Kolegij Računalstvo | Datum:
Vježba # 10 | Tema: Visual Studio
Vježbu pripremio: doc. dr. sc. Edouard Ivanjko, dipl. ing. Dino Čakija

Upute za izradu laboratorijske vježbe/zadataka

Laboratorijsku vježbu potrebno je izraditi na računalu korištenjem razvojne okoline MS Visual Studio. Prije dolaska na vježbu potrebno je istu proučiti i izraditi pripremu. Bez proučene vježbe i izrađene pripreme nije moguće pristupiti izradi same vježbe.

Prilikom završetka izrade vježbe, nakon što je dežurni asistent sve pregledao:

1. Izraditi sve zadatke te rezultate pokazati dežurnom asistentu. Izrađena laboratorijska vježba koja ne sadrži rezultate unesene u sustav Merlin NEĆE biti priznata kao odrađena!
2. Sve napravljene projekte u razvojnoj okolini MS Visual Studio potrebno je arhivirati s nazivom u slijedećem obliku: *JMBAG_GRUPA_GGGG-MM-DD_VJEZBA#.zip* te unijeti u sustav Merlin.

   Primjer ispravnog naziva ZIP datoteke: 0135123456_A_2011-10-25_VJEZBA01.zip

Vrijeme za izradu laboratorijske vježbe iznosi 90 minuta.

Cilj vježbe: Upoznavanje sa razvojnim okruženjem MS Visual Studio. Kreiranje novog projekta, unos izvornog kôda, prevođenje kôda, kreiranje izvršne datoteke te njezino pokretanje. Postavljanje točki prekida te izvršavanje programa naredbu po naredbu.

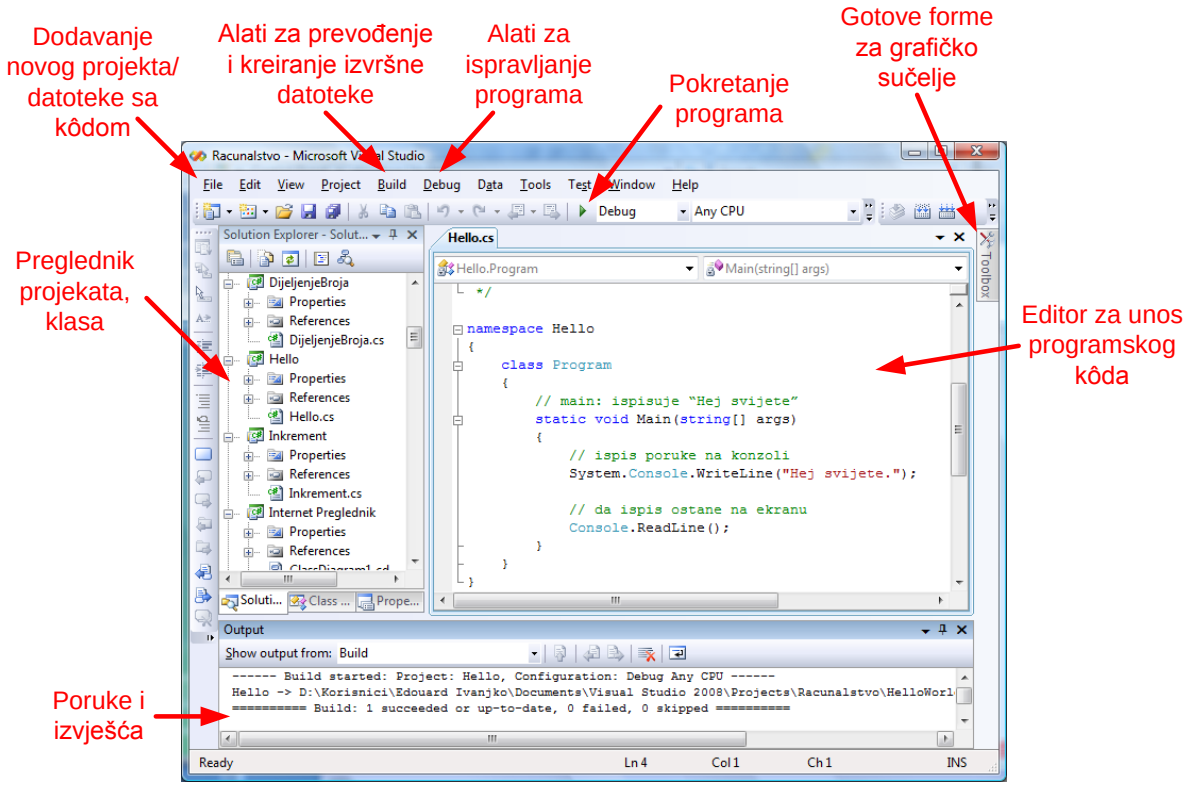
Opis vježbe

Razvojna okolina MS Visual Studio namijenjena je razvoju aplikacija u različitim višim programskim jezicima uključujući i C# (izgovara se „si sharp“). Sastoji se editora za unos programskog kôda i alata za prevođenje, pokretanje te ispravljanje izrađenog programa. Pripadni dijelovi sučelja prikazani su na slici 10.1. Izgled i pozicija pojedinih dijelova sučelja može se mijenjati sukladno potrebama korisnika.

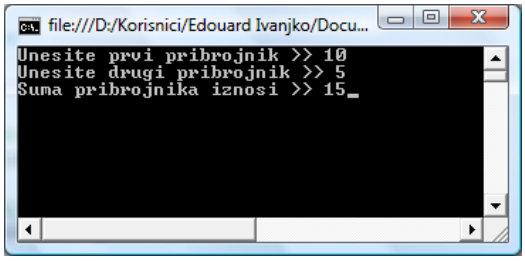
Ista razvojna okolina koristi se za izradu aplikacija sa tekstualnim sučeljem (primjer eng. Console Applications) te za aplikacije sa grafičkim sučeljem (primjer eng. Windows Form Applications). U ovoj vježbi potrebno je izraditi aplikaciju sa tekstualnim sučeljem odnosno aplikaciju koja se izvodi unutar konzole. Takvo sučelje omogućuje ispis tekstualnih poruka te unos tekstualnih podataka. Tekstualni podaci uključuju i numeričke podatke obzirom računalo svaki alfanumerički znak smatra tekstom odnosno nizom znakova. Primjer izvršavanja aplikacije sa tekstualnim sučeljem za zbrajanje dva broja prikazan je na slici 10.2.

Razvojna okolina MS Visual Studio sadrži već gotove predloške projekata za izradu raznih vrsta aplikacija uključujući aplikacije sa tekstualnim sučeljem. U slučaju projekta sa tekstualnim sučeljem potrebno je odabrati predložak „Console Application“. Njegovim odabirom automatski se podesi imenički prostor, klasa za željeni program te glavna metoda koju operacijski sustav poziva prilikom

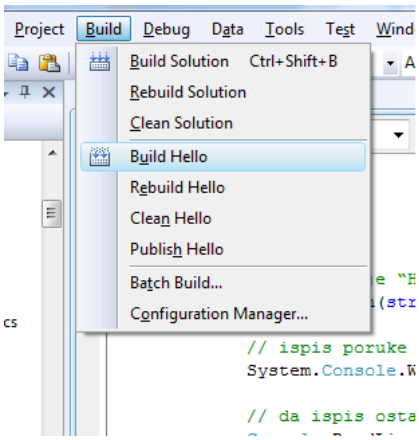
pokretanja programa. Unutar glavne metode pripremljen je prostor omeđen vitičastim zagradama za unos programskog kôda. Automatski se kreira i datoteka sa programskim kôdom jednakog imena kao i novo kreirani projekt. Navedena datoteka može se prepoznati po nastavku „.cs“ i moguće ju je otvoriti dvostrukim klikom lijeve tipke miša na pripadni grafički objekt unutar preglednika projekata (lijevi dio na slici 10.1, eng. Solution Explorer). Nakon što se unese programski kôd potrebno ga je prevesti i kreirati izvršnu datoteku. U tu svrhu koriste se alati unutar opcije izbornika „Build“ kako je prikazano na slici 10.3. Ime projekta automatski se pridodaje uz pojedine alate, a za prevođenje kôda i kreiranje izvršne datoteke odabire se opcija „Build ime“.



Slika 10.1 Sastavni dijelovi radnog prozora razvojne okoline MS Visual Studio

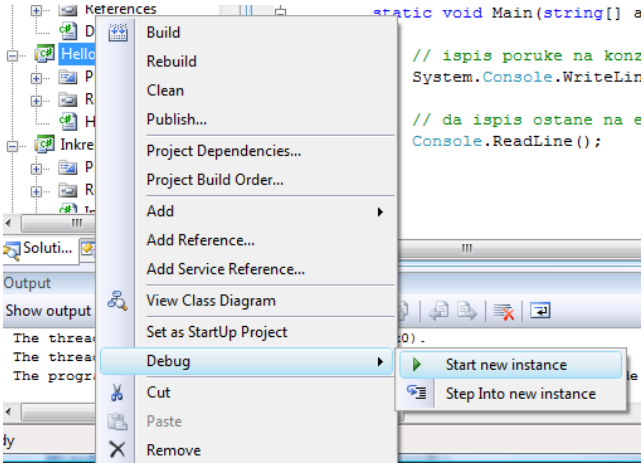


Slika 10.2 Primjer izvršavanja aplikacije sa tekstualnim sučeljem



Slika 10.3 Pozivanje alata za prevođenje kôda i kreiranje izvršne datoteke

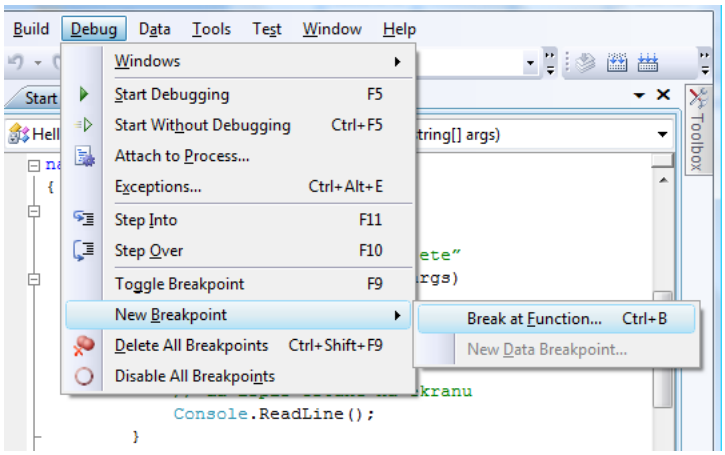
U slučaju postojanja pogrešaka unutar programskog kôda, razvojna okolina automatski podcrtava neispravne dijelove kôda. Nakon prevođenje u dijelu prozora za poruke i izvješća pojavi se popis problema. Postoje dva tipa problema: upozorenje (eng. „warning“) i greška (eng. „error“). Upozorenje je moguće zanemariti, a greške je potrebno ispraviti. Uz postojanje grešaka nije moguće kreirati izvršnu datoteku te je u tom slučaju potrebno upotrijebiti alate za otklanjanje grešaka (slika 10.5). Nakon uspješnog kreiranja izvršne datoteke, istu je moguće pokrenuti klikom lijevom tipkom miša na zelenu strelicu na traci sa alatima prikazanoj na slici 10.1 ili preko skočnog izbornika prikazanog na slici 10.4. Pripadni skočni izbornik otvara se desnim klikom miša na grafički objekt projekta unutar preglednika projekata.



Slika 10.4 Pokretanje aplikacije

Nakon što je aplikacija pokrenuta pristupa se testiranju ispravnosti rada. Faza testiranja obično se vrši upotrebom probnih podataka i provjerom rezultata. U slučaju neispravnog rezultata moguće je kao pomoć iskoristiti alate za otklanjanje grešaka (slika 10.5). Oni su aktivni tijekom izvođenja aplikacije, a omogućuju izvođenje iste naredbu po naredbu uz istodoban pristup vrijednostima pojedinih varijabli. U svrhu bolje kontrole procesa otklanjanja grešaka aplikacije moguće je koristiti tzv. prekidne točke (engl. „breakpoints“). Značajka im je da se aplikacija koja se ispravlja normalno izvodi do točke prekida, nakon čega se od programera očekuje da nastavi izvođenje aplikacije ručnom kontrolom redak po redak (ovisno o broju i lokacijama prekidnih točaka). Istovremeno su programeru dostupne sve vrijednosti pojedinih varijabli do linije programskog kôda u kojoj se nalazi točka prekida. Daljnje izvođenje aplikacije moguće je ručno aktivirati odabirom opcije „Step Into“. Tada se izvođenje aplikacije nastavlja do sljedeće točke prekida ili naredbe koja zahtijeva unos podataka od strane korisnika. Točka prekida postavlja se u kritičnu liniju kôda na način da se prvo pokazivač pozicionira u

odgovarajuću liniju kôda i odabere opcija „Debug -> Toggle Breakpoint“ ili klikne lijevom tipkom miša na sivu traku na lijevoj strani editora.



Slika 10.5 Alati za otklanjanje grešaka

Priprema za vježbu

- Proučiti predavanja vezana za dijagrame toka.
- Proučiti predavanje 9 vezano za uvod u C# programiranje.
- Instalirati besplatnu inačicu MS Visual Studio Express C# na vlastito osobno računalo te testirati instalaciju izradom osnovne aplikacije „Hej svijete!“ prema uputama u nastavku.
- Popuniti sljedeću tablicu općenitim dijelovima programa objašnjenim na primjeru programa „Zbrajanje“ u predavanju 9 i pokazati dežurnom asistentu na početku vježbe.

Napomena:

Pripremu za laboratorijsku vježbu potrebno je napisati vlastoručno na ovom papiru. Student koji prilikom ulaska u dvoranu nema napisanu pripremu nema pravo pristupa izvođenju iste i smatra se da ju nije odradio.

Dio	Naziv dijela programa	Funkcija dijela programa
1.)	Pretprocesorske direktive	Omogućuju upravljanje razvojnom okolinom, uključivanje biblioteka sa standardnim metodama (System, System.Collection.Generic i System.Text)
2.)		
3.)		
4.)		
5.)		
6.)		
7.)		
8.)		
9.)		

Potpis studenta

Potpis dežurnog asistenta

Rad na vježbi

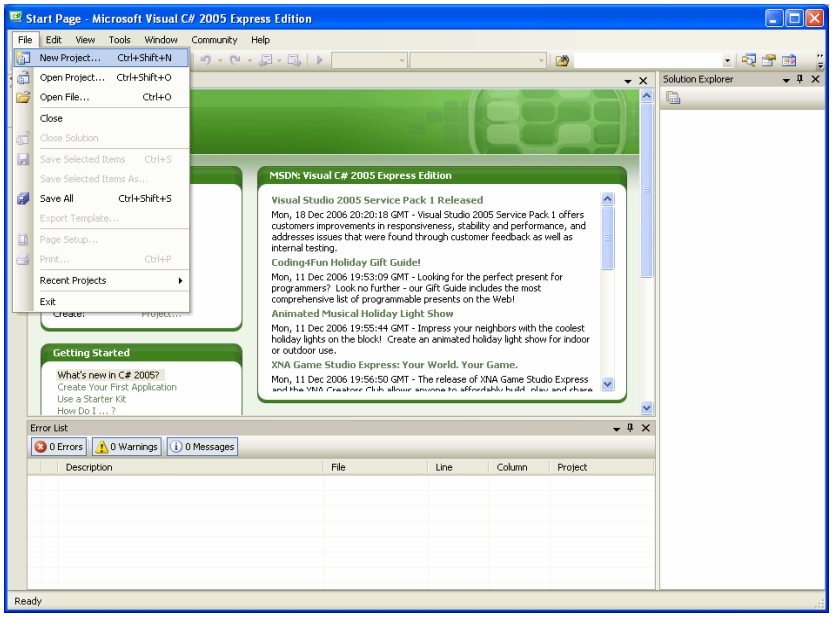
Rad na vježbi sastoji se od unosa danog programskog kôda te njegovog testiranja i ispravljanja unutar besplatne inačice razvojne okoline MS Visual Studio Express Edition.

„Hej svijete!“ – izrada i testiranje prvog projekta

Prvi dio vježbe sastoji se izrade jednostavnog „Hej svijete!“ projekta. U tu svrhu potrebno je, nakon pokretanja razvojne okoline, odabrati opciju „File -> New project“ kao što je prikazano na slici 10.6. Otvara se prozor sa mogućim predlošcima projekta te prostorom za unos imena istog. Obzirom da se radi o projektu sa tekstualnim sučeljem, potrebno je odabrati predložak „Console Application“. U prostor za zadavanje imena projektu potrebno je unijeti „proba1“. Potvrdom unosa sa „OK“ razvojna okolina automatski kreira sve potrebne datoteke te strukturu mapa potrebnu za novo kreirani projekt.

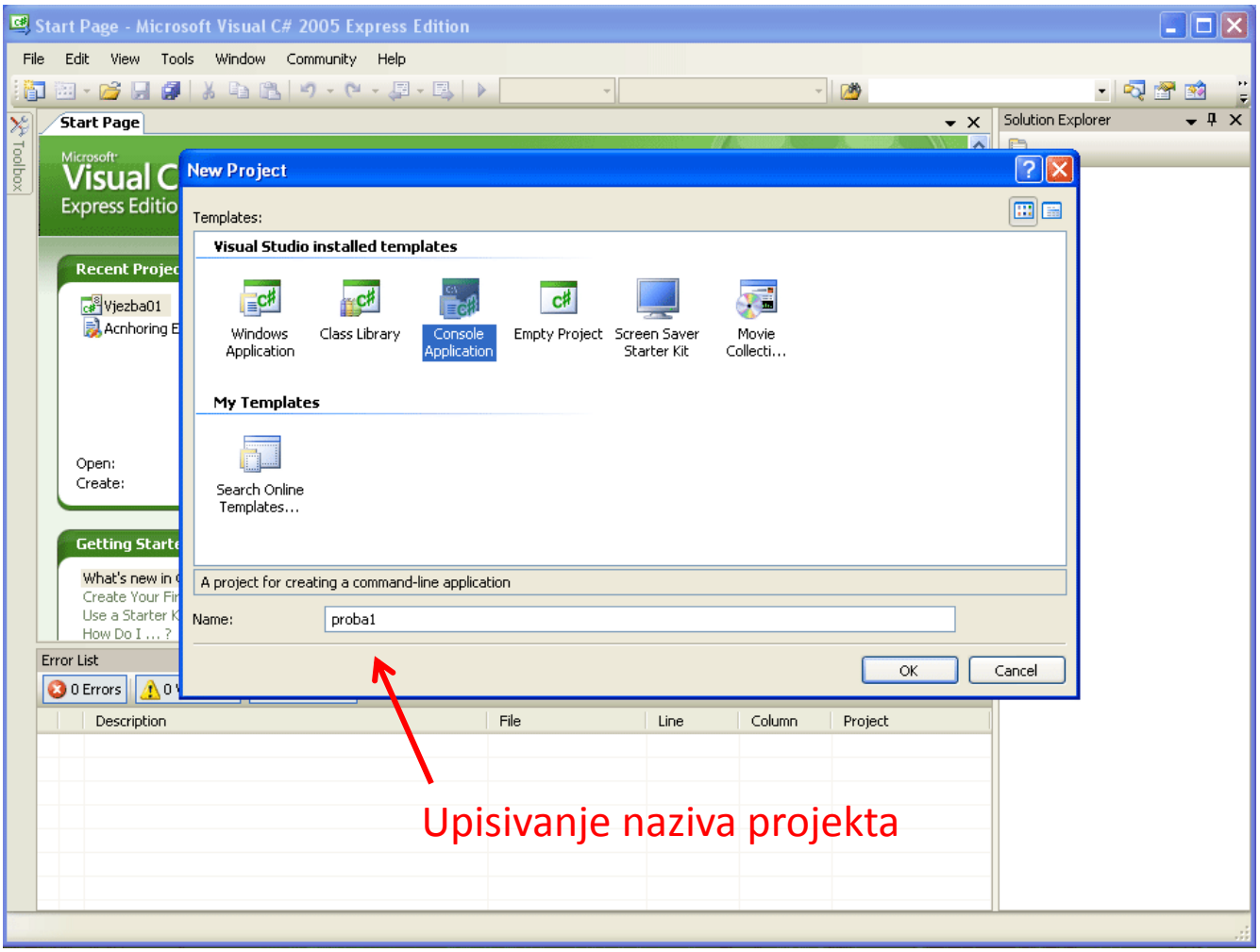
Rezultat ove operacije prikazan je na slici 10.8. U editoru za unos programskog kôda nalazi se dio vezan za uključivanje imeničkih prostora sa sistemskim metodama te imenički prostor i klasa novo kreiranog projekta sa pripadnom glavnom metoda. Glavna metoda označena je nazivom „Main“ i vitičaste zagrade ispod nje vezane su za programski kôd koji pripada glavnoj metodi. Navedeni programski kôd se naziva i tijelo metode. U taj prostor potrebno je unijeti sljedeći programski kôd za ispis pozdravne poruke:

```
System.Console.WriteLine("Hello world!");
System.Console.ReadKey();
```

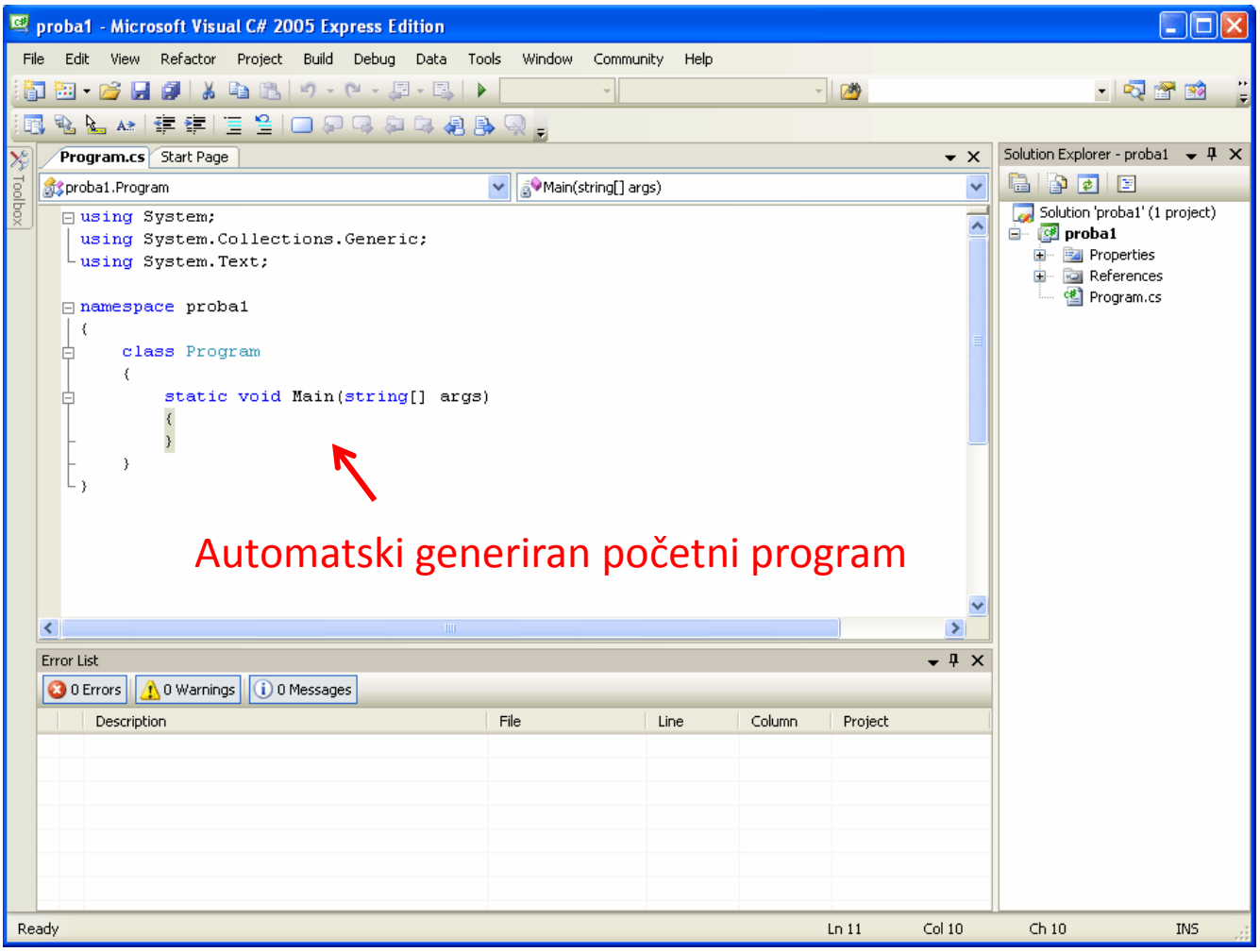


Slika 10.6 Kreiranje novog projekta

Izgled editora programskog kôda nakon unosa prikazan je na slici 10.9. Nakon završenog unosa programskog kôda potrebno je napraviti provjeru ispravnosti te kreirati izvršnu datoteku odabirom opcije izbornika „Build -> Build proba1“ istovjetno prikazanome na slici 10.3. Ukoliko nakon kreiranja izvršne datoteke u donjem dijelu prozora nema poruka o pogreškama, aplikacija je spremna za izvršavanje. Za izvršavanje aplikacije moguće je iskoristiti funkciju na traci sa alatima označenoj na slici 10.9. Na zaslonu monitora pojaviti će se prikaz kao na slici 10.10. Pritiskom na bilo koju tipku prozor s konzolom se zatvara i aplikacija prestaje sa radom.



Slika 10.7 Odabir predloška aplikacije



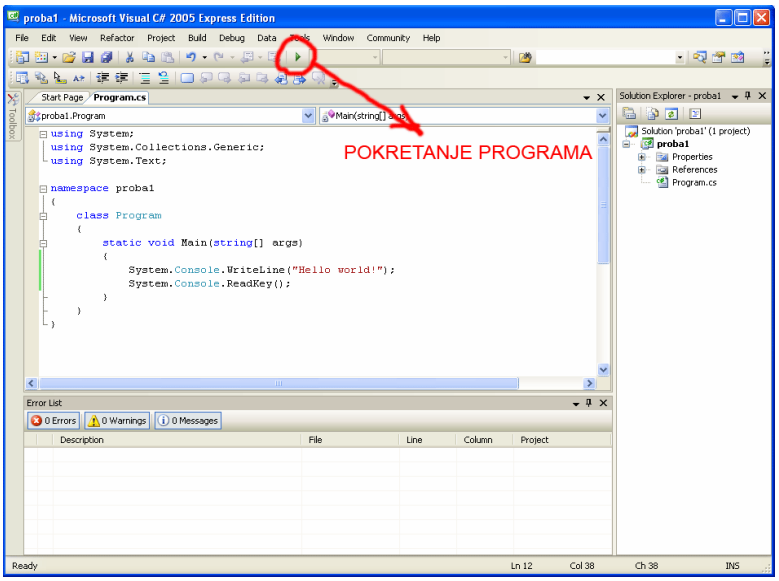
Slika 10.8 Generirani predložak za aplikaciju sa tekstualnim sučeljem

Izrada i testiranje aplikacije za zbrajanje dva broja

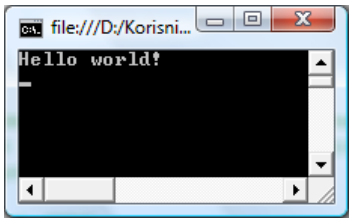
U drugom dijelu vježbe unijeti će se programski kôd koji omogućuje zbrajanje dva cijela broja te testirati dobivenu aplikaciju. Pri tome se aplikacija sastoji od 5 dijelova: (i) deklaracija varijabli, (ii) inicijalizacija varijabli, (iii) učitavanje podataka, (iv) obrada podataka te (v) ispis rezultata izvođenja aplikacije. Pripadni programski kôd koji je potrebno unijeti dan je u nastavku.

```
// deklaracija varijabli
int pribrojnik1, pribrojnik2, suma;

// inicijalizacija varijabli
pribrojnik1 = pribrojnik2 = suma = 0;

// ucitavanje podataka
Console.Write("Unesite prvi pribrojnik >> ");
pribrojnik1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Unesite drugi pribrojnik >> ");
pribrojnik2 = Convert.ToInt32(Console.ReadLine());

// obrada podataka
suma = pribrojnik1 + pribrojnik2;

// ispis podataka
Console.Write("Suma pribrojnika iznosi >> ");
Console.Write(suma);
```

```
// da ispis ostane vidljiv
Console.ReadLine();
```



Slika 10.9 Pokretanje programa nakon uspješnog kreiranja izvršne datoteke

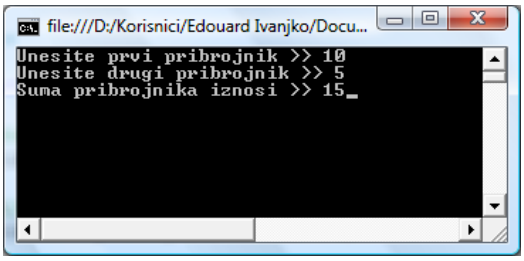


Slika 10.10 Rezultat izvođenja

Zadnji redak (naredba *Console.ReadLine()*) dodan je kako bi ispis rezultata izvođenja aplikacije ostao vidljiv u konzoli. Tekst označen zelenom bojom je komentar koji se može ispustiti prilikom unosa kôda u razvojnu okolinu, a služi za opis funkcionalnosti kôda.

Za rješenje ovog dijela vježbe potrebno je zatvoriti prvi projekt „proba1“ odabirom opcije „File -> Close solution“. Nakon toga potrebno je kreirati novi projekt sa nazivom „Zbrajanje“ na način kako je opisano u prvom dijelu vježbe. U tako kreirani novi predložak potrebno je unijeti programski kôd dan na prethodnoj stranici. Odabirom opcije „Build -> Build Zbrajanje“ izvrši se provjera ispravnosti programskog kôda i kreira izvršna datoteka. Ukoliko je sve ispravno moguće je pristupiti izvršavanju aplikacije te njenom testiranju. U slučaju postojanja grešaka potrebno je iste ispraviti upotrebom alata za ispravljanje programskog koda na način opisan u uvodu 3. dijela vježbe (stranica 10).

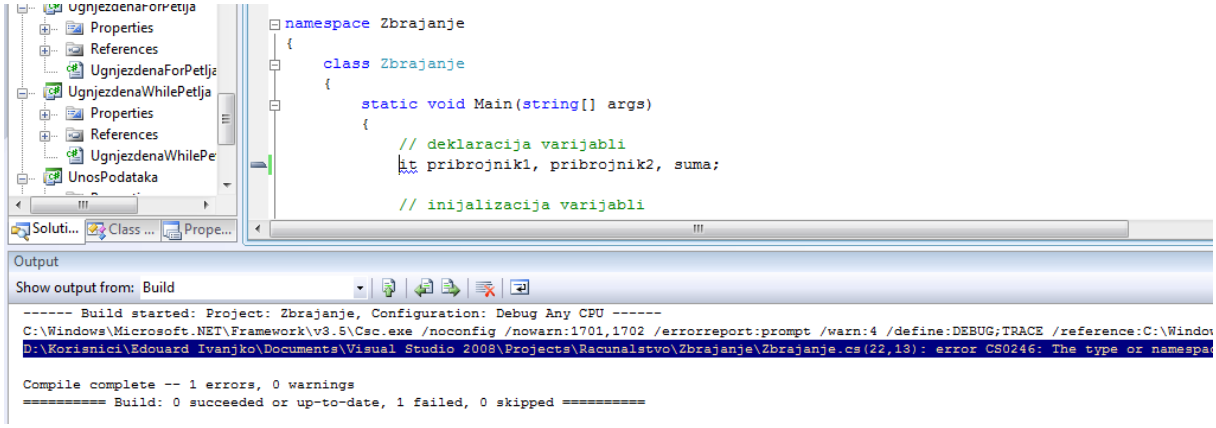
Pokrenite aplikaciju pripadnom funkcijom za pokretanje programa na traci sa alatima. Prilikom izvođenja dobiva se prikaz kao na slici 10.11. Ulazne vrijednosti za pojedine pribrojnike moguće je zadati po volji poštujući format zadavanja cijelih brojeva (samo znamenke bez decimalne točke ili zareza). Unos se potvrđuje pritiskom na tipku „Enter“. Za završetak aplikacije pritisnite bilo koju tipku tipkovnice.



Slika 10.11 Rezultat izvođenja aplikacije za zbrajanje dva cijela broja

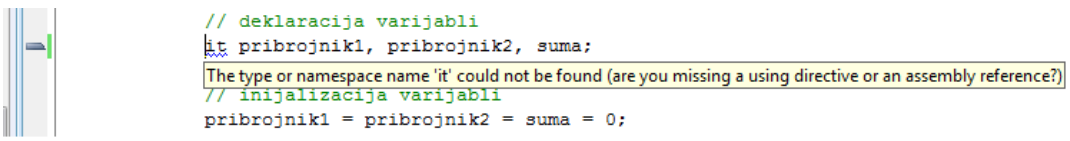
Korištenje alata za ispravljanje programskog kôda aplikacije

U trećem dijelu vježbe proučiti će se mogućnosti alata za ispravljanje programskog kôda aplikacije. Prvi korak je provjera ispravnosti programskog kôda vezana za sintaksu naredbi. Obavlja se prilikom prevođenja i kreiranja izvršne datoteke. Uzmimo npr. da je pogrešno napisana ključna riječ cijeloga broja za deklarirane varijable odnosno umjesto „int“ u programskom kodu nalazi se „it“. U dijelu prozora razvojne okoline za poruke i izvješća (prozor „Output“) pojavljuje se poruka o grešci. Dvostrukim klikom lijevom tipkom miša na opis pojedine greške moguće je pozicioniranje na pripadnu liniju kôda. Linija kôda sa greškom označena je kratkom horizontalnom svjetlo plavom te vertikalnom zelenom crticom sa lijeve strane prozora za unos programskog kôda. Pri tome je dio problematičnog koda podcrtan plavom valovitom crtom radi lakšeg otkrivanja problema. Opisana situacija prikazana je na slici 10.12. Ukoliko postoji više grešaka, procedura pozicioniranja i ispravljanja ponavlja se za svaku otkrivenu grešku.



Slika 10.12 Pozicioniranje na liniju kôda sa pogreškom

Za testiranje razumijevanja ovog postupka unesite u programski kôd opisanu grešku te koristite opisane korake ispravljanja. Rezultate izvješća greške unesite u tablicu 10.1. Nakon toga izbrišite deklaraciju varijable „suma“ i ponovite cijelu proceduru ispravljanja grešaka. Rezultate izvješća greške unesite u tablicu 10.2. Radi lakšeg otklanjanja greške moguće je dobiti i detaljnije objašnjenje problema tako da se pokazivač miša pozicionira iznad problematičnog dijela kôda označenog sa valovitom plavom crtom. Pripadni slučaj prikazan je na slici 10.13.



Slika 10.13 Detaljnije objašnjenje problema

Ukoliko je sintaksa programskog kôda ispravna moguće je da postoje logičke greške koje se pojavljuju tijekom izvođenja aplikacije. Takve greške teže se otkrivaju i za njihovo otkrivanje koriste se ulazni testni podaci sa poznatim rezultatima kako bi se mogla napraviti usporedba sa rezultatom koji vraća

aplikacija. Alat koji se koristi u tom slučaju su točke prekida (eng. Breakpoints). Za analizu izvođenja izrađene aplikacije potrebno je postaviti točke prekida na liniju kôda za inicijalizaciju varijabli, na kraj učitavanja vrijednosti varijabli, na liniju obrade podataka te na kraj ispisa podataka kako je prikazano na slici 10.14.

Pokretanjem aplikacije se sada automatski uključi način rada specifičan za praćenje izvođenja. Aplikacija se izvodi do prve točke prekida kada staje i omogućuje programeru da provjeri stanje pojedinih varijabli kako je prikazano na slici 10.15.

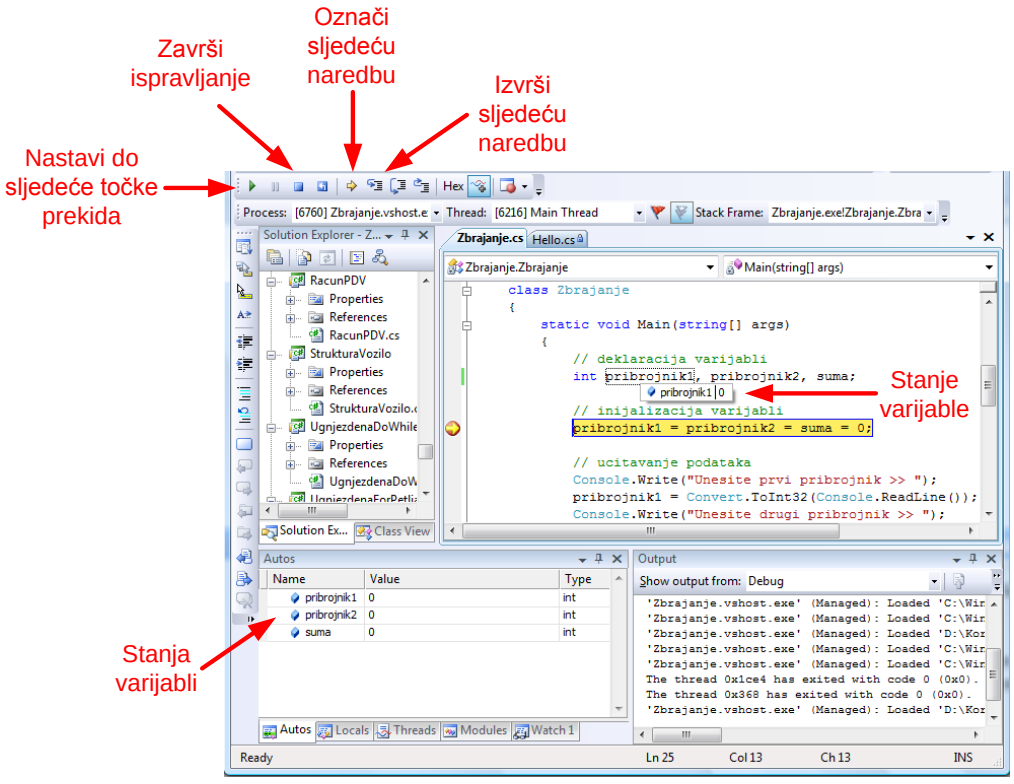
Žutom strelicom označena je naredba koja će se sljedeća izvesti, a ne koja se izvodi trenutno (trenutno se izvodi kôd u prethodnom retku). U donjem lijevom uglu nalazi se prozor za ispis stanja varijabli korištenih u trenutno aktivnoj liniji kôda. Stanje pojedine varijable u ovom načinu rada razvojne okoline može se dobiti tako da pokazivač miša pozicioniramo iznad interesantne varijable u kôdu. Kako je prikazano na slici 10.15 ispod varijable se pojavi pravokutnik sa imenom varijable i njezinom vrijednošću. Pri tome se otvori konzola i korisnik koristi aplikaciju kao i obično uz ograničenje da se unos podataka te prikaz rezultata izvršava u trenutku izvršavanja pripadne linije kôda. Aplikacija se ne izvršava sve dok se pripadnom funkcijom alatne trake ne pokrene izvršavanje potrebnog kôda.

Za daljnje izvršavanje programa koriste se funkcije u gornjem lijevom dijelu trake sa alatima kako je prikazano na slici 10.15. Od opcija je moguće koristiti nastavak izvršavanja programa do sljedeće točke prekida (zelena strelica), završiti ispravljanje aplikacije (plavi pravokutnik), označiti sljedeću naredbu za izvršavanje (žuta strelica) te izvršiti samo sljedeću naredbu. Zadnja opcija je najkorisnija kada se želi detaljnije ispitati kritičan dio kôda.

Nakon što je izvođenje aplikacije došlo do prve točke prekida provjerite stanja pojedinih varijabli na gore opisani način. Zatim aktivirajte funkciju izvršavanja sljedeće naredbe te pratite što se događa u donjem lijevom uglu gdje se ispisuju stanja varijabli koje se trenutno obrađuju. Koristite ovu funkciju dok program ne dođe do sljedeće točke prekida. Nakon toga koristite funkciju alatne trake za izvršavanje aplikacije do sljedeće prekidne točke. Obratite pažnju da je kod dijelova kôda za učitavanje podataka potrebno unijeti tražene brojeve u konzoli da bi aplikacija nastavila s izvršavanjem. Vrijednosti varijabli za svaku točku prekida unesite u tablicu 10.3.

```
namespace Zbrajanje
{
    class Zbrajanje
    {
        static void Main(string[] args)
        {
            // deklaracija varijabli
            int pribrojnik1, pribrojnik2, suma;

            // inijalizacija varijabli
            pribrojnik1 = pribrojnik2 = suma = 0;

            // ucitavanje podataka
            Console.Write("Unesite prvi pribrojnik >> ");
            pribrojnik1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Unesite drugi pribrojnik >> ");
            pribrojnik2 = Convert.ToInt32(Console.ReadLine());

            // obrada podataka
            suma = pribrojnik1 + pribrojnik2;

            // ispis podataka
            Console.Write("Suma pribrojnika iznosi >> ");
            Console.Write(suma);

            // da ispis ostane vidljiv
            Console.ReadLine();

        }
    }
}
```

Slika 10.14 Postavljanje točki prekida



Slika 10.15 Stanja varijabli kod prve točke prekida

Rezultati laboratorijske vježbe

Kôd pogreške	MS Visual Studio izvješće pogreške	Broj linije kôda

Tab. 10.1 Izvješće ispravljanje pogreške krivog tipa podatka

Kôd pogreške	MS Visual Studio izvješće pogreške	Broj linije kôda

Tab. 10.2 Izvješće ispravljanja pogreške nedostajuće deklaracije varijable

Varijabla	pribrojnik1	pribrojnik2	suma
Unijete vrijednosti			
Prva točka prekida			
Druga točka prekida			
Treća točka prekida			
Četvrta točka prekida			

Tab. 10.3 Vrijednosti varijabli prilikom izvođenja programa

Napomena

Rezultate laboratorijske vježbe unesene u tablice 10.1 do 10.3 potrebno je na kraju laboratorijske vježbe pokazati dežurnom asistentu. U slučaju da student nije pokazao rezultate laboratorijske vježbe smatra se da istu nije odradio te ju je dužan nadoknaditi.

Potpis studenta Potpis dežurnog asistenta

Korisne napomene za rad na vježbi

- Vitičaste zagrade se dobivaju na HR tipkovnici kombinacijama tipki „Alt Gr“ + „b“ (lijeva vitičasta zagrada -> „**{**“) te „Alt Gr“ + „n“ (desna vitičasta zagrada -> „**}**“).
- Uglate zagrade se dobivaju na HR tipkovnici kombinacijama tipki „Alt Gr“ + „f“ (lijeva uglata zagrada -> „**[**“) te „Alt Gr“ + „g“ (desna uglata zagrada -> „**]**“).
- Tipkom „Delete“ ili „Del“ se briše znak ispred kursora.
- Tipkom „Backspace“ se briše znak iza kursora.
- Tipkom „Ins“ ili „Insert“ se aktivira opcija pisanja preko postojećeg teksta tako da novi znak prebriše postojeći znak. Aktivira se pritiskom na tipku „Ins“ ili „Insert“, a deaktivira ponovnim pritiskom.
- Potvrda unosa podatka u tekstualnom sučelje je pritisak na tipku „Enter“.
- Znak „**|**“ se dobiva na HR tipkovnici kombinacijom tipki „Alt Gr“ + „w“.

- Znakovi manje i veće se dobivaju na hrvatskoj tipkovnici kombinacijama tipki „Alt Gr“ + „**,**“ za znak „**<**“ te „Alt Gr“ + „**.**“ za znak „**>**“.