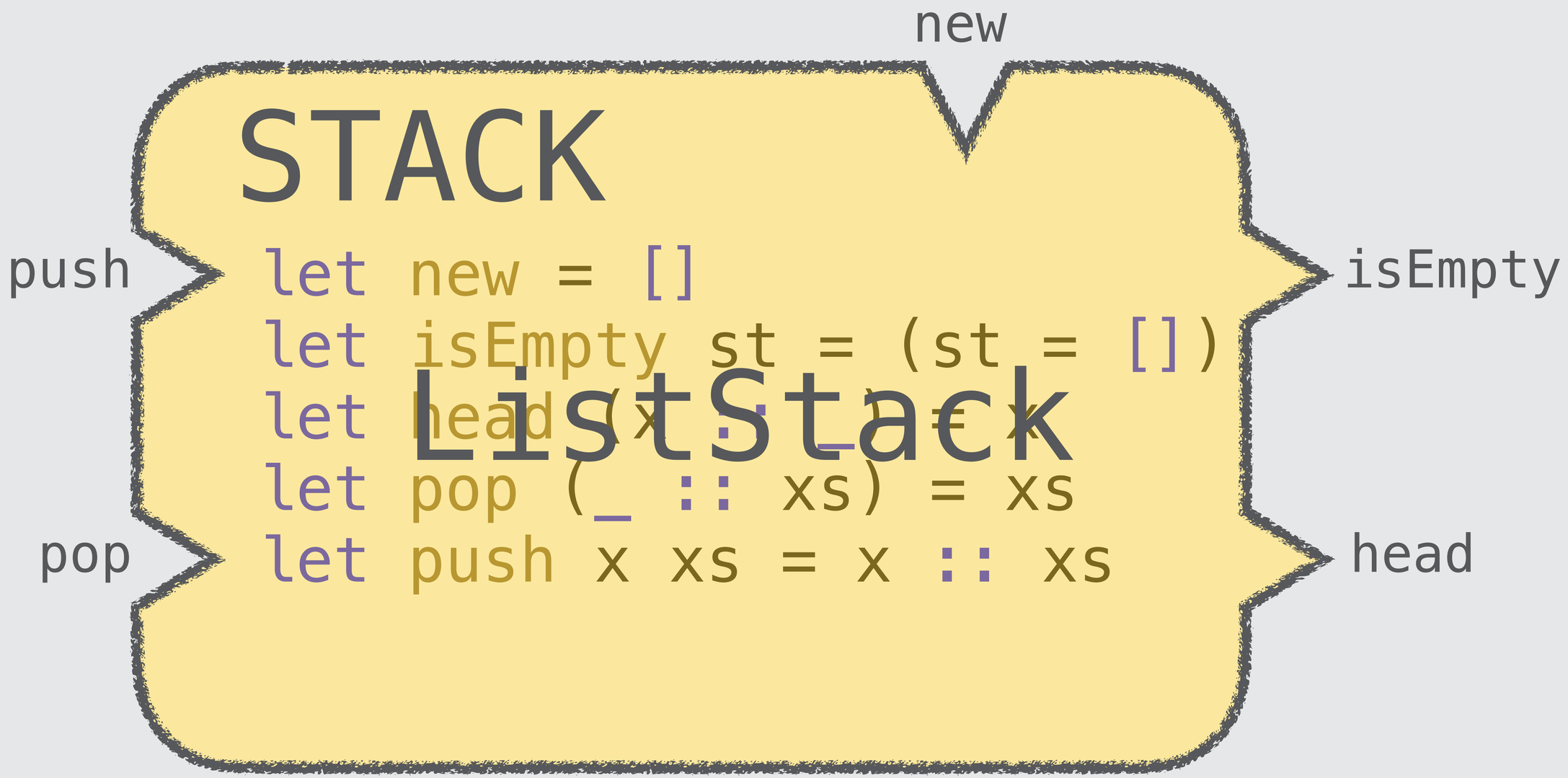
Logično programiranje

prejšnjič...

Ločili smo med **specifikacijami** in **implementacijami**



Ogledali smo si prednosti **abstraktnih** implementacij

varnost

– robustnost

– zagotavljanje invariant

lažji razvoj

– hkratno delo

– delitev odgovornosti

hitrost

– ločeno prevajanje

– lažje optimizacije

preglednost

– skrivanje detajlov

– organizacija kode

Videli smo, kakšno **podporo** abstrakciji nudijo različni **jeziki**

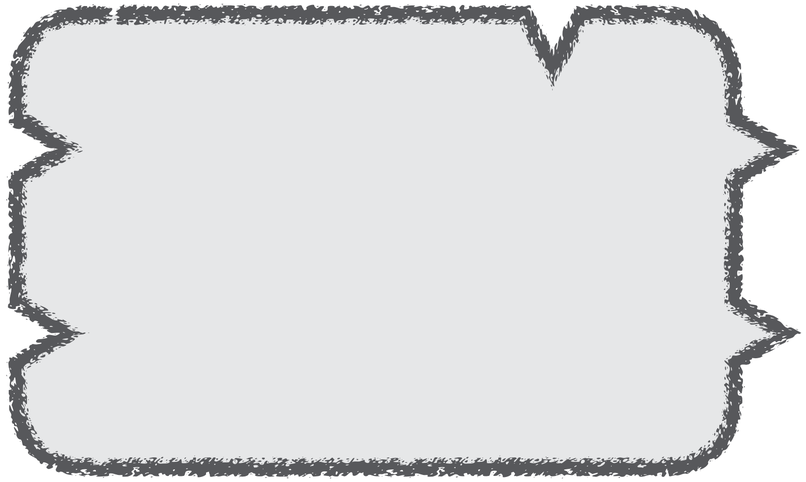
	specifikacija	**implementacija**	**abstrakcija**
Java	vmesniki	razredi	`private` `public` `protected`
C	`*.h`	`*.c`	❌
Python	❌	`*.py`	`_ime_metode` 😐
OCaml	signature `*.mli`	moduli `*.ml`	✅

Programe lahko pišemo **neodvisno od implementacije**

ListStack

ArrayStack

DFS



Hornove formule

Logično programiranje je nov pristop k programiranju

proceduralno programiranje

Iskani rezultat dosežemo z **zaporedjem ukazov**.

funkcijsko programiranje

Iskani rezultat opišemo s **sestavljanjem izrazov**.

logično programiranje

Iskani rezultat določimo z **logičnimi formulami**.

Omejili se bomo na bolj obvladljive **Hornove formule**

$$\forall x_1, \ldots, x_m \,.\, (\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \Rightarrow \psi)$$

$$\varphi, \psi ::= P(t_1, \ldots, t_n)$$

$$t ::= x \mid f(t_1, \ldots, t_n)$$

Alfred Horn, 1918–2001

$$\forall x\, y\, z\,.\,($$
$$\text{otrok}(x, y) \wedge \text{otrok}(y, z) \wedge \text{zenska}(z)$$
$$\Rrightarrow \; ???(x, z)$$
$$)$$

$$\forall x\, y\, z\,.$$

$$\text{otrok}(x, z) \wedge \text{otrok}(y, z)$$

$$\wedge\ \text{zenska}(x) \wedge \text{zenska}(y)$$

$$\Rightarrow\ ???(x, y)$$

S Hornovimi formulami lahko izražamo **funkcije**

$$\forall n \,.\, n + 0 = n$$

$$\forall k, m \,.\, k + m^{+} = (k + m)^{+}$$

$$\forall n \,.\, \text{vsota}(n, 0, n)$$

$$\forall k, m, n \,.\, \text{vsota}(k, m, n) \Rightarrow \text{vsota}(k, m^{+}, n^{+})$$

$$x \cdot y = z \iff \text{produkt}(x, y, z)$$

Prolog

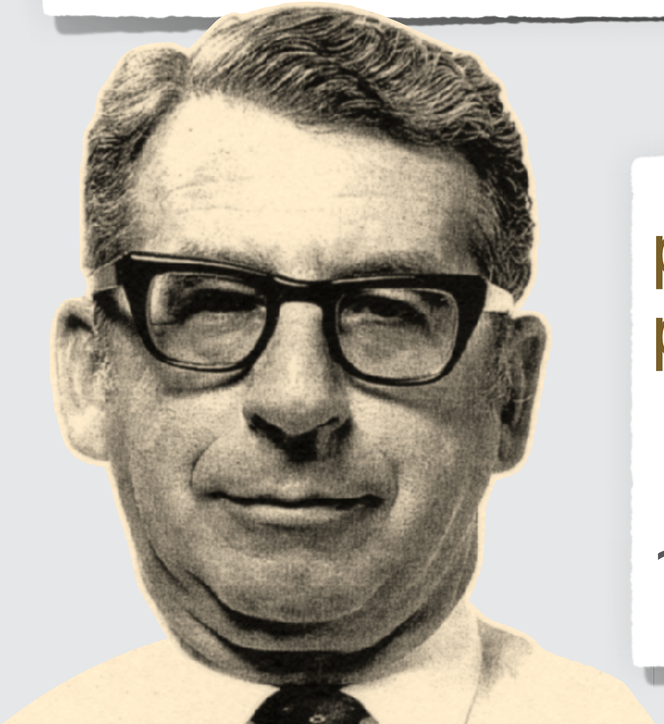
Pisanje formul in poizvedb v Prologu

$\forall x$. produkt$(x,0,0)$

$\forall x, y, z, w$. produkt$(x, y, w) \wedge$ vsota(x, w, z)

$\Rightarrow$ produkt(x, y^+, z)

? $\exists p, q$. produkt$(p, q,4)$



```
prod(_, zero, zero).
prod(X, succ(Y), Z) :-
     prod(X, Y, W),
     sum(X, W, Z).
?- prod(P, Q, succ(succ(succ(succ(zero))))).
```

fakulteta

v Prologu

Prolog v pravilih omogoča tudi **disjunkcijo**

$$\varphi_1 \vee \varphi_2 \Rightarrow \psi$$

$$\varphi_1 \Rightarrow \psi$$

$$\varphi_2 \Rightarrow \psi$$

```
grandmother(X, Z) :-
    mother(X, Y), father(Y, Z);
    mother(X, Y), mother(Y, Z).
```

seznami

v Prologu

S Hornovimi formulami izrazimo **induktivno podane relacije**

$(\eta, c) \mapsto (\eta', c')$

$$\frac{\eta \mid e \hookrightarrow n}{(\eta, x := e) \mapsto (\eta[x \mapsto n], \texttt{skip})} \qquad \frac{(\eta, c_1) \mapsto (\eta', c_1')}{(\eta, c_1; c_2) \mapsto (\eta', c_1'; c_2)} \qquad \frac{}{(\eta, \texttt{skip}; c_2) \mapsto (\eta, c_2)}$$

$$\frac{\eta \mid b \hookrightarrow \texttt{true}}{(\eta, \texttt{if}\, b\, \texttt{then}\, c_1\, \texttt{else}\, c_2\, \texttt{end}) \mapsto (\eta, c_1)} \qquad \frac{\eta \mid b \hookrightarrow \texttt{false}}{(\eta, \texttt{if}\, b\, \texttt{then}\, c_1\, \texttt{else}\, c_2\, \texttt{end}) \mapsto (\eta, c_2)}$$

$$\frac{\eta \mid b \hookrightarrow \texttt{true}}{(\eta, \texttt{while}\, b\, \texttt{do}\, c\, \texttt{done}) \mapsto (\eta, c; \texttt{while}\, b\, \texttt{do}\, c\, \texttt{done})} \qquad \frac{\eta \mid b \hookrightarrow \texttt{false}}{(\eta, \texttt{while}\, b\, \texttt{do}\, c\, \texttt{done}) \mapsto (\eta, \texttt{skip})}$$

COMM

v Prologu

Iskanje dokaza

V logičnem programiranju **iščemo dokaz** cilja iz danih predpostavk

predpostavke

$$H_1 = \forall x_1, \ldots, x_k \,.\, (\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_\ell \Rightarrow \psi)$$

$$H_2 = \cdots$$

$$\vdots$$

$$H_k = \cdots$$

$$? \; G = \exists y_1, \ldots, y_m \,.\, p(t_1, \ldots, t_n)$$

poizvedba/cilj

Poenostavljen postopek reševanja, če **nimamo spremenljivk**

iskanje

Zaporedoma gremo čez formule in najdemo **prvo** formulo $H = \varphi_1 \wedge \cdots \wedge \varphi_m \Rightarrow G$, ki imajo **sklep enak cilju** G.

sestopanje

Rekurzivno rešimo vse predpostavke $\varphi_1, \ldots, \varphi_m$. Če kakšne ne moremo rešiti, **poiščemo naslednjo** združljivo formulo H'.

$$X \wedge Y \Rightarrow C \qquad A \wedge B \Rightarrow C$$

$$X \Rightarrow B \qquad A \Rightarrow B \qquad A$$

$$? \quad C$$

Splošen postopek reševanja $\exists x_1, \ldots, x_m \,.\, P(t_1, \ldots, t_n)$

iskanje

Zaporedoma gremo čez formule in izberemo **prvo** formulo oblike
$\forall y_1, \ldots, y_k \,.\, \varphi_1 \wedge \cdots \wedge \varphi_\ell \Rightarrow P(u_1, \ldots, u_n)$.

združevanje

Določimo **najbolj splošne** vrednosti σ spremenljivk $x_1, \ldots, x_k$ in $y_1, \ldots, y_m$,
da velja $\sigma(t_1) = \sigma(u_1), \ldots, \sigma(t_n) = \sigma(u_n)$.

sestopanje

Rekurzivno rešimo $\sigma(\varphi_1), \ldots, \sigma(\varphi_\ell)$, kjer spremenljivke **kvantificiramo eksistenčno**.
Če kakšna predpostavka nima rešitve, **poiščemo naslednjo** združljivo formulo H', **če obstaja**.

$$\mathsf{sodo}(0)$$

$$\forall x \,.\, \mathsf{sodo}(x) \Rightarrow \mathsf{liho}(x^{+})$$

$$\forall y \,.\, \mathsf{liho}(y) \Rightarrow \mathsf{sodo}(y^{+})$$

$$? \quad \mathsf{sodo}(0^{++})$$

$$\mathsf{sodo}(0)$$

$$\forall x \,.\, \mathsf{sodo}(x) \Rightarrow \mathsf{liho}(x^+)$$

$$\forall y \,.\, \mathsf{liho}(y) \Rightarrow \mathsf{sodo}(y^+)$$

$$? \quad \exists z \,.\, \mathsf{liho}(z)$$

$$\forall x \,.\, \mathrm{sodo}(x) \Rightarrow \mathrm{liho}(x^+)$$

$$\forall y \,.\, \mathrm{liho}(y) \Rightarrow \mathrm{sodo}(y^+)$$

$$\mathrm{sodo}(0)$$

$$? \quad \exists z \,.\, \mathrm{liho}(z)$$

prihodnjič...

Spoznali bomo programiranje **z omejitvami**

```
?- [P,Q] ins 1..99, P * Q #= 42, label([P, Q]).
P = 1, Q = 42 ;
P = 2, Q = 21 ;
P = 3, Q = 14 ;
P = 6, Q = 7 ;
P = 7, Q = 6 ;
P = 14, Q = 3 ;
P = 21, Q = 2 ;
P = 42, Q = 1.
```

Reševali bomo **sudoku**