1. KOMPÜTERDƏ MƏSƏLƏLƏRİN HƏLL MƏRHƏLƏLƏRİ

Məsələlərin kompüterdə həll olunmaq üçün hazırlanması hələ də çox mürəkkəb olaraq qalmaqla bərabər həm də çoxlu zəhmət tələb edir. Bu məsələnin həlli bir neçə mərhələdən ibarətdir və onların əksəriyyəti kompüterin istifadə olunması ilə birbaşa əlaqədar deyildir. Bununla bərabər, məhz bu mərhələnin yerinə yetirilməsinə sərf olunan vaxt və zəhmət ümumiyyətlə bütövlükdə məsələnin kompüterdə həll olunmasına sərf olunan vaxt və zəhmətdən daha çox tələb olunur. Bu da onunla əlaqədardır ki, hər hansı məsələnin kompüterdə həll edilməsi üçün o gərək tərtib olunmuş olsun, yəni ilkin məlumatdan son nəticə almaq üçün göstərişlər ardıcıllığından ibarət sistem şəklində yazılsın. Əməliyyatların belə ardıcıllığı *alqoritm,* məsələnin maşında həll edilməsi üçün hazırlanması prosesi isə *alqoritmləşdirmə* adlandırılır.

Alqoritm anlayışı ilk dəfə *IX* əsrdə onluq say sistemində hesab əməllərinin yerinə yetirilməsi qaydalarını tərtib etmiş görkəmli özbək riyaziyyatçısı *Əl-Xarəzmi*nin adından götürülmüşdür. *XII* əsrdə bu qaydalar tərcümə olunaraq əvvəllər rum say sisteminin hökm sürdüyü *Avropa* ölkələrində tətbiq olunmağa başlamışdır.

Ümumiyyətlə, *alqoritm eyni tipli müəyyən məsələləri həll etmək üçün bir neçə əməliyyatlar sisteminin yerinə yetirilməsi ardıcıllığının təsviridir.* Alqoritmin bu tərifi o qədər də dəqiq olmasa da, praktik məqsədlər üçün bu tərifi qəbul etmək olar. Əgər ilkin məlumatdan son nəticə almaq üçün qayda mövcuddursa, onda bu tərif istənilən məsələnin həll edilməsinin mümkünlüyünü təsdiq edir. Alqoritmin üç əsas xassəsi vardır:

1. Alqoritm *kütləvi* olmalıdır, yəni tərtib olunmuş alqoritm ilkin verilənlərin qiymətlərinin geniş intervallarda

dəyişdikləri hallar üçün yararlı olmalıdır. Alqoritmin kütləviliyi müəyyən çərçivə daxilində başa düşülməlidir, yəni mütləq universal alqoritm yoxdur. Müəyyən bir məsələnin həlli üçün tərtib olunmuş alqoritm yalnız bu qəbildən olan müəyyən sinif məsələlərin həlli üçün yararlı ola bilər. Məsələn, xətti cəbri tənliklər sisteminin həlli üçün tərtib olunmuş alqoritm istənilən sayda tənlikdən ibarət xətti tənliklər sistemi üçün yararlı ola bilər, lakin bu o demək deyildir ki, həmin alqoritm, məsələn, xətti diferensial tənliklər sisteminin həlli üçün də yararlı olmalıdır.

2. Alqoritm *müəyyən* (*determinik*) olmalıdır, yəni verilənlərin emal olunması üçün ciddi əməliyyatlar ardıcıllığı mövcud olmalıdır. Bundan başqa alqoritmlərdə sıfra bölmə, mənfi ədədlərdən kvadrat kökalma, mənfi ədədlərin loqarifmlərinin hesablanması və s. kimi əməliyyatlara yol verilməməlidir.

3. Alqoritm *nəticəli* olmalıdır, yəni hər bir alqoritm üzrə hesablama nəticəsində cavab alınmalıdır, başqa sözlə, axtarılan nəticə alınmaqla alqoritmik proses öz axını ilə dayanmalıdır.

Alqoritmin bu üç əsas xassələrinin yerinə yetirilməsini istənilən misalda müşahidə etmək olar.

Məlumdur ki, verilmiş üç *a, b* və *c* ədədlərindən ən böyüyünün tapılması aşağıdakı ardıcıllıqla müəyyən edilir.

1. a və b ədədlərinin müqayisə et. Əgər a<b olarsa, onda x=b, əks halda isə x=a qəbul et və ikinci bəndi yerinə yetir.

2. x və c ədədlərini müqayisə et. Əgər x>=c olarsa, onda üçüncü bəndi yerinə yetir, əks halda x=c qəbul et və üçüncü bəndi yerinə yetir.

3. x ədədini nəticə kimi qəbul et və dayan.

Asanlıqla görünür ki, bu sadə alqoritmdə hər üç xassəyə əməl olunur. Belə ki, bu alqoritm *a, b* və *c* dəyişənlərinin

istənilən qiymətlərində doğrudur, bundan başqa ədədlərin müqayisəsi və ən böyük ədədin seçilmə ardıcıllığı o qədər müəyyəndir ki, səhv nəticə alınması ehtimalını tamamilə aradan qaldırır.

İki natural *x* və *y* ədədlərinin ən böyük ortaq bölənlərinin tapılmasından ibarət daha bir misala baxaq. Bu ardıcıllıqlar sistemi *Evklid* alqoritmi adlanır və o aşağıdakı bəndlərdən ibarətdir.

1. x və y ədədlərini müqayisə et. Əgər x>y olarsa, onda dördüncü bəndi yerinə yetir, əks halda ikinci bəndi yerinə yetir.

2. y və x ədələrini müqayisə et. Əgər y>x olarsa, onda beşinci bəndi yerinə yetir, əks halda üçüncü bəndi yerinə yetir.

3. Ən böyük ortaq böləni nəticə kimi qəbul et (z=x) və hesablamanı dayandır.

4. x=x-y qəbul et və birinci bəndi yerinə yetir.

5. y=y-x qəbul et və birinci bəndi yerinə yetir.

Baxılan misalda, əgər ən böyük ortaq bölən mövcuddursa, onda göstərilən alqoritm hökmən axtarılan nəticəni verəcəkdir. Ümumi halda məsələnin kompüterdə həll edilməsi bir neçə mərhələdən ibarətdir ki, onların ən əsasları aşağıdakılardır.

1. Məsələnin riyazi qoyuluşunun tərtib olunması. Bu mərhələdə fiziki, kimyəvi, iqtisadi və s. mahiyyət kəsb edən istənilən məsələ riyazi şəkildə tərtib olunur və ya məsələnin riyazi qoyuluşu hazır şəkildə proqramçıya verilir. Məsələnin riyazi təsvirində məsələnin məqsədi, ilkin verilənlər, onların dəyişmə hədləri, axtarılan kəmiyyətlərin xarakteristikaları, dəqiqliyi və məsələnin həllinin nəticəsinin təsvir olunma formaları göstərilməlidir.

2. Alqoritmin işlənməsi. Bu mərhələdə məsələnin həll

alqoritmi işlənir və ya mövcud alqoritmlər içərisindən münasib olanları seçilir. Bundan başqa alqoritmin müəyyən hissələrinin yerinə yetirilməsi üçün ədədi həll üsullarının seçilməsi və qiymətləndirilməsi vacibdir. Üsulun seçilməsi məsələnin həlli üçün tələb olunan dəqiqliyə və kompüterin imkanlarına uyğun olmalıdır.

3. Məsələnin alqoritminin blok-sxeminin tərtib olunması. Bu mərhələ əvvəlki mərhələnin məntiqi davamıdır, lakin bu mərhələdə əməliyyatlar daha dərindən araşdırılır. Bu mərhələnin yerinə yetirilməsi nəticəsində alqoritmin bütün məntiqi-riyazi aparatını özündə əks etdirən və əməliyyatlar ardıcıllığını göstərən müfəssəl blok-sxem tərtib edilir.

4. Proqramlaşdırma. Bu mərhələdə tərtib olunmuş alqoritm hər hansı bir alqorıtmik dildə proqram şəklində yazılır. Proqramlaşdırma mərhələsini yerinə yetirmək üçün ilkin sənəd kimi blok-sxem qəbul olunur. Bu mərhələyə daha müfəssəl dördüncü bölmədə baxılacaqdır. Proqramlaşdırma mərhələsi məsələnin kompüterdə həll olunmaq üçün hazırlıq mərhələsinin sonuncu - yekunlaşdırıcı mərhələsidir.

Baxılan bütün mərhələlər bir-biri ilə sıx əlaqədədir və vahid bir zəncir təşkil edir. Lakin bunların içərisində nisbətən böyük həcmli və məsul mərhələ blok-sxemin tərtibi və proqramlaşdırma mərhələləridir. Ona görə də növbəti bölmələrdə bir sıra misalların nümunəsində blok-sxemlərin və onların əsasında proqramların tərtib olunma texnikasını öyrənəcəyik.

2. MƏSƏLƏNİN HƏLL ALQORİTMLƏRİNİN İŞLƏNMƏSİ

Məsələnin kompüterdə həll olunmaq üçün hazırlanma mərhələləri içərisində alqoritmin blok-sxeminin işlənməsi ən vacib məsələlərdən biridir.

Alqoritmlərin təsvir olunması üçün əsasən iki formadan: *mətn və qrafik* formalardan istifadə olunur. Alqoritmlərin mətnlə yazılışında izahedici mətnlərlə müşaiyət olunan müxtəlif riyazi işarə və ifadələrdən istifadə olunur. Əslində alqoritmin bu təsvir forması məsələnin həlli qaydasının müfəssəl yazılışıdır. Alqoritmin belə təsvir formasına yuxarıda göstərdiyimiz ən böyük ədədin və ən böyük ortaq bölənin tapılması misallarının alqoritmlərini misal göstərə bilərik. Alqoritmlərin mətn forması ilə təsvir üsuluna *operator sxemi* üsulu və *alqoritmik dillər* də aiddir.

Alqoritmlərin qrafik təsvir üsulunda isə hər biri müəyyən funksiyanı yerinə yetirən və *blok* adlanan müxtəlif həndəsi fiqurlardan istifadə olunur. Bloklar nömrələnir və bir-biri ilə istiqamətlənmiş üfqi və şaquli xətlərlə birləşdirilir. Lazım gələrsə, blokların daxilində qısa izahatlar yazmaq olar. Blokların həndəsi ölçüləri və sxemlərin qurulma qaydaları mövcud standartlar əsasında yerinə yetirilir.

Ən çox istifadə olunan bloklar *cədvəl 1*-də göstərilmişdir.

Alqoritmin qrafik üsulla təsvir forması ən geniş yayılmış üsuldur. Alqoritmin mətnlə yazılış formasından fərqli olaraq bu üsul daha əyanidir və adətən hər bir bloka alqoritmik dillərdə hansı isə bir operator uyğun gəlir.

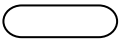
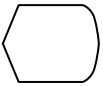
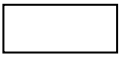
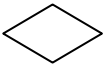
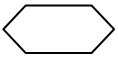
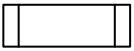
Riyazi mahiyyətindən asılı olaraq hesablama prosesləri *xətti, budaqlanan* və *dövrü* struktura malik olur. Bu hesablama proseslərinin blok-sxemlərinin özünəməxsus xüsusiyyətləri vardır və onların öyrənilməsi ilə məşğul olaq.

2.1. XƏTTI STRUKTURLU ALQORITMLƏR

Xətti hesablama proseslərində hər bir addımda əməliyyatlar bir-birinin ardınca yerinə yetirilir. Bu zaman aralıq nəticələr növbəti hesabatlar üçün ilkin verilənlər kimi istifadə olunur.Bu hesablama proseslərində adətən daxiletmə,

Cədvəl 1

Ən çox istifadə olunan bloklar

Blokun adı	**Blokun həndəsi təsviri**	**Blokun funksiyası**
Başlanğıc və son		Alqoritmin və ya proqramın başlanğıcı və ya sonu;
Daxiletmə		Monitordan ilkin verilənlərin daxil edilməsi
Proses (Hesablama bloku)		Blokun daxilində göstərilən düstur üzrə hesablamanın yerinə yetirilməsi
Budaqlanma (Müqayisə-etmə) bloku		Blokun daxilində göstərilən müqayisənin yerinə yetirilməsi
Modifikasiya (Dövr bloku)		Alqoritmdə dövrlərin təşkil edilməsi
Sənəd (Çapetmə bloku)		Nəticələrin çap qurğusunda çap edilməsi
Alt proqrama müraciət		Alt proqram üzrə əməliyyatın yerinə yetirilməsi
Birləşdirici		Bir səhifə daxilində blokların birləşdirilməsi
Birləşdirici		Müxtəlif səhifələrdə yerləşən blokların birləşdirilməsi

çapetmə və hesablama bloklarından istifadə olunur. Xətti hesablama proseslərinin blok-sxemlərinin ümumi görünüşü *şəkil 1*-də göstərildiyi kimidir.

Xətti hesablama proseslərini və onların blok-sxemlərinin tərtib olunma qaydasını öyrənmək üçün misala baxaq.

Misal 2.1. Tərəfləri *a, b* və *c* olan üçbucağın hündürlüklərini aşağıdakı düsturlar ilə hesablamaq üçün blok-sxem tərtib etməli:

$$h_a = (2/a)\sqrt{p(p-a)(p-b)(p-c)}$$

$$h_b = (2/b)\sqrt{p(p-a)(p-b)(p-c)}$$

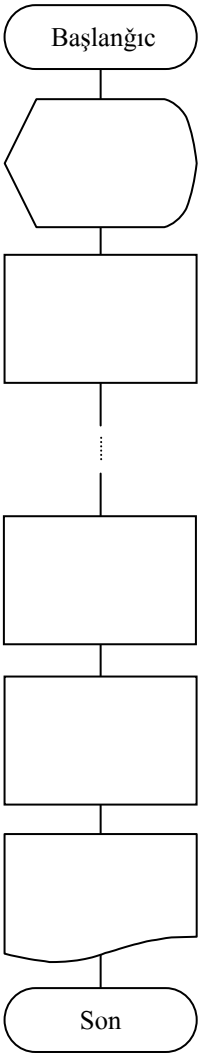
$$h_c = (2/c)\sqrt{p(p-a)(p-b)(p-c)}$$

Aydındır ki, məsələni kompüterdə həll etmək üçün *a, b, c* ilkin verilənlərini daxil etmək və sonra isə *p*-ni hesablamaq lazımdır. Bundan sonra isə h_a, h_b və h_c hündürlüklərini hesablamaq olar. Lakin burada fikir vermək lazımdır ki, $2\sqrt{p(p-a)(p-b)(p-c)}$ ifadəsi üç dəfə təkrar olunur. Ona görə də bu əməliyyatı üç dəfə təkrar etməmək üçün həmin ifadəni hesablayıb hər hansı bir dəyişənə mənimsətmək lazımdır. Onda məsələnin blok-sxemi *şəkil 2*-də göstərildiyi kimi olacaqdır.

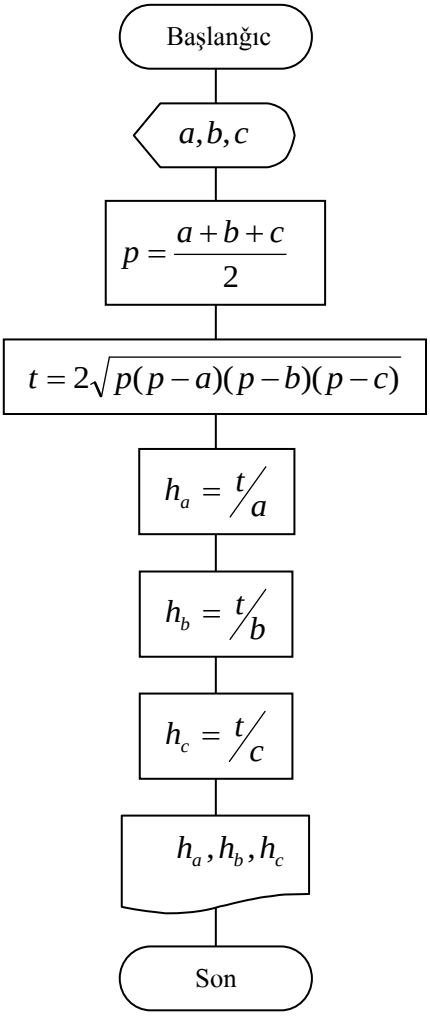
2.2. BUDAQLANAN STRUKTURLU ALQORİTMLƏR

Əgər hesablama proseslərində bu və ya digər mümkün həllərdən birini seçmək lazım gələrsə, onda belə hesablama prosesləri budaqlanan hesablama prosesləri adlanır. Sadə halda bu və ya digər şərtin yerinə yetirilib-yetirilməməsindən asılı olaraq hesablama istiqaməti seçilir. Bu şərt əvvəlcədən verilə bilər və ya hesablama prosesinin gedişində məsələnin özündə yarana bilər. İstənilən halda bu şərt müqayisə blokunda göstərilir. Müqayisə blokunun "*böyükdür*", "*kiçik*-

Xətti strukturlu alqoritmlərin təsviri



Şəkil 1



Şəkil 2

dir" və "*bərabərdir*" münasibət əməliyyatlarına uyğun üç çıxışı vardır. Xüsusi halda blokun iki çıxışı da ola bilər ki, bunlar müqayisə əməliyyatının yerinə yetirilməsinə ("*hə*") və yerinə yetirilməməsinə ("*yox*") uyğun gəlir. Budaqlanan hesablama proseslərinin blok-sxemlərinin ümumi görünüşü *şəkil 3*-də göstərildiyi kimidir.

Misal 2.2. Aşağıdakı funksiyanı hesablamaq üçün həll alqoritminin blok-sxemini tərtib etməli:

$$y=\begin{cases} 2{,}5e^{c^2-x}, & c^2-x>0 \quad olarsa;\\ e^{a/b}, & c^2-x=0 \quad olarsa;\\ \sqrt[3]{c^2-x}+ln|c^2-x|, & c^2-x<0 \quad olarsa, \end{cases}$$

burada $c=a\sqrt[3]{a}+3{,}4ln(a-b).$

Göründüyü kimi verilən funksiyanı hesablamaq üçün c^2-x ifadəsinin bir neçə dəfə hesablanması tələb olunur. Əvvəlki misalda olduğu kimi burada da həmin ifadəni bir dəfə hesablayıb hər hansı bir dəyişənə mənimsətmək lazımdır. Bu məsələnin blok-sxemi *şəkil 4*-də göstərilmişdir.

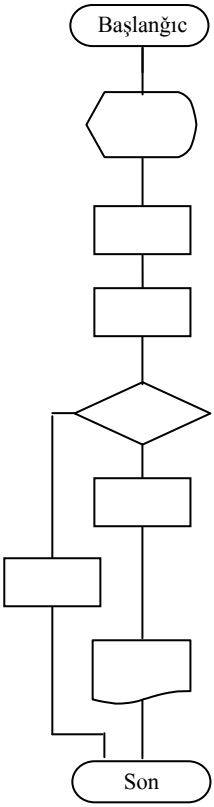
Daha mürəkkəb məsələlərdə bir neçə budaqlanma istiqamətləri ola bilər.

Misal 2.3. Aşağıdakı funksiyanı hesablamaq üçün həll alqoritminin blok-sxemini tərtib etməli:

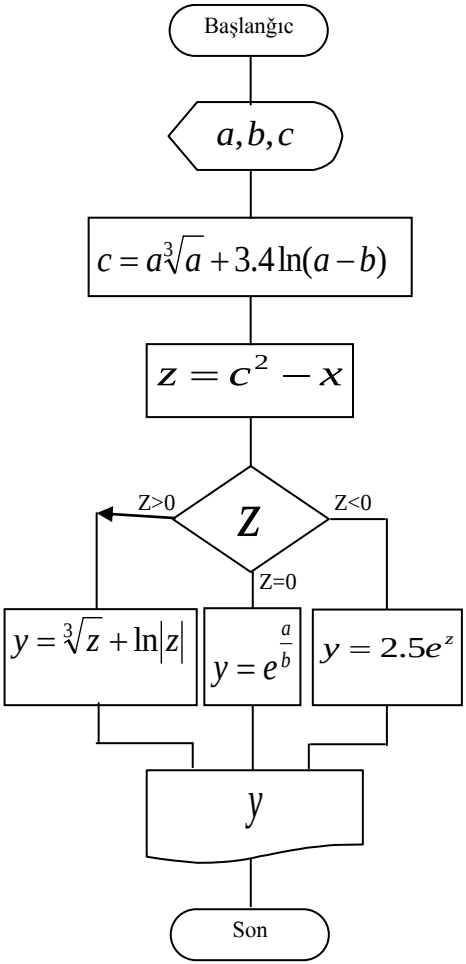
$$z=\begin{cases} y_1+\dfrac{x-x_1}{x_2-x_1}(y_2-y_1), & x_1\le x\le x_2 \ olarsa;\\ y_2+\dfrac{x-x_2}{x_3-x_2}(y_3-y_1), & x_2\le x<x_3 \ olarsa;\\ y_3, & x=x_3 \quad olarsa. \end{cases}$$

Bu məsələnin həlli üçün üç müqayisə bloku lazımdır: əgər $x_1 \le x$ olarsa, onda $x \le x_2$ şərtini yoxlamaq lazımdır. Əgər bu şərt ödənərsə,onda funksiya birinci düstür üzrə hesablanır,

Budaqlanan strukturlu alqoritmlərin təsviri



Şəkil 3



Şəkil 4

əgər $x_1 \le x$ şərti ödənməzsə və ya $x_1 \le x$ şərti ödənərsə, lakin $x \le x_2$ şərti ödənməzsə, onda $x = x_3$ şərtini yoxlamaq lazımdır. Əgər $x = x_3$ şərti ödənərsə, funksiya üçüncü düsturla ($z = y_3$) hesablanır, əks halda yəni $x = x_3$ olmazsa, bu $x_2 \le x < x_3$ şərtinə uyğun gəlir və funksiya ikinci düsturla hesablanacaqdır (*şəkil 5*).

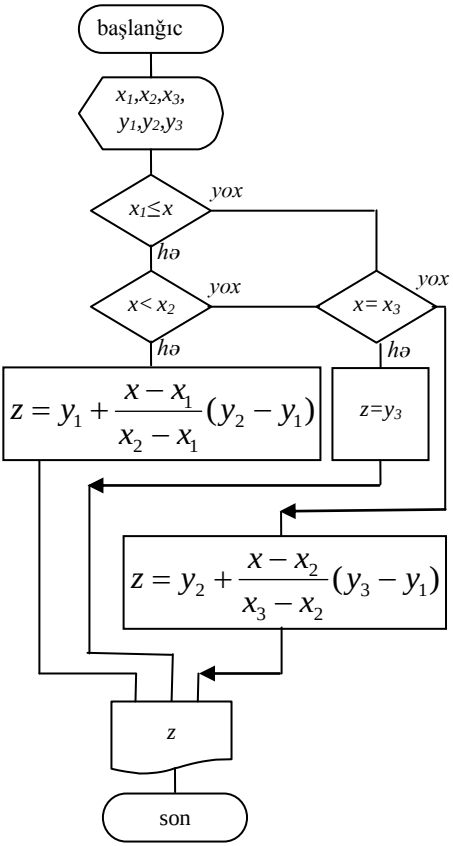
Misal 2.4. Məsələnin həlli prosesində iki *x* və *y* ədədləri hesablanır. Aşağıdakı hesablama prosesinin blok-sxeminin fraqmentini tərtib etmək lazımdır: əgər $k = x + y$ cəmi vahiddən kiçikdirsə, onda *x* və *y* ədədlərindən böyüyünü onların cəminin yarısı ilə əvəz etməli. Əgər $k \ge 1$ olarsa, onda *x* və *y* ədədlərindən kiçiyini *0,5* vahid azaltmalı. Bu məsələnin blok-sxemi *şəkil 6*-da göstərilmişdir.

2.3. DÖVRÜ STRUKTURLU ALQORİTMLƏR

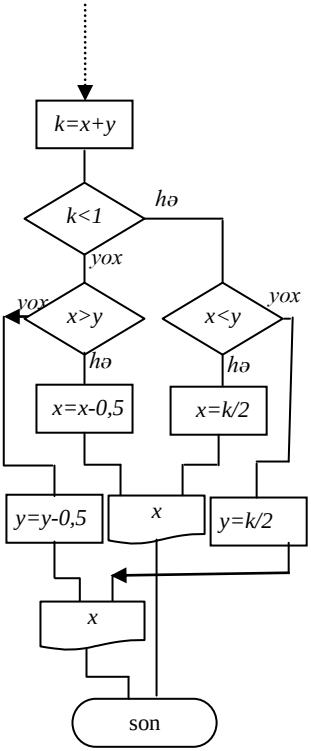
Əgər hesablama proseslərində eyni riyazi asılılıqlar üzrə, ilkin verilənlərin müxtəlif başlanğıc qiymətlərində dəfələrlə hesablamalar aparılarsa, onda belə proseslər dövrü hesablama prosesləri adlanır. Dövrü hesablama proseslərinin iki forması mövcuddur: təkrarən hesablamanın sayı məlum olan dövrlər (*şəkil 7*) və təkrarən hesablamanın sayı məlum olmayan dövrlər (*şəkil 8*). Təkrarən hesablamanın sayı məlum olmayan dövrlərə *iterasiyalı* dövrlər deyilir ki, belə dövrlərdən *"çıxış"* müəyyən şərtin ödənməsi ilə baş verir.

Təkrarən hesablamanın sayı məlum olan dövrlər də öz növbəsində iki növ olur. Bu dövrlərdən birincisinin parametri sadə dəyişən olur və bu parametr ədədi silsilə ilə dəyişir, məsələn, *5*-dən *10*-a qədər *1,2* addımı ilə. Ikinci dövrlər isə *indeksli* dövrlər adlanır. Belə ki, bu dövrlərdə dövrün parametri massivin indeksini göstərir və bu indekslər vasitəsi ilə massivin elementlərinə müraciət olunur. Bu parametrlər

Budaqlanan strukturlu alqoritmlərin təsviri

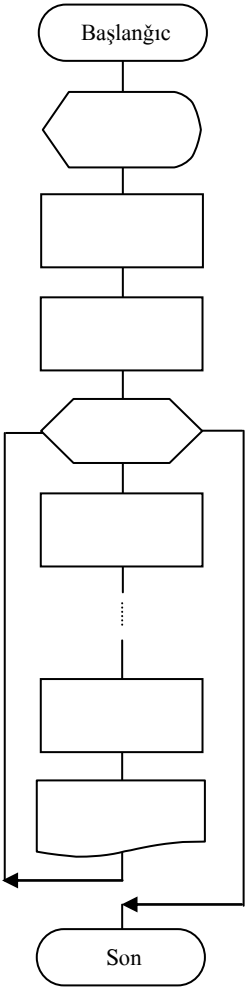


Şəkil 5

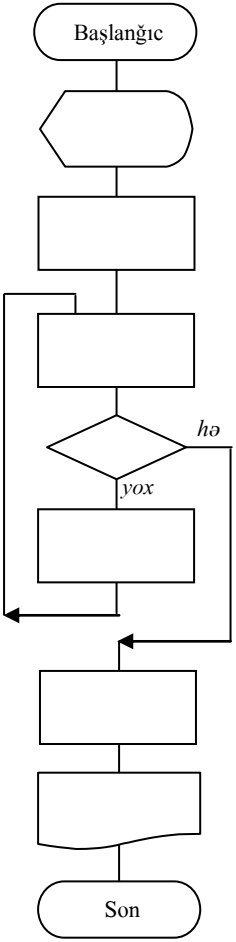


Şəkil 6

Dövrü strukturlu alqoritmlərin təsviri



Şəkil 7



Şəkil 8

tam ədədlər olur, məsələn dövrün hər hansı parametri *2*-dən *20*-yə qədər *2* addımla dəyişir. Lakin bu dövrlərin hər ikisi blok-sxemin qurulması nöqteyi-nəzərindən bir-birindən fərqlənmir və onlar *şəkil 7*-də göstərilən ümumi struktura malik olur.

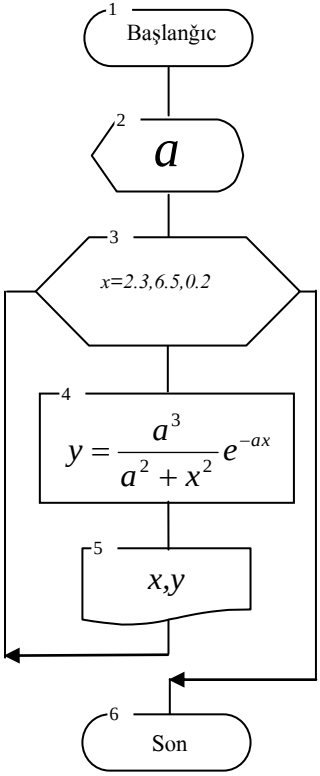
Dövrü proseslərə ən sadə misal olaraq funksiyaların cədvəl şəklində hesablanmasını göstərmək olar.

Misal 2.5. Arqumentin qiymətini *2,3*-dən *6,5*-ə qədər *0,2* addımla dəyişməklə $y = \frac{a^3}{a^2 + x^2} e^{-ax}$ funksiyasını hesablamaq üçün blok-sxem tərtib etməli.

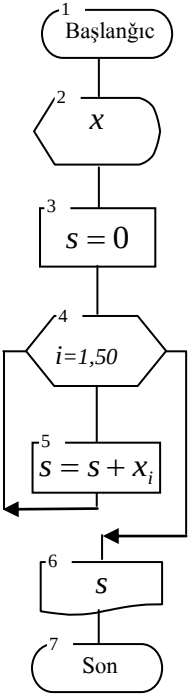
Aydındır ki, funksiyanın birinci qiyməti arqumentin $x=2,3$ qiymətində, sonrakı qiymətləri isə $x=x+\Delta x$ $(\Delta x=0,2)$ qiymətlərində hesablanacaqdır. $x=6,5$ olduqda funksiya sonuncu dəfə hesablanacaq və dövrdən "çıxış" baş verəcəkdir. Dövrün parametrinin başlanğıc, son və addım qiymətləri dövr blokunda göstərilir (*şəkil 9*). Göstərilən blok-sxemdə dövr aşağıdakı qayda ilə yerinə yetirilir: 3-cü blokda dövrün parametri özünün $x=2,3$ başlanğıc qiymətini alır, 4-cü blokda funksiya hesablanır və 5-ci blokda nəticə çap olunur. 3 və 5-ci blokların əhatəsində yerləşən bloklar dövrün oblastını təşkil edirlər. Dövrün parametri x özünün sonuncu $x=6,5$ qiymətini almadığı üçün 3-5-ci bloklar yenidən təkrar olunur. $x=6,5$ olduqda isə 4 və 5-ci bloklar sonuncu dəfə yerinə yetirilir və yalnız bundan sonra 6-cı blok yerinə yetirilir.

Misal 2.6. $X = x_1, x_2, ..., x_{50}$ massivinin elemetlərini cəmləmə alqoritminin blok-sxemini tərtib etməli.

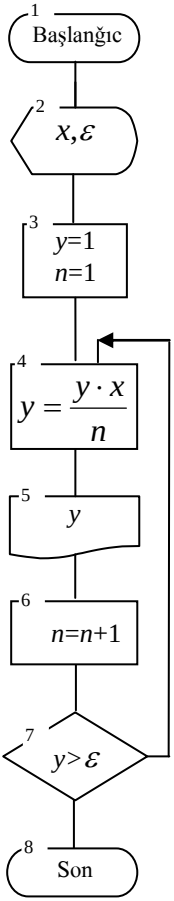
Cəmləmə əməliyyatını yerinə yetirmək üçün hər hansı bir dəyişənə sıfır qiyməti mənimsədib ($s=0$) dövrün daxilində həmin dəyişənin üzərinə cəmlənən dəyişənləri (massivin elementlərini) əlavə etmək lazımdır ($s=s+x_i$). Bu məsələnin blok-sxemi *şəkil 10*-da göstərilmişdir.



Şəkil 9



Şəkil 10



Şəkil 11

Misal 2.7. Sonsuz

$$x, x^2/2!, x^3/3! ,..., x^{n-1}/(n-1)! ,...$$

sırasının hədlərini, hər hansı $x^n/n!$ həddinin verilmiş ε ədədindən kiçik olanadək hesablamaq üçün alqoritmin blok-sxemini tərtib etməli. Baxılan məsələdə dövrlər sayı və ya dövrün parametrinin son qiyməti məlum olmadığından bu məsələnin həlli üçün iterasiyalı dövrdən istifadə etmək lazımdır. Sıranın cari elementini hesablamaq üçün

$$y_n=y_{n-1}\cdot x/n$$

rekurrent düsturundan istifadə etmək lazımdır. Bu düsturda dəyişən arqument kimi həddin nömrəsi götürülür. Ona görə də rekurrent düsturu

$$y=y\cdot x/n$$

şəklində yazmaq lazımdır. Burada birinci həd üçün $y=1$ və $n=1$ olacaqdır. Təsvir olunan alqoritmin blok-sxemi *şəkil 11*-də göstərilmişdir.

2.4. BƏZİ MÜRƏKKƏB MƏSƏLƏLƏRİN BLOK-SXEMLƏRİNİN QURULMASI

Mühəndis məsələlərinin həlli praktikasında yalnız bu və ya digər strukturlu hesablama proseslərindən ibarət məsələlərə çox az hallarda təsadüf edilir. Hətta ən sadə məsələlərin alqoritmlərinin işlənməsində qarışıq strukturlu blok-sxemlərdən istifadə etmək lazim gəlir. Belə blok-sxemlərdə həm xətti, həm budaqlanan, həm də dövrü strukturlu hesablama prosesləri iştirak edir. Məlumdur ki, blok-sxem tərtib olunduqdan sonra birbaşa proqramlaşdırma mərhələsi yerinə yetirilir. Ona görə də blok-sxem tərtib olunarkən elə simvollardan istifadə etməyə çalışmaq lazımdır ki, həmin simvollar proqramda da istifadə oluna bilsin.

Misal 2.8.

$$X = x_1, x_2 ,..., x_{50}$$

massivinin x_{max} ən böyük elementini və onun sıra nömrəsini tapmaq üçün alqoritmin blok-sxemini tərtib etməli.

Alqoritmin mahiyyəti ondan ibarətdir ki, birinci element ən böyük element kimi ($x_{max}=x_1$) qəbul edilməklə yerdə qalan elementlərlə müqayisə edilir. Əgər həmin elementlərdən hər hansı biri x_{max} –dan böyük ($x_i>x_{max}$) olarsa, onda həmin element ən böyük element kimi qəbul edilir, yəni $x_{max}=x_i$. Elə bu anda da həmin elementin nömrəsi $n_{max}=i$ təyin olunur. Bu alqoritmin blok-sxemi *şəkil 12*-də göstərilmişdir.

Misal 2.9. Ən böyük ortaq bölənin tapılması üçün *Evklid* alqoritminin blok-sxemini tərtib edin.

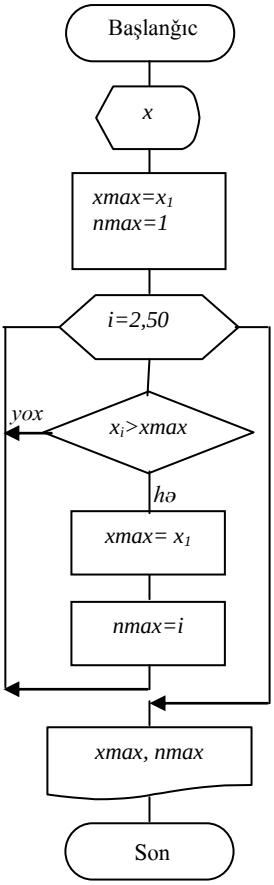
Bu məsələnin mətnlə yazılmış alqoritmi yuxarıda izah edilmiş, blok-sxemi isə *şəkil 13*-də göstərilmişdir.

Misal 2.10. $A=|a_{ij}|$ matrisinin müsbət sətir elementlərinin cəmindən ibarət olan $B=|b_i|$ matrisini yaradın və bu matrisi çap edən alqoritmin blok-sxemini tərtib edin ($i=1, 2 ,...,10; j= 1, 2 ,..., 8$).

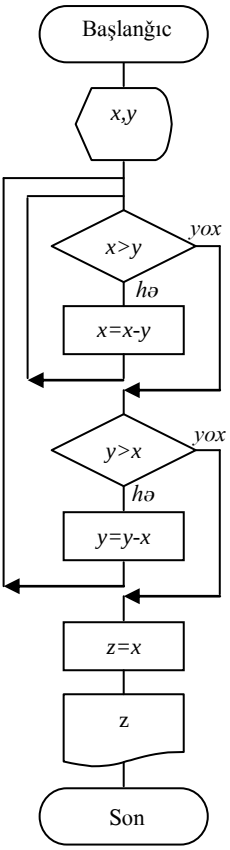
Bu alqoritmdə iki dövr təşkil etmək lazımdır. Bunlardan biri (xarici dövr) matrisin hər bir sətrinə müraciət üçün, digəri (daxili dövr) isə hər bir sətirdə müsbət elementləri tapıb onları cəmləmək üçün nəzərdə tutulmalıdır. Daxili dövrdən əvvəl sətir elementlərinin cəminə uyğun dəyişənə $s=0$ başlanğıc qiyməti mənimsətmək lazımdır. Bu məsələnin blok-sxemi *şəkil 14*-də göstərilmişdir.

Nisbətən mürəkkəb olan daha bir misala baxaq.

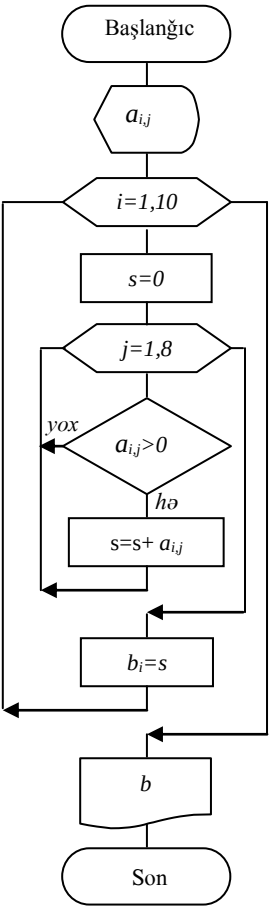
Misal 2.11. Fərz edək ki, hər hansı bir eksperimental qurğuda təcrübənin aparılması nəticəsində elementlərinin sayı uyğun olaraq n və m olan iki pərakəndə $A=|a_i|$ və $B=|b_j|$ massivləri tərtib edilmişdir. Bu massivləri bir C massivində birləşdirməyi , alınmış massivin elementlərini artma sırası ilə



Şəkil 12



Şəkil 13



Şəkil 14

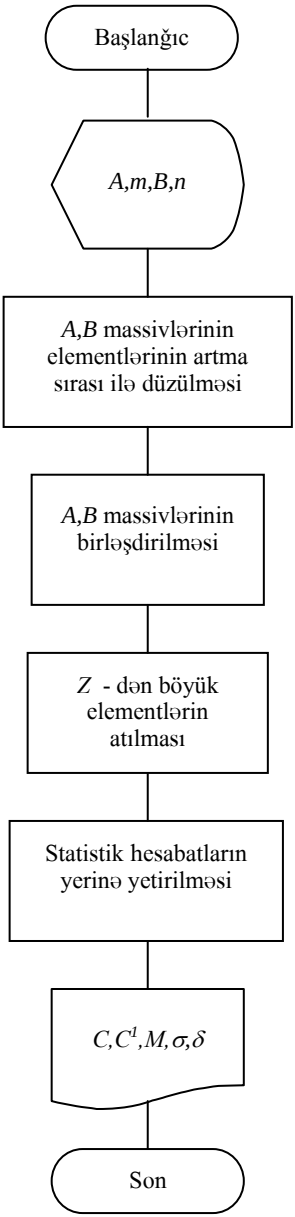
düzməyi, hər hansı bir verilmiş *Z* ədədindən böyük elementləri C massivindən atmağı və alınmış *C'* massivi üçün statistik hesablamaları aparmağı təmin edən alqoritmin blok-sxemini tərtib etməli.

Göründüyü kimi bu məsələ bir neçə müstəqil məsələlərin həllindən ibarətdir. Ona görə də əvvəlcə bütöv məsələnin iriləşmiş blok-sxemini, sonra isə hər bir müstəqil məsələlərin daha müfəssəl blok-sxemlərini tərtib edərək son nəticədə onları vahid bir blok-sxemdə birləşdirmək məqsədəuyğundur.

Bu məsələnin iriləşmiş blok-sxemi *şəkil 15*-də göstərilmişdir.

Ayrı-ayrı məsələlərin blok-sxemlərinin qurulmasına baxaq.

Massivin elementlərinin artma sırası ilə düzülməsi məsələsi bir neçə üsulla yerinə yetirilə bilər. Bu üsulun seçilməsi və qiymətləndirilməsi məsələni həll edənin özündən asılıdır. Fərz edək ki, bu məsələnin həlli üçün yerdəyişmə üsulu seçilmişdir. Yerdəyişmə üsulunun mahiyyəti ondan ibarətdir ki, massivin elementləri ardıcıl olaraq müqayisə olunaraq onların yerləri elə dəyişdirilir ki, nəticədə monoton artan (və ya azalan) ardıcıllıq alınsın. Artan ardıcıllıq almaq üçün əvvəlcə birinci iki element müqayisə olunur. Əgər $x_1, x_2,...,x_n$ ardıcıllığında $x_2<x_1$ olarsa, onda x_1 elementi bir yer sağa sürüşdürülür, onun yerinə isə x_2 yazılır. Əgər $x_2>x_1$ olarsa, onda bu elementlərin yeri dəyişdirilmir, x_2 və x_3 elementləri müqayisə olunur və s. Ümumi halda elementlərin "yerdəyişmə" prosesini belə təsvir etmək olar. Əgər $x_i<x_j$ ($i=i-1,..., 1$) olarsa, onda j-cu element sağa sürüşdürülür və x_i elementi x_{j-1} elementi ilə müqayisə olunur. Əgər $x_i>x_j$ olarsa, onda j-cu element öz yerində qalır, i-ci element isə $j+1$-ci yerə sürüşdürülür. Elementlərin dəfələrlə müqayisə edilməsi və onların yerlərinin dəyişdirilməsi lazım olan nəticəyə gətirir. Yerdəyişmə üsulu ilə massivin element-


Başlanğıc
A,m,B,n
A,B massivlərinin elementlərinin artma sırası ilə düzülməsi
A,B massivlərinin birləşdirilməsi
Z - dən böyük elementlərin atılması
Statistik hesabatların yerinə yetirilməsi
C,C1,M,σ,δ
Son

Şəkil 15

lərinin artma sırası ilə düzülməsi alqoritminin blok-sxemi *şəkil 16*-da göstərilmişdir.

Massivlərin birləşdirilməsi məsələsinin alqoritmi iki

$$a_1<a_2<...<a_i<...<a_{m-1}<a_m$$

$$b_1<b_2<...<b_j<...<b_{n-1}<b_n$$

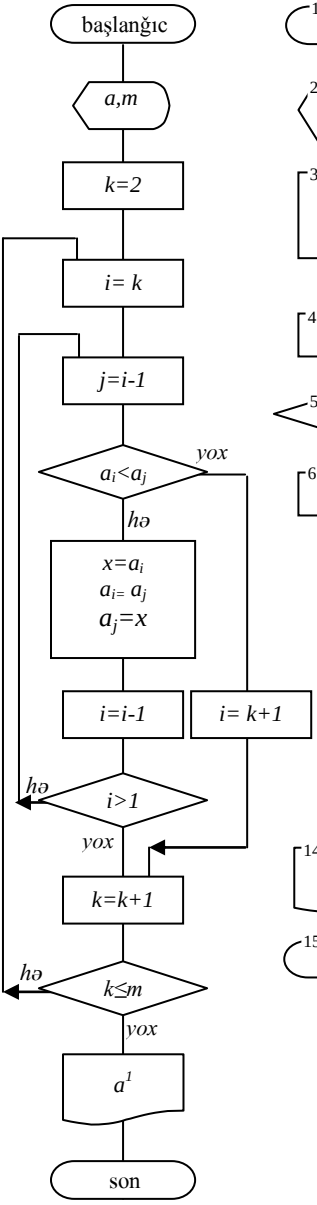
massivlərinin elementlərinin bir *C* massivində artma sırası ilə birləşdirilməsi, yəni

$$c_1<c_2<...<c_{i+j}<...<c_{m+n-1}<c_{m+n}$$

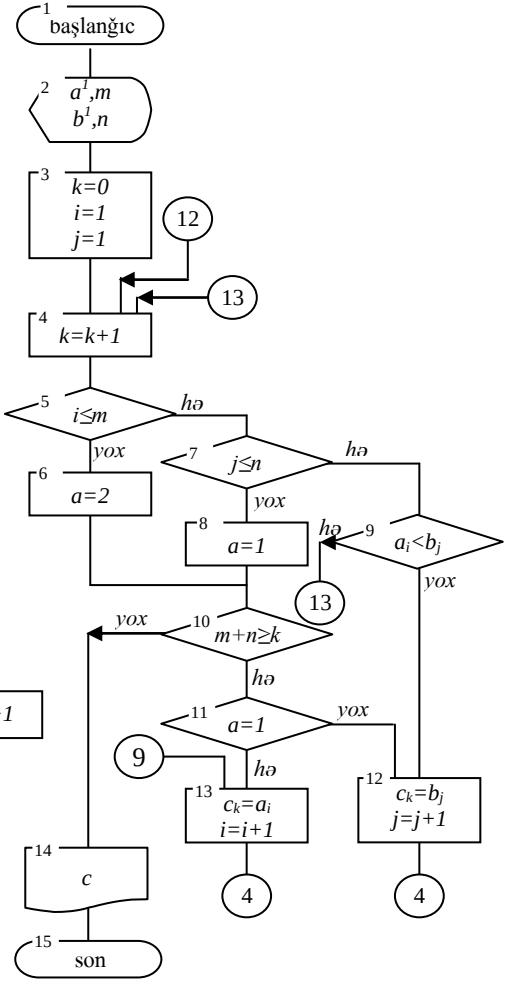
elementlərinin tapılması məsələsinin alqoritmi bütün elementlərin seçilməsi, yoxlanılması ($i \le m$, $j \le n$), müqayisə edilməsi ($a_i<b_j$) və massivlərin iki elementindən ən kiçiyinin ($c_k=a_i$ və ya $c_k=b_j$) yeni *C* massivinə göndərilməsi kimi dövrü proseslərdən ibarətdir. Əgər $a_1<b_1$ olarsa, onda a_1, əks halda b_1 yeni *C* massivinin birinci elementi olur. Növbəti dəfə a_2 ilə b_1 (və ya a_1 ilə b_2) müqayisə olunur və onlardan ən kiçiyi *C* massivinə yazılır və s. Bu alqoritmdə əsas çətinlik indekslərin düzgün yazılmasındadır. Burada *A, B* və *C* massivlərinə uyğun olaraq üç indeks - *i, j* və *k* götürmək lazımdır. Massivlərin birləşdirilməsi məsələsinin blok-sxemi *şəkil 17*-də göstərilmişdir.

Massivin bəzi elementlərinin atılması. Bu məsələnin mahiyyəti ondan ibarətdir ki, dövr daxilində *C* massivinin hər bir elementi *Z* dəyişəni ilə müqayisə edilir. Əgər hər hansı bir element üçün $C_i<Z$ olarsa, onda həmin elementdən başlayaraq massivin bütün elementləri atılır (çünki, massivin elementləri artma sırası ilə düzülmüşdür). Yeni massivi *C'* ilə işarə edək. Bu massivin elementləri üzərində statistik hesablamalar aparmaq lazım gəldiyindən *C'* massivinin elementlərini saymaq üçün dövr daxilində sayğac (*p*) nəzərdə tutmaq lazımdır.

Statistik hesablamaları sadələşdirərək ədədi orta qiyməti



Şəkil 16



Şəkil 17

$$M = \frac{1}{P}\sum_{i=1}^{P} C_i^{`},$$

orta kvadratik meyli

$$\sigma = \sqrt{\frac{1}{P-1}\sum_{i=1}^{P}(C_i - M)^2}$$

və orta kvadratik səhvi

$$\delta = \sigma / \sqrt{P}$$

düsturları ilə hesablamaqla kifayətlənək.

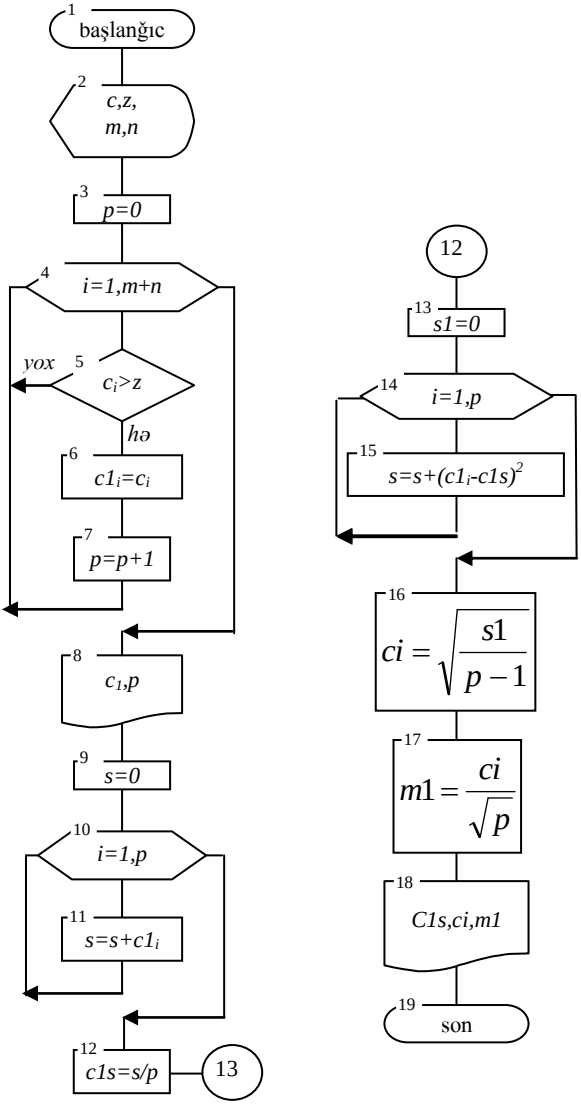
Şəkil 18-də massivin bir neçə elementinin atılması və statistik hesablamaların birgə alqoritminin blok-sxemi göstərilmişdir. Beləliklə, baxılan məsələnin ayrı-ayrı məsələlərinin müstəqil blok-sxemlərini tərtib etdik. Bu blok-sxemlərin birləşdirilməsi heç bir çətinlik törətmir və bu məsələnin həllini tələbələrin özlərinə həvalə edirik.

3.ALQORİTMLƏR VƏ PROQRAMLAŞDIRMA

Proqramlaşdırma mərhələsi məsələlərin kompüterdə həll edilməsi üçün hazırlanması prosesinin yekun mərhələsidir.Bu mərhələdə alqoritm hər hansı bir alqoritmik dildə yazılır.

Alqoritmin alqoritmik dildə yazılması *proqramlaşdırma*, yazılışın özü isə *proqram* adlanır. Məsələnin blok-sxemə uyğun olaraq proqramlaşdırılması çox asanlıqla yerinə yetirilir, belə ki, adətən, hər bir bloka alqoritmik dildə bir operator uyğun gəlir. Bunu *cədvəl 2*-də göstərilən blok-sxem fraqmentlərindən və onların *Basic* və *Pascal* dillərində yazılmış proqramlarından əyani görmək mümkündür.

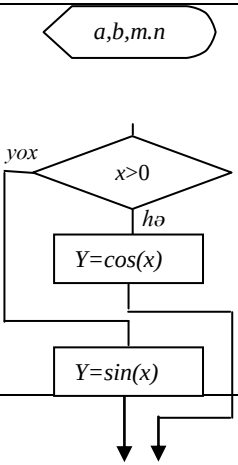
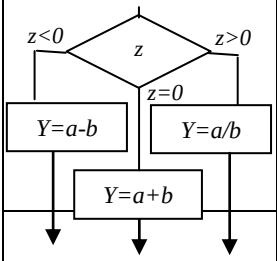
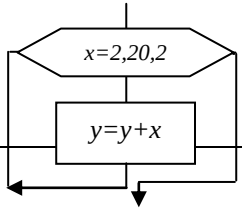
Növbəti misallarda hesablama riyaziyyatının bir neçə üsullarının uyğun blok - sxemlər üzrə proqramlarının tərtib


1
başlanğıc
2
c,z,
m,n
3
p=0
4
i=1,m+n
yox
5
ci>z
hə
6
c1i=ci
7
p=p+1
8
c1,p
9
s=0
10
i=1,p
11
s=s+c1i
12
c1s=s/p
13
12
13
s1=0
14
i=1,p
15
s=s+(c1i-c1s)2
16
ci = √(s1/(p−1))
17
m1 = ci/√p
18
C1s,ci,m1
19
son

Şəkil 18

Cədvəl 2

Blok-sxem fraqmentləri və onların proqram ekvivalentləri

Blok-sxem fraqmenti	Basic dilində proqramın yazılışı	Pascal dilində proqramın yazılışı
	20 *input a,b,m,n*	*read(a,b,m,n);*
	… 50 *if x>0 then* *y=cos(x)* *else* *y=sin(x)* …	… *if x>0 then* *y:=cos(x)* *else* *y:=sin(x);* …
	… 50 *if z=0 then* *y=a+b* 60 *if z>0 then* *y=a/b* *70 if z<0 then* *y=a-b* …	*if z=0 then* *y:=a+b;* *if z>0 then* *y:=a/b;* *if z<0 then* *y:=a-b;* …
	… 80 *for x=2 to 20* *step 2* 90 *y=y+x* 100 *next x* …	… *x:=2;* *repeat* *y:=y+x;* *x:=x+2;* *until x>20;* …

olunması nümayiş etdirilmişdir.

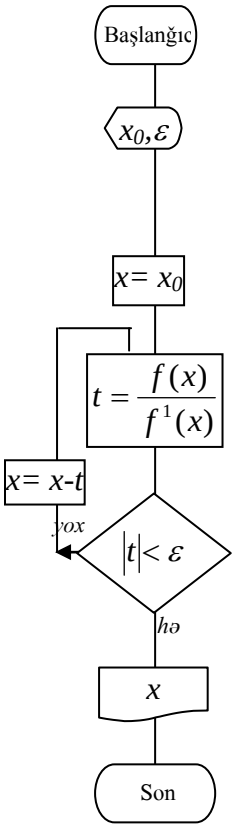
Şəkil 19-da cəbri və transsendent tənliklərin *Nyuton* (toxunanlar) üsulu ilə həllinin blok-sxemi və proqramları göstərilmişdir.

Şəkil 20-də yerdəyişmə üsulu ilə massivin elementlərinin artma sırası ilə düzülməsi məsələsinin blok-sxemi və *Pascal* dilində proqramı göstərilmişdir.

Şəkil 21-də isə *Simpson* üsulu ilə müəyyən inteqralın təqribi hesablanmasının blok-sxemi və *Pascal* alqoritmik dilində proqramı nümayiş etdirilmişdir.

Qeyd. Pascal dilində yazılmış proqramları istifadə etdikdə proqramda istifadə olunan dəyişənlərin təsvirini proqrama əlavə etmək lazımdır.

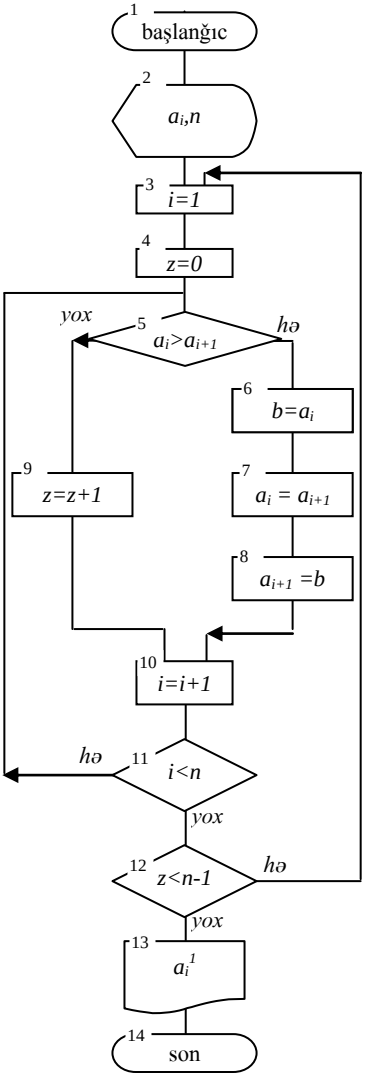
Nyuton üsulunun blok-sxemi və *Basic* və *Pascal* dillərinin hər bloka uyğun operatorları



	Basic proqramı	*Pascal proqramı*
	10 *rem Nyuton üsulu*	*begin*
	20 input x0,e	*read(x0,e);*
	30 x=x0	*x:=x0;*
	40 t =f(x)/f¹(x);	*1: t:=f(x)/f¹(x);*
	50 if abs(t)<=e go to 80	*if abs(t)<=e then goto 2;*
	60 x=x-t	*x:=x-t;*
	70 goto 40	*goto 1;*
	80 print x	*2: writeln(x);*
	90 end	*end.*

Şəkil 19

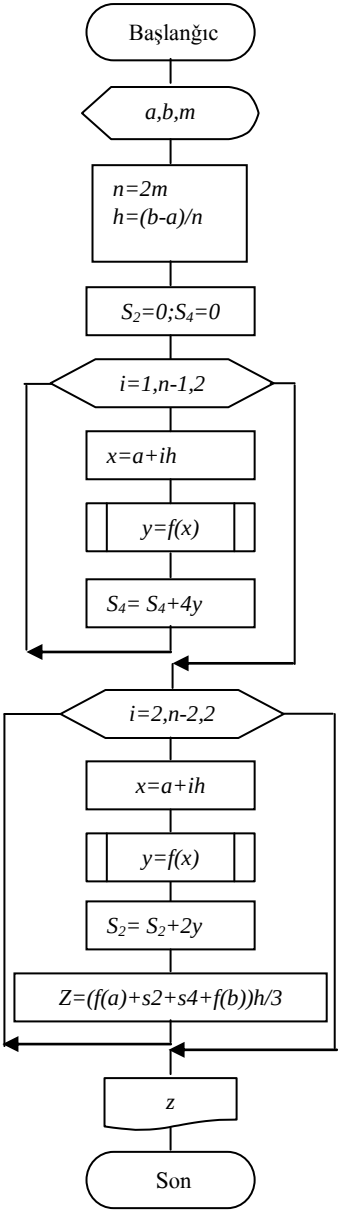
Yerdəyişmə üsulunun blok-sxemi və *Pascal* proqramı




```
begin

read (n);
for i:=1 to n do readln(a[i]);
1:
   i:=1;
   z:=0;
2:
   if a[i]>a[i+1]
    then
        begin
           b:=a[i];
           a[i]:=a[i+1];
           a[i+1]:=b;
        end
   else
      z:=z+1;


      i:=i+1;


      if i<n  then go to 2;


      if z<n-1  then go to 1

      for i:=1 to n do
      writeln(a[i]);

end;
```

Şəkil 20

Simpson üsulu




```
begin
    read(a,b,m);
    n:=2*m;
    h:=(b-a)/n;
    s2:=0; s4:=0;
    i:=1;
    while i<=n-1 do
        begin
          x:=a+i*h;
          y:=f(x);
          s4:=s4+4*y;
          i:=i+2;
        end;
    i:=2;
    while i<=n-2 do
        begin
          x:=a+i*h;
          y:=f(x);
          s2:=s2+2*y;
          i:=i+2;
        end;
    z:=(f(a)+f(b)+s2+s4)*y/3;
    writeln(z:8:3);
end.
```

Şəkil 21

ƏDƏBİYYAT

1.Ə.H.Nağıyev, Z.T.Məhərrəmov, A.H.Hüseynov, Ş.R.Rəhimov. FORTRAN alqoritmik dilində laboratoriya işləri (Metodik göstərişlər). Sumqayıt, 1992. –78 s.

2.Ə.A.Vəliyev, Z.T.Məhərrəmov, Y.Ə.Əbilov. Delphi: nəzəriyyə və təcrübə. Bakı, 2004. – 336 s.

3.H.V.Meladze, Ə.A.Vəliyev, V.Ə.Sadıxov, N.M.Sxirtladze, P.A.Sereteli. Əyləncəli informatika və modelləşdirmə. Bakı, 2005. -248 s.

MÜNDƏRICAT

1.Kompüterdə məsələlərin həll mərhələləri..... 3

2. Məsələnin həll alqoritmlərinin işlənməsi 5

2.1.Xətti strukturlu alqoritmlər..... 6

2.2. Budaqlanan strukturlu alqoritmlər..... 9

2.3. Dövrü strukturlu alqoritmlər..... 13

2.4. Bəzi mürəkkəb məsələlərin blok-

sxemlərinin qurulmasi 15

3. Alqoritmlər və proqramlaşdirma 22

ƏDƏBİYYAT