Predavanje 1

# Programiranje II: Koncepti programskih jezikov

Iztok Savnik, FAMNIT

February, 2024.

# Vsebina

- Organizacija predmeta
- Zgodovina programskih jezikov
- Programski modeli
- Koncepti programskih jezikov
- Meta jezik ML
- Metode predmeta

# Organizacija predmeta

- Predavanja
  - Iztok Savnik
- Vaje
  - Peter Muršič
- Pisni izpit
  - 4-5 nalog iz snovi predstavljene na predavanjih in vajah
  - 90 min
- Domače naloge
  - Pisanje programov v Ocaml
  - 3 domače naloge
- Ustni izpit
  - Pregled pisnega izpita in domače naloge
  - 3 vprašanja iz snovi predmeta

# Ocenjevanje

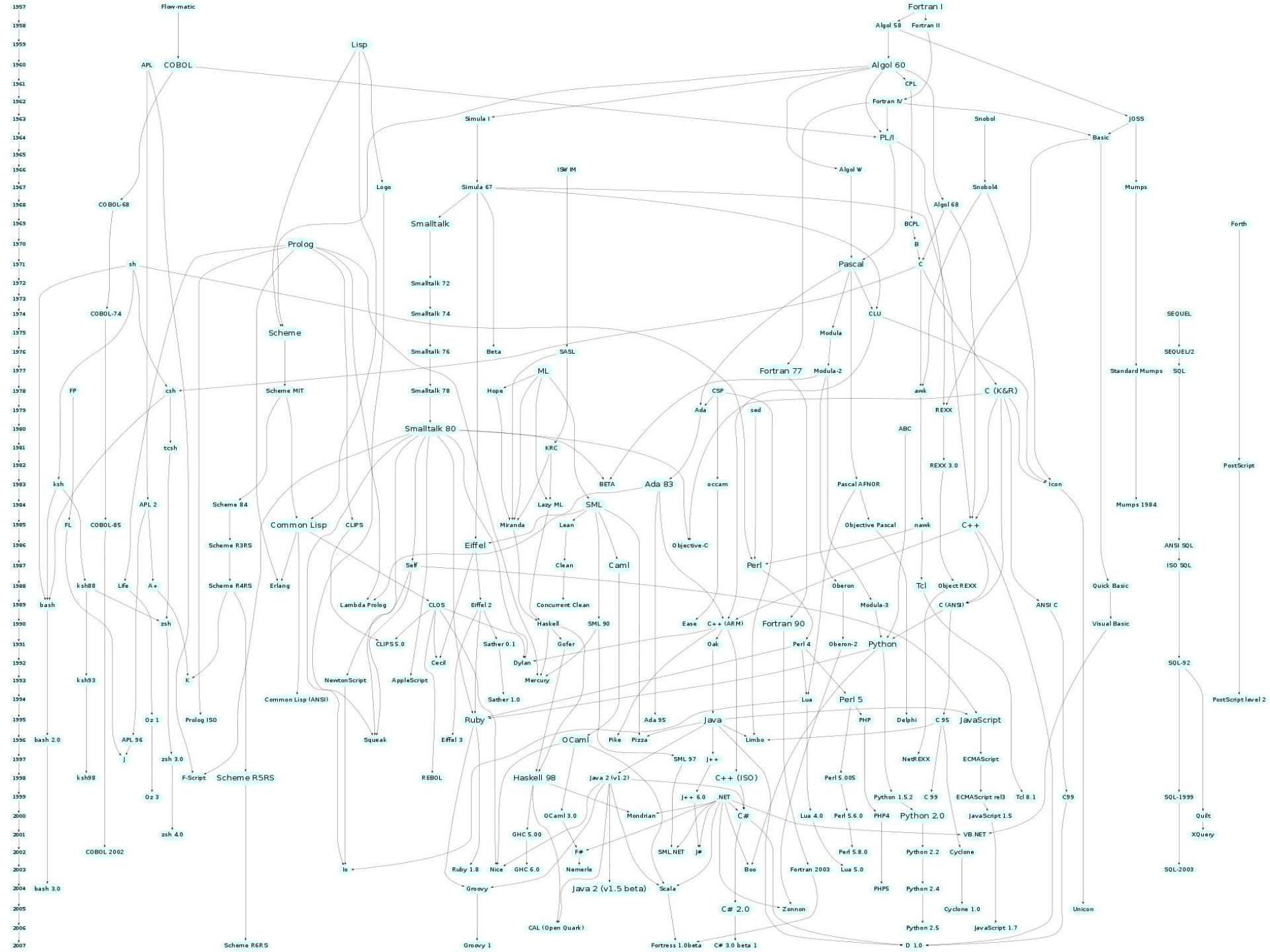
1) Pisni izpit – 40%
2) Domače naloge – 40%
3) Ustni izpit – 20%

Ocene vseh delov morajo biti več od 50%.

Dela ocene (1) in (2) morata biti >50% za pristop na ustni izpit.

# Literatura


1. John Mitchell,
   Concepts in Programming Languages,
   Cambridge Univ Press, 2003.
2. Michael L. Scott,
   Programming Language Pragmatics (3rd ed.)
   Elsevier, 2009.
3. Prosojnice predavanj Programiranje 2
4. Zbirka vaj iz Programiranja 2


1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
Flow-matic
Fortran I
Algol 58
Fortran II
Lisp
APL
COBOL
Algol 60
CPL
Fortran IV
Simula I
Snobol
JOSS
PL/I
Basic
Algol W
ISWIM
Logo
Simula 67
Snobol4
Mumps
Forth
COBOL-68
Algol 68
Smalltalk
BCPL
B
Prolog
C
Pascal
sh
Smalltalk 72
Smalltalk 74
COBOL-74
CLU
SEQUEL
Modula
Scheme
Smalltalk 76
Beta
SASL
SEQUEL/2
Fortran 77
Modula-2
Standard Mumps
SQL
FP
csh
Scheme MIT
Smalltalk 78
ML
Hope
CSP
awk
C (K&R)
Ada
sed
REXX
Smalltalk 80
tcsh
ABC
KRC
REXX 3.0
PostScript
ksh
BETA
Ada 83
occam
Pascal AFNOR
Icon
APL 2
Scheme 84
Lazy ML
SML
Mumps 1984
FL
COBOL-85
Common Lisp
CLIPS
Miranda
Lean
Objective Pascal
nawk
C++
Eiffel
ANSI SQL
Scheme R3RS
Objective-C
Self
Clean
Caml
Perl
ISO SQL
ksh88
Life
A+
Scheme R4RS
Erlang
Oberon
Tcl
Object REXX
Quick Basic
bash
Lambda Prolog
CLOS
Eiffel 2
Concurrent Clean
C (ANSI)
ANSI C
zsh
Haskell
SML 90
Ease
C++ (ARM)
Modula-3
Fortran 90
Visual Basic
CLIPS 5.0
Sather 0.1
Gofer
Oak
Oberon-2
Python
ksh93
Cecil
Dylan
Perl 4
SQL-92
K
NewtonScript
AppleScript
Mercury
Common Lisp (ANSI)
Sather 1.0
Lua
Perl 5
PostScript level 2
Ruby
Ada 95
Java
PHP
Delphi
C 95
JavaScript
0z 1
Prolog ISO
Squeak
Pike
Pizza
Limbo
OCaml
bash 2.0
APL 96
J
Eiffel 3
SML 97
J++
NetREXX
ECMAScript
zsh 3.0
ksh98
F-Script
Scheme R5RS
REBOL
Haskell 98
Java 2 (v1.2)
C++ (ISO)
Perl 5.005
Tcl 8.1
C99
0z 3
J++ 6.0
.NET
Python 1.5.2
C 99
ECMAScript rel3
SQL-1999
OCaml 3.0
Mondrian
C#
Lua 4.0
Perl 5.6.0
PHP4
Python 2.0
JavaScript 1.5
Quilt
zsh 4.0
GHC 5.00
VB.NET
XQuery
COBOL 2002
F#
SML.NET
J#
Perl 5.8.0
Python 2.2
Cyclone
Io
Ruby 1.8
Nice
GHC 6.0
Nemerle
Boo
Fortran 2003
Lua 5.0
SQL-2003
bash 3.0
Groovy
Java 2 (v1.5 beta)
Scala
PHP5
Python 2.4
C# 2.0
Zonnon
Cyclone 1.0
Unicon
CAL (Open Quark)
Python 2.5
JavaScript 1.7
Scheme R6RS
Groovy 1
Fortress 1.0beta
C# 3.0 beta 1
D 1.0

# Vsebina

- Organizacija predmeta
- Zgodovina programskih jezikov
- Programski modeli
- Koncepti programskih jezikov
- Meta jezik ML
- Metode predmeta

# Charles Babbage

- Izvori programabilnega digitalnega računalnika
  - Angleški matematik, filozof, izumitelj in (strojni) inženir.
  - V 1837 je načrtal prvi mehanski računalnik *Analytical engine*, ki je vodil do načrtovanja bolj kompleksnih računalnikov.
- Anlytical engine
  - Mehansko računanje aritmetičnih operacij
  - Sekvence in vejitve
  - Zanke v katerih je rezultat prejšnjega izvajanja uporabljen kot vhod v naslednjo iteracijo.
- Turingov jezik

# Konrad Zuse

- Prvi elektromehanski računalnik
  - Nemški inženir in izumitelj Konrad Zuse
  - Z1, Z2, Z3, 1936-1945, neodvisno od vseh ostalih
  - Mehanični digit. kalkulator za števila s plavajočo vejico Z1
    - Binarni sistem; vrsta izračunov aritmetičnih operacij
  - Z2 = Z1 + Uporaba telefonskih relejev za spomin
  - Z3 = Izboljšan Z2 (22-bit aritmetika za štev. s plav. vejico)
- Prvi programski jezik Plankalkuel
  - Sekvence aritmetičnih operacij
  - Zanke
  - Ni vejitev
    - Čeprav ni vejitev je Turingov jezik

# ENIAC

- Prvi splošno namenski elektronski računalnik.
  - 1943 Uni. of Pennsylvania (J. Mauchly and J.P. Eckert)
  - ENIAC (Electronic Numerical Integrator and Computer)
  - 1000x hitrejši od elektro-mehaničnih računalnikov
  - Programiranje
    - Desetiški sistem, aritmetika, akumulatorji, splošno vodilo
    - Kompleksne funkcije, vejitve, zanke, procedure
- UNIVAC se je razvil iz ENIAC
  - Prvi komercialni računalnik

# Elektronski računalniki

- Mark 1, Mark 2, Mark 3
  - ASCC (Automatic Sequence Controlled Calculator)
  - Razvit na Univerzi Harvard, 1944
  - Originalni koncept predstavljen na IBM, H.Aiken, 1937
  - Sledi modelu Charles Babbage
  - Zasnova, arhitektura
    - Decimalni sistem, 60 množic po 24 stikal za vhod, 72 števil
    - ~2000 števcev, 72 strojev za seštevanje
    - Luknjan trak, program, podatki
    - Instrukcije, izvajanje trenutne inst., branje naslednje
    - Zanke, luknjan trak pričvrščen na začetek
    - Vejitve, ročno, kasneje elektronsko (1946)
    - 800km žic, 5 ton, 5KW moč
-

# Elektronski računalniki

- von Neumann arhitektura
  - Princeton arhitektura, 1945
  - Prvi osnutek poročila o EDVAC
    - Naslednik ENIAC, Univerza v Pensilvaniji, 1944
    - U.S. Army's Ballistics Research Laboratory
    - 44-bit beseda, 5.6KB spomin, 5,937 vakumskih cevi, 12,000 diod, moč 56 kW, 45.5 m², (+) 0.864 ms, (*) 2.9 ms
    - von Neumann se je pridružil kot svetovalec
  - Osnovne abstrakcije, predstavitev osnovnih funkcij računalnika
    - Centralno procesna enota
    - Glavni spomin
    - Vhodno/izhodne naprave

# Prvi prevajalnik

- Programski jezik A-0
  - Algorithmic Language version 0
  - Razvit na računalniku UNIVAC I
  - Vodi: Grace Hopper v 1951, 1952
- Program v jeziku A-0 je bil sekvenca klicev podprogramov skupaj s parametri
- Prevajalnik je samo povezovalnik (linker) in nalagalnik (loader)
- Sledijo A-1, A-2, A-3
  - Komercialna imena: ARITH-MATIC, MATH-MATIC (A-3), FLOW-MATIC (B-0)

# Fortran

- Prvi kompletni prevajalnik
- John W. Backus, 1953, IBM
  - IBM 704 mainframe
- Mathematical **FOR**mula **TRAN**slating System
  - IBM 360
- Gradniki Fortran
  - DO, GOTO, IF, SUBRUTINE, CALL
- Numerično procesiranje
- Sledi imperativna družina
  - C, Pascal, Modula, Ada, Java
- Fortran II, Fortran 77, Fortran III, Fortran 90, Fortran 95, Fortran 2003, ...

# FLOW-MATIC

- Bussiness Language version 0, ali, B-0.
  - Računalnik UNIVAC I, Remington Rand
  - Grace Hopper, 1955.
  - Poslovni svet nima rad formul; uporabi Angleški jezik
- Skupina Hopper je razvila prevajalnik 1959.
  - Prvi jezik, kjer se je ločilo med podatki in programom
- Močen vpliv na razvoj programskega jezika COBOL.

# Information Processing Language

- Information Processing Language (IPL)
  - Allen Newell, Cliff Shaw in Herbert Simon
  - Rand Corporation v 1956.
  - Zbirnik – operacije delajo na seznamih
- IPL računalnik
  - Množica simbolov, množica celic, množica prim. funkcij
  - Funkcije definirane na celicah in seznamih
  - Primitivno (run-time) okolje programskega jezika
  - Abstraktno in primerno za AI programe
  - Programi niso učinkoviti

# Algol

C.A.R. Hoare: "Here is a language so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors."

- Razvit na ETH Zuerich, 1958.
  - Skupina raziskovalcev iz Evrope in US
    - Allan J. Perlis, Peter Naur, John McCarthy, in drugi
  - Originalno: Algebrajski jezik, IAL
  - Izogibanje problemom opaženih v FORTRAN
- Prispevki
  - John Backus razvil BNF za specifikacijo ALGOL 58
  - Formalna semantika, dva načina prenosa parm.: kpv in kpr, bloki kode, vgnezdene funkcije, leksikalno definicijko območje
  - Študij imperativnih učinkov kpr na lambda račun
- Vpliv na:
  - Pascal, C, Modula, Java, C++, itd.

# Lisp

- Lisp razvit na MIT, John McCarthy, 1958
- Lisp je osnovan na lambda računu
  - Alonzo Church, 1930
  - Churchev članek o neodločljivem problemu
  - Velik vpliv je imel IPL
  - Seznami uporabljeni za predstavitev progr. in podatkov
- Novi koncepti
  - Avtomatsko delo s spominom, dinamični tipi, prevajalnik Lispa napisan v Lispu (self-hosting )
- AI jezik
  - Drugi najstarejši prog. jezik, ki je še vedno v uporabi
- Common Lisp, Scheme.

# Cobol

- Načrt naredil CODASYL, 1959
  - Conference/Committee on Data Systems Languages
  - Minulo delo Grace Murray Hopper
- Programski jezik za poslovne aplikacije
  - Imperativen in proceduralen
  - Od 2002, objektno-usmerjen
- Programski jezik sloni na angleščini
  - Okorna sintaksa
  - Cobolski programi delajo tudi danes!
  - Sintaksa je bila spremenjena

# Pascal

- Niklaus Wirth, ETH Zürich, 1970
- Zelo popularen jezik v Evropi
- Strukturiran programski jezik
  - Algolska družina
  - Podatkovne strukture, kazalci, polja, variabilni zapisi, zanke, procedure/funkcije, rekurzija, …
- Veliko različnih implementacij
  - VAX Pascal, Turbo Pascal, …
  - Delphi: zelo lepo programsko okolje
- Modula-2, Modula-3, Oberon

# Programski jezik C

- Dennis Ritchie, Bell Laboratories, 1972
  - C je bil načrtovan za Unix
  - Jezik B, osnovan na BCPL
- Gradniki jezika
  - Algolska družina
  - Še vedno en boljših jezikov (C++ =? C + iluzija objektov)
  - Blizu strojne opreme, sistemsko programiranje
- Polja in reference so tesno povezani
  - E1[E2] = *((E1)+(E2))
  - Aritmetika naslovov
- Ritchie je napisal:
  - “C is quirky, flawed, and a tremendous success.”

# Vsebina

- Organizacija predmeta
- Zgodovina programskih jezikov
- Programski modeli
- Koncepti programskih jezikov
- Meta jezik ML
- Metode predmeta

# Programski modeli

- Imperativni
- Funkcijski
- Objektno-usmerjeni
- Modularni
- Skriptni
- Logični

# Imperativni programski jeziki

- Osnovani na strojnem jeziku (zbirniku)
  - Instrukcije, procedure, zanke, vejitve
  - Program je sekvenca instrukcij
  - Rezultat se zgradi med izvajanjem instrukcij
  - Instrukcije spreminjajo vsebino spomina
- Dodana abstraktna sintaksa
  - Spemenljivke, IF, LOOP, FOR, PROCEDURE, FUNCTION
- Fortran
  - John Backus, 1956
- Pascal, C, Ada, Modula

# Funkcijski jeziki

- Osnovani na logiki
  - Lambda calculus, 1930, Alonzo Church
- LISP
  - John McCarthy, 1958
  - Implementacija lambda računa
- Meta jezik
  - ML, Rob Milner, 1970
- Haskel, Caml, Ruby, Scala
- Bolj primerni jeziki za učenje?
  - Strogi tipi
  - Program je dokaz
  - Elementi deklarativnega programiranja

# Objektno-orientirani jeziki

- Prvi OO jezik je Simula
  - 1960, Norwegian Computing Center
  - Programsko okolje za simulacijo
  - Vseboval je vse kar vsebujeo današnji OO jeziki
- Smalltalk je bil med najbolj popularnimi OO jeziki
  - Adele Goldberg, Xerox, Palo Alto, 1970
  - Vse je objekt!
  - Vsak objekt pripada enem razredu (ali večim)
  - Razred je prototip + razred je objekt!
  - Kompleksna hierarhija dedovanja
- C++, Java, Objective C, Eiffel, Ruby, Scala, C#
- Java
  - J.Gosling, B.Joy, G.Steele, G.Bracha
    The Java Language Specification
    https://docs.oracle.com/javase/specs/

# Modularni jeziki

- Imperativni pogled na module
  - Program je razdeljen na več programskih enot ali modulov.
  - Modul združuje kodo za implementacijo konceptualne entitete ali fizičnega objekta
    - Zaslon, gonilnik, naprava, podatkovna struktura, algoritmi, itd.
  - Modul temelji na vmesniku in implementaciji
    - Selektiven dostop in uporaba modulov iz drugih modulov
- V funkcijskih jezikih modul implementira ADT
  - Abstraktni podatkovni tip (ADT)
    - Skupne abstraktne podatkovne strukture in
    - Množico operacij za delo z APS.
  - Vmesnik/implementacija ≡ Signatura/struktura
- Programski jeziki:
  - ML, C, Pascal, Modula 2, Scala, Go, Erlang, Perl, Phyton, itd.

# Skriptni jeziki

- Programski jezik tesno povezan z nekim obstoječim sistemom
  - Okolja skriptnih jezikov: urejevalniki, Web strežnik operacijski sistem, igre, aplikacijski programi, itd.
- Zelo visoko-nivojski programski jeziki
  - Uporabljajo lepo definirane abstraktne strukture
    - Kolekcije, množice, slovarje, razpršilne tabele, itd.
  - Pogosto delujejo kot kontrolni jeziki sistema
  - Zelo počasni jeziki; nekaj 100-1000 krat počasnejši od C
  - Nekateri se razvijejo v splošno uporabni PJ (npr. Python)
- Primeri jezikov:
  - csh, bash, sed, AWK, Python, Perl, Tcl, Lua, JavaScript, PhP, JSP, itd.
  - Prvi skriptni jeziki so bili razviti okoli 1960 (lupine)

# Logično programiranje

- Programming in Logic, Robert Kowalski
- Predikatni račun
  - Hornovi stavki, unifikacija, resolucija
- Močen, enostaven in abstrakten programski jezik
- Rekurzija, vračanje (angl. back-tracking)
  - Deklarativni in proceduralni pomen Prologa
  - Operacija CUT (!)
- Prog. jezik za delo s podatkovnimi bazami
  - Datalog
- Implementacije: Sicstus, SB Prolog, SWI Prolog, itd.

# Vsebina

- Organizacija predmeta
- Zgodovina programskih jezikov
- Programski modeli
- Koncepti programskih jezikov
- Meta jezik ML
- Metode predmeta

# Koncepti programskih jezikov

- Programski jezik je orodje za načrtovanje in implementacijo računalniških programov
  - Programski jezik je lahko tekstovni, grafični, 2D, 3D, itd.
- Koncepti programskih jezikov so abstrakcije uporabljene pri predstavitvi strukture in obnašanja modeliranega sistema.

# Abstrakcija

- Abstrakcije v filozofiji
  - Specifična operacija intelekta, ki jo sestavlja (iz)ločitev (angl. detach) nekaterih lastnosti in ohranitev drugih lastnosti opazovane stvari (entitete).
- Aristotel
  - Konstruktiven mentalni proces povezan z relacijo med formami (idejami) in konkretnimi objekti.
  - Človeški um aktivno abstrahira ali ekstrahira forme objektov v katerih so realizirani.
- St. Thomas Aquinas
  - Dve vrsti abstrakcije
  - Um združi lastnosti, ki so bile med sabo ločene
  - Um loči lastnosti, ki so bile prej eno.

# Koncepti programskih jezikov

- Abstrakcije integrirane v programski jezik
  - Spremenljivke, vrednosti, vejitev, iteracija, kompozicija funkcij, rekurzija, funkcije, parametri, procedure, fun.višjega reda, polimorfizem, objekti, metode, razred, abstraktni razred, specializacija, agregacija, classifikacija, moduli, funktorji, itd.
- Abstrakcije definirajo programski model jezika
  - Imperativni jeziki: spremenljivke, zanke, procedure.
  - Funkcijski jeziki: funkcije, kompozicija funkcij, rekurzija.
  - Objektni jeziki: objekti, metode, razredi.

# Koncepti programskih jezikov

- Algoritmične in podatkovne abstrakcije
  - Algoritmične abstrakcije
    - Imperativne: sekvenca, zanke, vejitev, bloki
    - Funkcijske: rekurzija, polimorfizem, itd.
  - Podatkovne abstrakcije
    - Enostavne: integer, bool, real
    - Strukturirane: n-terke, seznami, polja, slovarji, množice, rekurzivne pod.strukture, itd.
  - Algoritmične in podatkovne abstrakcije
    - Objekti & razredi: enkapsulacija, classifikacija, dedovanje, itd.
    - Moduli: abstraktni podatkovni tipi, funkcije + abstraktni objekt

# Koncepti programskih jezikov

- Implementacija konceptov prog. jezikov
  - Function
    - Klic funkcije, prenos parametrov, aktivacijski zapisi
  - Rekurzija
    - Uporaba sklada aktivacijskih zapisov
  - Spremenljivke
    - Tabela simbolov, model vrednosti/referenc, imenski prostori.
  - Podatkovne strukture v spominu
    - Seznami, n-terice, zapisi, unije, objekti, razredi, moduli
  - Objekti/razredi
    - Statično/dinamično povezovanje, dedovanje.
  - Delo s spominom
    - Sklad, kopica (heap)
    - Čiščenje spomina

# Nekatere lastnosti abstrakcij

- Bližje strojni opremi, bolj enostaven jezik
  - Osnovni tipi: celo/realno število, znaki, nizi, itd.
  - Imperativni jeziki: zanke, funkcije, procedure, itd.
  - Običajno zelo učinkoviti jeziki
- Višje-nivojske abstrakcije
  - Objekti, razredi, moduli, funktorji.
    - Niso tako učinkoviti (dedovanje, din.povezovanje)
    - Uporabni za določene probleme (simulacija, protokoli, vmesniki, modularni sistemi, ...)
  - Zelo visok nivo abstrakcij, bolj usmerjen jezik
    - Jezik načrtovan za natančno določene vrste problemov
    - Jezik z abstraktnimi podat. strukturami, npr. tabele (PL/SQL)
    - Tokovi podatkov v porazdeljenem okolju (Spark)
    - Analiza velikih množic podatkov (Map-Reduce)

# Nekatere lastnosti abstrakcij

- Splošni programski jeziki so v sredini
  - Vsebujejo več različnih nivojev abstrakcij
  - Običajno niso med najbolj učinkovitimi PJ
  - Tipično so med bolj kompleksnimi jeziki
    - C++, C#, F#, Ocaml, Java
  - Splošno uporabni
    - Finančne analize, numerične analize (Fortran, Ocaml)
    - Sistemsko programiranje (C++, Ocaml)
    - Informacijski sistemi (C#, F#, Java)

# Abstrakcije

- Študij abstrakcij lahko izboljša učni proces študentov

JEFF KRAMER, IS ABSTRACTION THE KEY TO COMPUTING?
COMMUNICATIONS OF THE ACM April 2007/Vol. 50, No. 4

# IS ABSTRACTION THE KEY TO COMPUTING?

*Why is it that some software engineers and computer scientists are able to produce clear, elegant designs and programs, while others cannot? Is it possible to improve these skills through education and training? Critical to these questions is the notion of abstraction.*

*By* JEFF KRAMER

For over 30 years, I have been involved in teaching and research in computer science and software engineering. My teaching experience ranges from courses in programming, to distributed systems, distributed algorithms, concurrency, and software design. All these courses require that students are able to perform problem solving, conceptualization, modeling, and analysis. My experience is that the better students are clearly able to handle complexity and to produce elegant models and designs. The same students are also able to cope with the complexities of distributed algorithms, the applicability of various modeling notations, and other subtle issues.

# Zakaj študij konceptov PJ?

- Učimo se možnih načinov za izražanje idej.
  - Koncept PJ nudi orodje za izvedbo neke ideje
- Koncept PJ definiran z abstrakcijo in implementacijo.
  - Vsaka abstrakcija ima svojo ceno v implementaciji.
  - Rekurzija je draga; iteracija je lahko zelo kompleksna
  - Delo z objekti je dražje kot delo z enostavnimi PS
- Učimo se kako uporabiti dano abstrakcijo korektno.
  - Orodje je potrebno dobro poznati, da ga lahko učinkovito uporabljaš.
  - Formalno matematično ozadje programskega jezika.
  - Izkušnje z uporabo orodja in študij primerov programov.

# Vsebina

- Organizacija predmeta
- Zgodovina programskih jezikov
- Programski modeli
- Koncepti programskih jezikov
- Meta jezik ML
- Metode predmeta

# Lambda račun

- Lambda račun je temeljni formalizem za študij programskih jezikov
- Teorija programskih jezikov
  - Uporaba sklepanja s pravili
    - Izpeljava tipov izrazov
    - Evaluacija izrazov
  - Statiča in dinamična semantika PJ
  - Dokazovanje lastnosti jezikov (determinizm, strogi tipi, zaključitev izvajanja, itd.).
- Lambda račun je tema naslednjega predavanja

# Meta-Jezik (ML)

- Razvit nenamenoma pri delu na sistemu za automatsko dokazovanje izrekov
- Logika za izračunljive funkcije
  - Angl. Logic for Computable Functions, (LCF)
    - Automatski dokazovalnik izrekov
    - Robin Milner, Stanford 1970-71, Edinburgh 1972-1995
  - Teoretična osnova: Logic of Computable Functions
    - Dana Scott, LCF je verzija lambda računa
    - Sklepanje je izvajanje ML programa
- Meta-jezik (ML) za sistem LCF
  - ML uporabljen za:
    - zapis dokazov (korak za korakom),
    - zapis odločitev (taktike) pri dokazovanju in
    - implementacijo automatskega dokazovalnika.

# Programski jezik ML

- Zgodba uspeha iz funkcijske veje programskih jezikov
  - Standard ML, SML/NJ, Alice ML, Ocaml, Moscow ML, Mlton, MLKit, SML.NET, ...
- Raziskave povezane z ML
  - Hindley-Milner algoritem za preverjanje tipov
    - Si ga bomo ogledali
  - Polimorfični lambda račun (System F, Girard)
    - System F implementiran v okviru ML (LCF)
  - Razvoj sistemov tipov, meta-programski jeziki
    - CoffeeScript in TypeScript,
  - ML je imel zelo velik vpliv na razvoj PJ

# Objective Caml

- Objective Caml je referenčen jezik
  - Jedro Caml je čisti lambda račun.
  - Teoretično dobro preučen jezik.
  - Caml ima stroge tipe.
  - Imperativne gradnike.
  - Parametrični polimorfizem.
  - Lep model objektov.
  - Moduli in funktorji.
- Ocaml nudi več kot en sam programski model
  - Imperativen + funkcijski + objektno-usmerjen + modularen programski model

communications of the acm
| november 2 0 1 1 | vo l . 5 4 | n o. 1 1

DOI:10.1145/2018396.2018413

Article development led by acmqueue
queue.acm.org

**Why the next language you learn should be functional.**

BY YARON MINSKY

# OCaml for the Masses

Functional programming is an old idea with a distinguished history. Lisp, a functional language inspired by Alonzo Church's lambda calculus, was one of the first programming languages developed at the dawn of the computing age. Statically typed functional languages such as OCaml and Haskell are newer, but their roots go deep—ML, from which they descend, dates back to work by Robin Milner in the early 1970s relating to the pioneering Logic for Computable Functions (LCF) theorem prover.

# Kdo uporablja Ocaml v industriji?

- Facebook
- Jane Street
- Microsoft
- Docker
- Bloomberg
- Citrix
- Poglej: https://ocaml.org/learn/companies.html

# F#

- F# je podoben C#
  - .NET jezik
  - F# je osnovan na OCaml
  - C# je podoben C, Java in C++
- Značilne lastnosti
  - Enostavna sintaksa
  - Privzeto so vrednosti nespremenljive (konstante)
  - Izpeljava tipov in avtomatična posplošitev
  - Funkcije kot v funkcijskih jezikih
  - Izrazne podatkovne tipe
  - Ujemanje vzorcev
  - Asinhrono programiranje

# Predmeti, ki uporabljajo Ocaml

https://ocaml.org/learn/teaching-ocaml.html

**North America**

- Boston College - Computer Science I (CS 1101)
- Brown University - An Integrated introducion (CS 17/18) (along with Racket, Scala and Java)
- Caltech - Fundamentals of Computer Programming
- Columbia University - Programming Languages and Translators
- Cornell University - Data Structures and Functional Programming (CS 3110)
- Harvard University - Principles of Programming Language Compilation (CS153)
- Harvard University - Introduction to Computer Science II: Abstraction & Design (CS51)
- McGill University - Programming Languages and Paradigms (COMP 302)
- Princeton University - Functional Programming (COS 326)
- Rice University - Principles of Programming Languages (COMP 311)
- University of California, Los Angeles - Programming Languages (along with Python, Java) (CS 131)
- University of California, San Diego - Programming Languages: Principles and Paradigms (CSE130-a) (with Python, Prolog)
- University of Illinois at Urbana-Champaign - Programming Languages and Compilers (CS 421)
- University of Maryland (along with Ruby, Prolog, Java) - Organization of Programming Languages (CMSC 330)
- University of Massachusetts Amherst - Programming Languages (CMPSCI 631)
- University of Massachusetts - Programming Languages (CS691F)
- University of Minnesota Twin Cities — Advanced Programming Principles (CSCI 2041)
- University of Pennsylvania - Compilers (CIS341)
- University of Pennsylvania - Programming Languages and Techniques I (CIS120)
- University of Virginia - Programming Languages (CS 4610)

# Predmeti, ki uporabljajo Ocaml

https://ocaml.org/learn/teaching-ocaml.html

**Europe**

- Aarhus University - The compilation course (along with Java)
- Aix-Marseille University - Functional Programming
- Epita - Introduction to Algorithms (Year 1 & 2)
- ISAE/Supaéro - Functional programming and introduction to type systems
- Universidade da Beira Interior - Programming Languages and Compilers Design
- University Pierre & Marie Curie - Types and static analysis (5I555)
- University Pierre & Marie Curie - Models of programming and languages interoperability (LI332)
- University of Birmingham - Foundations of Computer Science (FOCS1112)
- University of Cambridge - Advanced Functional Programming (L28)
- University of Innsbruck - Programming in OCAML (SS 06)
- University of Rennes 1 - Programming 2 (PRG2)
- University of Wrocław - Functional Programming
- Université Paris-Diderot - Advanced Functional Programming (PFAV)
- Université Paris-Diderot - Functional Programming (PF5)

**Asia**

- Indian Institute of Technology, Delhi - Introduction to Computers and Programming (CSL 101) (along with Pascal and Java)

# Vsebina

- Organizacija predmeta
- Zgodovina programskih jezikov
- Programski modeli
- Koncepti programskih jezikov
- Meta jezik ML
- Metode predmeta

# Metode predmeta

- Koncepti programskih jezikov
  - Jezik je »konceptualno vesolje” (Perlis)
- Prva klasifikacija PJ na osnovi modela jezika
  - Funkcijski, imperativni, objektni, modularni
- Učenje več programskih jezikov
  - Lambda račun, ML, Python, Java, C
- Primerjava konceptov PJ
  - Kateri gradnik je bolje uporabiti v danem primeru
  - Izvedbe konceptov so lahko precej različne
- Implementacija konceptov PJ
  - Poznavanje implementacije omogoča pisanje učinkovitih programov

# Uporabljeni programski jeziki

- Lambda račun
  - Formalna osnova
  - Osnovni principi funkcijskih PJ
  - Lisp: prvi PJ osnovan na LC
- Meta-Jezik (ML)
  - Matematične osnove, strogi tipi, v osnovi funkcijski PJ
  - Ocaml vsebuje več modelov PJ (večino gradnikov)
- Jeziki za primerjavo: C, Java, Python
  - Python je trenutno najbolj popularen jezik
  - C je še vedno med najbolj popularnimi jeziki
  - Java uporablja čist objektno-usmerjen model
- Dodatni jeziki: C++, Fortran, Scala, Go
  - Predstavitev specifičnih konceptov PJ