**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijska tehnologija

**ANGELA PLAZIBAT**

**SEMINARSKI RAD**

**Case study: Non-mainstream programski jezici**

**ERLANG**

Split, siječanj 2022

**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Preddiplomski stručni studij Informacijska tehnologija

**Predmet:** Objektno orijentirano programiranje

**SEMINARSKI RAD**

**Kandidat:** Angela Plazibat

**Naslov rada:** Case study: Non-mainstream programski jezici

Split, siječanj 2022

Sadržaj

[1. UVOD 1](#_Toc93524153)

[2. UVOD U ERLANG 2](#_Toc93524154)

[3. OSNOVNE KARAKTERISTIKE 3](#_Toc93524155)

[4. PROCESI 4](#_Toc93524156)

[5. FUNKCIJE 5](#_Toc93524157)

[6. TIPOVI PODATAKA U ERLANGU 6](#_Toc93524158)

[7. PROGRAMI U ERLANGU 7](#_Toc93524159)

[7.1. PRIMJERI PROGRAMA 8](#_Toc93524160)

[7.1.1. OSNOVNI PROGRAM 8](#_Toc93524161)

[7.1.2. REKURZIJA 9](#_Toc93524162)

[7.1.3. UVJETNA NAREDBA IF ELSE 11](#_Toc93524163)

[7.1.4. PROCES RODITELJ DIJETE 13](#_Toc93524164)

[8. USPOREDBA ERLANGA SA C++ PROGRAMSKIM JEZIKOM 14](#_Toc93524165)

[9. UPOTREBA 15](#_Toc93524166)

[10. ZAKLJUČAK 16](#_Toc93524167)

[11. LITERATURA 17](#_Toc93524168)

# UVOD

Postoji veliki broj programskih jezika, a svakodnevno nastaju novi i napredniji. Neki od njih vrlo brzo postignu veliku popularnost (main stream programski jezici), dok neki bez obzira na svoje kvalitete ostanu neprepoznati od široke zajednice programera.

Ovaj rad se bavi jednim od ne „main stream“ programskih jezika: Erlangom.

Opisana su svojstva jezika, njegova primjena i usporedba s „main stream“ jezikom C++.

Kroz nekoliko manjih programa glavna svojstva jezika opisana su i na praktičan način.

# UVOD U ERLANG

Programski jezik Erlang je konkurentni, funkcijski jezik namijenjen za raspodijeljene pouzdane sustave koji rade u stvarnom vremenu, a najviše se koristi u telekomunikacijama, npr. u Ericssonovim telekomunikacijskim sustavima.

Konkurentnost označava mogućnost istodobnog izvođenja dvije ili više aktivnosti. Dvije ili više aktivnosti se mogu izvesti istodobno ako ne utječu jedna na drugu.

Erlang je programski jezik otvorenog koda, jako skalabilan i siguran.

Nastao je 1986. godine u Švedskoj od strane tvrtke Ericsson u Ericsson CSLab -u, odakle i potječe naziv Erlang kao kratica od ERicsson LANGuage.

Razvoj Erlanga je počeo s istraživanjima u potrazi za programskim jezikom koji će omogućiti programiranje velikih industrijskih sustava. Zaposleni na ovom projektu željeli su napraviti novi programski jezik koji će biti najbolje rješenje za programiranje aplikacija za telekomunikacije, a to su uradili dodavanjem konkurentnosti u programski jezik Prolog, zbog čega se Erlang jedno vrijeme smatrao dijalektom Prologa. Nakon toga postao je potpuno novi programski jezik, veoma brzo se razvijao, a nastajale su i nove verzije, kao i standardna biblioteka OTP (engl. The open Telecom Platform).

Na nastanak jezika Erlang je najviše utjecao programski jezik Lisp, u sintaksi je značajnu ulogu imao programski jezik Prolog, a iz jezika Smalltalk i PLEX su preuzeti neki koncepti, dok je sam Erlang utjecao na nastanak jezika Elixir i Scala.

1998. godine Erlang je postao dostupan preko otvorenog koda i od tada zajednica oko Erlanga polako i sigurno raste. Posebnu pozornost je privukao posljednjih nekoliko godina pojavom procesora s više jezgri gdje njegova svojstva dolaze do izražaja.

Najzaslužniji za njegov nastanak i razvoj je informatičar Joe Armstrong, engleski računalni znanstvenik koji je radio na području distribuiranih sustava [otpornih na greške](https://en.wikipedia.org/wiki/Fault-tolerant).

Jedna od namjena programskog jezika Erlang je pisanje što sigurnijih programa, koje je moguće popraviti bez isključivanja čitavog sustava.



Slika Logo Erlanga

# OSNOVNE KARAKTERISTIKE

Erlang je izgrađen na iskustvima drugih programskih jezika slične vrste koji su svoje karakteristike gradili na proširivanju postojećih jezika (Concurrent Pascal, Occam i drugi), pa su u njega ugrađene sljedeće osobine:

* **Deklarativna sintaksa**

Ne naređuje se izravno nego se opisuje ono što program treba raditi. Nema sporednih pojava (side effects). Varijabli se vrijednost može dodijeliti samo jednom (single assignement language) čime se dobiva čistiji i pouzdaniji kod, a olakšava pronalaženje pogrešaka (debugging).

* **Konkurentnost**

Programi u Erlangu mogu se izvoditi paralelno. Konkurentnost se gradi na procesima koji komuniciraju izmjenom poruka. Procesi se stvaraju i ukidaju uz malo zauzeće memorijskog prostora (light-weight processes) i mali utrošak vremena.

* **Stvarno vrijeme**

Programi u Erlangu zadovoljavaju uvjete programiranja u stvarnom vremenu (soft real-time), a vrijeme odziva (response time) je reda veličine milisekunde.

* **Non-stop sustav**

Omogućuje izmjenu koda bez prekidanja rada programa. Primjena u nekim specifičnim uvjetima (npr. telefonska centrala ili kontrola zračnog prometa) ne dozvoljava zaustavljanje programa radi promjene koda, za što Erlang ima ugrađene mehanizme.

* **Robustnost**

Ugrađeni su mehanizmi koji omogućuju detekciju pogreške u sustavu i zadržavaju sustav u radu iako je došlo do pogrešaka. Erlang programi su otporni na pogreške u radu čvorova (računala) i veza (mreža) između njih.

* **Upravljanje memorijom**

Memorija se zauzima i oslobađa prema potrebi u stvarnom vremenu (real-time garbage collector) te se na taj način rješavaju uobičajeni problemi sa zauzimanjem memorije.

* **Raspodijeljenost**

Erlang ne koristi zajedničku memoriju već svu komunikaciju između procesa obavlja asinkronom izmjenom poruka.

* **Povezanost s drugim programskim jezicima**

Erlang ima ugrađenu vezu prema drugim programskim jezicima kao npr. C, Java i sl.

* **Platforme**

Postoje verzije Erlanga za Solaris, Windows, Linux, FreeBSD i Mac OS X.

# PROCESI

Jezik Erlang radi samo s procesima.

Proces u Erlangu definiran je kao samostalna obradna jedinka koja postoji konkurentno s drugim procesima u sustavu pri čemu ne postoji nikakva hijerarhija procesa osim one koju uvede programer.

Program je sastavljen od modula koji sadrže procese, a procesi funkcije. Novi paralelni proces pokreće se funkcijom *spawn* koja ujedno vraća identifikator procesa.

Aplikacije u Erlangu su izgrađene od vrlo jednostavnih Erlang procesa u Erlang „runtime“ sustavu koji se mogu promatrati kao "živi" objekti (Objektno orijentirano programiranje), s enkapsulacijom podataka i prosljeđivanjem poruka, ali sposobnih promijeniti ponašanje tijekom vremena izvođenja.

Erlang „runtime“ sustav osigurava strogu izolaciju između Erlang procesa što uključuje prikupljanje podataka i smeća, odvojeno svakim Erlang procesom pojedinačno i transparentnu komunikaciju između procesa na različitim Erlang čvorovima (na različitim hostovima).

Erlang je dizajniran s mehanizmom koji vanjskim procesima olakšava praćenje rušenja (ili hardverskih kvarova), a ne mehanizmom u procesu kao što je rukovanje iznimkama koje se koristi u mnogim drugim programskim jezicima. Rušenja se prijavljuju kao i druge poruke što je jedini način na koji procesi mogu komunicirati međusobno. Bazira se na filozofiji „Let it crash“ što znači da se proces potpuno ponovno pokrene, a ne pokušava se oporaviti od ozbiljnog neuspjeha.

Joe Armstrong, suizumitelj Erlanga, sažeo je principe procesa u svojoj doktorskoj tezi :

* Sve je proces
* Procesi su jako izolirani
* Stvaranje i uništavanje procesa je lagana operacija
* Prosljeđivanje poruka jedini je način interakcije procesa
* Procesi imaju jedinstvena imena
* Ako znate naziv procesa, možete mu poslati poruku
* Procesi ne dijele resurse
* Rukovanje pogreškama nije lokalno
* Procesi rade ono što bi trebali učiniti ili ne uspijevaju

# FUNKCIJE

Funkcijska paradigma jedna od sastavnica jezika Erlang. Svaku funkciju čini jedna ili više rečenica (clause) odijeljenih s ";", a svaka pojedina rečenica sastoji se od zaglavlja (head), opcijskog uvjeta „guard“ i tijela (body). Da bi se izvela funkcija, uvjet treba biti ispunjen.

Erlang funkcije definirane su ovako:

*function := function\_clause | function\_clause ";" function*

*function\_clause := clause\_head clause\_guard clause\_body*

*clause\_head := "naziv funkcije"(" argumenti odijeljeni zarezima")*

*clause-guard := "when" guard | e*

*clause\_body := "-" jedan ili više izraza*

*guard := "true" | guard\_tests*

gdje je *guard\_test* - jedna ili više usporedbi i poziva predefiniranih provjera odijeljenih zarezima, a *e* prazno.

Postoji mnogo korisnih naredbi u Erlangovoj ljusci, a neke od njih su:

* *f().* Kaže ljusci da zaboravi sve varijable koje smo dodijelili. Ova funkcija je korisna kod isprobavanja.
* *c(Mod).* Prevodi izvorni kod modula u Erlangu u izvršni kod tj. iz datoteke npr. x.erl stvara x.beam.
* *pwd().* Ispisuje trenutni direktorij u kojem se nalazi ljuska.
* *cd("c:/work").* Premješta ljusku u direktorij work na C disku.
* *q().* Završava izvođenje ljuske i gasi virtualni stroj.
* *init:get\_argument(home).* Vraća direktorij u kojem je pokrenut virtualni stroj.
* *code:get\_path().* Vraća listu direktorija u kojima virtualni stroj traži prevedene module.
* *code:add\_patha(Dir).* Dodaje direktorij na početak liste direktorija koji se pretražuju kod učitavanja prevedenog modula.
* *code:add\_pathz(Dir).* Dodaje direktorij na kraj liste direktorija koji se pretražuju kod učitavanja prevedenog modula.

# TIPOVI PODATAKA U ERLANGU

Tipovi podataka u Erlangu mogu se podijeliti na jednostavne i složene.

Jednostavni su brojevi i atomi.

Brojevi predstavljaju cijele i realne brojeve (pr. -1, -2.2, 0, 100.1, 400), a atomi su tekstualna polja zatvorena u jednostruke navodnike(pr. atom, atom\_a, 'ovo je atom').

Izvedene vrste su n-torke (Tuples) koje sadrže fiksni broj elemenata i liste (Lists) koje su slične n-torkama, ali nisu fiksne duljine, što znači da tijekom izvršavanja programa mogu mijenjati svoju duljinu spajanjem ili dijeljenjem.

U Erlangu se stringovi zapisuju kao liste brojeva, gdje je svaki znak jedan broj.

Primjer: String=“HIKJ“. Varijabla String je zapravo lista [72,73,75,74]. Znakovi se kodiraju standardom ISO8859-1, odnosno Latin-1. Kod nekog slova možemo dobiti pomoću znaka $ npr. kod za slovo H možemo dobiti pomoću izraza $H. Funkcije koje manipuliraju stringovima se nalaze u modulu string.

# PROGRAMI U ERLANGU

Programi u Erlangu su organizirani u module u kojima se nalaze skupine srodnih i logički povezanih funkcija. Svaki modul počinje s ključnom riječi '*module*', iza koje se navodi naziv modula: *module(ModuleName).*

Moduli se spremaju u datoteke čiji je naziv oblika: ime\_modula.erl.

Pozivanje funkcije ovisi o nazivu modula (ModuleName), nazivu funkcije (FunctionName) te o pojavljivanju naziva funkcije u uvezenoj (import) ili izvezenoj (export) deklaraciji modula.

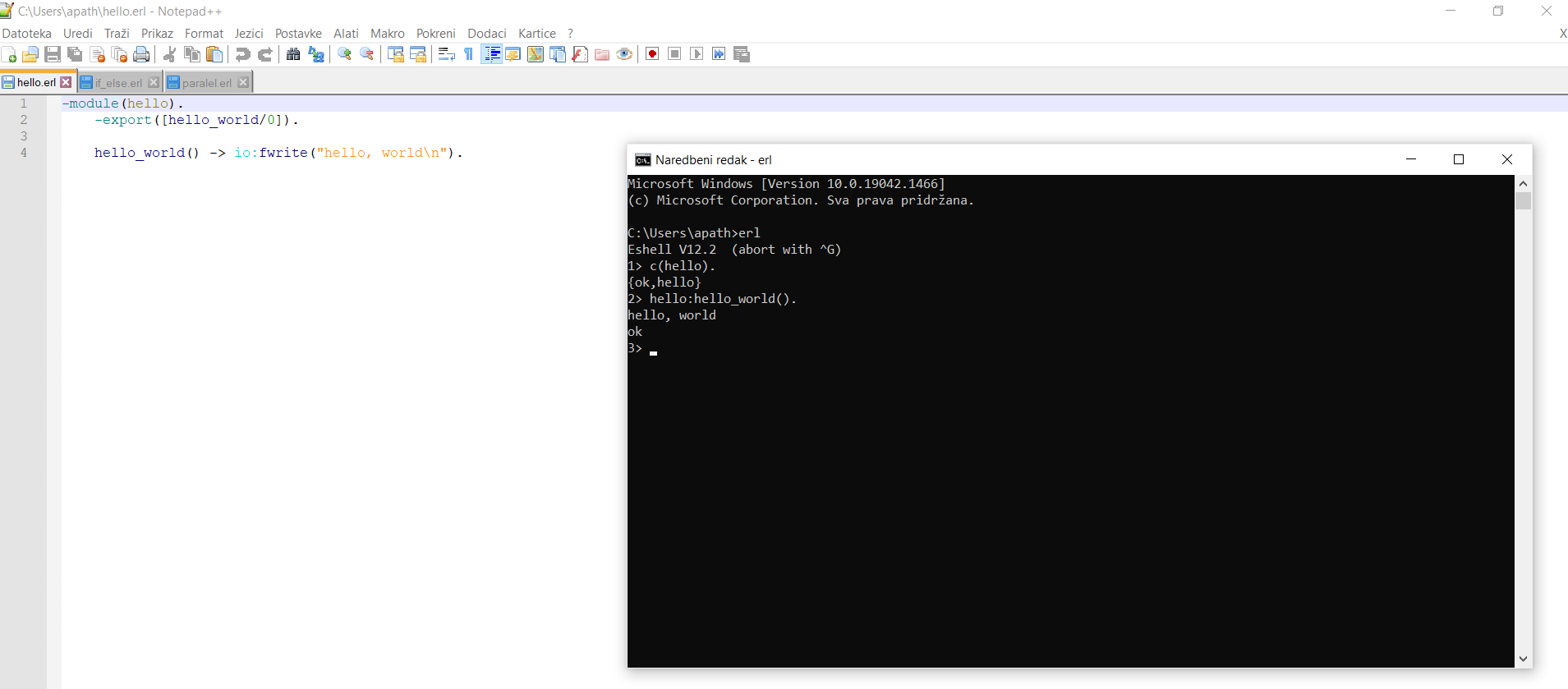
Sve funkcije koje se mogu pozivati izvan modula (tj. ne pozivaju same sebe unutar modula) se moraju deklarirati u dijelu export, piše se ključna riječ '*export'* iza koje se navodi popis funkcija s brojem parametara koje koriste.

Da bi se funkcija mogla pozvati iz drugog modula treba biti deklarirana ovako: export(FunctionName), a da bi modul mogao pozvati funkciju iz drugog modula moramo ga uvesti kao što slijedi: import(ModuleName, FunctionName).

## PRIMJERI PROGRAMA

### 7.1.1. OSNOVNI PROGRAM

Kako bi se upoznali s osnovnom sintaksom samog programa, u primjeru se može vidjeti jednostavan program koji prikazuje poruku „Hello world“.

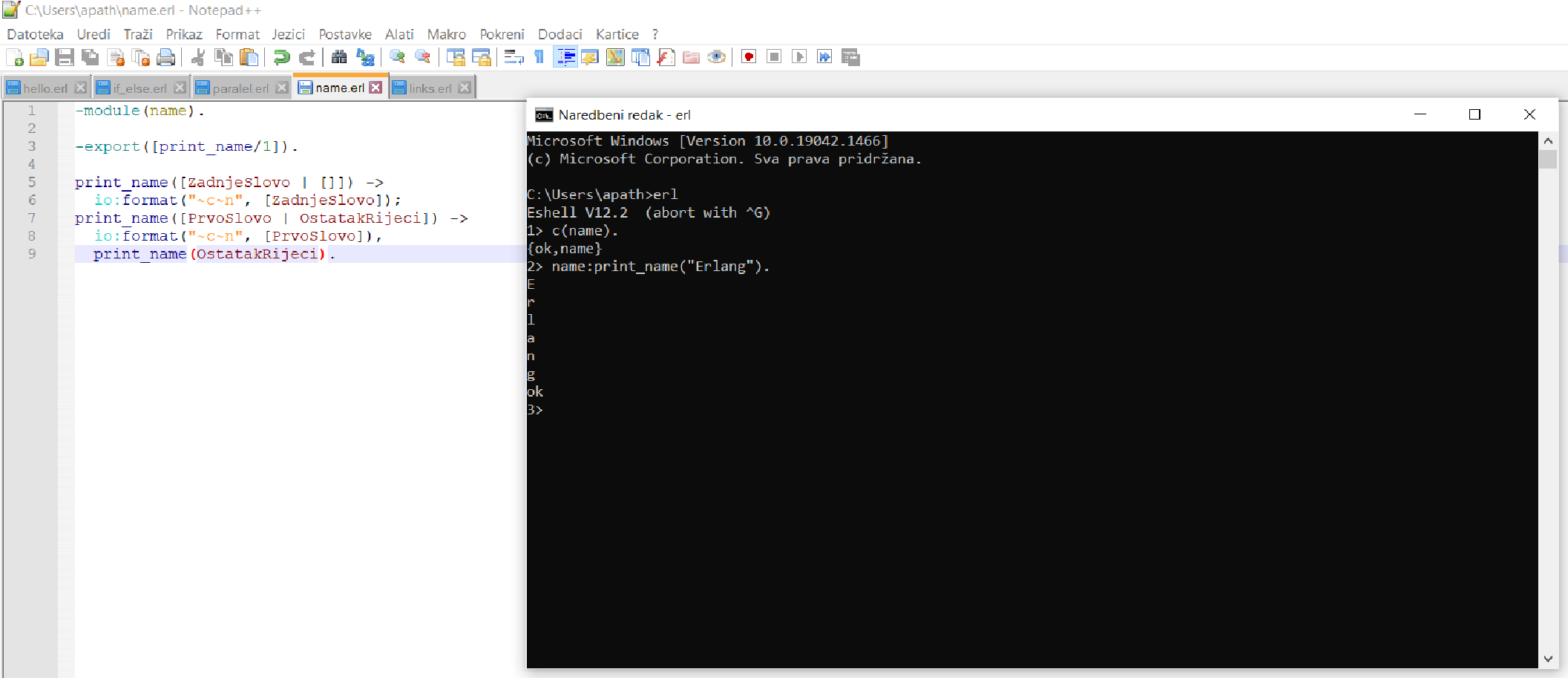


Slika Prikaz "Hello World" programa

Kao što se može vidjeti iz primjera, naredba za ispis u Erlangu izgleda ovako: *io:fwrite(„tekst“).*, a kraj naredbe završava se točkom.

### REKURZIJA

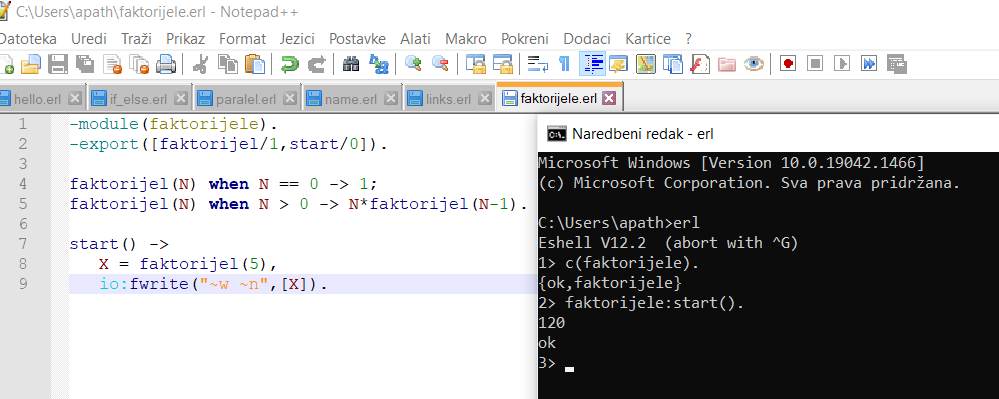
Rekurzija je funkcija koja poziva samu sebe. Rekurzivno programiranje je često korištena tehnika kojom je moguće implementirati razne algoritme i vrlo je bitna kod programiranja u Erlangu. Rekurzija uvijek mora imati neko ograničenje koje će ju zaustaviti jer bi se u suprotnom funkcija pozivala beskonačan broj puta i program se nikada ne bi izvršio.



Slika Primjer rekurzije ispis slovo po slovo

Na slici je prikazan primjer rekurzije koji se može koristiti za ispis izraza slovo po slovo. U principu funkcionira na način da prvo provjerava jeli neki izraz samo jedno slovo i ako je ispisuje ga i završava, a ako nije 'dijeli ga' na prvo slovo i ostatak tog izraza te ispisuje prvo slovo i ponovno poziva samu sebe (rekurzija), a kao argument se šalje ostatak izraza.

Još jedan, ujedno i najpoznatiji primjer korištenja rekurzije je i računanje faktorijela.



Slika Rekurzivni izračun faktorijela

Poziva se funkcija *faktorijel()* kojoj se šalje neka vrijednost, ako je ta vrijednost jednaka nuli funkcija će vratiti 1 i neće se rekurzivno pozivati, ali ako je vrijednost veća od nule funkcija će uraditi množenje te vrijednosti N sa povratnom vrijednosti funkcije *faktorijel(N-1)* koja će se pozivati sve dok N ne bude 0 (tada će vratiti jedinicu i završiti).

### UVJETNA NAREDBA IF ELSE

Bitno svojstvo svakog programskog jezika je uvjetno grananje koje obavlja različite akcije ovisno o postavljenim uvjetima. Način pisanja uvjetnog grananja (if else) razlikuje se u različitim programskim jezicima.

U C++ programu sintaksa if else grananja izgleda ovako:

*if(uvjet)*

*{*

*ono što će se izvršiti ako je uvjet ispunjen;*

*}*

*else*

*{*

*Ono što će se izvršiti ako početni uvjet nije ispunjen;*

*}*

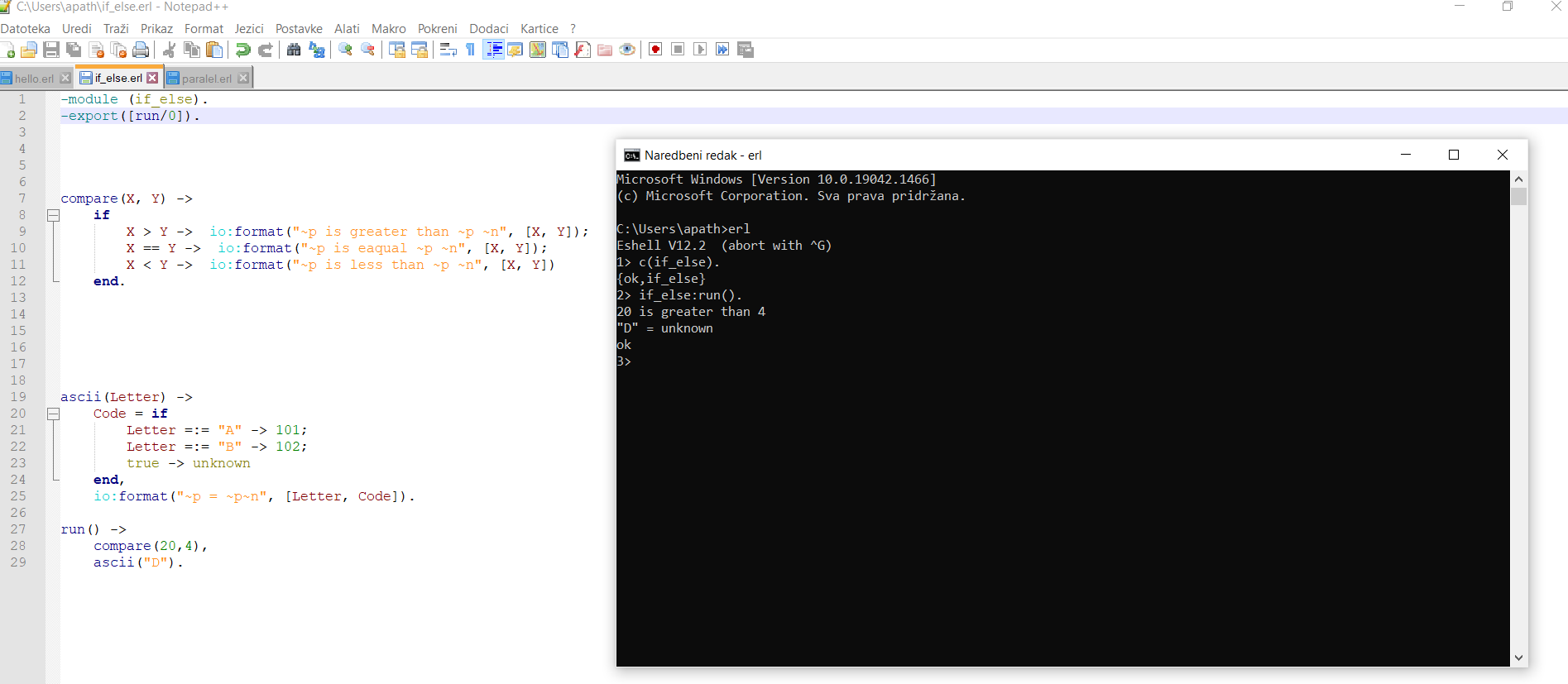
dok se u Erlangu sintaksa piše na slijedeći način:

*if*

*Uvjet->ono što će se izvršiti ako je uvjet ispunjen;*

*true->ono što će se izvršiti ako uvjet nije ispunjen*

*end.*



Slika Primjer if else naredbe

U ovom primjeru prikazan je jedan od načina korištenja uvjetnog grananja gdje uspoređujemo dva broja. Program uspoređuje dva broja X i Y na način da se u prvom koraku provjerava je li X veći od Y te ako je ispisuje odgovarajuću poruku, a ako nije ponovno ih se uspoređuje i provjerava se jesu li jednaki, ako jesu ispisuje se odgovarajuća poruka, a ako ni to nije slučaj, provjerava se je li X manji od Y te ako je ispisuje se odgovarajuća poruka za taj slučaj. Ovaj način grananja može se usporediti s if…else if načinom grananja u C++ programskom jeziku. U drugoj funkciji ovog programa uspoređuje se varijabla *Letter* sa slovima A i B, te ako je *Letter* identičan A u varijablu *Code* sprema se ASCII vrijednost 101, ako je *Letter* identičan B u varijablu *Code* sprema se ASCII vrijednost 102, a ako nije identičan ni A ni B onda se u varijablu *Code* dodijeli vrijednost 'unknown' te se nakon toga ispiše vrijednost varijabli *Letter* i *Code*.

### PROCES RODITELJ DIJETE

Kao što je ranije navedeno Erlang radi isključivo s procesima, stoga procesi stvaraju i svoje podprocese. Kada jedan proces stvori drugi taj odnos možemo promatrati kao odnos roditelj-dijete (eng. Parent child process). Svaki proces može stvoriti mnogo podređenih procesa, ali će imati najviše jedan roditeljski proces. Podproces (dijete) nasljeđuje svojstva procesa koji ga je stvorio (roditelj). Podproces se uvijek završi prije procesa roditelja te kad se završi neke informacije vraćaju se procesu roditelju.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika Stvaranje i vezivanje procesa

U primjeru na slici 6 prikazan je način stvaranja procesa te stvaranje veze roditelj-dijete. Novi proces se stvara naredbom *spawn*, a veza roditelj-dijete naredbom *link*. Naredba *spawn* vraća identifikator procesa (pid).

Općeniti način pisanja naredbe *spawn* je*:*

*spawn(Module, FunctionName, ArgumentList),* gdje *Module* predstavlja naziv modula u kojem se nalazi proces, *FunctionName* naziv funkcije koja se stvara, a *ArgumentList* je lista argumenata.

Vezivanje procesa u roditelj-dijete odnosu se vrši naredbom *link* u roditeljskoj funkciji. Izgled naredbe za vezivanje je: *link(pid djeteta)*, gdje je *pid djeteta* identifikator podprocesa.

# USPOREDBA ERLANGA SA C++ PROGRAMSKIM JEZIKOM

Obzirom da je Erlang manje popularan i manje poznat programski jezik prirodno se upitati zašto ga koristiti pored popularnijih programskih jezika. Usporedimo li Erlang sa C++ programskim jezikom primijetiti ćemo da je C++ popularniji i češće korišten jezik koji ima imperativne, objektno orijentirane i generičke programske značajke, ali i pruža mogućnosti za manipulaciju memorijom niske razine. Kompajlira se izravno u izvorni kod stroja, omogućujući mu da bude jedan od najbržih jezika na svijetu (ako je optimiziran) pa je stoga brži od Erlanga. S druge strane Erlang je open source i real time programski jezik koji se koristi za izgradnju masivno skalabilnih sustava u stvarnom vremenu sa zahtjevima visoke dostupnosti pa je svoju primjenu pronašao u telekomunikacijama, bankarstvu, e-trgovini i računalnoj telefoniji. Erlangov runtime sustav ima ugrađenu podršku za istodobnost, distribuciju i toleranciju grešaka.

Kao razloge za korištenje Erlanga  mora se naglasiti njegovu sposobnost rukovanja konkurentnim i distribuiranim programiranjem. Pod istodobnošću se podrazumijevaju programi koji mogu rukovati s nekoliko niti izvršavanja u isto vrijeme. Na primjer, moderni operacijski sustavi omogućuju korištenje procesora teksta, proračunske tablice, klijenta e-pošte i ispisa koji se izvršavaju u isto vrijeme. Svaki procesor (CPU) u sustavu vjerojatno obrađuje samo jednu nit (ili posao) u isto vrijeme, ali se mijenja između poslova takvom brzinom da daje iluziju da ih sve izvodi u isto vrijeme. Lako je stvoriti paralelne niti izvršavanja u Erlang programu i dopustiti tim nitima da međusobno komuniciraju. Dakle, Erlang je izvrstan alat za tvrtke koje se bave računalnim igrama, trgovinom i velikim podacima.

# UPOTREBA

Erlang je programski jezik koji se najčešće koristi u telekomunikacijama, upotrebljava se u GPRS , 3G i LTE mobilnim mrežama diljem svijeta.

Mnogo je tvrtki koje su izgradile svoj proizvod u Erlangu, a među najpoznatijim su   
Amazon koji je koristio Erlang za implementaciju SimpleDB-a, pružajući usluge baze podataka kao dio Amazon Elastic Compute Cloud (EC2), Facebook koji je koristio Erlang za napajanje pozadine svoje usluge chata, obrađujući više od 100 milijuna aktivnih korisnika i WhatsApp koji je koristio Erlang za pokretanje poslužitelja za razmjenu poruka, postižući do 2 milijuna povezanih korisnika po poslužitelju.

Otkako je pušten kao open source, Erlang se širi izvan telekomunikacija u grane kao što su FinTech, Gaming, Zdravstvo, Automobilska industrija, IoT i Blockchain, a za svoje proizvode koristile su ga i tvrtke Vocalink (tvrtka MasterCard), Goldman Sachs, Nintendo, AdRoll, Grindr, BT Mobile, Samsung, OpenX, SITA i mnoge druge.

# ZAKLJUČAK

Erlang je funkcionalni i dinamički pisani programski jezik koji se koristi kao svojevrsni standard u sustavima koji zahtijevaju veliku konkurentnost. Karakteriziraju ga značajke poput konkurentnosti, otpornosti na greške te distribuiranosti.

Može se zaključiti da su najveće prednosti korištenja Erlanga jer je u stvarnom vremenu, podržava mnoge distribuirane aplikacije te što omogućuje izmjenu koda bez prekida rada.

Najčešće se koristi u telekomunikacijama jer je idealan za usluge s milijunima korisnika.

Iako Erlang nije toliko popularan i poznaje ga manji broj računalnih programera korišten je u Facebookovom servisu za razmjenu poruka i u WhatsApp-u gdje je dnevno potrebno prenijeti ogromnu količinu poruka, bez ikakvih problema za korisnike, ali i u mnogim drugim servisima.

# LITERATURA

1. Armstrong, J.A., „Programming Erlang: Software for a Concurrent World“, The Pragmatic Bookshelf, 2007.
2. <https://www.erlang.org/>
3. <https://en.wikipedia.org/wiki/Erlang_(programming_language)>
4. <https://erlangbyexample.org/>
5. <https://learnyousomeerlang.com/content>