# B. Korištenje prevoditelja

*„Go ahead, make my day."*
*(„Učini to i uljepšaj mi dan.")*

*Clint Eastwood u filmu „Prljavi Harry"*

U ovom prilogu ukratko će biti opisana upotreba dvije često korištene razvojne okoline:

- *Code::Blocks*,
- Microsoft *Visual C++*, odnosno *Visual Studio*.

*Visual C++/Studio* je komercijalni proizvod no dostupan je i u besplatnoj inačici *Visual C++ Express* koja je „olakšana" verzija komercijalne razvojne okoline, ali pruža sve što je potrebno za učenje jezika C++.

Upute koje slijede su napisane tek da posluže za „prvu ruku" – za dodatne i detaljnije upute svakako proučite dokumentaciju koja dolazi uz alate. Podrazumijevane postavke su zadovoljavajuće za primjere iz ove knjige, tako da ih većina korisnika (barem u početku) neće morati mijenjati.

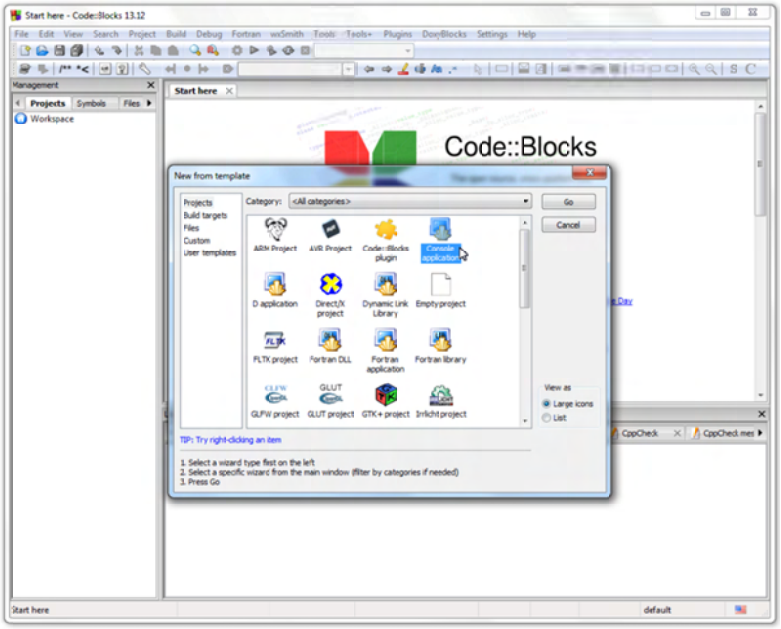
## B.1. *Code::Blocks*

*Code::Blocks* (*http://www.codeblocks.org/*) je besplatna razvojna okolina koja koristi *gcc* prevoditelj, ali se može vrlo jednostavno prilagoditi i za neki drugi komandni prevoditelj. U rujnu 2014. godine, zadnja aktualna verzija je bila 13.12 (iz prosinca 2013. godine). Dostupne su verzije za Windows, Linux i MacOS.

Za instalaciju trebate na gornjoj adresi potražiti binarnu datoteku s instalacijom. Početnicima preporučamo instalaciju koja u sebi već ima uključen *GCC* (*GNU Compiler Collection*) prevoditelj za C++ te *GDB (GNU) debugger*. Taj prevoditelj je u potpunosti usklađena sa Standardom C++11, odnosno C++14. Naknadno možete ažurirati prevoditelja ili ga čak i promijeniti. *Code::Blocks* se može jednostavno integrirati s čitavim nizom prevoditelja, uključujući i Microsoftov Visual C++ te Clang.

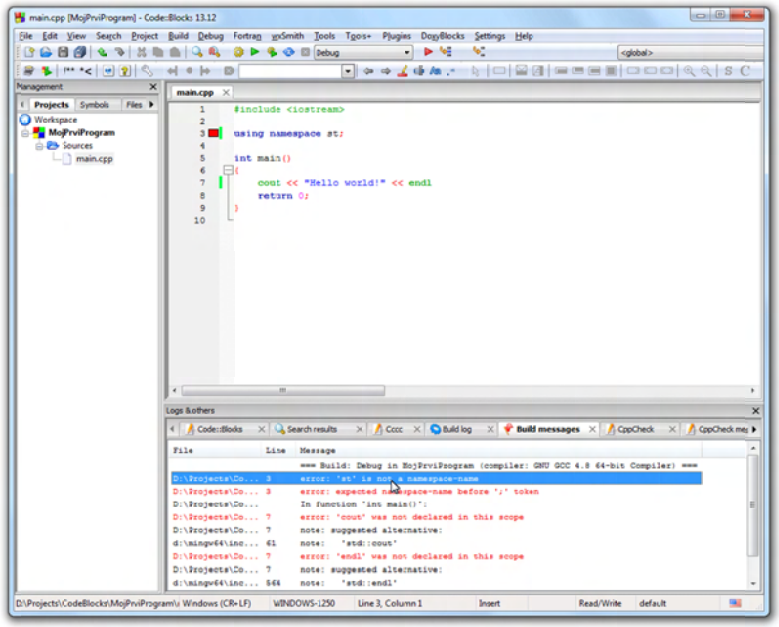
Evo sažetih uputa kako napraviti program u *Code::Blocks* razvojnoj okolini:

1. Iz izbornika *File* odabere se *New – Project*. Izabere se *Console Application* (svi primjeri kôda iz knjige će raditi isključivo u programima napravljenim kao konzolne aplikacije!) te pritisnete tipku *Go*. (slika B.1)
2. Nakon uvodnog dijaloga, u drugom dijalogu potvrdite odabir C++, a u trećem dijalogu upišete ime projekta, npr. *MojPrviProgram* te eventualno promijenite lokaciju gdje želite da vam datoteke budu smještene.




**Slika B.1.** Kreiranje novog projekta u *Code::Blocks*

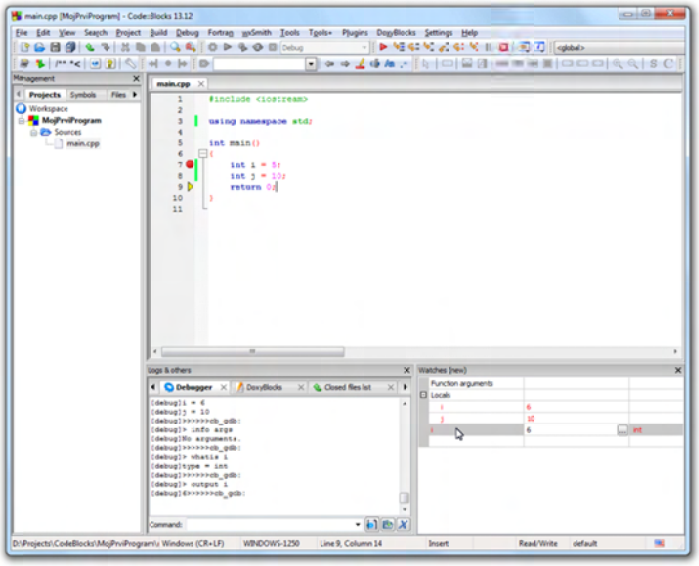
3. Nakon odabira kazala i potvrde u sljedećem dijalogu, razvojna okolina će automatski kreirati datoteku *main.cpp* s kosturom budućeg programa: neophodnim uključivanjima te funkcijom `main`. Taj kostur ćemo popuniti našim kôdom. Da biste otvorili tu datoteku i u nju mogli pisati svoj kôd trebate u lijevom prozoru *Projects*, proširiti granu *Sources* te mišem dvokliknuti na ime datoteke.
4. Budući da je *Code::Blocks* sam kreirao tijelo funkcije main, možete odmah pokrenuti program tako da u izborniku *Build* izaberete *Build and Run*.
5. U slučaju pogreške prilikom prevođenja, u *Build log* prozoru pri dnu glavnog prozora će se ispisati pogreške i upozorenja (slika B.2). Dvostrukim klikom na poruke, kurzor će se prebaciti na mjesto u kôdu gdje je pogreška uočena, a linija kôda s pogreškom će biti jasno označena crvenim kvadratićem. Valja svakako paziti da uzrok pogreške ne mora biti u toj liniji kôda. Na primjer, ako naredbu zaboravimo zaključiti znakom ; prevoditelj može javiti pogrešku za sljedeću liniju kôda. Nakon što pogreške ispravimo, kôd moramo ponovno prevesti.
6. Ako je prevođenje uspješno obavljeno, možemo pokrenuti program preko izbornika *Build – Run*. Otvorit će se komandni prozor u kojem će se ispisivati poruke, odnosno u koji ćemo upisivati podatke koje program traži.
7. Za pronalaženje pogrešaka u programu služe nam komande u *Debug* izborniku. Želimo li u nekoj naredbi zaustaviti izvođenje programa da bismo provjerili vrijednosti varijabli, možemo postaviti točku prekida (engl. *breakpoint*): postavimo kur-




**Slika B.2.** Prikaz pogreške u *Code::Blocks*

zor na željenu liniju kôda te preko *Debug – Toggle Breakpoint* aktiviramo (ili naknadno deaktiviramo) točku prekida. Ispred linije koda će se pojaviti crvena točka kao indikacija da je postavljena točka prekida. Nakon pokretanja programa s *Debug-Start/Continue*, izvođenje će se zaustaviti na prvoj točki prekida na koju program naiđe.

8. Kada se izvođenje programa zaustavi u točki prekida, u *Watches* prozoru pri dnu prozora (može se aktivirati pomoću izbornika *Debug-Debugging Windows-Watches*) automatski će se pokazati sve lokalne varijable sa svojim vrijednostima. Želimo li pogledati vrijednost neke varijable ili objekta koja nije prikazana, možemo ju upisati u prazno polje u prozoru *Watches* (slika B.3). Štoviše, vrijednost možemo i promijeniti tako da kliknemo mišem na polje s vrijednošću i upišemo željenu vrijednost.
9. Nakon što smo zaustavili izvođenje programa u točki prekida, možemo nastaviti izvođenje naredbu po naredbu pomoću komande *Debug-Next Line*, možemo ući u funkciju koja se poziva naredbom pomoću komande *Debug-Step Into* te možemo nastaviti normalno izvođenje programa naredbom *Debug-Start/Continue*.



**Slika B.3.** Pregled varijable u točki prekida u *Code::Blocks*

## B.2. Microsoft *Visual C++*

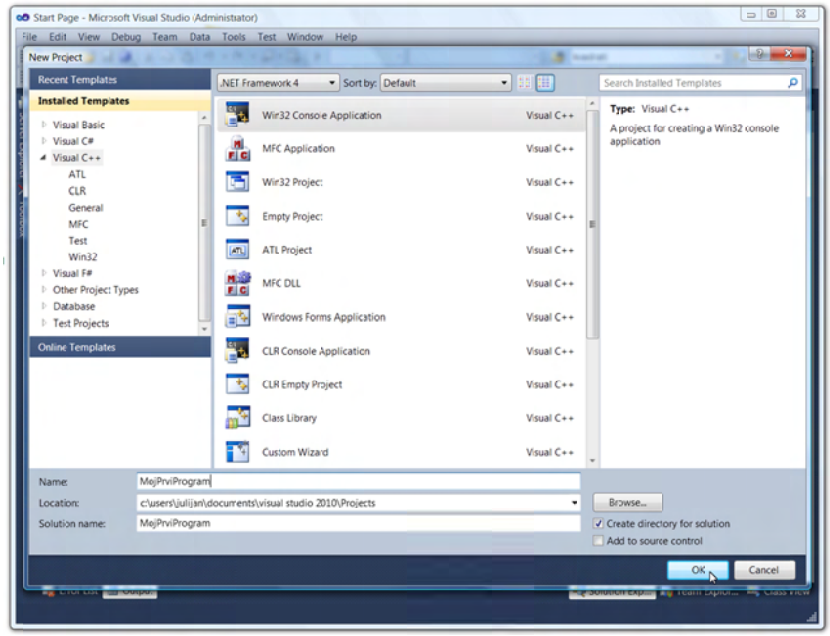
Besplatna inačica *Visual C++* razvojne okoline može se naći na adresi *www.microsoft.com/express/Downloads*. U vrijeme pisanja ovog teksta, aktualna je bila verzija Visual C++ 2013, s time da je za isprobavanje programa u jeziku C++ potrebna verzija *Express 2013 for Windows Desktop*.

Evo sažetih uputa kako napraviti program u *Visual C++* razvojnoj okolini:

1. Iz izbornika *File* odabere se *New – Project*. Izabere se *Win32 Console Application* (svi primjeri kôda iz knjige će raditi isključivo u programima napravljenim kao konzolne aplikacije!) te se upiše ime projekta, npr. *MojPrviProgram* (slika B.4).
2. Nakon pritiska na tipku *OK*, razvojna okolina će sama kreirati datoteku *MojPrviProgram.cpp* s funkcijom main. U tu datoteku ćemo pisati kôd primjera. Primijetite kako main funkcija ima nešto drugačiji oblik nego što je propisano standardom. Razlog za to su makro-pretvorbe koje pretprocesor radi – to vas ne treba smetati. Želimo li provjeriti ispis u komandnom prozoru tijekom izvođenja programa, na kraj funkcije `main`, prije naredbe `return` možemo dodati naredbu:

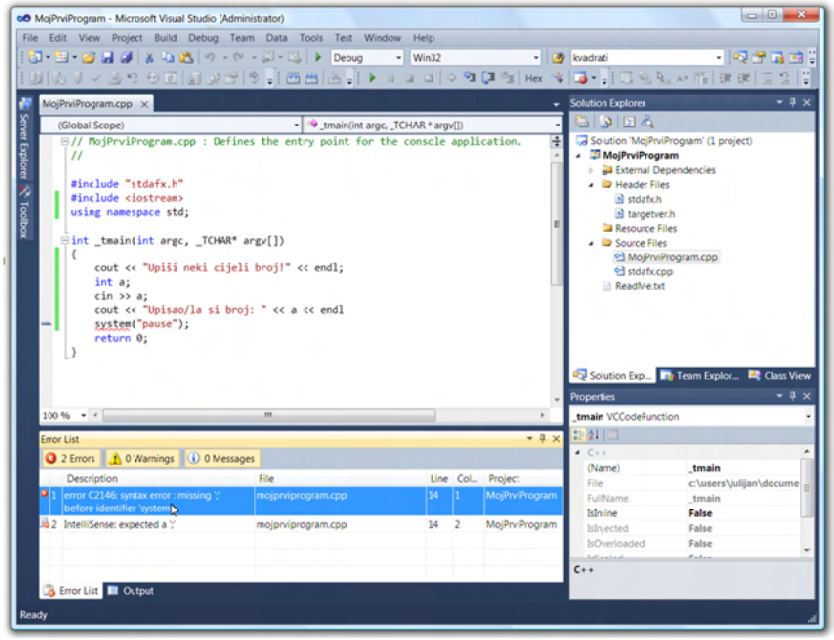
```
system("PAUSE");
```

koja će zaustaviti izvođenje programa prije nego se komandni prozor zatvori.

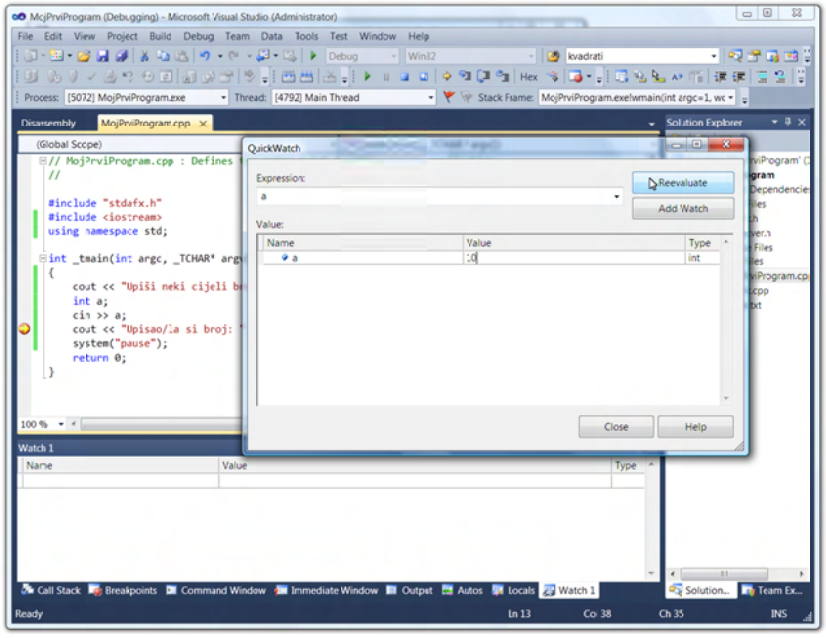



**Slika B.4.** Kreiranje novog projekta u *Visual C++*

3. Nakon što završimo s pisanjem kôda, moramo ga prevesti: izbornik *Build – Build Solution.*
4. U slučaju pogreške prilikom prevođenja, u *Error List* prozoru pri dnu glavnog prozora će se ispisati pogreške i upozorenja (slika B.5). Dvostrukim klikom na poruke, kurzor će se prebaciti na mjesto u kôdu gdje je pogreška uočena. Valja svakako paziti da uzrok pogreške ne mora biti u toj liniji kôda. Na primjer, ako naredbu zaboravimo zaključiti znakom ; prevoditelj će javiti pogrešku za sljedeću liniju kôda. Nakon što pogreške ispravimo, kôd moramo ponovno prevesti.
5. Ako je prevođenje uspješno obavljeno, možemo pokrenuti program preko izbornika *Debug – Start Debugging*. Otvorit će se komandni prozor u kojem će se ispisivati poruke, odnosno u koji ćemo upisivati podatke koje program traži.
6. Za pronalaženje pogrešaka u programu služe nam komande u *Debug* izborniku. Želimo li u nekoj naredbi zaustaviti izvođenje programa da bismo provjerili vrijednosti varijabli, možemo postaviti točku prekida (engl. *breakpoint*): postavimo kurzor na željenu liniju kôda te preko *Debug – Toggle Breakpoint* aktiviramo (ili naknadno deaktiviramo) točku prekida. Nakon pokretanja programa s *Debug – Start Debugging*, izvođenje će se zaustaviti na prvoj točki prekida na koju program naiđe.




**Slika B.5.** Prikaz pogreške u *Visual C++*




**Slika B.6.** Pregled vrijednosti objekta u *Visual C++*

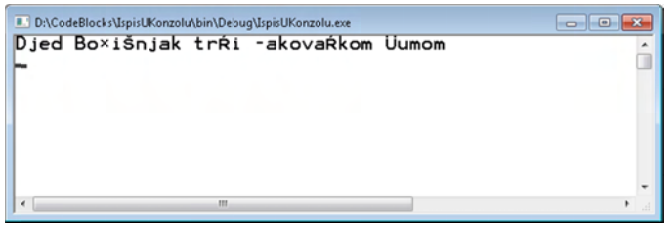
7. Želimo li pogledati vrijednost neke varijable ili objekta u točki prekida, pomoću *Debug – Quick Watch* možemo provjeriti vrijednost neke varijable (slika B.6). Štoviše, vrijednost možemo u dijalogu za prikaz i promijeniti.
8. Nakon što smo zaustavili izvođenje programa u točki prekida, možemo nastaviti izvođenje naredbu po naredbu pomoću komande *Step Over*, možemo ući u funkciju koja se poziva naredbom pomoću komande *Step Into* te možemo nastaviti normalno izvođenje programa naredbom *Continue*.

## B.3. Ispis naših znakova u konzoli

Ako u porukama koje ispisujete koristite naše znakove (č, ć, š, ž), ti će znakovi u komandnom prozoru ma Windowsima biti ispisani drugačije. Tako će naredba:

```
cout << "Djed Božićnjak trči đakovačkom šumom." << endl;
```

dati ovakav ispis:




što će sasvim sigurno mnogima ići na živce. Slično vrijedi prilikom učitavanja teksta upisanog u komandni prozor: te podatke će debugger prikazati drugačijim znakovima (iako je naknadni ispis nepromijenjenog unesenog teksta identičan upisanom tekstu).

Uzrok problema leži u činjenici da za ispis u konzolu Windowsi koriste kodnu stranicu različitu od kodne stranice u samim Windowsima. Kodna stranica u konzoli na Windowsima s hrvatskim lokalnim postavkama je obično 852[76], dok sami Windowsi (uključujući i prevoditelje na njima) koriste kodnu stranicu 1250.

Najjednostavnije rješenje iziskuje dva koraka. Kao prvo, na početak svakog programa treba dodati naredbe kojima se mijenja kodna stranica u konzoli:

```
#include <windows.h>    // SetConsoleCP() i SetConsoleOutputCP()

int main()
{
    SetConsoleCP(1250);         // za ispravan unos
    SetConsoleOutputCP(1250);   // za ispravan ispis
    //...
}
```

[76] 852 je 8-bitna kodna stranica korištena u DOS-u.

Potom treba osigurati da umjesto podrazumijevanog rasterskog pisma (*fonta*), komandni prozor koristi pismo koje podržava kodnu stranicu 1250, npr. *Lucida Console* ili *Console*. Desnim klikom na zaglavlje komandnog prozora otvori se kontekstni izbornik iz kojeg se odabere *Defaults* ili *Properties*, te se u pripadajućem dijalogu odabere novo pismo: