Proqramlaşdırma methodları və prosesləri

1.1 Xətti və qeyri-xətti proqramlaşdırma

Xətti proqramlaşdırma, hər hansı bir şərt yaxud dövr olmadan, düz-xəttin tənliyini xatırladan bir şəkildə həll olan məsələlərin həllində istifadə olunur. Burada sadəcə verilənlər hazır olur , onlar daxil edikdən sonra qiymətləri yazılıb həll edilir. A=5, B=4, C=A+B cavab: 9 və ya A= “alma”, B= “meyvə” C=A+ “ “ + B cavab: “alma meyvə” və s misallar göstərmək olar.

Qeyri-xətti proqramlaşdırma isə iki yerə bölünür. Budaqlanan və Dövrü proqramlaşdırma. Budaqlanan: burada məhdudiyyət şərtləri mövcud olur. Həmin şərtlər əsasında nəticəyə çatılır. Misal üçün: A ədədi 0 dan böyükdürsə, A – nı 2 yə vur, kiçikirsə, 2-yə böl, burada iki şərt daxilində iki əməliyyat var. artıq bir əməliyyatdan iki budaq meydana gəldi. Və bu budaqların(şərtlərin) sayı ikidən çox ola bilir. Eyni məsələyə eyni anda istədiyimiz qədər şərt qoya bilirik. Dövrü: Dövrü proqramlaşdırmada isə proses verilmiş şərtə görə daima dövr edir. Missal üçün: hər bir işçinin maşı verilənə qədər bugalter(şirkətin büdcə cavabdehi) “maaş hesablama” cihazını dəfələrlə çalışdırsın.

1.2 Proqramın yazılma mərhələləri

Proqramlaşdırma – proqram yaratmaqla bağlı nəzəri və praktiki yaradıcılıq sahəsidir.
Kompüterin proqram təminatı iki yerə bölünür: sistem və tətbiqi.
Kompüterdə məsələnin həlli aşağıdakı mərhələlərdən ibarətdir:

1. Məsələnin qoyuluşu: məsələ haqqında informasiyanın toplanması; məsələnin şərtinin formalaşdırılması; son məqsədin müəyyən olunması; nəticələrin formasının müəyyən olunması; verilənlərin təsviri (onların tipləri, dəyişmə diapazonu, strukturu və s.).

2. Məsələnin, modelin analizi və tədqiqi: movcud analoqların analizi; texniki və proqram vasitələrinin analizi; riyazi modelin hazırlanması (işlənilməsi); verilənlərin strukturunun hazırlanması (işlənilməsi).

3. Alqoritmin hazırlanması (işlənilməsi): alqoritmin layihələndirilməsi üsulunun seşilməsi; alqoritmin yazılış formasının seçilməsi (blok-sxem, psevdokod və s. testin və testləşdirmə üsulunun seçilməsi; alqoritmin layihələndirilməsi.

4. Proqramlaşdırma: proqramlaşdırma dilinin seçilməsi; verilənlərin təşkili qaydalarının dəqiqləşdirilməsi; seçilmiş proqramlaşdırma dilində alqoritmin yazılması.

5. Testdən keçirmə və sazlama (otladka): sintaksisin yoxlanılması; məntiqi quruluşun və semantikanın yoxlanılması; test hesablamalar və testin nəticələrinin analizi; proqramın təkmilləşdirilməsi.

6. Məsəslənin həllinin nəticələrinin analizi və lazim gələrsə riyazi modelin dəqiqləşdirilməsi (2-5 mərhələlərinin təkrar ediliməsi).

7. Proqramın müşayiəti:
konkret məsəslələrin həlli üçün proqrama əlavələrin edilməsi; həll olunan məsələ, riyazi model, alqoiritm, proqram, testlərin toplanması və istifadəçi üçün sənədləşmənin tərtibi.

1.3 Alqoritm

Alqoritm — verilmiş məsələni həll etmək üçün ilkin verilənlərlə icra olunan hesabi və hər hansı məsələnin həlli üçün məntiqi əməliyyatların sonlu sayda ardıcıllığıdır .

Latınca qayda-qanun deməkdir. Alqoritm 783 - 850-ci illərdə Xarəzmdə (indiki Özbəkistanda şəhər) yaşamış IX əsrin məşhur fars riyaziyyatçısı Məhəmməd İbn Musa əl-Xarəzminin (yəni Xarəzmli Musa oğlu Məhəmmədin) adının latın hərflərilə olan "alqoritmi" yazılışıyla bağlıdır. Əl-Xarəzminin yazdığı traktatın XII əsrdə latın dilinə tərcümə olunması sayəsində avropalılar mövqeli say sistemi ilə tanış olmuş, onluq say sistemini və onun hesab qaydalarını alqoritm adlandırmışlar. Ümumiyyətlə, alqoritm-verilmiş məsələnin həlli üçün lazım olan əməliyyatları müəyyən edən və onların hansı ardıcıllıqla yerinə yetirilməsini göstərən formal yazılışdır. Hesablama maşınlarının əsas fərqləndirici xüsusiyyətlərindən biri də onun proqramla idarə olunmasıdır. Yəni, istər sadə, istərsə də mürəkkəb məsələni maşının həll etməsi üçün proqram tərtib edilməlidir.

1.4 Alqoritmin xasisələri və təsvir üsulları

Məsələnin maşında həlli üçün tərtib edilən alqoritm bir çox şərtləri ödəməlidir. Bu şərtlərə alqoritmin xassələri deyilir. Həmin xassələr aşağıdakılardır:

1. *Diskretlilik xassəsi.* Hər bir alqoritm məsələnin həll prosesini sadə addımların yerinə yetirilməsi ardıcıllığı şəklində ifadə edir və hər bir addımın yerinə yetirilməsi üçün sonlu zaman fasiləsi tələb olunur, yəni başlanğıc verilənlərlə icra olunan hesabi və məntiqi əməliyyatların yerinə yetirilməsi və nəticənin alınması zamana görə diskret yerinə yetirilir.

2. *Müəyyənlik xassəsi.* Hər bir alqoritm dəqiq, birqiymətli olmalıdır. Bu xassəyə əsasən alqoritm yerinə yetirildikdə istifadəçinin və onun istifadə etdiyi kompüterdən asılı olmayaraq eyni nəticə əldə edilməlidir.

3. *Kütləvilik xassəsi.* Müəyyən sinif məsələnin həlli üçün qurulmuş alqoritm bu sinfə aid olan yalnız başlanğıc qiymətləri ilə fərqlənən bütün məsələlərin həllini təmin etməlidir. Məsələn, $ax^2 + bx + c = 0$ kvadrat tənliyi üçün qurulmuş alqoritm a, b, c – nin ixtiyari qiymətləri üçün məsələni həll edir.

4. *Nəticəlilik və sonluluq xassəsi.* Alqoritm sonlu sayda addımdan sonra başa çatmalı və verilmiş məsələnin həlli tapılmalıdır.

Təsvir üsulları: 1.Mətn şəkildə **2.Blok-sxem** 3. Cədvəl **4. Proqram**

1.5 Kompilyasiya və İnterpretasiya

Translyasiyanın iki qaydası var: interpretasiya və kompilyasiya. İnterpretasiya – şifahi tərcüməyə oxşayır. Giriş proqramının hər bir təlimatı tərcümə olununur və yerinə yetirilir. Bu qaydada təkrar təlimatlar hər dəfə kodlaşdırılır. Kompilyasiya isə yazılı tərcüməyə bənzəyir. Proqram yerinə yetirilməzdən qabaq proqramın bütün tərcüməsi yığılır.

İnterpretasiya böyük çevikliyə malik olmaqla asan realizə olunur. Kompilyasiya isə daha effektif proqram yaradır.

Proqramçı isə proqramlaşdırma dillərini bilməklə, qarşıya qoyulan məsələnin kompüterdə həllini həyata keçirmək üçün proqram yazır və onu kompüterdə yerinə yetirir.

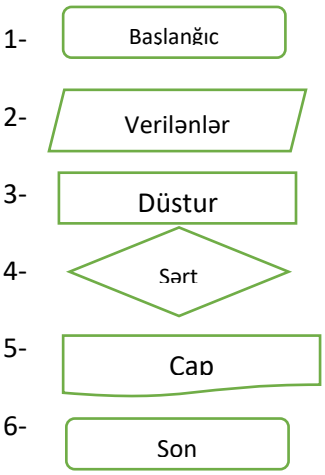
Proqramlaşmanın bütün dilləri verilənlərin aşağıda göstərilən tipləri ilə işləməyə imkan verilir:

Tam ədədlər; Məntiqi ədədlər; Həqiqi ədədlər; Simvollar; Mətn tipli ədədlər; Birtipli verilənlər cədvəli; Fayllar.

Kompüterin alqoritmi başa düşməsi üçün proqramlaşdırma dillərindən istifadə edilir. Məsələ həll edərkən əvvəlcə yerinə yetiriləcək əməliyyatların alqoritmi tərtib edilir, daha sonra bu əməliyyatlar hər-hansı alqoritm (proqramlaşdırma) dilində əmrlər şəklində yazılır. Tərtib olunmuş proqram xüsusi əlavələr (translyator proqramlar) vasitəsilə yerinə yetirilir və ya maşın koduna çevrilir.

1.6 Block-sxem

1. Alqoritmin başlangıcı və sonu (1) və (6)fiqur icərisində yazılır. –
2. İlkin verilənlərin daxil edilməsi paraleloqram fiquru ilə təsvirolunur və onun içərisində qiymətləri daxil edilməli olan dəyişənlərin adı yazılır (2)
3. Hesanlama blokunun daxilində yerinə yetirilməli olan əməliyyatlar yazılır(3)
4. Şərtin yoxlanma əmri romb şəklində təsvir olunur. Ödəniləcək şərt onun içərisində yazılır . şərtin ödənilib-ödənilməməsindən asılı olaraq hesablama prosesi iki mümkün istiqamətdən biri üzrə davam etdirilir(4)
5. İçərisində qiyməti çap edilməli olan dəyişənlərin adı yazılır.(5)



Məsələnin Qoyuluşu: İlk öncə məsələ oxunmalı, sonra riyazi üsulmu, məntiqi yolmu o seçilməli. Ardından məsələnin həlli beyində qurulmalı, hansı verilənlərdən, hansı düsturlardan istifadə ediləcəyi düşünülməli. Bu proseslər alqoritmin qurulmasıdır. Alqoritm proseslər ardıcıllığıdır. 4 üsuldan biri də Block – sxemdir. Və beləcə yuxarıda deyilən ardıcıllıqların hər biri block – sxemədə aid edilir.

1.7 Pythona giriş (xətti proqramlama)

Verilənlər: 3 tipə müraciət edirik: 1-İnteger(int)(bura tam olan bütün rəqəm və ədələr aiddir), 2-String(str)(hərflər, simvollar və bütün rəqəm birləşmələri), 3-float(rəqəmlər və onluq kəsrlər). İnteger və Floatları toplamaq, çıxmaq, vurmaq və bölmək adi riyazi qaydada olur. Verilənlər hər hansı hərf birləşməsi ilə ifadə olunur. və bərabərliyin solunda yazılır. Bəzən verilən adı ilə stringlər qarışdırılır. Amma unutmayaq, stringlər bərabərliyin sağında "" içərisində olur. abc= "abc" soldakı abc verilənin adıdır. Tanıtma sözüdür. Amma sağdakı isə dəyərdir. Verilənin dəyəri. Sağda olan ya string, ya integer ya da float tipində olur.

Riyazi əməliyyatlar: (+) toplama əməliyyatı, (-) fərq əməliyyatı, (*) vurma əməliyyatı, (**) qüvvətə yüksəltmə, (/) qalıqlı bölmə əməliyyatı. (//) qalıqsız bölmə əməliyyatı. (%) qalığı tapmaq əməliyyatı

İnput: proqram server ilə istifadəçi arasında əlaqə yaradan bağdır. Server iki veriləni oxuyur. Bunlardan bir qrupu proqramist daxil edir, digər qrupu isə istifadəçi. İstifadəçi proqram "run" edildikdən sonra onun üçün ayrılan boş sahəyə ondan soruşulanı daxil edir. Boş sahəni isə proqramist input() funksiyası ilə yaradır. İstifadəçi əgər string daxil etməlidirsə, sadəcə ad=input("adı daxil et:") yazılır. Yox əgər istifadəçi integer daxil edəcəksə, reqem=int(input("ededi daxil et:")) yazılır, yox əgər floatdırsa, bu zaman da reqem=float(input("onluq kesri daxil et:")) yazılır. ***(qeyd: verilənlər hökmən ingilis əlifbasi ilə yazılır. Amma stringlər və inputun içərisində yazılanlar ixtiyari əlifbada yazıla bilər. Çünkü proqram verilənlərə kod kimi baxır amma dırnaqda olanları isə heç görmür, sadəcə kopyalayıb, yapışdırır)***.

Print funksiyası: verilənlər üzərində əməllər aparıldıqdan sonra nəticə digər bir verilənə mənimsədilir. A=9, B=8, C=A+B – mənimsədilmə olduqdan sonra C elementi print funksiyası ilə ekrana çağırılır. print(C).

Məsələ: İstifadəçi maşının sürətini qeyd edib. Və maşın 5 saniyə yol gedir. Yolu tap. *Həlli:* İstifadəçi qeyd edir deyirsə, demək ki, biz input ilə yer ayırmalıyıq. Sürət integer olacağına görə suret=int(input("sürət:")), zaman=5 qeyd edirik. yol=suret*zaman. print(yol).

Stringlərin toplanması: Stringlərin toplanması onların birləşdirilməsidir. verilen1= "alma", verilen2= "armud", birleşme = verilen1+verilen2. print(birleşme): cavab: almaarmud. Beləcə iki sözü birləşdirir. Stringlərin integerlərdən fərqli olaraq, çıxma və bölmə əməliyyatları yoxdur. Ümumiyyətlə iki veriləni toplamaq üçün, onların aldıqları dəyərlər yaxud input ilə alacaqları dəyərlər eyni tipdə olmalıdır. Bir integer digəri string yaxud biri float digəri string olarsa error verir. Amma integer və float tiplərini problemsiz, toplayıb-çıxır. A=6, B= "alma", C=A+B, print(C) cavab: error. Bunun üçün hər ikisi ya string ya da integer olmalıdır. Alma sözünü integerə çevirə bilmirik. Amma 6 rəqəmini stringə çevirə bilərik. Yuxarıda dediyimiz kimi stringlər "" daxilinə yazılır. Və proqram daxildə olanları anlamır. Sadəcə kopyalayıb, ekrana çıxarır. Demək ki, biz 6 rəqəmini "" arasına yazsaq, bu zaman proqram onun rəqəm olduğunu bilməyəcək. Kopyalayıb yapışdıracağına görə onu sadəcə string kimi anlayacaq. 6 /= "6" – 6 rəqəmə "6" ya bərabər deyil. A= "6", B= "alma" C= A+B print(C) cavab: 6alma – sadece iki string kimi alqılayıb ekrana onları birləşdirərək çıxarır. **str():** bu funksiya integer kimi tanıdılmış veriləni stringə çevirmək üçün istifadə edilir. A=5 A veriləni integer tipdədi. Tipini string etmək üçün str(A) yazmağımız kifayətdir. Çünki str() = "". **int():** bu funksiyası əməliyyatı geri qaytarır. Göründüyü kimi artıq A stringdir. Yenidən onu integer etmək üçün int(A) yazmaq kifayətdir. Nümunə: eded1=5, ad= "kitab", sira=eded1+ad. Tipləri ayrı olduğu üçün error verəcək. Bunun üçün hər ikisini string etməliyik. eded2=str(eded1), sira = eded2+ad. print(sıra). Artıq eded2 və ad hər ikisi stringdir. Buna görə də error verməyəcək.

1.8 şərt operatorları (budaqlanan alqoritm)

Əgər məsələnin həllində hər hansısa, bir şərt əsas rol oynayırsa, bu zaman python kodlarına budaqlanan alqoritmi tətbiq edirik. Burada if əmrindən istifadə edilir. Məsələ: verilmiş ədəd 0 dan böyükdürsə, ekrana "Müsbət" sözünü çıxar. eded=5, if(eded>0): , print("Müsbət"). İf əmrinin içərisində şərti yoxlamaq üçün, bərabərdirsə(==), bərabər deyilsə(!=), böyükdürsə(>), kiçikdirsə(<), böyük bərabərdirsə(>=) və kiçik bərabərdirsə(<=) işarələrindən istifadə edilir. Burada bir şərtin daxilində bir neçə operatordan istifadə edilir. **Operatorlar: and, or.** And - əgər hər iki şərt ödənərsə, proses doğru çalışır. Or – burada isə prosesin doğru çalışması üçün iki şərtdən birinin ödənməsi kifayətdir. *Məsələ:* verilmiş ədəd özü müsbət, həm də kvadratı 15-dən böyükdürsə ekrana "yes", kiçikdirsə, "No" çıxarsın. *Həlli:*

eded=3, if (eded>0 and eded**2 > 0): cavab "No" olaraq, ekrana çıxacaq. Çünki 3, 0-dan böyükdür. Amma kvadratı 15-dən böyük deyil. Verilmiş iki şərtdən biri ödənir. Amma digəri deyil. Buna görə ekrana "No" çıxır. *Məsələ:* ya eded 0-dan böyük ya da kvadratı 5-dən böyükdürsə, ekrana "yes", əks halda isə "no" çıxarsın. *Həlli:* "ya da" dediyi üçün iki şərtdən birinin ödənməsi kifayətdir. Ona görə "or" operatorundan istifadə edirik. eded=2, if(eded>0 or eded**2>5): , ekrana "yes" çıxacaq. Çünki iki şərtdən birinin ödənməsi bəs edir. Ededin kvardratı 5 dən böyük deyil, amma eded müsbətdir. "yes" olur.