PROGRAMSKA OKRUŽENJA

Bilo ko ko želi da programira suočava se sa problemom izbora takozvanog programskog okruženja. Pod programskim okruženjem podrazumeva se skup softverskih alata pomoću kojih programer projektuje i razvija programe. Programsko okruženje uvek uključuje editor, kompajler, biblioteku predhodno razvijenih programa (run time environment), dibager (debugger). Pored ovoga, programeru mogu biti na raspolaganju i drugi softverski alati kao što su alati za analizu i projektovanje sistema (UML, na primer), alati za grafički dizajn korisničkog interfejsa itd.

Mogu se razlikovati dve vrste programskih okruženja: integrisana okruženja (integrated development environment, ili IDE skraćeno) i linijski editori komandi.

Integrisano okruženje je grafički korisnički interfejs koji sadrži sve potrebne alate za razvoj programa kao što su editor, kompajler, grafički dizajner (za forme i sl.), dibager, pojekt menadžer itd.

Okruženje sa linijskim editorom komandi je jednsotavno okruženje gde programer korišćenjem jednostavnih tekst editora unosi naredbe programskog koda, naredbe za kompilaciju i izvršavanje.

Danas se za programiranje najčešće koriste integrisana okruženja. Za programere početnike se međutim preporučuje korišćenje jednostavnih linijskih editora kako bi mogli da prate sve korake u izradi programa, koji su u IDE ponekad sakriveni.

Naravno, IDE su znatno pogodnija za razvoj složenih programskih sistema jer često uključuju i elemente upravljanja projektima.

Poznati primer koji se koristi pri izučavanju bilo kog programskog jezika je primer »Hello World« programa. To je veoma jednostavan program kojim se na ekranu štmpa poruka »Hello World«. To je dobar primer kako je ponekad jednostavnije koristiti linijske editore nego IDE.
I mi ćemo ovde kao prvi primer napisati C# programa koji na monitoru prikayuje tekst »Hello World«.

Evo kako taj program izgleda:

```
using System;
class Hello
{
      static void Main() {
            Console.WriteLine("Hello, World");
      }
}
```

Datoteke koje sadrže C# program obično imaju ekstenziju .cs. Pod pretpostavkom da smo gore navedeni program smestili u datoteku pod imenom hello.cs, onda bismo korišćenjem linijkog editora komandi mogli koristiti komandu cs hello.cs kojom se poziva Microsoft-ov C# kompajler da izvrši prevođenje programa u mašinski kod koji će biti smešten u datoteci hello.exe.

Sada ćemo analizirati ovaj jednostavan program sa ciljem da razumevajući njegovu strukturu, razumemo i čitav proces razvoja programa.

Program počinje using System naredbom čime se označava da želimo da u programu koristimo biblioteku klasa koju je Microsoft (.NET Framework biblioteka) razvio i stavio na raspolaganje programerima. Ova biblioteka sadrži oko 8.000 klasa podeljenih u nekoliko tematskih oblast: za rad sa podacim (System.Data) , za rad sa windows formama (System.Windows.Forms), za rad sa ulazno-izlaznim uređajima (IO) itd.

Zato što smo using System naredbu uvrstili u naš program, možemo da koristimo Console.WriteLine metodu koja je deo gornje biblioteke.

Dalje uočavamo da program počinje definicijom klase hello (class hello), koja ima samo jedan metod pod imenom Main. Ispred nazva Main stoje reči static i void. Reč static označava da se metoda Main ne odnosi ni na jedan poseban objekat, to jest da se static metod koristi bez reference na objekat. Reč void označava da metod Main ne vraća vrednost. Po konvenciji static metod pod imenom Main je ulazna tačka u svaki C# program.

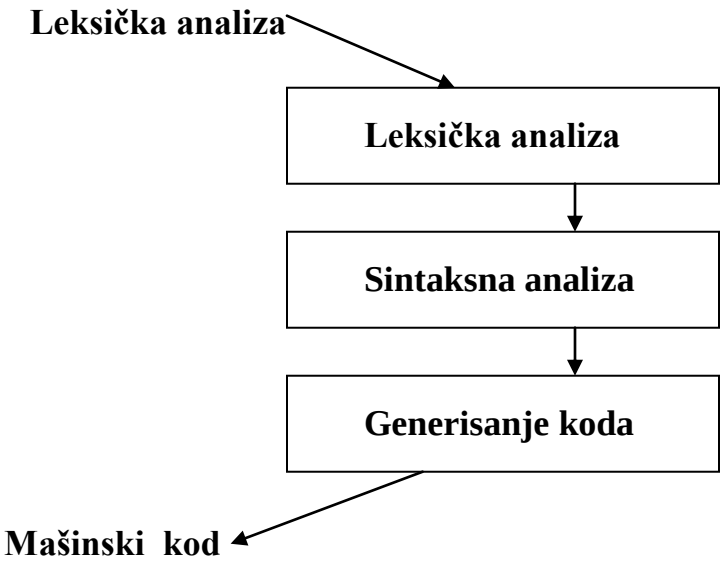
Metod Main sadrži samo jednu liniju koda, koja zapravo i nije neka C# naredba, već poziv metode Console.WriteLine koja je deo biblioteke System, a ta metoda će automatski biti pozvana od strane C# kompajlera.

Kompajleri i interpreteri

Postoje dva načina kako se programi nanpisani u programskim jezicima mogu izvršavati u kompjuterima. To su kompilacija i interpretacija.

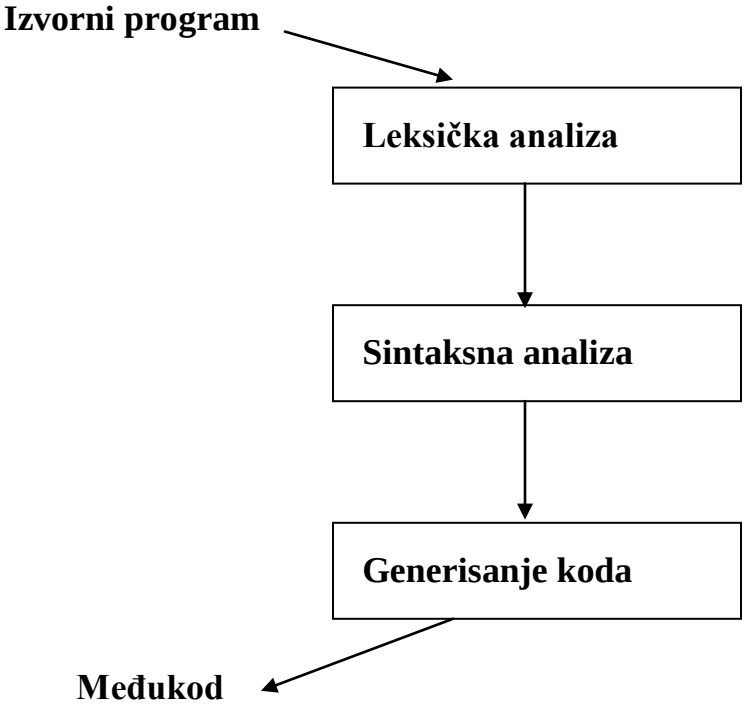
Kompilacija (koja se obavlja kompajlerima) je postupak u kojem se program napisan u izvornom obliku prevodi u mašinski kod datog kompjutera.

Sledeća slika ilustruje proces kompilacije programa od izvornog oblika to objektnog (mašinskog) koda.



Interpretacija (koja se obavlja interpreterima) je postupak u kome se program napisan u izvornom obliku prevodi u neki »međukod«, to jest neki jednostavniji jezik sličan mašinskom jeziku, a zatim kompjuter izvršava (intrepretira) naredbe međukoda. Time se postiže da se isti međukod može izvršavati na različitim kompjuterima, jer nije zavistan od mašinskog jezika nekog specifičnog kompjutera. Za međukod bi se moglo reći da predstavlja univerzalni mašinski jezik.

Sledeća slika ilustruje proces interpretacije.



Kompilirani programi obično rade brže, a prednost interpretera je što su obično iteraktivni i nije potrebno prolaziti sve pripremne faze kao kod kompilacije da bi se program izvršio.

Linkeri i loderi (loadres)

Linkeri i loderi su važni elementi u procesu dobijanja izvršnog (.exe) programa . Njihova uloga je da izvrše povezivanje (linkovanje) različitih internih delova programa kao i eksternih bibliotečkih potprograma u jednistvenu izvršnu celinu, kao i da tako pripremljen izvršni program prenesu (loduju) u memoriju računara kako bi počelo njegovo izvršavanje (egzekucija).

Programski editori

Programski editor je deo integrisanog okrušenja i sluši za editovanje različitih komponenti izvornog koda (programskih naredbi, formi, itd.).

Dibager (Debugger)

Dibager je softverski alat koji služi za testiranje ispravnosti rada programa. On omogućava izvršavanje programa korak-po-korak pri čemu programer može da prati promenu vredosti varijabli.

Dibageri mogu da na automatizovan način prave “trace” tabele koje smo ranije pominjali kao pogodno sredstvo za testiranje algoritama.

Dinamičko povezivanje programa (DLL)

Umesto pravljenja izvršnog programa u .exe obliku, četo se pribegava kompilaciji programa do takozvanog .dll oblika. To je skraćenica od Dynamic Link Library (dll) što znači da je ovako generisan mašinski kod moguće dinamički povezivati tokom izvršenja nekog drugog programa koji može da poziva .dll metode.

Povezivanje sa softverima za baze podataka (ODBC)

***O**pen **D**ata**B**ase **C**onnectivity (ODBC)*, je jedan standardni metod za povezivanje programa sa bazama podataka. Standard je razvila SQL Access grupa 1992 godine sa ciljem da se bilo kom podatku može pritupiti iz bilo koje aplikacije nezavisno od sistema za upravljanje bazama podataka (DBMS). To se postiže umetanjem drajvera za baze podataka kao interfejsa (srednjeg sloja) između aplikacije i DBMS-a. Ovi drajveri mogu se naći u Microsoft .NET Framework biblioteci.

Korisnički interfejs

Pod korisničkim interfejsom, u užem smislu , podrazumeva se izgled ekrana koji korisnik nekog programa vidi na svom monitoru. Međutim, u širem smislu, korinsički interfejs predstavlja bilo koji način na koji čovek interaguje sa kompjuterom. Korisničkim interfejsima se poklanja velika pažnja, jer od toga kakav je korisnički interfejs zavisi

upotrebljivost programa i lakoća sa kojom čovek-korisnik može da koristi neki softver. Dobar korisnički interfejs olakšava korisniku da obavi posao koji mu je potreban. Često se ispred reči korisnički intergejs stavlja i reč "grafički" kojom se označava da softver sadrži grafičke prikaze (forme, slike, ikone, itd.) kojima se olakšava korišćenje programa. Grafički interfejsi se označavaju sa GUI (Graphic User Interface).

Pitanja

1. Šta je programsko okruženje?
2. Koji su najčešće komponente IDE?
3. Šta je kompajler?
4. Šta je interpreter?
5. Šta je linker?
6. Šta je loder?
7. Šta je linijski editor?
8. Šta je dibager?
9. Šta je ODBC?
10. Šta je GUI?

Virtuelna porodica u virtuelnoj Švajcarskoj

Izvor: Virtual Lab, EPFL, Switzerland