



Tema 01 Uvod u objektno orijentisano programiranje i jezik C++

Prof. dr Miodrag Živković

Tehnički fakultet

OBJEKTNO ORIJENTISANO PROGRAMIRANJE 2



Sadržaj

1. Uvod
2. Razvoj i standardizacija jezika C i C++
3. Proces razvoja aplikacija u jeziku C++
4. Sintaksa jezika C++
5. Primer programa



1. Uvod

1. Razvoj programskih jezika
2. Objektno orijentisano programiranje (podsetnik)
3. Razvoj objektno orijentisanih programskih jezika



1.1 Razvoj programskih jezika

- **Programski jezik** je formalni jezik za pisanje programskog koda u obliku niza naredbi koje računar prevodi i izvršava
- Prvi programski jezik s prevodiocem
 - računar Univac i jezik **A1**, prva poslovna primena računara 1951. godine. Autor jezika je Grays Mary Hooper, kasnije autor jezika COBOL
- Programski jezici za velike centralne računare
 - FORTRAN, COBOL, Algol, Pascal
- Programski jezici za mini i mikroračunare
 - C, BASIC , Visual Basic, Turbo Pascal
- Programski jezici za moderne računare u svetskoj mreži
 - C++, Java, C#, PHP, Javascript, R, Python














Vrste programskih jezika














- Programski jezici niskog nivoa
 - mašinski jezici (binarno kodirane instrukcije)
 - asemblerski jezici (simbolički zapis mašinskih instrukcija i adresa)
- Programski jezici visokog nivoa
 - proceduralni programski jezici (za opis algoritama):
C/C++, **Java**, **C#**, **Python**
 - neproceduralni jezici (ne opisuju način, već traženi rezultat):
Prolog, **SQL**
- Proceduralni programski jezici mogu biti
 - klasični : **Pascal**, **C**
 - funkcionalni: **LISP**, **F#**
 - objektno-orijentisani : **C++**, **Java**, **C#**, **Python**

Zašto je potrebno znati jezik C++?

- Jezik C++ je jedan od najpopularnijih programskih jezika za razvoj sistemskog softvera i aplikacija (naredni slajdovi)
- Veliki broj kompanija koristi C++ za razvoj softvera namenjenog za sopstvene potrebe ili tržište (naredni slajdovi)
- Jezik C++ je između mašinski orijentisanih jezika i viših programskih jezika (srednjeg nivoa, *middle-level*)
- C++ je objektno orijentisano proširenje programskog jezika C
- Programi u jeziku C++ su *prenosivi*, jer postoje prevodioci za sve važnije procesore i operativne sisteme
- Jezik C++ je relativno jednostavan jezik (ali ima obimne biblioteke programa)

4.1 Pregled upotrebe programskih jezika opšte namene (IEEE, 2020)

Rank	Language	Type	Score
1	Python▼	  	100.0
2	Java▼	  	95.3
3	C▼	  	94.6
4	C++▼	  	87.0
5	JavaScript▼		79.5
6	R▼		78.6
7	Arduino▼		73.2
8	Go▼	 	73.1
9	Swift▼	 	70.5
10	Matlab▼		68.4

Rank	Language	Type	Score
13	SQL▼		64.6
14	PHP▼		63.8
15	Assembly▼		63.7
16	Scala▼	  	63.5
17	HTML▼		61.4
...			
19	Julia▼		56.0
23	C#▼	   	48.1
36	Prolog▼		34.6

Web  Enterprise  Mobile  Embedded 

<https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020>

Ilustracija: Dužina/obim poznatih programa u LOC (*Lines of Code*)

		400.000
		2.500.000
		65.000.000
• Moderni luksuzni automobili		100.000.000
		
		
		
		
		

Neko od poznatijih programa implemetiranih u jeziku C++

- **Adobe Systems** - *Photoshop, ImageReady, Illustrator, Premier*
- **Google** - *Google file system, Google Chromium, Google Earth, Picasa, Google Desktop Search, MapReduce*
- **Mozilla** - *Firefox, Thunderbird*
- **MySQL** - DBMS
- **Apple** – *OS X, iPod*
- **Microsoft** - *Windows 95, 98, Me, 2000, XP, 8,10, Office, Internet Explorer, Visual Studio*
- **Symbian OS** i brojni ugrađeni sistemi
- **Virtuelne mašine i programski prevodioci** svih proizvođača

1.2 Objektno orijentisano programiranje (podsetnik)

- U **proceduralnom** programiranju, programi se sastoje od funkcija, a podaci se predstavljaju zasebno

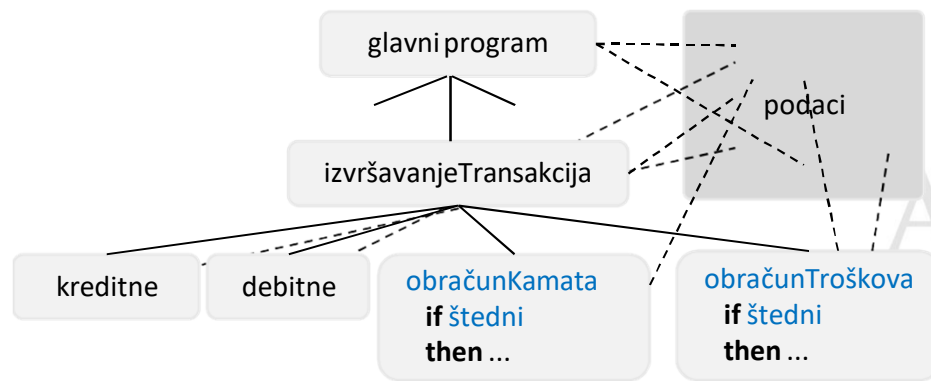
Složene interakcije *programskog koda* i *podataka* iz različitih delova programa takav program čine složenim i teškim za održavanje, pa je za realizaciju složenih softverskih sistema potreban drugačiji pristup

- **Objektno orijentisani pristup** složene softverske sisteme predstavlja kao *skup objekata*, koji se sastoje od *podataka* i *metoda* kojima se vrše operacije nad objektima

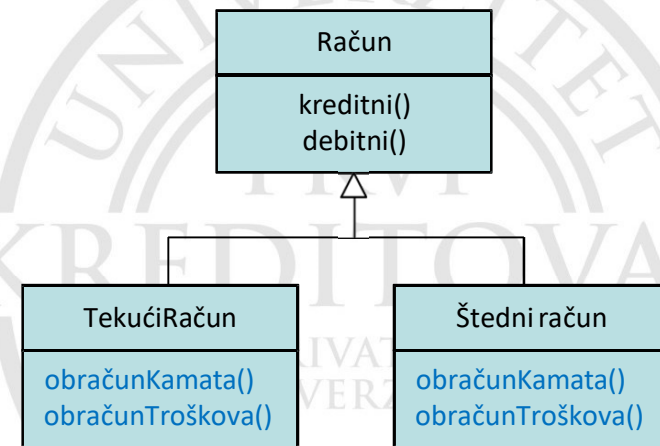
Ovaj pristup organizuje programe na način kako je organizovan stvarni svet, gde su predmeti *u međusobnoj vezi*, kako po atributima, tako i po aktivnostima

Objektno orijentisani razvoj softvera

- Objektno orijentisani pristup rešava mnoge probleme koji su svojstveni proceduralnim programiranju, gde su podaci i operacije su međusobno *razdvojeni*, tako da je neophodno slanje podataka metodima
- Objektno orijentisano programiranje smešta podatke i operacije koje se na njih odnose *zajedno* u objektu



(a) Proceduralna organizacija sistema



(b) Objektna organizacija sistema

Svojstva objektno orijentisanog programiranja i programskih jezika

- **Apstrakcija** (*abstraction*) - mogućnost definisanja novih tipova podataka (klase objekata)
- **Enkapsulacija** (*encapsulation*) - skrivanje detalja realizacije nekog tipa (klase)
- **Nasleđivanje** (*inheritance*) - kreiranje novih tipova (klasa) pomoću postojećih, koji će naslediti sve osobine starih tipova i dodati svoje specifičnosti
- **Polimorfizam** (*polymorphism*) - pojavljivanje tipa (klase) u više oblika, jer se mogu ne samo dodavati svoji elementa, već i menjati nasleđeni

1.3 Razvoj objektno orijentisanih programskih jezika

- Prvi objektno orijentisani jezici
 - Simula, 1967
 - Smalltalk, 1972-1980
- Većina savremenih programskih jezika su objektno orijentisani i mogu biti
 - **objektni** jezici (*class-based*), koji imaju mogućnost definisanja klasa i nasleđivanje, npr. *C++*, *Java*, *C#* i *Python*
 - **objektno zasnovani** (*prototype-based*), s ograničenim objektnim svojstvima (kao što je nasleđivanje), koji pretežno koriste postojeće ugrađene objekte, npr. *JavaScript*
- Primeri jezika koji *nisu* objektno orijentisani su *C*, klasični *Pascal* (ne *Object Pascal*) i ostali stariji proceduralni programski jezici

2. Razvoj i standardizacija jezika C i C++

1. Razvoj i svojstva jezika C
2. Razvoj jezika C++



2.1 Razvoj i svojstva jezika C

- **Proceduralni** programski jezik, čija je prvobitna namena bila lakši razvoj sistemskih programa
 - jezik je razvio američki naučnik Dennis Ritchie, koji je s kolegom Kenom Thompsonom u *Bell Labs* od 1969. godine razvijao operativni sistem za male računare *Unix* (od *Multics*, operativnog sistema velikih računara)
- Za potrebe ubrzanja razvoja sistemskog softvera razvijeni su novi jezici **B** (Thompson) i kasnije **C** (Ritchie)
 - osnovna inspiracija bio je imperativni proceduralni jezik *Algol* (*Algorithmic Language*)
 - iako se prevodio u mašinski jezik, programi su bili prenosivi, pošto su prevodioci implementirani za mnoštvo različitih platformi
- Jezik **C** je standardizovan 1989. godine (ANSI C/ISO)

Ilustracija: Funkcija faktorijel u jezicima Algol 68, Ada i C

Algol 68

```
PROC factorial = (INT upb n)LONG LONG INT:(  
    LONG LONG INT z := 1;  
    FOR n TO upb n DO z *:= n OD;  
    z  
);
```

Ada

```
function Factorial (N : Positive) return  
    Positive is  
    Result : Positive := N;  
    Counter : Natural := N - 1;  
begin  
    for I in reverse 1..Counter loop  
        Result := Result * I;  
    end loop;  
    return Result;  
end Factorial;
```

C

```
int factorial(int n) {  
    int result = 1;  
    for (int i = 1; i <= n; ++i)  
        result *= i;  
    return result;  
}
```


2.2 Razvoj jezika C++

- Danski naučnik Bjarne Stroustrup razvio je jezik 'C s klasama' 1979. godine
- Osnovna namera bila je kreiranje jezika koji će imati osobine jezika visokog nivoa kao *Simula* za razvoj složenih sistema i efikasnost jezika niskog nivoa, kao što je bio *BCPL*, prethodnik jezika *B* i *C*
 - radeći na svojoj doktorskoj tezi, imao je problema s lošim stranama oba programska jezika
- Proširio je popularni jezik *C* objektnim svojstvima jezika *Simula*, kao i elementima jezika kao što su *Ada* i *Algol 68*
- Jezik je promenio naziv u C++ 1983. godine
- Jezik C++ je standardizovan 1998. godine (ISO) →

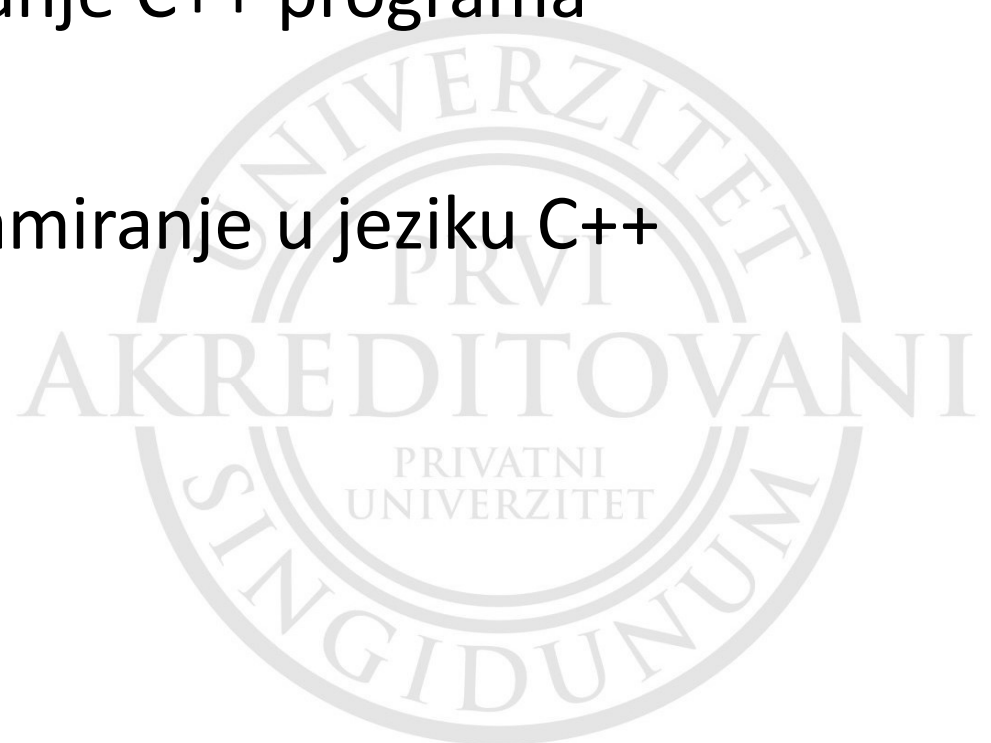
Standardizacija jezika C++

<https://isocpp.org/std>

Godina	ISO standard	Neformalni naziv
1998	ISO/IEC 14882:1998	C++98
2003	ISO/IEC 14882:2003	C++03
2011	ISO/IEC 14882:2011	C++11
2014	ISO/IEC 14882:2014	C++14
2017	ISO/IEC 14882:2017	C++17
2020	ISO/IEC 14882:2020	C++20
2023	<i>nije određen (usvojen plan razvoja)</i>	C++23

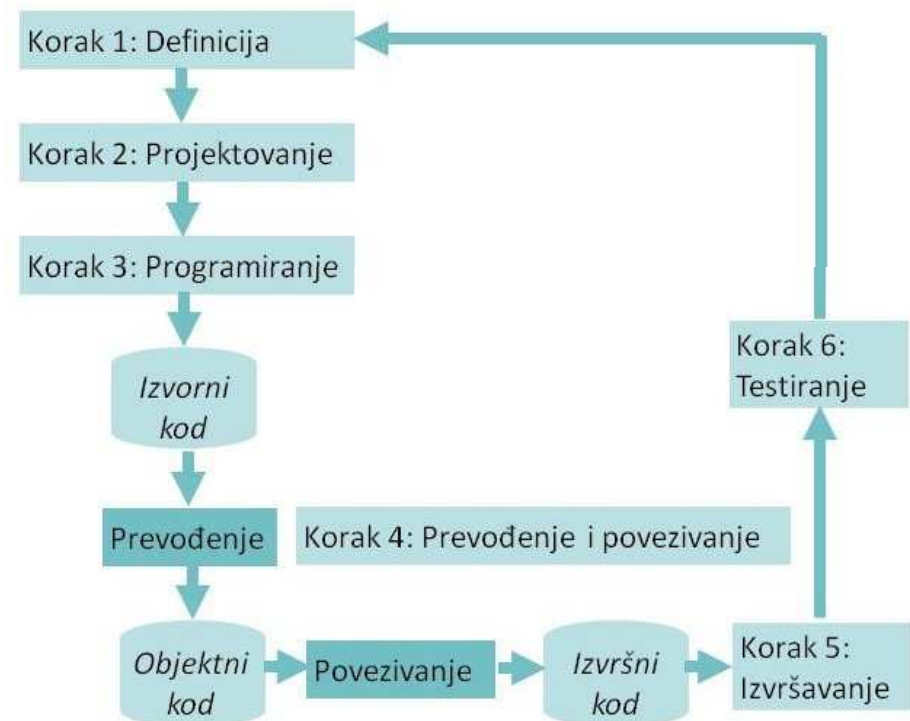
3. Proces razvoja aplikacija u jeziku C++

1. Proces razvoja aplikacija
2. Integrisana razvojna okruženja (IDE)
3. Struktura programa u jeziku C++
4. Unos, prevođenje i izvršavanje C++ programa
5. Pretpocesorske direktive
6. Stilske preporuke za programiranje u jeziku C++



3.1 Proces razvoja aplikacija

- Proces razvoja aplikacija u jeziku C++ obuhvata:
 1. Definisanje problema
 2. Projektovanje rešenja
 3. Programiranje
(pisanje izvornog koda)
 4. Prevođenje u objektni mašinski kod (kompilacija) i **povezivanje** programa
 5. Izvršavanje programa
 6. Testiranje



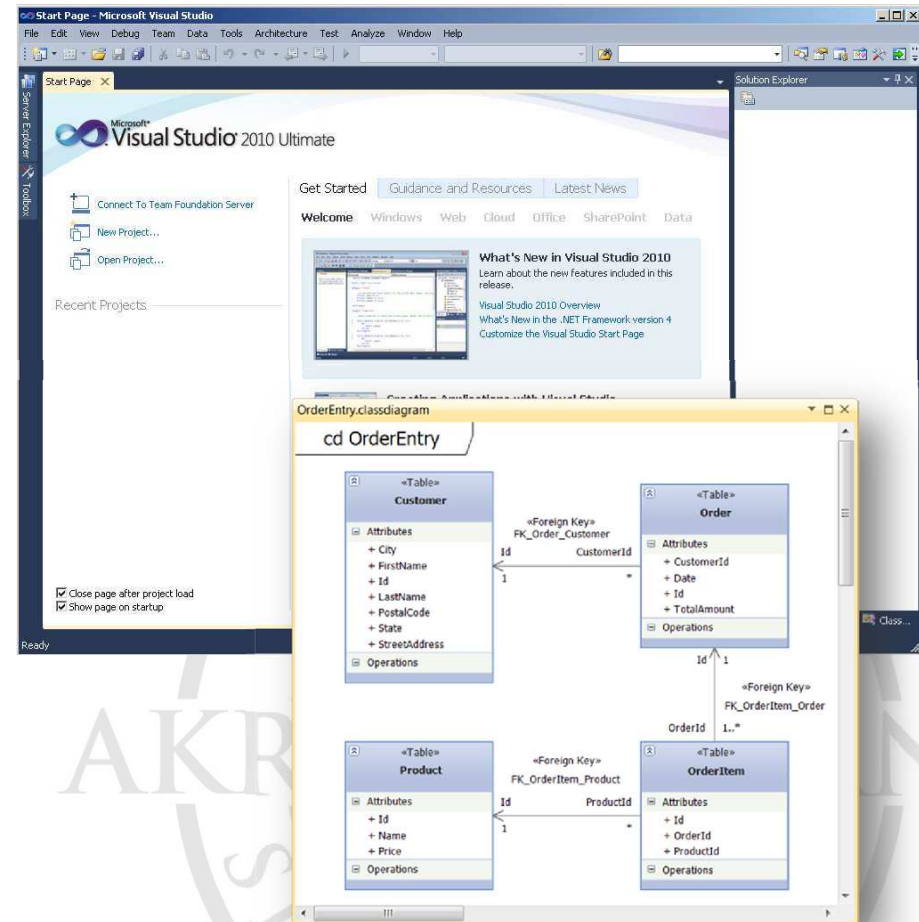
3.2 Integrisana razvojna okruženja (IDE)

- Integrisan programska okruženja povezuju više različitih alata za razvoj softvera pomoću jedinstvenog interfejsa
- Postoji veliki broj integrisanih okruženja za razvoj programa za različite operative sisteme, npr.
 - Microsoft [Visual Studio/Visual Studio Community/Visual Studio Code](#)
 - Eclipse CDT
 - NetBeans C++ IDE
 - C++Builder
 - Code::Blocks
 - CodeLight
 - KDevelop
 - Qt Creator



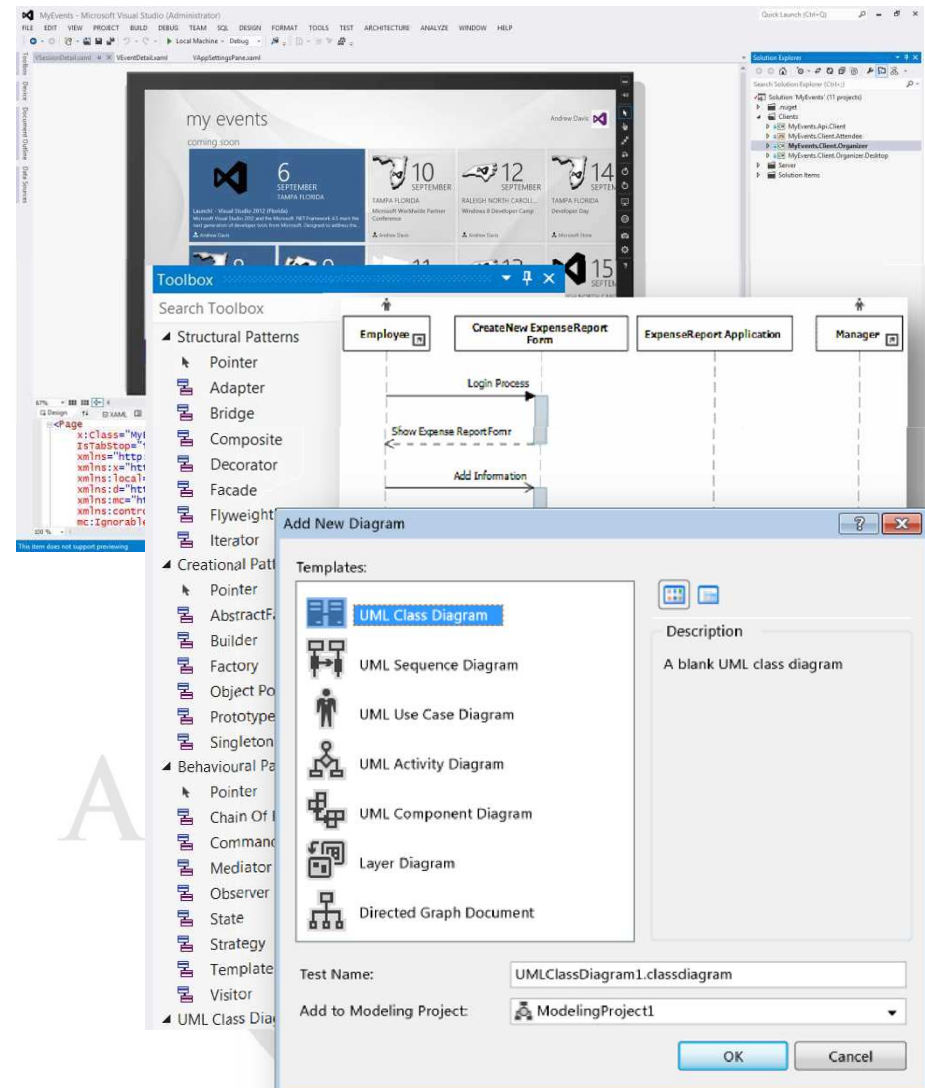
Microsoft Visual Studio 2010

- Integrisani alat, koji podržava sve faze razvoja složenih sistema
- podržava timski rad, verzija Ultimate podržava UML modelovanje (6 dijagrama)
- podržava više tehnologija i arhitektura (C++, C#, Java, Veb servisi, SOA)
 - C++03
 - C++11 delimično

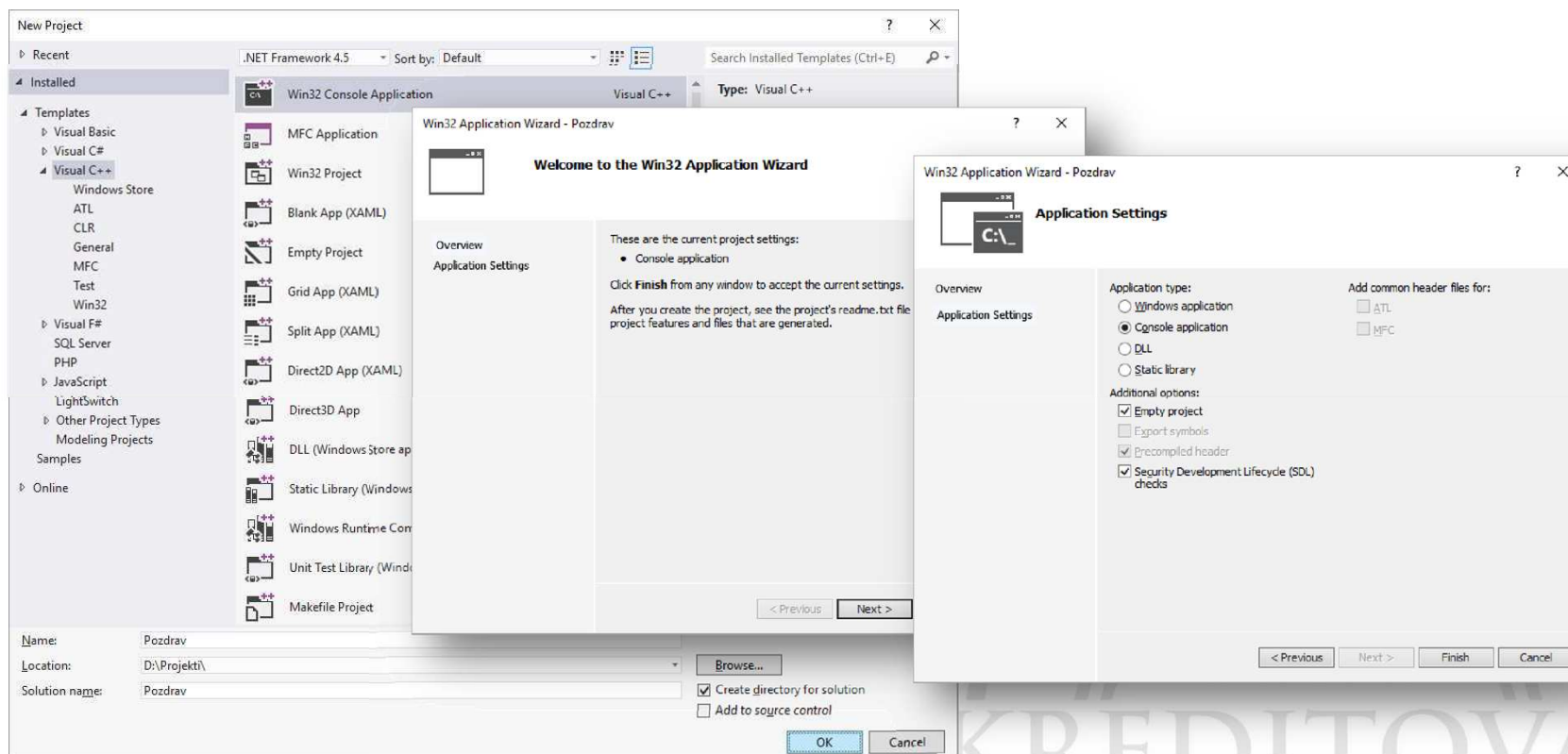


Microsoft Visual Studio 2012-2019

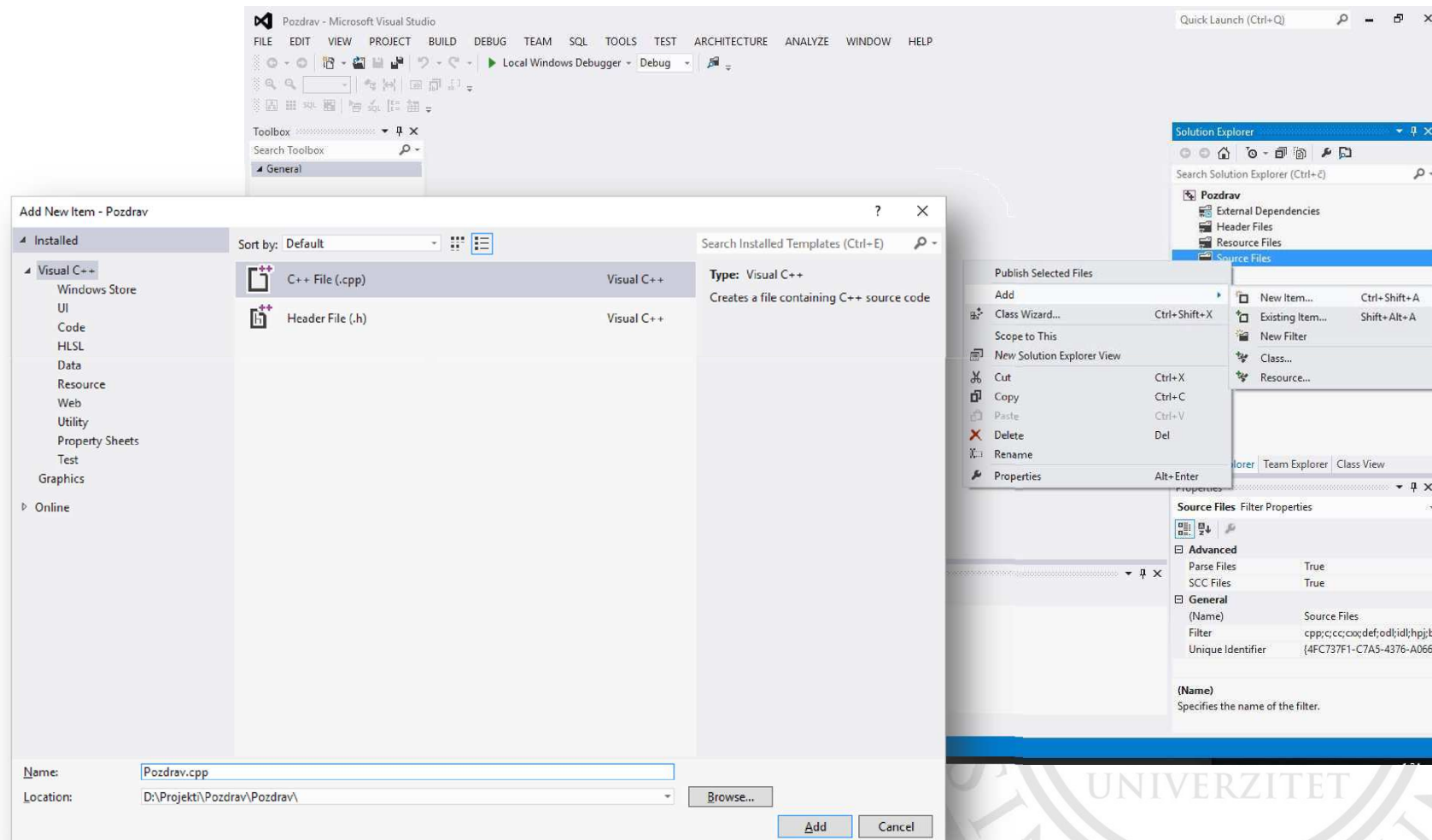
- Integrirani alat, koji podržava sve faze razvoja složenih sistema
- podržava timski rad (TFS), verzije Ultimate/Enterprise do 2015 podržavaju UML modelovanje (6 dijagrama)
- podržava više tehnologija i arhitektura (C++, C#, F#, Java, Veb servisi, SOA)
 - C++11 do C++17
 - C++20 delimično



Kreiranje konzolne aplikacije (VS 2012-2013)



Kreiranje C++ programa (VS 2012-2013)



3.3 Struktura programa u jeziku C++

- Jednostavni C++ program:

```
// Prvi program u jeziku C++
#include <iostream> //pretprocesorska direktiva
int main()          //glavna funkcija
{
    std::cout << "Ovo je prvi C++ program!\n";
    return 0;
}
```



Elementi jednostavnog programa: komentar i pretprocesorske direktive

- Komentari u jezicima C/C++ mogu biti u jednoj ili više linija

```
// komentar u jednoj liniji
```

```
/* komentar  
    u više linija */
```



Elementi jednostavnog programa: komentar i pretprocesorske direktive

- Pretprocesorska *direktiva*

```
#include <iostream>
```

u kod programa uključuje datoteku **iostream.h** u kojoj su deklaracije funkcija koje vrše ulazno izlazne operacije



Elementi jednostavnog programa: glavni program

- Naredba

```
int main() { ... }
```

definiše programski kôd koji treba da se izvrši kad se pokrene izvršavanje programa

- Velike zagrade u jezicima C/C++ grupišu skupove naredbi u blokove
- Funkcija `main()`
 - ne može biti rekurzivna i ne može se pozivati bilo gde u programu
 - može da ima *argumente* različitih tipova (iz komandne linije)

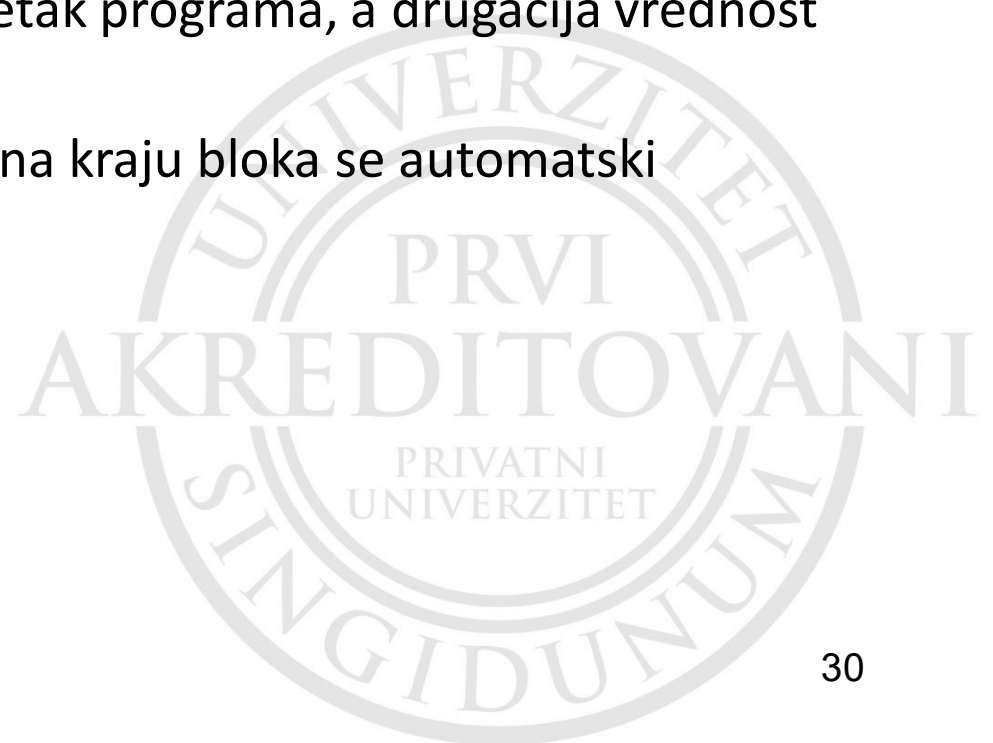
Elementi jednostavnog programa: glavni program

- Naredba

```
return 0
```

završava izvršavanje programa i vraća vrednost 0 procesu koji je pokrenuo program (obično operativni sistem)

- vrednost 0 označava ispravni završetak programa, a drugačija vrednost terminiranje s navedenim kodom
- ukoliko se naredba `return` izostavi, na kraju bloka se automatski izvršava naredba `return 0`



Elementi jednostavnog programa: konzolni ulaz-izlaz

- Naredba

```
std::cout << "Ovo je prvi C++ program!\n";
```

definiše konzolni ispis teksta u standardni izlazni tok `std` (`iostream`). Oznaka `<<` je operator umetanja (*insertion*) podataka u izlazni tok `cout` (*console out*) - ekran računara

- Ukoliko se na početak programa doda naredba

```
using namespace std;
```

kojom se definiše oblast imena koja će se u programu koristiti, dovoljno je napisati samo `cout`

- Analogno, standardni ulazni tok je `cin`, a operator izdvajanja (*extraction*) `>>` asocira na smer toka podataka

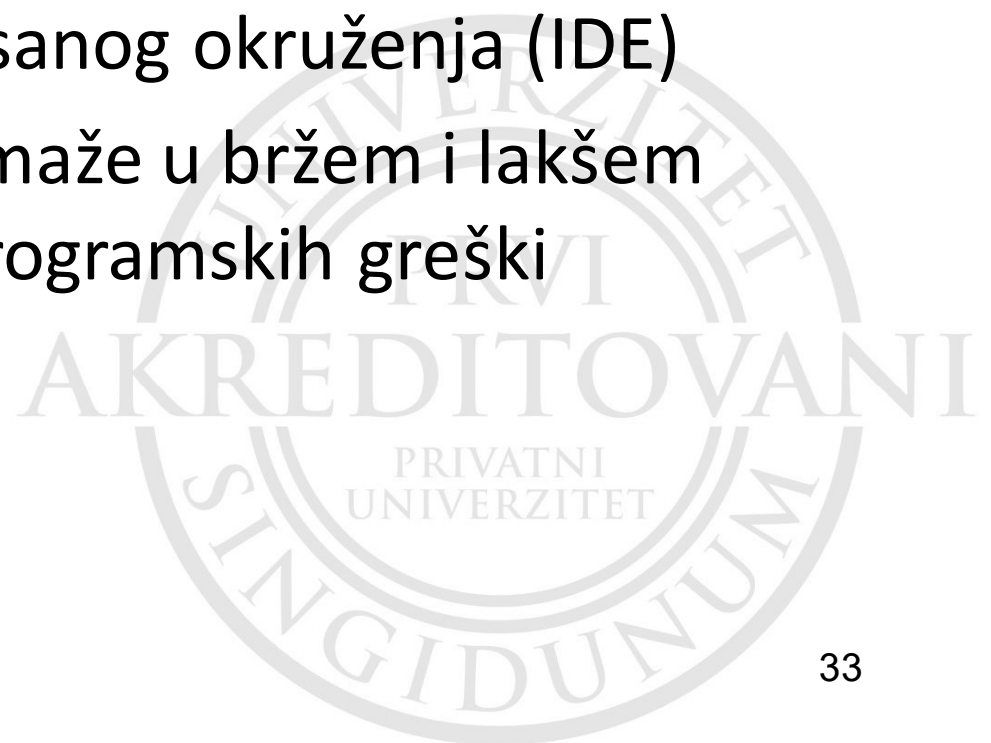
Struktura programa u jeziku C++ (2)

- Jednostavni C++ program s *imenikom* std:

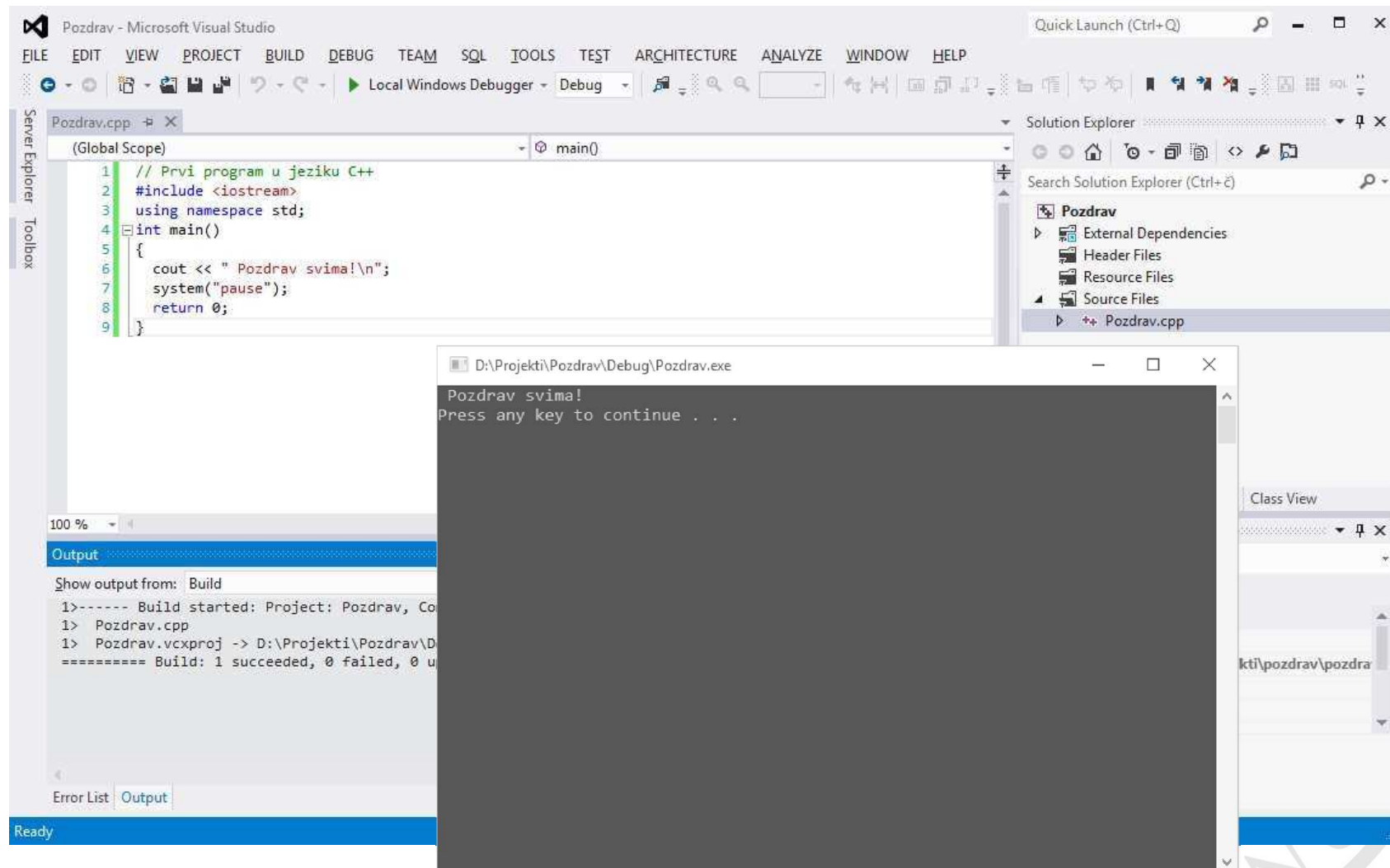
```
// Prvi program u jeziku C++
#include <iostream> //pretprocesorska direktiva
using namespace std; //prostor imena (imenik) std
int main()          //glavna funkcija
{
    cout << "Ovo je prvi C++ program!\n";
    return 0;
}
```


3.4 Unos, prevođenje i izvršavanje C++ programa

- Unos programskog koda u datoteke sa sufiksom **cpp** pomoću editora ili integrisanog okruženja (IDE)
- Prevođenje, povezivanje i izvršavanje pomoću prevodioca u komandnoj liniji (npr. **cl.exe**) ili pomoću komande integrisanog okruženja (IDE)
- Integrisano okruženje pomaže u bržem i lakšem otkrivanju i otklanjanju programskih greški (*debugging*)



Unos i izvršavanje jednostavnog programa u jeziku C++



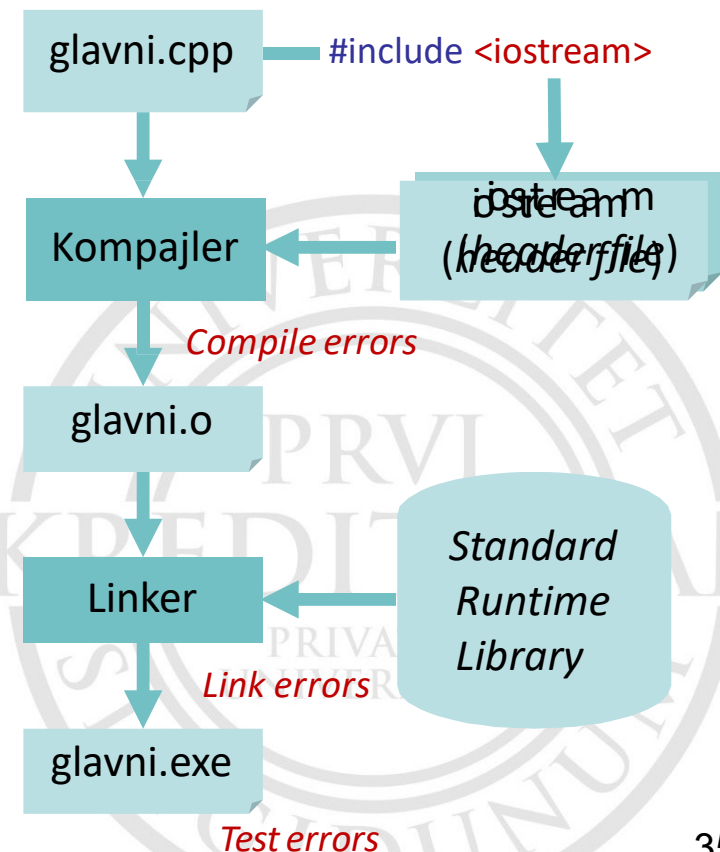
3.5 Pretprocesorske direktive

- Prevodioci programskih jezika C i C++ koriste pretprocesor za izmene i dopune izvornog koda *pre prevođenja*
- Opšti oblik direktive je

#direktiva **parametri**

- pretprocesorske direktive počinju znakom **#** i ne završavaju znakom **;**
- jedna od najčešćih je direktiva **include**, kojom se u izvorni kod programa uključuje sadržaj drugih datoteka
- često se koristi i direktiva **define** kojom se definišu konstante i izrazi, npr.

```
#define TABLE_SIZE 100  
#define square(x) x * x
```



3.6 Stilske preporuke za programiranje u jeziku C++

- Skup pravila za upotrebu jezika C++, uvek za određenu svrhu i u određenom okruženju
 - ne postoji jedan standard za sve namene i sve korisnike, već se stil prilagođava aplikaciji, kompaniji, području primene i izmenama u jeziku
- Tipični elementi standarda kodiranja
 - strukturiranje koda
 - imenovanje objekata
 - softverski alati
- Standardi i preporuke
 - ISO Standard <https://isocpp.org/wiki/faq/coding-standards>
 - B. Stroustrup <http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines>
 - Google <https://google.github.io/styleguide/cppguide.html>

Ilustracija: Programski kod na tri načina

Stil 1	Stil 2	Stil 3
<pre>namespace mojprostor { bool ima_faktor(int x, int y) { int faktor{ nzd(x, y) }; if (faktor > 1) {</pre>	<pre>namespace mojprostor { bool ima_faktor(int x, int y) { int faktor{ nzd(x, y) }; if (faktor > 1) { return true; } else {</pre>	<pre>namespace mojprostor { bool ima_faktor(int x, int y) { int faktor{ nzd(x, y) }; if (faktor > 1) return true; else</pre>

funkcija nzd(x,y) pronalazi najveći zajednički delilac dva broja

4. Sintaksa jezika C++

1. Identifikatori
2. Osnovni tipovi i strukture podataka
3. Selekcija i iteracija
4. Funkcije i rekurzija
5. Upravljanje memorijom



4.1 Identifikatori

- Objektima programa dodeljuju se imena, tzv. *identifikatori*, koja treba da zadovolje određene pravila:
 - identifikator mora da bude sastavljen od slova engleske abecede (**A..Z** i **a..z**), brojeva **0..9** i donje crte (**_**)
 - prvi znak mora da bude slovo ili donja crta
 - identifikator ne može da bude **ključna reč** jezika C++ ili alternativna oznaka nekog operatora
 - posebnu namenu ima *dvostruka donja crta*, pa je treba izbegavati, jer se koriste za nazive implementacija jezika C++ i nazive biblioteka

Rezervisane - ključne reči jezika C++ i alternativni nazivi operatora

asm	do	if	return	typedef
auto	double	inline	short	typeid
bool	dynamic_cast	int	signed	typename
break	else	long	sizeof	union
case	enum	mutable	static	unsigned
catch	explicit	namespace	static_cast	using
char	export	new	struct	virtual
class	extern	operator	switch	void
const	false	private	template	volatile
const_cast	float	protected	this	wchar_t
continue	for	public	throw	while
default	friend	register	true	
delete	goto	reinterpret_cast	try	

and	bitand	compl	not_eq	or_eq	xor_eq
and_eq	bit_or	not	or	xor	

4.2 Osnovni tipovi i strukture podataka

- Ugrađeni tipovi podataka su
 1. celi brojevi (**int**)
 2. realni brojevi (**float**, **double**)
 3. logičke vrednosti (**bool**)
 4. nizovi znakova (**char**, **w_char**)
 5. **void** (bez vrednosti)
- Odgovaraju tipovima koje koriste procesori, što omogućava razvoj efikasnog programskog koda
- Jezik C++ omogućava konstruisanje složenijih tipova podataka, kao što su nabrajanja, strukture i klase

4.3 Selekcija i iteracija

- Selekcija: **if**, **switch**
- Iteracija: **for**, **while**, **do-while**
- Ostale upravljačke naredbe:
break, **continue**, **goto**



4.4 Funkcije i rekurzija

- Program u jeziku C++ predstavlja skup funkcija
- Funkcije vraćaju vrednost određenog tipa; ako ne vraćaju nikakvu vrednost deklarišu se kao tip **void**
- Svi nazivi funkcija su *globalni*
- Dozvoljena je rekurzija



4.5 Upravljanje memorijom

- Više načina, nema automatizma (skupljanja smeća, *garbage collection*), već je programer odgovoran za upravljanje memorijom ([new/delete](#))



5. Primeri programa

1. Implementacija minimalnog programa u različitim programskim jezicima
2. Implementacija programa s dinamičkim poljem u jeziku C++



5.1 Implementacija minimalnog programa u različitim programskim jezicima

- Primer programa koji samo ispisuje na ekran (konzolu) poruku "Pozdrav svima!"
 - implementacija u tri različita savremena programska jezika
 - programi u jeziku C++ prevode se u *mašinski kod* ciljnog računara (efikasniji)
 - programi u jeziku Java i Python se prevode u *međukod* (*bytekod*)
 - program u jeziku Python je najkraći i najjednostavniji

C++

```
#include <iostream>
using namespace std;
int main() {
    cout << "Pozdrav svima!" << endl;
}
```

Java

```
public class PozdravSvima {
    public static void main(String[] args)
    { System.out.println("Pozdrav
    svima!");
    }
}
```

Python

```
print("Pozdrav svima!")
```

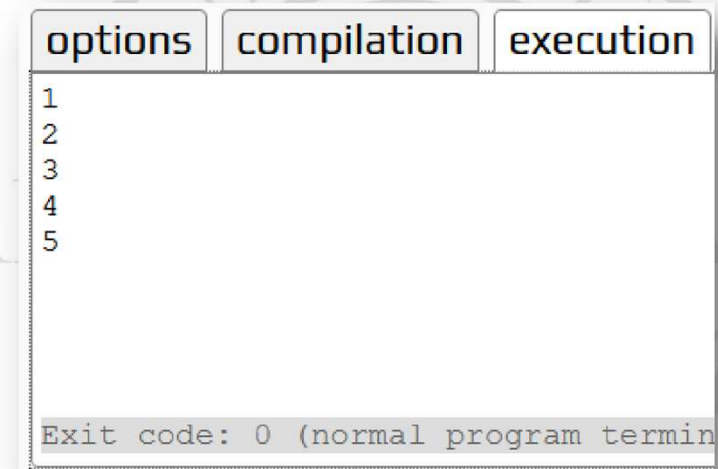
5.2 Implementacija programa s dinamičkim poljem u jeziku C++

```
// Kreiranje, sortiranje i prikaz liste celih brojeva
#include <iostream>
#include <vector>
#include <algorithm>

int main () {

    std::vector<int> v = {2, 1, 5, 3, 4};
    std::sort(v.begin(), v.end());
    for (auto e : v)
        std::cout << e << std::endl;

}
```



Literatura

1. Branović I., *Osnove objektno orijentisanog programiranja: C++*, Univerzitet Singidunum, 2013
2. Stroustrup B., *The C++ Programming Language*, 4th Ed, Addison Wesley, 2013
3. Horton I., Van Weert P., *Beginning C++ 20*, 6th Edition, Apress, 2020
4. Horton I., *Beginning Visual C++ 2013*, Wox/John Wiley&Sons, 2014
5. Horton I., *Using the C++ Standard Template Libraries*, Apress, 2015
6. Galowitz J., *C++ 17 STL Cookbook*, Packt, 2017
7. Horstmann C., *Big C++*, 2nd Ed, John Wiley&Sons, 2009
8. Predavanja i materijali vežbi
9. Veb izvori
 - <http://www.stroustrup.com/>
 - <https://en.wikipedia.org>
 - <https://isocpp.org/>
10. Knjige i priručnici za *Visual Studio* 2010/2012/2013/2015/2017/2019

