DELO S PODATKI

Algoritem

Algoritem je postopek za reševanje problema z računalnikom. Zapisan je lahko v naravnem jeziku, s psevdokodo, z diagramom poteka ali kot program/funkcija v programskem jeziku.
Pri algoritmu nas najbolj zanimata **ustavljivost** in **pravilnost**: ali se algoritem pri vseh veljavnih vhodih za podani problem ustavi in ustvari pričakovani izhod. Seveda za isti problem obstaja več algoritmov, zato najraje izberemo tistega, ki **porabi najmanj časa**, **najmanj prostora** (posebej pri velikih zbirkah podatkov), ki je **razumljiv** (če pri porabi časa in prostora ni bistvenih razlik, raje uporabimo lažje razumljive algoritme) in **prilagodljiv** (kako lahko algoritem prilagodimo za večje ali splošnejše različice probleme).

Programski jezik

Programski jeziki so posebni jeziki, s katerimi računalniku podajamo navodila.
Računalnik razume zgolj strojni jezik, zato se mora naš program pred/med izvajanjem pretvoriti v enakovrednem program v strojnem jeziku:

- **prevajalnik** pretvorbo v strojni jezik v celoti izvrši pred izvajanjem,
- **tolmač** program bere ukaz za ukazom in sproti podaja strojne ukaze.

Temeljni elementi vsakega programskega jezika so **izrazi** (zaporedje simbolov z neko vrednostjo), **spremenljivke** (poimenovan prostor v pomnilniku, ki hrani poljubno vrednost), **zanke** (del programa, ki ga računalnik izvede večkrat zaporedoma), **tabele** (zaporedje podatkov, ki imajo soroden pomen), **funkcije** (zaporedje stavkov, ki ima neko ime, lahko tudi parametre) in **nizi** (zaporedje znakov).

Primer izseka kode iz C# za ponazoritev

```
static void Main()
{
	int a = 5;                          // spremenljivka
	int b = a + 2;                      // izraz
	string boi = "primer";
	while (b < 10) {                    // zanka
		b += 1;
	}
	int[] tabela = { 1, 2, 3, 4, 5 };   // tabela
	char[] niz = boi.ToCharArray();     // niz
}
static int NaTri(int stevilo)           // funkcija
{
	return stevilo * stevilo * stevilo;
}
```

Poznamo več načinov programiranja:

- **strukturirano**: jasno in učinkovito programsko kodo napišemo tako, da problem razbijemo na zaporedje manjših problemov, vsakega od teh pa rešimo s funkcijo,
- **objektno**: podatke in funkcije združimo v samostojno enoto, ki ji rečemo objekt; namesto posredovanja podatkov funkcijam jih naložimo v objekt in nato kličemo funkcije objekta, ki na podlagi podatkov v objektu ustvarijo želeni rezultat,
- **dogodkovno**: tok programa določajo različni dogodki (kot so npr. klik na ikono), izhodi senzorjev, sporočila; prevladuje pri GUI programih,
- **funkcijsko**: programe sestavljamo iz matematičnih funkcij, izogibamo se imperativnem programiranju, določanju stanj programa in mutabilnim podatkom,
- **logično**: programe sestavljamo iz logičnih stavkov,
- **skriptno**: povezujemo že obstoječe komponente, ki so ponavadi narejene v sistemskem programskem jeziku (npr. skripte v UNIX lupinah).

Podatkovna baza

Podatkovna baza je **model realnosti** v računalniku, ki vsebuje preučevane lastnosti in povezave, ki nas zanimajo. Njegovo delovanje ustreza realnim razmeram.

ANSI zahteve za podatkovno bazo so:

1) podatki so v bazi **povezani** in **urejeni** po določenem vrstnem redu,
2) baza je urejena tako, da lahko podatke v njej **istočasno uporablja** več uporabnikov,
3) isti podatki se v bazi **ne ponavljajo**,
4) baza je **shranjena** v računalniku.

Sistem za upravljanje podatkovnih baz (oz. SUPB, ang. DBMS) omogoča funkcionalno obdelavo podatkov. Naloge funkcionalne obdelave podatkov so:

- zagotoviti **pravilnost** in **ažurnost** podatkov,
- **sočasno nuditi** podatke vsem uporabnikom brez ogrožanja celovitosti baze,
- **posredovati** podatke takrat, ko jih uporabniki potrebujejo,
- omogočiti vsem uporabnikom **dostopnost** do podatkov, ki jih potrebujejo,
- posredovati podatke o tem, **kaj se je** zgodilo (zgodovina) in **kaj se lahko** zgodi (napovedi).

Podatkovna baza je lahko **organizirana** kot:

- **centralizirana baza**: upravlja se z enim sistemom,
- **porazdeljena baza**: več računalnikov na različnih lokacijah, povezanih v omrežje, upravlja se z več sistemi za upravljanje.

Podatkovni model je strukturiran mehanizem za opis realnosti s podatki. Vsebuje množico pravil za **organizacijo podatkov** (strukturo) in **operacije nad podatki**. Podatkovni model lahko naredimo na dva principa:

- **princip »od spodaj«** (**podatkovna analiza**): iz obstoječih dokumentov izluščimo atribute in naredimo logični model;
- **princip »od zgoraj«** (**analiza realnega procesa**):
  1. analiza realnega procesa → **globalni model**
  2. določitev entitet, atributov, razmerij, številnosti → **konceptualni model (E-R)**
  3. vsako entiteto podrobno opišemo z njenimi atributi tako, da je vsaka entiteta v modelu enolična določena → **logični model**
  4. zgradimo fizično podatkovno bazo → **fizični model**

Osnovni konstrukti modela E-R so:

- **entiteta** (objekti/osebki iz realnega sveta) in entitetni tip (množica podobnih entitet),
- **atribut** (lastnost neke entitete) in tip podatkovnega elementa (število, datum …),
- **primarni ključ** oz. entitetni identifikator (enolično določa posamezni zapis),
- **številnost** (**kardinalnost**) **razmerja** pove, koliko primerkov ene entitete nastopa v povezavi z enim primerkom druge entitete.
  - **1 : n** → v prvi relaciji se primerek pojavi enkrat, v drugi pa n-krat
  - **1 : 1** → v obeh relacijah se primerek pojavi le enkrat
  - **m : n** → v obeh relacijah se pojavi n-krat; razrešimo z dvema 1:n

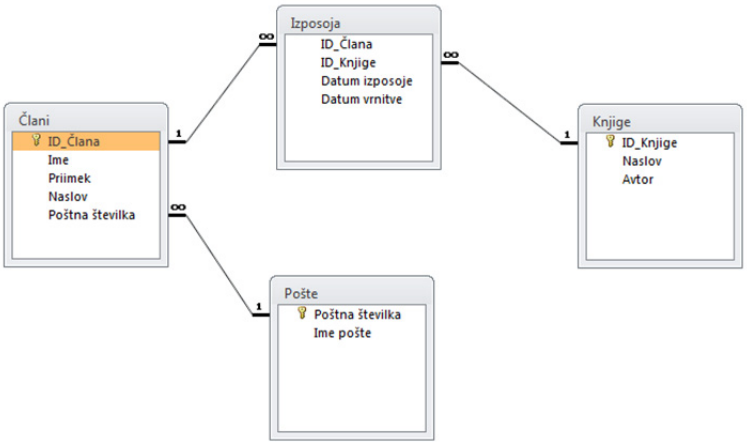
Zakaj je **relacijski podatkovni model** dober? Odlikujejo ga naslednje lastnosti:

- formalno je **definiran** in osnovan na matematičnih strukturah – **relacijah**,
- ne vsebuje elementov fizičnega shranjevanja podatkov, s čimer je zagotovljena **podatkovna neodvisnost**,
- relacije so predstavljive s **tabelami**, ki so človeku dobro razumljive,
- vsaka entitetna množica je predstavljena z **eno ali več relacijami** = tabelami.

Stolpci v tabeli (= relaciji) predstavljajo **atribute** entitetne množice. **Vrstice** predstavljajo **zapise** entitetne množice. Vsak atribut ima svojo **domeno** = zalogo vrednosti. **Relacijska shema** je naslovna vrstica tabele.

Poizvedbe so orodje, s katerim dobimo poglobljene informacije. Zberemo podatke, ki jih potrebujemo in jih uredimo tako, kot želimo. Rezultat poizvedbe je tabela z želenimi podatki. Tabela je začasna, saj jo SUPB pri vsaki poizvedbi ustvari na novo z najnovejšimi podatki. Pogosto jih uporabljamo, če želimo prikazati le **določena polja tabele**, če želimo **izbrati le določene zapise** (**filtriranje**), če želimo **rezultate razvrstiti** po določenem vrstnem redu (**razvrščanje**) ali če želimo **podatke iz več tabel združiti skupaj**.

Obrazci ljudem olajšajo delo s podatki. Npr. za izposojo v knjižnici knjižničar ne bo podatkov direktno vpisoval v bazo, ampak bo uporabil obrazec.



Poročila uporabimo, ko imamo že izdelane poizvedbe. Namen poročil je ustvariti človeku prijazne dokumente za branje. Podatke lahko združujemo v skupine, oblikujemo dokument za lažje branje, poudarjamo določene podatke itd.

Z **referenčno integriteto** med relacijama zagotavljamo, da eni entiteti ne moremo določiti neobstoječe entitete iz druge relacije; zapisa iz glavne relacije ne moremo odstraniti, dokler ne odstranimo vseh povezanih zapisov. Referenčna integriteta torej varuje povezane podatke pred nehotenim spreminjanjem in brisanjem.

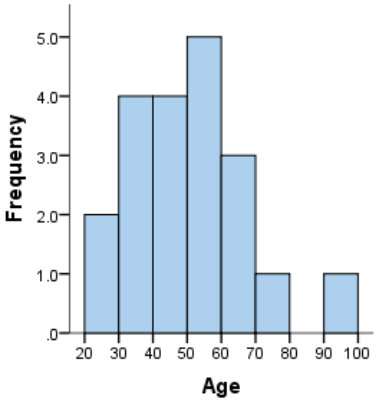
Preglednica

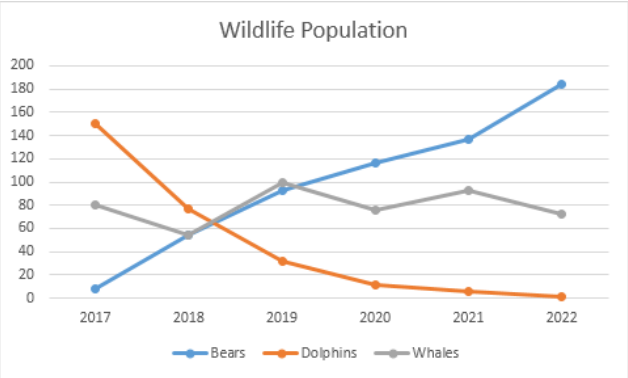
Glavni koncept vsake **preglednice** je **mreža celic**, ki jo imenujemo **delovni list**, preglednica pa jih lahko vsebuje več, zato jo včasih imenujemo tudi delovni zvezek. Celice delovnega lista so združene v **vrstice** in **stolpce**. Stolpci so označeni s črkami od A do Z, nato od AA do AZ in tako naprej. Vrstice so oštevilčene od zgoraj navzdol. Pri preglednicah lahko uporabljamo **formule**, vgrajene **funkcije** in **makro skripte**. Podatke v preglednici lahko razporedimo npr. po abecednem vrstnem redu, naraščajoče/padajoče itd.

Pri **analizi »kaj-če«** v Excelu lahko za raziskovanje različnih rezultatov v eni ali več formulah uporabimo različne nabore vrednosti. Npr. predvidevamo proračun glede na različne vrednosti letnega prihodka, št. zaposlenih itd. Z upraviteljem scenarijev ustvarimo **scenarij**: nabor vrednosti, ki jih Excel shrani in jih lahko samodejno nadomesti v celicah.

GRAFIKONI

- **histogram**: omogoča odkrivanje in prikaz dejanske frekvenčne porazdelitve množice neprekinjenih podatkov; neprekinjene podatke razdelimo v intervale oz. razrede in tako dobimo osnovno porazdelitev; uporabljamo jih npr. za prikaz porazdelitve ocen, doseženih rezultatov itd.





- **lomljenka**: uporabljamo za prikaz neprekinjenih sprememb skozi čas; npr. za opazovanje sprememb temperature v mesecu juniju, spremembe populacije divjih živali itd.
- **krožni grafikon**: prikažemo odnose med deli in celoto; npr. za prikaz verske sestave prebivalstva države.

Tehnologije znanja

S **tehnologijami znanja** mislimo katerokoli tehnologijo, s katero si pomagamo pri delu z znanjem. Osnovni načini upravljanja z znanjem so:

- umetna inteligenca,
- ekspertni sistemi,
- strojno učenje,
- robotika.

Poznamo več vrst znanja:

- **deklarativno**: dejstva, pravila, odgovori in druge predstave, pridobljene z neposrednim opazovanjem realnosti in od drugih ljudi (knjige, slike, pogovor …),
- **proceduralno**: potrebujemo ga, da neko opravilo uspešno izvedemo,
- **strateško**: potrebujemo ga, ko se odločamo, katero znanje uporabiti.

Proces upravljanja z znanjem:

1. **odkrivanje** znanja: raziskovanje, beleženje rezultatov itd.,
2. **formaliziranje** znanja: zajamemo, zapišemo, shranimo znanje,
3. **uporaba** znanja: pravim ljudem posredujemo pravo znanje.

Pri **procesu odločanja** izbiramo med več možnostmi – variantami. Izbrana varianta naj čimbolj ustreza danim ciljem. **Faze odločitvenega procesa** so:

1. **opredelitev problema**: natačno opišemo problem in opredelimo cilje reševanja,
2. **določitev kriterijev**: glede na zastavljene cilje določimo kriterije, po katerih bomo ocenjevali posamezne variante; kriteriji naj bodo merljivi; prepričamo se, da nismo spregledali kriterijev, ki pomembno vplivajo – načelo polnosti,
3. **strukturiranje kriterijev**: kriterije razporedimo glede na pomembnost oz. jim določimo uteži; pri večparametrskem odločanju zgradimo drevo kriterijev – združimo kriterije, ki so vsebinsko povezani (ponavadi ne več kot 3),
4. **določitev zaloge vrednosti za kriterije**: vsakemu kriteriju določimo vrednosti, ki jih lahko zavzame; določimo le toliko različnih vrednosti posameznega parametra, da lahko razlikujemo bistvene razlike med variantami, višje po drevesu naj število vrednosti za posamezni (združeni) kriterij postopoma raste,

5. **določitev funkcije koristnosti**: oblikujemo ustrezna pravila oz. uporabimo uteži, iz funkcije je nato razvidno, kateri parametri so pomembni in ali pri danem problemu nastopajo **izločitveni kriteriji**,
6. **opis variant z vrednostmi po kriterijih**: variante opišemo z vrednostmi po osnovnih parametrih, pazimo na čim večjo popolnost podatkov,
7. **vrednotenje variant**: končno oceno določi program glede na model, varianta z najvišjo oceno je praviloma najboljša, tako tudi razvrstimo variante
8. **analiza variant**: ugotavljamo razloge za rezultat vrednotenja, jih primerjamo,
9. **odločitev**: končna izbira.

Pri ročni metodi **Abacon** uporabljamo obrazec z dvema stolpcema. V levega vpisujemo parametre, v desnega pa vrednosti parametrov. Vrednosti parametrov morajo biti urejene od najslabših do najboljših, od leve proti desni. Merske lestvice pa lahko poenotimo tako, da pripišemo parametrom vrednost od 0 do 100.

Pri metodi z **elektronsko preglednico** je postopek podoben metodi Abacon, pomembnost parametrov določimo s pomočjo uteži v drugem stolpcu. Uteži naj bodo normalizirane (njihova vsota naj bo enaka enoti, npr. 100). Ocena posamezne variante je potemtakem utežena vsota vrednosti vseh parametrov.

Pri metodi z **lupinami ekspertnega sistema** uporabimo računalniške programe kot so DEXi. Najprej zgradimo model, ki je sestavljen iz drevesa kriterijev in odločitvenih pravil. Z odločitvenimi pravili pa določamo vrednosti parametrov v vozliščih drevesa vse do korena, ki predstavlja končno oceno predmeta odločanja. Z odločitvenimi pravili lahko upoštevamo, da je vpliv nekega parametra na končno oceno variante odvisen od njegove vrednosti, kar pomeni, da uteži niso konstante. Interpretacija rezultatov, ki jo mora narediti človek, poteka od korena preko vozlov do listov odločitvenega drevesa. Za bolj nazorno predstavo pa lahko z DEXi-jem naredimo različne grafične predstavitve: od primerjav variant po posameznih parametrih, do radarskih diagramov, kjer lahko grafično primerjamo med seboj variante po več parametrih.