izdanja izdavačke kuće "Kompjuter biblioteka". Nakon prevođenja, svako izdanje prolazi kroz njegovo stručno vrednovanje i recenziju prevoda, sa ciljem da se osigura da prevodi budu ne samo jasni, precizni i prilagođeni čitaocima, već i da održe visok kvalitet i stručnu relevantnost knjiga.

Kratak sadržaj

Sadržaj

PREDGOVOR DRUGOM IZDANJU

Kada sam pre više od 27 godina započeo karijeru u oblasti računarske bezbednosti, svoju veštinu sam sticao pretežno pronalazeći i istražujući nesigurno rukovanje memorijom u C programima — klasi ranjivosti koja je čak i tada bila stara preko 20 godina. Tokom karijere u kompaniji BlackBerry, dok sam pregledavao ogromne količine koda, iz prve ruke sam video koliko je C opasan za one bez odgovarajuće obuke. Danas, kao glavni tehnički direktor Nacionalnog centra za računarsku bezbednost Velike Britanije, svakodnevno viđam posledice loše napisanog koda u jeziku C u našem povezanom društvu na nacionalnom nivou.

Danas se i dalje suočavamo s brojnim izazovima u pisanju bezbednog i profesionalnog koda u jeziku C. Brojne inovacije na nivou kompajlera i operativnog sistema u cilju smanjenja ranjivosti mogu biti, i redovno jesu, zaobiđene. I dok svedočimo naprednim inovacijama u modernim jezicima i hardveru, potražnja za jezikom C i dalje raste, naročito u oblastima Interneta pametnih uređaja (IoT) i drugim okruženjima sa veoma ograničenim resursima, gde C takođe održava postojeće sisteme. Kombinacija profesionalnog koda u jeziku C i hardverskih arhitektura kao što je CHERI predstavlja način na koji obezbeđujemo okruženja koja nikada neće preći na druge jezike.

Robert je autoritet u profesionalnom i bezbednom programiranju u jeziku C. Više od decenije preporučujem njegove materijale klijentima i internim timovima. Nema bolje osobe od njega da podučava kako programirati u jeziku C na profesionalan, a između ostalog, i bezbedan način.

Pisati profesionalni kod u jeziku C danas znači pisati kod koji je efikasan, bezbedan i siguran. Time ćete doprineti našoj povezanoj zajednici, a da pritom ne povećavate njen tehnički dug.

Ova knjiga će pomoći onima sa malo ili bez ikakvog iskustva u jeziku C da brzo razviju znanje i veštine potrebne da postanu profesionalni C programeri, pružajući snažnu osnovu za razvoj sistema koji su efikasni, bezbedni i sigurni.

Oli Vajthaus
CTO, Nacionalni centar za računarsku bezbednost, Velika Britanija

PREDGOVOR PRVOM IZDANJU

Prvi put sam naišao na ime Roberta Sikorda 2008. godine. Robert je već bio poznato ime u svetu C programiranja zbog svog rada na *The CERT® C Coding Standard* i Dodatku K standarda za jezik C. Ali te 2008. godine, prošlo je tek nekoliko godina otkako sam — mlad i naivan — započeo projekat Frama-C kako bih obezbedio odsustvo nedefinisano ponašanja u C programima. U jednom trenutku, dokument *CERT Vulnerability Note* o tome kako kompajleri za C (posebno GCC) uklanjaju određene provere prekoračenja u aritmetici pokazivača, privuklo je moju pažnju. Kompajleri su imali razlog da uklone te provere; naivno napisano, oni su pozivali nedefinisano ponašanje kada bi došlo do prekoračenja.

Kompajlerima za C je takođe bilo dozvoljeno da programeru ne prijave nikakvu grešku, čak ni na najvišem nivou upozorenja. Nedefinisano ponašanje u jeziku C može biti nemilosrdno. Krenuo sam da rešim upravo ovaj problem. Robert je bio jedan od autora tog dokumenta.

Knjiga *Efikasni C* će vas naučiti programiranju u jeziku C za savremeno doba. Pomoći će vam da usvojite dobre navike koje će vas sačuvati od korišćenja nedefinisanog ponašanja, bilo namerno ili iz nehata. Neka čitalac bude upozoren: u velikim C programima, izbegavanje uobičajenih programskih grešaka možda neće biti dovoljno za izbegavanje nedefinisanog ponašanja uzrokovanog proizvoljnim unosima!

Ova knjiga pruža neuporediv naglasak na aspekte bezbednosti u programiranju u jeziku C. Moja lična preporuka je da, nakon što je pročitate, iskoristite sve dostupne alate koje predstavlja kako biste izbegli nedefinisano ponašanje u C programima koje pišete.

Paskal Kuok
Glavni naučnik, TrustInSoft

UVOD

Jezik C razvijen je kao jezik za sistemsko programiranje 1970-ih godina i, čak i posle toliko vremena, ostaje veoma popularan. Sistemski jezici osmišljeni su da omogućavaju visok učinak i jednostavan pristup osnovnom hardveru, dok istovremeno nude karakteristike višeg nivoa u programiranju. Iako drugi jezici mogu pružiti modernije funkcionalnosti, njihovi kompajleri i biblioteke često su napisani u jeziku C.

Kao što je Karl Sagan jednom rekao: „Ako želite da napravite pitu od jabuka iz početka, prvo morate da stvorite univerzum.” Tvorci jezika C nisu stvorili univerzum; oni su dizajnirali C tako da funkcioniše sa različitim računarima i arhitekturama koje su, zauzvrat, bile ograničene zakonima fizike i matematike.

Jezik C je smešten direktno iznad računarskog hardvera, što ga čini osetljivijim na razvoj hardverskih karakteristika, poput vektorskih instrukcija, za razliku od jezika višeg nivoa čija efikasnost zavisi od jezika C.

Prema TIOBE indeksu (https://www.tiobe.com/tiobe-index/) — koji rangira jezike na osnovu broja kvalifikovanih inženjera, dostupnih kurseva i dobavljača za svaki jezik — jezik C je od 2001. godine ili najpopularniji programski jezik ili drugi po popularnosti. Popularnosti jezika C može se verovatno pripisati nekoliko osnovnih principa, poznatih kao *duh jezika C*:

- Verujte programeru. Jezik C podrazumeva da znate šta radite i omogućava vam to. Ovo nije uvek dobro (na primer, ako ne znate šta radite).
- Nemojte sprečavati programera da obavi ono što treba. Pošto je jezik C za sistemsko programiranje, potrebno je da može da rukuje raznim zadacima na niskom nivou.
- Zadržite jezik malim i jednostavnim. Jezik C je dizajniran tako da bude blizak hardveru i da zauzima malo prostora.
- Obezbedite samo jedan način za izvođenje operacije. Takođe poznato kao očuvanje mehanizama, jezik C nastoji da ograniči uvođenje duplikata mehanizama.
- Neka bude brz, čak i ako to ne garantuje prenosivost. Pisanje optimalno efikasnog koda je prioritet. Odgovornost za obezbeđivanje prenosivosti, bezbednosti i sigurnosti koda prepuštena je vama, programeru.

Jezik C se koristi kao ciljni jezik za kompajlere pri izradi operativnih sistema, za obuku u osnovama programiranja, kao i za ugradne sisteme i programiranje opšte namene.

Postoji velika količina nasleđenog koda napisanog u jeziku C. Odbor za standardizaciju jezika C izuzetno pazi da ne naruši postojeći kod, omogućavajući nesmetanu modernizaciju ovog koda kako bi se iskoristile savremene karakteristike jezika.

Jezik C se često koristi u ugradnim sistemima jer je mali i efikasan. Ugradni sistemi su mali računari koji su ugrađeni u druge uređaje, kao što su automobili, kućni aparati i medicinski uređaji.

Vaš omiljeni programski jezik i biblioteke verovatno su napisani u jeziku C (ili su to bili u nekom trenutku). Dostupno je mnogo biblioteka za jezik C, što olakšava pronalaženje biblioteka koje mogu poslužiti za obavljanje uobičajenih zadataka.

Sve u svemu, jezik C je moćan i svestran jezik koji je i danas naširoko korišćen. On je dobar izbor za programere kojima je potreban brz, efikasan i prenosiv jezik.

Kratka istorija jezika C

Programski jezik C razvijen je početkom 1970-ih godina u laboratorijama Bell Labs kao jezik za implementaciju sistema za tada novi operativni sistem Unix, i danas je i dalje izuzetno popularan (Ritchie 1993). Sistemski jezici osmišljeni su da omogućavaju visok učinak i jednostavan pristup osnovnom hardveru, dok istovremeno nude karakteristike višeg nivoa u programiranju. Iako drugi jezici mogu pružiti modernije funkcionalnosti, njihovi kompajleri i biblioteke često su napisani u jeziku C. Jezik C funkcioniše kao „lingua franca” za prevođenje između različitih sistema i jezika.

Prvi opis jezika C dali su Kernigan i Riči 1978. godine u knjizi '*The C Programming Language*' (Kernighan i Ritchie 1988). Danas je jezik definisan kroz revizije ISO/IEC 9899 standarda (ISO/IEC 2024) i drugih tehničkih specifikacija. Odbor za standarde jezika C brine o njegovom očuvanju i razvoju, sarađujući sa širom zajednicom na održavanju i unapređivanju jezika C. Godine 1983. Američki nacionalni institut za standarde (ANSI) formirao je komitet X3J11 kako bi ustanovio standardnu specifikaciju za jezik C, a 1989. godine standard za jezik C je usvojen kao ANSI X3.159-1989, „Programming Language C". Ova verzija jezika iz 1989. godine naziva se ANSI C ili C89.

Godine 1990. ANSI C standard je usvojen (bez promena) od strane zajedničkog tehničkog odbora Međunarodne organizacije za standardizaciju (ISO) i Međunarodne elektrotehničke komisije (IEC) i objavljen kao prvo izdanje standarda za C, pod nazivom C90 (ISO/IEC 9899:1990). Drugo izdanje standarda, C99, objavljeno je 1999. godine (ISO/IEC 9899:1999), a treće izdanje, C11, 2011. godine (ISO/IEC 9899:2011). Četvrta verzija, objavljena 2018. kao C17 (ISO/IEC 9899:2018), ispravlja nedostatke u C11. Najnovija verzija standarda za C (u trenutku pisanja ovog teksta) je peta verzija, objavljena 2024. godine kao C23 (ISO/IEC 9899:2024). Od septembra 2023. godine, ja sam koordinator radne grupe ISO/IEC JTC1/SC22/WG14, međunarodne grupe za standardizaciju programskog jezika C.

Tokom 20 godina, koliko TIOBE indeks popularnosti prati popularnost programskih jezika, jezik C je ostao ili na prvom ili na drugom mestu (TIOBE Index 2022).

Standard jezika C

Standard za jezik C (ISO/IEC 9899:2024) definiše sam jezik i predstavlja krajnji autoritet za njegovo ponašanje. Iako standard može delovati nejasno ili teško razumljivo, potrebno je da ga razumete ako planirate da pišete prenosiv, bezbedan i siguran kod. Standard jezika C pruža značajnu slobodu implementacijama, kako bi omogućio njihovu optimalnu efikasnost na različitim hardverskim platformama. U standardu se pojam *implementacije* odnosi na kompajlere i definisan je na sledeći način:

> Specifičan skup softvera, koji radi u specifičnom prevodilačkom okruženju pod specifičnim kontrolnim opcijama, koji vrši prevod programa i podržava izvršenje funkcija u određenom izvršnom okruženju.

Ova definicija ukazuje da se svaki kompajler s određenim skupom opcija komande, zajedno sa standardnom C bibliotekom, smatra posebnom implementacijom, a različite implementacije mogu imati znatno različita ponašanja koja su definisana implementacijom. Ovo se posebno primećuje u GNU kolekciji kompajlera (GCC), koja koristi opciju `-std=` za određivanje standarda jezika. Moguće vrednosti za ovu opciju uključuju `c89`, `c90`, `c99`, `c11`, `c17` i `c23`. Podrazumevana vrednost zavisi od verzije kompajlera. Ako nijedna opcija za dijalekt jezika C nije navedena, podrazumevana vrednost za GCC 13 je `-std=gnu17`, koja obezbeđuje dodatke jeziku C. Radi prenosivosti, navodite standard koji koristite. Za pristup novim karakteristikama jezika, navedite noviji standard. Karakteristike C23 dostupne su od GCC 11. Da biste omogućili podršku za C23, dodajte opciju kompajlera `-std=c23` (ili eventualno `-std=c2x`). Svi primeri u ovoj knjizi napisani su za C23.

Zbog toga što implementacije imaju širok raspon ponašanja i zato što su neka od tih ponašanja nedefinisana, ne možete razumeti jezik C samo pisanjem jednostavnih test programa za ispitivanje ponašanja. (Ako želite da pokušate, Compiler Explorer je odličan alat; pogledajte https://godbolt.org.) Ponašanje koda može se promeniti kada se kompajlira različitom implementacijom na različitim platformama ili čak istom implementacijom uz različit skup opcija ili različitu implementaciju standardne C biblioteke. Ponašanje koda može se promeniti i između verzija kompajlera. Standard jezika C precizira koja su ponašanja zagarantovana za sve implementacije i gde treba da planirate za moguću varijabilnost. Ovo je uglavnom važno prilikom razvoja prenosivog koda, ali može takođe uticati na bezbednost i sigurnost vašeg koda.

CERT Standard za kodiranje u jeziku C

Knjiga *CERT® C Coding Standard: 98 Rules for Developing Safe, Reliable, and Secure Systems*, drugo izdanje (Addison-Wesley Professional, 2014), je referentna knjiga koju sam napisao dok sam upravljao timom za bezbedno kodiranje na Institutu za softversko inženjerstvo Univerziteta Karnegi Melon. Knjiga sadrži primere uobičajenih grešaka u programiranju u jeziku C i načine za njihovo ispravljanje. U ovoj knjizi ćemo se pozivati na neka od tih pravila kao izvore za detaljne informacije o specifičnim temama vezanim za programiranje u jeziku C.

Uobičajeni popis slabosti

Uobičajeni popis slabosti (CWE), kreacija organizacije MITRE, je lista uobičajenih slabosti hardvera i softvera koja se koristi za identifikaciju slabosti u izvornom kodu i operativnim sistemima. CWE lista se održava kao zajednički projekat, čiji su ciljevi razumevanje nedostataka u softveru i hardveru i kreiranje alata koji se mogu koristiti za identifikaciju, ispravku i prevenciju tih nedostataka. Povremeno ćemo se u ovoj knjizi pozivati na specifične stavke CWE liste prilikom rasprave o klasama nedostataka koji mogu dovesti do bezbednosnih ranjivosti. Za više informacija o CWE, pogledajte https://cwe.mitre.org.

Kome je namenjena ova knjiga

Ova knjiga je uvod u jezik C. Napisana je tako da bude što pristupačnija svima koji žele da nauče programiranje u jeziku C, ali bez pojednostavljivanja u meri koja bi umanjila razumevanje. Drugim rečima, nismo previše pojednostavili programiranje u jeziku C, kao što to čine mnoge druge uvodne knjige i kursevi. Ta preterano pojednostavljena literatura može vas naučiti kako da kompajlirate i pokrenete kod, ali taj kod može i dalje biti netačan. Programeri koji uče programiranje u jeziku C iz takvih izvora obično razvijaju nekvalitetan, pogrešan i nesiguran kod, koji će na kraju morati da se prepravi (često pre nego kasnije). Nadamo se da će ti programeri s vremenom imati koristi od starijih kolega u svojim organizacijama, koji će im pomoći da prevaziđu ove štetne zablude o programiranju u jeziku C i počnu da razvijaju profesionalni C kod.

S druge strane, ova knjiga će vas brzo naučiti kako da razvijate tačan, prenosiv, profesionalno kvalitetan kod; izgradite osnovu za razvoj sistema od ključnog značaja za bezbednost i sigurnost; a možda ćete naučiti i neke stvari koje čak ni iskusni programeri u vašoj organizaciji ne znaju.

Efikasni C: Uvod u profesionalno programiranje u jeziku C, drugo izdanje, predstavlja koncizan uvod u ključne tehnike programiranja u jeziku C, koji će vas brzo uvesti u pisanje programa, reša-

vanje problema i izgradnju funkcionalnih sistema. Primeri koda su idiomatski i jasni. Takođe ćete naučiti dobre prakse u softverskom inženjeringu za razvoj tačnog i bezbednog C koda.

U ovoj knjizi naučićete osnovne koncepte programiranja u jeziku C i vežbaćete pisanje kvalitetnog koda kroz zadatke za svaku temu. Izvori koda iz ove knjige i dodatni materijali mogu se naći na GitHub platformi na adresi https://github.com/rcseacord/effective-c. Posetite stranicu knjige na https://nostarch.com/effective-c-2nd-edition ili http://www.robertseacord.com za novosti i dodatne materijale, ili me kontaktirajte ako imate dodatna pitanja ili ste zainteresovani za obuku.

Sadržaj ove knjige

Knjiga počinje uvodnim poglavljem koje pokriva dovoljno materijala da odmah počnete sa programiranjem. Zatim se vraćamo i analiziramo osnovne gradivne blokove jezika. Knjiga kulminira sa dva poglavlja koja vam pokazuju kako da sastavite sisteme iz stvarnog sveta koristeći ove osnovne blokove i kako da otklonite greške, testirate i analizirate kod koji ste napisali. Poglavlja su sledeća:

- **Poglavlje 1: Početak rada sa jezikom C**
  Napisaćete jednostavan C program kako biste se upoznali sa korišćenjem `main` funkcije. Pogledaćete i neke opcije za uređivače i kompajlere.
- **Poglavlje 2: Objekti, funkcije i tipovi**
  Ovo poglavlje istražuje osnove kao što su deklaracija promenljivih i funkcija, kao i upotreba osnovnih tipova podataka.
- **Poglavlje 3: Aritmetički tipovi**
  Naučićete o celobrojnim i pokretnim tačkama aritmetičkim tipovima podataka sa pokretnim zarezom.
- **Poglavlje 4: Izrazi i operatori**
  Naučićete o operatorima i kako pisati jednostavne izraze za izvođenje operacija nad različitim vrstama objekata.
- **Poglavlje 5: Kontrola toka**
  Naučićete kako da kontrolišete redosled izvršavanja pojedinačnih izraza. Uvešćemo izraze i složene izraze koji definišu radnje koje treba obaviti, zatim ćemo pokriti kontrolne izraze koji određuju koje se blokovi koda izvršavaju i kojim redosledom: selekcija, iteracija i skokovi.
- **Poglavlje 6: Dinamički alocirana memorija**
  Naučićete o dinamički alociranoj memoriji, koja se alocira iz *hipa* tokom izvršavanja programa. Dinamička memorija je korisna kada tačni zahtevi za skladištenjem nisu poznati pre vremena izvršenja.
- **Poglavlje 7: Karakteri i niske**
  Ovo poglavlje pokriva različite skupove karaktera, uključujući ASCII i Unicode, koji se mogu koristiti za sastavljanje stringova. Naučićete kako su stringovi predstavljeni i manipulisani pomoću funkcija iz standardne biblioteke jezika C, interfejsa za proveru granica, kao i POSIX i Windows API.

- **Poglavlje 8: Ulaz i izlaz**
  Ovo poglavlje vas uči kako da izvodite ulazno-izlazne operacije radi čitanja podataka sa terminala i sistema datoteka, kao i pisanja podataka u njih. Pokrićemo tehnike koje koriste standardne C tokove i POSIX deskriptore datoteka.
- **Poglavlje 9: Pretprocesor**
  Naučićete kako da koristite pretprocesor za uključivanje datoteka, definisanje makroa sličnih objektima i funkcijama, i uslovno uključivanje koda na osnovu specifičnih karakteristika implementacije.
- **Poglavlje 10: Struktura programa**
  Naučićete kako da organizujete program u više prevodilačkih jedinica koje se sastoje od izvornog koda i datoteka za uključivanje. Takođe ćete naučiti kako povezati više objekata kako biste kreirali biblioteke i izvršne datoteke.
- **Poglavlje 11: Otklanjanje grešaka, testiranje i analiza**
  Ovo poglavlje opisuje alate i tehnike za izradu programa bez grešaka, uključujući tvrdnje tokom kompajliranja i izvršavanja, otklanjanje grešaka, testiranje, statičku i dinamičku analizu. Poglavlje takođe obrađuje koje se opcije kompajlera preporučuju za različite faze procesa razvoja softvera.
- **Dodatak: Peto izdanje standarda jezika C (C23)**
  Ovaj dodatak nabraja neke dodatke i promene u C23 standardu. To je praktičan način za upoznavanje s novinama u jeziku C i identifikovanje razlika u odnosu na prethodni standard (C17). Knjiga je ažurirana u odnosu na prethodno izdanje kako bi pokrila karakteristike i ponašanja standarda C23. Prema anketnim podacima iz 2022. godine kompanije JetBrains (https://www.jetbrains.com/lp/devecosystem-2022/c/), 44% programera koristi C99, 33% koristi C11, 16% koristi C17, a 15% koristi verziju C prilagođenu ugradnim sistemima.

Krećete na putovanje iz kojeg ćete izaći kao novoformirani, ali profesionalni C programer.

1

POČETAK RADA SA JEZIKOM C

U OVOM POGLAVLJU ĆETE NAPISATI SVOJ PRVI PROGRAM U JEZIKU C: TRADICIONALNI PROGRAM „ZDRAVO, SVETE!". PROĆI ĆEMO KROZ RAZLIČITE ASPEKTE OVOG JEDNOSTAVNOG PROGRAMA, ZATIM ĆEMO GA KOMPAJLIRATI I POKRENUTI. NAKON TOGA, PREDSTAVIĆU VAM NEKE OPCIJE UREĐIVAČA TEKSTA I KOMPAJLERA, KAO I UOBIČAJENE PROBLEME SA PRENOSIVOŠĆU KODA SA KOJIMA ĆETE SE BRZO SUSRESTI TOKOM RADA U JEZIKU C.

Izrada vašeg prvog programa u jeziku C

Najefikasniji način da naučite programiranje u jeziku C jeste da odmah počnete da pišete C programe, a tradicionalni program za početak je „Zdravo, svete!". Otvorite svoj omiljeni uređivač teksta i unesite program iz prikaza 1-1.

hello.c

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main() {
    puts("Zdravo, svete!");
    return EXIT_SUCCESS;
}
```

Prikaz 1-1: Program „Zdravo, svete!"

Prve dve linije koriste direktivu preprocesora `#include`, koja ubacuje sadržaj navedene datoteke na tom mestu. U ovom programu, `<stdio.h>` i `<stdlib.h>` su zaglavlja. *Zaglavlje* je izvorna datoteka koja, prema konvenciji, sadrži definicije, deklaracije funkcija i definicije konstanti koje su potrebne korisnicima te datoteke. Kao što imena datoteka sugerišu, `<stdio.h>` definiše interfejs za standardne funkcije za ulaz i izlaz (I/O) u jeziku C, dok `<stdlib.h>` deklariše različite tipove i funkcije opšte namene, kao i nekoliko makroa. Potrebno je da uključite deklaracije svih bibliotečkih funkcija koje koristite u programu. (Više o odgovarajućem korišćenju zaglavlja naučićete u poglavlju 9.)

U ovom primeru, uključujemo `<stdio.h>` da bismo imali pristup deklaraciji funkcije `puts`, koju poziva funkcija `main`. Takođe uključujemo `<stdlib.h>` da bismo imali pristup definiciji makroa `EXIT_SUCCESS`, koji se koristi u `return` iskazu.

Ova linija definiše funkciju `main`, koja se poziva pri pokretanju programa:

```
int main() {
```

Funkcija `main` predstavlja ulaznu tačku programa koji se izvršava u okruženju sistema kada se program pokrene iz komandne linije ili iz drugog programa. Jezik C predviđa dva moguća okruženja za izvršavanje: nezavisno i sistematsko. *Nezavisno* okruženje možda neće imati operativni sistem i najčešće se koristi u ugrađenim sistemima. Ova implementacija pruža minimalan skup bibliotečkih funkcija, a naziv i tip funkcije koja se poziva pri pokretanju programa zavise od same implementacije. Većina primera u ovoj knjizi pretpostavlja da je funkcija `main` jedina ulazna tačka programa.

Kao i drugi proceduralni jezici, C programi sadrže *funkcije* koje mogu prihvatiti argumente i vraćati vrednosti. Svaka funkcija predstavlja jedinicu posla koja se može koristiti više puta u programu. Funkcija `puts` se poziva iz funkcije `main` kako bi ispisala liniju `Zdravo, svete!`:

```
puts("Zdravo, svete!");
```

Funkcija `puts` je funkcija iz standardne biblioteke jezika C koja upisuje prosleđenu nisku u izlazni tok `stdout` i dodaje novi red na kraju ispisa. Tok `stdout` obično predstavlja konzolu ili terminal. `"Zdravo, svete!"` je fiksna niska koja se ponaša kao niska samo za čitanje. Ovim pozivom funkcije, tekst `Zdravo, svete!` se ispisuje na terminalu.

Kada vaš program završi sa izvršavanjem, poželežćete da ga pravilno zatvorite. Naredba `return` u funkciji `main` zatvara funkciju `main` i vraća celobrojnu vrednost u sistematsko okruženje ili skript koji je pokrenuo program:

```
return EXIT_SUCCESS;
```

Makro `EXIT_SUCCESS` je makro sličan objektu koji može biti definisan na sledeći način:

```
#define EXIT_SUCCESS 0
```

Svako pojavljivanje `EXIT_SUCCESS` zamenjuje se nulom, koja se zatim vraća u sistematsko okruženje iz poziva funkcije `main`. Skript ili proces koji pokreće program može proveriti status povratne vrednosti kako bi odredio da li je izvršavanje programa bilo uspešno. Povratak iz početnog poziva funkcije `main` ekvivalentan je pozivu funkcije `exit` iz standardne biblioteke jezika C, sa vrednošću koju funkcija `main` vraća kao argument.

Poslednja linija ovog programa sadrži zatvorenu vitičastu zagradu (}), koja zatvara blok koda koji smo otvorili deklaracijom funkcije `main`:

```
int main() {
    // -- kod --
}
```

Otvorenu zagradu možete da postavite na istu liniju gde je deklaracija funkcije ili na posebnu liniju, kao u sledećem primeru:

```
int main()
{
    // -- kod --
}
```

Ova odluka je isključivo stilske prirode, jer prazni znakovi (uključujući nove linije) generalno nemaju sintaksno značenje u jeziku C. U ovoj knjizi, obično postavljam otvorenu zagradu na liniju sa deklaracijom funkcije jer je to stilistički kompaktnije.

Za sada sačuvajte ovu datoteku kao `hello.c`. Ekstenzija datoteke `.c` označava da datoteka sadrži izvorni kod na jeziku C.

NAPOMENA:

Ako koristite elektronsku knjigu, možete kopirati i nalepiti program u uređivaču teksta. Korišćenje funkcije za kopiranje i lepljenje može smanjiti mogućnost grešaka prilikom prekucavanja koda.

Kompajliranje i izvršavanje programa

Sledeći korak je da kompajlirate i pokrenete program, što se obavlja u dva koraka. Komanda za kompajliranje programa zavisi od kompajlera koji koristite. Na Linux operativnim sistemima i drugim Unix sistemima, u komandnoj liniji unesite `cc`, a zatim naziv datoteke koju želite da kompajlirate:

```
$ cc hello.c
```

Ako ste program ispravno uneli, komanda za kompajliranje će kreirati novu datoteku pod nazivom `a.out` u istom direktorijumu gde se nalazi vaš izvorni kod.

NAPOMENA:

Na drugim operativnim sistemima, kao što su Windows ili macOS, kompajleri se pozivaju na različite načine. Proverite dokumentaciju za specifični kompajler koji koristite.

Da biste pregledali sadržaj svog direktorijuma, koristite sledeću komandu:

```
$ ls
a.out hello.c
```

Datoteka `a.out` koju vidite na izlazu je izvršni program, koji sada možete pokrenuti u komandnoj liniji:

```
$ ./a.out
Zdravo, svete!
```

Ako je sve ispravno, program bi trebalo da ispiše `Zdravo, svete!` u prozoru terminala. Ako se to ne dogodi, uporedite tekst programa iz prikaza 1-1 sa svojim programom i proverite da li su identični.

Komanda `cc` prihvata brojne opcije kompajliranja. Na primer, opcija kompajlera `-o datoteka` omogućava vam da izvršnoj datoteci dodelite prepoznatljivo ime umesto podrazumevanog `a.out`. Sledeća komanda daje ime izvršnoj datoteci `hello`:

```
$ cc -o hello hello.c
$ ./hello
Zdravo, svete!
```

U knjizi ćemo predstaviti i druge opcije kompajlera i povezivača (poznate kao zastavice), a posebno ćemo im posvetiti jedan odeljak u poglavlju 11.

Povratne vrednosti funkcija

Funkcije često vraćaju vrednost koja je rezultat određenog izračunavanja ili koja signalizira da li je funkcija uspešno završila svoj zadatak. Na primer, funkcija `puts` koju smo koristili u programu „Zdravo, svete!“ prihvata nisku za ispis i vraća vrednost tipa `int`. Funkcija `puts` vraća vrednost makroa `EOF` (negativan ceo broj) ako dođe do greške pri upisu; u suprotnom, vraća nenegativnu celobrojnu vrednost.

Iako je malo verovatno da će funkcija `puts` u našem jednostavnom programu vratiti EOF, ova situacija je moguća. S obzirom da poziv funkcije `puts` može da bude neuspešno i vrati EOF, to znači da naš prvi program u jeziku C sadrži grešku ili se, makar, može poboljšati na sledeći način:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    if (puts("Zdravo, svete") == EOF) {
        return EXIT_FAILURE;
        // ovde se kod nikada ne izvršava
    }
```

```
    return EXIT_SUCCESS;
    // ovde se kod nikada ne izvršava
}
```

Ova izmenjena verzija programa „Zdravo, svete!“ proverava da li poziv funkcije `puts` vraća vrednost `EOF`, što bi ukazivalo na grešku pri upisu. Ako funkcija `puts` vrati `EOF`, program vraća vrednost makroa `EXIT_FAILURE` (koji ima nenula vrednost). U suprotnom, funkcija se izvršava uspešno, a program vraća `EXIT_SUCCESS`. Skript koji pokreće program može proveriti status povratne vrednosti kako bi utvrdio da li je izvršenje bilo uspešno. Kod nakon `return` naredba predstavlja *mrtav kod* koji se nikada ne izvršava. Ovo je naznačeno jednolinijskim komentarom u izmenjenom programu. Sve što se nalazi nakon `//` ignoriše kompajler.

Formatizovan izlaz

Funkcija `puts` je jednostavan način za ispisivanje niske na `stdout`, ali za ispis argumenata koji nisu niske potrebno je koristiti funkciju `printf`. Funkcija `printf` prima formatiranu nisku koja definiše kako će izlaz biti oblikovan, praćen promenljivim brojem argumenata koji predstavljaju stvarne vrednosti koje želite da ispišete. Na primer, ako želite da koristite funkciju `printf` za ispisivanje Zdravo, svete!, možete to napisati na sledeći način:

```
printf("%s\n", "Zdravo, svete!");
```

Prvi argument je formatirani string `"%s\n"`. Deo `%s` je specifikacija konverzije koja govori funkciji `printf` da pročita drugi argument (fiksnu nisku) i da ga ispiše na `stdout`. `\n` je specijalna sekvenca karaktera koja predstavlja karakter za novi red i signalizira funkciji da pređe u novi red nakon ispisane niske. Bez ove sekvence za novi red, sledeći karakteri (verovatno komandna linija) bi se pojavili u istom redu. Ovaj poziv funkcije daje sledeći izlaz:

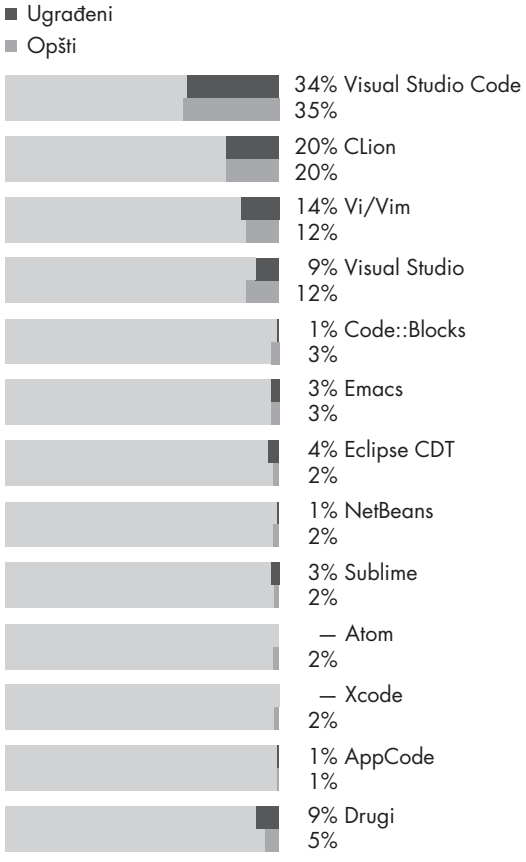
```
Zdravo, svete!
```

Važno je da ne prosleđujete podatke koje unosi korisnik kao prvi argument funkciji `printf`, jer to može dovesti do bezbednosnih ranjivosti u formatiranom izlazu (Seacord 2013).

Najjednostavniji način za ispisivanje niske je korišćenje funkcije `puts`, kao što je ranije prikazano. Međutim, ako koristite `printf` umesto `puts` u izmenjenoj verziji programa „Zdravo, svete!”, primetićete da program više neće raditi na isti način, jer funkcija `printf` vraća status drugačije nego `puts`. Funkcija `printf` vraća broj karaktera koji su uspešno ispisani ili negativnu vrednost ako je došlo do greške pri ispisu ili kodiranju. Pokušajte da izmenite program „Zdravo, svete!” tako da koristi funkciju `printf` kao vežbu.

Uređivači teksta i integrisana razvojna okruženja

Možete koristiti različite uređivače teksta i integrisana razvojna okruženja (IDE) za razvoj vaših programa u jeziku C. Na slici 1-1 prikazani su najčešće korišćeni uređivači teksta prema anketi kompanije JetBrains iz 2023. godine (https://www.jetbrains.com/lp/devecosystem-2023/c/).



Slika 1-1: Najpopularnija integraisana razvojna okruženja (IDE) i uređivači teksta

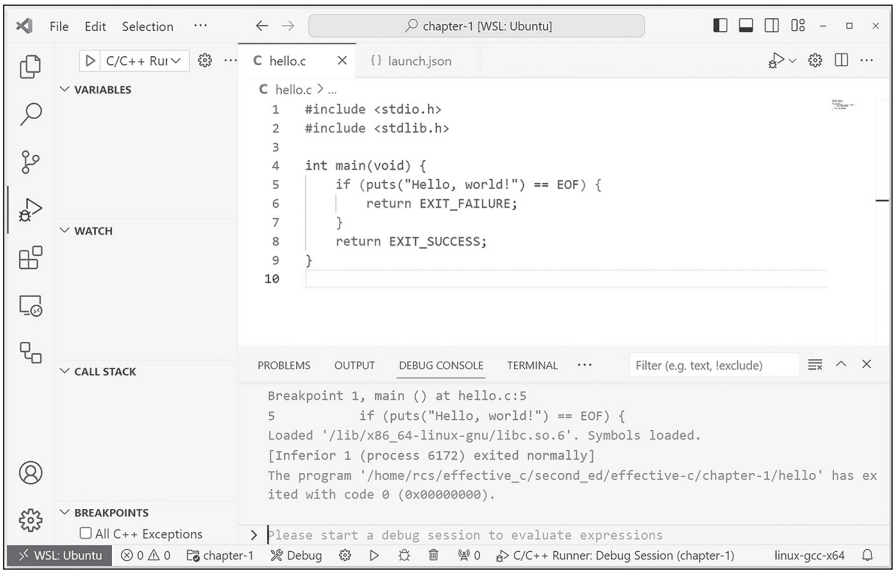
Dostupni alati zavise od operativnog sistema koji koristite.

Za Microsoft Windows, očigledan izbor je Visual Studio IDE (dostupan na https://visualstudio.microsoft.com). Visual Studio dolazi u tri izdanja: Community, Professional i Enterprise. Community izdanje ima prednost što je besplatno, dok druga dva izdanja dodaju dodatne funkcionalnosti uz naknadu. Za ovu knjigu, potrebno je samo Community izdanje.

Za Linux je izbor manje očigledan jer postoji više opcija. Popularan izbor je Visual Studio Code (VS Code). VS Code je optimizovan uređivač koda koji podržava razvojne operacije kao što su otklanjanje grešaka, pokretanje zadataka i kontrola verzije (koja je obrađena u poglavlju 11). Pruža sve potrebne alate za brz ciklus pisanja, kompajliranja i otklanjanja grešaka u kodu. VS Code radi na operativnim sistemima macOS, Linux i Windows, i besplatan je za privatnu ili

komercijalnu upotrebu. Uputstva za instalaciju dostupna su za Linux i druge platforme (https://code.visualstudio.com).

Slika 1-2 prikazuje VS Code koji se koristi za pisanje programa „Zdravo, svete!" na Ubuntu sistemu. U konzoli za otklanjanje grešaka prikazano je da je program završio sa status kodom `0`, što je očekivani ishod.



Slika 1-2: Visual Studio Code pokrenut na Ubuntu sistemu

Vim je izbor uređivača teksta za mnoge programere i napredne korisnike. Ovo je uređivač teksta zasnovan na `vi` editoru koji je sedamdesetih godina razvio Bil Džoj za Unix verziju. Vim nasleđuje komande `vi` uređivača teksta, ali dodaje funkcionalnosti i mogućnosti proširenja koje nedostaju u originalnom `vi` uređivaču teksta. Po želji, možete instalirati Vim dodatke kao što su YouCompleteMe (https://github.com/ycm-core/YouCompleteMe) ili deoplete (https://github.com/Shougo/deoplete.nvim), koji omogućavaju smisleno sintaksno dopunjavanje koda za programski jezik C.

GNU Emacs je proširiv, prilagodljiv i besplatan uređivač teksta. U svojoj osnovi, on je interpretator za Emacs Lisp, dijalekt programskog jezika Lisp sa proširenjima za uređivanje teksta—iako ovo nikada nisam doživljavao kao prepreku. Za potpunu transparentnost: skoro sav proizvodni kod u jeziku C koji sam pisao bio je uređivan u Emacs uređivaču teksta.

Kompajleri

Dostupno je mnogo kompajlera za jezik C, pa ih ovde neću sve obrađivati. Različiti kompajleri implementiraju različite verzije standarda jezika C. Mnogi kompajleri za ugrađene sisteme podržavaju samo C89/C90 standard. Popularni kompajleri za Linux i Windows, međutim, teže ka tome da podrže savremene verzije standarda jezika C, uključujući i podršku za C23.

GNU familija kompajlera

GNU familija kompajlera (GCC) uključuje klijentske kompajlere za jezike C, C++ i Objective-C, kao i za druge jezike (https://gcc.gnu.org). GCC prati jasno definisan plan razvoja pod vođstvom upravnog odbora za GCC.

GCC je postao standardni kompajler za Linux sisteme, iako su dostupne verzije i za Microsoft Windows, macOS i druge platforme. Instalacija GCC programa na Linux operativnom sistemu je jednostavna. Na primer, sledeća komanda će instalirati GCC na Ubuntu:

```
$ sudo apt-get install gcc
```

Verziju GCC programa koji koristite možete proveriti sledećom komandom:

```
$ gcc --version
```

Izlaz će prikazati verziju i informacije o autorskim pravima za instaliranu verziju GCC programa.

Clang

Još jedan popularan kompajler je Clang (https://clang.llvm.org). Instalacija Clang kompajlera na Linux operativnom sistemu je takođe jednostavna. Na Ubuntu sistemu možete koristiti sledeću komandu za instalaciju Clang kompajlera:

```
$ sudo apt-get install clang
```

Verziju Clang kompajlera možete proveriti sledećom komandom:

```
$ clang --version
```

Ova komanda prikazuje verziju instaliranog Clang kompajlera.

Microsoft Visual Studio

Kao što je ranije pomenuto, najpopularnije razvojno okruženje za Windows je Microsoft Visual Studio, koje uključuje i integrisano razvojno okruženje i kompajler. Visual Studio (https://visualstudio.microsoft.com/downloads/) dolazi sa Visual C++ 2022, koji uključuje kompajlere za jezike C i C++.

Opcije za Visual Studio možete postaviti na stranicama Project Property. Na kartici Advanced pod C/C++, uverite se da kompajlirate kao C kod pomoću opcije Compile as C Code (/TC), a ne kao C++ kod sa opcijom Compile as C++ Code (/TP). Podrazumevano, kada datoteku ima ekstenziju `.c`, ona se kompajlira sa opcijom `/TC`. Ako datoteka ima ekstenziju `.cpp`, `.cxx` ili neke druge slične ekstenzije, kompajliraće se sa opcijom `/TP`.

Prenosivost

Svaka implementacija kompajlera za jezik C ima svoje specifičnosti. Kompajleri se stalno razvijaju—na primer, kompajler kao što je GCC može u potpunosti podržavati standard C17, ali biti u procesu dodavanja podrške za C23. U tom slučaju, možda će imati implementirane neke funkcionalnosti C23, ali ne i sve. Zbog toga kompajleri podržavaju širok spektar verzija standarda jezika C (uključujući i međuverzije). Evolucija implementacija jezika C je generalno spora, a mnogi kompajleri značajno zaostaju za najnovijim verzijama C standarda.

Programi napisani u jeziku C mogu se smatrati *strogo usklađenima* ako koriste samo one karakteristike jezika i biblioteke koje su navedene u standardu. Ovi programi su zamišljeni tako da budu maksimalno prenosivi. Međutim, zbog različitih ponašanja implementacija, nijedan stvarni C program nije u potpunosti u skladu sa standardom, niti će ikada biti (a verovatno ni ne bi trebalo). Umesto toga, C standard vam omogućava da pišete programe *koji su u skladu* sa standardom, ali koji mogu zavisiti od specifičnih karakteristika jezika i biblioteke koje nisu prenosive.

Uobičajeno je da se kod piše za jednu referentnu implementaciju kompajlera, ili ponekad za više njih, u zavisnosti od platformi na koje planirate da primenite kod. C standard osigurava da se ove implementacije ne razlikuju previše, omogućavajući vam da ciljate više platformi bez potrebe da svaki put učite novi jezik.

U Dodatku J standarda za jezik C navedeno je pet vrsta problema sa prenosivošću:

- Ponašanje definisano implementacijom
- Neodređeno ponašanje
- Nedefinisano ponašanje
- Ponašanje specifično za lokalna podešavanja
- Uobičajena proširenja

Tokom učenja jezika C, susretaćete se sa primerima svih pet vrsta ponašanja, tako da je važno da precizno razumete šta oni predstavljaju.

Ponašanje definisano implementacijom

Ponašanje definisano implementacijom je ponašanje programa koje nije precizirano standardom jezika C i može dati različite rezultate između različitih implementacija, ali ima dosledno, dokumentovano ponašanje unutar jedne implementacije. Primer ponašanja definisanog implementacijom je broj bitova u jednom bajtu.

Ponašanja definisana implementacijom uglavnom su bezopasna, ali mogu izazvati probleme prilikom prenošenja koda na različite implementacije. Kad god je to moguće, izbegavajte pisanje koda koji zavisi od ponašanja definisanih implementacijom koja se razlikuju među C implementacijama sa kojima planirate da kompajlirate kod. Potpuna lista ponašanja definisanih implementacijom navedena je u Dodatku J.3 standarda jezika C. Možete dokumentovati svoje zavisnosti od ovih ponašanja koristeći naredbu `static_assert`, o kojoj će biti reči u poglavlju 11.

Neodređeno ponašanje

Neodređeno ponašanje je ponašanje programa za koje standard pruža dve ili više opcija, ali ne precizira koja opcija će biti izabrana u konkretnom slučaju. Svako izvršavanje određenog izraza može dati različite rezultate ili proizvesti različitu vrednost u poređenju sa prethodnim izvršavanjem istog izraza. Primer neodređenog ponašanja je način raspoređivanja parametara funkcije u memoriji, koji može varirati među pozivima funkcije unutar istog programa. Neodređena ponašanja navedena su u Dodatku J.1 standarda jezika C.

Nedefinisano ponašanje

Nedefinisano ponašanje je ponašanje koje nije definisano standardom jezika C ili, rečeno preciznije, predstavlja „ponašanje pri korišćenju neprenosivog ili pogrešnog konstrukta programa ili pogrešnih podataka za koje standard ne postavlja zahteve" (ISO/IEC 9899:2024). Primeri nedefinisanog ponašanja uključuju prelivanje pri operacijama sa označenim celobrojnim vrednostima i dereferenciranje neispravne vrednosti pokazivača. Kod koji ima nedefinisano ponašanje često je neispravan, ali ne uvek. Nedefinisana ponašanja su identifikovana u standardu na sledeći način:

- Kada je narušen zahtev „mora" ili „ne sme", a taj zahtev nije deo ograničenja, ponašanje je nedefinisano.
- Kada je ponašanje izričito opisano rečima „nedefinisano ponašanje."
- Odsustvom bilo kakvog eksplicitnog opisa ponašanja.

Prve dve vrste nedefinisanog ponašanja često se nazivaju *eksplicitnim nedefinisanim ponašanjem*, dok se treća vrsta naziva *implicitnim nedefinisanim ponašanjem*. Između njih nema razlike u naglasku; svi opisuju ponašanje koje je nedefinisano. Dodatak J.2 standarda jezika C, „Nedefinisano ponašanje", navodi eksplicitna nedefinisana ponašanja u jeziku C.

Programeri često pogrešno tumače nedefinisano ponašanje kao greške ili propuste u standardu jezika C, ali je odluka da se ponašanje klasifikuje kao nedefinisano *namerna* i *promišljena*. Odbor za standardizaciju jezika C klasifikuje ponašanja kao nedefinisana iz jednog od sledećih razloga:

- Daje implementatorima slobodu da ne detektuju greške u programu koje je teško dijagnostikovati.
- Izbegava definisanje komplikovanih slučajeva koji bi pogodovali jednoj strategiji implementacije u odnosu na drugu.
- Identifikuje oblasti mogućih proširenja jezika koja su u skladu sa standardom, gde implementator može proširiti jezik definisanjem zvanično nedefinisanog ponašanja.

Ova tri razloga su prilično različita, ali se svi smatraju pitanjima prenosivosti. U knjizi ćemo se susretati sa primerima svih ovih aspekata. Prilikom susreta sa nedefinisanim ponašanjem, kompajleri imaju slobodu da postupaju na sledeći način:

- Potpuno ignorisanje nedefinisanog ponašanja, što može dovesti do nepredvidivih rezultata
- Ponašanje na dokumentovan način karakterističan za okruženje (može, ali ne mora, prikazati dijagnostičke poruke)
- Prekid prevođenja ili izvršavanja programa (uz prikaz dijagnostičke poruke)

Nijedna od ovih opcija nije idealna (posebno prva), pa je najbolje izbegavati nedefinisana ponašanja, osim u slučajevima kada kompajler specificira da su ta ponašanja definisana kako bi omogućila određeno proširenje jezika. Kompajleri ponekad nude *pedantni* način rada koji može pomoći programeru da prepozna ovakve probleme sa prenosivošću.

Ponašanje specifično za lokalna podešavanja i uobičajena proširenja

Ponašanje specifično za lokalna podešavanje zavisi od lokalnih konvencija kao što su nacionalnost, kultura i jezik, što svaka implementacija dokumentuje. *Uobičajena proširenja* se često koriste u mnogim sistemima, ali nisu prenosiva na sve implementacije.

Rezime

U ovom poglavlju naučili ste kako da napišete jednostavan program u jeziku C, kompajlirate ga i pokrenete. Razmotrili smo različite uređivače teksta i integrisana razvojna okruženja, kao i nekoliko kompajlera koje možete koristiti za razvoj C programa na Windows, Linux i macOS sistemima. Preporučuje se korišćenje novijih verzija kompajlera i drugih alata jer oni uglavnom podržavaju novije funkcije jezika C i pružaju bolje dijagnostičke informacije i optimizacije. Međutim, možda nećete želeti da koristite najnovije verzije kompajlera ako one mogu da naruše vaš postojeći kod ili ako se spremate da ga objavite, kako biste izbegli nepotrebne promene u već testiranoj aplikaciji. Poglavlje smo završili diskusijom o prenosivosti programa u jeziku C.

U narednim poglavljima ćemo detaljno razmotriti specifične karakteristike jezika C i njegove biblioteke, počevši od objekata, funkcija i tipova u sledećem poglavlju.