1. Algoritmi

1. Pojam algoritma i osnovna svojstva algoritama

Riječ algoritam susreće se sve više u raznim prilikama svakodnevnog života. Moglo bi se reći da za to postoji i stanovito opravdanje. Naime, obavljanje različitih poslova svodi se na izvođenje pojedinih osnovnih operacija zadanim redoslijedom. Pojedine vještine koje dijete stječe učenjem i oponašanjem odraslih mogu se svesti na svojevrsne algoritamske postupke. Primjerice, postupak prelaženja ulice svodi se na provođenje sljedećih koraka: najprije pogledati na lijevo, zatim pogledati na desno; ako se s bilo koje strane približava vozilo, pričekati da ono prođe; kada se u blizini ne vidi vozilo, brzo prijeći ulicu. Upute za uporabu pojedinih kućanskih aparata mogu se također smatrati svojevrsnim algoritmima u kojima se po točkama opisuje kako se obavlja pojedina operacija.

Riječ **algoritam** dolazi od imena arapskog matematičara Muhamed ibn Musa al Horezmi (u prijevodu s arapskog jezika to bi značilo: Muhamed sin Muse iz Horezma) koji je živio u IX. stoljeću. On je razradio i u posebnoj knjizi opisao pravila za provođenje aritmetičkih operacija s brojevima zapisanim u dekadskom obliku. Arapski original te knjige je izgubljen ali postoji prijevod te knjige na latinski iz XII. stoljeća. U tom prijevodu ispred svakog pravila piše "Dixit Algorizmi" ("Algorizmi je govorio", tj. zadnji dio imena al Horezmi pretvoren je u Algorizmi). U drugim latinskim tekstovima ime je pretvoreno u Algorithmus (u prijevodu s latinskog na hrvatski jezik: Algoritam). S vremenom je zaboravljeno da je Algoritam ime čovjeka i naslov nad pravilima "Algoritam je govorio" pretvorio se u "algoritam glasi". Na taj način su se pravila počela nazivati algoritmima.

U prvo su vrijeme algoritmima nazivana samo pravila računanja s brojevima zapisanim u dekadskom sustavu da bi se kroz naredna stoljeća taj naziv počeo upotrebljavati za pravila obavljanja raznovrsnih zadataka. U dugom vremenskom

razdoblju, sve do sredine dvadesetog stoljeća, taj su naziv upotrebljavali samo matematičari, no pojavom računala, pojam se rasprostranio, najprije u područje računarstva, a zatim i u druge djelatnosti, gdje se pod pojmom algoritma podrazumijevaju točno opisana pravila za postizanje željenog rezultata.

U matematici se i nadalje algoritmi izučavaju na sustavan teorijski način. Matematička teorija algoritama bavi se mogućim oblicima algoritama i istražuje svojstva samih algoritama. S uporabnog stanovišta algoritam nije objekt izučavanja već se algoritmi, kao što je već i rečeno, koriste za jasno *utvrđivanje pravila dostizanja nekog postavljenog cilja*. Međutim, potrebno je ipak utvrditi neka osnovna svojstva algoritama i ustanoviti kada se neki postupak može nazvati algoritmom.

Prije toga korisno će biti proučiti nekoliko primjera.

Primjeri algoritama

Primjer 1.1. Kuharski recepti su jedna vrsta algoritama. Evo recepta za pripremanje brodeta:

Sastojci:

1 kg morske ribe
2 žlice maslinovog ulja
4 glavice luka
3 rajčice
2 lovorova lista
grančica ružmarina
1 režanj bijelog luka
pire od rajčica
čašica vinjaka
1 dl crnog vina
sol, papar, ocat

Priprema:

Na maslinovom ulju propržiti luk, dodati na komade razrezanu ribu i razrezanu rajčicu. Uliti malo vode i sve pirjati. Posoliti, popapriti, dodati začine, zgnječeni režanj bijelog luka, pire od rajčica i vinjak. Nastaviti s pirjanjem i paziti da se riba ne raspadne. Pri kraju dodati ocat i crno vino. □

Primjer 1.2. Uputa za sijanje nekih povrtnih kultura može, na primjer, glasiti ovako. Prije sijanja grabljama usitniti i poravnati zemlju. Uz pomoć dva kolčića i uzice izdupsti motičicom ravne brazde na razmaku od 30 do 35 centimetara. U brazde na razmaku od 15, 20 ili 30 centimetara (ovisno o vrsti biljke) staviti 10 do 15 sjemenki. Zasuti sjemenke slojem zemlje debljine 1 centimetar. □

Primjer 1.3. Zbrajanje dvaju ili više decimalnih brojeva (u dekadskom sustavu zapisani brojevi čiji su cijeli i razlomljeni dijelovi razdvojeni decimalnom točkom) ne predstavlja ni pučkoškolcima nikakav problem. Međutim, u tom sasvim uobičajenom poslu primjenjuje se sljedeći, ne baš jednostavni, algoritam:

Napisati decimalne brojeve jedan ispod drugog tako da znamenke iste težine budu u istom vertikalnom stupcu (decimalne točke su pritom jedna ispod druge). Ako kod nekih brojeva lijevo ili desno nedostaju znamenke, zamisliti da su na tim mjestima zapisane nule. Zatim zbrajati znamenke u istom stupcu i pribrajati im eventualni prijenos s nižeg mjesta. Počinje se zbrajanjem zdesna (brojevnim mjestom najmanje težine) i smatra se da je tu prijenos jednak nuli. Postupak se zaustavlja kada se dosegne krajnji lijevi stupac.

Tako je, primjerice:

$$\begin{array}{rr} & 3.\,14159 \\ & 0.\,52 \\ + & 27. \\ \hline & 30.\,66159 \end{array}$$

□

* * *

Svojstva algoritama

Iz gornja tri primjera mogu se uočiti neka važna svojstva algoritama.

U prvom redu, uz svaki algoritam moraju jasno biti definirani početni *objekti*[1] nad kojima se obavljaju operacije. U primjeru 1.1 to su sastojci za kuhanje brodeta, u primjeru 1.2 sjemenke, a u primjeru 1.3 decimalni brojevi. Kao ishod provođenja algoritma pojavljuju se završni objekti ili *rezultati*. U primjeru 1.1 to je jelo brodet, u primjeru 1.2 zasijana gredica nekog povrća, a u primjeru 1.3 decimalni broj dobiven zbrajanjem.

Algoritam mora biti sastavljen od *konačnog broja* koraka koji ukazuju na slijed operacija koje treba obaviti nad početnim objektima kako bi se dobili završni objekti ili rezultati. Svaki korak opisuje se *instrukcijom*[2]. Obavljanje algoritma naziva se **algoritamskim procesom**[3]. Tijekom odvijanja algoritamskog procesa i postupne izgradnje završnog objekta mogu se pojaviti i neki međurezultati.

Za obavljanje algoritma potreban je **izvoditelj algoritma**, koji *razumije algoritam* i znade *točno obaviti svaki korak* algoritma. Trajanje algoritamskog procesa određeno je brzinom kojom izvoditelj obavlja korake algoritma.

[1] object od latinskog *objectus* — postavljanje pred što, zapreka (u smislu: stvar koja se može dotaknuti, opipati)

[2] instrukcija, od latinskog *instruere* — narediti, poučavati

[3] proces, od latinskog *processus* — napredak, rastenje

Neki algoritmi su *specijalizirani*[4] i mogu se primijeniti samo na pojedine početne objekte. Druga, *općenita vrsta* algoritama dozvoljava različite vrijednosti početnih objekata. Primjer specijaliziranog algoritma je recept za pripremanje brodeta, dok je primjer općenitog algoritma uputa za zbrajanje decimalnih brojeva. Taj drugi, općeniti algoritam dozvoljava zbrajanje bilo kojih decimalnih brojeva.

Kod općenitih algoritama definira se *klasa*[5] ulaznih objekata koji su dozvoljeni. Djelovanje algoritma je ispravno samo nad dozvoljenom klasom ulaznih objekata i rezultira klasom izlaznih objekata. Na primjer, algoritam iz primjera 1.3 može se primijeniti na klasu pozitivnih decimalnih brojeva, a rezultat također pripada toj klasi.

Algoritam je **uporabljiv** ako se za bilo koji član iz početne klase objekata može dobiti rezultat iz klase dozvoljenih završnih objekata u konačnom vremenu, tj. *uz konačni broj koraka u algoritamskom procesu*. Drugim riječima, *algoritam je neuporabljiv ako se njegovo izvođenje ne može završiti u konačnom vremenu ili ako se u nekom koraku pojavi prepreka koja onemogućava dovršenje koraka*.

Neki teorijski uporabljivi algoritmi su krajnje neprikladni za uporabu jer trajanje njihovog izvođenja može biti neprihvatljivo dugo. Takvi algoritmi su onda praktički neuporabljivi za neki podskup klase dozvoljenih objekata.

Isto tako, neki teorijski neuporabljivi algoritmi mogu biti **praktički uporabljivi** za određeni podskup klase početnih objekata, ako se u njih ugrade ograničenja koja zaustavljaju algoritamski proces nakon određenog broja koraka ili ga zaustavljaju nakon pojave prepreke u nekom od koraka.

Primjer 1.4. Algoritam dijeljenja decimalnih brojeva opće je poznat. Polazna klasa objekata je skup racionalnih brojeva s tim da nula ne može biti djelitelj. (Za vježbu opišite taj algoritam riječima!). Algoritam se zaustavlja kada se postigne ostatak jednak nuli. Tako se, primjerice, pri dijeljenju broja 4.401 s brojem 3 dobiva:

$$
\begin{array}{rl}
4.401 & : 3 = 1.467 \\
-3 & \\
\hline
14 & \\
-12 & \\
\hline
20 & \\
-18 & \\
\hline
21 & \\
-21 & \\
\hline
0 &
\end{array}
$$

[4] specijalizacija, od latinskog *species* — prilika, pojava, vrsta

[5] klasa, od latinskog *classis* — dio, odio, razred

Međutim, ako se, primjerice, broj 10 dijeli s brojem 3, tada se algoritamskim procesom dobiva:

$$
\begin{array}{rl}
1\,0 & : \;\; 3 = 3.333\ldots \\
-\,9 & \\
\hline
\quad 1\;0 & \\
\quad -\,9 & \\
\hline
\qquad 1\;0 & \\
\qquad -\,9 & \\
\hline
\qquad\quad 1\;0 &
\end{array}
$$

Proces se u ovom drugom slučaju ne zaustavlja nakon konačnog broja koraka i algoritam dijeljenja bi se mogao smatrati neuporabljivim. No, algoritam postaje uporabljiv ako se unaprijed ograniči broj znamenaka kvocijenta i proces zaustavi kada se te znamenke izračunaju. □

Primjer 1.5. Promotrimo primjer algoritma čiji algoritamski proces mora biti prekinut zbog prepreke u jednom od koraka. Početni objekt je prirodni broj. Algoritam je opisan sa sljedećih pet koraka:

1. Početni prirodni broj pomnožiti s brojem 2 i prijeći na drugi korak.
2. Dobivenom broju dodati broj 1 i prijeći na treći korak.
3. Naći ostatak dijeljenja te sume s brojem 3 i prijeći na 4. korak.
4. Podijeliti početni broj s dobivenim ostatkom iz koraka 3. Taj kvocijent je završni objekt. Prijeći na peti korak.
5. Zaustaviti algoritamski proces.

Ako je početna vrijednost jednaka 6, tada će se u pojedinim koracima dobivati sljedeći rezultati:

korak 1. $6 \cdot 2 = 12$;
korak 2. $12 + 1 = 13$;
korak 3. ostatak dijeljenja s 3 je 1;
korak 4. $6 : 1 = 6$, rezultat je 6;
korak 5. kraj.

Međutim, ako je početna vrijednost jednaka 7, tada se u pojedinim koracima dobiva:

korak 1. $7 \cdot 2 = 14$;
korak 2. $14 + 1 = 15$;
korak 3. ostatak dijeljenja s 3 je 0;
korak 4. $7 : 0$, dijeljenje s nulom nije dozvoljeno i algoritamski se proces ne može završiti.

Ovaj neuporabljivi algoritam može se pretvoriti u uporabljivi, ako se u trećem koraku ugradi ograničenje koje dozvoljava prijelaz u četvrti korak samo ako je ostatak dijeljenja različit od nule, a u protivnom javlja nastalu iznimku i prelazi na peti korak. Modificirani algoritam bi glasio:

1. Početni prirodni broj pomnožiti s brojem 2 i prijeći na drugi korak.
2. Dobivenom broju dodati broj 1 i prijeći na treći korak.
3. Naći ostatak dijeljenja te sume s brojem 3. Ako je ostatak jednak nuli, javiti nastalu iznimku i prijeći na peti korak, inače prijeći na četvrti korak.
4. Podijeliti početni broj s ostatkom. Taj kvocijent je završni objekt. Prijeći na peti korak.
5. Završiti algoritamski proces. □

* * *

Kao što je već rečeno, jedna od osnovnih pretpostavki za ispravno odvijanje algoritamskog procesa na temelju danog algoritma jest *postojanje izvoditelja koji razumije instrukcije od kojih je sastavljen algoritam*. U algoritmu iz primjera 1.3 pretpostavlja se da izvoditelj razumije što je znamenka jedinice, desetice, stotice, itd., da znade pravilno napisati jedan broj ispod drugoga, te da znade zbrojiti dvije ili više znamenaka.

Treba uočiti da obavljanje algoritma ne zahtijeva nikakvu domišljatost, štoviše, ono ne dozvoljava nikakvu inicijativu izvoditelja. *Izvršitelj djeluje sasvim mehanički*. To znači da izvoditelji ne moraju biti ljudi. Neke jednostavne algoritme mogu obavljati životinje ako ih se to dresurom nauči. Isto tako, može se sagraditi strojeve koji prepoznavaju instrukcije algoritma i obavljaju ih automatski. Ta činjenica je osnovica za izgradnju strojeva za izvođenje algoritama. Takvi strojevi su današnja računala. O ustroju i načinu rada računala govorit će se kasnije.

2. Jezik za zapisivanje algoritama — instrukcije za obavljanje operacija

Algoritmi i dozvoljeni objekti algoritama mogu se opisati prirodnim govornim jezikom, kao što je to učinjeno u dosadašnjim primjerima. Izvoditelj algoritma mora poznavati taj jezik. Na primjer, izvoditelj algoritma iz primjera 1.1, dakle — kuhar, mora poznavati riječi koje opisuju postupak pripremanja jela, a isto tako mora poznavati objekte — namirnice, kako bi mogao pripremati jelo. Treba uočiti da se u algoritmu pojavljuju samo *imena* namirnica, a ne same namirnice. Ako se recept prevede na neki strani jezik, ta imena bit će drugačija. Isto će tako i postupak biti opisan drugačijim riječima i rečenice će imati drugačiju konstrukciju.

Ako kuhar poznaje taj strani jezik, on će bez poteškoća provesti jednaki postupak, tj. algoritamski proces će biti jednak onom izvedenom iz prvobitnog opisa na hrvatskom jeziku.

Algoritmi se mogu opisati na bilo kojem govornom jeziku. Isto tako, moguće je sačiniti i **umjetne jezike** za zapisivanje algoritama. Oni moraju biti dovoljno dobro izgrađeni da bi se njima mogli opisati raznovrsni algoritmi. Algoritmi zapisani dobro oblikovanim umjetnim jezikom mogu biti mnogo sažetiji, obično su mnogo pregledniji i mogu se samo jednoznačno tumačiti. Algoritmi zapisani takvim jezikom nazivaju se i **programima**, a jezici **programskim jezicima**.

Ako se želi da algoritme automatski obavljaju računala, onda algoritme treba prevesti u program sastavljen od **instrukcija** koje služe za upravljanje strojem. Računala mogu obavljati samo vrlo jednostavne korake, pa se algoritmi moraju zapisati jezikom u kojem postoje samo vrlo jednostavne instrukcije. Takav jezik je tzv. **strojni jezik** računala.

Jezici za zapisivanje algoritama

Strojni jezici su teško razumljivi i vrlo neprikladni za čovjeka. Stoga su sačinjeni umjetni jezici za zapisivanje algoritama razumljiviji čovjeku. Ti jezici nazivaju se **višim programskim jezicima** za razliku od strojnog jezika računala kojeg bi se moglo nazvati **nižim** odnosno **osnovnim jezikom**. Viši programski jezici moraju biti tako oblikovani da se programi mogu prevesti u strojni jezik računala, kako bi se zatim mogli izvesti računalom.

Među više programske jezike ubrajaju se jezici s nazivima: FORTRAN[6], BASIC[7], ALGOL[8], Pascal[9], Ada[10], C[11]. U njima su definirane vrste objekata koji se mogu upotrebljavati i operacije nad tim objektima koje se mogu koristiti u zapisivanju algoritama. S obzirom da su vrste osnovnih objekata u takvim jezicima relativno jednostavne i da najčešće poprimaju oblik brojki ili slova, objekti se najčešće nazivaju **podacima**. Umjesto naziva klasa objekata upotrebljava se naziv **tip podataka**. Već smo ustanovili da se algoritam napisan programskim jezikom naziva programom i da se pojedini koraci zapisuju instrukcijama. Svaka je instrukcija sastavljena od simbola *operatora*[12] koji kazuje koju operaciju treba provesti s odabranim operandima, koje zovemo još i *argumentima*[13]. Nakon izvođenja instrukcije

[6] FORTRAN, skraćenica od engleskog *FORmula TRANslator* — prevodilac formula

[7] BASIC, skraćenica od *Beginner's All-purpose Symbolic Insruction Code* — opći simbolički instrukcijski kôd za početnike

[8] ALGOL, skraćenica od engleskog *ALGOrithmic Language* — algoritamski jezik

[9] Pascal je dobio ime po francuskom matematičaru i filozofu Blaisu Pascalu (1623–1662), koji se bavio i konstruiranjem mehaničkih naprava za računanje

[10] Ada je ime Ade Lovelace, kćerke pjesnika lorda Byrona, koja je engleskom matematičaru i inženjeru Charlesu Babbageu (1792–1871) pripremala programe za njegove mehaničke računske strojeve

[11] ime C dolazi od toga što su autori jezika nazivali varijante jezika u nastajanju redom slovima A, B, C i bili zadovoljni trećom varijantom

[12] operator, od latinskog *opera* — rad, posao

[13] od latinskog *argumentum* — čime se što razjašnjuje, u matematici: ulazna varijabla funkcije

dobiva se *rezultat*[14]. Rezultati prethodnih instrukcija mogu biti argumenti sljedećih i tada se oni nazivaju međurezultatima algoritma, odnosno programa. Početne vrijednosti objekata nazivaju se **ulaznim podacima**, a završne vrijednosti objekata **izlaznim podacima** programa.

Pisanje programa u nekom višem programskom jeziku zahtijeva detaljno poznavanje pravila zapisivanja instrukcija. Učenje tih pravila i njihovo strogo pridržavanje često zasjenjuje osnovne postavke zadatka koji rješavamo i postaje važnije od samih zamisli algoritama. Stoga se za jednostavan način zapisivanja algoritama koriste pojednostavljeni programski jezici. Takvi jezici nemaju stroga pravila kao viši programski jezici, ali su ipak dovoljno detaljno definirani tako da se sve potrebne aktivnosti mogu jasno i jednoznačno opisati. Nakon što se u takvom jeziku opiše algoritam, odnosno program, on se može prepisati u bilo koji od viših programskih jezika. Stvaranje programa započinje njegovim zasnivanjem u **jeziku zasnivanja programa** (engl. Program Design Language — PDL), a zatim se tako zapisan algoritam prepisuje u konačni oblik u nekom višem programskom jeziku.

Ovdje je odabran jezik zapisivanja algoritama, koji je vrlo blizak konstrukcijama programskog jezika Pascal, no zapisani algoritmi mogu se jednostavno pretvoriti u programe u bilo kojem višem programskom jeziku. Algoritamske konstrukcije su opisane riječima hrvatskog jezika i uvedena su pojednostavljenja kako bi se pri zapisivanju postigla preglednost i razumljivost. Jezik zasnivanja programa se, osim toga, može postupno dograđivati onako kako to određuje uvođenje novih pojmova.

Kao što je već rečeno, osnovni sastavni dio programa je instrukcija. Instrukcije se mogu grubo podijeliti na tri vrste:

- instrukcije za obavljanje operacija;
- instrukcije za određivanje toka programa;
- instrukcije za ulaz i izlaz podataka.

Razmotrimo najprije instrukcije za obavljanje operacija. Instrukcija prihvaća argumente i proizvodi rezultate. U *najjednostavnijem obliku* instrukcija se sastoji od jednog operatora, te prihvaća jedan ili dva operanda i proizvodi jedan rezultat. Instrukcija završava znakom ";", kojim je odvojena od sljedeće. Zbog preglednosti, najprikladnije je u jednom retku pisati samo jednu instrukciju. *Operator* unutar instrukcije određuje operaciju koja se provodi u jednom koraku programa.

Argumente zovemo *ulaznim varijablama*, a rezultate *izlaznim varijablama* i označavamo svaku od njih svojim *simbolom*[15] — imenom. Operacije se također označavaju simbolima operatora i to što sličnijim onima uobičajenim u matematici. Tako se za aritmetičke operacije upotrebljavaju sljedeći operatori:

[14] rezultat, od latinskog *resultare* — odjeknuti

[15] simbol, od grčkog *simbolon* — znak raspoznavanja

$+$ za zbrajanje;
$-$ za oduzimanje;
$*$ za množenje;
$/$ za dijeljenje.

Zagrade, kojima se može promijeniti uobičajeni redoslijed obavljanja operacija, samo su okrugle. Zato pri zapisivanju i čitanju složenijih izraza treba paziti na pravilno sparivanje otvorenih i zatvorenih zagrada.

Rezultat obavljene operacije neke instrukcije zapisuje se u varijablu rezultata. Simbol *operatora pridruživanja* rezultata varijabli sastoji se od dvotočke koju slijedi znak jednakosti, tj. ":=". Tako se, primjerice, instrukcija koja zbroj dvaju brojeva a i b zapisuje kao rezultat c, piše ovako:

$$\texttt{C} := \texttt{A} + \texttt{B};$$

i čita: "varijabli `C` pridružiti vrijednost zbroja varijabli `A` i `B`".

Za označavanje simbola varijabli koristit će se velika slova ili više uzastopno napisanih velikih slova i znamenki, s tim da prvi znak uvijek bude slovo. Zdesne strane operatora pridruživanja mogu se nalaziti i složeniji izrazi s više varijabli, te se, primjerice, može pisati

$$\texttt{Y} := \texttt{A} * \texttt{B} + \texttt{C} * \texttt{D};$$

Znak pridruživanja ne smije se zamijeniti sa znakom jednakosti u matematici. Treba uočiti da, primjerice, instrukcija

$$\texttt{X} := \texttt{X} + 1;$$

izaziva sljedeću operaciju:

- uzima se kao argument vrijednost zapisana u `X`;
- ta se vrijednost uvećava za jedan;
- rezultat uvećavanja se zapisuje pod imenom `X`.

Tako je `X` nakon izvođenja instrukcije uvećan za jedan. Ako bi se znak pridruživanja poistovjetio s matematičkim simbolom jednakosti, napisani izraz ne bi imao nikakvog smisla (protumačite zašto!).

Argumente nazivamo *ulaznim varijablama*[16] stoga što mogu poprimiti različite vrijednosti. Međutim, neki argumenti mogu biti stalne, nepromjenjive veličine i oni se stoga zovu *konstantama*[17].

Postoji stanovita sličnost između zapisivanja i djelovanja operatora pridruživanja i načina zapisivanja funkcija u matematici. Ako je x element skupa A, a y element skupa B, pravilo koje za svaki element x određuje pripadnu vrijednost y u matematici se naziva funkcijom f i zapisuje se ovako:

$$f : A \to B,$$

[16] varijabla, od latinskog *variabilis* — promjenljiv

[17] konstanta, od latinskog constans — čvrst, postojan.

ili ovako:

$$y = f(x).$$

Skup A je domena ili područje definicije funkcije, a x se naziva nezavisnom varijablom. Skup B je kodomena ili područje vrijednosti funkcije, a y se naziva zavisnom varijablom.

Primjer 1.6. U prvom su stupcu zapisane neke jednostavne matematičke funkcije, a u drugom stupcu pripadne instrukcije u jeziku zasnivanja programa, koje za zadane vrijednosti ulaznih varijabli izračunavaju vrijednost funkcije i pridružuju tu vrijednost izlaznoj varijabli `Y`.

funkcija	instrukcija
$y = x$	`Y := X;`
$y = a \cdot x + b$	`Y := A * X + B;`
$y = x \cdot z$	`Y := X * Z;`
$y = 1 : x$	`Y := 1 / X;`
$y = x \cdot x + 1$	`Y := X * X + 1;`
$y = x \cdot (x - 1)$	`Y := X * (X - 1);`
$y = a \cdot (x \cdot (x - 1) + b)$	`Y := A * (X * (X - 1) + B);`

□

Primjer 1.7. Sljedeći slijed instrukcija određuje redom prvih pet *potencija*[18] ulazne varijable X:

```
Y := 1;
Y := Y * X;
Y := Y * X;
Y := Y * X;
Y := Y * X;
Y := Y * X;
```

Opišimo riječima što se događa pri izvođenju danog slijeda instrukcija. Nakon prve instrukcije varijabla `Y` poprima vrijednost 1. Nakon obavljanja druge instrukcije `Y` poprima vrijednost `X`, a nakon treće vrijednost $\mathtt{X} \cdot \mathtt{X} = \mathtt{X}^2$. Sljedeće instrukcije redom izračunavaju sljedeće potencije varijable `X`. Nakon zadnje instrukcije varijabla `Y` poprima vrijednost $\mathtt{X}^5$. Treba se prisjetiti da svaki broj različit od nule na nultu potenciju ima vrijednost 1, tako da se prvo pridruživanje $\mathtt{Y} := 1$ može smatrati nultom potencijom varijable `X`.

Uvjerite se da se uz $\mathtt{X} = 2$ za varijablu `Y` dobiva sljedeći niz vrijednosti:

2^0	2^1	2^2	2^3	2^4	2^5
1	2	4	8	16	32

[18] potencija, od latinskog *potentia* — moć, sila, snaga

To je prvih šest potencija broja dva. Uočite da je $2^0 = 1$. Ove potencije vrijedi zapamtiti, jer će se potencije broja dva često upotrebljavati u daljnjem tekstu. □

Grupe instrukcija i programi

Često se javlja potreba da se više instrukcija objedini u **grupu instrukcija**. Grupiranje instrukcija označavat će se vertikalnom crtom ispred cijele grupe. U nekim višim programskim jezicima se početak i završetak grupe označavaju posebnim znacima ili posebnim riječima. Tako se u programskom jeziku Pascal početak i završetak grupe označavaju riječima **begin** i **end**. U programskom jeziku C početak grupe označava se vitičastom zagradom otvorenja $\{$, a završetak grupe vitičastom zagradom zatvorenja $\}$. Mi ćemo koristiti *vertikalnu crtu* ispred instrukcija koje čine grupu. Također ćemo radi preglednosti zapisa algoritma paziti da instrukcije iste grupe pišemo točno jednu ispod druge.

Primjer 1.8. Složene matematičke funkcije mogu se izraziti s pomoću operacija s dva operanda. Tako se, primjerice, funkcija

$$y = 4x^2 + 3$$

može napisati na sljedeći način:

$$y = 4(x \cdot x) + 3,$$

gdje je zagradama određen redoslijed obavljanja pojedinih opearcija. To izračunavanje može se prikazati grupom instrukcija:

```
Y := X * X ;
Y := 4 * Y ;
Y := Y + 3 ;
```

Isto tako, možemo se uvjeriti da grupa instrukcija

```
Y := 2 * X ;
Y := Y + 3 ;
Y := Y * Y ;
Y := Y - 1 ;
Y := Y * Y ;
Y := 2 * Y ;
```

izračunava postupno funkciju:

$$y = 2((2x + 3)^2 - 1)^2.$$

Vertikalna crta ispred instrukcija označava da se radi o jednoj cjelini. Ovaj će se način označavanja primjenjivati u daljnjem tekstu za grupiranje instrukcija. □

Osim simbola, u programskim se jezicima, pa i u jeziku zasnivanja programa, upotrebljavaju odabrane riječi ili kratice za označavanje nekog djelovanja. To su **ključne** ili **rezervirane riječi**. U knjigama i ostalim publikacijama, one se obično tiskaju masnim slovima. Međutim, kada se programi pišu rukom, to je teško činiti (nespretno je pisati podebljana slova!), pa time nastaje razlika između tiskanih i rukom pisanih programa. Zbog toga se iz didaktičkih[19] razloga u ovoj knjizi za označavanje ključnih riječi upotrebljava podcrtavanje. Tako će algoritmi i programi biti na jednak način zapisani u knjizi, na školskoj ploči i u bilježnici. Ključne riječi uvodit će se i objašnjavati postupno.

Početak programa označava se riječju početak, a njegov završetak riječju kraj i točkom iza te riječi. Instrukcije programa treba zapisati jednu ispod druge. Pretpostavlja se da će one biti obavljane onim redom kojim su zapisane. Prema tome, program pisan jezikom zasnivanja ima sljedeći izgled:

```
početak
    instrukcija 1;
    instrukcija 2;
    instrukcija 3;
    ⋮
    instrukcija N;
kraj.
```

Pri zapisivaju pojedinih algoritama i dijelova algoritama u ovoj knjizi nećemo uvijek pisati početak i kraj, jer se i neće uvijek raditi o cjelovitim programima.

Instrukcije mogu biti jednostavne, kao što su one iz primjera 1.6, ali mogu biti i složene. Jednostavna instrukcija može se napisati u jednom retku i sastoji se od imena izlazne varijable, operatora pridruživanja i izraza koji se sastoji od imena ulaznih varijabli povezanih operatorima operacija. Grupu jednostavnih operacija možemo smatrati složenom instrukcijom. Vidjet ćemo da takvu grupu možemo kraće zapisati tako da taj zapis sliči jednoj instrukciji. I cijeli program može se u nekom smislu smatrati samo jednom instrukcijom, što simbolizira vertikalna crta ispred niza instrukcija.

[19] didaktika, od grčkog *didaktikos* — sposobnost učenja