

Vladas TUMASONIS

INFORMATIKA

Vilnius, 2004

Turinys

1	Algoritmo sąvoka	3
2	Formalus algoritmo atlikimas. Kompiuteris – algoritmo vykdytojas	5
3	Informacijos vaizdavimas kompiuteryje	7
4	Formalus programavimo kalbų apibrėžimas	10
5	Paskalio programavimo kalba	15
	5.1 Duomenų tipai	16
	5.2 Valdymo struktūros	29
	5.3 Procedūros ir funkcijos	32
6	Įvadas į algoritmų analizę	39
7	Masyvo rūšiavimas	40
	7.1 Mažiausio elemento metodas	40
	7.2 Burbulo metodas	40
	7.3 Greitojo rūšiavimo metodas	41
	7.4 Piramidės metodas	45
8	Paieškos algoritmai	48
	8.1 Dvejetainės paieškos algoritmas	48
	8.2 Maišos (<i>hash</i>) metodas	50
9	Dinaminės duomenų struktūros	52
	9.1 Statiniai ir dinaminiai kintamieji	52
	9.2 Sąrašai	54
	9.3 Stekas	59
	9.4 Eilė	60
	9.5 Dvejetainiai medžiai	62
	9.6 AVL medžiai	67
10	Rekursijos realizavimas kompiuteryje	81
11	Rekursinių funkcijų pavyzdžiai	83
12	Moduliai	88
13	Dinaminių masyvų modeliavimas	90
	Literatūra	91

INFORMATIKA

Informatika – tai mokslo ir technikos šaka, nagrinėjanti informacijos apdorojimą kompiuteriais.

1. ALGORITMO SĄVOKA

Algoritmas – nurodymų seka, kurią atlikus su kažkokiais objektais, gaunamas uždavinio sprendinys.

Algoritmo savybės:

1. Masiškumas. Vienu algoritmu išsprendžiama daug uždavinių.
2. Rezultatyvumas. Algoritmas turi duot kokį nors rezultatą – tai gali būti ne tik lygties sprendiniai, bet ir, pvz, atsakymas, kad sprendinių nėra arba pradiniai duomenys klaidingi.
3. Efektyvumas.
4. Aiškumas. Algoritmas pateikiamas vykdytojiui suprantama ir aiškia kalba.
5. Diskretumas. Algoritmą sudaro diskreti veiksmų seka.
6. Baigtinumas.

Euklido algoritmas (autentiškas) didžiausiam bendrajam dviejų skaičių dalikliui rasti:

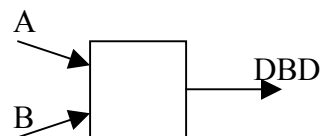
1. Laikyti A pirmuoju skaičiumi, o B – antruoju.
2. Palyginti pirmąjį ir antrąjį skaičius. Jei jie lygūs, pereiti į p. 5.
3. Jei pirmas skaičius mažesnis už antrąjį, tai juos sukeisti vietomis.
4. Iš pirmojo skaičiaus atimti antrąjį skaičių ir gautą skaičių laikyti pirmuoju skaičiumi. Pereiti į p.2.
5. Pirmąjį skaičių laikyti rezultatu. STOP.

Algoritmų užrašymo būdai

1. Natūralia kalba (pvz., lietuvių).
2. Matematinė simbolika + kalba.

Euklido algoritmas matematinė simbolika + kalba:

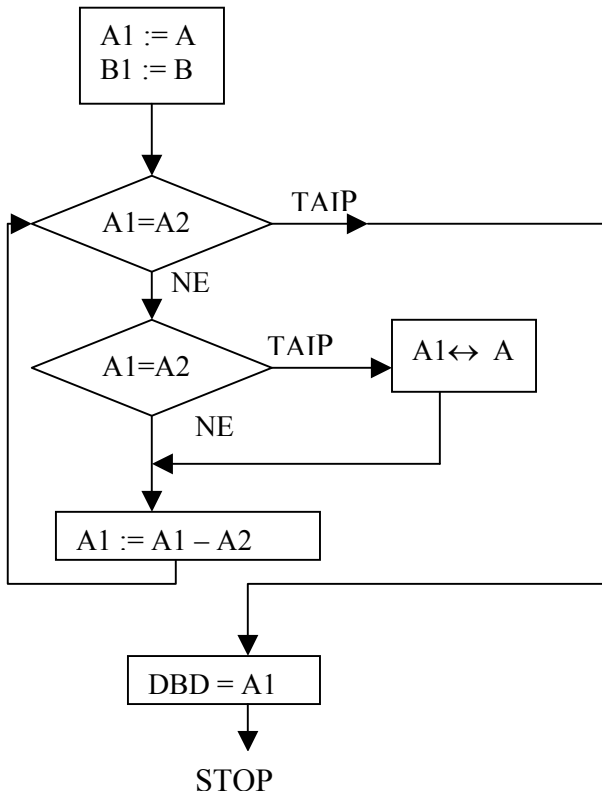
1. $A_1 = A; A_2 = B.$
2. Jei $A_1 = A_2$, tai į p. 5.
3. Jei $A_1 < A_2$, tai $A_1 \leftrightarrow A_2$.
4. $A_1 = A_1 - A_2$, į p.2.
5. $DBD = A_1$. STOP.



Atliksime šį algoritmą su skaičiais 153 ir 646.

A1	153	646	493	340	187	34	153	119	85	51	17	34	17
A2	646	153	34	17									
DBD	17												

3. Blokinė schema:



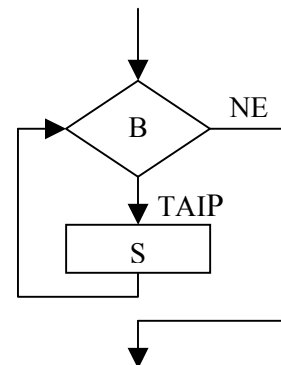
Programavimo kalba:

```

...
A1 := A; A2 := B;
While A1 <> A2 do
  Begin
    If A1 < A2 then begin
      r := A1;
      A1 := A2;
      A2 := r;
    end;
    A1 := A1 - A2;
  end;
DBD := A1;
...
  
```

Su algoritmo sąvoka betarpiškai yra susijusi algoritmo vykdytojo sąvoka: algoritmas kuriamas tam, kad kažkas atliktų algoritmo nurodomus veiksmus. Tas kažkas ir yra algoritmo vykdytojas. Sudarius ir užrašius algoritmą, jis gali būti atliekamas formaliai. Taigi, jo atlikimą galima pavesti automatui-kompiuteriui. Šitaip kompiuteris tampa algoritmo vykdytoju.

Ciklo while blokinė schema:



2. FORMALUS ALGORITMO ATLIKIMAS . KOMPIUTERIS – ALGORITMŲ VYKDYTOJAS

Kompiuterių istorija

1-oji karta 1945m. ENIAC – pirmasis kompiuteris (išrastas JAV)
 1951m. MESM – pirmasis kompiuteris buvusioje Sovietų Sąjungoje
 BESM
 STRELA

Fizikinis pagrindas šiuose kompiuteriuose – elektroninės lempos.

2-oji karta ~ 1960m. M-20
 BESM-2
 MINSK-2, 22, 22M

Fizikinis pagrindas šiuose kompiuteriuose – puslaidininkiai. Jų greitis neviršija 100 op./s (Hz)

3-oji karta ~ 1970m. BESM-6
 MINKS-32
 IBM 360
 ES 1020,1030,1040,1050, 1035, 1045, 1055

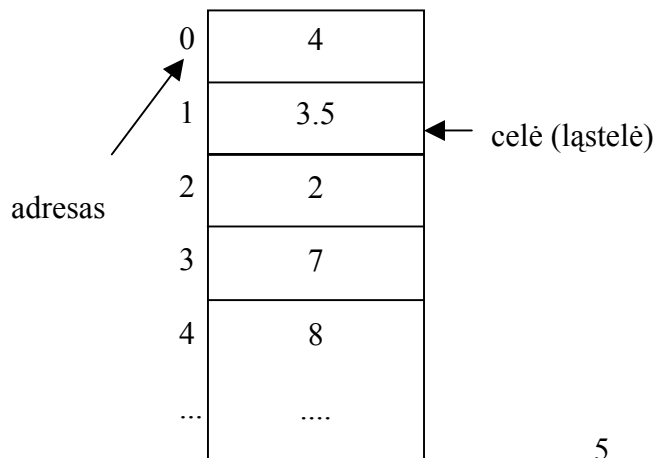
Fizikinis pagrindas šiuose kompiuteriuose – integrinės schemos. Greitis nuo 200 000 iki 2 000 000 op./s

4-oji karta ~ 1980m. SM-3, Robotron,
 Macintosh, PDP-8,
 CRAY, IBM PC

Fizikinis pagrindas šiuose kompiuteriuose – mikroprocesoriai. Greitis neviršija 500 000 000 op./s . Tuo metu jau ėmė vystytis kompiuteriniai tinklai.

Kompiuterio architektūra

Pagal Fon Neimano principą kompiuterio operatyvioji (greitoji) atmintis schematiškai sudaro struktūrą:



struktūra – tam tikra dalių priklausomybė

ląstelėje yra skaičius – operandas. Jis dalyvauja operacijose.

Įvairios (matematinės) operacijos kompiuterio atmintyje yra koduojamos tam tikrais kodais, pvz.

operacija	Operacijos kodas
STOP	00
+	01
-	02
/	03
*	04

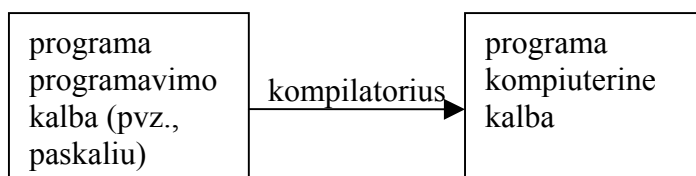
Pavyzdžiui, turime komandą:

01 002 005 004 0

pirmi du skaičiai reiškia operaciją (sudėtį), antra ir trečia skaičių grupės – operandų adresus (2 ir 5 ląstelės), ketvirta skaičių grupė – adresą ląstelės, į kurią turės būti patalpintas rezultatas.

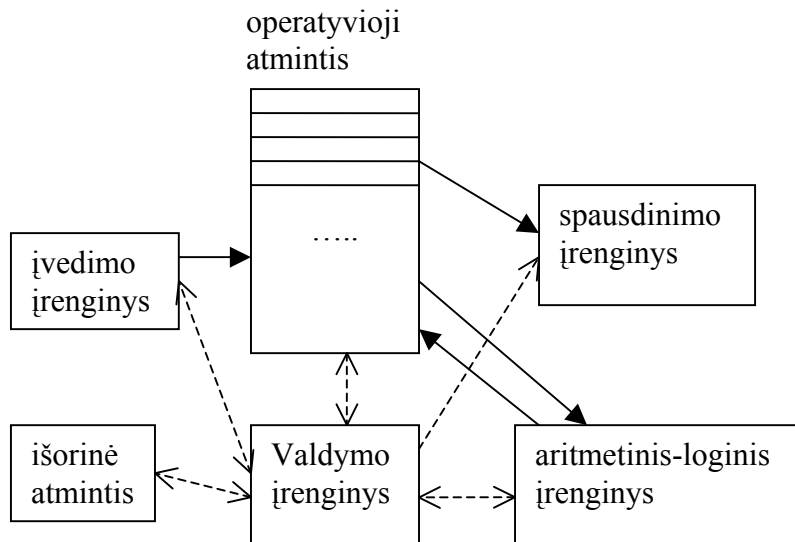
Komandų visuma sudaro **programą**, o komandų sistema sudaro **mašininę kalbą**.

Tam, kad programa, užrašyta kokia nors **programavimo kalba**, būtų perrašoma (sukompiliuojama) į kompiuterio kalbą, reikalingas kompiliatorius:

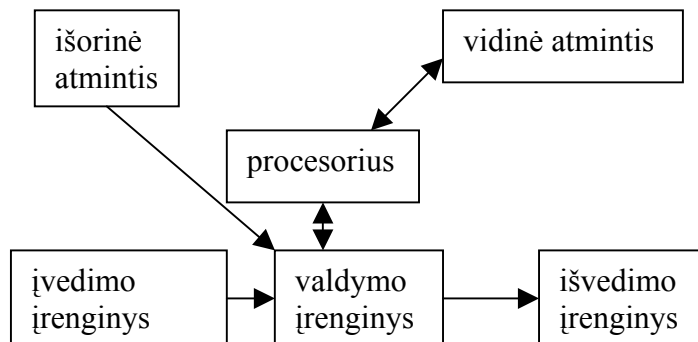


Pagrindinės kompiuterio dalys:

1. Atmintis
2. Įvedimo įrenginys
3. Išvedimo įrenginys
4. Aritmetinis-loginis įrenginys
5. Valdymo įrenginys



Pagal dabartinę terminiją:



Programinio valdymo principas.

Šis principas teigia, kad kompiuterio darbas yra visiškai apibrėžtas programos, kuri buvo įvesta į kompiuterį.

3. INFORMACIJOS VAIZDAVIMAS KOMPIUTERYJE

Sveikieji skaičiai (fiksoto kabelio formatas)

Sveikieji skaičiai kompiuteryje vaizduojami dvejetainiu kodu, skaičiui skiriant kažkokį kiekį baitų, priklausomai nuo to, kokiam tipui priskiriamas skaičius, pvz.:

Skaičius 25 atrods taip :

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Pirmas bitas skiriamas ženklui: „0“ ~ +

„1“ ~ -

$$\text{nes } 25_{10} = 11001_2$$

Sveikųjų skaičių intervalas :

nuo $0\ 000\ 0000\ 0000\ 0000_2$ iki $0\ 111\ 1111\ 1111\ 1111_2$

nepamirštant, kad pirmas bitas skiriamas ženklui. Dešimtainiais skaičiais tai atitiktų nuo 0 iki $2^{15} - 1$, tai yra nuo 0 iki 32767.

Neigiami skaičiai vaizduojami papildomu kodu. Tai leidžia atimtį versti sudėtimi su papildomu kodu. Skirtumą vaizduojame suma, kad nereiktų papildomos operacijos.

$$a - b = a + (-b)$$

Apibrėžimas. Skaičiaus papildomas kodas $N_{\text{papild.}} = p^n - N$, čia p – sistemos pagrindas, n – skilčių skaičius, skiriamas skaičiams vaizduoti.

Iliustruosime tai dešimtaine sistema dviženkliais skaičiams ($p=10, n=2$).

Pvz., turim skaičių 24. Neigiamas skaičius, -24, būtų atvaizduojamas tokiu atvirkštiniu kodu:

$$10^2 - 24 = *76, \text{ čia } * \text{ sąlyginai rodo skaičiaus minuso ženklą.}$$

Dešimtainiams skaičiams atimties vertimas sudėtimi atrodytų :

$$\begin{array}{r} + 37 \\ -24 \\ \hline + 37 \\ *76 \\ \hline 13 \end{array}$$

$$\begin{array}{r} + 24 \\ -37 \\ \hline + 24 \\ *63 \\ \hline -13 \end{array}$$

Atsakymas pagal daugiaženklų skaičių sudėties taisyklę:

$$\begin{array}{l} 37 + *76 = 113, \quad 113 - 10^2 = 13 \\ 24 + *63 = 87, \quad 87 - 10^2 = -13 \end{array}$$

Matome, kad papildomas kodas dar gali būti gaunamas keičiant kiekvieną skaitmenį skaitmeniu papildančiu iki didžiausio sistemos skaitmens (t.y. 9) ir pridant vienetą. Pvz., 2 keičiamas į 7, nes $9-2=7$.

Mūsų dvejetainiame vaizdavime, užimančiame 16 bitų, papildomas kodas $2^{16}-N$. Jis gaunamas 0 keičiant į 1, o 1 – į 0 (atliekama inversija ir gaunamas vadinamasis atvirkštinis kodas) ir pridant 1.

Pvz., norime gauti -15, kai žinome, kad $15_{10} = 0 \ 000 \ 0000 \ 0001 \ 1001_2$

$$\begin{array}{r} 0 \ 000 \ 0000 \ 0001 \ 1001 \\ 111 \ 1111 \ 1110 \ 0110 \quad - \text{atvirkštinis kodas} \\ + \quad \quad \quad 1 \\ \hline 1 \ 111 \ 1111 \ 1110 \ 0111 \quad - \text{papildomas kodas} \end{array}$$

$$\text{taigi, } -15_{10} = 1 \ 111 \ 1111 \ 1110 \ 0111$$

Norint iš neigiamo dvejetainio skaičiaus gauti jam priešingą (iš -15 gauti 15) reikia paimti papildomo kodo papildomą kodą, nes $(2^n - (2^n - N)) = N$:

$$\begin{array}{r} 1 \ 111 \ 1111 \ 1110 \ 0111 \\ 000 \ 0000 \ 0001 \ 1000 \\ + \quad \quad \quad 1 \\ \hline 0 \ 000 \ 0000 \ 0001 \ 1001 \end{array}$$

Galima pastebėti, kad mažiausių dviženklų dešimtainių skaičių papildomi kodai yra tokie:

* 0 2 ~ -98

* 0 1 ~ -99

*00 ~ -100

vadinasi, taip gaunami kodai atitiks skaičius iš intervalo [-100, 99]

Analogiškai mažiausio neigiamo skaičiaus kodas yra 1 000 0000 0000 0000. Jo reikšmę surandame paėmę dar kartą papildomą kodą:

$$\begin{array}{r|rrrr} 1 & 000 & 0000 & 0000 & 0000 \\ + & 111 & 1111 & 1111 & 1111 \\ \hline 1 & 000 & 0000 & 0000 & 0000 \end{array}$$

Taigi minimalus neigiamas skaičius yra $1\ 000\ 0000\ 0000\ 0000_2 = -2^{15}$

Gavome, kad po tiek baitų turintys skaičiai yra iš intervalo

$$[-2^{15}, 2^{15} - 1] = [-32768, 32767]$$

Skaičiai, priklausantys šiam intervalui, Paskalyje ir Turbo paskalyje vadinami integer tipo skaičiais.

Kiti sveikųjų skaičių tipai:

Tipai, užimantys po 2 baitus:

Integer tipas $[-2^{15}, 2^{15} - 1]$ $[-32768, 32767]$

Word tipas $[0, 2^{16} - 1]$ $[0, 65535]$

Tipai, užimantys po 1 baitą:

Shortint tipas $[-2^7, 2^7 - 1]$ $[-128, 127]$

Byte tipas $[0, 2^8 - 1]$ $[0, 255]$

Longint tipas $[-2^{31}, 2^{31} - 1]$ $[-2147483648, 2147483647]$

Užima 4 baitus.

Realieji skaičiai (slankaus kablelio formatas)

1) vaizdavimas su paklaida

Matematikoje trupmeniniai skaičiai gali būti išreikšti dešimtainėmis trupmenomis.

Trupmeninis skaičius gali būti užrašomas, pvz.:

$$14,15 = 0,1415 * 10^2$$

$$2/3 = 0,66666... \text{ (vaizdavimas su paklaida)}$$

skaičiaus dalį "0,1415" vadinsime **mantise**, o dalį "10²" vadinsime **eile**. Skaičiai su slankiu kableliu vaizduojami taip:

02	1	4	1	5
----	---	---	---	---

00	0	6	6	7
----	---	---	---	---

(priklausomai nuo to, kiek skaičiui skiriama baitų, dalys, kurios netelpa į mantisę, yra apvalinamos)

2) operacijos su paklaida

Matematikoje skaičiai sudedami stulpeliu:

$$\begin{array}{r} + 14,5 \\ 0,6667 \\ \hline 14,8167 \end{array}$$

Norint sudėti slankaus kablelio skaičius kompiuteryje, pirmiausia būtina suvienodinti jų eiles (paprastai žemesnė eilė pakeliama iki aukštesnės):

02	1	4	1	5
----	---	---	---	---

02	0	0	6	7
----	---	---	---	---

Tada sudedamos skaičių mantisės ir gaunama rezultato mantisė, o eilė lieka tokia pat, kaip ir abiejų sudedamųjų skaičių :

$$\begin{array}{r} + 02 \quad 1415 \\ 02 \quad 0067 \\ \hline 02 \quad 1482 \end{array}$$

Turbo Paskalyje slankaus kablelio skaičiai priskiriami tokiems tipams:

Real tipas	6 baitai	$[2,9 * 10^{-39} \dots 1,7 * 10^{35}]$
Double tipas	8 baitai	$[5,0 * 10^{-224} \dots 1,7 * 10^{206}]$
Simple tipas	4 baitai	$[1,5 * 10^{-46} \dots 1,4 * 10^{38}]$
Extended tipas	10 baitų	$[2,4 * 10^{-4392} \dots 6,1 * 10^{4832}]$

Loginės reikšmės

Loginės reikšmės gali įgyti 'klaidinga' arba 'teisinga' prasmę. Tai gali būti vaizduojama įvairiai:

Lietuvių kalboje:	matematinėje logikoje:	Turbo paskalyje:
"teisinga"	1	TRUE
"klaidinga"	0	FALSE

Turbo Paskalyje šias reikšmes atitinkantis tipas vadinamas **Boolean** tipu. Loginio (boolean) tipo reikšmė užima 1 baitą:

<div><div></div><div></div><div></div></div>	0000 0001 ~ TRUE
<div><div></div><div></div><div></div></div>	0000 0000 ~ FALSE
0	7

Jokie aritmetiniai veiksmai su loginėmis operacijomis neatliekami, išskyrus lyginimo operaciją (kaip galima matyti iš atitinkamų reikšmių, FALSE < TRUE)

Kiti loginiai tipai Turbo paskalyje: **byteboolean** (1 baitas), **wordbool** (2 baitai), **longbool** (4 baitai).

Simbolių vaizdavimas

Simboliai kompiuterio atmintyje vaizduojami juos atitinkančiais kodais. Kiekviena raidė (klaviatūros simbolis) turi savo kodą.

Vienam simboliui skiriamas vienas baitas (8 bitai). Sudaromos kodų lentelės, kuriomis vadovaudamasis, kompiuteris iškoduoja tekstą.

Lotyniškoms raidėms (simboliams) koduoti naudojama 437 lentelė (DOS). Lietuviškoms raidėms (besiskiriančių nuo lotyniškų jų yra 12) naudojama 775 lentelė.

4. FORMALUS PROGRAMAVIMO KALBŲ APIBRĖŽIMAS

Kalbą apibrėšime naudodamiesi gramatinėmis taisyklėmis.
Tarkime, kad

$$\Sigma = \{a; b; c\}$$

abėcėlė, ir ją sudaro simboliai (ženklai, rašmenys). Iš abėcėlės ženklų (raidžių) galima konstruoti žodžius, pvz. aab, babbbc.

Kalba iš abėcėlės Σ vadinsime aibę visų galimų žodžių, kuriuos sudaro simboliai, priklausantys abėcėlei Σ . Jei kalba baigtinė, tai ji turi apibrėžtą skaičių žodžių, pvz.:

$$L = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$$

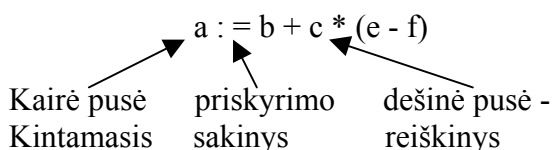
tada, pavyzdžiui, žodis “aa” priklauso kalbai L , o žodis “abc” nepriklauso kalbai L .

Visų galimų kalbų aibei iš abėcėlės Σ apibrėžti naudosime simbolį Σ^* . Tada, pavyzdžiui, tiek žodis “aa”, tiek žodis “abc” priklauso Σ^* , o žodis “abcd” nepriklauso Σ^* .

Du programavimo kalbos apibrėžimo aspektai:

- 1) Kalbos **sintaksė** ((gramatika) nusako, kaip sudaryta kalbos konstrukcija)
- 2) Kalbos semantika (nusako konstrukcijos prasmę (kokie veiksmai atliekami))

Pvz.:



$a [:= b * / + 5 (($ - pagal sintaksę klaidingas klaidingas sakiny

Simboliai, iš kurių konstruojamos kalbos (nebaigtinės) vadinami **terminaliniais simboliais**.

1. V_T – terminalinių (pagrindinių) simbolių aibė (terminalinė abėcėlė)
2. V_N – neterminalinių (pagalbinių) simbolių alfabetas (aibė)
 $V_T \cap V_N = \emptyset$
 $V = V_T \cup V_N$
3. S - pradinis simbolis.
 $S \in V_N$

4. P – išvedimo (gramatinių, sintaksinių) taisyklių aibė.

$P = \{x \rightarrow \beta\}$ - iš x išvedama β

$x \in V_N, \beta \in V^*$.

1 Apibrėžimas. Iš žodžio ψ betarpiškai išvedamas žodis φ ($\psi \rightarrow \varphi$), jeigu egzistuoja tokie žodžiai φ_1 ir φ_2 ($\varphi_1, \varphi_2 \in V^*$), jog:

$\varphi \div \varphi_1 x \varphi_2$

$\psi \div \varphi_1 \beta \varphi_2$

$X \rightarrow \beta \in P \quad x \in V_N, \beta \in V^*$

pvz1.:

$\varphi = abc, \psi = ef, \text{ jei } \xi = abcef, \text{ tai } \xi = \varphi\psi$

pvz2.:

$\varphi_1 = aaa$

$\varphi_2 = bbb$

$\varphi \div \varphi_1 x \varphi_2$

$\varphi = aaaXbbb$

$X \rightarrow cc \in P$

$\varphi = aaaccbbb$

$X \rightarrow ccY$

$Y \rightarrow dd \quad aaaXbbb \rightarrow aaaccYbb \rightarrow aaacddbb$

2 Apibrėžimas. Iš žodžio φ išvedamas žodis η , jeigu egzistuoja seka $\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_n : \varphi_0 = \varphi; \varphi_n = \eta$ ir $\varphi_i \rightarrow \varphi_{i+1}, i = 0, n = 1$.
 $\varphi \Rightarrow \eta$ (nebetarpiškai!)

3 apibrėžimas. Gramatika yra ketvertas $G = \langle V_T, V_N, P, S \rangle$

4 apibrėžimas (Pagrindinis apibrėžimas) Programavimo kalba L vadinama žodžių, sudarytų iš terminalinių simbolių, aibė, tokių, kad iš pradinio simbolio išvedamas S .

$L = \{s \in V_T^* : S \Rightarrow s\}$

žodis $\varphi \in \mathfrak{S}$, jeigu $S \Rightarrow \varphi$

$S \rightarrow \varphi_0 \rightarrow \varphi_1 \rightarrow \varphi_2 \rightarrow \dots \rightarrow \varphi_n, \varphi_n = \varphi$

Seka $\varphi_0, \varphi_1, \varphi_2, \dots, \varphi_n$ vadinama žodžio φ **išvedimu**. (norint įrodyti, kad žodis priklauso kalbai, reikia parodyti jo išvedimą)

Pvz:

$S \rightarrow abXccc$

$X \rightarrow aY$

$Y \rightarrow bbb$

$s \Rightarrow ababbbccc$

Kalbų pavyzdžiai

1) L_0 -universalinė kalba, kuriai priklauso bet kuri a ir b seka

a) $V_T = \{a, b\};$

$V_N = \{S\}$

$$P=\{S \rightarrow a; S \rightarrow b; S \rightarrow aS; S \rightarrow bS\}$$

Įrodysime, kad $aaab \in L_0$

$$S \rightarrow aS \rightarrow aaS \rightarrow aaaS \rightarrow aaab;$$

$$S \rightarrow aaab \Rightarrow aaab \in L_0.$$

$$b) \quad V_T = \{a_1, \dots, a_n\};$$

$$V_N = \{S\};$$

$$\left. \begin{array}{l} S \rightarrow a_1; \\ \dots \\ S \rightarrow a_n \end{array} \right\} n$$

$$\left. \begin{array}{l} \dots \\ S \rightarrow a_1; \\ \dots \\ S \rightarrow a_n \end{array} \right\} 2n \text{ taisyklių}$$

Neterminalinis simbolis **A** pažymėsime sąvoką aibę.

$$V_N = \{S, A\};$$

$$\left. \begin{array}{l} A \rightarrow a_1; \\ \dots \\ A \rightarrow a_n \end{array} \right\} n$$

$$\left. \begin{array}{l} S \rightarrow A; \\ S \rightarrow AS. \end{array} \right\} 2$$

$$(n+2) \text{ taisyklių}$$

Įrodysime, kad $a_1 a_2 a_4 \in L_0$

$$S \rightarrow AS \rightarrow AAS \rightarrow AAA \rightarrow a_1 AA \rightarrow a_1 a_2 A \rightarrow a_1 a_2 a_4;$$

$$S \rightarrow a_1 a_2 a_4; \Rightarrow a_1 a_2 a_4 \in L_0.$$

2) L_1 ;

Kalbai priklauso $\alpha \bar{\alpha}$, pvz., $aabbba$, kur $\alpha = aab$; $\bar{\alpha} = baa$.

$$V_T = \{a, b\};$$

$$V_N = \{S\};$$

$$P = \{S \rightarrow aa, S \rightarrow bb, S \rightarrow aSa, S \rightarrow bSb\}.$$

Įrodysime, kad $aabbba \in L_1$;

$$S \rightarrow aSa \rightarrow aaSaa \rightarrow aabbba;$$

$$S \rightarrow aabbba, aabbba \in L_1.$$

3) L_2 ;

Kalbai priklauso $\{a^m; b^n\}$.

$$a) \quad V_T = \{a, b\};$$

$$V_N = \{S, A, B\}, \text{ A-bet kokia } a \text{ raidžių seka, B- bet kokia } b \text{ raidžių seka.}$$

$$P = \{A \rightarrow a, A \rightarrow aA, B \rightarrow b, B \rightarrow bB, S \rightarrow A, S \rightarrow B, S \rightarrow AB\}$$

Irodysime, kad $aaabb \in L_2$.

$S \rightarrow AB \rightarrow aAB \rightarrow aaAB \rightarrow aaaB \rightarrow aaabB \rightarrow aaabb$, vadinasi, $aaabb \in L_2$.

b) $V_T = \{a, b\}$;

$V_N = \{S, B\}$;

$P = \{S \rightarrow aS, S \rightarrow a, S \rightarrow B, B \rightarrow bB, B \rightarrow b\}$.

4) Sveikųjų skaičių kalba

$V_T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}$;

$V_N = \{S, D, N\}$, S -sveikasis skaičius, D -skaitmuo, N -skaitmenų seka.

$P = \{D \rightarrow 0, D \rightarrow 1, D \rightarrow 2, D \rightarrow 3, D \rightarrow 4, D \rightarrow 5, D \rightarrow 6, D \rightarrow 7, D \rightarrow 8, D \rightarrow 9, N \rightarrow D, N \rightarrow DN, S \rightarrow N, S \rightarrow +N, S \rightarrow -N\}$;

Sveikųjų skaičių kalboje įvedami tokie pakeitimai:

1) $\beta = X_1$

$\beta = X_2$;

$\beta = X_3$;

$\beta = X_1 | X_2 | X_3$;

2) D keičiamas į $\langle \text{skaitmuo} \rangle$;

3) \rightarrow keičiamas į $::=$

4) $V_N = \{\langle \text{sveikasis skaičius} \rangle, \langle \text{skaitmuo} \rangle, \langle \text{skaitmenų seka} \rangle\}$

$\langle \text{skaitmuo} \rangle ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0$;

$\langle \text{skaitmenų seka} \rangle ::= \langle \text{skaitmuo} \rangle | \langle \text{skaitmuo} \rangle | \langle \text{skaitmenų seka} \rangle$;

$\langle \text{sveikasis skaičius} \rangle ::= \langle \text{skaitmenų seka} \rangle | \langle \text{skaitmenų seka} \rangle | \langle \text{skaitmenų seka} \rangle$.

Ši forma vadinama **BEKAUS–NAURO** forma (**BNF**).

5) Paskalio kalba

$V_T = \{A, B, \dots, Z; a, b, \dots, z; 0, 1, \dots, 9; +, -, (,), \dots, \langle, \rangle, =, >, <, \dots, :=\}$; baziniai žodžiai (pvz., **and, repeat, procedure, program** ir kiti)

$V_N = \{\langle \text{programa} \rangle, \langle \text{ciklo sakiny} \rangle, \langle \text{sveikasis skaičius} \rangle, \dots\}$

BNF forma

$\langle \text{programa} \rangle ::= \langle \text{programos antraštė} \rangle$

$\langle \text{programos antraštė} \rangle ::= \langle \text{programos blokas} \rangle$.

$\langle \text{programos antraštė} \rangle ::= \text{program} \langle \text{vardas} \rangle$;

$\langle \text{vardas} \rangle ::= \langle \text{raidė} | \langle \text{vardas} \rangle \langle \text{raidė} \rangle | \langle \text{vardas} \rangle \langle \text{skaitmuo} \rangle$;

$\langle \text{skaitmenų seka} \rangle ::= \langle \text{skaitmuo} \rangle | \langle \text{skaitmenų seka} \rangle \langle \text{skaitmuo} \rangle$.

Modifikuota BNF forma:

$\alpha ::= \{\beta\}^+$, kur $\alpha ::= \beta \beta \dots \beta$, $n \geq 1$;

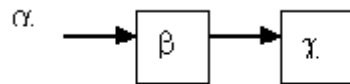
$\langle \text{skaitmenų seka} \rangle ::= \{\langle \text{skaitmuo} \rangle\}^+$;

Sąvokos pakartojimas n kartų reiškiamas $\alpha ::= \{\beta\}^*$, $n \geq 0$,

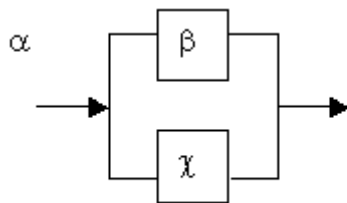
pvz., $\langle \text{vardas} \rangle ::= \langle \text{raidė} \rangle \{ \langle \text{raidė} \rangle | \langle \text{skaitmuo} \rangle \}^*$.

Sintaksinės diagramos

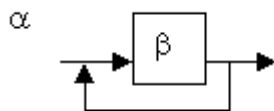
$\alpha ::= \beta \chi$



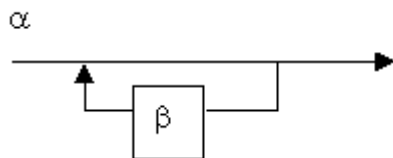
$\alpha ::= \beta | \chi$



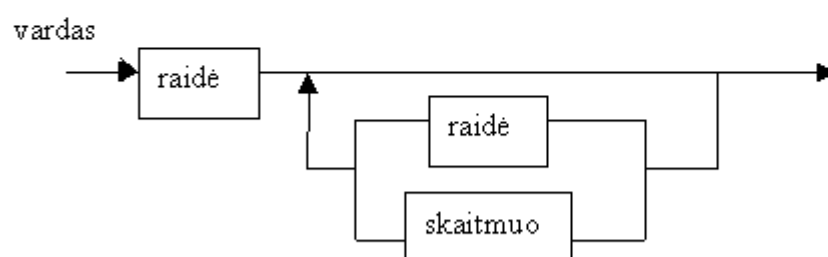
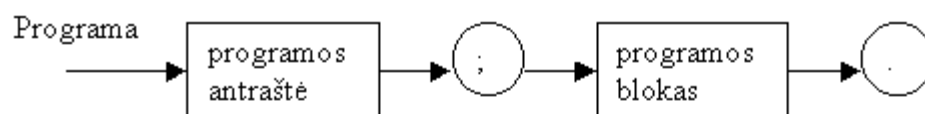
$\alpha ::= \{ \beta \}^+$



$\alpha ::= \{ \beta \}^*$



Stačiakampiais vaizduojami neterminaliniai simboliai, apskritimais–terminaliniai simboliai.



5. PASKALIO PROGRAMAVIMO KALBA

Paskalio programavimo kalbą (*Standard Pascal*) sukūrė šveicarų mokslininkas N. Virtas (Niklaus Wirth). Papildžius ją praktiniais elementais atsirado *Turbo Paskalis*. Paskutinioji jo versija *Turbo Paskalis 7.0* (*Turbo Pascal 7.0*), skirta personaliniams kompiuteriams, šiandien plačiai vartojama.

Norint išmokti programuoti reikia žinoti pagrindines programos dalis. Programą galima būtų palyginti su kulinariniu receptu.

Kulinarinis receptas:

1. ingredientai;
2. veiksmų nurodymai

Programa:

1. duomenys;
2. sakiniai.

Programoje sakiniiais yra nurodomi veiksmai, kuriuos privalu atlikti su duomenimis. Iš šio palyginimo tampa aišku, jo pirmiausia mes turime išnagrinėti Paskalio kalboje naudojamus duomenų tipus.

5.1. Duomenų tipai

Duomenų rūšį (sakoma – duomenų tipą) tipą programavimo kalbose sudaro: 1) reikšmių aibė ir 2) operacijų aibė.

Duomenys pagal tipus skirstomi į:

- paprastuosius;
- sudėtinius (jie dar vadinami struktūriniais);
- rodyklės tipą.

Paprastuosius duomenų tipus sudaro:

- sveikasis tipas (integer);
- loginis tipas (boolean);
- simbolinis tipas (char);
- atkarpos tipas;
- vardinis tipas;
- realusis tipas (real).

Pirmieji keturi tipai dar vadinami diskrečiais arba skaliariniais tipais. Tai reiškia, kad jų reikšmių aibė yra sutvarkyta nuo pradžios iki galo.

Duomenų tipą apibūdina:

- 1) reikšmių aibė;
- 2) operacijų (su šitomis reikšmėmis) aibė.

Šia struktūra remsimės apibūdindami kiekvieną duomenų tipą atskirai.

Sveikasis tipas

Integer

- 1) reikšmių aibė $[-2^{15}; 2^{15}-1]$.
- 2) Operacijų aibė:
 1. $+, -, *, \text{div}, \text{mod};$
 2. santykio operacijos $<, <=, >, >=, =, <>;$

3. funkcijos:

- *succ(i)* (jos rezultatas po i einanti reikšmė);
- *pred(i)* (jos rezultatas prieš i einanti reikšmė);
- *abs* (absoliutinis dydis);
- *sqr(i)* ($=i^2$).

Sveikiesiems skaičiams galima taikyti ir šias funkcijas, tik reikia nepamiršti, kad rezultatas bus **real** tipo.

- *sqr(i)* (\sqrt{i});
- *arctan(i)* (arctg(i));
- *sin(i), cos(i)*;
- *exp(i)* (e^i);
- *ln(i)*.

Šiam tipui priklauso konstanta **maxint**= $2^{15}-1$.

Dar yra skiriami tokie sveikojo tipo variantai:

Longint, kurio reikšmių aibė yra $[-2^{31}; 2^{31}-1]$.

Shortint, kurio reikšmių aibė yra $[-2^7; 2^7-1]$.

Word, kurio reikšmių aibė yra $[0; 2^{16}-1]$.

Byte, kurio reikšmių aibė yra $[0; 2^8-1]$.

Jų operacijų aibės sutampa su **integer** tipo operacijų aibe.

Vardinis tipas

1. Jo reikšmių aibę sudaro vartotojo sukurti vardai.

type diena=(pr,an,tr,kt,pe,št,se).

Vardai pr,an,tr,kt,pe,št,se priklausys vardinio tipo reikšmių aibe.

2. Operacijų aibė:

- santykio operacijos;
- priskyrimo operacija;
- funkcijos: *succ(i)*, *pred(i)*, *ord(i)* (pastaroji funkcija pateikia vardinio tipo Nr.).

Pvz.

type diena=(pr,an,tr,kt,pe,št,se);

var d,d1:diene;

Tuomet programoje bus galima nurodyti tokias operacijas:

- 1) **d:=pr**;
- 2) **d1:=succ(pr)** šis sakinytis atitinka **d1:=an**;
- 3) **if d=se then d:=pr**
 else d:=succ(d);
- 4) **ord(tr)=2**;
- 5) **for d:=pr to se do ...**

Tačiau reikėtų nepamiršti, kad šios operacijos gali būti neapibrėžtos, pvz., *pred(pr)* ir *succ(se)*

Loginis tipas (boolean)

- 1) reikšmių aibė: **type** *boolean*=(*true*, *false*).
- 2) Operacijų aibė:
 - santykio operacijos (*false*<*true*→*true*);
 - funkcijos *succ(i)*, *pred(i)*;
 - **or**, **and**, **not** (loginė sudėtis, loginė daugyba, neiginys).

Simbolinis tipas (char)

- 1) reikšmių aibė: simboliai, kurie vaizduojami tarp apostrofų, pvz., 'A', '1'.
- 2) Operacijų aibė:
 - santykio operacijos;
 - funkcijos *ord(i)* (pateikia simbolio kodą ASCII kodavimo sistemoje.) ir *chr(i)* pateiktą kodą paverčia simboliu.
pvz., *ord(A)*=41; *chr(41)*=A;
chr(ord(c))≡*c*.

Atkarpos tipas

- 1) Atkarpa apibrėžiama, apribojusi, bet kuri kitą diskretųjį tipą (jis vadinamas baziniu tipu), t. y. nurodant jos apatinį ir viršutinį rėžius. Apatinis rėžis turi būti ne didesnis už viršutinį. Pvz.
 - **type** *i*100=1..100;
 - **type** *raide*='a'..'z';
 - galima vartoti vadinamąjį anoniminį tipą:
var *i*:1..100;
 - **var** *raide*:'a'..'z'.
- 2) Su atkarpos tipo kintamaisiais atliekamos tokios pat operacijos, kaip ir su jo baziniu tipu.

Realusis tipas

Real

- 1) Reikšmių aibė yra $[2,9 \cdot 10^{-39}; 1,7 \cdot 10^{36}]$.
- 2) Operacijų aibė:
 - +, -, *, /;
 - santykio operacijos;
 - funkcijos
 - 1) *abs(x)*;
 - 2) *sqr(x)*;
 - 3) *sqrt(x)*;
 - 4) *sin(x)*; *cos(x)*;
 - 5) *arctan(x)*;
 - 6) *exp(x)*;
 - 7) *ln(x)*;
 - 8) *round(x)* (skaičiaus apvalinimas, pvz., *round(4,1)*=4, *round(4,5)*=5);

- 9) $\text{trunc}(x)$ (pašalina skaičiaus trupmeninę dalį, pvz., $\text{trunc}(4,1)=4$, $\text{trunc}(4,8)=4$).

8) ir 9) funkcijas sieja toks ryšys:

$$\text{round}(x) = \begin{cases} \text{trunc}(x+0,5), & \text{kai } x \geq 0; \\ \text{trunc}(x-0,5), & \text{kai } x < 0. \end{cases}$$

Dar egzistuoja tokie realiojo tipo variantai:

Single, jo reikšmių aibė $[1,5 \cdot 10^{-45}; 3,4 \cdot 10^{38}]$;

Double, jo reikšmių aibė $[5,0 \cdot 10^{-324}; 1,7 \cdot 10^{306}]$;

Extended, jo reikšmių aibė $[3,4 \cdot 10^{-4932}; 6,1 \cdot 10^{4932}]$.

Jų operacijų aibės sutampa su **real** tipo operacijų aibe.

Operacijų prioritetai

1. **not**;
2. daugybos operacijos: *****, **/**, **div**, **mod**, **and**, **shr**, **shl**.
 shr – Shift Right (postūmis dešinėn);
 shl – Shift Left. (postūmis kairėn).
 Operacijos: $i \text{ shl } j$ ir $i \text{ shr } j$ pastumia i reikšmę per j skilčių.
3. Sudėties operacijos: **+**, **-**, **or**, **xor**.
 xor – eXclusive OR (griežta disjunkcija), jos reikšmių lentelė tokia:

A	B	A xor B
True	True	False
True	False	True
True	True	True
False	False	False

4. Santykio operacijos: **=**, **<>**, **<**, **<=**, **.**, **>=**.

Konstantos, nurodančios reikšmes

Integer

$i:=10$;

$i:=255$;

Pastarąjį galima užrašyti ir šešioliktaine sistema, $i:=\text{\#FF}$.

Real

$r:=1.5$ (dešimtainės trupmenos trupmeninė dalis atskiriama tašku);

$r:=1\text{E-}15$, šis užrašas atitinka $r:=1 \cdot 10^{-5}$ arba $r:=0.00001$;

$r:=1.2\text{E+}20$.

Char

$c:=\text{'A'}$;

$c:=\text{chr}(55)$ (atitinka $c:=\text{'7'}$);

$c:=\text{\#55}$. (atitinka $c:=\text{'7'}$).

Boolean

$b:=\text{true}$;

$b:=\text{false}$.

Vardinis tipas

v:=pirmadienis;

v:=žalia.

Vardai uždavinio formuluotėje ir programoje

$$1) \quad S = \sum_{i=1}^n a_i;$$

$$a_0=1;$$

$$a_{k+1}=2*a_k+1, \quad k=0,\dots,n-1;$$

$$S = 0;$$

$$a = 0;$$

$$S = S + a;$$

$$a = 2 * a + 1;$$

} n kartų

$$S:=0;$$

$$a:=0;$$

for i:=1 to n do**begin**

$$S:=S+a;$$

$$a:=a*2+1$$

end;

$$2) \quad y_{15}=?$$

$$y_{n+1} = y_n + \frac{x_{n+1}^2 + 0.1x_n + 0.4}{y_n};$$

$$x_{n+1} = x_n + \frac{0.2}{z_{n+1}};$$

$$z_{n+1} = 0.1z_n + 1;$$

$$x_0=0,1; y_0=0,27; z_0=0.47;$$

$$y=0.27;$$

$$x=0.1;$$

$$z:=0.47;$$

for i:=1 to 15 do**begin**

$$z:=0.3*z+1;$$

$$x1:=x+0.2/z;$$

$$y:=y+(x1*x1+0.1*x+0.4)/y;$$

$$x:=x1$$

end;

$$3) \quad \text{Rasti } y = \sqrt[3]{x} \text{ (pagal rekurentinę formulę):}$$

$$y_{n+1} = \frac{1}{3} \cdot \frac{2y_n^3 + x}{y_n^2};$$

$$y_0=1 \text{ (tikslumu } 10^{-4});$$

```

y1:=1;
repeat
y:=y1;
y1:=(2*y*y*y+3)/(3*y*y)
until abs(y-y1)<1E-4.

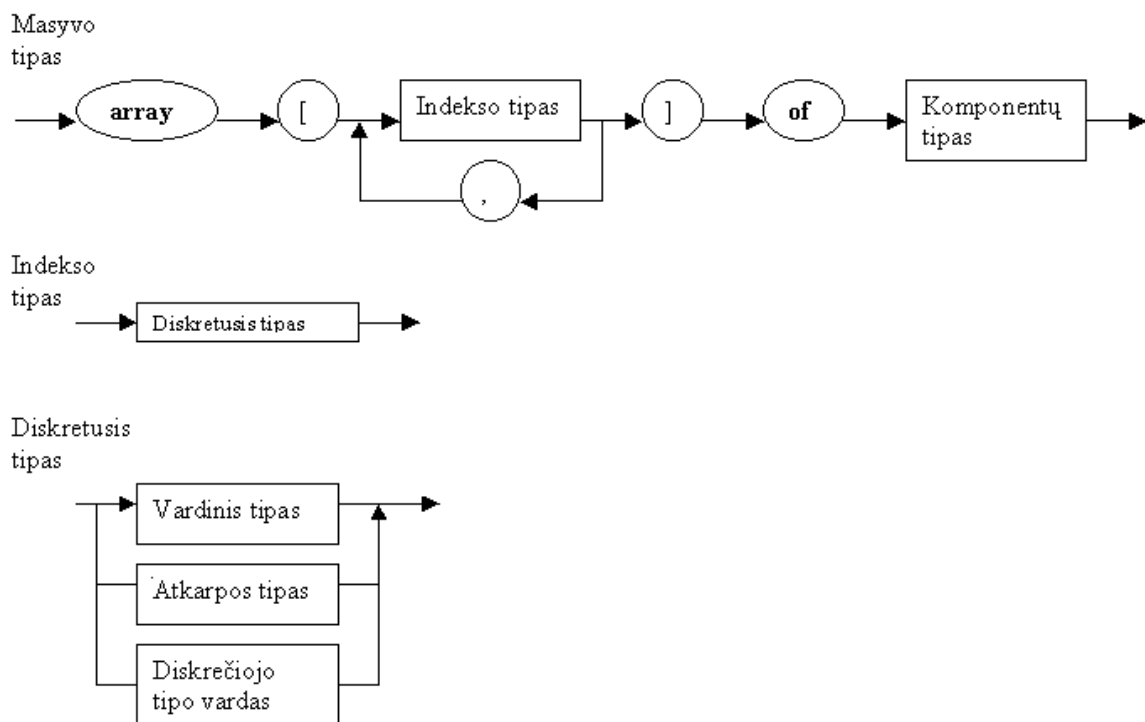
```

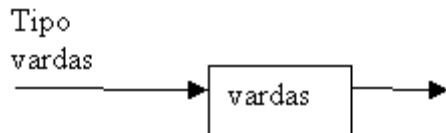
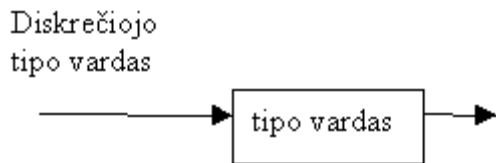
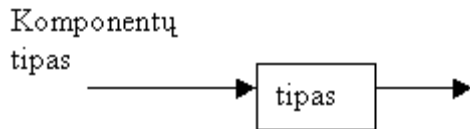
Struktūriniai duomenų tipai

Struktūriniai duomenų tipai skirstomi į

- masyvo tipą;
- įrašo tipą;
- aibės tipą;
- objekto tipą
- failo tipą.

Masyvo tipas





```

type masyvas10=array [1..10] of real;
      lentelė=array [(p,a,t,k,pe),1..3] of integer;
      mėnuo=(s,v,k,b,g,bi,l,r,ru,sp,la,gr);
      mm=array [mėnuo] of integer;
      mas=array ['A'..'Z'] of integer;
var m1:masyvas10;
      l1:lentelė
  
```

Jei masyve nurodytas vienas indekso tipas, masyvas vadinamas vienmačiu, pvz., m1, jei du indekso tipai atitinkamai dvimačiu, pvz., jei n indeksų tipų masyvas bus n -matis.

Laužtiniuose skliaustuose nurodomi masyvo rėžiai, t.y. masyvo m1 apatinis rėžis yra 1, o viršutinis–10. Turbo paskalyje masyvai yra statiniai, tai reiškia, kad jų dydis turi būti žinomas kompiliavimo metu. Dinaminio masyvo rėžiai įvedami.

Masyvus galima užrašyti tokiu būdu:

```

var a:array [1..10,1..20] of real
var a:array [1..10] of array [1..20] of real
  
```

Šie abu pateikti užrašai yra ekvivalentūs.

Operacijos

Su masyvai atliekama tik priskyrimo operacija, visi kiti veiksmai atliekami su masyvo elementais. Priskyrimo operaciją galima atlikti, tik kai abiejų masyvų tipai tapatūs. Pvz.

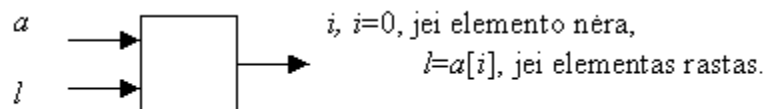
```

var m1,m2:array [1..10] of real
      m1:=m2.(Tą patį galima užrašyti ir tokiu būdu:for i:=1 to 10 do m1[i]:=m2[i]).
  
```

Užduotis:elemento paieška masyve

*Duota:*masyvas $a[1..n]$ ir elementas l .

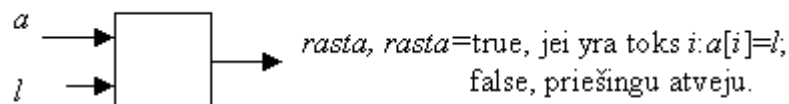
*Rasti:*ar elementas l yra masyve, jei taip pateikti jo indeksą.



```

const n=100;
var a:array [1..n] of real;
    l:real;
    i,k:integer.
begin
  ...
  i:=0;
  while (i=0) and (k<=n) do
    if a[k]=l then i:=k
      else k:=k+1;
  end.

```



```

...
rasta:=false;
k:=1;
while (k<=n) and (not rasta) do
  if a[k]=l then rasta:=true
    else k:=k+1;
...

```

Simbolių eilutės

Simbolių eilutės tai viena masyvo rūšių. Skiriamos trijų rūšių eilutės:

- 1) Pastovaus ilgio eilutės
array [1..n] of char;
- 2) Dinaminės eilutės
string[n], max(n)=255;
- 3) ASCIIZ eilutė
array[0..n] of char; max(n)=65535.

```

var e1,e2:array[1..7] of char;
    s1,s2:string[3];
    s:string;

```

(jei nenurodomas dinaminės eilutės ilgis, laikoma, kad ji gali turėti 255 simbolių)
s:='Vilniaus matematikos ir informatikos fakultetas';
s1:='AB';
s:='' (dinaminėms eilutėms galima priskirti tuščią eilutę);
s2:='ABCD' (bus priskirta tik 'ABC', nes nurodytas ilgis yra 3);

e1:='ABC' šis priskyrimas neteisingas, nes pastovaus ilgio eilutėms galima priskirti tik reikšmę, kuri turi tiek elementų, kiek yra nurodyta apraše, šiuo atveju tai yra 7.
e2:='ABCDEFGG' šis priskyrimas galimas.

Su dinaminėmis simbolių eilutėmis galima atlikti kelias papildomas operacijas, kurios neatliekamos su masyvu:

- 1) s1:='Vilnius';
- 2) Su dinaminėmis eilutėmis galima atlikti santykio operacijas
s1:='Vilnius';
s2:='Kaunas';
if s1>s2 **then** ...
Lyginami simboliai iš kairės į dešinę. Jei eilutė ilgesnė, tai kiekvienas jos simbolis, kurio neatitinka kitos eilutės simbolis turi didesnę reikšmę.
- 3) Apibrėžta sąjungos operacija, nurodoma simboliu +.
pvz. 'Vilnius'+'Kaunas'='Vilnius Kaunas'.
- 4) Dinaminėms eilutėms dar yra apibrėžtos šios funkcijos:
 - *length(s)*; pateikia eilutės ilgį;
s:='Vilnius';
length(s)=7;
 - *pos(s,s1)* jei s įeina į s1, tai pateikia pirmojo s simbolio eilutėje s1 numerį, priešingu atveju pateikia 0.
pvz.
užduotis: tarpus eilutėje pakeisti nuliais.
s:=' 286145';
while *pos(' ',s)*>0 **do** s[*pos(' ',s)*]:=0.
- 5) Galima naudoti ir šias procedūras:
 - *delete(s,k,n)* iš eilutės s pradedant k-uoju elementu išbraukiama n elementų.
pvz.
s:='informatika';
delete(s,1,2); (rezultatas bus s='formatika');
 - *insert(s,s1,n)* (įterpia eilutę s į eilutę s1 pradedant n-uoju elementu)
pvz.
s:='abcd'
insert('pq',s,3); (rezultatas s='abpqcd').

Irašo tipas

Įrašo tipo reikšmės sudaromos iš komponentų skaičiaus. Komponentai vadinami laukais.
Pvz.

type data=record

metai:integer;
mėnuo:1..12;
diena:1..31

end;

fakultetotipas=(...,EF,...MIF,...);

studentas=**record**

vardas, pavardė:string[25];
gimimodata:data;
fakultetas:fakultetotipas;


```

        kursas:1..4
    end;
var s:studentas;

s.vardas:='Petras';
s.pavardė:='Petraitis';
s.kursas:=1;
s.gimimodata.metai:=1983.

```

Tą patį galima užrašyti ir su sakiniu **with**:

```

with s do
begin
    vardas:='Petras';
    pavardė:='Petraitis';
    kursas:=1;
    gimimodata.metai:=1983
end.

```

Įrašą gali sudaryti pastovioji ir variantinė dalys. Įrašo variantinė dali aprašoma su **case** sakiniu:

```

type fig=(skritulys, kvadratas, stačiakampis, trikampis);
    figūra=record
        plotas:real;
        f:fig;
        case fig of
            skritulys : (r:real);
            kvadratas : (a:real);
            stačiakampis : (b,c:real);
            trikampis : (a1,a2,a3:real)
        end;

```

```

var I:figūra;

```

```

    I.f:=skritulys;
    I.r:=1.5;
...
    case I.f of
        skritulys : I.plotas:=3.14*sqr(I.r);
        kvadratas : I.plotas:=sqr(I.a);
        stačiakampis : I.plotas:=I.b*I.c;
        trikampis : begin
            p:=(I.a1+I.a2+I.a3)/2;
            I.plotas:=sqr(p(p-I.a1)(p-I.a2)(p-I.a3))
        end;
    end;

```

Operacijos

Su įrašai atliekama tik priskyrimo operacija, visos kitos atliekamos su įrašo laukais.
Pvz.

```
var s1,s2:studentas;  
    s1:=s2.
```

Aibės tipas

type abc=**set of** (a,b,c);
Bazinis tipas (nurodomas skliausteliuose, turi būti diskretus).

```
var k:abc;  
    k:=[a,b];  
    k:=[];
```

```
type produktai=(obuoliai, bananai, ledai, šokoladas, grietinėlė, sausainiai, cukrus);  
    desertas=set of produktai;  
var grietininiailedai, obuolių džemas, saldumynai:desertas;  
...  
grietininiailedai:=[ledai, šokoladas];  
saldumynai:=[sausainiai, ledai];  
grietininiailedai:= grietininiailedai+[bananai].
```

Operacijos

+ (sąjunga \cup);
* (sankirta \cap);
– (skirtumas);
in (priklauso \in);
=, \neq ;
 \leq (poaibis);
 \geq (viršaišis).

Pvz.

```
var s:set of 0..9;  
    n,sk,k:integer;  
begin  
    readln(n);  
    s:=[];  
    k:=0;  
    repeat  
        sk:=n mod 10;  
        if (sk in s) then begin  
            s:=s+[sk];  
            k:=k+1  
        end;  
        n:=n div 10;  
        writeln(n);  
    until n=0;  
end.
```

Failo tipas

file of...

Type **FFF = file of** <failo tipas>;

“**Failo**” vardas:

- 1) Failas
- 2) Rinkmena
- 3) Byla

Failas – tai vienodo tipo kintamųjų seka

Failo komponentai: irasai;

Masyvo komponentai: elementai;

Irašo komponentai: laukai;

Failo tipo kintamasis = failas

Buferinis kintamasis

F1^

F2^

F3^

Failai Paskalyje yra nuoseklūs

Jei norime nuskaityti n-tąjį elementą, tai reikia pirma nuskaityti (n-1) elementą. Tai yra Paskalyje failai nuoseklūs.

Operacijos su failais

1. Užrašymas (ir sukūrimas) į failą.

Var f1: file of char;

Begin

f1^:= 'a';

put(f1);

Procedūra put iš buferio duomenis perrašo į failą.

Iš buferio duomenis perrašius į failą, buferis tampa neapibrėžtas.

f1^:= 'a';

=> write (f1, 'a');

put (f1);

2. Nuskaitymas iš failo.

{true, jei failo rodyklė yra failo pabaiga

eof (f1)=> {false, jei ne.

reset (failo vardas); - failo rodyklė nukeliama į failo pradžią

var f: file of real;

x, s: real;

```

begin
    reset (f);
    s := 0;
while not eof (f) do
    begin
        read(f,x);      s := s + f^;    x := f^;
        Assign (f, 'c:XTP\failas,duomenys');
        ↗                ↖
        vidinis failas      išorinis failas

        s := s + x;      get (f);      get (f);
    end
end;

```

Išoriniai ir vidiniai (fiziniai ir loginiai) failai

Assign (failo kintamasis, failo vardas)

Pvz.:

Close (failo vardas) – uždaro failą

Pvz.:

Var t: text; {beveik, tas pats,kas var t: file of char; }

“**text**” – turi savybes:

- 1) Turi eilutės pabaigos simboli
- 2) Eoln (t) => {true, false
- 3) Writeln (t) įrašomas eilės pabaigos simbolis
- 4) Readln (t) sustojama ties kitos eilės pradžia

Procedūra Q(c)

R – veiksniai baigus eilutę

While not eof (x) do

Begin

While not eof (x) do

Begin

Read (x, ch);

Q (ch);

End;

R ;

Readln (x);

End;

Var x: text;

Ch: char;

Ilgis, maxilgis: integer;

Begin

Maxilgis:= 0;

While not eof (x) do

Begin

```

    Ilgis:= 0;
    While not eof (x) do begin
        Read (x, ch);
        Ilgis:= ilgis + 1;
    End;
    If ilgis > maxilgis then maxilgis:= ilgis
    Readln;
    End;
End;

```

```

Var input, output: text;

```

```

Assign (input, 'klaviatūra');
Assign (output, 'monitorius');

```

```

Write (output, 'a'); - išveda į ekraną
Write ('a');

```

```

Jei ;

```

Jei **nėra** failo vardo, tai darbas vyksta **ne** su failais, o su klaviatūra ir monitoriumi. Paprasčiausiai “**input**” ir “**output**”, kai dirbame **ne** su failais praleidžiami.

```

Write (x); - išvesti į ekraną
Write (f,x) – išvesti į failą f

```

```

Write ('A', 'ABC', '1,S', i);

```

“i” pirma paverčiamas į simbolius, o tik poto išvedamas (verčiamas iš vidinės į išorinę formą).

```

Var f: file of integer;
    T: text;
    Fc: file of char;
Readln (T) – taip rašyti galima
Readln (Fc) – negalima
Readln (f) – negalima

```

5.2. Valdymo struktūros

Struktūrinės schemas

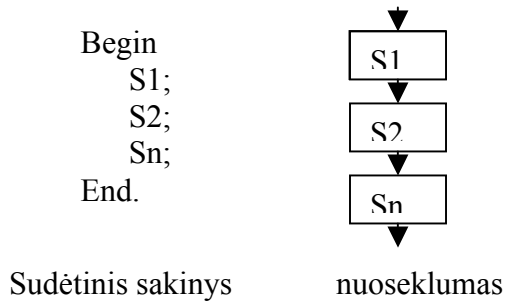
Struktūrinė schema parodo konstrukcijos atliekamus veiksmus.

1. Elementarus veiksmas
2. Šakojimasis
3. Kartojimas arba ciklas

4. Nuoseklumas

Visos šios 4 struktūros adinamos “D stuktūromis”(nuo anglų mokslininko Dijkstra).
Corrado Boehmas, Guiseppi Jacopini 1966 suformavo teorema:

Bet kuri programa gali būti išreikšta aukščiau esančiomis 4 struktūromis.



Kalbos konstrukcijos, išreiškiančios seką (kaip valdymo struktūrą), sintaksei.

Sąlyginis sakinyss

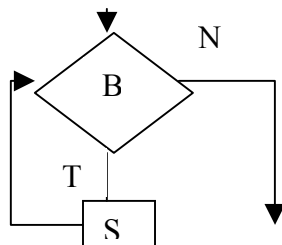
If B then S1 else S2;

If B then S1;

If B1 then [(if B2 then S1) else S2;]

TP traktuos pagal laužtinius skliaustus

While B do S;



Ciklo sintaksė

If B then begin

S;

While B do S;

End;

Ciklo semantika

(atliekami veiksmai)

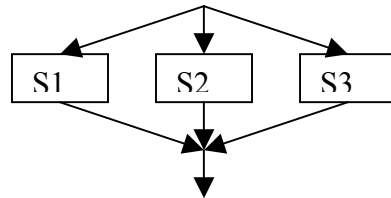
} while apibrėžimas

Rekursija ↑↑

D struktūros: 1)Seka (begin.....end;);
 2) Šakojimasis (if);
 3) Ciklas (while);

- D' struktūros
- 1) Daug šakų (var, case);
 - 2) Ciklas (repeat);
 - 3) Ciklas (for);

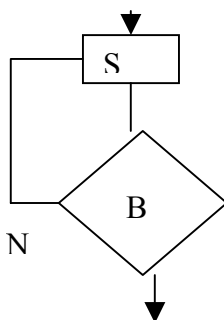
Variantinis sakinyys (case).



0, i = 1
 1, i = 3
 F = 2, i = 4
 7, i = 5
 10, i = kitas atvejis

case i if variantas
 variantas
 žymė
 1: F:= 0;
 3: F:= 1;
 4: F:= 2;
 5: F:= 3;
 else F:= 10;
 end;
 V2: S2;
 V3: S3
 else S4
 end;
 Repeat


al ir pan.




repeat S until B;



begin
 S;
 if not B then repeat S until B;
 end;

Jei norime  pakartoti n kartų tada:

i:= 0;
 ; } n kartų pakartos

```

i:= i +1
until i >= n

```

Repeat - vyriškas ciklas, o while – moteriškas.

For

```

      ↙ pradinė reikšmė
For v:= e1 to e2 do S;
      ↑
ciklo parametras      ↘ galutinė reikšmė

```

e1, e2 – diskrečiojo tipo reikšmės.

```

For i:= 1 to 100 do S:= S+i;

```

Pabaigus ciklą, ciklo parametro reikšmė neapibrėžta.

```

For i:= 1 to 100 do begin
    S:= S+i;
End;

```

```

a:= 1; b:= 100;
for i:= a to b do begin
    S:= S+i;
End;

```

```

For v:= e1 to e2 do S;
    F1:= e1; F2:= e2;
if F1 <= F2 then begin
    v:= F1; S;
    while v <> F2 do begin
        v:= succ (v);S
    end
end;

```

5.3. Procedūros ir funkcijos

Program **ppp**;

```

Const    c1=...;
          c2=...;
type     t1=...;
          t2=...;
var       v1,v2: t1;
          v3,v4: t2;

```

```

procedure p(...);
function  f(...): ...;

```

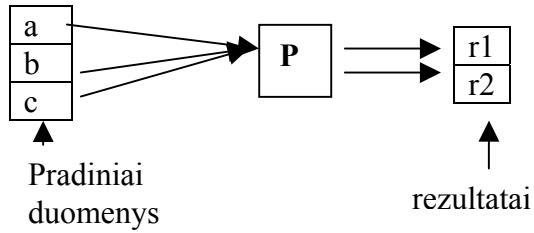
aprašų dalis

veiksmu dalis

• • •

•

Procedūros aprašas



- 1) Procedūra **P** turi 5 parametrus
- 2) Pradiniai duomenys : jie “imami”, bet nekeičiami.
- 3) Rezultatai : jie keičiami

```

Procedure Suma (a,b: integer; var s: integer);
Begin
    s:= a+b;
End;

```

Parametrų reikšmės ir parametrų kintamųjų palyginimas:

```
Program param;  
  Var a,b: integer;
```

Param . rekšmė	param. kintamasis
⇓	⇓

```

Procedure  p( x: integer; var y: integer);
Begin
    x:= x +1;
    y:= y +1;
    Writeln (x,y);
End;

```

```
Begin
  a:= 1;  b:= 1;

  p (a,b);
  writeln (a,b);
end.
```

```

Program   P;
  Const  n = ...;
  Type  mas = array [1...n] of integer;

```

```

Procedure pp (var a: mas; var min, max: integer);
  Var i: integer;
Begin
  Min:= a [1];
  Max:= min;
  For i:= 2 to n do
    If a[i] > max then max:= a[i]
    Else if a[i] < min then min:= a[i];
End;

```

Begin

```
.....
pp (AA, mi, ma)
end;
```

Jei procedūroje yra “var”, tai tiems kintamiems vieta neskiriama ir naudojama “pradinės reikšmės vieta”. Jei be “var” tai daroma duomenų kopija ir su ja dirbama.

Funkcijos

```
Var p, q, r : real;
S1, S2 : real;
Procedura suma (a, b, c : real; var suma: real);
Begin
    S:= a+b+c
End;
Begin
    Read (p, q, r);
    Suma (p, q, r, S1);
End;
```

```
Function suma (a, b, c : real) :real;
    Var r: real;
Begin
    r:= a+b;
    suma := r + c;
end;
    read (p, q, r);
    S1:= suma (p, q, r)*2;
```

```


$$\frac{r + c}{1) \text{ return } r + c}$$

2) suma := r + c
```

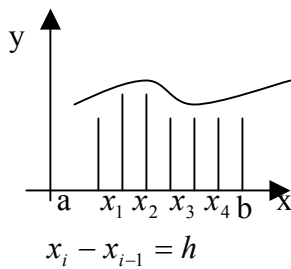
Parametrai

Skirstomi į:

- Formalūs – nurodomi funkcijos/procedūros apraše.
- Faktinius –nurodomi funkcijoje/procedūroje.

Taip pat parametrai skirstomi į:

- ✓ parametrus–reikšmes;
- ✓ parametrus–kintamuosius;
- ✓ parametrus–kintamuosius;
- ✓ parametrus–procedūras;
- ✓ parametrus–atviruosius masyvus;
- ✓ parametrus–atvirąsias **string** eilutes;
- ✓ parametrus–konstantas.



$$\begin{array}{c} x_0, x_1, \dots, x_n, x_{n+1} \\ \Downarrow \qquad \qquad \qquad \Downarrow \\ a \qquad \qquad \qquad b \end{array}$$

$$I = \int_a^b f(x) dx \quad \text{int}(a, b, n, f)$$

$$f(x_1) \cdot h + f(x_2) \cdot h + \dots + f(x_n) \cdot h + f(a) \frac{h}{2} + f(b) \frac{h}{2} = \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^n f(x_i) \right) \cdot h$$

```
function Int (a,b : real; n: integer; function f(x: real):real):real;
```

```
var h,s : real;
    i: integer;
begin
    h:= (b-a) / (n+1);
    s:= ( f(a) + f(b) + 0.5);
    for:= 1 to n do s:= s + f(a) + f(a + i * h );
    Int:= s* h;
End;
```

$$I_1 = \int_0^1 x^2 dx, \quad I_2 = \int_{-1}^1 x^2 dx, \quad I_3 = \int_0^1 \sin x dx$$

```
Program FFF;
Function f1 (x: real): real;
Begin
    f1 := x*x;
End;
Function f2 (x: real):real;
Begin
    f2 := sin(x);
end;
function Int (.....
begin.....
end;
begin
    writeln (int (0, 1, 100, f1));
    writeln (int (-1, 1, 200, f1));
    writeln (int (0, 1, 100, f2));
end.
```

Function **F** (x: real; var x: char; procedure p(x: real; var t: integer)):real;

Parametrai–atvirieji masyvai

Deklaruojami:

var m:array of real;

Atvirieji masyvai naudojami norint aprašyti funkcijas/procedūras skirtingo dydžio masyvams.

Pvz.,

function suma(var m:array of real):real;

var i:integer;

 s:real;

begin

 s:=0;

for i:=0 **to** high(m) **do** { funkcija high(i)–pateikia didžiausio elemento Nr. }

 s:=s+m[s];

 suma:=s;

end;

Tuomet funkcija tinkama visiems masyvams:

Pvz.,

var m1:array [0..99] of real;

 m2:array [-10..10] of real;

writeln(suma(m1));

writeln(suma(m2));

Naudojant atvirosius masyvus, keičiama masyvo numeracija:

High(m1)=99; (iš viso m1 turi 100 elementų);

High(m2)=20; (iš viso m2 turi 21 elementų).

Nr. funkcijoje	m1	m2	Nr. funkcijoje
0	0	-10	0
1	1	-9	1
99	99	10	20

Maksimalaus elemento inekso radimas

```

function maxind(var m:array of real):integer;
  var i,ind:integer;
begin
  ind:=0;
  for i:=1 to high(m) do      { funkcija high(i)–pateikia didžiausio elemento Nr.}
    if m[i]>m[ind] then ind:=i;
    maxind:=ind;
end;

writeln(m1[maxind(m1)]);
writeln(m2[maxind(m2)-10]);      {atlikta korekcija, nes funkcijoje naudojama
                                  numeracija nuo 0 iki 20}.

```

Parametrai–atvirosios string eilutės

Deklaruojama:
var s:openstring;

Naudojami kuriant funkcijas/procedūras skirtingai deklaruotiems masyvams.

Pvz.

```

procedure keistiAB(var s:openstring);
var i:integer;
begin
  for i:=1 to length(m) do
    if s[i]='A' then s[i]='B';
end;

var s1:string[5];
    s2:string[10];

```

```

keisti(s1);
keisti(s2);

```

Parametrai–konstantos

Deklaruojami:
const i:integer;

Pvz.

```

function f(const i:integer;i:integer):integer;
begin
  j:=j-1;
  f:=(i+1)*j;
end;

```

Parametrų konstantų savybės:

1. nekopijuojamas (tuo panašus į **var**);
2. jam negalima priskirti (panašiai kaip parametrams reikšmėms).

Vardų galiojimo sritys

Šiuolaikinės programavimo kalbos turi blokinę struktūrą (ir TP taip pat).

Pagrindini	{	Program ppp ;	}	vidinis blokas
Blokas		Var a, b, c : integer;		
	{	Procedure p;	}	
		Var a, b : integer;		
		Begin		
		a:= 2; b:= 2; c:= 2;		
	{	end;	}	
		begin		
	{	a:= 1; b:= 1; c:=1;	}	
		end.		

```

Program svirtys;
Var a, b, c, : integer;
Procedure p;
    Var a, b : integer
    Procedure r;
        Var a :integer;
        Begin
a:= 1; b:= 1; c:= 1;
        end;
    begin
        a:= 2; b:=2; c:=2; r; writeln (a, b, c)
    end;

```

```

procedure r;
var b, c : integer;
begin
    a:= 3; b:= 3; c:= 3;
    p;
    writeln (a, b, c, );
end;

```

```

begin
    a:= 4; b:= 4; c:= 4;
    r;
    writeln (a, b, c, );
end

```

```

{a,b,c {a,b {a}} {b,c}}
function f(a: t; b,c : real): real;

```

```

const
type

```

```

var                                lokalūs kintamieji
function
procedure
begin.....end;

```

```

if a = a then writeln ('normalu')
else else writeln ('neītikētina!');

```

```

program š. Efektas;
var i: integer;
function a: integer;
begin
    i:= i+ 1; a:= i;
end;

```

```

begin      {š. funkcijos efekta}
    i= 0;   ↓↓
            if a = a then writeln ('normalu')
            else writeln ('neītikētina!');
end.

```

```

Type nat = 0..maxint;
Function DBD (x,y :nat):nat;
Var r: nat
Begin
    While y <> 0 do
        Begin
            r:= x mod y;
            x:= y;
            y:= r
        End;
        DBD:= x;
    End;
End;

```

Su š. efektu

```

Function DBD x (var x, y : nat): nat;
    Var r: nat;
Begin
    While y <> 0 do
        Begin
            R:= x mod y;
            X:= y;
            Y:= r;
        End;
        DBD x:= x;
    End;
End;

```

6. ĪVADAS Ī ALGORITMU ANALĪZE

Algoritmų analizė – metodika, kaip nustatyti algoritmo greitį. Įvertinti algoritmo greitį reikia nusakyti, kaip vykdymo laikas didėja kaip funkcija nuo uždavinio matavimų (dydžio N):

$$t = f(N)$$

N – uždavinio matavimas. Pavyzdžiui, masyvo elementų skaičius, medžio mazgų skaičius, grafo viršūnių skaičius, rūšiuojamų vardų skaičius.

t- nurodo, kiek reikia laiko problemai N matavimų išspręsti.

Išanalizuosime tokio uždavinio algoritmą. Masyve A[1...n] reikia rasti elementą L. Čia uždavinio matavimas yra n.

```
i:= 1;    rasta:= false;
while (i <= n) and not rasta do
if A[i] = L then rasta:= true
else i := i + 1;
```

Galimi ciklo kartojimo skaičiai: 1, 2,, n

Ciklo kartojimo skaičiaus vidurkis:

$$\frac{1+2+3+\dots+n}{n} = \frac{1}{n} \cdot \frac{1+n}{2} \cdot n = \frac{n}{2} + \frac{1}{2};$$

$$t = \frac{n}{2} + \frac{1}{2} \approx \frac{n}{2} = \frac{1}{2} \cdot n = O(n);$$

Tiesinis algoritmas $O(n)$ \uparrow

$$g(x) = O(f(x));$$

$$g(x) = C \cdot f(x);$$

$$\lim_{n \rightarrow \infty} \frac{g(x)}{f(x)} = C;$$

7. MASYVO RŪŠIAVIMAS

Masyvo rūšiavimas (rikiavimas) – tai jo elementų išdėstymas didėjimo (mažėjimo) tvarka. Pagal rikiavimo metodą rūšiavimo algoritmai gali būti šie:

- Didžiausio elemento išrinkimo metodas (sudėtingumas $O(n^2)$);
- „Burbulo“ metodas (sudėtingumas $O(n^2)$);
- Greitojo rūšiavimo (Quick sort) algoritmas (sudėtingumas $O(n \cdot \log_2 n)$);
- Piramidės (Heap sort) metodas (sudėtingumas $O(n \cdot \log_2 n)$);

7.1. Mažiausio elemento metodas

1.....n-1 kartojamas žingsnis

- randamas minimalus [1.....n]
- keičiamas vietomis su i – tuoju

for i:= 1 to n-1 do


```

begin
minind:= i;
for j:= i+1 to n do if
A[ j ] < A [ minind] then minind := j;

```

```

r:= A[ i ];
A[ i ]:= A[ minind];
A[ minind]:= r;
End;

```

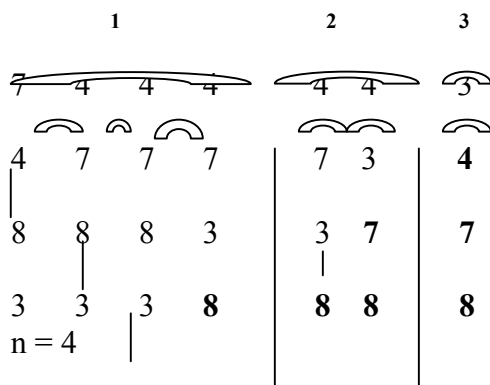
i = 1 2 n-1
n-1 n-2

$$T(n) = 1 + 2 + \dots + n - 1 = \frac{1+n-1}{2} \cdot (n-1) = \frac{n(-1)}{2} = \frac{n^2}{2} - \frac{n}{2} \approx \frac{n^2}{2} = Q(n^2)$$

Sudėtingumas kvadratinis

Vykdomo laikas \approx sudėtingumas

7.2. Burbulo metodas



```

For i:= 1 to n-1 do
For j:= 1 to n-i do
  if a [ j ] > a [ j + 1] then
  Begin
r:= a [ j ]; a [ j ]:= a [ j +1];
A [ j + 1]:= r;
  End;

```

i = 1 2 n-1
Vidinis ciklas n-1 , n-21

$$1 + 2 + \dots + (n-1) = \frac{1+n-1}{2} * (n-1) = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} \approx \frac{n^2}{2} = Q(n^2)$$

Kvadratinis algoritmas $Q(n^2)$

7.3. Greitojo rūšiavimo metodas

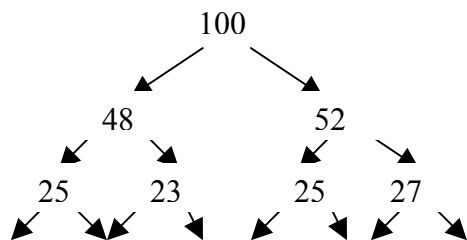
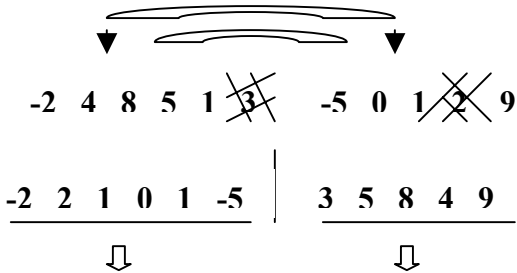
C.A. Hoare

Sudėtingumas $O(n \log_2 n)$

1	2	3	4	5	6	7	8	9	10	11
-2	4	8	5	1	3	-5	0	1	2	9

Ši rūšiavimą sudaro du etapai.:

- 1) Suskaidymas į dvi dalis
- 2) Tų dalių rūšiavimas



1. Suskaidymas į dvi dalis.

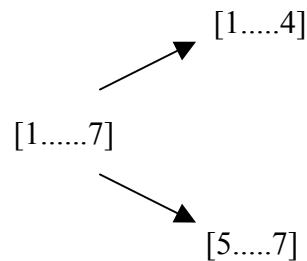
2. Šių dalių rūšiavimas.

1	2	3	4	5	6	7
4	8	1	5	2	4	7

4 4 1 5 2 8 7

1	2	3	4	5	6	7
<u>4</u>	<u>4</u>	<u>1</u>	<u>2</u>	<u>5</u>	<u>8</u>	<u>7</u>

1	2	3	4	5	6	7
4	4	1	2	5	8	7



i:= 1;

while a[i] < x do i:= i +1;

while a[j] > x do j:= j -1;

procedure **sort** (L, r: integer);

var i, j, x, w : integer;

```

begin
    i:= L; j:= r;
    x:= a[ (L +r) div2];
repeat
    while a[ i ] < x do i:= i + 1;
    while a[ j ] > x do j:= j + 1;
if i <= j then
    begin
        w:= a[ i ]; a[ i ]:= a[ j ];
        a[ j ]:= w;
        i:= i+ 1;
        j:= j+1;
    End;
    Until i > j;
    If L < j then sort ( L, j );
    If i < r then sort ( i, r );
End;
Program greitrusiavimas;
Const n=100;
Type mas = array [ 1.....n] of integer;
Var a:= mas;
    n: integer;
procedure sortmas ( var a: mas; min, max : integer);
    procedure sort( L, r :integer);
    begin
        .....
    end;
begin
    sort( min, max);
end;
Begin
Randomize;
For i:= s to n do b[ i ]:= random (3000);
Sortmas ( b, 1, n );
For i:= 1 to n do write [ b [ i ]:8);
Writeln;
End.

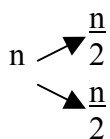
```

Metodo įvertinimas

N – matavimas (masyvo elementų skaičius)

C(N) – lyginimų sk.

1. geriausias atvejis



$$C(N) = 2 \cdot C\left(\frac{N}{2}\right) + (N) \otimes$$

$\nwarrow_2 \quad \swarrow_1$

$$C(2^n) = 2 \cdot C(2^{n-1}) + 2^n$$

$$C(2^{n-1}) = 2 \cdot C(2^{n-2}) + 2^{n-1}$$

$$C(2^0) = C(1) = 0;$$

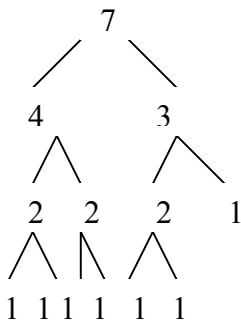
$$C(2) = C(2^1) = 2 \cdot C(2^0) + 2^1 = 2 \cdot 0 + 2 = 2;$$

tarkim, kad galioja

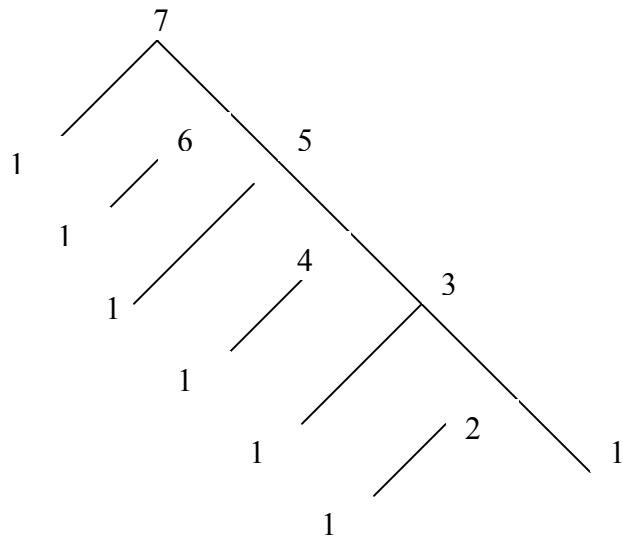
$$C(2^n) = 2^n \cdot n;$$

$$C(2^{n+1}) = 2 \cdot C(2^n) + 2^{n+1} = 2(C(2^n) + 2^n) = 2(2^n \cdot n + 2^n) =$$

$$= 2 \cdot 2^n (n + 1) = 2^{n+1} \cdot (n + 1)$$



Blogiausias atvejis



Geriausias atvejis

2. Blogiausias atvejis

$$C(N) = N + C(N-1)$$

$$C(N-1) = N - 1 + C(N-2)$$

$$C(2) = 2 + C(1)$$

$$C(1) = 0$$

- 1) Geriausias atvejis ($C(N) = N \log_2 N$)
- 2) Blogiausias atvejis ($C(N) = O(N^2)$)
- 3) Vidutinis atvejis ($O(N \log_2 N)$)