

Microsoft Visual C++ 2008

Uputstvo za upotrebu
i jednostavni primeri

Sadržaj

- ◆ Objedinjeno razvojno okruženje
- ◆ Osobine C/C++ razvojnih okruženja
- ◆ Detalji rada u MSVC
- ◆ Primeri
 - Jednostavan C program
 - Složeniji C primer
 - Kreiranje biblioteke
- ◆ Završne napomene

Objedinjeno razvojno okruženje

- ◆ Istorijat

- ◆ Program i okruženje

- ◆ Put do izvršnog programa

- ◆ Struktura tipičnog okruženja:

- Editor izvornog programskog koda
- Prevodilac (engl. *compiler*)
- Povezivač (engl. *linker*)
- Debager (engl. *debugger*)
- Menadžer datoteka

Istorijat [1/2]

- ◆ Rani programi su pripremani korišćenjem bušenih kartica ili magnetnih doboša (1890 – 1975).
- ◆ Tastatura je postala sredstvo za unos komandi sa pojavom UNIX operativnih sistema, korišćenim na prvim računarima koji su imali silicijumske čipove (1970 -).

Istorijat [2/2]

- ◆ Sa rastom zahteva koje softver treba da ispuni porasli su i zahtevi za uslovima koje razvojno okruženje treba da obezbedi programerima.
- ◆ To je dovelo do nastanka objedinjenih razvojnih okruženja (engl. *Integrated Development Enviroment*)
- ◆ IDE se može posmatrati kao jedinstven alat u kojem se obavlja čitav razvoj softvera.

Program i okruženje [1/2]

- ◆ Program više nije mogao biti razvijan unutar samo jedne datoteke (engl. *file*).
- ◆ Savremeni programi se sastoje od većeg broja datoteka koje su organizovane u projekat (engl. *project*).
- ◆ Svrha projekta je da informacije iz svih datoteka budu lakše međusobno dostupne.

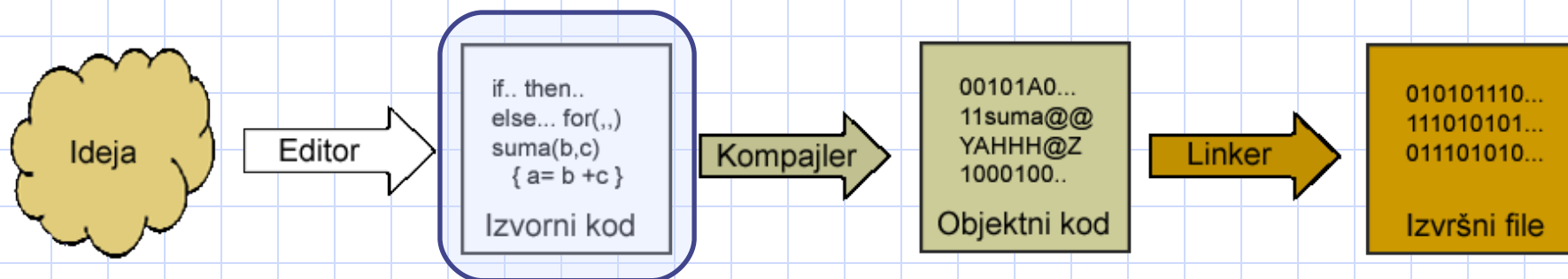
Program i okruženje [2/2]

◆ Osnovne celine unutar projekta:

- Datoteka (fajl, engl. *file*) – skup informacija ili izvornog koda koji je organizovan (snimljen) kao jedna sistemska celina
- Direktorijum (engl. *folder*) – sadrži više datoteka i/ili druge direktorijume, služi za njihovo organizovanje

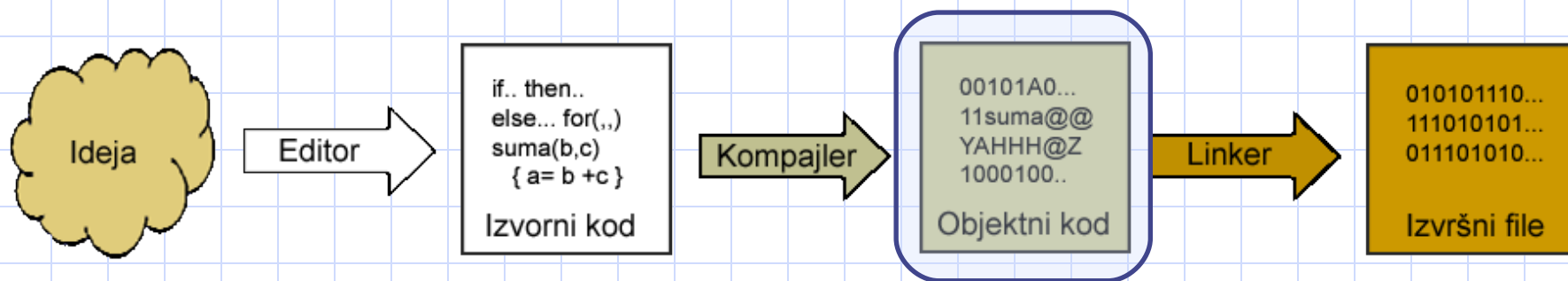
◆ Rad sa projektima se obavlja unutar IDE.

Put do izvršnog programa [1/3]



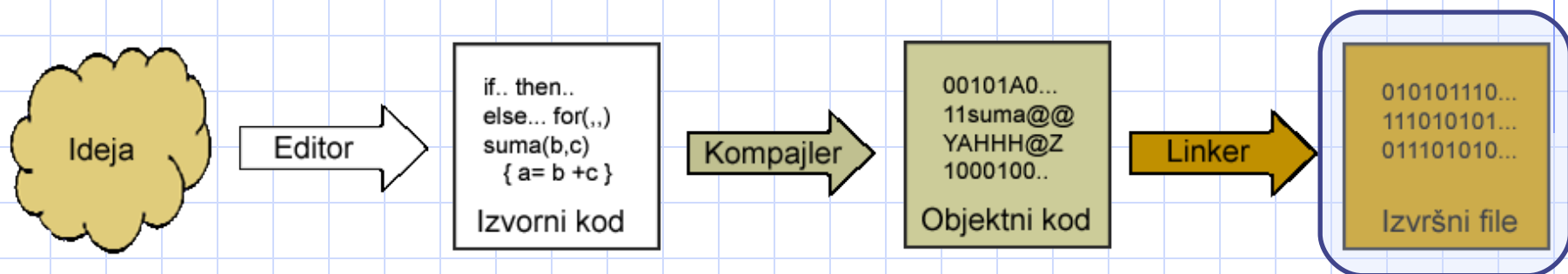
- ◆ IDE omogućava da se čitav proces razvoja programa, od ideje do konačnog rezultata (izvršni *file*), obavi na jednom mestu.
- ◆ Izvorni programski kod (engl. *source code*) je skup naredbi napisan u nekom od programskih jezika.

Put do izvršnog programa [2/3]



- ◆ Objektni kod (engl. *object code*) je mašinski kod generisan iz izvornog koda.
- ◆ Objektni kod, iako je mašinski kod, ne može se izvršavati.

Put do izvršnog programa [3/3]



- ◆ Izvršni fajl (engl. *executable*) je fajl čiji sadržaj računar interpretira kao program.
- ◆ Izvršni fajl se može pokretati samostalno, nezavisno od okruženja u kojem je razvijan.
- ◆ Sačinjen je od binarnih informacija (0 i 1).

Editor izvornog programskog koda

- ◆ Editor izvornog programskog koda je tekstualni editor specijalno prilagođen za uređivanje izvornog koda, olakšava i ubrzava rad programera svojim specijalnim funkcionalnostima.
- ◆ I obični tekst editori (npr. "Notepad") mogu se koristiti za uređivanje koda, ali usled nedostatka specijalnih funkcionalnosti ne mogu se smatrati "editorima izvornog programskog koda".

Specijalne funkcionalnosti [1/2]

- ◆ *Syntax highlighting* – delove izvornog koda editor automatski prikazuje u drugoj boji i/ili drugim fontom u zavisnosti od njihovog značenja i konteksta.
- ◆ *Autocomplete* – na osnovu predviđanja editor omogućava programeru da automatski, bez unošenja čitave celine, kompletira taj deo izvornog koda.

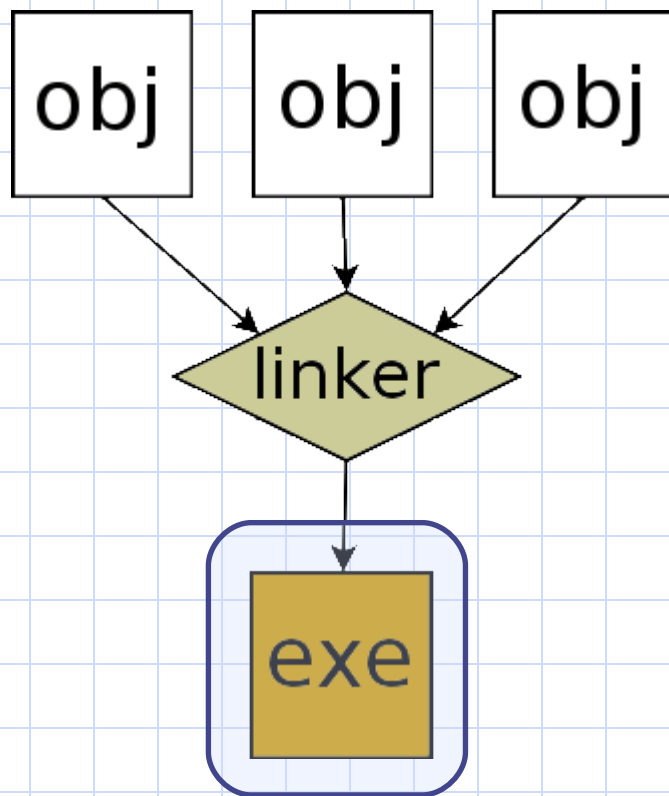
Specijalne funkcionalnosti [2/2]

- ◆ Automatsko formatiranje koda – editor omogućava da se vizuelno strukturira izvorni kod na način koji omogućava programeru lakše praćenje istog.
- ◆ Izveštaji o greškama – u toku samog unošenja koda editor prijavljuje bazične greške, semantičkog ili sintaksnog tipa.

Compiler

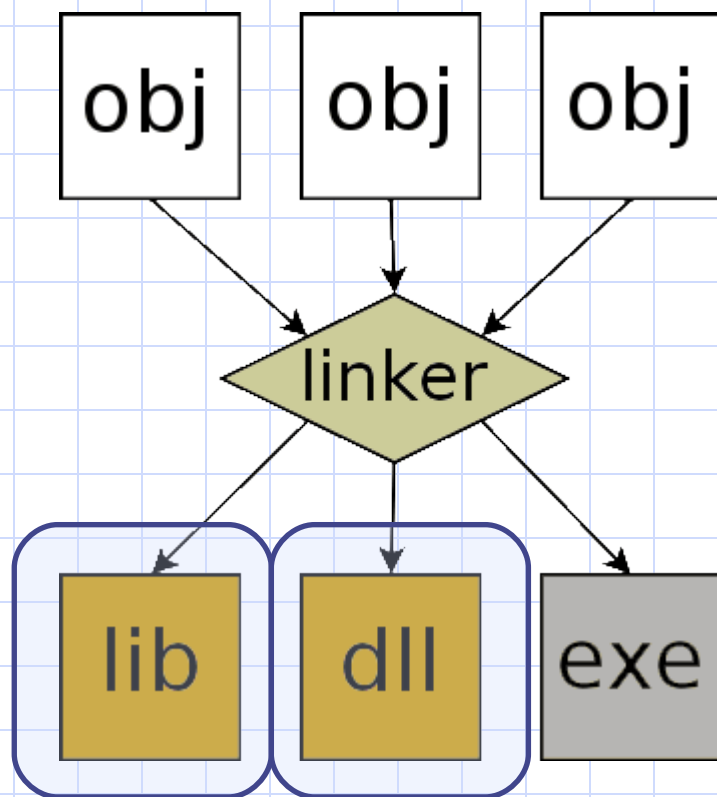
- ◆ U najširem smislu, *Compiler* je alat koji prevodi tekst napisan u nekom od programskih jezika u drugi programski jezik.
- ◆ Praktična upotreba je prevođenje iz programskog jezika višeg u jezik nižeg nivoa, najčešće u simbolički mašinski jezik (često se pogrešno naziva i *assembler*) ili mašinski jezik.

Linker [1/2]



- ◆ *Linker* je alat koji od jedne ili više datoteka koje sadrže objektni kod kreira jedinstveni izvršni *file* ili biblioteku.
- ◆ Ekstenzija u sistemu izvršnog fajla je *.exe*, od engl. *executable*.

Linker [2/2]



- ◆ Rezultat rada linkera ne mora biti *executable*, već i biblioteka.
- ◆ Biblioteke (eng. *library*) su skupovi potprograma.
- ◆ Statičke biblioteke (`lib`) se koriste u prevođenju i njihov kod se neposredno ugrađuje u izvršni program.
- ◆ Dinamičke biblioteke (`dll`) se ne ugrađuju u izvršni program, ali moraju biti dostupne kada se program izvršava.

Debugger

- ◆ *Debugging* – metodički proces otkrivanja i eliminisanja grešaka u programu.
- ◆ *Debugger* pomaže da se nađe mesto u kodu u kojem je uzrok greške tako što pokazuje kontekst izvršavanja u svakom trenutku i vrednosti relevantnih promenljivih.
- ◆ Postoji više načina da se sprovede *debugging* postupak.

Menadžer datoteka

- ◆ Savremena razvojna okruženja vizuelno prikazuju strukturu projekta, uključene direktorijume i datoteke. Direktno iz menadžera se datoteke uključuju ili isključuju iz projekta.
- ◆ Datoteke mogu biti sistemske ili ih piše sam programer.
- ◆ Iz menadžera datoteka programer odabira datoteku kojoj želi da menja sadržaj, ona se otvara u editoru izvornog programskog koda.

Objedinjeno razvojno okruženje

◆ Primeri:

- Microsoft Visual Studio
- Eclipse
- ActiveState Komodo

◆ Eclipse je primer višejezičkog okruženja i jezik Java je podržan u njemu ali postoje dodaci za C/C++, Python, Perl, PHP, Fortran, Cobol...

Visual IDE

- ◆ Stalno se povećava interesovanje inženjera za vizuelnim programiranjem (visual programming) što je dovelo do razvoja mnogih vizuelnih IDE
- ◆ Vizuelno IDE omogućava korisnicima pravljenje aplikacija razmeštanjem grafičkih elemenata na ekranu i radom sa tim elementima, što je najčešće brže od uobičajenog programiranja

Osobine C/C++ razvojnih okruženja

- ◆ Rad sa projektima
- ◆ Standardne biblioteke
- ◆ Tok kreiranja programa

Rad sa projektima

◆ Projekat sadrži jednu ili više datoteka koje mogu biti sledećih tipova:

- (`.c`, `.cpp`, `.cxx`, `.cc`, `.C`) *source* – sadrži izvorni kod
- (`.h`, `.hpp`, `.hxx`) *header* – sadrži definicije funkcija i tipove podataka koji se nalaze u drugim datotekama
- (`.obj`) *object* – kompajlirane datoteke koje sadrže objektni kod

Napomena:
.obj fajl
ne sadrži objekte

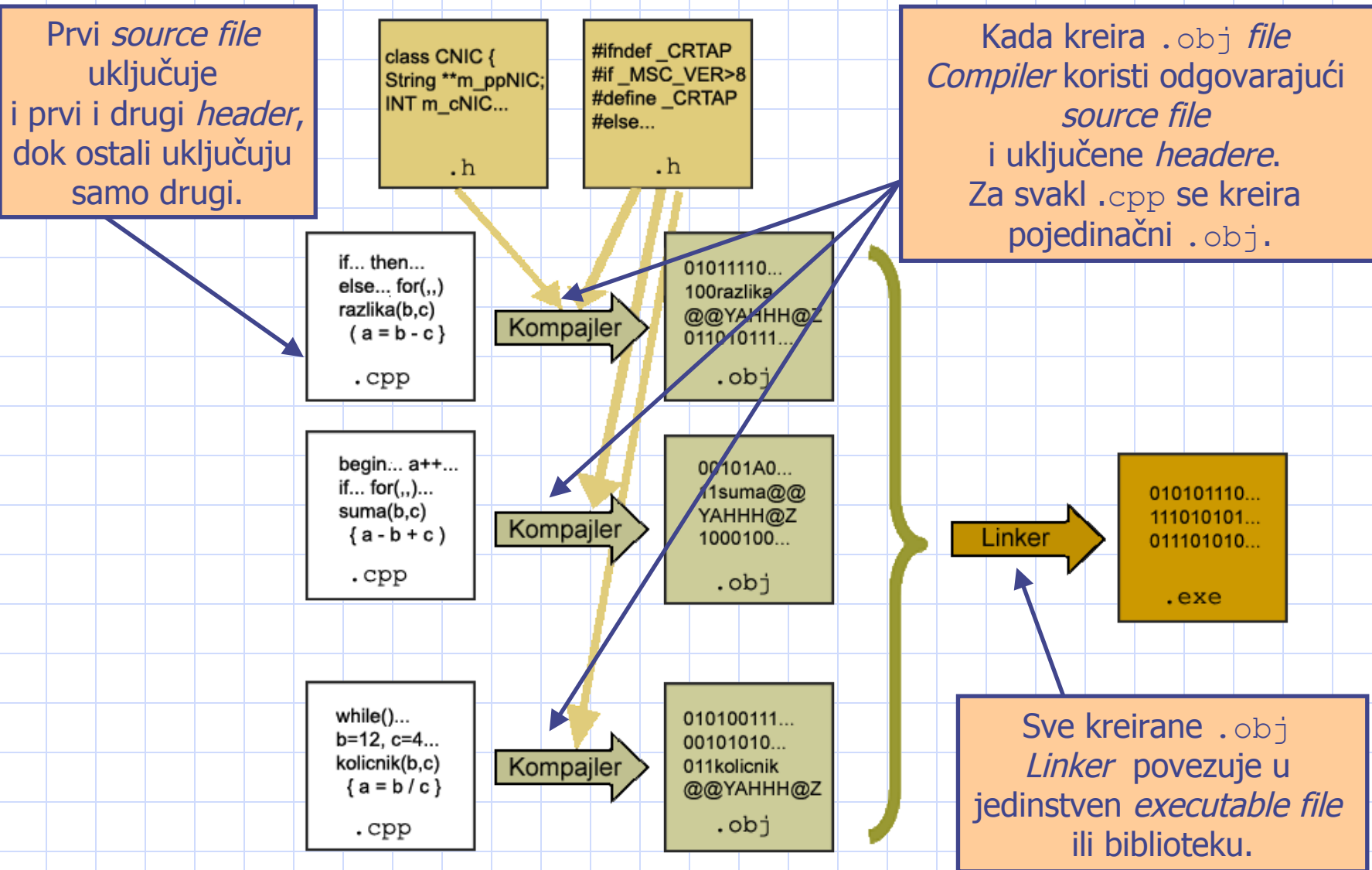
Standardne biblioteke

- ◆ Zbirke funkcija, konstanti, klasa i objekata koje proširuju osnovnu funkcionalnost C/C++ jezika i omogućavaju mu komunikaciju sa operativnim sistemom i korišćenje nekih standardnih algoritama
- ◆ Biblioteke se koriste pomoću zaglavlja, koja se po potrebi zasebno uključuju iz svakog fajla unutar projekta

Zaglavlja (C)

- ◆ `stdio.h` – komunikacija sa I/O uređajima
- ◆ `stdlib.h` – rad sa memorijom, konverzije podataka
- ◆ `time.h` – rad sa datumima i vremenskim jedinicama
- ◆ `string.h` – manipulacija stringovima (tekst)
- ◆ `math.h` – matematičke operacije

Tok kreiranja programa



Detalji rada u MSVC

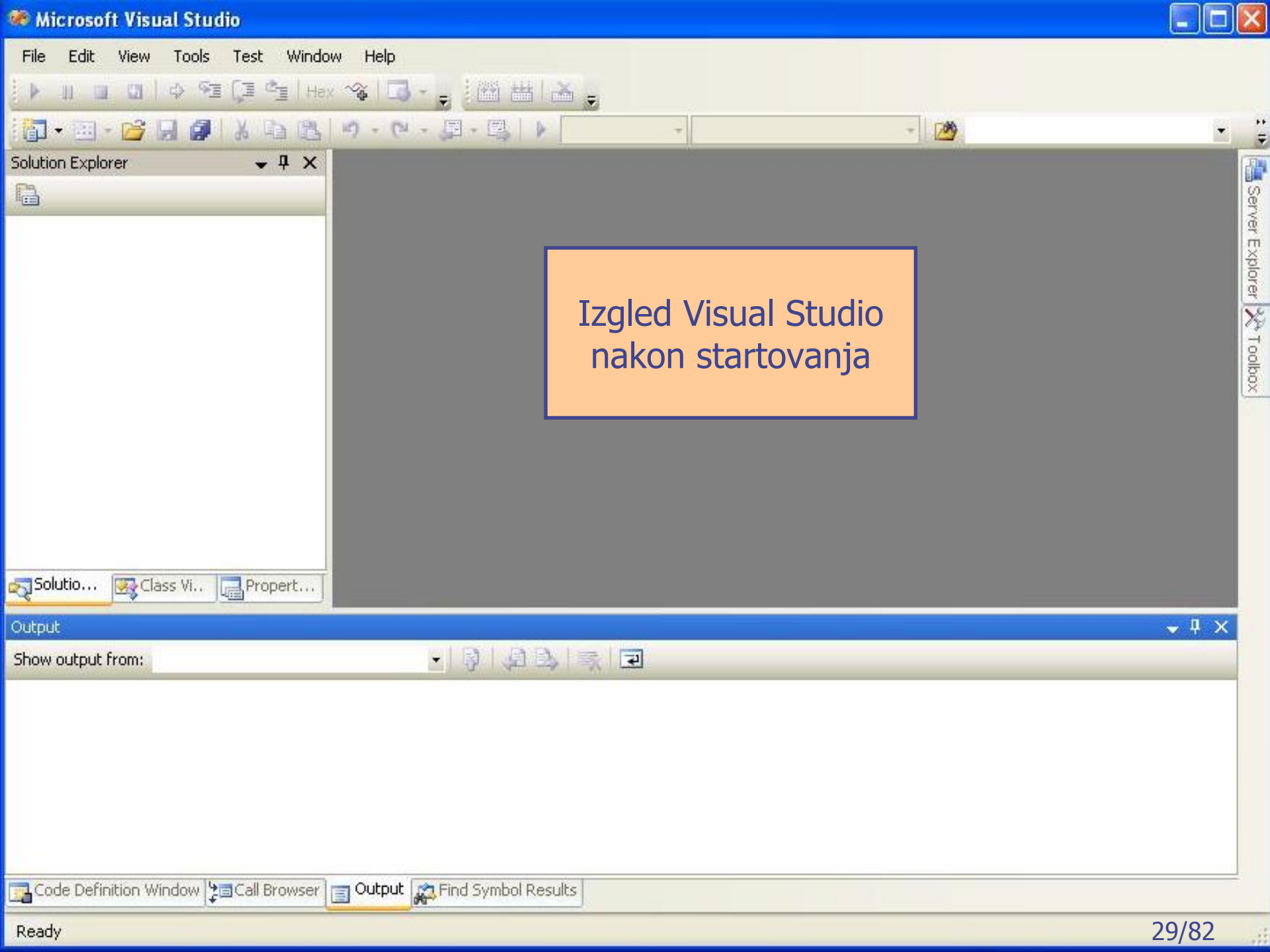
- ◆ Uvod u MSVC
- ◆ Radni prostor
- ◆ Kreiranje projekta
- ◆ Dodavanje fajla u projekat
- ◆ Unos koda
- ◆ Palete sa alatkama (engl. *toolbars*)
- ◆ *Build* meni
- ◆ Pokretanje aplikacije
- ◆ Korišćenje *Debugger* alata

Uvod u MSVC [1/2]

- ◆ MSVC (Microsoft Visual C++) je IDE za razvoj C/C++ aplikacija.
- ◆ Visual C++ je deo paketa Visual Studio 2008, proizvoda kompanije Microsoft
- ◆ Aktuelna verzija nosi oznaku 2008 (objavljena novembra 2007. godine).
- ◆ Visual Studio 2008 sadrži mnoge komponente, od kojih su najvažnije:
 - Visual C++
 - Visual C#
 - Visual Basic

Uvod u MSVC [2/2]

- ◆ Upoznavanje sa MSVC biće pokazano kroz jednostavan primer koji sabira dva broja i prikazuje poruku.
- ◆ Preduslov za dalje adekvatno praćenje tutorijala je instaliran MSVC 2008 na računaru.



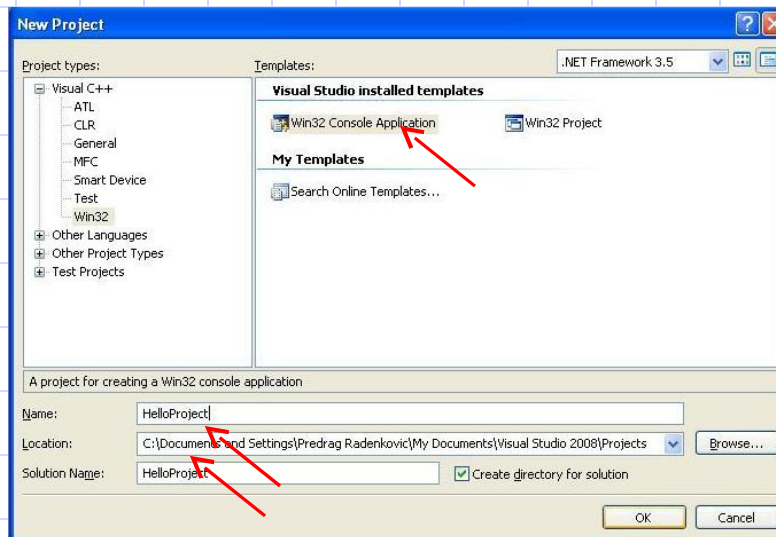
Izgled Visual Studio
nakon startovanja

Server Explorer
Toolbox

Radni prostor i solution

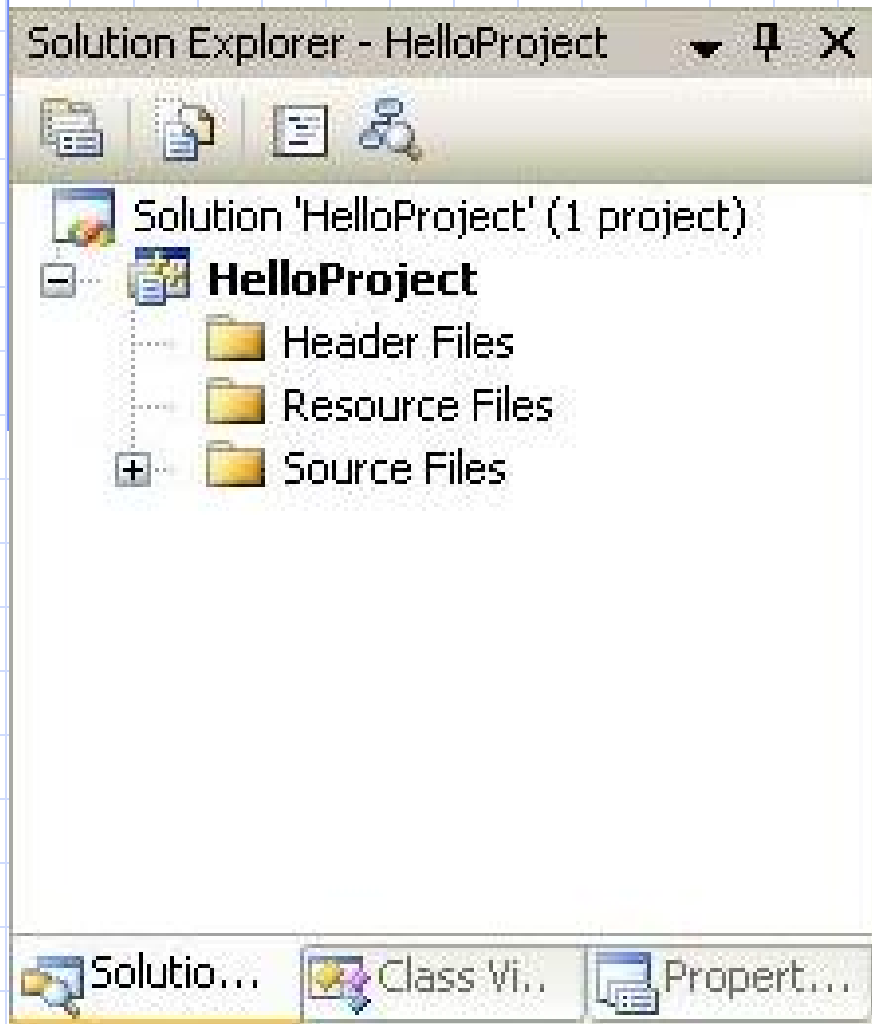
- ◆ Radni prostor (engl. *workspace*) je, u vizuelnom smislu, površina na kojoj se nalaze svi prozori, meniji i opcije razvojnog okruženja
- ◆ Projekti su grupisani u okviru rešenja (engl. *solution*)
- ◆ Jedno rešenje može sadržati veći broj projekata

Kreiranje novog projekta



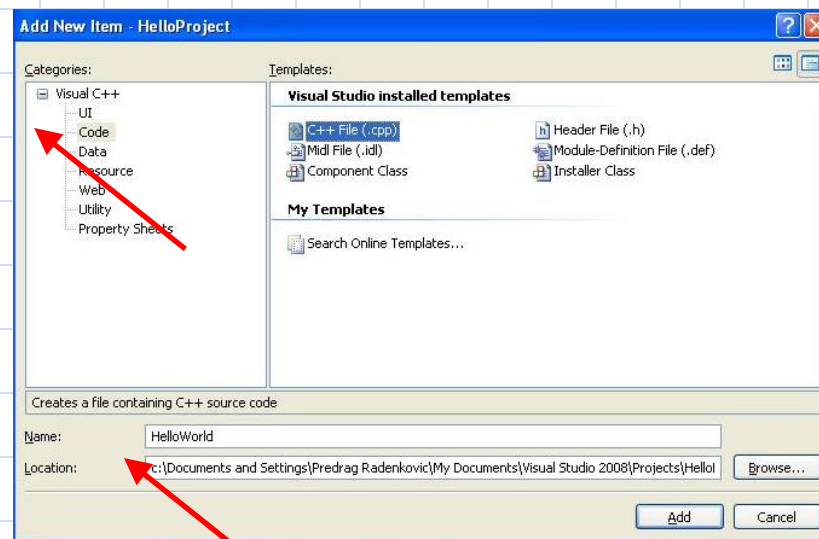
- ◆ U meniju "File" izabrati podmeni "New" i u njemu "Project"
- ◆ Od ponuđenih opcija u dijalogu odabrati "Win32 Console Application"
- ◆ U "Location" odabrati putanju do direktorijuma gde će se nalaziti program
- ◆ U "Name" upisati ime projekta (u ovom primeru: "HelloProject")
- ◆ Pritisnuti "OK".
- ◆ U novonastalom dijalogu odabrati "An empty project"

Solution Explorer



- ◆ Solution Explorer omogućava pregled svih projekata i fajlova uključenih u jedan *solution*.
- ◆ Preko kontekstnog menija je moguće obaviti mnoge potrebne operacije vezane za projekte sadržane u rešenju.

Dodavanje fajla u projekat



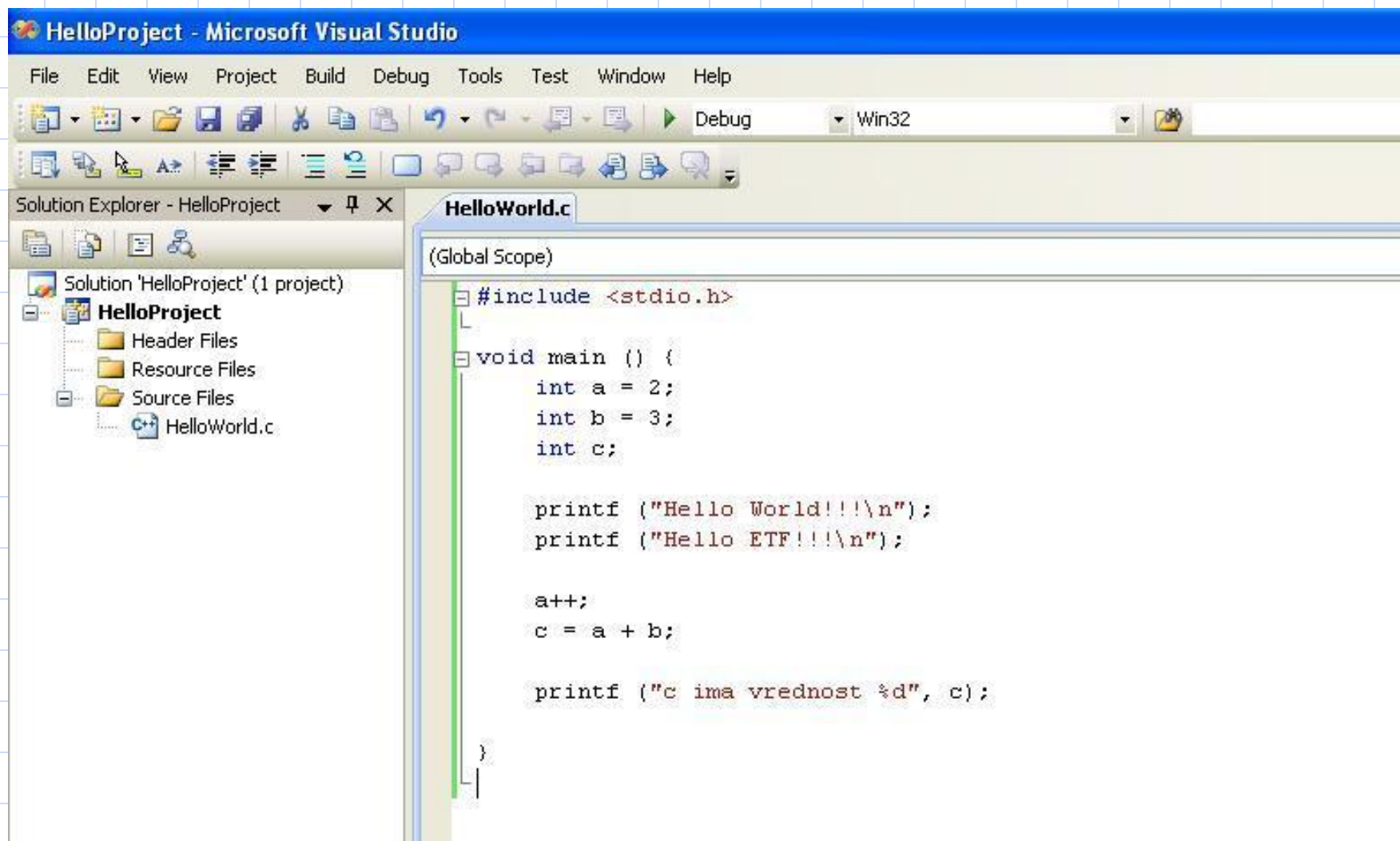
- ◆ U Solution Exploreru desni klik na projekat (ili na folder "Source Files") i u meniju "Add" izabrati "New Item"
- ◆ Od ponuđenih opcija u dijalogu odabrati "Code" i "C++ File (.cpp)"
- ◆ U polju "Name" upisati ime fajla (u ovom primeru "HelloWorld.c")
- ◆ Pritisnuti "Add"

Unos programskog koda

- ◆ U Solution Exploreru raširiti folder sa izvornim fajlovima (npr. "Source Files")
- ◆ Željeni fajl (npr. fajl "HelloWorld.c") dvostrukim klikom otvoriti za uređivanje
- ◆ U editoru sa desne strane uneti kod
- ◆ Snimiti fajl, odabirom "Save" iz menija "File"

Napomena:
Opcija "Save All"
snima sve fajlove
otvorene unutar projekta.

Primer rada sa jednim fajlom



Paleta sa alatkama (engl. *toolbars*)



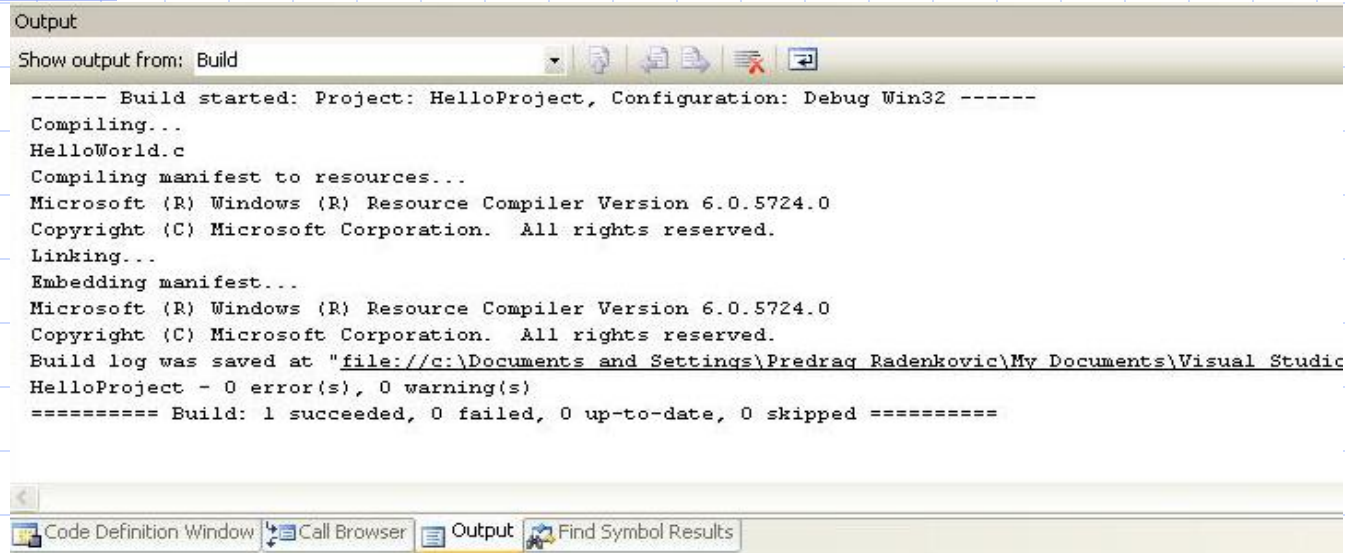
- ◆ Paleta sa alatkama (engl. *toolbar*) je skup dugmadi prikazan kao blok.
- ◆ Pozicija bilo koje palete sa alatkama nije fiksna, moguće je preurediti ih po želji.
- ◆ Ukoliko neki *toolbar* nije prikazan, uključuje se odabirom opcije "Customize" iz menija "Tools", (izabere se kartica "Toolbars" i željeni *toolbar*).

Build i Debug palete sa alatka



- ◆ Build (F7): prevodi sve datoteke u projektu
- ◆ Compile (Ctrl+F7): prevodi samo trenutnu datoteku
- ◆ Stop Build (Ctrl+Break): prekida prevođenje
- ◆ Start Debugging (F5): pokreće projekat-program za kontrolisano izvršavanje (*debug* režim)
- ◆ Start Without Debugging (Ctrl+F5): pokreće projekat-program bez kontrole (**IZBEGAVATI!**)

Izgradnja izlaznih fajlova



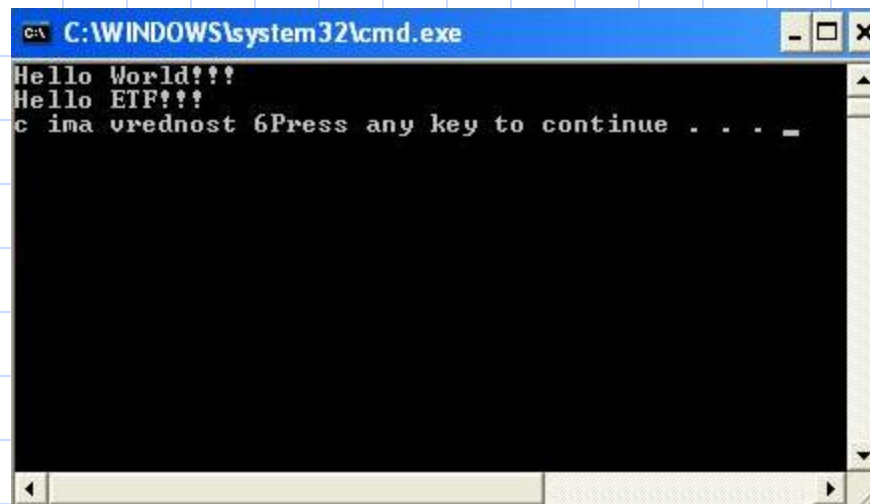
The screenshot shows the 'Output' window in Visual Studio. The 'Show output from:' dropdown is set to 'Build'. The output text is as follows:

```
----- Build started: Project: HelloProject, Configuration: Debug Win32 -----  
Compiling...  
HelloWorld.c  
Compiling manifest to resources...  
Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0  
Copyright (C) Microsoft Corporation. All rights reserved.  
Linking...  
Embedding manifest...  
Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0  
Copyright (C) Microsoft Corporation. All rights reserved.  
Build log was saved at "file:///c:/Documents and Settings/Predrag Radenkovic/My Documents/Visual Studio  
HelloProject - 0 error(s), 0 warning(s)  
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

At the bottom of the window, there is a tab bar with four tabs: 'Code Definition Window', 'Call Browser', 'Output' (which is selected), and 'Find Symbol Results'.

- ◆ Nakon pokretanja opcije "Build" pojavljuje se tekst koji sadrži izveštaj (engl. *log*) o izvršenim akcijama i eventualnim pronađenim greškama u izvornom kodu ili greškama prilikom povezivanja datoteka unutar projekta.
- ◆ U ovom primeru *log* prikazuje da nije bilo grešaka (engl. *errors*) niti upozorenja (engl. *warnings*).

Pokretanje aplikacije



```
C:\WINDOWS\system32\cmd.exe
Hello World!!!
Hello ETF!!!
c ima vrednost 6Press any key to continue . . . _
```

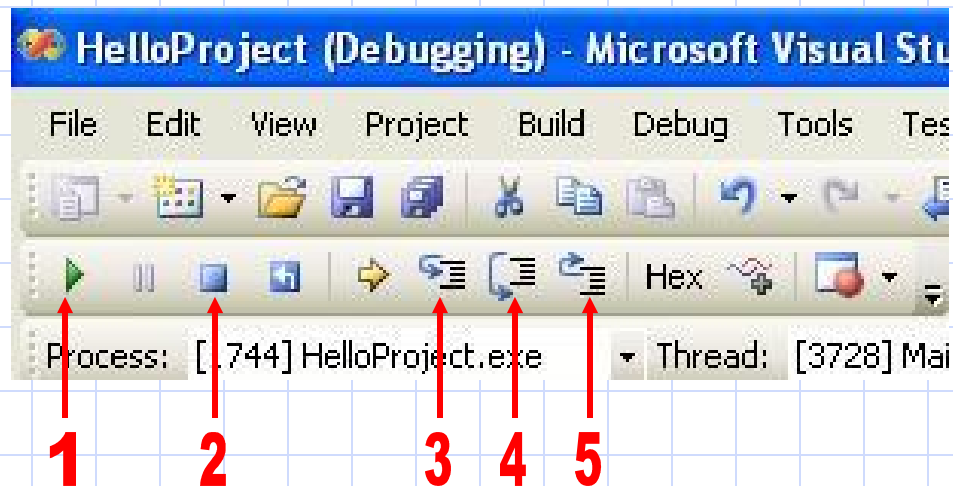
◆ Pokretanje primera "HelloWorld":

- Provera da li je fajl snimljen
- Build
- Execute

◆ Program se pokreće u MS-DOS konzoli.

◆ Program se može samostalno pokretati, pokretanjem fajla "HelloProject.exe".

Korišćenje *Debugger* alata [1/4]



- ◆ 1 – Start: pokretanje programa u *debug* režimu
- ◆ 2 – Stop Debugging: prekid *debug* režima
- ◆ 3 – Step Into: ulazak u sledeću funkciju u liniji na koju pokazuje žuta strelica, odnosno izvršavanje proste naredbe ako nema funkcije u pokazanoj liniji
- ◆ 4 – Step Over: izvršavanje linije na koju pokazuje žuta strelica, bez ulaska u funkciju (ako postoji)
- ◆ 5 – Step Out: izvršavanje tekuće funkcije do kraja i izlazak u pozivajuću funkciju

Korišćenje *Debugger* alata [2/4]

- ◆ U trenutku zaustavljanja programa i aktiviranja *Debugger* alata prikazuju se četiri dodatna prozora: "Watch", "Autos", "Locals" i "Call Stack".
- ◆ "Watch" prikazuje vrednosti promenljivih koje programer odabere i prebaci u listu za praćenje vrednosti (engl. *watchlist*); promenljiva koja se prati ne mora biti deo koda u kojem se aktivirao *Debugger*.

Korišćenje *Debugger* alata [3/4]

- ◆ "Autos" prikazuje trenutni kontekst izvršavanja, tj. vrednosti svih promenljivih koje su aktivne u tom delu koda.
- ◆ "Locals" prikazuje vrednosti svih lokalnih promenljivih u datom trenutku.
- ◆ "Call Stack" prikazuje redosled poziva svih funkcija koje su pozvane da bi se došlo do trenutne funkcije (više informacija [ovde](#))

Korišćenje *Debugger* alata [4/4]

- ◆ Dvostruki klik na funkciju u "Call Stack" će prikazati kontekst vezan za tu funkciju (izvorni kod u editoru programskog koda, vrednosti lokalnih promenljivih u "Locals")
- ◆ Ako je funkcija izabrana u "Call Stack" različita od one u kojoj je zaustavljeno izvršavanje ("u kojoj je žuta strelica"), mesto dokle je program stigao (sa gledišta tekuće funkcije) je označeno zakrivljenom zelenom strelicom

hello world (Debugging) - Microsoft Visual Studio

File Edit View VAssistX Project Build Debug Tools Test Window Help

Process: [0xDB4] hello world.exe Thread: [0xFF8] Main Thread Stack Frame: hello world.exe!main() Line 11

Solution Explorer - Sol... X

Solution 'hello world' (1 project)

- hello world
 - Header Files
 - Resource Files
 - Source Files
 - sabirac.c

sabirac.c*

(Global Scope) main()

```
#include <stdio.h>

int main()
{
    int deljenik, delilac, kolicnik, ostatak;

    printf("Unesite deljenik i delilac: |");
    scanf("%d%d", &deljenik, &delilac);
    if (0 != delilac)
    {
        kolicnik = deljenik / delilac;
        ostatak = deljenik % delilac;
        printf("Kolicnik: %d\nOstatak: %d\n");
    }
    return (0 == delilac);
}
```

Locals

Name	Value	Type
kolicnik	0xffffffff	int
ostatak	0xffffffff	int
deljenik	0x00000017	int
delilac	0x00000005	int

Autos

Name	Value	Type
delilac	0x00000005	int
deljenik	0x00000017	int
kolicnik	0xffffffff	int

Watch 1

Name	Value	Type
deljenik / delilac	0x00000004	int
deljenik % delilac	0x00000003	int
deljenik >> 1	0x0000000b	int
deljenik >> 2	0x00000005	int
deljenik >> 3	0x00000002	int
deljenik >> 4	0x00000001	int
deljenik >> 5	0x00000000	int

Call Stack

Name	Language
hello world.exe!main() Line 11	C
hello world.exe!__tmainCRTStartup() Line 586 + 0x19 bytes	C
hello world.exe!mainCRTStartup() Line 403	C
kernel32.dll!7c817067()	
[Frames below may be incorrect and/or missing, no symbols loaded]	

mesto u kodu u kojem je program zaustavljen

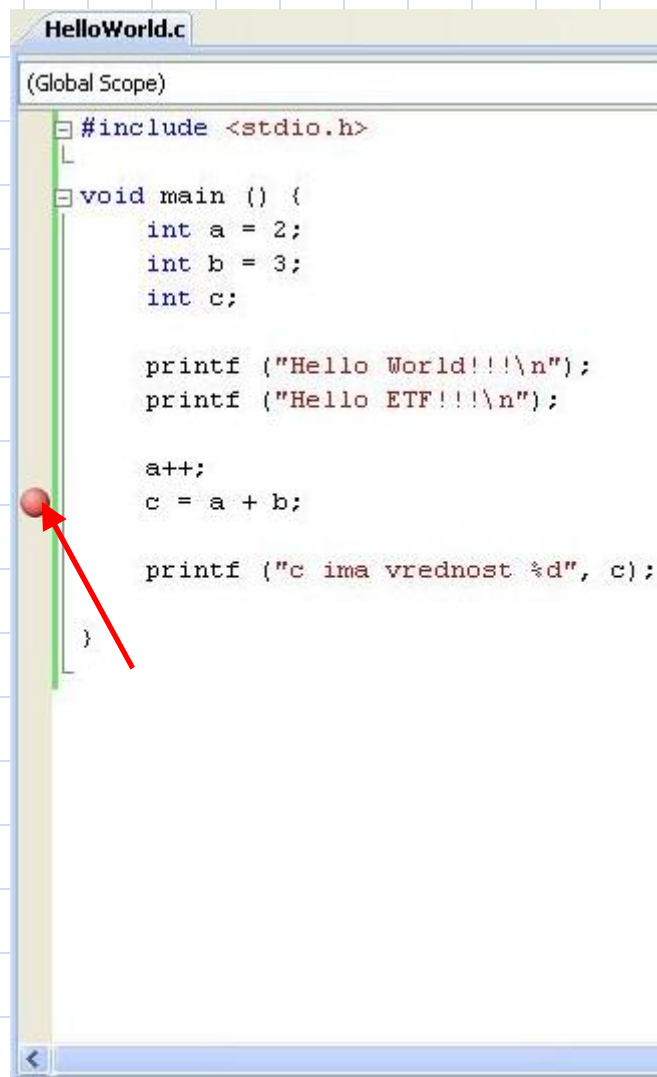
autos

watch

locals

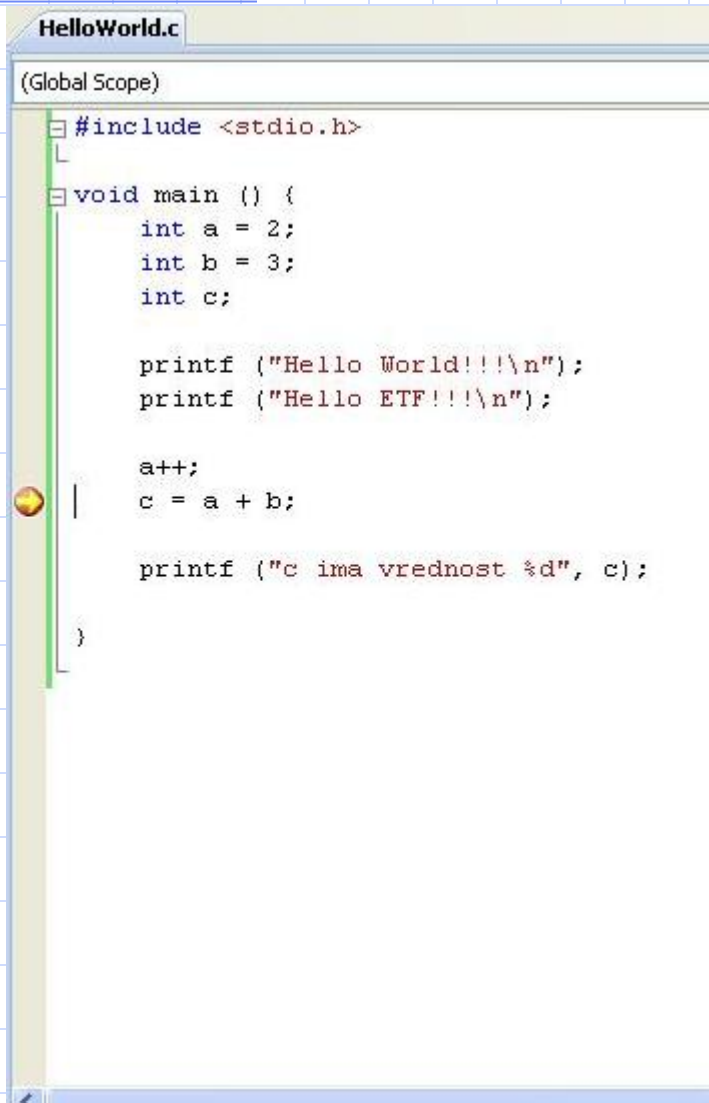
Call Stack

Breakpoint [1/2]



- ◆ Kursor se pozicionira u liniji u kojoj želimo da se program zaustavi.
- ◆ Pritisne se "Toggle Breakpoint" dugme iz "Debug" menija (ili F9)
- ◆ Ili pritiskom na levi taster miša uz levu marginu koda
- ◆ Uz levu marginu koda MSVC sam dodaje marker kako bi naznačio da je tu tačka prekida.

Breakpoint [2/2]



```

HelloWorld.c
(Global Scope)
#include <stdio.h>

void main () {
    int a = 2;
    int b = 3;
    int c;

    printf ("Hello World!!!\n");
    printf ("Hello ETF!!!\n");

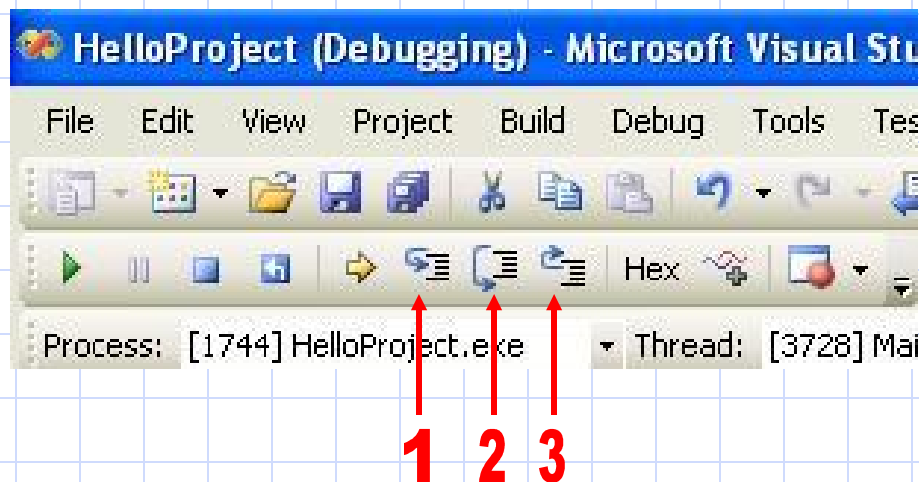
    a++;
    | c = a + b;
    printf ("c ima vrednost %d", c);
}

```

The screenshot shows a code editor window titled 'HelloWorld.c' with '(Global Scope)' indicated. The code is a C program that prints 'Hello World!!!' and 'Hello ETF!!!', increments a variable 'a', and calculates 'c = a + b'. A yellow circular breakpoint icon is placed on the line 'c = a + b;'. The code is enclosed in a light green border.

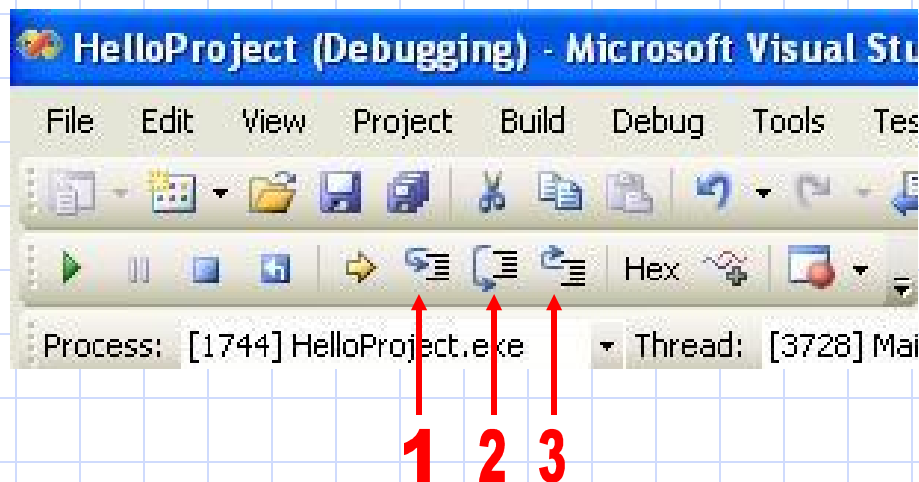
- ◆ Program se pokreće pritiskom "Start Debugging" dugmeta iz "Debug" menija (ili F5).
- ◆ Program se izvršava do nailaska na tačku prekida, gde se zaustavlja.

Korišćenje "Step By Step" [1/2]



- ◆ Program se pokreće pritiskom na "Step Into" dugme (1) iz "Debug" menija (ili Ctrl+F10), ili pritiskom na "Step Over" dugme (2) (ili F10).
- ◆ "Step Into" je izvršavanje korak po korak, sa ulaskom u kod funkcija koje se pozivaju, gde se izvršavanje identično nastavlja.

Korišćenje "Step By Step" [2/2]



- ◆ "Step Over" je izvršavanje korak po korak, bez ulaska u kod pozivanih funkcija.
- ◆ "Step Out" (3) izvršava do kraja kod funkcije u kojoj se nalazi i vraća se u funkciju koja ju je pozvala.

Razlike između C i C-unutar-C++

- ◆ Veliki broj C rešenja je u C++ zamenjen drugim pristupom
- ◆ Ipak, skoro sva C rešenja je moguće koristiti u C++ izvornom kodu
- ◆ Iako se skoro cela sintaksa poklapa, postoje manje razlike u onome što je u jeziku C++ nasleđeno od jezika C
- ◆ Navedene razlike su opisane [ovde](#)

Razlike između C i C++ fajlova

- ◆ U MSVC je moguće pisati izvorni kod u sintaksi C jezika, bez obzira na postojeće razlike između C i C++.
- ◆ Datoteke koje sadrže C izvorni kod imaju ekstenziju `.c`, ali u MSVC mogu imati i ekstenziju `.cpp` rezervisanu za datoteke koje sadrže C++ izvorni kod.

Podrazumevano:

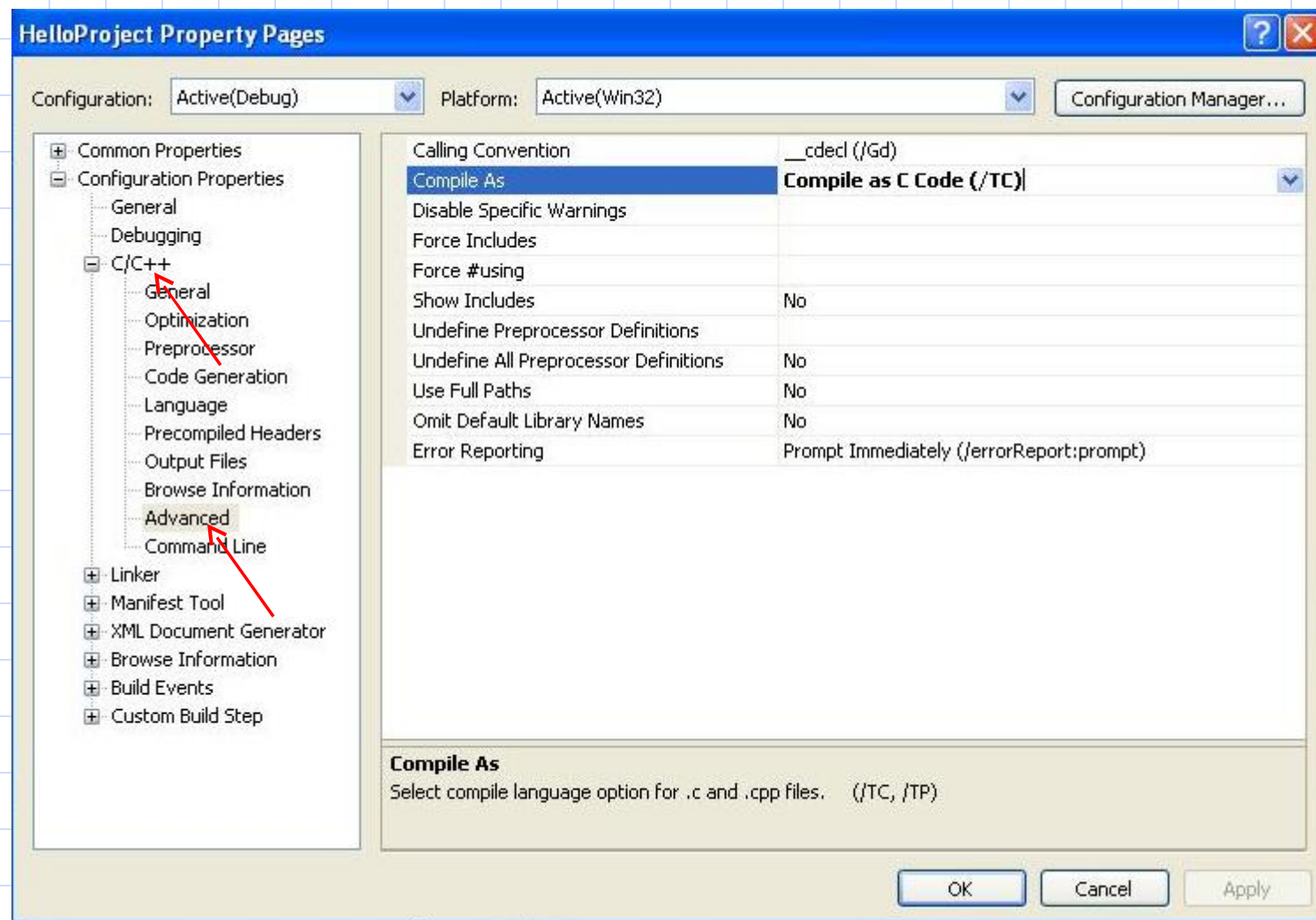
`.c = C`

`.cpp = C++`

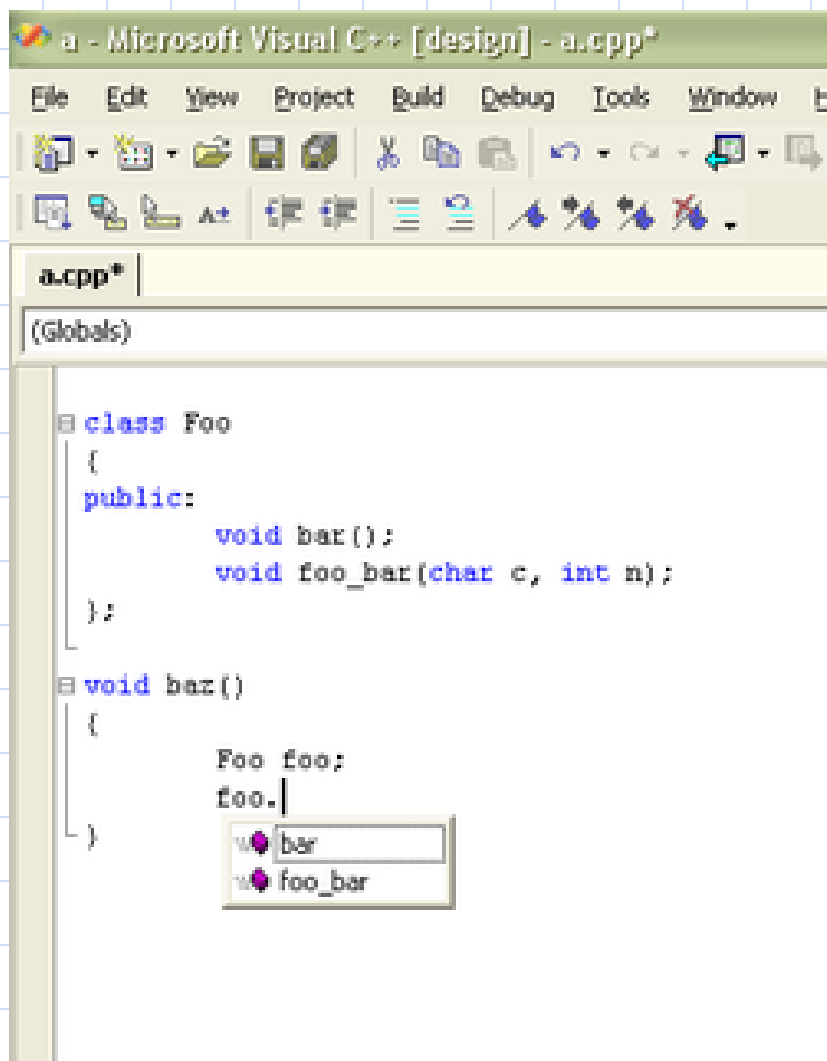
Prevođenje prema ekstenziji [1/2]

- ◆ Podrazumevano, prevodilac bira jezik kojim će prevesti fajl prema ekstenziji fajla
- ◆ U VS 2008 može se eksplicitno zadati način kako da prevodilac prevede dati projekat
- ◆ Desnim klikom na projekat u Solution Explorer i pritiskom na stavku Properties otvara se dijalog u kome se bira stavka C/C++ i u njemu Advanced
- ◆ Tada se u dijalogu može izabrati jezik prevođenja (Compile as C Code (/TC))

Prevođenje prema ekstenziji [2/2]



Intellisense



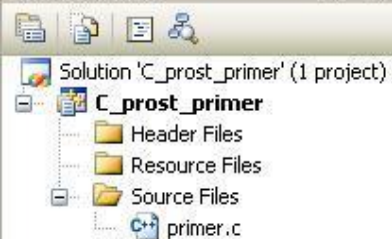
- ◆ Intellisense je način na koji je u VS ostvarena autocomplete funkcionalnost.
- ◆ To je način da se lakše pristupi imenima identifikatora, opisima funkcija i njihovim listama argumenata.

Jednostavan C program

- ◆ C primer, koji će biti pokazan, sastoji se samo od jedne izvorne datoteke.
- ◆ Program zahteva od korisnika da unese 3 elementa niza a zatim taj niz ispisuje.



Solution Explorer



primer.c

(Global Scope)

main()

```
#include <stdio.h>

int main ()
{
    int niz[3]; //niz od 3 promenljive
    int i;

    /*ucitavanje pocetnih vrednosti */
    for (i=0; i<3; i++)
    {
        printf ("Unesite vrednost");
        printf (" za promenljivu niz[%d] ", i+1);
        scanf ("%d", &niz[i]);
    }

    for (i=0; i<3; i++)
    {
        printf ("Vrednost niz[%d] = %d\n", i+1, niz[i]);
    }

    return 0;
}
```

Solution... Class Vi... Propert...

Output

Show output from: Build

```
C_prost_primer - 0 error(s), 1 warning(s)
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Pretprocesor [1/2]

- ◆ Pretprocesor jezika C je deo prevodioca koji vrši pripremnu obradu izvornog teksta programa tj. vrši razne transformacije teksta kojima se stvara konacni oblik teksta koji treba da bude preveden.
- ◆ Na ovaj način programeru je omogućeno da uslovno prevodi neki kod, da specificira poruke za greške u vreme prevođenja i još mnogo toga.

Pretprocesor [2/2]

- ◆ Radom pretprocesora upravlja se specijalnim naredbama koje se nazivaju direktive pretprocesora.
- ◆ Direktive pretprocesora pišu se u zasebnim linijama i počinju zankom #.

Direktive define i undef [1/2]

- ◆ Pretprocesorskim direktivama omogućeno je da se svaka pojava nekog identifikatora u kodu zameni nizom leksičkih simbola.
- ◆ Definisanje identifikatora omogućeno je direktivom **#define**. Posle direktive **#define** navodi se ime identifikatora (po pravilu se ime zadaje velikim slovima) i niz leksičkih simbola kojima pretprocesor posle ove direktive menja svako pojavljivanje definisanog identifikatora.

Direktive define i undef [2/2]

- ◆ Direktivom **#undef** se uklanja trenutna definicija identifikatora.
- ◆ Obično se direktive **#define** i **#undef** uparuju da se da bi se kreirao određeni deo koda programa u kome definisani identifikator ima neko posebno značenje.
- ◆ Ukoliko u direktivi **#define** nakon imena identifikatora nije naveden simbol tada se takav identifikator zamenjuje praznim tekstom.

Uslovno prevođenje [1/2]

- ◆ Generalna ideja uslovnog prevođenja je da se delovi koda mogu selektivno prevoditi u zavisnosti od toga da li je određen identifikator definisan ili ne.
- ◆ Uslovno prevođenje počinje direktivama **#if**, **#ifdef** i **#ifndef**.
- ◆ Iza direktive **#if** se navodi konstantan izraz i ako je taj izraz istinit redovi izvornog koda do sledeće uslovne direktive se prevode, u suprotnom se ne prevode.

Uslovno prevođenje [2/2]

- ◆ Uslovno prevođenje vrlo često zavisi samo od postojanja ili ne postojanja nekog identifikatora. U tim slučajevima se koriste direktive **#ifdef** i **#ifndef**.
- ◆ Posle početka uslovnog prevođenja mogu se koristiti i direktive **#elif** i **#else**.
- ◆ Uslovno prevođenje se završava direktivom **#endif**.

Sprečavanje višestrukog prevođenja [1/2]

- ◆ Višesturko prevođenje jedne datoteke zaglavlja može se sprečiti uslovnim prevođenjem ako se na početku te datoteke definiše identifikator sa imenom koje odgovara imenu fajla.

Na primer u datoteci akcije.h sprečavanje višestrukog prevođenja postiže se ako se kod obaviye direktivama:

```
#ifndef _akcije_h_  
#define _akcije_h_  
  
#endif
```

Sprečavanje višestrukog prevođenja [2/2]

```
≡ /*
    kad god se u .h definisu neki tipovi podataka,
    poželjno je na pocetak staviti uslovno prevodjenje,
    takvo da uslovljava prevodjenje celog sadrzaja.

    na ovaj nacin se izbegava greska redefinisavanja tipa,
    koja se moze javiti ako se .h sa definicijama tipova
    kroz razne #include ukljuci vise puta unutar istog .c
    */

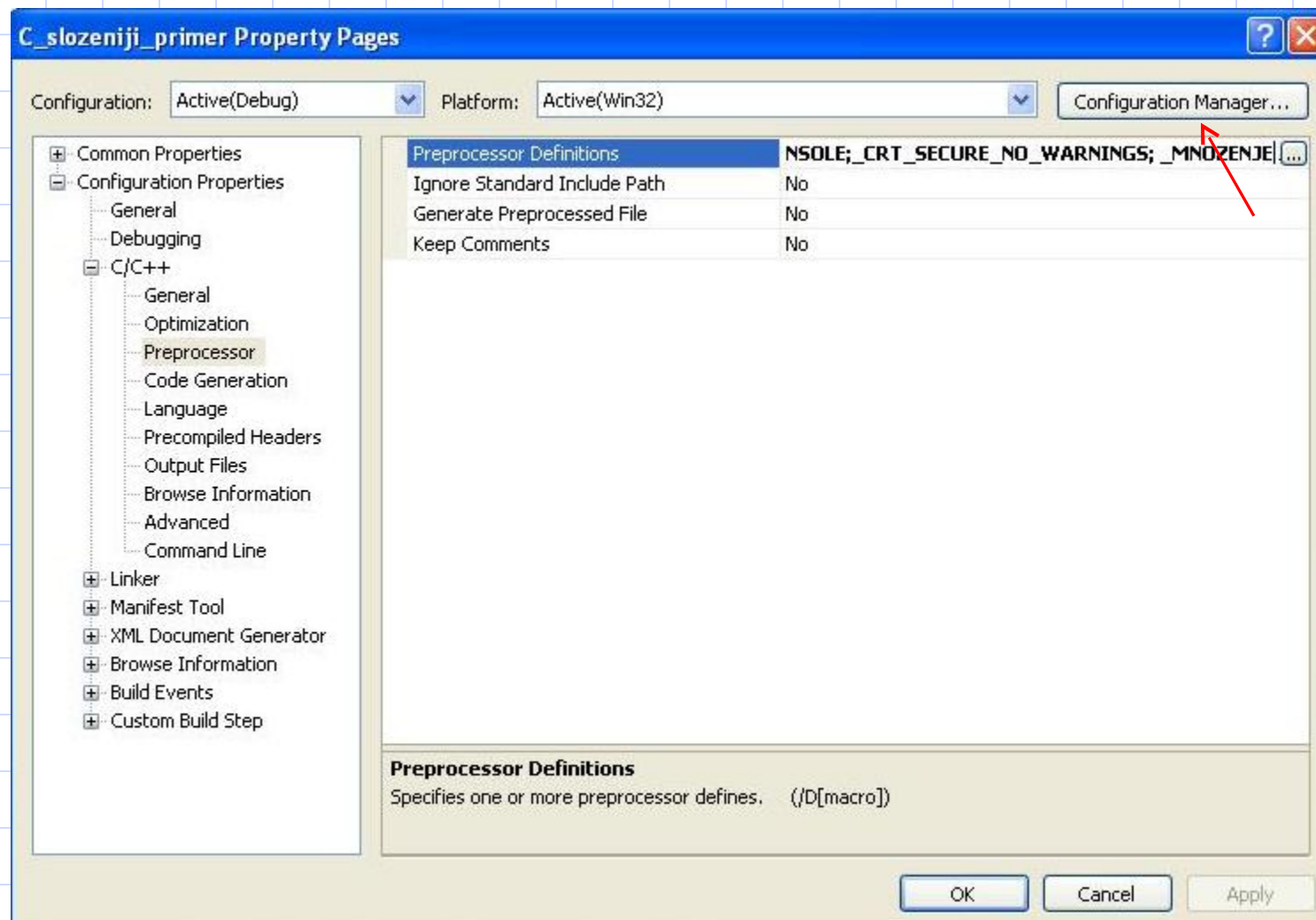
/* ako nije definisana simbolicka konstanta */
≡ #ifndef __C_PRIMER__
    /*
       definisemo tu simbolicku konstantu,
       ime izabrati tako da ima veze sa namenom datog .h
       i da se razlikuje od ostalih u programskom sistemu
    */
    #define __C_PRIMER__

    /*
       nakon definisanja konstante,
       navodimo sve to treba navesti u datom .h
    */
    */
```

Definisanje identifikatora pretprocesora [1/2]

- ◆ U VS pretprocesoru se mogu definisati identifikatori koje on koristi u uslovnom prevođenju u dijalogu do koga se dolazi desnim klikom na projekat u Solution Explorer. Zatim se biraju Properties, Configuration Properties, C/C++, Preprocessor, respektivno. U Pretprocesor Definitions upisuje se ime identifikatora koji se koristi u uslovnom prevođenju.

Definisanje identifikatora pretprocesora [1/2]



Uslovno prevođenje

```
/* ukoliko je u projektu pretprocesoru definisan identifikator _MNOZENJE_
   program ce prevesti i deo gde se poziva funkcija pomnozi,
   u suprotnom prevесce se samo deo koda koji poziva funkciju saberi i
   krajnji rezultat koji se ispisuje na standardni izlaz бice zbir promenljivih x i y.
*/
    rezultat = saberi(x,y);

#ifdef _MNOZENJE_
    pomnozi(y, rezultat, &rezultat);
#endif

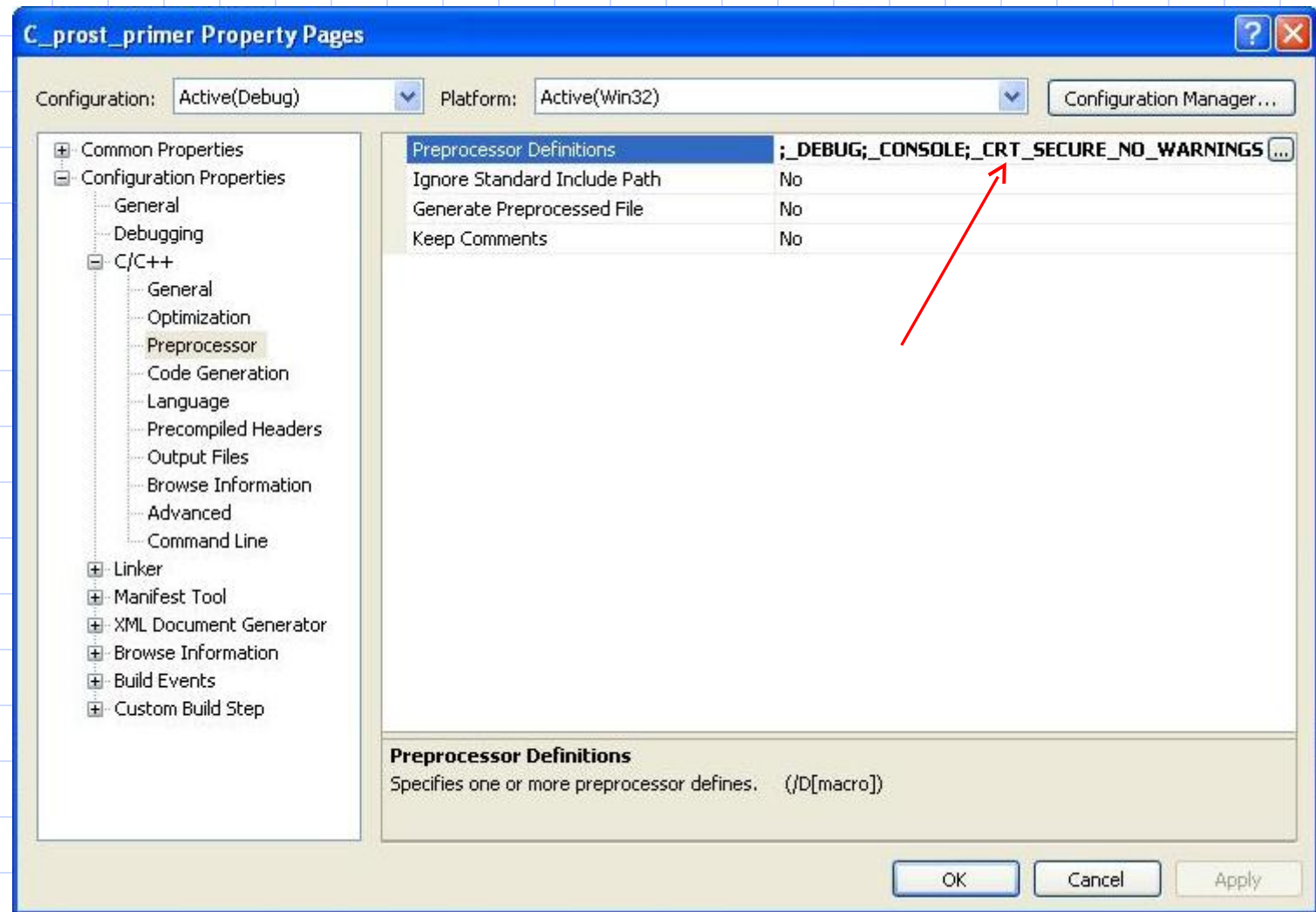
    ispisi(rezultat);
}
```

- ◆ Ukoliko se pretprocesoru definiše identifikator `_MNOZENJE_` prevodilac će prevesti i poziv funkcije `pomnozi()`. U suprotnom deo koda koji se nalazi između direktiva `#ifdef` i `#endif` neće biti preveden.
- ◆ Uslovno prevođenje i zaštita od višestrukog prevođenja detaljno su opisani u projektu C složeniji primer.

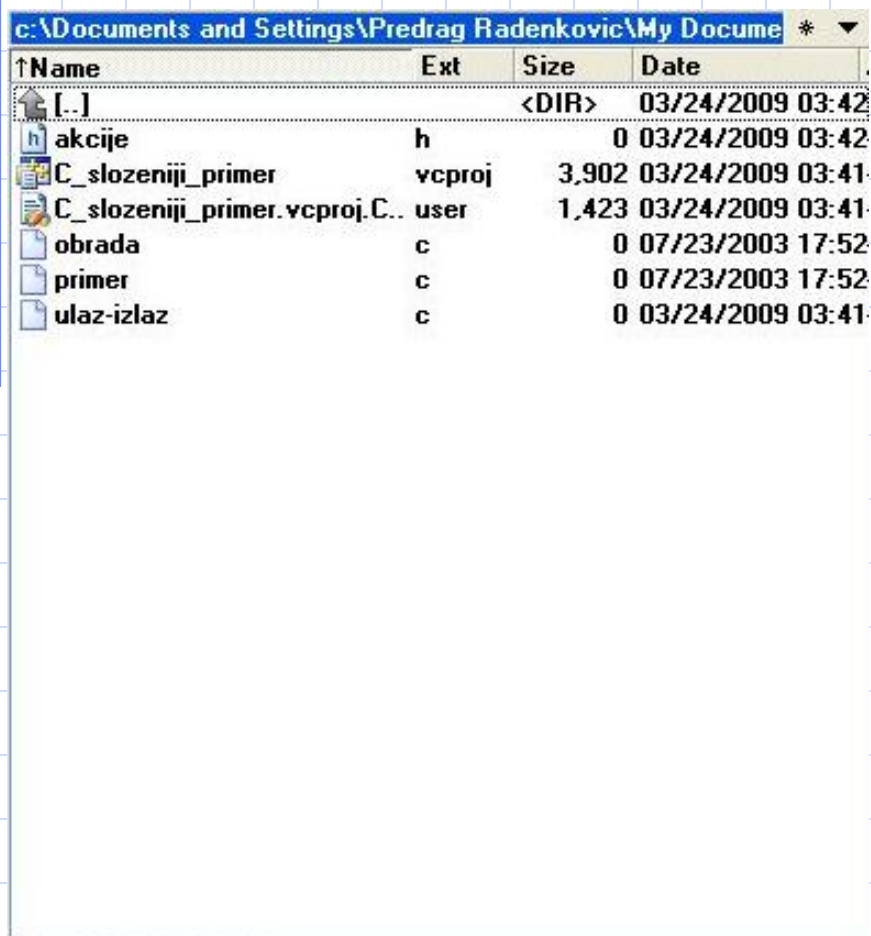
Jedno podešavanje VS okruženja [1/2]

- ◆ Visual Studio okruženje ima specifično ponašanje da upozori korisnika da koristi potencijalno opasnu funkciju (npr. prilikom korišćenja C funkcija `scanf()` i `printf()`)
- ◆ Upozorenja ovog tipa mogu se isključiti tako što se u definicije pretprocesora dodaje simbol `_CRT_SECURE_NO_WARNINGS`.

Jedno podešavanje VS okruženja [2/2]



Složeniji C primer [1/4]



Name	Ext	Size	Date
[.]	<DIR>		03/24/2009 03:42
akcije	h	0	03/24/2009 03:42
C_slozeniji_primer	vcproj	3,902	03/24/2009 03:41
C_slozeniji_primer.vcproj.C..	user	1,423	03/24/2009 03:41
obrada	c	0	07/23/2003 17:52
primer	c	0	07/23/2003 17:52
ulaz-izlaz	c	0	03/24/2009 03:41

- ◆ Projekat "c_primer", koji se nalazi uz tutorijal, sadrži 4 fajla sa izvornim kodom (3 *source* i 1 *header*).
- ◆ U direktorijumu projekta nalaze se i fajlovi koje MSVC automatski kreira, najbitniji su:
 - .vcproj – fajl koji sadrži podešavanja projekta
 - .sln – fajl koji sadrži podešavanja za *Solution*

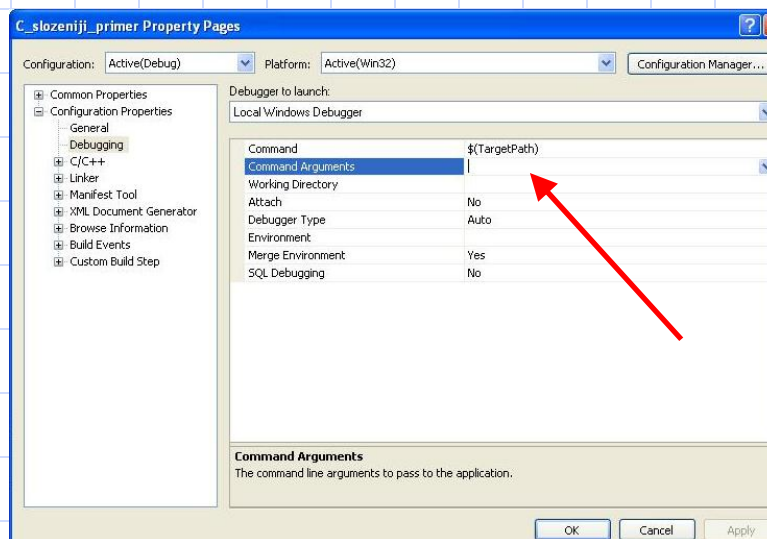
Složeniji C primer [2/4]

- ◆ U *header* fajlu `akcije.h` nalaze se definicije funkcija koje sadrže *source* fajlovi `obrada.c` i `ulaz-izlaz.c`.
- ◆ *Header* fajl može da sadrži i definicije simboličkih konstanti.

Složeniji C primer [3/4]

- ◆ Izvršavanje programa
će krenuti od tačke u programskom kodu
gde se nalazi `main` funkcija.
- ◆ Tok programa:
 - Naredbom `#include "akcije.h"`
fajlu `c_primer.c`
je omogućeno korišćenje prototipova(zaglavlja)
funkcija `ispisi`, `ucitaj`, `saberi` i `pomnozi`.
 - Na kraju program u zavisnosti do toga da li je
definisan identifikator `_MNOZENJE_` u definicijama
pretprocesora ispisuje vrednost:
 $\text{rezultat} = y * (x + y)$ ili $\text{rezultat} = x + y$.

Složeniji C primer [4/4]

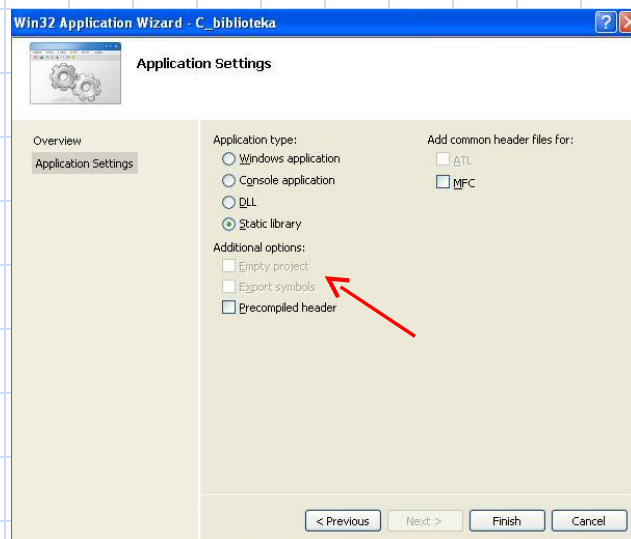


- ◆ Podešavanja argumenata komandne linije vrši se u dijalogu "Property Pages" (meni Project, pa opcija Properties).
- ◆ Podaci uneti u kontroli "Command Arguments" će biti prosleđeni glavnom programu kao argumenti komandne linije

Kreiranje biblioteke [1/2]

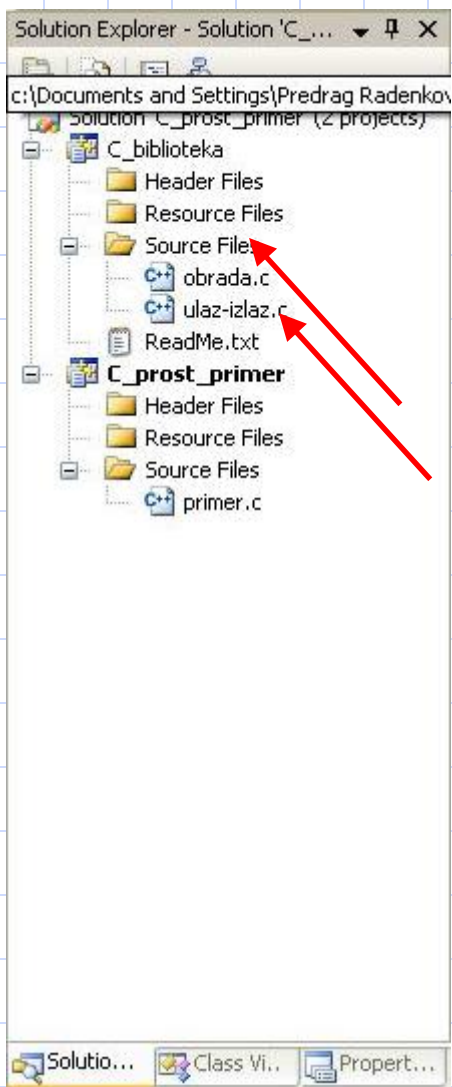
- ◆ Za izradu statičke biblioteke (.lib fajl) koriste se *source* fajlovi iz projekta "C primer".

Kreiranje biblioteke [2/2]



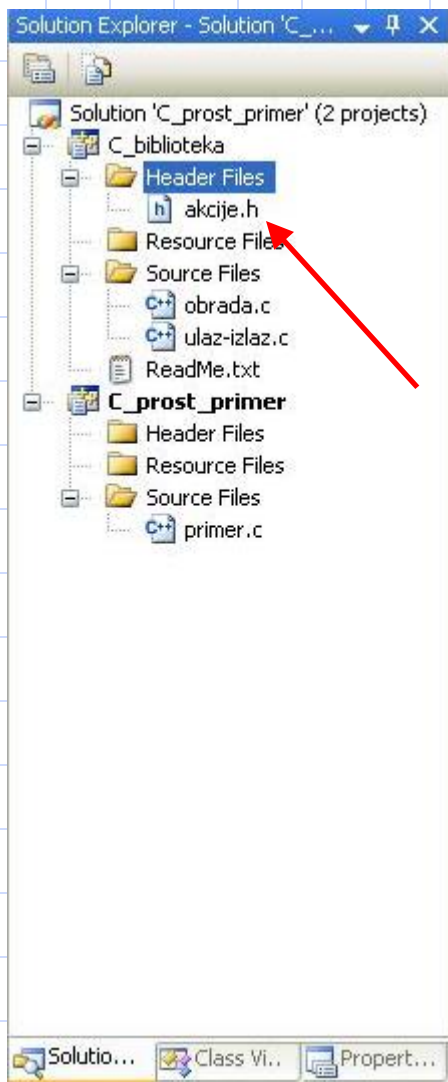
- ◆ Dvoklikom na fajl `c_primer.sln` otvara se *solution* koji sadrži projekat "C primer".
- ◆ Desni klik na projekat i u meniu "Add" dodati novi projekat
- ◆ U dijalogu odabrati opciju "Static Library".
- ◆ U polje "Project name" upisati "c_biblioteka".
- ◆ Na taj način u postojeći *solution* biće dodat novi projekat.

Dodavanje fajlova u biblioteku [1/2]



- ◆ Desnim klikom na folder "Source Files" u okviru projekta "c_biblioteka", otvara se meni gde se odabira opcija "Add" i izabere se opcija "Existing Item".
- ◆ Lociraju se fajlovi obrada.c i ulaz-izlaz.c, dodaju se klikom na "OK".

Dodavanje fajlova u biblioteku [2/2]



- ◆ Na identičan način dodaje se u folder "Header Files" fajl `akcije.h`.
- ◆ *Header* fajl je neophodan kako bi se biblioteka mogla koristiti u drugim projektima.

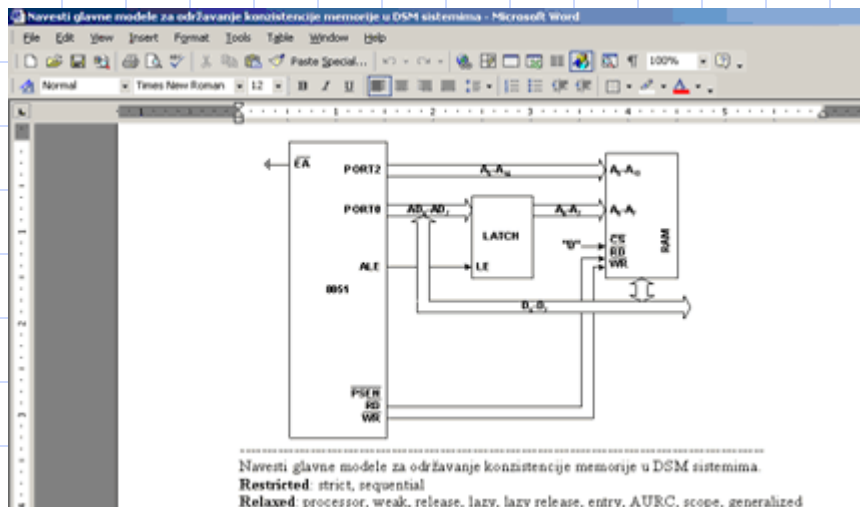
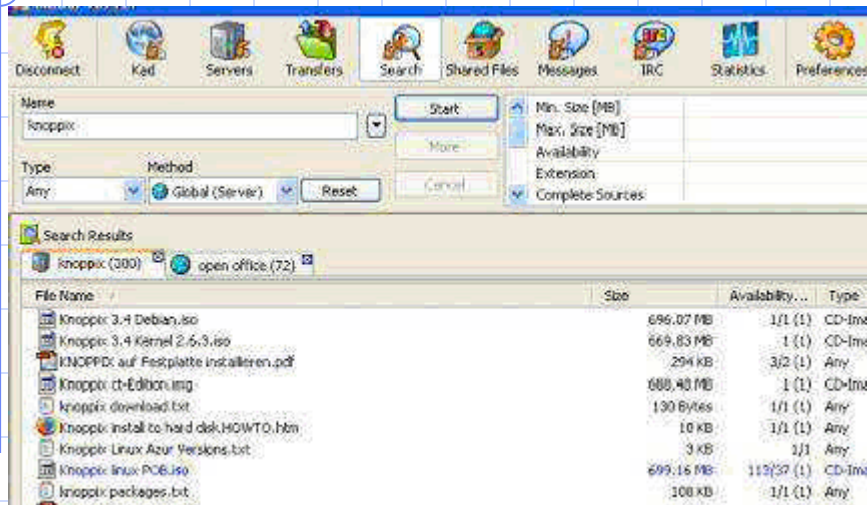
Kreiranje lib fajla

- ◆ Iz menija "Build" odabere se opcija "Build c_biblioteka.lib" (ili F7).
- ◆ Rezultat je fajl `c_biblioteka.lib`, koji se nalazi u direktorijumu `c_primer\c_biblioteka\Debug\` ili `c_primer\c_biblioteka\Release\`, zavisno od aktivne konfiguracije projekta (podešava se u Project Settings)

Praktična primena MSVC

- ◆ Danas je MSVC jedan od najkorišćenijih alata za razvoj komercijalnih aplikacija.
- ◆ Operativni sistem "Microsoft Windows" je dobrim delom razvijen u MSVC.

Aplikacije razvijene u MSVC



Dalje učenje

- ◆ Sve domaće izdavačke kuće (Mikro knjiga, Kompjuter biblioteka...) imaju u svojoj ponudi knjige za osnovne i naprednije kurseve učenja C/C++ jezika.
- ◆ Na Internetu se nalazi veliki broj tutorijala i besplatnih knjiga u elektronskom formatu. Neki od linkova:
 - [link 1](#)
 - [link 2](#)

Internet zajednice

- ◆ Saveti i rešenja za svoje probleme se mogu pronaći i u Internet zajednicama.
- ◆ Zajednice su organizovane ili kao forumi ili kao mailing liste. U prvom slučaju poruke ostaju zapisane na Internet stranama, dok u drugom stižu na mail učesnicima.
- ◆ Neki forumi:
 - [EliteSecurity](#), domaći forum o informatici
 - [Go4expert](#)
 - [Dev Articles](#)

Citat

◆ Proces stvaranja programa za kompjutere je izuzetno atraktivan, ne samo zbog svoje ekonomske i naučne isplativosti, već i zato što može biti umetničko iskustvo nalik komponovanju poezije ili muzike.

- Donald Knuth