**Mühazirə 1**

**“İnformatika”** termini (frans. Informatique)*information* (informasiya) və*automatique*

(avtomatika) fransız sözlərindən törəyir və əsl mənada**“informasiya avtomatikası”**

deməkdir.

Bu terminin hərfi mənası**“kompyuter elmi”** demək olan ingilis dilli**“Сomputer**

**science”** variantı da geniş yayılmışdır.

İnformatika – kompyuter texnikasından istifadə edilməsinə əsaslanan, informasiyanın

strukturunu və ümumi xassələrini, həmçinin insan fəaliyyətinin müxtəlif sahələrində onun

yaranmasının, saxlanmasının, axtarışının, çevrilməsinin, ötürülməsinin və təbiqinin

qanunauyğunluqlarını və üsullarını öyrənən fəndir.

1978-ci ildə beynəlxalq elmi konqress*“informatika”* anlayışı altında insan həyatının

bütün sahələrinə kompyuter texnikasının kütləvi tətbiq olunduğu sahələri – kompyuterlər

və onların proqram təminatı, həmçinin kompyuterləşdirmənin təşkilati, kommersiya, inzibati

və sosial-siyasi cəhətləri daxil olmaqla informasiya emalı sistemlərinin işlənilməsi,

yaradılması, istifadə edilməsi və maddi-texniki xidməti ilə əlaqədar olan sahələri rəsmi

təstiq etmişdir.

Beləliklə, informatika kompyuter texnikasına əsaslanır və onsuz  təsəvvüredilməzdir.

İnformatika — ən geniş tətbiq diapazonuna malik kompleks elmi fənndir. Onun əsas

istiqamətləri aşağıdakılardır:

hesablama sistemlərinin və proqpam təminatının işlənməsi;

informasiyanın ötürülməsi, qəbulu, çevrilməsi və saxlanması ilə əlaqədar olan

prosesləri öyrənən informasiya nəzəriyyəsi;

riyazi modelləşdirmə, hesablama və tətbiqi riyaziyyatın üsulları və onların

müxtəlif elm sahələrində fundamental tətbiqi tədqiqatlara tətbiq edilməsi;

süni intellekt metodları, insanın intellektual fəaliyyətində (məntiqi nəticə,

öyrənmə, nitqin başa düşülməsi, vizual qavrama, oyunlar və s.) məntiqi və analitik

təfəkkürün modelləşdirici metodları;

müxtəlif xarakterli mürəkkəb problemlər üzrə qərarların hazırlanması və

əsaslandırılması üçün istifadə olunan metodoloji vasitələri öyrənən sistem analizi;

bioloji sistemlərdə informasiya proseslərini öyrənən bioinformatika;

cəmiyyətin informasiyalaşması proseslərini öyrənən sosial informatika;

maşın qrafikası, animasiya metodları, multimedia vasitələri;

telekommunikasiya sistemləri və şəbəkələri, o cümlədən bütün bəşəriyyəti

vahid informasiya cəmiyyətində birləşdirən qlobal kompyuter şəbəkələri;

istehsalat, elm, təhsil, tibb, ticarət, kənd təsərrüfatı və təsərrüfat və ictimai

fəaliyyətin bütün digər növlərini əhatə edən müxtəlif proqramlar.

Rusiya akademiki A.A. Dorodnitsin informatikada qırılmaz surətdə və əhəmiyyətli

dərəcədə əlaqəli üç hissəni qeyd edir – texniki, proqram və alqoritmik vasitələr.

*Texniki vasitələr* və ya kompyuterlərin aparat hissəsi ingilis dilində***Hardware*** sözü ilə

işarə edilir ki, o da hərfi mənada «bərk məmulat» kimi tərcümə olunur.

***Software*** (hərfi mənada – «yumşaq məmulat») sözündən istifadə olunur. Bu söz maşının

özünün və proqram təminatının eyni əhəmiyyətə malik olduğunu, həmçinin proqram

təminatının modifikasiya olunmaq, uyğunlaşmaq və inkişaf etmək qabiliyyətini qeyd edir.

Məsələnin proqramlaşdırılmasından əvvəl həmişə onun ilkin verilənlərdən axtarılan

nəticəyə aparan əməliyyatlar ardıcıllığı şəklində olan həll üsulunun işlənilməsi, başqa sözlə

məsələnin həll alqoritminin işlənilməsi gəlir. İnformatikanın alqoritmlərin işlənilməsi və

onların qurulmasının metod və yollarının öyrənilməsi ilə əlaqədar olan hissəsinin işarə

olunması üçün***Brainware*** (ingiliscə brain — intellekt) terminindən istifadə olunur.

Kompyuter texnikasının imkanlarının tədricən artması, informasiya şəbəkələrinin

inkişafı, yeni informasiya texnologiyalarının yaranması cəmiyyətin bütün sahələrində:

istehsalda, elmdə, təhsildə, tibdə və s. mühüm dəyişikliklərə  gətirir.

**«İnformasiya»** termini**məlumat, izah, şərh** mənasını verən**«informatio»** latın

sözündən törəyir. Bu terminin geniş yayılmasına baxmayaraq, informasiya anlayışı elmdə

ən mübahisəli anlayışlardan biridir. İndiki zamanda elm geniş*informasiya* anlayışına xas

olan ümumi xüsusiyyətləri və qanunauyğunluqları tapmağa cəhd edir, lakin hələlik bu

anlayış əsasən intuitiv olaraq qalır və insan fəaliyyətinin müxtəlif sahələrində müxtəlif

məna tutumları kəsb edir:

**adi həyatda** kimisə maraqlandıran hər hansı verilənləri və ya məlumatları

informasiya adlandırırlar. Məsələn, hər hansı hadisə, kiminsə fəaliyyəti barədə

məlumatı və s. Bu mənada*«informasiya vermək» «öncə məlum olmayan nəyisə*

*xəbər vermək»* deməkdir;

**texnikada** informasiya dedikdə işarə və ya siqnallar formasında ötürülən

xəbərlər başa düşülür;

**kibernetikada** informasiya dedikdə istiqamətlənmə, aktiv fəaliyyət, idarəetmə

üçün, yəni sistemin yadda saxlanması, təkmilləşdirilməsi, inkişafı məqsədilə istifadə

olunan biliklər sahəsi başa düşülür (N. Viner).

İnformasiyanın ötürülməsi, qəbulu, çevrilməsi və yadda saxlanması ilə əlaqədar olan

prosesləri öyrənən elmin – informasiya nəzəriyyəsinin əsaslarını qoymuş amerika alimi

[Klod](http://file:///C:/Downloads/informatika/informatika/www.kbsu.ru/~book/theory/chapter1/shannon.html) [Şennon](http://file:///C:/Downloads/informatika/informatika/www.kbsu.ru/~book/theory/chapter1/shannon.html)**informasiyaya nəyinsə barəsində bizim biliklərimizin aradan qaldırılmış**

**qeyri-müəyyənliyi kimi baxır**.

Daha bir neçə tərifi göstərək:

İnformasiya – ətraf mühitin obyektləri və hadisələri, onların parametrləri,

xüsusiyyətləri və vəziyyəti haqqında onlara dair əldə olan qeyri-müəyyənlik

dərəcəsini, biliklərin natamamlığını azaldan məlumatlardır (N.V. Makarova);

İnformasiya – entropiyanın inkar edilməsidir (Leon Brillüen);

İnformasiya – strukturların mürəkkəblik ölçüsüdür (Mol);

İnformasiya – inikas olunmuş müxtəliflikdir (Ursul);

İnformasiya – inikas posesinin məzmunudur (Tuzov);

İnformasiya – seçim ehtimalıdır (Yaqlom).

İnformasiya barədə müasir elmi təsəvvürü kibernetikanın «atası» Norbert Viner çox

dəqiq ifadə etmişdir:

İnformasiya – bizim xarici aləmə uyğunlaşmağımız prosesində xarici aləmdən alınan

məzmunun və bizim hislərimizin ona uyğunlaşmasının işarə edilməsidir.

İnsanlar xəbər formasında informasiya mübadiləsi edirlər. Xəbər informasiyanın nitq,

mətn, əl-qol hərəkəti, baxış, təsvir, rəqəmli verilənlər, qrafiklər, cədvəllər və s. Şəklində

təqdim olunma formasıdır.

*Eyni bir informasiya xəbəri* (qəzetdə məqalə, elan, məktub, teleqram, arayış, hekayə,

çertyoj, radio verilişi və s.)*müxtəlif insanlar üçün onların əvvəlki biliklərindən, bu xəbərin*

*başa* *düşülmə* *səviyyəsindən* *və* *ona* *maraqdan* *asılı* *olaraq* *müxtəlif* *miqdarda*

*informasiyaya malik ola bilər*.

Belə ki, yapon dilində tərtib olunmuş xəbər bu dili bilməyən insan üçün heç bir yeni

informasiya daşımır, lakin yaponca bilən insan üçün yüksək informativ ola bilər. Əgər

xəbərin məzmunu başa düşülmürsə və ya artıq məlumdursa, məlum dildə ifadə olunmuş

xəbər də heç bir yeni informasiyaya malik deyil.

İnformasiya xəbərin xarakteristikası deyil, xəbərlə onun istehlakçısı arasında

münasibətdir. Heç olmasa, potensial istehlakçı olmasa, informasiya haqqında danışmaq

mənasızdır.

Hər hansı texniki qurğular vasitəsilə informasiya ilə avtomatlaşdırılmış iş haqqında

danışarkən, adətən ilk növbədə xəbərin məzmunu ilə deyil, bu xəbərdə nə qədər simvol

olması ilə maraqlanırlar.

Verilənlərin kompyuterlə emalına gəlincə, informasiya dedikdə simvolik işarələrin

(hərflər, rəqəmlər, kodlaşmış qrafik təsvirlər və səslər və s.) məna yükü daşıyan və

kompyuterin anladığı şəkildə təqdim olunan hər hansı ardıcıllığı başa düşülür. Belə simvol

ardıcıllığında  hər  yeni  simvol  xəbərin informasiya həcmini artırır.

**İnformasiya hansı şəkildə mövcud olur?**

İnformasiya aşağıdakı şəkillərdə mövcud ola bilər:

* mətn, şəkil, çertyoj, fotoqrafiya;
* işıq və ya səs siqnalları;
* radiodalğalar;
* elektrik və əsəb impulsları;
* maqnit yazıları;
* jest və mimika;
* iy və dad duyğuları;
* orqanizmlərin əlamət və xüsusiyyətlərini irsi olaraq ötürən xromosomlar və s.

İnformasiya xüsusiyyətləri nöqteyi-nəzərindən baxılan predmetlər, proseslər, maddi

və ya qeyri-maddi xüsusiyyətli hadisələr*informasiya obyektləri* adlanır.

**İnformasiya miqdarının ölçülməsi**

Məsələn, «Hərb və sülh» romanının mətnində, Rafaelin freskalarında və ya insanın

genetik kodunda nə qədər informasiya var? Elm bu suallara cavab vermir və ehtimal ki,

tezliklə verməyəcək. Bəs obyektiv olaraq informasiya miqdarını ölçmək

mümkündürmü? İnformasiya nəzəriyyəsinin ən vacib nəticəsi aşağıdakıdır:

Müəyyən, çox geniş şəraitdə informasiyanın keyfiyyət xüsusiyyətlərini nəzərə

almamaq, onun miqdarını ədədlə ifadə etmək, həmçinin verilənlərin müxtəlif qruplarında

olan informasiya miqdarını müqayisə etmək olar.

İndiki zamanda «informasiya miqdarı» anlayışının təyin olunmasına elə yanaşmalar

geniş yayılmışdır ki, onlarxəbərdəki informasiyanı onun yeniliyi və ya başqa sözlə,

bizim obyekt haqqındakı biliklərimizin qeyri-müəyyənliyinin azalması mənasında

qeyri-sərt izah etməyə əsaslanırlar. Bu yanaşmalar*ehtimal* və*loqarifm* riyazi

anlayışlarından istifadə edirlər.

Amerika mühəndisi R. Xartli 1928-ci ildə informasiyanın alınması prosesinə N sayda

bərabər ehtimallı xəbərdən ibarət əvvəlcədən verilmiş sonlu çoxluqdan bir xəbərin

seçilməsi kimi baxmış, seçilmiş xəbərdə olan I informasiya miqdarını isə N-in ikili loqarifmi

kimi təyin etmişdir.

Xartli düsturu:

*I* = log2*N*

Tutaq ki, birdən yüzə qədər ədədlər yığımından bir ədəd tapmaq lazımdır. Xartli

düsturuna görə bunun üçün tələb olunan informasiya miqdarını hesablamaq olar:

I=log2100=6,644. Beləliklə, düzgün tapılmış ədəd haqqında xəbər təxminən 6,644-ə

informasiya vahidinə bərabər informasiya miqdarına malikdir.

Bərabər ehtimallı xəbərlərə digər misallar göstərək:

1. metal pulu atdıqda «bir üzü düşdü», «digər üzü düşdü»;

2. kitabın səhifəsində «hərflərin sayı cütdür», «hərflərin sayı təkdir».

İndi müəyyən edək ki, «binanın qapısından birinci qadın çıxacaq» və «binanın

qapısından birinci kişi çıxacaq» xəbərləri bərabər ehtimallıdırmı. Bu suala birmənalı cavab

vermək olmaz. Hər şey ondan asılıdır ki, hansı binadan söhbət gedir. Əgər bu, məsələn,

metro stansiyasıdırsa, onda qapıdan birinci çıxmaq ehtimalı kişi və qadın üçün eynidir,

amma əgər bu, hərbi kazarmadırsa, onda kişi üçün bu ehtimal qadına nisbətən çox

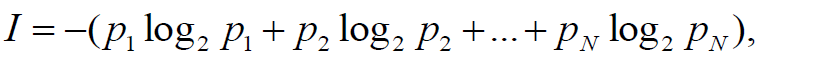
yüksəkdir.

Bu növ məsələlər üçün amerika alimi Klod Şennon 1984-cü ildə informasiya

miqdarının təyin olmasının yığımdakı xəbərlərin mümkün eyni olmayan ehtimalını nəzərə

alan digər düsturunu təklif etmişdir.

Şennon düsturu:



burada*pi* - N sayda xəbərdən ibarət yığımda məhz i-ci xəbərin seçilməsinin

ehtimalıdır.

Asanlıqla görünür ki, əgər p*1,...,* pN ehtimalları bərabərdirsə, onda onların hər biri

*1/N*-ə bərabərdir və Şennon düsturu Xartli düsturuna çevrilir.

İnformasiya miqdarının müəyyən olunmasına baxılan iki yanaşmadan əlavə digər

yanaşmalar da var. Yadda saxlamaq lazımdır ki, ixtiyari nəzəri nəticələr yalnız başlanğıc

yol verilmələrlə qeyd olunan müəyyən hadisələr dairəsinə tətbiq oluna bilər.

İnformasiya vahidi kimi Klod Şennon bir**bit** (*ingiliscə bit –****bi****nary digi****t*** – ikili rəqəm)

qəbul etməyi təklif etmişdir.

İnformasiya nəzəriyyəsində bit – iki bərabər ehtimallı xəbərin (pulun «bir üzü» –

«digər üzü», «cüt» – «tək» və s. növ) fərqləndirilməsi üçün zəruri olan informasiya

miqdarıdır. Hesablama texnikasında kompyuterin yaddaşının verilənlərin və əmrlərin

maşın daxilində təqdim olunması üçün istifadə olunan iki «0» və «1» işarələrindən birinin

yadda saxlanması üçün zəruri olan ən kiçik «payını»*bit*  adlandırırlar

Bit həddən artıq kiçik informasiya vahididir. Praktikada çox zaman daha iri vahiddən –

**səkkiz bitə** bərabər olan**bayt**dan istifadə edirlər. Kompyuterin klaviaturasındakı əlifbanın

256 simvolundan hər hansını kodlaşdırmaq üçün məhz 8 bit tələb olunur (256=28).

1 bayt = 8bit;

1Kbayt (Kb)= 1024 bayt;

1Meqabayt (Mb) = 1024 Kbayt;

1Qiqabayt (Qb) = 1024 Meqabayt;

1 Terabayt (Tb)= 1024 Qiqabayt;

1 Petabayt (Pb)= 1024 Tearbayt(250 bayt).

Informasiya texnologiyası - konkret texniki və proqram vasitəsidir ki, onun köməyi ilə biz həyatın bütün sahələrində müxtəlif əməliyyatlar yerinə yetiririk.

Tətbiqi informatikada mərkəzi yeri kompüter tutur (kompüter ingilis dilində hesablama deməkdir). Fərdi kompüter – informasiyanın qəbulu, emalı, saxlanması və ötürülməsini təmin edən elektron qurğudur.

Iqtisadi informasiya – istehsal prosesini,maddi rifahın paylanması, mübadiləsi və onun istehlakını əks etdirən və ona xidmət edən informasiyadır.

Kateqoriya kimi “iqtisadi informasiya” bir tərərfdən informasiya anlayışına uyğun olur, digər tərəfdən kəsilməz surətdə iqtisadiyyat və xalq təsərrüfatının idarəedilməsi ilə bağlıdır.

Informasiya – insanın öz hissiyyat üzvləri vasitəsilə təbiətdən və cəmiyyətdən qəbul etdiyi məlumatlardır. Informasiyanı yaratmaq, ötürmək, saxlamaq, emal etmək mümkündür. Informasiya siqnallar vasitəsilə ötürülür, kodlaşdırma ilə yaddaşda saxlanılır və iki yerə bölünür: ədədi və analoq informasiyalar.

Insanlar hissiyyat üzvlərinə görə analoq, kompüter isə ədədi iinformasiyalarla işləyir.

Analoq və ədədi informasiyaların fərqi birinciniin kəsilməz, ikincisinin diskret olmasıdır. Məsələn, kəsilməz funksiyanın qrafiki analoq informasiyadır.

Informasiyalar siqnallar vasitəsilə ötürüldüyündən onları ikilik sistemdə kodlaşdırmaq lazımdır. Kodlaşdırma isə simvollarla yerinə yetirildiyindən artıq istənilən informasiya kompüterdə ədədi informasiyaya çevrilir və o, *bit* ölçü vahidi ilə ölçülür. Bu ölçü *hə* - *yox* və ya *0 1* qiymətini ala bilər. Buradan da ikilik say sistemi yaranmışdır.

1 bayt = 8bit;

1Kbayt (Kb)= 1024 bayt;

1Meqabayt (Mb) = 1024 Kbayt;

1Qiqabayt (Qb) = 1024 Meqabayt;

1 Terabayt (Tb)= 1024 Qiqabayt;

1 Petabayt (Pb)= 1024 Tearbayt(250 bayt).

Əgər bir səhifədə 2500 işarə yerləşərsə, onda 1 Mb 400 səh., 1Qb 400 000 səhifə tutar.



Əgər bir səhifədə 2500 işarə yerləşərsə, onda 1 Mb 400 səh., 1Qb 400 000 səhifə tutar.



1642 –ci ildə Blez Paskal ədədlərin toplanmasını yerinə yetirən qurğu, 1673 – cü ildə Leybnits arifmometr düzəltdilər. **1834cü il.** İngilis alimi Çarlz Bebbic*"analitik" maşının* layihəsini tərtib etdi. Bu

maşına informasiyanın giriş və çıxış qurğuları, ədədlərin saxlanması üçün yaddaş

qurğusu, hesab əməllərini yerinə yetirə bilən qurğu və maşının əməliyyatlar ardıcıllığını

idarə edən qurğu daxil edilmişdi. Əmrlər perfokartların köməyilə daxil edilirdi.

Layihə reallaşmadı.

**1876-cı il.** İngilis mühəndisi Aleksander Bell*telefonu* ixtira etdi.

**1890-cı il.** Amerika mühəndisi German Xollerit*statistik tabulyator* yaratdı, harada ki,

perfokartlara yazılmış informasiya elektrik cərəyanı ilə deşifrə olunurdu. Tabulyator ABŞ-

da əhalinin siyahıya alınma nəticələrinin emal edilməsi üçün istifadə edilmişdi.

**1892-ci il.** Amerika mühəndisi U.Barrouz birinci kommersiya*cəmləyicisini* istehsala

buraxdı.

**1897-ci il.** İngilis fiziki C. Tomson*elektron-şüa borusunu* icad etdi.

**1901-ci il.** İtalyan fiziki Qulelmo Markoni Avropa ilə Amerika arasında*radiorabitə*

qurdu.

**1904—1906-cı illər.** Elektron*diod* və*triod* ixtira edildi.

**1930-cu il.** Massaçuset texnoloji institutunun (MTİ) professoru Vannevar Buş

*diferensial analizator* qurdu ki, onun meydana gəlməsi ilə müasir kompyuter erasının

başlanğıcını əlaqələndirirlər. Bu, təyyarə kimi hərəkət edən obyektlərin davranışını və ya

qüvvə sahələrinin, məsələn qravitasiya sahəsinin təsirini əvvəlcədən xəbər verməyə imkan

verən mürəkkəb diferensial tənlikləri həll edə bilən ilk maşın idi.

**1945-ci il.** Con fon Neyman «Edvak maşını haqqında ilkin məruzə» hesabatında

müasir kompyuterlərin əsas iş prinsiplərini və komponentlərini qısa və dürüst ifadə etdi.

**1946-cı il** Amerikalılar C. Ekkert və C. Mouçli ilk elektron rəqəmli**«Eniac»**

(Electronic Numerical Integrator and Computer) kompyuterini hazırladılar. Maşın 20 min

elektron lampaya və 1,5 min releyə malik idi. O, bir saniyədə 300 vurma və ya 5000

toplama yerinə yetirərək «Mark—1»-ə nisbətən min dəfə tez işləyirdi.

**1948-ci il** Bell Laboratories amerika firmasında fiziklər Uilyam Şokli, Uolter Bratteyn

və Con Bardin*tranzistor* yaratdılar. Bu müvəffəqiyyət üçün onlara Nobel mükafatı verildi.

**1948-ci il.** Norbert Viner süni intellekt sahəsində bütün sonrakı tədqiqatlara təsir

göstərmiş "Kibernetika" kitabını dərc etdirdi.

**1949-cu il.** İngiltərədə Moris Uilksin rəhbərliyi altında dünyada ilk dəfə proqramları

yaddaşda saxlayan**EDSAC** kompyuteri yaradıldı.

**1951ci il.** Kiyevdə kontinental Avropada 600 elektron lampaya malik ilk**MESM** (kiçik

elektron hesablama maşını) kompyuteri yaradıldı. Yaradıcısı S.A. Lebedevdir.

**1952-ci il.** S.A. Lebedevin rəhbərliyi altında Moskvada**BESM-1** (böyük elektron

hesablama maşını) kompyuteri – o zaman üçün Avropada ən məhsuldar və dünyada ən

yaxşı olan maşınlardan biri yaradıldı.

**1953-cü il.** Cey Forrester*maqnit nüvələrində əməli yaddaşı* (core memory)

reallaşdırdı, hansı ki, kompyuterləri əhəmiyyətli dərəcədə ucuzlaşdırdı və onların sürətini

artırdı. Maqnit nüvələrindəki yaddaş 70-ci illərin əvvəlinə qədər geniş istifadə olunurdu.

Onu yarımkeçirici elementlərdə olan yaddaş əvəz etdi.

**1958-ci il.** Texas Instruments firmasından Cek Kilbi ilk*inteqral sxemi* yaratdı.

**1957-ci il. Fortran** (Con Bekus) dili barədə ilk məlumat.

**1957ci il.** NCR amerika firması tərəfindən*tranzistorlar üzərində qurulmuş ilk*

*kompyuter* yaradıldı.

**1959-cu il.** Proqramlaşdırma dilləri sahəsində uzun müddət standart olmuş**Alqol** dili

barədə ilk məlumat.

**1961-ci il.** IBM Deutschland firması*modem* vasitəsi ilə kompyuterin telefon xəttinə

qoşulmasını reallaşdırdı.

**1964-cü il. IBM/360** – üçüncü nəsil maşınları ailəsinin istehsala buraxılması

başlandı.

**1965-ci il.** Dortmund kollecində (ABŞ) C. Kemeni və T. Kurts**Basic** proqramlaşdırma

dilini işlədilər.

**1965-ci il.** Seymur Peypert (Seymour Papert)**LOGO** dilini – uşaqlar üçün kompyuter

dilini işlədi.

**1968-ci il.** Мikroprosessorlar və kompyuterin digər inteqral sxemlərinin istehsalı

sahəsində tanınmış lider olan**Intel** firmasının əsası qoyuldu.

**1970-ci il.** İsveçrəli Niklaus Virt**Pascal** dilini işlədi.

**1971-ci il.** Edvard Xoff mismar başından böyük olmayan kristalda yerləşmiş 2250

tranzistordan ibarət**Intel-4004** mikroprosessoru yaratdı. Bu mikroprosessor ümumiyyətlə

süni intellektual sistemlərin və xüsusilə fərdi kompyuterlərin yaradılmasına yol açan

həqiqətən inqilabi ixtira oldu.

**1971-ci il.** Fransız alimi Alan Kolmari**Proloq** (PROgramming in LOGic) məntiqi

proqramlaşdırma dilini işlədi.

**1972-ci il.** Bell Laboratories-dən olan Dennis Ritçi**C** dilini işlədi.

**1973-cü il.** Ken Tompson və Dennis Ritçi**UNIX** əməliyyat sistemini yaratdılar.

**1973-cü il. IBM** (International Business Machines Corporation) firması*«vinçester»*

**1961-ci il.** IBM Deutschland firması*modem* vasitəsi ilə kompyuterin telefon xəttinə

qoşulmasını reallaşdırdı.

**1964-cü il. IBM/360** – üçüncü nəsil maşınları ailəsinin istehsala buraxılması

başlandı.

**1965-ci il.** Dortmund kollecində (ABŞ) C. Kemeni və T. Kurts**Basic** proqramlaşdırma

dilini işlədilər.

**1965-ci il.** Seymur Peypert (Seymour Papert)**LOGO** dilini – uşaqlar üçün kompyuter

dilini işlədi.

**1968-ci il.** Мikroprosessorlar və kompyuterin digər inteqral sxemlərinin istehsalı

sahəsində tanınmış lider olan**Intel** firmasının əsası qoyuldu.

**1970-ci il.** İsveçrəli Niklaus Virt**Pascal** dilini işlədi.

**1971-ci il.** Edvard Xoff mismar başından böyük olmayan kristalda yerləşmiş 2250

tranzistordan ibarət**Intel-4004** mikroprosessoru yaratdı. Bu mikroprosessor ümumiyyətlə

süni intellektual sistemlərin və xüsusilə fərdi kompyuterlərin yaradılmasına yol açan

həqiqətən inqilabi ixtira oldu.

**1971ci il.** Fransız alimi Alan Kolmari**Proloq** (PROgramming in LOGic) məntiqi

proqramlaşdırma dilini işlədi.

**1972-ci il.** Bell Laboratories-dən olan Dennis Ritçi**C** dilini işlədi.

**1973-cü il.** Ken Tompson və Dennis Ritçi**UNIX** əməliyyat sistemini yaratdılar.

**1973-cü il. IBM** (International Business Machines Corporation) firması*«vinçester»*

**1961-ci il.** IBM Deutschland firması*modem* vasitəsi ilə kompyuterin telefon xəttinə

qoşulmasını reallaşdırdı.

**1964-cü il. IBM/360** – üçüncü nəsil maşınları ailəsinin istehsala buraxılması

başlandı.

**1965-ci il.** Dortmund kollecində (ABŞ) C. Kemeni və T. Kurts**Basic** proqramlaşdırma

dilini işlədilər.

**1965-ci il.** Seymur Peypert (Seymour Papert)**LOGO** dilini – uşaqlar üçün kompyuter

dilini işlədi.

**1968-ci il.** Мikroprosessorlar və kompyuterin digər inteqral sxemlərinin istehsalı

sahəsində tanınmış lider olan**Intel** firmasının əsası qoyuldu.

**1970-ci il.** İsveçrəli Niklaus Virt**Pascal** dilini işlədi.

**1971-ci il.** Edvard Xoff mismar başından böyük olmayan kristalda yerləşmiş 2250

tranzistordan ibarət**Intel-4004** mikroprosessoru yaratdı. Bu mikroprosessor ümumiyyətlə

süni intellektual sistemlərin və xüsusilə fərdi kompyuterlərin yaradılmasına yol açan

həqiqətən inqilabi ixtira oldu.

**1971ci il.** Fransız alimi Alan Kolmari**Proloq** (PROgramming in LOGic) məntiqi

proqramlaşdırma dilini işlədi.

**1972-ci il.** Bell Laboratories-dən olan Dennis Ritçi**C** dilini işlədi.

**1973-cü il.** Ken Tompson və Dennis Ritçi**UNIX** əməliyyat sistemini yaratdılar.

**1973-cü il. IBM** (International Business Machines Corporation) firması*«vinçester»*

tipli ilk*sərt diski* hazırladı.

Proqramla idarəolunan hesablama maşınları yaratmaq ideyası Kembric Universitetinin professoru Ç. Bebbis tərəfindən XIX əsrin I yarısında deyilmişdir. O, böyük Britaniya üçün çox əhəmiyyətli olan dənizçilik elmi üçün cədvəllər düzəldirdi. O, mexaniki hesablama avtomatının proyektini işlədi və onu analitik maşın adlandırdı. Lakin bu dahi iş realizə olunmadı.

1890 – cı ildə ABŞ – da Xolleriz verilənlərin işlənməsini avtomatlaşdıran və informasiya daşıyıcısı kimi perfokartlardan istifadə etdi və İBM (İnternational Business Mashines ) firmasının əsasını qoydu (1896).

1937 il – Atanasov ədədləri onluq say sistemindən ikilik say sisteminə çevirən elektromexaniki qurğu yaratdı.

1937 – Eytkenin ideyası ilə İBM firması Mark – 1 releli elektrik maşınını yarat­­­­­­­­­­­­dı (vurmanı 3 saniyəyə yerinə yetirən).

1937 il – K. Tsuze proqramla idarəolunan elektromexaniki hesablayıcı maşın yaratdı. 1945 –ci il –P. Ekert və C. Monçli lampalı ENİAK maşını yaratdılar (sürəti Mark 1 –dən 1000 dəfə artıq).

SSRI –də 1947 ci ildə başlandı və 1951 –ci ildə ilk MESM maşını yaradıldı (Akademik Lebedev).

1943 –cü il – dünyada ilk işlək EHM yaradıldı (Böyük Britaniya, Türinq).

1945 – ci ildə Con fon Neyman proqramı öz yaddaşında saxlayan maşın haqında öz məşhur tezislərini hazırladı:

EHM –in işini idarə edən proqramlar ayrı – ayrı əmrlər ardıcıllığından ibarət olmalı;

Alınan aralıq nəticələrdən asılı olaraq hesablamalar ardıcıllığını dəyişməyə imkan verən şərti keçiddən istifadə edilməsi;

Proqramın saxlanması. Əmrlər ədədi şəkildə göstərilir və verilənlərin saxlanıldığı operativ yaddaşda saxlanılır; beləliklə, əmrləri hesabi qurğuya göndərmək və onları adi ədədlər kimi çevirmək olar ki, bu da hesablama prosesi zamanı öz-özlərini dəyişməyi bacaran proqramlar yaratmağa imkan verir;

Informasiyanın göstərilişi üçün ikilik say sistemindən istifadə;

Ierarxik yaddaş qurğusunun tətbiqi.

1949 – cu ildə Böyük Britaniyada Moris Uilks bu prinsiplər əsasında ilk kompüter yaratdı. Neyman kompüterin necə qurulmasını göstərirdi:

Hesabi – məntiqi qurğu – hesabi və məntiqi əməlləri yerinə yetirir;

Idarəetmə qurğusu – proqramın yerinə yetirilməsi prosesini təşkil edir;

Yaddaş qurğusu – proqram və verilənlərin saxlanmasını həyata keçirir;

Xarici qurğu – informasiyanın giriş – çıxışı üçündür.

**Mühazirə 2**

**Say sistemləri**

Iki növ say sistemi istifadə olunur: mövqesiz və mövqeli.

Mövqesiz say sistemlərində ədədin yazılışında rəqəmlər on­ların durduğu yerdən asılı deyil. Məsələn, rum rəqəmləri möv­qesiz say sistemidir. XXXII (otuz iki) ədədinin yazı­lı­şın­da X simvolu hər yerdə on rəqəmini ifadə edir (üç onluq və iki­ təklik).

Mövqeli say sistemlərində isə hər bir rəqəm onun durduğu yerlə müəyyən olunur. Məsələn, onluq say sistemində 424 ədədinin yazılışında birinci 4 yüzlüyü, ikinci 4 isə təkliyi göstərir. Həmin ədədi aşağıdakı kimi yaza bilərik:

400 + 20 + 4 = 4 ∙ 102 + 2 ∙101 + 4 ∙100

Hər bir say sistemi öz əsası ilə xarakterizə olunur. Say sisteminin əsası istənilən narural ədəd ola bilər – 2, 3, 4, və s. Hər bir ədəd aşağıdakı şəkildə yazıla bilər:

a n-1 q n-1 + a n-2 q n-2 + ... + a 1 q1 + a0 q0 + a-1 q-1 + ... + a-m q-m

Burada ai  say sisteminin əsası, n və m isə uyğun olaraq tam və kəsr hissəni göstərən ədədlərdir.

Məsələn,

1011,11 =1∙23 +0∙22 + 1∙21 + 1∙20+1∙2 -1 +1∙2 -2

**Say sistemləri necə yaranır?**

**Hər bir say sistemində istifadə olunan rəqəmlərin sayı**  əsası göstərən rəqəmin sayı qədər olur. Məsələn, ikilik say sis­te­mində iki rəqəm – 0 və 1 istifadə olunur. Üçlük say sis­te­min­də üç rəqəm – 0,1 və 2, səkkizlikdə səkkiz rəqəm – 0-dan 7-yə qə­dər və s. Istifadə olunur. Bütün say sistemlərində ən böyük rəqəm 10 kimi işa­rə olunur. Aşağıda ilk on tam ədədin müxtəlif say siste­mlə­rin­də yazılışı göstərilmişdir:

Ikilik say sistemində: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001;

Üçlük say sistemində: 0, 1, 2, 10, 11, 12, 20, 21, 22, 100;

Beşlik say sistemində: 0, 1, 2, 3, 4, 10, 11, 12, 13, 14;

Səkkizlik say sistemində: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11.

**Kompüterdə əsasən hansı say sistemindən istifadə olu­nur­?**

Kompüterdə əsasən əsası 2 - nin qüvvəti şəklində olan say sistemləri istifadə olunur.

* ikilik (0, 1 rəqəmləri istifadə olunur);
* səkkizlik (0, 1, ..., 7 rəqəmləri istifadə olunur);
* onaltılıq (0, 1, ..., 9 rəqəmləri və A, B, C, D, E, F sim­vol­ları istifadə olunur ).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **10-luq** | **2 -lik** | **8-lik** | **16-lıq** | **10-luq** | **2-lik** | **8-lik** | **16-lıq** |
| **0** | 0 | 0 | 0 | **10** | 1010 | 12 | A |
| **1** | 1 | 1 | 1 | **11** | 1011 | 13 | B |
| **2** | 10 | 2 | 2 | **12** | 1100 | 14 | C |
| **3** | 11 | 3 | 3 | **13** | 1101 | 15 | D |
| **4** | 100 | 4 | 4 | **14** | 1110 | 16 | E |
| **5** | 101 | 5 | 5 | **15** | 1111 | 17 | F |
| **6** | 110 | 6 | 6 | **16** | 10000 | 20 | 10 |
| **7** | 111 | 7 | 7 | **17** | 10001 | 21 | 11 |
| **8** | 1000 | 10 | 8 | **18** | 10010 | 22 | 12 |
| **9** | 1001 | 11 | 9 | **19** | 10011 | 23 | 13 |

* ***Səkkizlik və onaltılıq say sistemlərində verilmiş ədəd­ləri ikilik say sisteminə çevirmək üçün hər bir rəqəmi uy­ğun ola­­­raq onların ikilik triada (üçlük) və ya******tetradası (dörd­­lük­) ilə əvəz etmək lazımdır.***
* Məsələn,







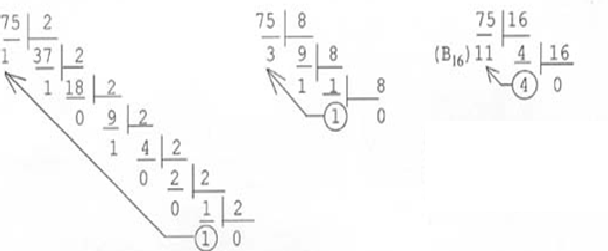
***Ikilik say sistemində verilmiş ədədi səkkizlik və onal­tı­lıq say sistemlərinə çevirmək üçün ikilik ədədi vergüldən so­la (tam hissəni) və vergüldən sağa (kəsr hissəni) uyğun ola­raq triadalara və tetradalara bölmək lazımdır.***

Tamamlanmayan triada və ya tetradaya sıfırlar əlavə olu­nur (tam hissədə əvvələ, kəsr hissədə isə sona). Məsələn,



Onluq say sistemində verilmiş N tam ədədini q əsaslı say sis­teminə çevirmək üçün N-i q-yə bölmək lazımdır. N ədə­di­nin yeni say sistemində yazılışı bölmədən alınan qalıqların tərs dü­zülüşdə yazılışı olur

**Misal .** Onluq say sistemində verilmiş 75 ədədini ikilik, səkkizlik və onaltılıq say sisteminə çevirək:

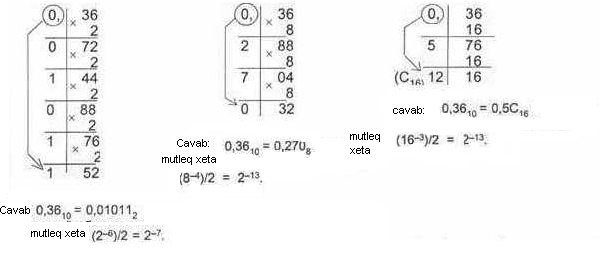


**Cavab**: 75 = 1 001 0112 = 1138 = 4B16

**Düzgün onluq kəsrin digər mövqeli say sistemlərinə çevril­məsi. F** düzgün onluq kəsrini **q** əsaslı say sisteminə çevirmək üçün F kəsrini q-yə vurmaq lazımdır, nəticə yenə onluq say sisteminə yazılır, sonra q-yə vurmadan alınan kəsr hissə yenidən q-yə vurulur və s., bu o vaxta qədər davam etdirilir ki, ya növbəti hasildə kəsr hissə 0-a bərabər olsun və ya müəyyən dəqiqliklə çevirməni sona çatdırmaq lazımdır.

F ədədinin kəsr hissəsinin yeni q say sistemində göstərişi onların alınma sırası ilə ya­zıl­­maq­la alınmış hasillərin tam hissələr ardıcıllığı olacaq. Əgər F ədədinin çevrilməsi üçün tələb olunan dəqiqlik ver­gül­dən sonra k sayda simvoldan ibarətdirsə, onda bu zaman limit müt­ləq xəta q-(k+1)/2 –yə bərabər olar.

Onluq say sistemində verilmiş 0,36 ədədini iki­lik, səkkizlik və onaltılıq say sistemlərinə çevirək:



Aşağıdakı əməliyyat hansı say sistemində veril­miş­dir?

21 + 24 = 100  
**Həlli:** Fərz edək ki, x - axtarılan say sisteminin əsasıdır. Onda

100x = 1 · x2 + 0 · x1 + 0 · x0,

21x = 2 · x1 + 1 · x0,

24x = 2 · x1 + 4 · x0.

Beləliklə,

x2 = 2x + 2x + 5

və ya

x2 - 4x - 5 = 0.

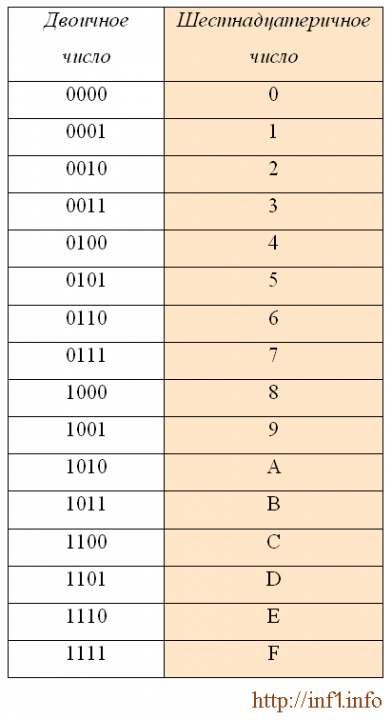
Bu kvadrat tənliyin müsbət kökü x = 5-dir.   
**Cavab**: Ədədlər beşlik say sistemində verilmişdir.

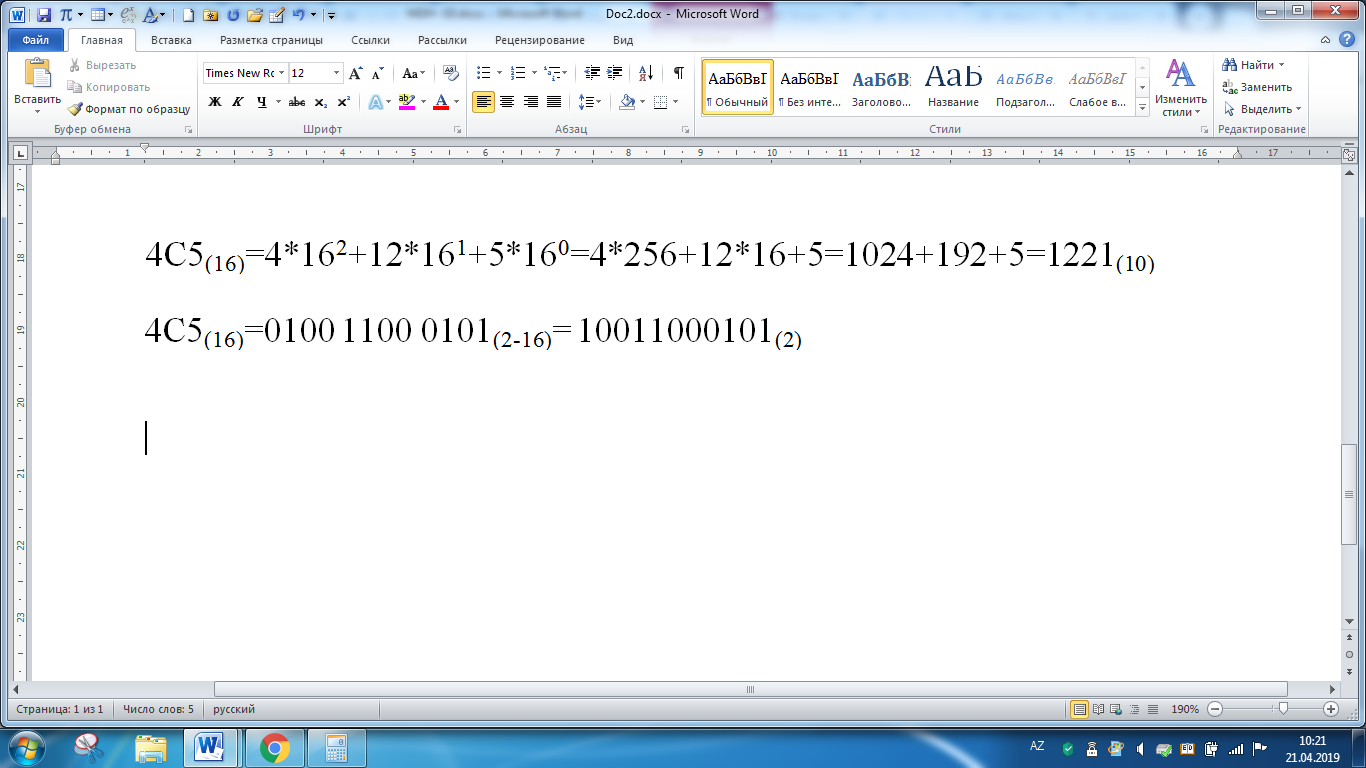
Kompüterdə istifadə olunan daha başqa alternativ say sistemi olaraq **“16-lıq sistemi”** göstərmək olar. Bu sistemdə də 4ədəd 2-lik registrdən istifadə olunur. Onluq say sistemindən fərqli olaraq burada istifadəsiz qalan rəqəm yoxdur. İstifadə olunan 16 rəqəm aşağıdakılardır:

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

Lakin 9-dan sonrakı rəqəmlər iki mərtəbə ilə göstərilmiş ola bilməzlər. Çünki pozision say sistemlərində ədədi təsvir etmək üçün yalnız sistemin əsası qədər rəqəmdən istifadə oluna bilər. Yəni hər mərtəbədə ancaq bir rəqəm yerləşməlidir prinsipi götürülməlidir. Bunun üçün 9-dan sonra əlavə rəqəm təsviri belə qəbul edilmişdir:

10-A; 11-B; 12-C; 13-D; 14-E; 15-F





Göründüyü kimi, təsvirin yığcamlığı baxımından 16-lıq sistem üstünlüyə malik deyildir. Lakin, kompüterin daxili quruluşunda bəzi halarda ondan istifadə səmərəli olduğu üçün ona müraciət olunur.

**Say sistemləri operativ yaddaşda ünvanlaşdırmanın əsası kimi**

Operativ yaddaşın tutumu, ümumiyyətlə,bit- informasiyanın həcmi ilə müəyyən olunur. Məsələn, 4 Qiqa-bit ümumən yaddaşda informasiya vahidinin (yuva) olduğunu göstərir. Bunların hamısı, təbii, ünvanlaşdırılan deyildirlər, yəni yalnız onlardan yaradılmış hər hansı qrupun ünvanlaşdırılması nəzərdə tutulur ki, onlara da müraciət oluna bilir. Ünvanlaşırma əksər hallarda 2-lik vəya 16-lıq sistemlərdə həyata keçirilir.

Ünvanlaşırmada ən kiçik müraciət olunan qrup baytdır. Sonra 2 bayt, 4 və 8 bayt olaraq “kompüter sözü” əmələ gətirilir ki, bunlara konkret ünvanlar verilir.

Ünvanlaşdırma ilə əlaqədar olan bir məsələ bayt düzülüşünün myəyyənləşdirilməsidir. Baytların birləşdirilərək müraciət olunan ünvan təşkil etməsini artan və azalan sıra ilə həyata keçirmək mümkündür.

Məsələ ondadır ki, bəzi hallarda kompüterlərin ünvanlaşdırılmasında göstərilən fərqli qaydalar informasiyanın ötürülməsində nəzərə alınmalı olur ki, bu da əlavə proqram tələb edir.

Operativ yaddaşa yazma və oxuma zamanı səhvlərin olub-olmadığını aşkar etmək üçün **nəzarət alqoritmləri** tətbiq edilir. Nəzarət bir dərəcəli və çox dərəcəli ola bilər. Əlbəttə, bir dərəcənin itirilməsinə qarşı qoyulan nəzarət nisbətən aşağı etibarlılığa malikdir. Lakin buna baxmayaraq, kompüterdə səhv nəticələrin alınmasının qarşısını xeyli dərəcədə ala bilir.

Ən geniş tətbiq olunan nəzarət üsulu vahidlərin “cütlülük” şərtinin yoxlanılmasıdır. Kompüter sözündə nəzarət üçün bu məqsədlə bir dərəcəsi istifadə olunur. Buraya informasiyanı daşıyan kodda “1”-lərin cüt olduğu halda “1”-in yazılması , əks halda isə “0” –ın qalması həyata keçirilir. Əgər kod itgisi baş vermişdirsə (**və o yeganədirsə**) bu səhv vahidlərin sayında özünü göstərəcəkdir. Bu zaman ya proqramın işi dayandırılacaq, ya da əməliyyat ötürüləcəkdir. Kompüterin səhv nəticə verməsi onun işinin dayanmasından qat-qat çox təhlükəli hesab olunur.

Qeyd edək ki, mükəmməl alqoritmlərin tətbiqi nəticəsində nəinki səhvlər aşkar oluna bilir, həm də onların korreksiya olunması həyata keçirilə bilir. Belə alqoritmlərə nümunə olaraq Xemminqin **[Hamming, 1950**] alqoritmini götərmək olar.

Daha çox dərəcəli nəzarət alqoritmləri vardır ki, bunları realizə etmək əksər hallarda hesablama sürətini həddən artıq azaltmış olur.

Əlifba-rəqəm informasiyasının kodlaşdırılması prinsip etibarı ilə 1 baytlıq registr əsasında problemsiz təsvir oluna bilər. Buna səbəb belə mətn simvollarının sayının 256 –dan çox olmamasıdır. Əgər klaviaturanın böyük və kiçik hərflərini, rəqəmləri, digər simvolları nəzərə alsaq, belə təsvir elementlərinin sayının 256-dan az olduğu qənaətinə asınlıqla gəlmək olur.

Burada məsələni qismən mürəkkəbləşdirən klaviaturaların milli mahiyyət daşıması və yazı şriftlərində bu arqumentin nəzərə alınmalı olduğu ilə bağlıdır. Bu sahədə kodlaşdırma prinsip etibarı ilə çox sadə həll olunan məsələdir. Simvollar ardıcıl olaraq unikal nömrələrlə təchiz olunur və onların nömrələr cədvəli tərtib olunur.

Kompüter texnikası və informasiya texnologiyaları ilə bağlı başqa sahələrdə standartlaşdırma məqsədi ilə əlifba-rəqəm informasiyasının ümumi kodlaşdırma sistemləri yaradılmışdır. Bunlar içərisində **ASCII** (**American Standard Code for Information Interchange** – informasiya mübadiləsi üçün Amerika standart) kodaşdırma sistemini qeyd etmək lazımdır.

Bu sistem 7-bit ölçüyə hesablanmış olduğundan belə registrdə 128 simvolun təsvirinə imkan varır. Bu sistemdə 0-dən 1F-ə qədər olan kodlar (16-lıq say sistemində) idarəedici simvollar olub, çap olunan simvollara aid deyildirlər. 128-dən 255-ə qədər olan kodlar ASCII-in kodlar siyahısında istifadə olunmurlar. Bu kodlar kompüterdə istifadə olunan digər məqsədlər üçün lazımdır.

Bu kodlar bir çox informasiya mübadilə sistemlərində informasiya massivlərinin başlıqlarının qeyd olunması üçün xüsusi simvol olaraq istifadə olunurlar.

**Unicode**

Kompüter sənayesi əsasən Amerika Birləşmiş Ştatlarında inkişaf etdiyindən kodlaşdırma da əsas etibarı ilə ingilis dilinə uyğunlaşdırılmışdır. Lakin kompüterin ümumdünya əhəmiyyəti daşıması və kompüterlərin istehsal və istehlak ərazilərinin bütün dünya olduğu səbəbindən həm klaviaturanın, həm də kodlaşdırma sisteminin uyğunlaşdırılması bir problem kimi diqqəti cəlb etmişdir. İlk təşəbbüs olaraq **Latin-1** kodlama sistemi meydana çıxır ki, bu kodlaşdırma 8-dərəcədən istifadə etməyə başlayır. Bu ona görə zəruri olur ki, lazım olan şriftlərin siyahısınə daha 128 simvol əlavə edilir. Buraya latın hərfləri əlavə olunur.

Növbəti cəhd **IS 8859** stanartı sayıla bilər ki, buraya **kod səhifəsi** anlayışı daxil edilməli olur. Kod səhifəsi dedkdə 256 simvoldan ibarət toplam başa düşülür. Bu simvollar 1,2,...256 –ardıcıllığınan ibarətdir.

Məsələn 1-simvolu IS 8859-dan sonra yazılır və o **Latin**-1 qrupunu müəyyən edir. **IS 8859-2** standartı slavyan qrupunu, IS 8859-3 iə türkiyə, malta qruplarını əks etdirir. Lakin belə kodlaşdırma sisteminin o mənfi cəhəti vardır ki, kompüter təmamilə dil qrupundan asılı vəzyyətə düşür. Proqram təminatı özünü doğrultmayan dərəcədə mürəkkəbləşmiş olur. Digər tərəfdən bu sistem çin və Yapon ieroqifləri ilə işi heç cürə təmin edə bilmir.

Bir sıra kompüter şirkətləri qrupu yeni **Unicode**sistemi yaratmağa nail ola bildilər ki, sonradan onu beynəlxal standart kimi elan edə bildilər. Bu kodlaşdırma sistemin əsas cəhəti dərəcələrin sayının 16-ya çatdırılması ilə kodlaşdırma obyktlərinin sayının olduqca çoxaldılmasıdır. Hazıırda bu kodlaşdırma sistemi get-gedə daha çox milli əlifbaları əhatə etmiş olur.

**Mühazirə 3**

Kompyuterə yazılmış mətində hər bir simvol (əlifba hərfi, rəqəm və ya durğu işarələr)

unikal bit ardıcıllığı ilə əvəz olunur. Mətnlərdə istifadə olunan hər bir simvol üçün Amerika Milli Standartlar İnstitutu

(ANSİ) tərəfindən 1918ci ildə standart***ASCII kod*** (*American Standard Codefor*

*Information Interchanging*) qəbul edilmişdir. Bu kod bir simvolun təsviri üçün

7 bitdən istifadə edir. Onun vasitəsi ilə yalnız ingilis əlifbasının böyük və kiçik hərflərini, 0-dan

9-a kimi rəqəmləri, durğu işarələrini və bir sıra simvolları (növbəti sətrə keçmə,

tabulyasiya, aralıq və s.) təsvir etmək mümkündür. Bugünki ASCII standart koduna 8ci bit

 əlavə

olunub. Hər simvolun təsvirində böyük bit sıfra bərabər götürülür, yəni bir simvolun

təsviri üçün 1 baytdan istifadə olunur. Böyük bitin qiyməti 1 olduqda, ASCII standartına

daxil

olunmayan əlavə 128 simvolu təsvir etmək mümkündür (digər əlifbaların simvollarını)

Sənədləri müxtəlif dillərdə təsvir etmək üçün hal-hazırda***Unicode*** kodlaşdırma

sistemindən istifadə olunur. Bu sistemdə bir simvolun təsviri üçün 16 bit ardıcıllığı istifadə

olunur, yəni bu sistemdə 216=65536 sayda simvolu təsvir etmək olar (çin və

 yapon

dillərinin simvollarını, digər əlifbaları və s.).

Digər kodlaşdırma sistemi***ISO*** (*International Organization for Standardization* –

Beynəlxalq Standartlaşdırma Təşkilatı) tərəfindən yaradılmışdır. Bu sistemdə hər simvolun

təsviri üçün 32 bitdən istifadə olunur, yəni bu kodlaşdırma vasitəsi ilə milyardlarla

 (232)simvolu təsvir etmək mümkündür.

Əgər əlifba *N simvoldan* ibarətdirsə *(gücü N-ə bərabərdirsə), L uzunluqlu* müxtəlif məlumatların sayı : *Q = Nl*

*Əlifba 256 simvoldan ibarətdirsə, 10 səhifəlik mətndə informasiyanın miqdarını təyin edin (hər səhifədə 32 sətir və hər sətirdə 64 simvol var).*

1)Simvolun informasiya həcmi :

256 = 28\**i*= 8 bit= 1 bayt

2)Səhifədə simvolların sayı :

32·64 = 25·26= 211

3)Simvolların ümumi sayı :

*L*= 10·211

4)Məlumatın informasiya həcmi :

*I*= *L*·*i*= 10·211·1 bayt= 20 Кbayt

**Diskret (rəqəmli) siqnallar**

**Diskret siqnal**—qiymətlər ardıcıllığıdır. Hər bir qiymət sonlu çoxluğa daxildir.

*Xassələr*:

•Siqnal zamanın müəyyən anlarında təyin olunur(*zamana görə diskretləşdirmə*);

•Siqnal bir neçə mümkün olan qiymət ala bilər (*səviyyəyə görə diskretləşdirmə*

Əlifba-rəqəm informasiyasının kodlaşdırılması prinsip etibarı ilə 1 baytlıq registr əsasında problemsiz təsvir oluna bilər. Buna səbəb belə mətn simvollarının sayının 256 –dan çox olmamasıdır. Əgər klaviaturanın böyük və kiçik hərflərini, rəqəmləri, digər simvolları nəzərə alsaq, belə təsvir elementlərinin sayının 256-dan az olduğu qənaətinə asınlıqla gəlmək olur.

Burada məsələni qismən mürəkkəbləşdirən klaviaturaların milli mahiyyət daşıması və yazı şriftlərində bu arqumentin nəzərə alınmalı olduğu ilə bağlıdır. Bu sahədə kodlaşdırma prinsip etibarı ilə çox sadə həll olunan məsələdir. Simvollar ardıcıl olaraq unikal nömrələrlə təchiz olunur və onların nömrələr cədvəli tərtib olunur.

Kompüter texnikası və informasiya texnologiyaları ilə bağlı başqa sahələrdə standartlaşdırma məqsədi ilə əlifba-rəqəm informasiyasının ümumi kodlaşdırma sistemləri yaradılmışdır. Bunlar içərisində **ASCII** (**American Standard Code for Information Interchange** – informasiya mübadiləsi üçün Amerika standart) kodaşdırma sistemini qeyd etmək lazımdır.

Bu sistem 7-bit ölçüyə hesablanmış olduğundan belə registrdə 128 simvolun təsvirinə imkan varır. Bu sistemdə 0-dən 1F-ə qədər olan kodlar (16-lıq say sistemində) idarəedici simvollar olub, çap olunan simvollara aid deyildirlər. 128-dən 255-ə qədər olan kodlar ASCII-in kodlar siyahısında istifadə olunmurlar. Bu kodlar kompüterdə istifadə olunan digər məqsədlər üçün lazımdır.

Bu kodlar bir çox informasiya mübadilə sistemlərində informasiya massivlərinin başlıqlarının qeyd olunması üçün xüsusi simvol olaraq istifadə olunurlar.

**Unicode**

Kompüter sənayesi əsasən Amerika Birləşmiş Ştatlarında inkişaf etdiyindən kodlaşdırma da əsas etibarı ilə ingilis dilinə uyğunlaşdırılmışdır. Lakin kompüterin ümumdünya əhəmiyyəti daşıması və kompüterlərin istehsal və istehlak ərazilərinin bütün dünya olduğu səbəbindən həm klaviaturanın, həm də kodlaşdırma sisteminin uyğunlaşdırılması bir problem kimi diqqəti cəlb etmişdir. İlk təşəbbüs olaraq **Latin-1** kodlama sistemi meydana çıxır ki, bu kodlaşdırma 8-dərəcədən istifadə etməyə başlayır. Bu ona görə zəruri olur ki, lazım olan şriftlərin siyahısınə daha 128 simvol əlavə edilir. Buraya latın hərfləri əlavə olunur.

Növbəti cəhd **IS 8859** stanartı sayıla bilər ki, buraya **kod səhifəsi** anlayışı daxil edilməli olur. Kod səhifəsi dedkdə 256 simvoldan ibarət toplam başa düşülür. Bu simvollar 1,2,...256 –ardıcıllığınan ibarətdir. Məsələn 1-simvolu IS 8859-dan sonra yazılır və o **Latin**-1 qrupunu müəyyən edir. **IS 8859-2** standartı slavyan qrupunu, IS 8859-3 iə türkiyə, malta qruplarını əks etdirir. Lakin belə kodlaşdırma sisteminin o mənfi cəhəti vardır ki, kompüter təmamilə dil qrupundan asılı vəzyyətə düşür. Proqram təminatı özünü doğrultmayan dərəcədə mürəkkəbləşmiş olur. Digər tərəfdən bu sistem çin və Yapon ieroqifləri ilə işi heç cürə təmin edə bilmir.

Bir sıra kompüter şirkətləri qrupu yeni **Unicode**sistemi yaratmağa nail ola bildilər ki, sonradan onu beynəlxal standart kimi elan edə bildilər. Bu kodlaşdırma sistemin əsas cəhəti dərəcələrin sayının 16-ya çatdırılması ilə kodlaşdırma obyktlərinin sayının olduqca çoxaldılmasıdır. Hazıırda bu kodlaşdırma sistemi get-gedə daha çox milli əlifbaları əhatə etmiş olur.

**Mühazirə 4**

Müasir kompyuterlərdə mətnlərdən və ədədlərdən əlavə qrafiki, səsli və

video informasiya saxlanılır. Bu cür informasiya***multimediya*** adlanır.

Kompyuterdə təsvir üsullarından asılı olaraq bütün qrafiki obyektləri 2 qrupa bölmək

olar:***rastr*** (*bitmap techniques*) və***vektor təsvirlər*** (*vector techniques*).

Rastr təsviri***piksel*** (pixel) adlanan nöqtələr yığımından təşkil olunur. Yaddaşda ağ-

qara rastr təsvirləri uzun bitlər ardıcıllığı ilə təşkil olunurlar. Pikselın ağ və ya

 qara olmağından asılı olaraq ardıcıllıqda hər bir bit 0 və ya 1 qiymətini ala bilir. Rənqli

təsvirlər daha mürəkkəbdir, çünki hər bir pikselın təsviri üçün onun rəngini təyin edən

bitlər ardıcıllığından istifadə olunur.

Kompyuterin müasir xarici qurğuları, məsələn, faks, videokamera, skaner təsvirləri

rastr formatına çevirir. Bu qurğular pikselın rəngini 3 əsas rəngin kombinasiyası kimi təsvir

edirlər: qırmızı, yaşıl və göy. Hər bir rəngin intensivliyini göstərmək məqsədilə

1 baytdan istifadə olunur, yəni 1 pikselın saxlanılması üçün 3 baytdan istifadə olunur. Kompyuterin

monitorlarında da informasiya bu cür təsvir olunur. Monitorun ekranı

 milyardlarla

piksellərdən ibarətdir. Hər bir piksel 3 rəngin qarışıgıdır: qırmızı, yaşıl və göy.

Əgər hər hansı bir təsvir 1024 pikselı olan 1024 sətrdən təşkil olunursa və hər piksel

3 bayt yer tutursa, onda hesablayıb demək olar ki, rastr formatında bu təsvirin ölçüsü bir-

neçə meqabayt (1024\*1024\*3 bayt) təşkil edəcək. Təsvirlərin həcmini müəyyən qədər

azaltmaq məqsədilə kompyuterdə onların sıxılmış formatı istifadə olunur, məsələn, GIF,

JPEG, JPG.

Rastr formatının bir çatışmayan cəhəti var: şəkilləri istənilən ölçüdə böyütmək olmur.

Şəkili böyütmək üçün yeganə bir yol var  o da piksellərin böyüdülməsidir.

Piksellər böyüdükçə, şəkil öz dəqiqliyini itirir.

Digər vektor formatı bu qüsurdan azaddır. Vektor təsvirində şəkillər düz xətt və qövslər vasitəsi ilə çəkilir.

Müasir printerlərdə və monitorlarda istifadə olunan müxtəlif şriftlər (fonts)

vektor

formatında kodlaşdırılıb. Həmin şriftlərdə istifadə olunan simvolların ölçülərini dəyişəndə,

onların təsviri korlanmır. Bu cür şriftlər***miqyaslanan şriftlər*** adlanır. Məsələn, TrueType

və PostScript şriftləri miqyaslanan şriftlərdir.

Vektor formatı avtomatik layihələndirmə sistemlərində istifadə olunur,

məsələn,

AutoCAD, ArchiCAD. Vektor formatı 3 ölçülü obyektləri yaratmağa və onları

monitorda

idarə etməyə imkan verir. Lakin, rastr formatından fərqli olaraq, vektor formatı

şəkillərin

fotoqrafik keyfiyyətini təmin etmir. Məhz buna görə, müasir rəqəmli kameralarda

rastr formatından istifadə olunur.

Rastr formatından istifadə edən proqramlara misal olaraq Adobe Photoshop, Paint,

Corel Photopaint proqramlarını, vektor formatından istifadə edən proqramlara misal olaraq

CorelDRAW, Adobe Illustrator proqramlarını göstərmək olar.

Kompyuterdə təsvir üsullarından asılı olaraq bütün qrafiki obyektləri 2 qrupa bölməkolar:***rastr*** (*bitmap techniques*) və***vektor təsvirlər*** (*vector techniques*).Rastr təsviri***piksel*** (pixel) adlanan nöqtələr yığımından təşkil olunur.

Yaddaşda ağqara rastr təsvirləri uzun bitlər ardıcıllığı ilə təşkil olunurlar. Pikselın ağ və ya qara olmağından asılı olaraq ardıcıllıqda hər bir bit 0 və ya 1 qiymətini ala bilir.

Rənqli təsvirlər daha mürəkkəbdir, çünki hər bir pikselın təsviri üçün onun rəngini təyin edən bitlər ardıcıllığından istifadə olunur.

Kompyuterin müasir xarici qurğuları, məsələn, faks, videokamera,

skaner təsvirləri.

Bu qurğular pikselın rəngini 3 əsas rəngin kombinasiyası kimi təsvir

edirlər: qırmızı, yaşıl və göy. Hər bir rəngin intensivliyini göstərmək məqsədilə

 1 baytdan

istifadə olunur, yəni 1 pikselın saxlanılması üçün 3 baytdan istifadə olunur. Kompyuterin

monitorlarında da informasiya bu cür təsvir olunur. Monitorun ekranı

milyardlarlapiksellərdən ibarətdir. Hər bir piksel 3 rəngin qarışıgıdır: qırmızı, yaşıl və göy.

Əgər hər hansı bir təsvir 1024 pikselı olan 1024 sətrdən təşkil olunursa və hər

 piksel

3 bayt yer tutursa, onda hesablayıb demək olar ki, rastr formatında bu təsvirin

ölçüsü bir neçə meqabayt (1024\*1024\*3 bayt) təşkil edəcək. Təsvirlərin həcmini müəyyən qədər

azaltmaq məqsədilə kompyuterdə onların sıxılmış formatı istifadə olunur,

məsələn, GIF, JPEG, JPG.

Rastr formatının bir çatışmayan cəhəti var: şəkilləri istənilən ölçüdə böyütmək

 olmur.

Şəkili böyütmək üçün yeganə bir yol var – o da piksellərin böyüdülməsidir. Piksellər böyüdükçə, şəkil öz dəqiqliyini itirir.

Digər vektor formatı bu qüsurdan azaddır. Vektor təsvirində şəkillər düz xətt və qövslər vasitəsi ilə çəkilir.

Müasir printerlərdə və monitorlarda istifadə olunan müxtəlif şriftlər (fonts)

vektor formatında kodlaşdırılıb. Həmin şriftlərdə istifadə olunan simvolların

ölçülərini dəyişəndə,

onların təsviri korlanmır. Bu cür şriftlər***miqyaslanan şriftlər*** adlanır. Məsələn,

TrueType

və  PostScript şriftləri miqyaslanan şriftlərdir.

**Şəkillərin təsviri. Səsin təsviri.**

**Rastr təsvir.** Bu termin optikadan götürülmüş və sonralar televiziya texnikasının yaranması ilə əlaqədar geniş istifadə olunmuşdur. Kompüterin funksional məsələlərindən biri şəkillərin emalı və təsviri olduğundan bu termin tezliklə kompüter texnikasında mühüm anlayışlardan birinə çevrilmişdir.

Rastr təsvir bəzən “**bitmap**” və ya “**pixmap**” kimi də texniki ədəbiyyatda istifadə olunur. Mənası damalı sahədə bit informasiyanın və ya piksellərin yaratdığı təsvir kimi izah olunur.

Məlumdur ki, rəng çalarlarının əldə olunması RGB sistemində müvafiq rənglərin (**qırmızı, yaşıl, göy**) parlaqlıqlarının müxtəlifliyi ilə əldə edilir. Rastr qrafikasında həmin parlaqlıqları əks etdirən ədədlərdən ibarət matrisin tərtib olunması və ekrana çıxarılmasından bəhs olunur. Bu səbəbdən təsvirinin miqyasının artırılması ekranda təsvirin keyfiyyətini heç cürə yaxşılaşdıra bilmir.

Lakin rastr qrafikası texniki olaraq geniş yayılmış olduğundan ondan istifadə əhəmiyyət daşıyır.

Rastr qrafikasının keyfiyyətini xarakterizə edən aşağıdakı göstəriciləri qeyd etmək olar:

1.***Rənglərin dərinliyi.***Bu göstərici kimi təsvirdə bir pikselin təsviri üçün hər üç rəng daxil olmaqla necə bit informasiyadan istifadə olunduğu qəbul edilmişdir. Bu ədəddən asılı olaraq ümumən neçə rəng çalarının mövcud olduğu asaanlıqla hesablana bilər. Fərz edək ki, hər rəng üçün parlaqlıq intervalını, yəni intervalı  diskret addım ilə dəyişdirilir. Hər addıma uyğun bir rəng çaları əmələ gəldiyi üçün və rənglərin sayı üç olduğundan variantların ümumi sayının olduğu aşkardır.

2. ***Təsvirdə mövcud olan ayırd etmə qabiliyyəti*** . Bu göstərici vahid səthə, vəya vahid uzunluğa düşən piksellərin sayı ilə müəyyən olunur. Yuxarıda qeyd etdik ki, bu kəmiyyət təsvirdə detallaşdırmanın imkan dairəsini müəyyən edir. Əlbəttə, məlum həqiqətdir ki, belə detallaşdırma vahid səthə düşən piksellərin sayının artmasını tələb etdiyindən təsvirin keyfiyyəti ilə faylın ölçüləri arasında kompromiss münasibət həll edilməlidir.

Rastr təsvirlərin müsbət və mənfi cəhətləri müəyyən edilmişdir.

**Müsbət cəhətlər**:

-Rəng çalarlarının genişliyi;

- Rənglər keçidlərinin dəqiq verilə bilməsi;

- Kiçik detalların əks etdirilməsi üçün prinsipial maneələrin olmaması.

**Mənfi cəhətlər**:

-Təsvirin kiçildilməsi ilə əks etdirilə bilən piksellərin sayı azalır, yəni kiçik detallar ayırd oluna bilmir;

- Təsvirin böyüdülməsi ilə əlaqədar olaraq əlavə piksellərin yaranmadığı səbəbindən təsvir pikselləşmiş görüntü verir;

- Təsvirin ayırd etmə qabiliyyətinin və rəng çaları çoxluğunun artırılması yalnız faylın ölçülərinin artırılması hesabına əldə edilə bilir.

**Vektor təsvir**

Rastr qrafikasından prinsipial olaraq fərqlənən vektor qrafikasının aşkar üstünlükləri vardır. Qrafik təsvir obyektinin ekran görüntüsü miqyasının istənilən qədər böyüdülməsi obyektin keyfiyyətinə mənfi təsir göstərmir. Bunun səbəbini təsvirin sanki “**həndəsiləşdirilməsi**”ndə axtarmaq lazımdır. Təsvir elementar həndəsi fiqurların (**həndəsi primitivlərin**) istifadəsi əsasında **modelləşdirilir**. Belə sadə **primitivlər** içərisində çevrə, ellips, müxtəlif qalınlığa və parametrlərə malik xətt və b. həndəsi fiqurları görmək olar. Bu fiqurların əsas parametrləri, məs. çevrənin mərkəzinin koordinatları, radiusu , sektor bucağı yaddaşa köçürülür. Məhz buna görə də miqyası dəyişmək heç bir çətinlik törətmir. Miqyas dəyişdirildikdə qeyd olunan parametrlər müvafiq surətdə bəyişdirilir. Elementar **primitivlərin** koordinatlarının və digər parametrlərinin eyni miqyasa dəyişdirilməsindən təsvirin özünəməxsus görüntüsü dəyişməz olaraq qalır.

Ümmiyyətlə AI vektor formatı “Adobe” şirkətinə məxsus qrafik redaktor alətlər kompleksidir.

Bütün qrafik redaktorlarda oldüğü kimi vektor qrafikasında da **“təbəqə”** anlayışı geniş istifadə olunur. Təbəqələr fərdi surətdə miqyaslana bilirlər. Son nəticədə təbəqə təsvirləri toplanma əməliyyatını məruz qalaraq, təsviri ifadə edirlər.

**Səsin qeyd olunması (təsviri).**

**a) Səsin təbiəti və əsas parametrləri**

Səs təbiətdə mövcud olan minlərlə rəqsi proseslərdən biridir. Səsin dalğa təbiətli olması həm onun harmonik xassəsi ilə, həm də onun yayıldığı mühitin elastik xassəli olması ilə əlaqədardır. Sadə halda səs mənbəyi titrəyişə məruz qalan cisim,elastik xassə daşıyan hava isə onun yayıldığı mühit nəzərdə tutulur.

Təbiidir ki, səs təzyiqi bir fiziki kəmiyyət kimi qabaqcadan çevrilərək elektrik analoq formasına gətirilir.

Səs müxtəlif elastik mühitdə müxtəlif sürətlə yayılır. Havada yayılma sürəti 340-360 dəyişir. Bütün harmonik rəqsi proseslər kimi səs aşağıdakı ifadə ilə riyazi təsvir olunur:

 (1)

harada ki, səs mənbəyinin yerləşdiyi nöqtədə havanın təzyiqinin dəyişmə qanunu;amplituda;səsin tezliyi;zamandır.

Səs müxtəlif tezliklə və amplitudlu harmonik siqnalların toplanmasından meydana çıxan mürəkkəb analoq siqnalıdır.

ŞSəs siqnalı nə qədər mürəkkəb olsa da zamanın verilmiş bir anında səs təzyiqinin yalnız br qiymət aldığı aşkardır (şək.b). Ona görə də səsin rəqəmsallaşdırılması dedikdə iki analoq dəyişənini diskretləşdirməkdən və sonra isə hər diskret qiyməti rəqəmsala çevirməkdən sohbət gedir.

Səsin diskretləşdirilməsinin hansı addımlar üzrə həyata keçirilməsinin səsin rəqəmsallaşdırılma texnikası üçün böyük əhəmiyyəti vardır

Beləliklə səsin qeyd olunmasını həyata keçirən texniki quruluşlarda nəzərə alınması lazım olan parametrlər içərisində aşağıdakıları göstərmək olar:

1. Səsin **tezlik diapazonu**. Bu göstərici daha geniş mənada **spektr** adlandırılır.

2. **Səsin gücü**- yəni səsin amplitudası. Ümumiyyətlə səs sinusoidaların (harmonikaların) cəmindən ibarət olduğu üçün səs siqnalının gücü fziki olaraq amplitudanın kvadratı ilə mütənasib olaraq dəyişir. Lakin, akustikada səsin amplitudasını səsin intensivliyi kimi daha çox istifadə edirlər.

**b)*Səsin rəqəmsallaşdırılma texnologiyası***

Səsin qeyd olunmasında iki mərhələ diqqəti cəlb edir:

- a) Səs siqnalının zamana görə diskretləşdirilməsi;

- b) Amplituda görə kvantlama.

Siqnalın zamana görə diskterləşdirilməsində “**diskretləşdirmə tezliyi”** əsas rol oynayan parametrdir. Analoq siqnalının sonlu sayda zaman anlarında aldığı qiymətlər oxunub-yaddaşa köçürülür. Bu zamana görə dskretləşdirmədir. Vahid zamanda belə “oxunub-yadda saxlama” prosesinin sayı zamana görə diskretləşdirmə tezliyi adlanır. Təbii bu parametr nə qədər çox olarsa, diskret siqnal analoq siqnala (təbii siqnala) bir o qədər dəqiq yaxın götürülmüş olacaqdır.

Əgər saniyə ərzində **“oxunub-yadda saxlama**”- dəfə həyata keçirilmişdirsə, diskretləşdirmə tezliyi düsturu ilə hesablanmalı olacaqdır. Bu kəmiyyət [hs; khs; Mhs;Qhs] kimi ölçü vahidləri ilə ifadə olunacaqdırlar.

Ampltudaya görə kvantlaşma analoq siqnalının kod kimi ifadə olunmasıdır. İmpuls texnikasında kvantların sayı amplitudanın dəyişdiyi intervalın bərabər bölgülənməsindən yaranan pillənin hündürlüyü kimi başa düşülür:



harada ki, uyğun olaraq amplituda intervalının minimal, maksimal qiymətləri, kvantın qiyməti, kvantların sayıdır.

Ümumiyyətlə, əgər səsin qeyd olunmasında zəruri olan “oxunma- yadda saxlama” kimi ilkin mərhələ nəzərə alınmazsa, səsin rəqəmsallaşdırılma prosesi ARÇ (analoq-rəqəm çeviricisi) –də istifadə olunan alqoritmlərdən ibarət olduğu aşkar görünür.

Burada siqnalın səsdən ibarət olması bir sıra əlavə mərhələlər meydana çıxarır ki, onların içərisində aşağıdakıları göstərmək lazımdır:

- səsin öncə süzgəclənməsi,

- hər diskret zaman anında əldə olunmuş siqnalın koda çevrilmə alqoritmi,

- səsin qeyd olunması üçün nəzərdə tutulan “izafi dərəcəlilik “ və ondan istifadə olunaraq səhv kodların təshih olunması,

- oxunma prosesində dekodaşdırma funksiyasının yerinə yetirilməsi.

Qeyd edək ki, göstərilən mərhələlərdə yalnız **kanal kodlaşdırılması** (bu ad “yazma-oxuma aparat vasitələrində informasiyanın daşınma kanallarından istifadə ilə əlaqədar olaraq verilmişdir) sxemdə öz əksini tapmamışdır. Həmin funksiya informasiya daşıyıcısına “yazma” mərhələsindən sonra (yuxarıdakı şəklə nəzər et!), “oxunma” mərhələsinən isə əvvəl yerinə yetirilməlidir.

Səsin qeyd olunmasında bərpaedici kodlaşdırmanın bu vəya digər alqoritminin istifadə olunması faktiki olaraq səs fomatının müəyyən edir.

**Mühazirə 5**

Müasir kompyuterlər çox geniş müxtəlifliyə malikdir. Lakin onların strukturu aşağıdakı

əsas qurğuları fərqləndirməyə imkan verən ümumi məntiqi prinsiplərə əsaslanır.

* nömrələnmiş xanalardan ibarət olan yaddaş (yaddaş qurğusu, YQ);
* idarəetmə qurğusu (İQ) və hesab-məntiq qurğusundan (HMQ) ibarət olan
* prosessor;
* giriş qurğusu;
* çıxış qurğusu.

Bu qurğular informasiyanın ötürüldüyü rabitə kanalları ilə birləşmişlər.

Yaddaşın funksiyaları:

 digər qurğulardan informasiyanın qəbul edilməsi;

 informasiyanın yadda saxlanması;

 maşının digər qurğularına sorğu əsasında informasiyanın ötürülməsi.

Prosessorun funksiyaları:

 hesab və məntiq əməllərinin yerinə yetirilməsi yolu ilə verilmiş proqrama

görə

verilənlərin emal edilməsi;

 kompyuter qurğularının işinin proqramla idarə olunması.

Prosessorun əmrləri yerinə yetirən hissəsi***hesab-məntiq qurğusu***, qurğuların idarə

edilməsi funksiyalarını yerinə yetirən digər hissəsi isə***idarəetmə qurğusu*** adlanır.

Adətən bu iki qurğu şərti olaraq fərqləndirilir, konstruktiv olaraq onlar bölünmürlər.

Prosessorun tərkibində***registr*** adlanan xüsusi əlavə yaddaş xanaları vardır.

Registr ədədin və ya əmrin qısa müddətli yadda saxlanması funksiyasını yerinə

yetirir. Xüsusi elektron sxemləri bəzi registrlərin içindəkilərin üzərində

bəzi

manipulyasiyaları yerinə yetirə bilərlər. Məsələn, sonrakı istifadə üçün əmrin ayrı-ayrı

hissələrini «kəsmək» və ya ədədlər üzərində müəyyən hesab əməllərini yerinə yetirmək.

Registrin əsas elementi bir dənə ikilik ədədi (ikilik kodun mərtəbəsini)

 yadda

saxlamaq qabiliyyətinə malik***trigger*** adlanan elektron sxemidir.

Registr ümumi idarəetmə sistemi vasitəsilə bir-biri ilə müəyyən qayda ilə birləşmiş

triggerlərin yığımıdır.

Yerinə yetirilən əməliyyatların növünə görə bir-birindən fərqlənən bir neçə registr

növü mövcuddur. Bəzi mühüm registrlərin öz adları var, məsələn:

**cəmləyici** – HMQ-nın hər əməlin yerinə yetirilməsində iştirak edən registri;

**əmrlər sayğacı** – içindəkilər növbəti yerinə yetirilən əmrin ünvanına uyğun

gələn İQ registri; ardıcıl yaddaş xanalarından proqramın avtomatik

seçilib

götürülməsi vəzifəsini yerinə yetirir;

**əmrlər registri** – əmrin yerinə yetirilməsi üçün zəruri olan bir müddət ərzində

əmrin kodunu yadda saxlayan İQ registri. Onun mərtəbələrinin bir

 hissəsi

**əməliyyatın kodunu**, digər hissəsi isə**operandların ünvanlarının kodlarını** yadda

saxlamaq üçün istifadə olunur.

Kompyuterlərin böyük əksəriyyətinin qurulmasının əsasını 1945-ci ildə amerika alimi

Con fon Neymanın ifadə etdiyi aşağıdakı ümumi prinsiplər təşkil edir.

1.*Proqramla idarə edilmə prinsipi*. Bundan belə nəticə çıxır ki,**proqram** prosessorun

avtomatik olaraq bir-birinin ardınca müəyyən ardıcıllıqla yerinə yetirdiyi əmrlər yığımından

ibarətdir.

Proqramın yaddaşdan seçilib götürülməsi*əmrlər sayğacı* vasitəsilə həyata keçirilir.

Prosessorun bu registri onda saxlanılan növbəti əmrin ünvanını ardıcıl olaraq əmrin

uzunluğu qədər artırır.

Proqramın əmrləri yaddaşda bir-birinin ardınca yerləşdiyindən, ardıcıl yerləşən

yaddaş xanalarından əmrlər zəncirinin seçilib götürülməsi təşkil edilir.

Əgər əmr yerinə yetiriləndən sonra sonrakıya deyil, hər hansı digər əmrə keçmək

lazımdırsa, sonrakı əmrin yerləşdiyi yaddaş xanasının nömrəsini əmrlər sayğacına yazan

*şərti* və ya*şərtsiz keçid* əmrlərindən istifadə olunur. Yaddaşdan əmrlərin seçilib

götürülməsi «STOP» əmri gələndən və yerinə yetiriləndən sonra dayandırılır.

Beləliklə, prosessor proqramı insanın müdaxiləsi olmadan avtomatik yerinə yetirir.

2.*Yaddaşın bircinsliyi prinsipi*. Proqramlar və verilənlər eyni yaddaşda yerləşir. Ona

görə də kompyuter yaddaşın verilən xanasında nəyin – ədəd, mətn və ya əmrin

saxlanmasını fərqləndirmir. Əmrlər üzərində verilənlər üzərində olan əməliyyatları yerinə

yetirmək olar. Bu isə bir sıra imkanlar açır. Məsələn, proqram özünün yerinə yetirilmə

prosesində yenidən işlənməyə məruz qala bilər ki, bu da proqramın özündə onun bəzi

hissələrini əldə etmək qaydalarını verməyə (bu yolla proqramda dövrlərin və

altproqramların yerinə yetirilməsi təşkil olunur) imkan yaradır. Bundan əlavə, eyni

bir

proqramın əmrləri digər proqramın yerinə yetirilməsi nəticələri kimi əldə oluna

 bilər.

Yüksək səviyyəli proqramlaşdırma dilindən konkret maşın dilinə proqramın mətninin

translyasiya – tərcümə metodları bu prinsipə əsaslanır.

3.*Ünvanlıq prinsipi*. Struktur olaraq əsas yaddaş nömrələnmiş xanalardan ibarətdir;

prosessor ixtiyari zaman anında istənilən xanaya müraciət edə bilir. Buradan yaddaşın

hissələrinə ad vermək imkanı meydana çıxır ki, onlarda yadda saxlanılan qiymətlərə

sonradan müraciət etmək və ya onları mənsub edilmiş adlardan istifadə etməklə

proqramın yerinə yetirilməsi prosesində dəyişdirmək mümkün olsun.

Bu prinsiplər üzərində qurulan kompyuterlər***fon-neyman tipinə*** aid edilir. Lakin, fon-

neyman tipindən prinsipial fərqlənən kompyuterlər də mövcuddur. Onlar üçün,

məsələn,proqramla idarə edilmə prinsipi yerinə yetirilməyə bilər, yəni onlar cari yerinə yetirilən proqram əmrini göstərən «əmrlər sayğacı» olmadan işləyə bilərlər. Yaddaşda saxlanılan

hər hansı dəyişənə müraciət etmək üçün bu kompyuterlərdə dəyişənə ad vermək

vacib deyil. Belə kompyuterlər***qeyri-fon-neyman tipli kompyuterlər*** adlanırlar.

Kompyuter qurğularına baxdıqda onların arxitekturası və strukturu

 bir-birindən fərqlənir.

*Kompyuterin* *arxitekturası* istifadəçinin proqramlaşdırma imkanlarının, əmrlər

sisteminin, ünvanlanma sisteminin, yaddaşın təşkilinin və s. təsviri daxil olmaqla

onun hər

hansı ümumi səviyyədə təsvirinə deyilir. Arxitektura kompyuterin əsas məntiqi

 hissələrinin

– prosessor, əməli YQ, xarici YQ və periferiya qurğularının iş prinsipini, informasiya

əlaqələrini və qarşılıqlı birləşməsini müəyyən edir. Müxtəlif kompyuterlərin

 arxitekturasının

ümumiliyi istifadəçi nöqteyi-nəzərindən onların uyğunlaşmasını təmin edir.

*Kompyuterin strukturu* onun funksional elementlərinin və onlar arasında

 olan

əlaqələrin yığımıdır. Kompyuterin hissələri ən sadə sxemlərdən ən müxtəlif

qurğulara

qədər ola bilər. Kompyuterin strukturu qrafiki olaraq struktur sxemlər şəklində təsvir edilir.

Bu sxemlər vasitəsilə hər hansı detallaşdırma səviyyəsində kompyuterin təsvirini

vermək olar.

Ən çox yayılmış aşağıdakı arxitekturalardır:

 **Klassik arxitektura** (fon Neyman arxitekturası) – verilənlər axınının

keçdiyi bir HMQ və əmrlər axınının (proqramın) keçdiyi bir İQ. Bu,*birprosessorlu*

*kompyuterdir*.*Ümumi şinli* fərdi kompyuterin arxitekturası da bu tip arxitekturaya

aiddir. Burada bütün funksional bloklar öz aralarında*sistem magistralı* adlanan

ümumi şinlə əlaqələndirilirlər.

Fiziki olaraq magistral elektron sxemlərinin qoşulması üçün yuvalara malik olan çox

naqilli xətdir. Magistralın naqilləri ayrı-ayrı qruplara bölünür: ünvan şini, verilənlər şini və

idarəetmə şini.

Periferiya qurğuları (printer və s.) xüsusi*kontrollerlər* – periferiya qurğularını idarə

edən qurğular vasitəsilə kompyuterə qoşulur.

*Kontroller* prosessoru periferiya avadanlığının fəaliyyətini bilavasitə idarə etməkdən

azad edərək verilən avadanlığı və ya rabitə kanallarını mərkəzi prosessorla əlaqələndirən

qurğudur.

 **Çoxprosessorlu arxitektura**. Kompyuterdə bir neçə prosessorun

olması o deməkdir ki, paralel olaraq çoxlu verilənlər axını və çoxlu əmrlər axını

təşkil edilə bilər. Beləliklə, paralel olaraq bir məsələnin bir neçə fraqmenti yerinə

yetirilə bilər.

**Çoxmaşınlı hesablama sistemi**. Burada hesablama sisteminə daxil

olan bir neçə prosessor ümumi əməli yaddaşa malik olmayıb hər birirnin öz

(lokal) əməli yaddaşı var. Çoxmaşınlı sistemdə hər kompyuter

klassik

arxitekturaya malikdir və belə sistem kifayət qədər geniş tətbiq edilir. Lakin belə

hesablama sisteminin tətbiqindən alınan effekt yalnız olduqca xüsusi struktura

malik məsələləri həll etdikdə əldə oluna bilər. Sistemdə nə qədər kompyuter

varsa, məsələ o qədər sayda zəif əlaqəli altməsələlərə bölünməlidir.

Çoxprosessorlu və çoxmaşınlı hesablama sistemlərinin birprosessorlu sistemlər

qarşısında sürətə görə üstünlüyü aşkardır.

 **Paralel prosessorlu arxitektura**. Burada bir neçə HMQ bir İQ-nun

idarəsi altında işləyir. Bu o deməkdir ki, verilənlər çoxluğu bir proqram ilə – yəni

bir əmrlər axını ilə emal edilə bilər. Belə arxitekturanın yüksək sürətini yalnız o

məsələlərdə almaq olar ki, həmin məsələlərdə eyni hesablama əməliyyatları

müxtəlif eyni tipli verilənlər yığımında eyni vaxtda yerinə yetirilir.

Müasir fərdi kompyuter bir neçə əsas konstruktiv komponentlərdən ibarətdir:

 Sistem bloku;

 Monitor;

 Klaviatura.

Kompyuterin məhsuldarlığını və imkanlarını sistem blokun xarakteristikaları müəyyən

edir.

Sistem blokunda aşağıdakı qurğular yerləşir:

 Qida mənbəyi – power supply;

 Sistem lövhəsi (ana lövhə) – motherboard;

 Sərt maqnit disk (HDD);

 Genişləndirmə lövhələri;

 Optik disk sürücüləri (CD, DVD);

 Digər texniki qurğular.

Sistem blokun əsas elementi sayılan ana lövhəyə praktiki olaraq kompyuterin əsas

elementləri qoşulur və ana lövhə vasitəsilə onlar arasında əlaqə yaranır. Ana lövhənin

əsas xarakteristikalarını onda yerləşən genişlənmə slotları və slotların növləri müəyyən

edir. Genişlənmə slotları dedikdə genişlənmə platası (lövhəsi) adlanan kontroller və ya

adapterlər nəzərdə tutulur. Belə slotlara kompyuterin daxilində yerləşən əlavə qurğular

qoşulur. Əlavə qurğulara videokartı, audiokartı, modemi və digər qurğuları göstərmək olar.

Sistem blokun əsas elementi sayılan ana lövhəyə praktiki olaraq kompyuterin əsas

elementləri qoşulur və ana lövhə vasitəsilə onlar arasında əlaqə yaranır. Ana lövhənin

əsas xarakteristikalarını onda yerləşən genişlənmə slotları və slotların növləri müəyyən

edir. Genişlənmə slotları dedikdə genişlənmə platası (lövhəsi) adlanan kontroller və ya

adapterlər nəzərdə tutulur. Belə slotlara kompyuterin daxilində yerləşən əlavə qurğular

qoşulur. Əlavə qurğulara videokartı, audiokartı, modemi və digər qurğuları göstərmək olar.

Məlumdur ki, sistem blokun gövdəsinə çoxlu sayda elementlər dəsti qoşulur və bu

elementlər dəsti iş prosesində özlərindən müəyyən temperatura malik istiliyi ətrafa

yayırlar. Buna görə də sistem blokunun daxilində normal temperatur yaratmaq üçün

istehsal olunan müasir kompyuterlər ventilyatorlar ilə (ing. cooler ) təmin edilirlər. Bəzi fərdi

kompyuterlərdə sistem blok daxilinə quraşdırılmış yüksək temperatura malik

istilik

mənbəyi yaradan videokartları və vinçesterləri soyutmaq üçün korpusda əlavə

ventilyatorlardan da istifadə edilir.

Dizayner və konstruksiya baxımından müxtəlif variantlarda istehsal olunan sistem

blokları mövcuddur. Kompyuter istifadəçiləri əsasən desktop, slimline, minitower, midi

tower, big tower, monoblok sistem bloku növlərindən istifadə edirlər.

Kompyuterin qida mənbəyi onun ən vacib və ən çox problemlər

 yaradan

hissələrindən biridir. Həddindən çox qızma, gurultulu səslə işləmə və xoşagəlməz səslər

qida mənbəyinin yaxşı vəziyyətdə olmadığını göstərir. Elektrik cərəyanının tez-tez

kəsilməsi, çox aşağı və çox yüksək göstəriciləri qida mənbəyində meydana gələn

problemlərin başlıca səbəblərindən biridir. Qida mənbəyində bu cür problemlər meydana

gəldiyində onun dəyişdirilməsi məqsədəuyğundur.

**Ana lövhə və ya ana kart** (ing.*motherboard, MB*, bununla bə rabə r ingilisc ə başqa

cür də adlandırılır*mainboard* — əsas lövhə ; slenq - ana ) — mürəkkəb çoxqatlı çap

lövhə sind ə f ə rdi kompyuterin  v ə ya  baza tipli serverin ə sas komponentl ə ri (m ə rkəzi

prosessor, operativ yaddaşın kontrollerlə ri, operativ yaddaş - RAM, yü klə yici ROM, ə sas

və ya baza daxiletm ə və xaricetm ə interfeysl ə rinin kontrolleri) yerl əşdirir. Ana kart onun

üzə rind ə yerləşdirilmiş komponentlər arasında informasiya mübadilə si yaradır və onlar ın

işlərini koordinasiya edir.

1. Prosessor yuvası;

2. Sistem məntiqini realizə edən mikrosxemlər (chipset);

3. Operativ yaddaş mikrosxemlərinin yerləşdi ril məsi üçün slotlar;

4. AGP şin interfeysi ;

5. PCI şin interfeysi;

6. Sərt, CD və DVD tipli İnformasiya daşıyıcılarının ana kart ilə birləşdi rilməsi üçün

istifadə olunan interfeyslər;

7. Daxiletmə və xaricetmə portlarının yerləşdiyi blok;

8. Qida mənbəyinin birləşdi rilməsi üçün istifadə olunan konnektor;

9. Disk daşıyıcılarının birləşdi rilməsi üçün istifadə olunan interfeys.

**Ana kartın struktur sxemi.** Ana kartında mərkəzi prosessor ilə bərabər iki ədəd

körpüdən istifadə olunur. Chipset mikrosxem topluğu (ing. chipset) — bir neçə

mikrosxemdən ibarət olub mərkəzi prosessoru  operativ ya d daş qurğusu və periferik

qurğuların kontrollerləri ilə bağlanmasını təmin edir. Müasir sistem məntiqini təmin edən

chipset, əsasən, «şimal» və «cənub» körpülərindən ibarətdir.

 ***Şimal  körpü*** (ing. Northbridge),  MCH  (Memory  controller  hub),  sistem

kontrolleri — mə rkəzi prosessoru yüksə k m əhsuldarlığa sahib şinlə ri istifadə

ed ən qurğulara (RAM, qrafik kontroller)  birl əşdirmək üçün isti fad ə edilir.

Mə rkə zi prosessoru sistem kontrollerin ə birləşdirmək üçün FSB (Front Side

Bus (FSB) — bu şin x86 dəstəkli mərkəzi prosessor ilə daxili qurğular arasında

bağlantını təmin edir), HyperTransport ( əvvəllər Lightning Data Transport

(LDT) kimi tanınılan  iki istiqam ətli yüksə k buraxma qabiliyyə tin ə və

kiçik

gecikm ə yə malik olan   ardıcıl/paralel kompyuter şin idir.

HyperTransport

Technology konsorsiumu t ə rəfind ən hazırlanmışd ır) və SCİ ( ing. Scalable

Coherent Interface (miqyaslaşdır ılabilə n uzlaşdırılm ış qarşılıqlı ə laqə və ya

interfeys) — komponentlər arasında böyük həcmdə və

qısa mesajların

ötürülməsini təmin edir. Bu mesajlar üçün gecikmə zamanı həll edici rol oynayır)

şinlə rind ə n istifad ə olunur. Adə t ən, sistem kontrollerin ə RAM bağlanır . Bel ə olan

halda o h əm d ə yaddaş  kontrollerinə d ə sahibdir.  T ə tbiq olunan sistem

kontrollerində n operativ yaddaşının maksimal hə cmi h əm də yaddaş şinin

buraxma qabiliyyə tində n asılıdır. Son zamanlar daxilə quraşdırılmış RAM

lardan istifad ə edilm əyə b aşlanmışdır (misal üçün daxilə quraşdırılmış yaddaş

kontrollerin AMD K8 v ə İntel Core i7 tipli prosessorlarda istifadə edilmişdir). Bu

isə öz növbə sind ə sistem kontrollerinin funksiyasını n işini asanlaşdırır və istilik

ayrilmalar ın ın miqdarını azaldır. Müasir ana kartlarda qrafiki kontrollerin

qoşulması üçün PCI Express slotlarından istifadə olunur . Köhnə ana kartlarda

isə ümumi şinlər(İSA, VLB, PCI və AGP) istifad ə edilirdi.

 ***Cənub körpü*** (ing. Southbridge), ICH (I/O controller hub), periferik kontroller —

periferik qurğuların kontrollerini, o cümlə dən ( s ə rt disk, Ethernet, Audio), PCI

və PCI Express şinlə rin ə qoşulan periferik qurğular üçün şin kontrolleri, yüksə k

buraxma qabiliyyə ti t ə lə b etm ə yə n komponentl ər üçün LPC şinin kontrol lerini

özündə birləşdirir. Bununla  bə rabər LPC şini  yüklə yici ROM (BIOS) və

multikontrollerin  bağlanması  üçün  də istifad ə edilir.  Multikontroller

mikrosxeml ə rd ən ibar ət olub m əlumatların ötürülməsi üçün istifadə edilən aşağı

m əhsuldarlığa sahib olan   «kö hnəlmiş» interfeyslə rdir (ardıcıl və paralel

interfeysl ə ri, maus və klaviatura kontrollerl ə ri).

Bir  qayda  olaraq, şimal  və cənub körpüləri ayrı-ayrı  IC(inteqral

mikrosxeml əri) şə klind ə realizə olunur.  T ək chipset şəklində realizə

 olunan

körpülər d ə vardır. Belə liklə , sistem m əntiqini realizə ed ən chipset ana kartın ə sas

xarakteristikalarını və hansı qurğuların bağlanmasını təyin edir.

**Prosessor (CPU)**

***Mə rk ə zi Prosessor*** (*CPU - Central Processing Unit*) - kompyuterin ə n əh əmiyyə tli

parçasıdır. Kompyuter üzə rind ə edilən bütün arifmetik və m ə ntiqi əm əliyyatlar bu qurğu

t ə rəfindən reallaşdırılır və n ə zarə t edilir.

Qeyd edək ki, kompyuterdə prosessorların sayı çox olur. Videokart, səs kartı, çoxlu

sayda xarici qurğular (məsələn, printer) öz prosessorları ilə təchiz olunurlar. Çox vaxt isə,

bu qurğuların prosessorları Mərkəzi Prosessorla məhsuldarlıq nöqteyi-nəzərindən rəqabət

də apara bilirlər. Lakin onların hamısı Mərkəzi Prosessordan fərqli olaraq, dar bir çərçivədə

xüsusiləşdirilmiş olurlar. Onlardan biri səsin emalı ilə, digəri isə 3-ölçülü təsvirlərin

yaradılması ilə məşğul olur. Mərkəzi Prosessorun əsas və fərqləndiriçi xüsusiyyəti – onun

universal olmasıdır.

İstənilən prosessor – xüsusi texnologiya ilə hazırlanan silisium kristalından ibarətdir

və ona görə də bəzən ona “daş” da deyirlər. Lakin bu “daş”in daxilində bir-biriləri ilə

körpülər-kontaktlar vasitəsilə birləşdirilmiş tranzistorlar kimi çoxlu sayda ayrı-ayrı

elementlər olur. Məhz bu elementlər kompyuteri “düşünməyə”, daha doğrusu kompyuterə

daxil olunan ədədlər üzərində hesablama əməliyyatlarını yerinə yetirməyə kömək edirlər.

Əlbəttə ki, bir tranzistor heç bir hesab əməlini apara bilməz. Bu elektron

çevirici

ancaq siqnalı ya özündən keçirə bilər, ya da onu saxlaya bilər. Siqnalın olması məntiqi

vahidə, olmaması isə məntiqi sıfıra uyğun gəlir.

Lakin prosessor sadə tranzistorlar toplusu yox, bir sıra vacib qurğular toplusudur.

Istənilən prosessor kristalında aşağıdakı qurğular yerləşir:

 ***Prosessorun nüvəsi*** və ya əsas hissəsi. Buna*əsas hesablama qurğusu*

deyilir. Məhz burada prosessora daxil olan bütün verilənlər üzərində emal prosesi

yerinə yetirilir;

 ***Soprosessor****-* ən mürəkkəb riyazi hesablamalar, o cümlədən, sürüşən

nöqtəli əməliyyatlar üçün əlavə blokdur. Qrafiki və multimedialı

proqramlarla

işlədikdə soprosessordan istifadə edilir.

 ***Keş-yaddaş*** – bufer yaddaşı olub, verilənlər üçün yığıcı rolunu oynayır.

Prosessor tez-tez istifadə olunan verilənləri əldə etmək üçün ləng işləyən əməli

yaddaşa və sərt diskə müraciət etmək əvəzinə, daxilində quraşdırılan keş-yaddaşa

müraciət edir. Müasir prosessorlarda 3 növ keş-yaddaşdan istifadə olunur: 100

kilobayta qədər tutuma malik olan çox sürətlə işləyən I-ci dərəcəli keş-yaddaş (L1).

Hesablamaların aralıq nəticələrinin saxlanılması üçün istifadə olunur. II-ci dərəcəli

keş-yaddaşın (L2) sürəti bir qədər az olur, lakin tutumu meqabaytlarla ölçülür. III-cü

dərəcəli keş-yaddaş (L3) ən ləng işləyən keşdir, lakin onun həcmi çox böyükdür (24

Mbaytdan çox). L1 və L2-dən sürəti zəif olan L3 yenə də əməli yaddaşdan

sürətlidir. Çoxprosessorlu sistemlərdə ümumi istifadədə olur və L2 keşin

məlumatlarının sinxronlaşdırılması üçün istifadə edilir. Bəzən IV-cü dərəcəli keş

yaddaşdan da istifadə olunur. Bu yaddaş tək bir mikrosxemdə yerləşir və yüksək

məhsuldarlığa malik olan serverlərdə və meinfreymlərdə istifadə olunur. Məhz, keş-

yaddaşın tutumu ilə prosessorların müxtəlif modifikasiyaları bir-birindən fərqlənirlər.

 ***Verilənlər şini*** – informasiya magistralı olub, onun sayəsində prosessor

kompyuterin digər qurğuları ilə verilənlər mübadiləsini apara bilir.

Müasir prosessorlar*mikroprosessor* şəklində hazırlanır. Fiziki olaraq mikroprosessor

inteqral sxemdir – sahəsi 5-6 kvadrat santimetr olan düzbucaqlı formalı nazik kristallı

silisium lövhədir. Bu inteqral sxemdə prosessorun bütün funksiyalarını reallaşdıran

sxemlər yerləşmişdir. Kristal-lövhə, adətən, plastmas və ya keramik müstəvi korpus

içərisinə yerləşdirilir və qızıl naqillərlə metal çıxışlara birləşdirilir ki, onu kompyuterin

ana kartına birləşdirmək mümkün olsun.

***Əməli yaddaş*** (ƏYQ, ingiliscə***RAM***, Random Access Memory — ixtiyari müraciətli

yaddaş) bilavasitə prosessorla əlaqədar olan və yerinə yetirilən proqramların və bu

proqramlarla emal edilən verilənlərin yazılması, oxunması və saxlanması üçün nəzərdə

tutulan çox da böyük həcmli olmayan sürətli yaddaş qurğusudur.

Əməli yaddaş yalnız verilənlərin və proqramların müvəqqəti saxlanması üçün istifadə

olunur, belə ki, kompyuter şəbəkədən açılanda ƏYQ-da yerləşənlərin hamısı silinir. Əməli

yaddaşın elementlərinə müraciət birbaşadır – bu, o deməkdir ki, yaddaşın hər baytı öz

fərdi ünvanına malikdir.

ƏYQ-ın həcmi adətən 512 Mbaytdan 2 Qbayta qədər təşkil edir. Sadə inzibati

məsələlər üçün 512 Mbayt ƏYQ da kifayət edir, lakin kompyuter dizaynının mürəkkəb

məsələləri 1 Qbaytdan 2 Qbayta qədər ƏYQ tələb edə bilər.

Adətən ƏYQ yaddaşın**SDRAM** (sinxron dinamik ƏYQ) inteqral

 mikrosxemləri

üzərində yerinə yetirilir. SDRAM-da hər informasiya biti yarımkeçirici kristalın strukturunda

yaradılmış kiçicik kondensatorun elektrik yükü şəklində yadda saxlanılır. Sızma cərəyanları

səbəbindən belə kondensatorlar tez boşalır, və xüsusi qurğular onları periodik olaraq

(təxminən hər 2 millisaniyədən bir) doldururlar. Bu proses*yaddaşın regenerasiyası*

(Refresh Memory) adlanır. SDRAM mikrosxemləri 512 — 2048 Mbit və daha çox yaddaşa

malikdir. Onlar korpusa quraşdırılır və*yaddaş modullarına* yığılır.

Müasir kompyuterlərin əksəriyyəti **DIMM** (Dual-In-line Memory Module —

mikrosxemləri iki cərgəli yerləşən yaddaş modulu)*tipli modullardan* yığılır.

*Əməli yaddaş* – bu, kompyuter prosessorunun işçi sahəsidir. Əməli yaddaşda

prosessorun əməliyyat apardığı proqram və məlumatlar saxlanılır. Əməli yaddaş sanki

müvəqqəti anbardır, çünki ondakı məlumat və proqramlar yalnız kompyuter işlədikdə

saxlanılır.

Əməli yaddaşı bəzən*ixtiyari müraciətli yaddaş* (**Random Access Memory** -

**RAM**) da adlandırırlar. Bu o deməkdir ki, əməli yaddaşda saxlanılan məlumatlara

müraciət onların (məlumatların) əməli yaddaşda yerləşmə sırasından asılı deyil. Lak in

bu termin bəzi qarışıqlıq salır və yanlış fikirlərə səbəb olur. Məsələ ondadır ki,*yalnız*

*oxunan yaddaş*(**Read-Only Memory** -**ROM**), həmçinin ixtiyari müraciətə malikdir, lakin

"**RAM**"-dan onunla fərqlənir ki, onda yerləşən məlumatlar kompyuter söndürüldükdən

sonra itmir və ona heç nə yazmaq olmaz. Sərt disklərin, həmçinin ixtiyari müraciətli

virtual yaddaş kimi istifadə oluna bilməsinə baxmayaraq, onları**RAM** kateqoriyasına aid

etmirlər.

Müasir fərdi kompyuterlərdə 2 əməli yaddaş tipi istifadə olunur:

 **DRAM** —**Dynamic** **random  access  memory** (Dinamik  İxtiyari  Müraciə tli

Yaddaş);

 **SRAM** —**Static RAM** (Statik əməli yaddaş).

**SIMM** (Single In-line Memory Module) - pinlərin (kontakt) bircərgə üzrə yerləşdiyi

yaddaş moduludur.

***Daimi yaddaş qurğusu*** (ing.*Read Only Memory* –***ROM***) – yalnız oxumaq üçün

istifad ə olunan yaddaşdır. Burada kompyuterin ə sas sistem proqram t əminatı olan BIOS

saxlanılır. Ə m əli yadda şa  nisb ə t ən tutumu  xeyli  az  olan  daimi  yadda ş

informasiyan ın saxlanmas ı v ə ötürülm əsi üçün istifad ə olunmur, y ə ni istifad əçil ə r

t ə r ə find ə n  oraya  informasiya yaz ılmas ı  mümkün  deyil.  İnformasiya  oraya

kompyuter in haz ırlanmas ı zaman ı yaz ıl ır v ə adi hallarda d əyi şdiril ə bilm ə z. Orada

tez-tez istifad ə olunan proqramlar v ə veril ə nl ər saxlan ıl ır (m ə s ə l ə n, ə m ə liyyat

sisteminin b əzi proqamlar ı, kompyuter in düzgün i şl ə m ə sini yoxlayan proqramlar v ə

s.). Daimi yadda ş enerjid ən as ıl ı olmur, y ə ni kompyuter ş ə b ə k ə d ən ç ıxar ıld ıqda

oradakı informasiya saxlanır.

ROM çipləri xüsusi əmrlər saxlayır, bu əmrlər birlikdə yerinə yetirilir. ROM ana plata

üzərində yerləşən kiçik batareyadan qidalanır.

Tərkibində bir və ya bir neçə giriş-çıxış registrləri olan və kompyuterin periferiya

qurğularını mikroprosessorun xarici şinlərinə qoşmağa imkan yaradan hər hansı elektron

sxem**port** adlanır.

Standart interfeys qurğularını: ardıcıl, paralel və oyun portlarını (və ya interfeysləri)

də port adlandırırlar.

*Ardıcıl port* prosessorla verilənləri bayt-bayt, xarici qurğularla isə bit-bit mübadilə edir.

*Paralel port* verilənləri bayt-bayt alır və ötürür.

Sistem blokunun arxa və ön panelində müxtəlif sayda portlar yerləşir. Portlar vasitəsi

ilə kompyuterə müxtəlif periferiya qurğularını birləşdirmək olar.

**Mövzu № 6.**

Müasir kompyuter elmində informasiya bitlər ardıcıllığı ilə təsvir olunur.*Bit,* yəni

ikilik

mərtəbə, iki qiymət ala bilər: 0 və ya 1. Yaddaşda bitlər ardıcıllığı vasitəsi ilə

 hərfləri,

müxtəlif simvolları, təsvirləri və səsləri təsvir etmək olar. Kompyuterdə bitin

saxlanılması

üçün elə bir qurğu lazımdır ki, onun yalnız iki vəziyyəti olsun. Məsələn, elektrik açar

ya

açıq, ya da qapalı vəziyyətdə, rele – açıq və ya bağlı ola bilər və s. Vəziyyətlərin biri – 0,

digəri isə 1-lə təsvir olunur. Müasir kompyuterlərdə bitlərin saxlanılmasına diqqət yetirək.

AND (və), OR (və ya), XOR (kənarlaşdıran və ya) məntiqi əməliyyatlarını

nəzərdən keçirək:

and

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

or

xor

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

not

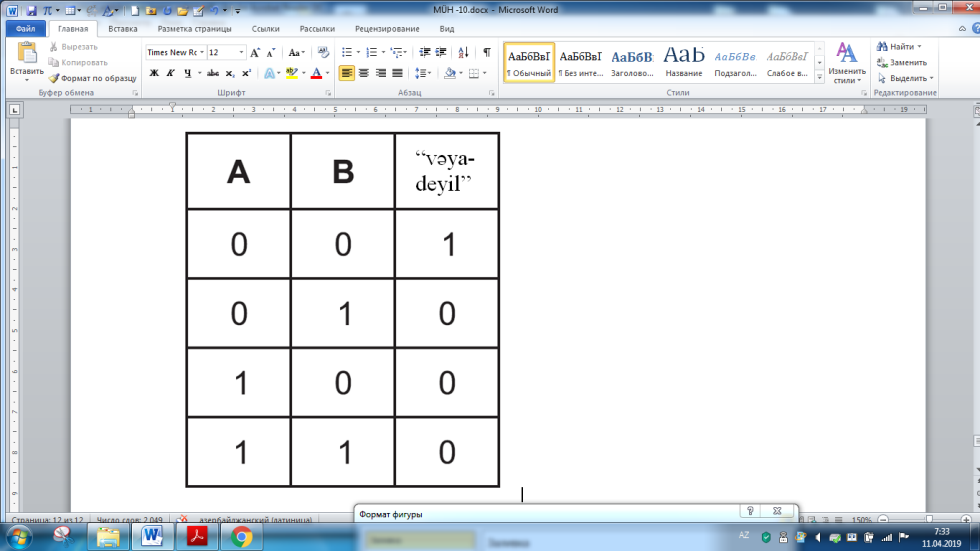
|  |  |
| --- | --- |
| 0 | 1 |
| 1 | 0 |

Kompüterdə müxtəlif funksiyaların yerinə yetirilməsi üçün ventillərdən yaradıılmış elektron quruluşların iş prinsipi ilə tanış olduq. Gördük ki, bu elektron qruluşlar “0” və “1” –lərdən ibarət siqnalları “və”, “vəya”, “və-deyil”, “vəya-deyil” kimi məntiq elementləri vasitsi ilə emal edərək, nəticdə ya “1” , ya da “0” siqnalları hasil edirlər. Funksional blokun dərəcəliliyindən asılı olaraq belə funksional bloklarıın da dərəcəliliyi müxtəlif ola bilir.

Qeyl olunmuşdu ki, Bul cəbri ümumiyyətl məntiqi araşdıırmaların həyata keçirilməsi üçün yaradılmışdır. İngiltərə riyaziyyatçısı Corc Bul (1815–1864) məntiq nəzəriyyəsinin əsasını qoymaqla demək olar ki, hesablama texnikasının inkişafı üçün geniş zəmin yaratmış oldu. Bul cəbri müasir rəqəmsal sistemlərin qurulmasında əsas və demək olar ki, yeganə metoddur. Açarlar və müxtəlif ventillərdən istifadə olunaraq məntiq sxemlərinin yaradılması bu sahədə ümumi istiqamət olduğundan Bul fnksiyası və bul cəbri bir çox hallarda rele funksiyası və rele cəbri kimi də adlandırılır.

**Bul və ya rele sxemləri cəbri** adi cəbrdə olduğu kimi dəyişən (arqment), funksiya (arqumentdən asılı olaraq bu vəya digər qiyməti alan dəyişən) anlayışlarından istifadə edir. Bu cəbrdə hesablama əməllərin müxtəlifliyi vardır. Məsələn, arqumenti **A** ilə işarə etmiş olsaq, və funksiyanı ilə işarə etmiş olsaq, əgər A dəyişəni “0” qiyməti alarsa, yəni A=0 olduqda funksiyası müxtəlif qiymətlər ala bilir.

Məntiq əməllərinin sayı çox deyildir. Məntiqi “**və**”, “**vəya**”, “**və-deyil**”, “**vəya-deyil**” kimi dörd əməliyyatdan ibarət funksiyanı nəzərdən keçirək. Həm də qəbul edək ki, bu əməllərdən istifadə olunurkən arqumentlərin sayını ikiyə bərabər götürəcəyik. Hər əməl yerinə yetirilərkən dörd müxtəlif nəticənin əldə oluna bildiyini və belə əməllərin sayının dörd olduğunu nəzərə alaraq nəticələrin ümumi sayının 16 olacağı tam aşkardır. Bu qiymətlərin (variantların) sayının az olduğu həmin bul cəbrində funksiyanın təsvir vasitəsinin seçilməsinə sadələşdirici təsir göstərmişdir. Belə ki, arqumentlər A,B hərəsi iki qiymət almaqla, cəmi 4 variantda öz əksini tapa bilər.



Əgər adi funksiyanın arqumenti sonsuz sayda müxtəlif qiymət ala bilirsə, məntiq cəbri elementləri, göründüyü kimi cəmi dörd variantda təsvir oluna bilmişdir.Yuxarıda təsvir olunan cədvələ məntiq funksiysının **döğruluq cədvəli** deyilir. İki dəyişən üçün dörd kombinasiyanın olduğundan istifadə edərək onların sıra ilə düzülüşünü aşağıdakı kimi nömrələmək dünyadaqəbul olunmuşdur: -“00:“01”; “10”; “11” ədədləri həmin ardıcılıtı ifadə edir.

Ümumiyyətlə göstərilən funksiyalar ikidən daha artıq arqumentə də malik ola bilirlər. Bu hal nə Bul cəbrinin ümumi müddəalarına, nə də rele sxemləri hesablama qaydalarına zidd deyildir.

**Məntiq sxemlərinin sintezi.**

Qeyd edək ki, çoxsaylı girişləri olan “və” əməli üçün digər bir modifikasiya da nəzərdə tutulmuşdur. Bu “və” əməli çoxluğa görə qiymətləndirmə kimi müəyyən olunmuşdur

Məntiq funksiyalarının yazılışı üçün doğruluq cədvəli forması yeganə üsul deyildir. Verilmiş doğruluq cədvəlinə diqqət yetirməklə görərik ki, məntiq funksiyasının “1” qiymət aldığı 4 hal mövcuddur. Həmin qiymətin əldə edilməsini təmin edən elementar əməliyyatı qeyd etməklə, məntiq funksiyasını ifadə etmək mümkündür.

Aşağıdakı mərhələləri yerinə yetirməklə ixtiyari **doğruluq cədvəlinə** əsasən **məntiq sxemini** qurmaq mümkündür:

1. Hər bir siqnalın inversiyasını almaq üçün sxemə invertorlar artırırıq

2. Doğruluq cədvəlindəki 1 nəticəli sətirlər sayı qədər “Və” ventili çəkirik.

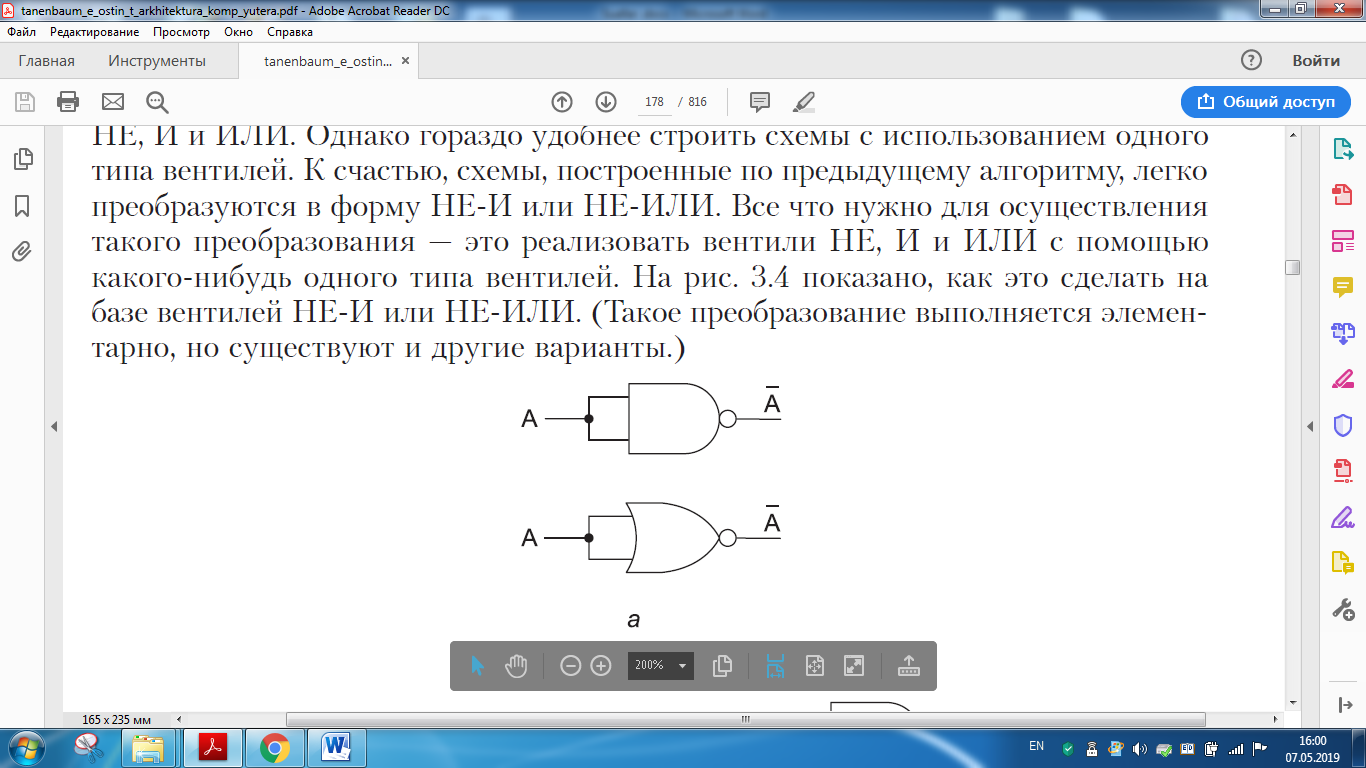
3. “Və” ventili ventillərini müvafiq giriş siqnalları ilə birləşdiririk.

5. Bütün “Və” ventillərinin çıxışlarını “Vəya” ventilinə birləşdiririk.

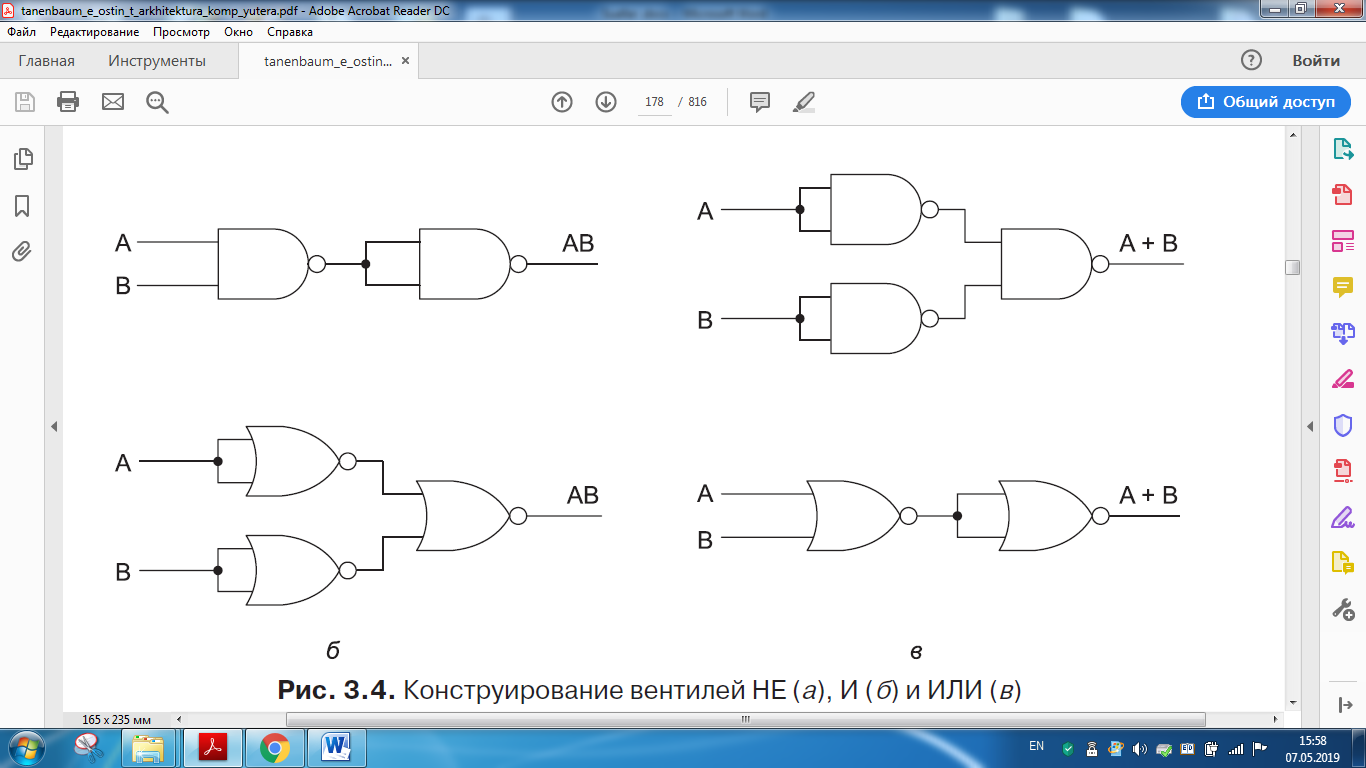
Nümunə üçün b) sxemini nəzərdən keçirək. Həmin sxemdə invertordan, “və” , “vəya” elementlərindən istifadə olunmuşdur. İnvertorun çıxışındakı qiyməti göstərmək üçün dəyişənin üstündə xəttin çəkilməsi qəbul edilmişdir. Bundan əlavə, “və” əməlini məntiqi hasil “**.**” kimi, “vəya” əməlini isə məntiqi toplama “+” kimi işarə edəcəyik. İndi M dəyişəninin “1” qiymətini alması üçün “vəya” elementinin dörd girişindən heç olmasa birində “1” –in olması zəruri oluğunu nəzərdə tutaraq və bu girişlərdən hər birinin dörd ədəd “və” ventilindən əmələ gəldiyinə diqqət yetirərək aşağıdakı məntiq funksiyasını yaza bilərik:



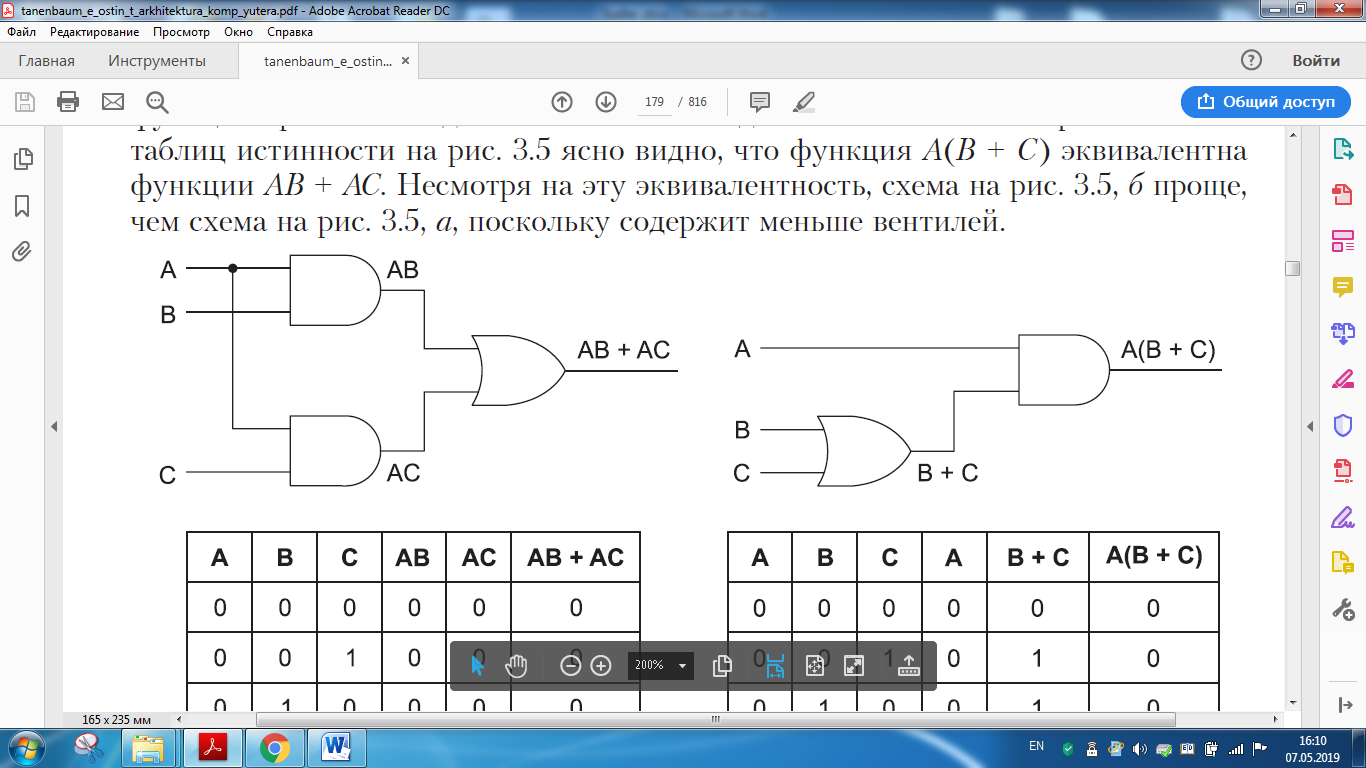
Məntiq sxemlərini “və”, “vəya” “və-deyil”, “vəya-deyil” ventillərindən **yalnız**birinin istifadəsi ilədəqurmaqmümkündür.Aşağıdainkarventiliolan “**deyil**”–iəldəedən2 variandabirləşməgöstərilmişdir:

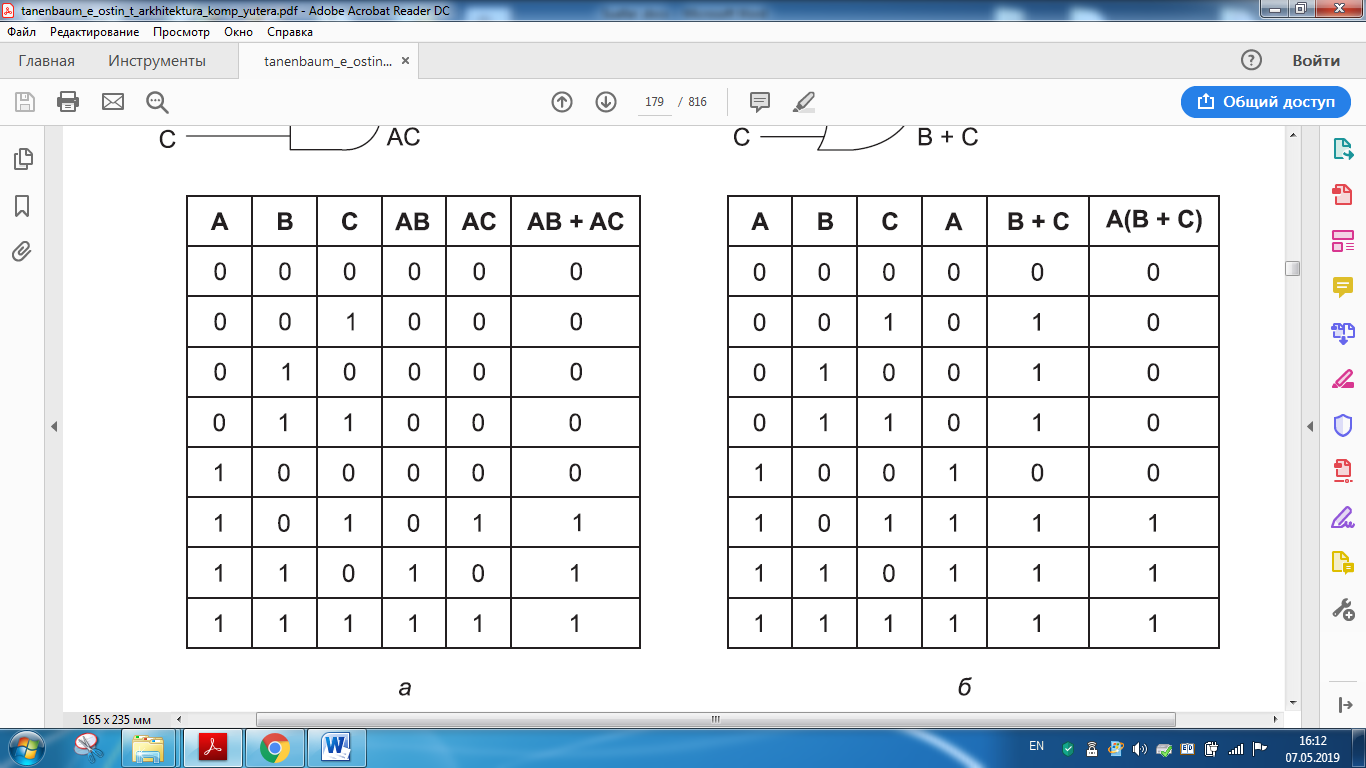


Məntiqihasilvəməntiqitoplamaəməlləriniyerinəyetirənventilbirləşmələriaşağıdagöstərilmişdir:



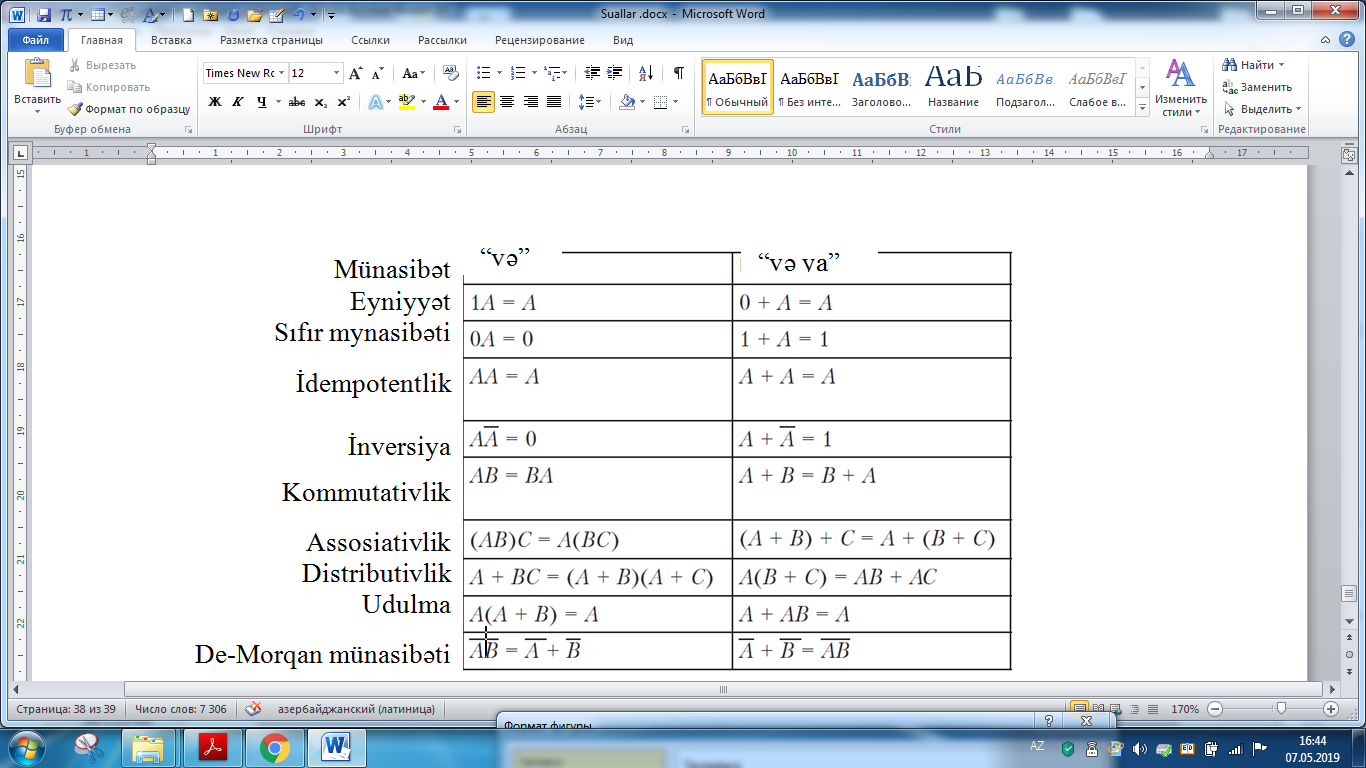
Bilmək lazımdır ki, eyni məntiq funksiyası müxtəlif quruluşlarda və müxtəlif tip ventillərdən istifadə olunaraq yaradıla bilər.





Şəkildəki sxemlər eyni M=AB+AC funksiyasını realizə edirlər. Ventillərin giriş siqnalları müvafiq cədvəllər vasitsi ilə göstərilmişdir. Şəkildən görünür ki, eyni funksiyanı yerinə yetirdiklərinə baxmayaraq, b) sxeminən istifadə daha məqsədəuyğun sayılmalıdır: - çünki istifadə olunan ventillərin sayı daha azdır.

Məntiq sxemlərinin qurulmasında (sintez olunmasında) məntiq cəbrinin əsas eyniyyətlərindən istifadə səmərəli olur. Bu eyniyyətlər aşağıdaılardan ibarətdir:



Kompyuterdə müxtəlif ventillərdən***funksional sxemlər*** adlanan mürəkkəb*sxemlər*

yığılır.

Aşağıdakı sxemdə təsvir olunmuş element*trigger* adlanır

(flip-flop). Triggerin

çıxışında 0 və ya 1 olur.

Məntiq elmi hər hansı bir obyekt haqqında söyləni­lən fikri, mühakiməni, təfəkkür tərzini və formasını öyrənən elmdir.

Riyazi məntiq, formal olaraq, müəyyən simvollar üzərində təyin olunan əməllərin (″bağlayıcıların″) hesabı kimi həm də elmlərdə irəli sürulən müxtəlif mülahizələ­rin düzgün olub-olmamasını təyin edən riyazi vasitədir.

Riyazi məntiqin ilk əsas anlayışlarından birisi ***müddəa*** anlayışıdır. Müddəa dedikdə nəqli cümlə şəklində söylənilən və konkret halda doğru və yalan olan mühakimə başa düşülür. Məsələn, ″Volqa çayı Qara dənizə tökülür″ − müddəadır və yalan qiymətini alır.

Hər söylənilən fikir müddəa deyildir. Məsələn, ″Yaşasın 1 May!″, ″İşıqlandı″, ″Onun boyu 2 metrdir″ − mühakimələrində qeyri-müəyyənlik olduğundan onlar müddəa deyildir.

Müddəalar məntiqində nəqli cümlə şəklində söylə­ni­lən müddəalara uyğun cümlələrin qrammatik quruluşu (sintaksisi) ilə maraqlanmırlar, yalnız onların doğru, yaxud yalan qiymətlərindən hansını alacağı ilə maraq­la­nırlar.

Müddəaları latın əlifbasının böyük hərfləri − A,B,C,D, ... və ya nömrələnmiş kiçik hərfləri ilə işarə edirlər.

Tutaq ki, müddəalar çoxluğu verilir və bu çoxluqda beş əməl − dizyunksiya, konyunksiya, implikasiya, ekvivalensiya və inkar adlanan əməlləri təyin olunur, yəni məntiq cəbrini vermək olur.

**Dizyunksiya əməli**

İki *A* və *B* müddəalarının ***dizyunksiyası*** onlarınheç olmazsa birisi doğru qiymətini aldıqda doğru olan müddəaya deyilir və belə işarə olunur: .

**Konyunksiya əməli**

İki *A* və *B* müddəalarının ***konyunksiyası*** yalnız və yalnız *A* və *B* müddəaları eyni zamanda doğru qiymətini aldıqda doğru olan müddəaya deyilir və belə işarə olunur:  və ya , və ya .

**İmplikasiya** (***izləmə***) **əməli**

 və  müddəalarının ***implikasiyası***  və ya  kimi işarə olunur. Yalnız və yalnız *A* müddəası doğru qiymətini, *B* isə yalan qiymətini aldıqda  müddəası yalan qiymətini alır.

 ifadəsində  şərt və  isə nəticə olan müddəadır.

**Ekvivalensiya əməli**

 ilə -nin ***ekvivalensiyası*** olan müddəa və ya  kimi işarə olunur.  ilə -nin hər ikisi eyni qiymətlər aldıqda  müddəası doğru qiymətini alır.

**İnkar əməli**

*A* müddəasının ***inkarı*** olan müddəa  və ya  kimi işarə olunur, yəni ″qeyri- *A*-dır″deməkdir.

Müddəanın doğru qiymətini 1 − simvolu ilə, yalan qiymətini 0 − simvolu ilə işarə edək və yuxarıdakı əməlləri onların qiymətlər cədvəli ilə verək.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

|  |  |
| --- | --- |
|  |  |
| 0 | 1 |
| 1 | 0 |

Müddəalar məntiqinin əlifbası (istifadə olunan simvollar çoxluğu) bunlardır:

1. hərfi simvollar və ya nömrələnmiş hərfi simvollar;
2. məntiqi əməllərin simvolları −  və bunlarla təyin olunan törəmə simvollar:  − Şeffer ştrixi,  − Pirs ibrəsi,  − iki moduluna görə

cəmləmə əməli;

1. kiçik mötərizələr və vergül simvolları: ( , ) .

Təbii dildə sadə cümlələrdən bağlayıcıların və əlifbanın simvollarından düzəldilən sözlərin köməyi ilə mürəkkəb cümlələr düzəldiyi kimi, müddəalar mənti­qində də mürəkkəb müddəalar düzəltmək olar. Belə müddəalar düsturlardır.

**Mövzu № 7.**

Hər bir EHM –in fiziki qurğuları ilə yanaşı əsas tərkib hissəsi onun proqram təminatıdır. Lazımi proqramlar olmadan əslində maşını heç bir iş görməyə vadar etmək olmaz.

FK – da müxtəlif məsələləri həll etmək üçün təyin olunmuş proqramlar yığımı və onları müşayiət edən sənədləşdirmələr **proqram təminatı - PT (software**) adlanır.

PT iki yerə bölünür: 1) sistem, 2) tətbiqi.

Kompüterlərin idarə olunması üçün, istifadəçinin digər proqramlarının yaradılması və yerinə yetirilməsini təşkil etmək üçün, habelə istifadəçiyə bütün mümkün xidmətlər növünü göstərmək üçün zəruri olan **PT –sistem proqram təminatı** (system software ) adlanır.

Sistem proqram təminatını (SPT) belə bölmək olar:

1. əməliyyat sistemləri (ƏS),
2. servis sistemlər,
3. proqram – instrumental sistemlər,
4. texniki xidmət sistemləri

Əsas yeri ƏS tutur. ƏS (operating system) – kompüterin bütün qurğularının işini idarə edən və tətbiqi proqramların yerinə yetirilmə prosesini tənzimləyən proqramlar toplusudur.

ƏS – in yerinə yetirdiyi əməliyyatlar:

* FK – in hissələrinin işləmə qabiliyyətinə nəzarət;
* Başlanğıc yüklənmə prosedurasının yerinə yetirilməsi;
* FK –in qurğularının işinin idarə olunması;
* Fayl sisteminin idarə olunması;
* Istifadəçi ilə FK – nın qarşılıqlı əlaqəsi;
* Tətbiqi proqramların yüklənmə və yerinə yetirilməsi;
* FK – in resurslarının bölünməsi(məs., operativ yaddaş, prosessor vaxtı və tətbiqi proqramlar arasında periferiya qurğuları).

Hal – hazırda SP/M, MS DOS, UNİX, Windows ƏS- i mövcuddur. MS DOS (Microsoft Disc Operating System – yəni Mikrosoft firmasının “diskli” ƏS - i) – 1981 – ci ildə İBM – in sifarişi ilə Microsoft firması tərəfindən yaranıb. Onun analoqları: PC DOS (Personal Computer Disk Operating System – FK –in diskli ƏS) İBM firması tərəfindən yaradılıb, PTS DOS – Fiztexsoft firması tərəfindən, Novell DOS isə Novell DOS isə Novell firması tərəfindən yaradılıb. DOS ƏS –i öz sadəliyi və səmərəliyinə baxmayaraq artıq köhnəlmişlər. Müasir ƏS –i aşağıdakıları təmin etməlidirlər:

* Çoxməsələlilik-yəni eyni zamanda bir neçə proqramın birgə yerinə yetirilməsini;
* Istifadəçi üçün inkişaf etmiş qrafik interfeys;
* Müasir mikroprosessorların bütün imkanlarından istifadə;
* Işdə dayanıqlılıq və müdafiə olunmaqlıq;
* Aparaturadan tam asılı olmamazlıq;
* Kompüterin təhlükəsizliyinin təmini vasitələri;
* MS DOS üçün işlənmiş bütün əlavələrlə uyğunluq.

Bundan başqa müasir ƏS şəbəkə funksiyalarının yardımını təmin etməlidir. Məsələn, şəbəkəyə məsafəli girişi, İnternetə inteqrasiyanı, daxili elektron poçtu və telekonfransların təşkili, fayl və printerlərin yüksək məhsuldarlıqda birgə istifadə olunması və s.

**MS DOS –un versiyaları:** 3.3, 6.22 (1994 – də buraxılıb) – versiyasında rus hərfləri daxil edilib və 7.0 versiyaları.

**ƏS –nin** **təsnifatı:** Prosessorun idarəedilmə alqoritminin xüsusiyyətlərindən asılı olaraq ƏS aşağıdakılara bölünür:

1) Birməsələli və çoxməsələli ƏS;

a) Birməsələli (MS DOS, MSX);

b) Çoxməsələli (OS ES,OS/2,Unix,Windows ƏS).

2) Biristifadəçili və çoxistifadəçili ƏS;

a) Biristifadəçili ƏS(MS DOS, Windows 3.x),

b) Çoxistifadəçili ƏS (Unix, Windows NT, Windows XP, Linux).

3) Birprosessorlu və çoxprosessorlu sistemlər;

4) Şəbəkə və lokal ƏS.

Şəbəkə ƏS- hər bir hesablama şəbəkəsinin əsasıdır. Hal-hazırda 32 mövqeli 5 əsas şəbəkə ƏS-ni ayırmaq olar:

* NET Were 4.1 (Novelle firması),
* Windows NT (Microsoft firması),
* Vines 6.0 (Banjan firması),
* OS/2 Warp Advanced Server İBM,
* Unix şəbəkə ƏS.

ƏS-nin işinin əsası kompüterdə müəyyən işləri yerinə yetirmək üçün proqramların fəaliyyətini təşkil etməkdən ibarətdir. Proqram əmrlərdən təşkil olunur. Əmr- format adlanan müəyyən edilmiş strukturaya malik olmalıdır. Əmrin formatını onun adı və parametri təyin edir. Əmrlər 2 yerə bölünür:

1. resident əmrlər (daxil) – sistemin tərkibində olan,
2. transit (xarici, utilit) əmrlər - genişlənməsi .*com* və . *exe* olan və diskdə yerləşən adi fayllardır.

**Servis sistemlər –** ƏS-nin imkanlarını genişləndirir, istifadəçiyə, həmçinin icraolunan proqramlara əlavə xidmətlər göstərir. Məsələn, proqram örtükləri, utilitlər və əməliyyat mühitləri.

Servis proqramları aşağıdakı funksiyaları yerinə yetirir:

* istifadəçi interfeysinin təkmilləşdirilməsi,
* verilənlərin mühafizəsi,
* verilənlərin bərpası
* xarici və əməli yaddaş arasındakı informasiya mübadiləsinin sürətləndirilməsi,
* arxivləşdirmək ( arxiv açmaq),
* komputer virusları ilə mübarizə.

Servis proqramlar təşkilinə, reallaşdırma üsullarına və yerinə yetirdikləri funksiyalara görə aşağıdakılara bölünür;

* örtük proqramlar,
* utilitlər,
* antivirus proqramlar (Doctor Web, Doctor Web for Windows, Dr. Solomon Antivirus 7.0, İBM Antivirus 3.01, Norton Antivirus 4.0). Norton Antivirus proqramı Semantec firması tərəfindən yaradılıb.

**Proqram instrumental vasitələr** – proqram təminatının işlənməsi üçün təyin olunmuş proqram məhsuludur.

FK üçün sistem və tətbiqi PT instrumental dillər vasitəsilə yaradılır. İnstrumental vasitələrə aiddir:

1. mikroassemblerlər (maşınyönlü dillər);
2. yüksək səviyyəli dillərin translyatorları;
3. proqramların redaktə, qurulma və yüklənmə vasitələri.

Göstərilən proqram vasitələri proqramçılara yeni proqramlar yaratmaq üçün lazımdır və buna görə də instrumental vasitələr adlandırılır. Bunlardan ən çox istifadə olunanları yüksək səviyyəli alqoritmik dillərdir: Pascal, Basic, C, C++. Fortran və s.

Hal-hazırda MS Visual Basic, Borland Delphi kimi prosedur yönümlü proqramlaşdırma sisteməli daha tez-tez işlədilir.

**Texniki xidmət sistemləri** – kompüterin iş prosesində nasazlıqları aşkar etmək üçün FK – in proqram – aparat vasitələridir. Onlar ayrı-ayrı hissələrin, blokların və ümumilikdə bütün maşının iş qabiliyyətinin yoxlanması üçün təyin olunublar.

**Tətbiqi sistemlər** – proqram təminatının istifadəsi üçün ən yaxşı tərəfidir. Onların əsas vəzifəsi yeni proqram yaratmaq deyil, hazır proqram təminatından istifadəçinin məqsədi üçün istifadə etməyi asanlaşdırmaqdan ibarətdir. Məsələn, mətnlə işləməyi asanlaşdıran mətn prosessorları, qrafik paketlər, müəyyən sinif riyazi məsələləri həll etmək üçün proqram paketləri və s. tətbiqi sistemlərə aiddir.

**Windows ƏS**

Windows 2000 – istifadəçinin müxtəlif tipli məsələlərinin həllinə yönəlmiş 4 şəbəkə əməliyyat sistemini əhatə edir.

1. Windows 2000 Professional - ofis və mobil FK – lər üçün təyin olunmuş şəbəkə ƏS –dir.
2. Windows 2000 Server – orta səviyyəli və böyük olmayan müəssisələrdə istifadə olunan, 4 – prossessorlu server və 4 Qbayt operativ yaddaşa malik olan universal şəbəkə ƏS –dir.
3. Windows 2000 Advanced Server – 8 prosessorlu server və 8 – Qbayt yaddaş tələb edən xüsusiləşmiş ƏS –dir. Bu ƏS Windows 2000 Server ƏS-nin bütün funksiyalarını daha böyük məhsuldarlıqla yerinə yetirə bilir.
4. Windows 2000 Datasenter Server – 32 prosessorlu arxitektura və 64 Qbayt operativ yaddaş tələb edən, daha böyük həcmli məsələlərin həlli üçün istifadə olunan ƏS –dir.

Windows ailəsinin davamı - Windows XP –dir (XP- e X Perience – təcrübə). O, daha geniş funksional imkanlara malikdir (əsasən də multimedia sahəsində), dayanmalara az dərəcədə meyllidir və bir çox tətbiqi proqramların daha tez buraxılmasını təmin edir.

Windows XP – 32 və 64 mövqeli İntel prosessorlu kompüterlər üçün işlənmişdir. O, üç versiyada buraxılmışdır: 1) XP Home Edition, 2) XP Professional, 3) XP 64 – bit Edition.

Windows XP Home Edition - ev və kiçik biznesdə istifadə üçün nəzərdə tutulmuşdur. Faktiki olaraq, o, Windows 9x – in modifikasiyasıdır.

Windows XP Professional – biznes sahəsində Windows 2000 və ya Windows NT istifadəçiləri üçün tətbiqə yönəlib. Ev sisteminə nəzərən o, daha geniş funksional imkanlara malikdir.

Windows XP Professional və Windows XP Home sistemlərinin aparat tələbləri oxşardır:

* Operativ yaddaşın həcmi – 128 Mbayt (minimum – 64 MB);
* 32 mövqeli prosessorun takt tezliyi ≥ 300 Mhs;
* Bərk diskdə sərbəst yerin həcmi ≥ 1,5 Qb;
* CD- ROM və ya DVD – ROM diskovodlarının varlığı;
* Videokart və ≥ 800 x 600 nöqtəli SVGA monitoru

Windows ® XP 64 – bit Edition – 64 mövqeli müştəri ƏS- dir. O (texniki və elmi əlavələrin işlənməsi ilə məşğul olan) texniki işçi stansiyalarının ixtisaslaşdırılmış istifadəşiləri, üçölçülü animasiya və s. üçün təyin olunub.

Bu ƏS üçün 16 QB operativ yaddaş və 8 Terabayt (Tb) virtual (mümkün olan) yaddaş nəzərdə tutulur.

Windows XP 64 Bit sistemi 64 mövqeli İntel İtanium TM prosessorlu kompüterlərə qoyulmaq üçün nəzərdə tutulub.

Windows XP – nin interfeysi – sadə və cəlbedicidir. “ İşçi masası ”nın xarici görünüşü xeyli dəyişib.

“Mənim şəkillərim”, “Mənim musiqim ”, “Mənim videom ”, “Mənim radiostansiyam” və b. qovluqlar əmələ gəlib. “İdarəetmə paneli ” xeyli dəyişib.

Windows XP sistemi NTFS fayl sistemilə işləmək üçün istiqamətlənib. Lakin müxtəlif ƏS ilə işləmək zərurəti olduqda ( 16 mövqeli prosessorlar üçün yazılmış) Windows XP FAT32 və FAT fayl sistemlərini himayə edir.

Windows 9x - 2000 üçün işlənmiş 32 mövqeli əksər əlavələr Windows XP – nin idarəsi altında işləməlidir.

Windows 2000 - ə nəzərən Windows XP – də idarəetmə sistemi təkmilləşib və yeni imkanlar əlavə olunub. Xüsusi halda, Windows XP müasir aparat vasitələrinin bütün üstünlüklərindən istifadə etməyə imkan yaradır.

Windows XP – də aşağıdakı idarəetmə sistemi funksiyaları əlavə olunub:

* Səhvlərdən müdafiəni təmin etmək üçün sistemin avtomatik bərpası;
* əvvəl qoyulmuş nasaz drayverlərin əvəzlənməsi və sistemə uyuşmayan drayverdən imtina;
* sistem və şəbəkə problemlərinin aradan qaldırılması və diaqnostikası ( internet xidmətin cari statusnu bilmək, kompüter haqqında məlumat almaq, ƏS haqqında məlumat, həmçinin modemlərin statusunu və şəbəkə adapterlərini yoxlamaq;
* uzaqlaşmış maşında olan sənəd və fayllara giriş üçün kompüterə distansion birləşmə;

Windows XP sisteminə İnternet Explorer 6.0 şərhçisi, Outlook Express ( VI versiya) poçt proqramı, Windows Mediya Player 8 – “proiqrivateli” və s. daxildir.

**Əməliyyat sisteminin arxitekturası.**

Əməliyyat sistemləri qarşılıqlı əlaqədə olan proqram komplekslərinin toplusudur. Bir-biri ilə əlaqədar olması və istifadəsi ilə dialoqa girərək, qəbul etdiyi məlumat əsasında kompüter daxilində prosesləri idarə etməsi əməliyyat sistemi haqqında bir fiziki qurğu təəssüratı yaradır.

Əməliyyat sistemlərini kompüterə yükləmək (instalyasiya etmək) üçün lisenziyalı kompakt-diskdən onun proqram kompleksi kompüterin sərt disk yadaşına yazılır. Bu əməliyyatı kompakt diskin üzərində olan xüsusi proqram yerinə yetirir. Məhz bu proqram proqramları sərt diskdə lazımi struktura uyğun yerləşdirə bilir.

Bu proqramların bir hissəs **rezident** proqramlar adlanırlar. Çünki onlar kompüterin işə qoşulduğu andan başlayaraq daimi olaraq onun operativ yaddaşında qalaraq, istifadəçinin arzu etdiyi funksiyaların icra olunmasını həyata keçirirlər.

ƏS-də digər proqramlar da vardır ki, onlara **tranzit** proqramlar deyilir. Onlar operativ yaddaşa periodik olaraq zərurət yarandıqda, yəni müəyyən funksiyaları yerinə yetirmək üçün çağırılırlar.

ƏS-in rezident hissəsini ƏY-a yükləyən proqram vardır ki, ona **“yükləyici”** deyilir.

Əməliyyat sistemlərində informasiyanın daxil edilməsini və çıxışa verilməsini həyata keçirən standart proqramlar və onları əlaqələndirən proqramlar da vardır ki, onlar “**Input — Output System”**proqram kompleksidir.

Kompüter ilə istifadəçi arasında dialoqu həyata keçirən proqramlar toplusu **interfeys** poqramlarıdır.

Kompüterin yaddaşında həm standart proqramlar, həm sistem proqramların çoxsaylı siyahısı içərisində lazım olanı tez tapmaq və operativ yaddaşa çağırmaq üçün əməliyyat sistemində **“faylların idarə olunması sistemi”** yaradılmışdır. Bu komponent ƏS-nin işini xeyli dərəcədə effektiv edir. Bu sistem faylları aid olduğu sinifə görə işarələyərək unikal ad ilə adlandırır. Məsələn, **.sys** — (əməliyyat sistemi proqramı), **.pas**—paskalda yazılmış proqram və s.

Əməliyyat sistemləri faylları disk yaddaşından operativ yaddaşa çağırmaq üçün fayllar cədvəli  **FAT (File Allocation Table)-**dən istifadə edirlər.

ƏS-nin ən mühüm funksiyalarından biri prosessorun “**dayandırma**”(kəsilmə) siqnallarına reaksiyanın həyata keçirilməsidir. Prosessorun fəaliyyətinin gedişində bir çox hallar vardır ki, bu hallar baş verdikdə prosessor fəaliyyətini dayandırmalı, situasiyanı təhlil etməli və qərar çıxarmalıdır. Qərar həm də işi davam etdirməkdən ibarət ola bilər. Müasir kompüterlərdə belə dayandırmalara səbəb ola biləcək halların sayı 256 –dır.

Bu göstərilən komponentlərdən başqa alqoritmik dillərdə tərtib olunmuş proqramları maşın koduna çevirmək üçün istifadə olunan **kompilyator** və **translyator** proqramları da vardır.

Kompüter arxitekturasının ierarxik səviyyələr düzülüşündə əməliyyat sistemi maşın əmrləri səviyyəsindən yüksəkdə durur. Bu o deməkdir ki, bu səviyyə aparıcı imtiyaza malikdir, yəni bu səviyyə əmrləri özündən aşağı səviyyə proqramlarını (əmrlərini) idarə etmək hüqüqüna malikdir. Belə tabeçilik səviyyə düzülüşünə **ierarxiya** deyilir. Maşın əmrləri səviyyəsini əməliyyat sistemi səviyyəsi ilə müqayisə etmiş olsaq, qeyd etməliyik ki, əgər maşın əmrləri səviyyəsi proqramçı üçün “görünməzdirsə” əməliyyat sistemi əmrlərini proqramçı nəinki lazım olduqda izləyə bilir, hətta operativ olaraq onları dəyişdirmək və korreksiya etmək imkanına da malikdir.

Əməliyyat sistemi əmrləri **“sistem çağırışları”**adlanırlar. Bu çağırışlar əməliyyat sisteminin tərkibində nəzərdə tutulmuş əmrləri işə salır. Bu əmrlərin yerinə yetirilməsi əvvəlki səviyyə əmrlərindən fərqli olaraq həmişə interpretasiya rejimində həyata keçirilir. İnterpretasiya rejimi prinsipinə uyğun olaraq hər bir əmr interpretasiya olunaraq, yəni maşın əmrləri ardıcıllığına çevrilərək icra olunduqdan sonra növbəti sistem əmri yerinə yetirilməyə başlayır. Lakin ierarxiya prinsipinə uyğun olaraq aşağı səviyyə (maşın əmrləri səviyyəsi) yerinə yetirilərkən bu prosesdə yuxarı səyiyyə heç cürə iştirak etmir.

Beləliklə əməliyyat sisteminə əlaqə vasitəçisi demək olar. Əlaqə təkcə kompüter ilə istifadəçi arasında deyil, həm də komputerdə yerinə yetirilən ayrı-ayrı funksiyalar arasında da olur.

Əməliyyat sisteminin vəzifəsi kompüterin resurslarından effektiv olaraq istifadə etməkdə istifadəyə şərait yaratmaqdır.

**Əməliyyat sisteminin əsas funksiyaları** (Virtual yadaş, Giriş-çıxış, Paralelizm )

Kompüterin resursları dedikdə əsasən aşağıdakılar nəzərdə tutulur:

- Prosesorun çalışdığı vaxt;

- Operativ yaddaşın (əsas yaddaş) həcmi;

- Giriş-çıxış aparat vasitələrindən istifadə;

- Xarici yaddaş yasitələrindən (informasya daşıyıcılarından) istifadə.

Kompüter işə salınan kimi, ilkin yükləyici proqramlar işə qoşulduqdan sonra onun əsas aparat blokları fəal vəziyyət alırlar. Bundan sonra əməliyyat sisteminin əsas **nüvəsi** kompüterə yüklənir və o öz funksiyalarını icra etməyə başlayır. Məhz əməliyyat sistemi fəaliyyətə başladıqdan sonra kompüterin artıq işlək vəziyyətə gəldiyi özünü göstərir.

Əməliyyat sistemi kompüterdə həyata keçirilən bütün proseslərdən üst mövqedə durur, yəni proqramla idarə olunan bütün hadisələrə reaksiya vermək qabiliyyətinə malik olur, prosesləri idarə edir.

**Əməliyyat sistemin əsas funksiyaları içərisində aşağıdakıları qeyd etmək lazımdır:**

— Istifadəçi ilə dialoqun təşkili;

— İnformasiyanın daxil-xaric olunmasının təşkili;

— proqramların yerinə yetirilməsinin təşkili;

— Kompüter resurslarının proqramlar arasında bölüşdürülməsi (operativ və keş yaddaş, prosessor, xarici qurğular);

— proqramların işə qoşulmasının təşkili;

— əlavə xidmət(servis) proqramlarının işinin təşkili;

— müxtəlif daxili qurğular arasında əlaqənin təşkili;

— periferiya qurğularının (displey, disk yaddaşı, printer və s.) işinin proqram təminatının təşkili.

Bu funksiyalar içərisində mühüm əhəmiyyət daşıyan üç funksiya xüsusi qeyd olunmalıdır ki, əməliyyət sistemin nə qədər mükəmməl olduğu onlardan asılıdır**. Bunlar aşağıdakılardır:**

- Virtual yaddaşış təşkili və idarə olunması;

- Giriş-çıxış virtual əmrlər;

- Paralelizmi təmin etmək üçün virtual əmrlər.

**Əməliyyat sistemi vasitəsi ilə virtual yaddaşın idarə olunması**

Qeyd etmişdik ki, operativ yaddaş (əsas yaddaş) kompüterin ən əsas texniki göstəricisidir ki, onun imkan dairəsini müəyyən edir. Xarici yaddaş imkanları həmişə genişdir, lakin ona müraciət çox aşağı sürətlidir və ona tez-tez müraciət etmək özünü doğrulda bilməz. Operativ yaddaşın həcminin xeyli dərəcədə genişləndirilməsinə imkan verən effektiv bir üsul **virtual yaddaşın** yaradılması kimi adlandırılmışdır.

Fərz edək ki, məhdud həcmdə operativ yaddaşımızın real fiziki həcmi 4096 sözdur. Kompüterdə həll edəcəyimiz məsələdə bu həcmdən xeyli artıq miqdarda yaddaş tələb olunur. Xəyalən tələb olunan yaddaşın 10 dəfə artıq olduğunu qəbul edək və proqramı bu baxımdan tərtib edək. Şəkildə ünvan sahəsi (solda) 4096-nın dəfəliliyi ilə qeyd olunduğu göstərilir, yəni , 0, 4096, 8191 ... . Proqramı tərtib edərkən, proqramçı birinci 4096-lıq sahəyə öz proqramında “0” nömrəsini, 2-ciyə “1” , üçüncüyə “2” rəqəmlərini artırmaqla xəyalən artırılmış sahələrdən hansının nəzərdə tutulduğunu qeyd edə bilər.

İndi bu əməliyyatı əməliyyat sistemi həyata keçirməlidir. Proqramın idarəedici sözündə müvafiq nömrəni oxuyaraq sahəni təyin edir və onu fiziki (əsas) yaddaşa köçürür. Bu əməliyyat həm yaddaşa yazma, həm də oxunma üçün eyni dərəcədə qüvvədə qalır.

Beləliklə virtual yaddaş məhz bu prinsipi əsas götürür. İndi fiziki yaddaşda əvvəldən qalan informasiya itə bilməz. Onun tutumu əməliyyat sistemi tərəfindən xarici yaddaşa köçürülür. Köçürülmə prosesi elə həyata keçirilməlidir ki, istənilən zaman informasiyanı diskdən oxumaq mümkün olsun. Burada isə **səhifə** anlayışından istifadə olunur. Qabaqcadan nömrələnmiş massiv səhifənin nömrəsini göstərmiş olacaqdır. **Ünvan fəzası** dedkdə xəyalən fərz etdiyimiz ünvan həcmini nəzərdə tutulur ki, bu da fiziki ünvan həcmindən istənilən qədər çox ola bilər.

Bu texnologiyanı həyata keçirmək üçün **əməliyyat sistemi** müvafiq **ünvanlaşdırma**, **diskə yazma**, **yeni ünvanı təyin etmə** kimi prosesləri yerinə yetirməli olur.

**Mövzu № 8.**

XX əsrin ən böyük nailiyyəti yeni riyazi fənn – alqoritmlər nəzəriyyəsidir. EHM nəzəriyyəsi, və təcrübəsi, kibernetika və s. onsuz keçinə bilmir.

“Alqoritm” sözü özbək riaziyatçısı Xorezminin (ərəbcə Əl-Xorezmi) adından götürülüb. O, onluq say sistemində ədədlər üzərində 4 hesab əməlini işləyib. Bu qaydaları Avropada “alqorizm” adlandırıblar.

Sonralar bu anlayışı yalnız riyaziyyatçılar müxtəlif məsələlərin həlli qaydalarını işarə etməklə istifadə ediblər.

**Alqoritm** – müəyyən bir dildə şərh olunmuş və ilkin verilənləri axtarılan nəticəyə çevirmək prosesini təyin edən qaydadır.

Alqoritm dedikdə məsələnin həllini almaq üçün dəyişənlərin ədədi qiymətləri üzərində aparılan hesabi və məntiqi əməllər ardıcıllığı başa düşülür.

Alqoritmlərin aşağıdakı əsas xassələrini qeyd etmək olar.

1. müəyyənliyi – alqoritmin hər bir addimi dəqiq təyin olunmalidir;
2. kütləviliyi–müxtəlif ilkin verilənlər üçün nəticə almağa imkan verməlidir;
3. nəticəliliyi–sonlu sayda addimdan sonra nəticə alinmasini təmin etməlidir;
4. diskretlilik;
5. birqiymətlilik ;
6. əyanililiyi – alqoritm təkcə onu yaradan üçün deyil, həm də icraçilar üçün aydın olmalıdır;
7. effektivlilik.

Alqoritmlər sözlə, blok – sxemlə və proqramla təsvir oluna bilərlər. Blok – sxem vasitəsilə təsvir sözlə və alqoritmik dillə təsvirə nəzərən daha əyani olur. Bu zaman alqoritm bloklar ardıcıllığı vasitəsilə göstərilir və blokların daxilində onun yerinə yetirdiyi funksiyalar haqqında məlumat verilir. Sistemin blokları ardıcıl olaraq nömrələnirlər.

Alqoritmin aşağıdakı növlərini fərqləndirirlər:

1. xətti;
2. budaqlanan;
3. dövri alqoritmlər.

Xətti alqoritmlər - sxemin blokları verildiyi ardıcıllıqla bir – birinin ardınca yerinə yetirilirlər. Buna misal olaraq yuxarıdakı alqoritmi (şəkil 1.1) göstərmək olar.

Budaqlanan alqoritmlər hər hansı bir məntiqi şərtin yerinə yetirilməsindən asılı olaraq hesablama prosesi bu və ya digər istiqamətdə cərəyan edir.

Alqoritmlər sözlə, blok – sxemlə və proqramla təsvir oluna bilərlər. Blok – sxem vasitəsilə təsvir sözlə və alqoritmik dillə təsvirə nəzərən daha əyani olur. Bu zaman alqoritm bloklar ardıcıllığı vasitəsilə göstərilir və blokların daxilində onun yerinə yetirdiyi funksiyalar haqqında məlumat verilir. Sistemin blokları ardıcıl olaraq nömrələnirlər.

Nassi – Şneyderman blok – sxemi alqoritmin qrafik təsvirinin ən əyani üsulu hesab olunur.

**Misaı 1.** Üçbucağın tərəfləri verildikdə onun sahəsini Heron düsturu ilə hesablamalı.

Alqoritmin blok – sxemi aşağıdakı kimi olar:

a,b,c-ni daxil etməli

2

p=



3

4

s-i tapmalı

5 5

S-i çap etməli

6 son

Alqoritmin aşağıdakı növlərini fərqləndirirlər:

1. xətti;
2. budaqlanan;
3. dövri alqoritmlər.

Xətti alqoritmlər - sxemin blokları verildiyi ardıcıllıqla bir – birinin ardınca yerinə yetirilirlər. Buna misal olaraq yuxarıdakı alqoritmi (şəkil 1.1) göstərmək olar.

Budaqlanan alqoritmlər hər hansı bir məntiqi şərtin yerinə yetirilməsindən asılı olaraq hesablama prosesi bu və ya digər istiqamətdə cərəyan edir.

*Misal* Verilmiş funksiyasının qiymətini hesablamalı.

Alqoritmin blok – sxemi:

2

X daxil et etməli

yox

4

Y = x

3



hə



5

6 5

Y – i çap etməli

7 son

Dövri alqoritmlər - məsələnin həlli zamanı hər hansı qiyməti eyni bir riyazi asılılıq vasitəsilə çoxlu sayda hesablama lazım gəlir. Hesablama prosesinin belə çox saylı təkrarolunma sahələri dövrlər adlanır. Dövri alqoritmlər sxemin həcmini və ona uyğun proqramı kifayıt qədər qısaltmağa imkan verir.

*Misal*  x arqumenti 0 –dan 2 – yə qədər addımı ilə dəyişdikdə

funksiyasının qiymətlərini hesablamalı və onları çapa verməli.

olar:

2

a -daxil et etməli

3

X=0

4



5 5

Y – i çap etməli

6 x=x+



hə

7



yox

8 son

Riyaziyyatda “alqoritm” anlayışı qoyulmuş məsələnin həllinə gətirən ciddi təyin olunmuş riyazi əməliyyatlar ardıcıllığından ibarət qaydalar sistemi kimi başa düşülür.Riyaziyyatda həll olunmayan bir sıra məsələlər var.Məsələn,

1.Bucağın pərgar və xətkeşin köməyi ilə 3 bərabər hissəyə bölünməsi;

2.Dairənin xətkeş və pərgar vasitəsiylə kvadratlandırılması məsələsi;

3.Kubun həcminin 2 qat artırılması məsələsi(xətkeş və pərgar vasitəsi ilə).

Belə məsələlər “alqoritmik həll olunmayan məsələlər” adlanır.Alqoritmin normal tərifini Alan Turinq vermişdir.O, 1936-cı ildə bir abstrakt maşının sxemini təsvir etmiş və bu maşın nəyi təsvir etməyi bacarırsa onu “alqoritm” adlandırmışdır.Turinq maşını ilə yerinə yetirilə bilməyən alqoritm ola bilməz.Turinq müəyyən bir konstruksyanın işinin təsvirinin köməyi ilə əməliyyatların yerinə yetirilmə qaydalarını vermişdir.Turinq maşını real qurğu deyil və əhəmiyyəti ondan ibarətdir ki,onun köməyi ilə müxtəlif məsələlərin həlli üçün alqoritmlərin olub-olmamasını isbat etmək olar.Turinq maşını **sonsuz lentden** və **başlıqdan** ibarətdir.Lent **oyuqlara** bölünmüşdür.Bu lent Turinq maşınının “yaddaş qurğusu” rolunu oynayır.Lent sonsuz olduğu üçün Turinq maşınını reallaşdırmaq mümkün deyil.Oxucu başlıq lentin üzəri ilə hərəkət edir və lazım gələn oyuğu tapa bilir.İlkin informasiya lentin üzərində ardıcıl simvollar şəklində sonlu sayda oyuqlarda yerləşir.Başlıq {q1,q2,..,qn} vəziyyətlərindən birində ola bilər və bir {qi} simvolunu oxuya bilər.Başlıq aşağıdakı əməliyyatları yerinə yetirir:

1.Oyuğa yeni simvol yazır;

2.Lentin üzərində bir oyuğ sağa və ya sola yerini dəyişir;

3.Yeni vəziyyətə keçir.

**Post maşını**

sonsuz lent başlıq

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **1** | **0** | **1** | **1** |  |  |

cari oyuq

Əlifba:{a1 ,a2 ,….,an}

Post maşınında əlifba A={o,1,boşluq}

Turinq və Post maşınları ekvivalent maşınlardır.Onlar **universal icraçı** adlanır(universal performer).

Church-Turinq tezisi:

Istənilən alqoritm Turinq maşını üçün proqram şəklində yazıla bilər. **Alqoritm** Turinq maşını üçün proqramdır.

**Mövzu № 9.**

Proqramlaşdırmada bütün “qara” işləri tərkibində mil­yon­larla tranzistor yerləşən iki kibrit qutusu ölçüdə olan lövhə – prosessor yerinə yetirir. Bir saniyə ərzində prosessor yüz mil­yon­­­larla sadə əməllər – yaddaşın oyuqlarından oxu­maq­/yaz­maq, toplama, çıxma və s. yerinə yetirməyə qadirdir.

Alqoritmik dildə yazılmış proqramı prosessor başa düşmür. Ona görə də onu içra olunmazdan əvvəl *Assembler* adlanan xü­susi xidmətçi proqramdan buraxmaq lazım gəlir. Assembler –hərf­lərlə yazılmış maşın əmrlərini maşına aydın olan sıfırlar və va­hidlər ardıcıllığına çevirir.

Assemblerlər proqramlaşdırmada irəliyə atılan çox böyük bir addım olsalar da, prosessorun əmrlərini ancaq başqa cür ya­zırdılar. Assembler proqramı yalnız bir tip kompüterdə işləyə bilirdi. Onu fərqli əmrlər sistemli başqa tip prosessorda yerinə yetirmək mümkün deyildi.

Ona görə də yüksək səviyyəli proqramlaşdırma dilləri (mə­sə­­lən, BASİC, PASCAL və s.) yaradıldı. Bu dillərdə yazılmış proq­ramlar ən müxtəlif kompüterlərdə yerinə yetirilə bilirlər. Bu zaman xüsusi proqram – kompilyator onu as­sem­bler ko­du­na çevirir.

Beləliklə də bu şəkildə yazılmış proqramlar prosessorun başqa əmrləri ilə asanlıqla başqa kompüterlərə köçürülə bilər.

İxtiyari proqramlaşdırma dilində yazılmış proqram ilkin proq­ram olur. Bu proqramların xüsusiyyəti ondan ibarətdir ki, onlar in­sana aydın olan və kompüterin prosessoruna aydın ol­ma­yan instruksiyalardan ibarət olur. Prosessorun ilkin proq­ramda ya­zıl­mış alqoritmə uyğun iş görməsi üçün bu proqram ma­­şın di­li­nə çevrilməlidir. Proqramın belə çevrilməsi trans­lya­si­ya (tran­sla­tion - tərcümə) adlanır və o xüsusi proqramlar – trans­­­l­ya­­­torlar vasitəsilə yerinə yetirilir.

Translyatorların 3 növü vardır:

1. interpretatorlar,
2. kompilyatorlar,
3. assemblerlər.

İnterpretator – proqramın mətnini mərhələlərlə (əmrlər üz­rə­) çevirən və ilkin proqramın translyasiya olunmuş əmrlərini dər­hal (yəni paralel olaraq) yerinə yetirən trans­lya­tor­dur.

Kompilyator – proqramın mətnini maşın dilində modula ötü­­rür, sonra proqram operativ yaddaşa yazılır və yalnız bun­dan sonra kompüterin prosessoru vasitəsilə yerinə yetirilir.

Əgər kompilyatorun maşın kodunun generatoru proq­ra­mın il­­kin mətnini lazımi formaya çevirdisə, deməli, proqramın mət­­­­­nin­­də sintaksis səhvlər yoxdur, lakin bu hələ alqoritmdə səhv­­­lə­­­rin olmaması demək deyildir.

Kompilyatorun işi sxematik olaraq Şəkil - də göstə­ri­lir:

*İlkin proqram* *İcraolunmuş proqram*

2

1

*Səhv haqqında məlumat*

1. proqramın mətninin sintaksis nəzarət bloku,
2. maşın kodunun generatoru.

Kompilyatorun iş sxemi

Formalarından asılı olmayaraq bütün translyatorlar aşa­ğı­da­­kı əsas məsələləri həll edirlər:

- translyasiya olunan proqramın sintaksisini yoxlayır və ana­­liz edir­lər;

- proqramın maşın kodunu yaradırlar;

- çıxış proqramı üçün yaddaşı bölüşdürürlər.

İnterpretasiya – şifahi tərcüməyə oxşayır. Giriş proq­ra­mı­nın hər bir təlimatı tərcümə olunur və yerinə yetirilir. Bu­ qay­da­da

təkrar təlimatlar hər dəfə kodlaşdırılır. Kompil­ya­siya isə ya­zı­lı tər­cüməyə bənzəyir. Proqram yerinə yetiril­məz­dən əv­vəl 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.

Kompüterin alqoritmi başa düşməsi üçün proq­ram­laş­dır­ma dil­lərindən istifadə edilir.

Məsələ həll edərkən əvvəlcə aşağı­­­da­kı addımlar yerinə ye­ti­­ril­məlidir:

1. Məsələnin riyazi qoyuluşu:

- nə verilir – ilkin verilənlərin sadalanmasi;

- nə tələb olunur – nəticələrin sadalanmasi ;

- ilkin verilənlərin məhdudiyyət şərtləri.

2. Riyazi model: nəticələri almaq üçün lazım olan bütün qayda və qanunlar.

3. Həll metodu: riyazi modelin optimal istifadə olunması.

Daha sonra məsələnin həlli üçün ye­ri­nə yetiriləcək əmə­liy­yat­­­la­­rın alqoritmi tərtib edilir və bu əməliyyatlar hər hansı al­qo­­ritm (proqramlaşdırma) dilində əmrlər şəklində yazılır. Tər­tib olun­­muş proqram yuxarıda qeyd edildiyi kimi xüsusi əla­və­lər (translyator proqramlar) va­si­tə­silə yerinə yetirilir və ya maşın koduna çev­rilir.

**Proqramlaşdırma dilləri** - proqram modullarından iba­rət­ olub, məsələnin həll mərhələsinə hazırlığını təmin edir.

Proqramlaşdırma dilləri adi dillərdən “sözlərin” (ancaq trans­lya­torun başa düşdüyü) sayına və əmrlərin ciddi yazılış qay­da­sı­na görə fərqlənir. [Kompüterdə](http://wapedia.mobi/az/EHM) proqram yazmaq üçün

isti­fadə olu­­nan formallaşmış dillərə **proqramlaşdırma dilləri** deyilir.

İstə­nilən proqramlaşdırma dilinin əsas elementləri bun­lar­dır: di­lin əlifbası, sintaksisi və semantikası.

* Dilin əlifbası dedikdə, həmin dildə işlənən bütün sim­vol­­lar nəzərdə tutulur.
* Sintaksis - əlifbada olan simvollardan dilin ayrı-ayrı kons­truksiyalarının (əmrlərin, operatorların)­­­­­ düzəl­dil­mə­sinin for­mal qaydalarıdır. Bu qaydalar müxtəlif həll alqoritmlərini proqramlaşdırmağa imkan verir.
* Semantika - dilin bu və ya digər sintaksis kon­struk­si­ya­­la­­­rının deyiliş qaydaları sistemidir. Məsələn, əgər proqramın bir yerində *y=a\*(b+c)* ifadəsinin hesablanması yazılıbsa, onda seman­tika qay­daları maşına “göstərir” ki, əvvəlcə *(b+c)* cəmini hesablasın, sonra həmin cəmi a-ya vursun.

Beləliklə, hər hansı [verilənlərin](http://wapedia.mobi/az/Veril%C9%99nl%C9%99r) emalı prosesini birbaşa hə­ya­ta ke­çir­­məyə imkan verən proqramlar, dili təyin edən sin­tak­sis qay­dalara uyğun olaraq əlifbadakı simvolların bir­ləş­mə­­si nəticəsində və semantika qaydalarını nəzərə almaqla işlənib hazırlanır.

Proqramlaşdırma dili vasitəsilə hazır proqram yox, ancaq qu­­rul­­muş alqoritmi təsvir edən mətn yaradılır. İnsanın başa düş­düyü dildə olan bu proqram maşının başa düşdüyü dilə çev­ril­məlidir. Bunun üçün kompüterdə translyatorlar və kom­pil­ya­tor­lar olur.

Proqram ancaq onların translyatorları olan halda icra oluna bi­­lərlər. Translyatordan fərqli olaraq kompilyatorlar .*exe*-fayl­la­rın yaradılması üçün istifadə olunur ki, onlar da sər­bəst icra olu­na bilərlər (yəni, proqramın yazıldığı mühitdən (sistemdən) ası­lı olmadan).

Müxtəlif tip prosessorlar müxtəlif tip əmrlər sisteminə ma­­­likdir. Əgər proqramlaşdırma dili konkret prosessor tipi­nə­­ yö­nə­libsə və onun xüsusiyyətlərini nəzərə alırsa, onda ona­­­­­ aşağı sə­viyyəli proqramlaşdırma dili deyirlər. Assem­b­ler­­­ aşağı sə­viy­yə­li proqramlaşdırma dilidir. Çünki o, bir əmri mne­mo­ni­ka ad­la­nan simvol işarələmələrinin köməyilə ədəd­­lər şək­lində yox, maşın kodları şəklində verir. Assemblerin kö­mə­yi­lə çox sə­mə­rə­li və kompakt proqramlar yaratmaq müm­kün­dür. As­sem­­­bler­dən adətən, sistem əlavə­lə­rin, drayver-proq­ram­­la­­rın, kom­pü­te­rin aparat resurslarına mü­raciət edən proqram mo­dul­la­rının ha­zır­lanması üçün isti­fa­də olunur. Aşağı səviyyəli proq­ram­laş­dır­ma dillərindən, adə­tən yüksək səviyyəli peşəkar proq­ramçılar is­tifadə edir. Yük­sək səviyyəli proqramlaşdırma dil­­ləri isə adi di­lə daha ya­xın və insan üçün daha aydın başa dü­şü­lən­dir. Çox ya­­yılmış, bəzi proq­ram­­laşdırma dilləri haq­qın­da məlu­mat verək.

Proqramlaşdırma dilləri aşağıdakı növlərə bölünür:

* Aşağı səviyyəli dillər (Assembler, Avtokod və s.),
* Yüksək səviyyəli dillər (Fortran, Alqol, Kobol, Basic, Pas­cal, C və s.),
* Ifratyüksək səviyyəli dillər (Alqol 68 və s.)

Aşağı səviyyəli proqramlaşdırma dillərində hər operatora bir maşın əmri uyğun gəlir. Bu dildə yazılan proqram az yer tu­tur və tez yerinə yetirilir. Aşağı səviyyəli dillərdən sistem proq­ram­çılar istifadə edir. Yuxarı səviyyəli proq­ram­laş­dır­ma dil­lə­rin­də hər operator bir neçə maşın əmri ilə əvəz edilə bi­lər, bu isə yaddaşda çox yer tutur. Yüksək səviyyəli dil­lər­dən isə tət­bi­qi proqramçılar istifadə edir.

Proqram dili anlayışı. Kompilyator və interpretatorlar.

Proqramlaşdırma dili kompüter proqramlarını yazmaq üçün istifadə olunan və müxtəlif işarələrdən ibarət olan formal sistemdir. Hər bir proqramlaşdırma dilinin 3 əsas anlayışı var.

1.Leksika

2.Semantika

3.Sintaksis

Leksika proqramlaşma dilinin söz yığımıdır, yəni dilin lüğətidir.

Sintaksis proqramların strukturunu və operatorların yazılış formasını təyin edən qaydaların yığımıdır. Proqramlaşdırma nəzəriyyəsində sintaksis Bekus-Naur qaydaları vasitəsi ilə təyin edilir.

Proqramlaşdırma dillərinin semantikası.

Semantikanın təyinində bir neçə yanaşma üsulları mövcuddur.

1.Operasion

2.Derivasion (Aksiomatik)

3.Denotasion (Riyazi)

Ümumiyyətlə semantika sözlərin mənası deməkdir. Proqramlaşdırmada isə semantika operatorların və dilin əsas kostruksiyalarinin başlanğıc mənasıdır.

1.Operasion yanaşmada proqramlaşdırma dilinin konstruksiyalarının icra olunması hansısa abstrakt EHM vasitəsi ilə interpretasiya olunur.

2.Derivasion semantikasında proqramlaşdırma dilinin konstruksiyalarının icrasının nəticəsi məntiqi ön və son şərtlərin vasitəsi ilə təsvir edilir.

3.Denotasion semantikasında çoxluq, uyğunluq, bərabərlik, mühakimə və təstiq kimi riyazi anlayışlardan istifadə edilir.

Nəzəri baxımdan proqramı adi insan dili vasitəsi ilə yazmaq olar. Belə yazılış metadildə proqramlaşdəma adlanır. Amma belə proqramı avtomatik olaraq maşın kodlarına çevirmək mümkün deyil.

Proqramlaşdırma dillərinin təsnifatı.

1.Aşağı səviyyəli və yüksək səviyyəli proqramlaşdırma dilləri.

Prosessor əmrləri və verilənləri elektrik siqnalları şəklində qəbul edən böyük inteqral mikrosxemdir. Faktiki olaraq prosessor sadə elektron elementlərin-tranzistorların çox böyük birləşməsi kimi təsvir edilə bilər. Hər tranzistorun 3 çıxışı olur. Qıraqdakı 2 çıxışa lazimi gərginlik verilir və bu yolla tranzistorda elektrik cərəyanı yaradılır. Ortadakı çıxışa isə gərginliyin verilməsi tranzistorun daxili müqavimətini idarə etməyə imkan verir. Prosessorda tranzistorlar trigger və ventil adlanan mikroelementlərə qruplaşdırılır. Triggerlərin 2 sabit- dayanıqlı vəziyyəti olur(açıq, bağlı). Elektrik siqnalları vasitəsi ilə onlar bir vəziyyətdən o biri vəziyyətə keçirlər. Bu davamlı vəziyyətlərə 0 və 1 riyazi anlayışları uyğun gəlir. Ventillər müəyyən qədər mürəkkəbdir. Onların bir neçə girişi ola bilər (Çıxışda olan gərginlik girişdə olan gərginliklərin kombinasiyasından asılıdır). Ventillər sadə arifmetik və məntiqi əməllərin yerinə yetirilməsini təmin edirlər. Prosessora əmrlər şinlər vasitəsi ilə verilir və əslində onlar da elektrik siqnallarıdır, yəni onlar 0 və 1-lərin kombinasiyası kimi təsvir edilə bilər. Müxtəlif əmrlərə müxtəlif kodlar uyğun gəlir. Buna görə də reallıqda proqramı kodların ardıcıllığı kimi təsvir etmək olar. Prosessorların müxtəlif növləri əmrlərin müxtəlif yığımlarına malikdir. Əgər proqramlaşdırma dili prosessorun konkret növünə yönəlibsə o aşağı səviyyəli proqramlaşdırma dili adlanır. Məsələn assembler dili. Assembler dili maşın kodunun hər bir əmrini mnemonika adlanan şərti simvol işarələrlə təsvir edir. Bir maşın instruksiyasının assembler əmrinə birmənalı çevrilməsi transliterasiya adlanır. Amma prosessorun hər bir modeli üçün instruksiyaların yığımı fərqlənir və hər bir konkret komputer arxitekturasına öz assembler dili uyğun gəlir və həmin dildə yazılan proqram ancaq bu mühitdə istifadə oluna bilər. Aşağı səviyyəli dilləri adətən kiçik sistem tətbiqlərinin yazılması üçün istifadə olunur. Məsələn,qurğuların, drayverlərin, qeyri-standart avadanlıqla birləşmə modulların və s. Bu zaman ən vacib tələblərdən biri proqramların çox sürətli olması,konkret olmasi,aparat resurslarına birbaşa müraciət üsulları nəzərdə tutulur. Yüksək səviyyəli proqramlaşdırma dilləri insane daha yaxın və aydındır. Konkret computer arxitekturası burada nəzərə alınmır. Buna gorə də yaradılan proqramlar ilkin mətnlərin səviyyəsindən cox asnlıqla başqa kompüter arxitekturasına (platformasına) keçirilir.

2.Proqramlaşdırma dilləri realizə üsullarına görə təsnif olunur.

Proqramlaşdırma dilləri kompilyasiya və interpretasiya olunan kimi realizə olunan kimi realizə olunur. Ümumiyyətlə tərcümə proqramlarının ümumi adı translyatorlardır. İş prinsiplərinə görə 2 qrupa bölünür: kompilyator və interpretatorlar. Kompilyator yüksək səviyyəli proqramlaşdırma dilində yazılan ilkin mətni prosessorun ikili kodlarına çevirir. İnterpretator birbaşa ilkin mətni tərcüməsiz yerinə yetirir (interpretasiya edir). Proqram isə yazılmış proqramlaşdırma dilində qalır və interpretatorsuz baxıla bilməz. Kompilyasiya edilmiş proqramlar daha sürətli icra olunur və onların icrasında əlavə proqramlar tələb olunmur , çünki artıq onlar maşın dilinə çevriliblər.Kompilyator proqramin mətnində sintaksik analiz aparır (bəzən bir neçə dəfə) və müəyyən sintaksis analizi icra edir. Nəticədə maşin kodu generasiya olunur. Çox vaxt kompilyasiya prosesində optimallaşdırma da tətbiq olunur.Məsələn, müəyyən tip prosessora yönəldilən instruksiyalar vasitəsi ilə lazım olmayan əmrlər və aralıq hesablamalar çıxarılıır. Nəticədə tamamlanmış proqram effektiv və kompakt alınır və interpretasiya alınan proqramdan dəfələrlə sürətli işləyir. Amma bu üstün cəhətlərə baxmayaraq proqramın mətnini hər dəfə dəyişdikdə onu yenidən kompilyasiya etmək lazımdır.Bu isə işləmə prosesini ləngidir. Kompilyasiya olunmuş proqram eynitipli komputerlərdə və eyni əməliyyat sistemində yerinə yetirilə bilər.İnterpretasiya olunan proqramlar nisbətən aşağı sürətlə yerinə yetirilir. Bundan başqa onlar interpretatorsuz yerinə yetirilə bilməzlər. Kompilyatorun şaxələnmiş modelinə baxaq.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Leksik analizator** | leksemalar | **Sintaksik analizator** | kompilyaturun  daxili kodu | **Kodun generatoru** |

yüksək səviyyəli

dildə proqram

|  |
| --- |
| **C ə d v ə l l ə r** |

maşın kodu

1.Leksik analizator.

Hər bir proqramlaşdırma dilində yazılmış proqram sözlərdən və onları bir-birindən ayıran simvollardan ibarətdir. Kompilyasiya nəzəriyyəsində sözlər leksemalar adlanır. Məsələn: for, i, =, :, n, do və s. hamisi leksemalardir. Ayıran simvol kimi “probel”- boş yeristifadə olunur. Amma bəzi leksemalar özləri ayıran simvol kimi istifadə oluna bilər , məsələn “;”.

2. Sintaksis analizator.

Sintaksis analizator dilin qrammatik qaydaları əsasında cədvəllərin yazılışını yoxlayır və leksemaları kompilyatorun daxili koduna çevirir.Daxili kodların ardıcıllığı komputer əməliyyatlarının ardıcıllığını təsvir edir. Amma bu hələki tam maşın kodu deyil.Kompilyasiya nəzəriyyəsində kompilyatorun daxili kodlarının bir neçə növü var: tridalar,tetradalar, ağaclar, atributlu ağaclar, p-kod.

3.Kodun generatoru.

Generator kompilyatorun daxili kodunukomputerin tam maşın koduna çevirir.

4.Cədvəllər.

İş vaxtı kompilyatorun bütün blokları ümumi cədvəllərə müraciət edir. Ümumi cədvəllərdə transilyasiya üçün ümumi məlumat yerləşdirilir. Məsələn, açar sözlərin cədvəli və hər bir proqrama uyğun olan individual məlumat, məsələn, identifikatorlar cədvəli. Bəzi proqramlaşdırma dilləri, məsələn, Java, C# kompilyasiya və interpretasiya proseslərinin arasında yerləşir. Bu dillərdə yazılmış proqram maşın koduna yox, maşından asılı olmayan aşağı səviyyəli koda kompilyasiya olunur. Bu kod byte-kod adlanır. Her emeliyyatın kodunun uzunluğu 1 bytedir. Hər instruksiya 0-dan 255-ə qədər 1 byteli koddur. Kodun arxasıyla registrlər və ya yaddaşın ünvanları yazılır. Byte-kod virtual maşının vasitəsi ilə yerinə yetirilir və burada interpretasiya istifadə olunur. Java mühitində virtual kod JVM (Java Virtual machine ), C# da isə CLR (Comman Language Runtime) ilə yerinə yetirilir. Məsəslən:

class Demo {

public static void main (String args[ ] ) {

System.out.println(“This is my first Java program!”);

} }

Javada bütün kod class içində yerləşir. Əsas sinfin adı proqramın adı ilə eyni olmalıdır. İlk mərhələdə javac Demo Java komplyatoru buraxılmalıdır. Əgər kompilyasiya uğurlu keçsə, onda yeni fayl Demo.class faylı yaradılır. Bu faylın içində interpretatorun vasitəsi ilə yerinə yetirilən virtual kod yerləşir. Proqramı işə buraxmaq üçün Java Demo interpretatoru istifadə olunur.

**Mühazirə 10**

C dili UNIX əməliyyat sisteminin bir məhsulu ola­raq­ ya­ran­mışdır. UNIX əməliyyat sisteminin orijinal ilk versiyasını Bell Laboratoriyasında çalışan Ken Thompson tək yazmışdı və UNIX-in bu ilk versiyasını DEC PDP-7 adlı hesablama ma­şı­nın­da tətbiq etmişdir. DEC PDP-7 ilk mini kom­pu­ter­lər­dən bi­ri idi və 16 K-a bərabər yaddaşa malik idi.

Zamanının digər əməliyyat sistemləri kimi UNIX də as­sem­­­­bler dilində yazılmışdı. Assembler dilində yazılan proq­­ram­­­ları inkişaf etdirmək çox çətin və zəhmətli oldu­ğun­dan­­, Thom­pson UNIX əməliyyat sistemini inkişaf etdirmək üçün, ma­­şın dilindən daha yüksək səviyyəli bir dilə ehtiyac duydu. Bu məqsədlə kiçik bir proqramlaşdırma dili hazırladı. Öz dilini ha­zırlayarkən Thompson, 1960 illərin ortalarında Mar­tın Richards tərəfindən inkişaf etdirilmiş BCPL dilindən is­ti­fa­də etdi (BCPL - Business Common Prog­ramming Lan­gu­age). Bu dil də CPL (Cambridge Prog­ram­ming Language) - dən törə­miş­dir. CPL-in qaynağı da ALQOL 60-dır.

Thompson inkişaf etdirdiyi bu dilin adını B qoydu. Den­nis­ Ritchie UNIX proyektinə qatılınca B dilində proq­ram­laş­dır­ma­­ya başladı. B dil daha da inkişaf etdirildi və artıq daha ye­ni tex­­nologiya olan PDP-11 kompüterlərində işlədildi. Thom­pson UNIX əməliyyat sisteminin bir qismini B dilində təkrar yaz­dı. Artıq 1971-ci ildə B dilinin PDP-11 kompüterləri və UNIX əmə­liyyat sisteminin inkişaf etdirilməsi üçün çox uy­ğun olma­dı­ğı ortaya çıxdı. Buna görə Ritchie B proq­ram­laş­­-dı­r­­ma dilinin da­ha yeni bir versiyasını inkişaf etdirməyə başladı. O, bu dili əv­­vəlcə NB (new B) adlandırdı. Amma inkişaf etdirdiyi dil B dilindən qopmağa və ayrı bir xarakter göstərməyə başla­yın­­ca dilin adını da C olaraq dəyişdirdi. 1973-cü ildə UNIX əmə­liyyat sisteminin böyük bir qismi C di­li­­­ ilə təkrar yazıldı.

C-nin təkamülü və inkişafı 70-ci illərdə də davam etdi. Ge­­niş kütlələr tərəfindən tanınması və istifadə edil­mə­yə­­ baş­lan­­ma­sı 1978-ci ildə Dennis Ritchie və Brian Ker­ni­gh­­an tə­rə­fin­dən yazılan "The C Programming Language" ki­ta­bı ilə ol­muş­­dur. Bu kitab eyni zamanda proqram möv­zu­sunda yazılan ən yax­­şı əsərlərdən biri olaraq qiymət­lən­di­ril­miş­dir.

C-nin artan məşhurluğu bəzi problemləri də özü ilə gə­tir­di. Proqramla məşğul olanlar istinad olaraq Ritchie və Ker­ni­gha­nın kitabını əsas götürürdülər, amma kitabda bəzi his­sə­lər çox da detallı bir formada açıqlanmamışdı. Xüsusilə han­sı his­sə­lə­rin C dilinin bir xüsusiyyəti, hansı hissələrin isə UNIX əmə­liy­yat sisteminin bir xüsusiyyəti olduğu o qə­dər­ açıq olmadığı üçün bir sıra qarışıqlıqlar ortaya çıxırdı.

Artıq C dilinin standartlaşdırılması haradasa zəruri bir ha­la gəl­­mişdi! C-nin standartlaşdırılması işləri 1983-cü ildə ANSI (American National Standards Institute) - nin dəstəyi ilə baş­ladı. Edilən bir çox dəyişiklikdən sonra stan­dart­laş­dırma işləri 1988-ci ildə sona çatdı və 1989-cu ilin­­ dekabr ayın­da ANSI – nin C standartı rəsmi olaraq təs­diq­ləndi. 1990-cı ildə isə ISO/IEC 9899 - 1990 nömrəsiylə ISO (International Standards Organization) tərəfindən qəbul edildi. Stan­dart­laş­dır­ma ta­mam­lan­dıqdan sonra C yüksək səviyyəli bir sistem proq­­ram­laş­dır­ma dili halına gəlmişdir. İndiki vaxtda da sistem proqram­laş­dı­r­­­­ma­­nın (redaktorlar, əməliyyat sistemləri) çoxu C dili ilə ya­zıl­maq­dadır. Bu gün cib telefonlarından təy­ya­rə­lə­rə qədər hər yerdə C kodları istifadə olunur. Məsələn, Bo­e­ing təyyarələrində 100.000 sətirdən çox C kodu işlədildiyi məlumdur.

**C/C++ dilinin əlifbası**

Dilin əlifbası – bu dilə aid olan simvollar və “hərf­lər”­­ kül­­lisidir, onlardan dilin bütün konstruksiyaları düzəldilir.

Istənilən dildən istifadə edərkən bütün tələffüz olunan və ya­­zılanların əlifba konstruksiyası nöqteyi nəzərindən (yəni sin­taksis) korrekt olmasını və müəyyən məna daşımasını (yəni semantikaya malik olması) izləmək lazımdır.

Proqramın yazıldığı dildən maşın dilinə çevrilməsi və onun sintaksisinin analizi ilə proqram - kompilyasiya (trans­l­ya­­­­tor) məşğul olur.

Proqramlarda izahedici mətndən, yəni şərhlərdən istifadə etmək nəzərə alınıb. Şərhlər təyin olunmuş şəkildə tərtib olunur və proqramın gedişinə heç cür təsir göstərmirlər.

**C/C++**  dilində şərhlər /\*...\*/ simvolu ilə işarə olunurlar. Şərhlər birsətirli və çoxsətirli ola bilərlər. Məsələn,

/\*C və C++ dili\*/

və

/\*C və C++

dilləri\*/.

**C/C++** dili çoxsətirli şərhlər üçün **C/C++**  dilində işlədilən işarələri istifadə edir. Birsətirli şərhlər üçün isə iki maili xətdən istifadə edilir:

//C və C++ dili

*Qeyd.* **C/C++**  dilində birsətirli şərh işarəsindən də istifadə oluna bilər.

Identifikatorlar (və ya adlar) – proqrama daxil edilən bütün obyektləri işarə etmək üçün istifadə olunur. Obyekt dedikdə - də­yişənlər, sabitlər, tiplər, altproqramlar və s. başa düşülür. İdentifikatorlar latın hərfləri ilə başlayır və latın hərfləri, rə­qəm­lər, alt xətt işarəsinə malik ola bilərlər.

Açar sözlər – birqiymətli təyin olunmuş məna daşıyır və yalnız dildə verildiyi kimi istifadə oluna bilərlər. Açar söz­lər­dən proqramçının daxil etdiyi adlar kimi istifadə etmək olmaz. Standart ANSI C və C++ dilində 32 açar söz var:

*auto, break, case, char, const, continue, defaul, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, sta­­tic, struct, switch, typedef, union, unsigned, void, volatile, while.*

Yalnız C++ - a daxil olan açar sözlər isə bunlardır:

*asm, catch, class, delete, friend, inline, new, operator, private, protected, public, template, this, throw, try, virtual.*

Bəzi proqramlaşdırma dillərində açar sözlərin kiçik ya da böyük hərf olması fərq etmir. Amma C/C++ də bütün açar sözlər kiçik hərf olaraq təyin olunmuşdur. C böyük – ki­­çik hərf həssaslığı olan bir dildir (case sensitive). Digər proq­ram­laş­dırma dillərinin çoxunda böyük - kiçik hərf həssaslığı yoxdur (case insensitive).

Məsələn, biz istifadə edəcəyimiz bir dəyişənə "register" adı­nı verə bilmərik. Çünki bu bir açar sözdür. Amma biz istə­di­yi­miz bir dəyişənə REGISTER, Register, RegisTER və s. kimi adlar verə bilərik, çünki bunlar artıq açar söz sayılmırlar. Açar söz olan yalnız tamamilə kiçik hərf ilə yazılan "re­gis­ter"­-dir.

C/C++ dilinin əlifbasına aşağıdakılar daxildir:

1. A...Z – böyük latın hərfləri;
2. a .. z – kiçik latın hərfləri;
3. 0 .. 9 - ərəb rəqəmləri;
4. Ayırıçılar: ,(vergül), .(nöqtə), ;(nöqtə vergül), ?,!, : , \, /, ~, \_,%, +, =, \*, (, ), &, <, >, [, ], {, },’,#, ^, ;
5. probel işarələri: SP, H\_TAB, CR, LF, V\_TAB, FF,CTRL-Z (mətn faylının sonu);
6. xüsusi simvollar:

\n – yeni sətir;

\t – üfüqi cədvəl;

\v - şaquli cədvəl;

\r – barabanı(val) qaytarmaq;

\f – yeni səhifə;

\a – səs siqnalı;

\ ‘ - apostrof;

\ “ -ikiqat dırnaq;

\\ - əks sleş.

**C/C++ dilində proqramın quruluşu**

**C/C++**  dilində yazılmış proqram prinsipial xarak­ter­ daşı­ma­­­yan bəzi fərqli cəhətlərə malikdir. Aşağıdakı sadə proqram verilmiş sətri standart C-də çap edir.

**Misal**

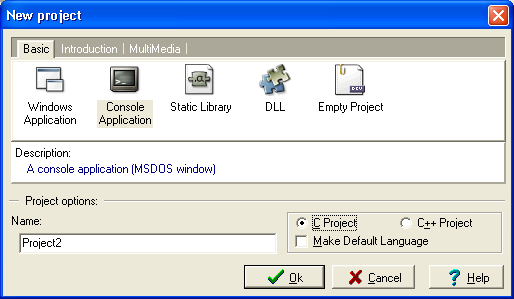
*# include <stdio.h>*

*void main()*

*{ printf(“ Azerbaycan, Baki\n”);*

*}*

Bu proqramı necə işə salmaq istifadə edilən sistem­də­n ası­lı­dır. Məsələn, proqramı Dev-C++-da buraxmaq üçün **File** men­yu­­sundan **New->Project**… altmenyusunu seçib, şəkildəki kimi açılan pən­­cə­rə­də **Console Application** və **C Project**  qeyd edib **OK** düyməsini sıxmaq lazımdır:



Bundan sonra, aşağıdakı kimi pəncərə açılır:

Yeni ad verib **Сохранить** düyməsini sıxmaq lazımdır. Indi isə proqramın mətnini yığmaq olar.

Proqramın mətnini yığıb qurtardıqdan sonra **Execute** men­yu­­sundan **Compile&Run** (və ya **F9**) altmenyusu seçilir. Faylın yadda saxlanılması tələb olunur. Ad verib **OK** düyməsini sıxdıqdan sonra ekranda *Azerbaycan, Baki* sözləri çıxarılacaq.

*#include*  direktivi *<stdio.h>* başlıq faylından (başlıq fay­­lı isə standart giriş- çıxış əməllərinə malikdir) ibarətdir.

*main()-* əsas proqramın adıdır, o eyni cür olub həmişə iştirak edir. *main* sözündən əvvəl gələn  *void* sözünün mənası ondan ibarətdir ki, proqram konkret qiymət qaytarmır;

*{ }*  - mötərizələri proqramın gövdəsini təşkil edir;

*printf()* – kiçik mötərizədə olan çıxış verilənlərinin çap funksiyasıdır.

Hər bir operator nöqtə - vergüllə qurtarır.

Proqramı C++-da yazsaq, əvvəldə olduğu kimi **File** men­yu­­sun­dan **New->Project**… altmenyusunu seçib, **Console Ap­pli­ca­tion** və **C++ Project** (proqramC++-da yazıldığı üçün**)** qeyd edib **OK** düyməsini sıxmaq la­zım­dır:

Yeni **Project-**ə ad verib, **OK** düyməsinisıxdıqdan sonra açılan pən­cə­rə­də proqramın mətnini yığmaq lazımdır:

*#include <iostream.h>*

*#include <conio.h>*

*using namespace std;*

*int main (int argc, char \*argv[])*

*{*

*cout <<”Azerbaycan, Baki\”;*

*getch();*

*return 0;*

*}*

C++-da çıxış üçün *cout* operatoru istifadə olunur.

**F9** düyməsini sıxdıqdan sonra ekranda yenə də *Azerbaycan, Baki* sözləri çıxarılacaq.

Başlıq fayllar funksiyanın prototipinə malik olurlar. Pro­totip kompilyatora funksiyanın sintaksisi haqqında mə­lu­­mat­ verir. Əgər başlıq faylını istənilən mətn redaktorla açsaq, on­da onun daxilində qoşulan funksiyanın prototipini təsvir edən sətir aşkar etmək olar.

Hesablama elementi daxil olan **C/C++**  proq­ra­mı­:

*# include <stdio.h>*

*void main()*

*{ int a, b, c;*

*a=5;*

*b=12;*

*c=a+b;*

*printf (“c=%d\n”,c);*

*}*

Proqramın ümumi quruluşu dəyişmədi, lakin bəzi əlavə­lər­ edildi.

*int a, b, c; -* tamqiymətli a, b, c dəyişənlərinin elanı;

*int-* dəyişənlərin tamlıq əlamətini göstərir.

*a=5;b=12;*  *a* və *b* dəyişənlərinə qiymət mənim­sə­dil­mə­si,

*c=a+b; a* və *b* – ni toplayıb nəticəni *c-* yə mənim­sə­dir.

*printf()* – funksiyası *%d*  konstruksiyasına malikdir, o, çe­vir­mə spesifikası adlanır. Onun yerində c dəyişəninin qiy­mə­ti duracaq, belə ki, % - yeri göstərir, d – qiymətin tipini (tam) gös­tərir. Çap sətri

*c=17*

şək­lində olacaq və kursor yeni sətirdə dayanacaq (\n- kur­so­ru yeni sətrə keçirir).

Adətən, proqramda giriş operatorundan da istifadə olunur.

***.*** ifadəsini hesablayan proqram.



*# include <stdio.h>*

*# include<conio.h>*

*void main()*

*{ float a, b, c;*

*printf ("b ve c qiymetlerini daxil edin:\n");*

*scanf ("%f%f ",&b,&c);*

*a=b/c;*

*printf("a=%f / %f=%2.1f\n" ,b,c,a );*

*getch();*

*}*

Burada *float a, b,c -* həqiqi tip üç dəyişənin elanı;

*scanf (“%f %f”, &b, &c)* isə *b və c* həqiqi ədədlərinin daxil olunma üsulunun təşkilidir.

Klaviaturadan ilk daxil edilən ədəd b-yə, növbəti isə c-yə mə­nim­sədiləcək.

*printf(“a=%f / %f=%2.1f \n” ,b,c,a)-* birinci  *%f* yerində *b* qiy­məti, ikincinin yerində *c,* üçüncünün yerində isə *a* qiy­mə­ti 2.1 formatı ilə çap olunacaq.

*getch*() isə ekranın saxlanılması prosedurasıdır, <*conio*.h> başlıq faylında təyin olunmuşdur.

Beləliklə, **C/C++** - proqramın quruluşu belədir:

1. *# include*  direktivi ilə başlıq faylları qoşulur;
2. istənilən proqram *main()* sözü ilə başlayır;
3. dəyişənin istifadə olunmasından əvvəl o, tipi gös­təril­mək­lə elan olunmalıdır.

Tərəfləri a, b, c olan üçbucağın perimetrini he­sab­­­­la­yın.

*#include <stdio.h>*

*void main()*

*{ int p,a,b,c;*

*scanf("%d%d%d",&a,&b,&c);*

*p=a+b+c;*

*printf("p=%d\n",p);*

*getch();*

*}*

***Qeyd 1.*** Ədədi qiymətlərin çapı zamanı aşağı­da­kı spesifi­ka­­tor­lardan istifadə olunur:

%i və %d– işarəli tam ədəd,

%u- işarəsiz tam ədəd,

%f – sürüşkən vergüllü kəsr ədəd,

%n,mf – sabit vergüllü kəsr ədədlər. Burada n- tam, m- kəsr hissənin rəqəmlərinin sayını göstərir.

***Qeyd 2****.* *printf* -lə yanaşı ekrana məlumatı çıxar­maq üçün *puts* funksiyasından da istifadə etmək olar. Bu zaman çap­­­dan sonra kursor avtomatik olaraq növbəti sətrin baş­lan­ğı­cı­na keçir.

***Qeyd 3.*** Proqram sona çatdıqdan sonra proqramın iş­lə­­di­yi pən­­cərənin digər pəncərə ilə örtülməməsi üçün (mə­sə­­­lən, Nor­ton­ Commander – in mətn redaktoru pəncərəsi ilə) proq­­ramın so­nuna

*print f(“ “);*

*getch ( );*

instruksiyasını əlavə etmək lazımdır.

**Mövzu № 11.**

Standart C-də giriş/ çıxış üçün *printf*  və *scanf*  işlə­di­­lir. C++-da isə uyğun olaraq *cout* və *cin* işlədilir.

***printf***  funksiyasının sintaksisi belədir:

*printf ( “idarəedici sətir”, [arqumentlərin siyahisi])*

Arqumentlərin siyahısı – sabitlər, dəyişənlər və ya ifa­də­lər ardıcıllığıdır; onların qiymətləri displeyin ekranına çı­xa­­rılır (ida­­rəedici sətrin formatı ilə). Məsələn,

*printf (“ i=%d, \n j=%d, a=% 6.2 f.\n” , i , j, a);*

Əgər

i=1234, j=127, a= 86.531 olarsa, ekranda

*i=1234,*

*j=127,* *a= 86.53*

görərik.

Idarəedici sətir 3 növ obyektə malikdir:

* adi simvollar ( ekrana dəyişdirilmədən çıxarılır);
* çevrilmə spesifikatorları (bunlardan hər birisi arqu­ment­lə­rin sonrakı siyahısından ekrana növbəti arqumentin qiymətinin çıxarılmasını yaradır);
* idarəedici simvol sabitlər.

Hər bir çevrilmə spesifikası % simvolu ilə başlayır və çevrilmə simvolu ilə qurtarır. Onların arasında aşağıdakılar yazıla bilər:

* “minus” işarəsi, onu göstərir ki, çıxarılan mətn sol sər­həd­lə bərabərləşir (susmaya görə sağ sərhədlə bəra­bər­lə­şir);
* çıxarış sahəsinin minimal ölçüsünü ve­rən rəqəmlər sət­ri;
* bölgü işarəsi olan nöqtə;
* çıxışın dəqiqliyini verən rəqəmlər sətri.

Çevrilmə simvolları aşağıdakılardır:

d- arqument onluq göstərilişə çevrilir;

o- arqument səkkizlik göstərilişə çevrilir;

x- arqument 16-lığa çevrilir;

c- arqumentin qiyməti simvoldur;

s- arqumentin qiyməti simvollar sətridir;

e- arqumentin qiyməti sürüşkən vergüllü formada float və ya double tipdir;

f- arqumentin qiyməti sabit vergüllü formada float və ya double tipdir;

g- f və ya e formatlarından biridir;

u- arqumentin qiyməti işarəsiz tam ədəddir;

p- arqumentin qiyməti göstəricidir (ünvan).

Məsələn, onluq tam ədədləri aşağıdakı üsullarla ekrana çıxara bilərik:

programın fraqmentinəticə:

*i = 5 ;*

*printf("%d",i) ; 5*

*printf("'i=%d",i) ; i=5*

*printf("i=") ;*

*printf("%d",i) ; i=5*

*printf("i=\n") ; i=*

*printf("%d",i) ; 5 alt sətrə keçir*

*printf('i=%d\n",i) ; i=5 alt sətrə keçir*

*printf("%d - %d",i, 5\*i); 5 - 25*

Simvolları ekrana çıxarmaq üçün:

*s = "ABCDEF"*

*printf("%10s ",s); . . . . A B C D E F*

*printf("%10.3s ",s); . . . . . . . A B C*

Həqiqi ədədləri isə belə formatla verə bilərik:

*x = 128.5 ;*

*printf("%6.2f",x) ; 128.50*

*x = 85.47 ;*

*printf("%6.3f",x) ; 85.470*

*printf("%4.1f",x) ; 85.5*

Beləliklə, “idarəedici sətir” arqumentin sayını və tipini təy­in edir. Aşağıdakı cədvəldə verilmiş idarəedici simvollar tez-tez­­ istifadə olunur:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | “16”-lıqda | “8”-likdə |  | ASCII |
| '\0' | '\x0' | '\0' | NULL simvolu | 0 |
| '\a' | '\x7' | '\07' | səs siqnalı (alert) | 7 |
| '\b' | '\x8' | '\010' | kursoru bir addım dala qay­­tarmaq (back space) | 8 |
| '\t' | '\x9' | '\011' | üfüqi cədvəlqurma (tab) | 9 |
| '\n' | '\xA' | '\012' | aşağı sətir (new line) | 10 |
| '\v' | '\xB' | '\013' | Şaquli cədvəlqurma (vertical tab) | 11 |
| '\f' | '\xC' | '\014' | Səhifə (form feed) | 12 |
| '\r' | '\xD' | '\015' | Sətir başı (carriage return) | 13 |
| '\"' | '\x22' | '\042' | qoşa dırnaq (double quote) | 34 |
| '\\' | '\x5C' | '\134' | əks sleş (back slash) | 92 |

***scanf*** funksiyası da *printf* funksiyası kimi təsvir olunur.

*scanf (“ idarəedici\_ sətir”,arqumentlər\_siyahısı);*

Arqumentlər üyğun qiymətlərə göstəricilər olmalıdır, bu­­nun üçün dəyişənin adının qarşısında & simvolu yazılır. İdarəedici sə­tir çevrilmə spesifikasiyalarına malikdir və ar­qu­mentlərin tipləri və sayların təyini üçün istifadə olunur (*printf* - ə anoloji olaraq).

*scanf(“ % d % f % c % s” , &i, &a, &ch, r);*

Burada r - simvollar sətridir, onun özü göstərici oldu­ğun­­­dan qarşısında & (ampersand) işarəsi qoyulmur.

İdarədəedici sətirdə probel, cədvəlqurma və yeni sətrə keç­­­mə simvolları istifadə etmək olar. Çıxışda onlar inkar edi­lir.

Əgər klaviaturadan daxil edilərkən verilənlər bölgü işa­rə­­­ləri (məsələn, vergül) ilə ayrılırsa, onda idarəedici sətirdə də­­ çe­virmə spesifikaları bu işarə ilə ayrılmalıdır.

C++ -da giriş/çıxış baytların axınıkimi icra olunur. Giriş/çıxışın geniş spektrdə emalı üçün *iostream* kitab­xa­na­sın­da çoxlu sayda funksiyalar və siniflər təyin olunmuşdur. Çıxış üçün istifadə olunan *cout* (the standard output stream - standart çıxış axını, *si-aut* kimi oxunur) funksiyası aşağıdakı kimi təs­vir olunur:

*cout<<dəyişən<<endl;*

burada << - *axına yerləşdirmə* əməliyyatı adlanır. *endl* (end line –“sətrin sonu” söz birləşməsinin abbreviaturasıdır)*-* axının manipulyatoru adlanır. “ ; “ isə operatorun sonunu ifadə edir.

Giriş üçün *cin* (the standard input stream - standart giriş axı­nı, *si-in* kimi oxunur) operatoru aşağıdakı kimi təsvir olunur:

*cin >>dəyişən;*

>> - *axından götürmə* əməliyyatı adlanır.

**Misal 4.1.** ifadələrini he­sab­­­la­ya­n proqram.

*#include <stdio.h>*

*#include<conio.h>*

*#define a 3.1*

*void main()*

*{*

*double s,v;*

*float R,r;*

*printf("R ve r- i daxil edin:\n");*

*scanf ("%f%f",&R,&r);*

*s=4\*a\*(R\*R+r);*

*v=4\*a\*(R\*R\*R-r\*r)/3;*

*printf("s=%4.3f\nv= %3.2f ",s,v);*

*getch();*

*}*

**Mövzu № 12.**

Proqramlar xətti, budaqlanan, dövri və mürəkkəb olur­lar.

Xətti proqramlar – operatorlar əvvəldən axıradək bir-bi­ri­­nin ar­dınca yerinə yetirilirlər (təkrar olunmayaraq və ye­ri­nə­ ye­tirilmə ardıcıllığını dəyişmədən).

Xətti proqramlara –mənimsətmə operatorları, riyazi funk­si­ya­lar, hesab əməlləri və giriş/çıxışla bağlı funksiyalar və ope­­­ra­torların ardıcıllığını pozmayan digər operatorlar da­xildir. **C/C++** dilində istifadə olunan bəzi riyazi funksiyalar Cədvəl də verilmişdir.

**Cədvəl . C/C++** dilində istifadə olunan riyazi funksiyalar.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Funksiya­nın adı | Funksiya­nın işarəsi | Tip | | Təsvir faylının adı |
| Qayta-rılan qiymət | Arqu-ment |
| mütləq qiymət  arkkosinus  arksinus  arktangens  kosinus  sinus  ex  xy qüv. f-sı  natural loq­­  onluq loq  kvarat kök  tangens | abs(x)  cabs(x)  fabs(x)  acos(x)  asin(x)  atan(x)  cos(x)  sin(x)  exp(x)  pow(x,y)  log(x)  log10(x)  sqrt(x)  tan(x) | int  double  float  double  double  double  double  double  double  double  double  double  double  double | int  double  float  double  double  double  double  double  double  double  double  double  double  double | <stdlib.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h>  <math.h> |

Ifadə - qiymətə malik olan proqram fraqmentidir: adətən, bu proqram buraxılan zaman hesablanmalı olan qiymətdir. Məsələn, . Bu halda ifadə literallardan(ədədlərdən), operatorlardan (/ və +) və kiçik mötərizədən ibarətdir. Mürəkkəb ifadələrdə mötərizələr iç-içə ola bilərlər (bu o deməkdir ki, bir alt ifadə digər altifadənin daxilində yerləşir). Məsələn, .

*Dəyişən = ifadə* dəyişənə ifadənin qiymətini mənimsətmək deməkdir. Bu za­man­ əvvəlcə ifadənin qiyməti hesablanır və bu qiymət sol tə­­rəf­dəki dəyişənin qiyməti olur.

Yadda saxlamaq lazımdır ki, “=” işarəsi mənim­sət­mə işa­rəsidir, bərabərlik işarəsi deyil. Mənimsətmə ifadə olur­. Bu qaydaya görə *i=(j=k)* da ifadə olur, belə ki, *j=k* ifa­­dədir və kon­kret qiymətə malikdir. Mötərizəni yazmamaq olar, onda *i=j=k* ifadəsi k qiymətini i və j dəyişənlərinə mənimsədir*, i=j=k=12* ifadəsi isə 12 qiymətini hər üç dəyişənə mənimsədir.

**İfadələr**

Ifadə - qiymətə malik olan proqram fraqmentidir: adətən, bu proqram buraxılan zaman hesablanmalı olan qiymətdir. Məsələn, . Bu halda ifadə literallardan(ədədlərdən), operatorlardan (/ və +) və kiçik mötərizədən ibarətdir. Mürəkkəb ifadələrdə mötərizələr iç-içə ola bilərlər (bu o deməkdir ki, bir alt ifadə digər altifadənin daxilində yerləşir). Məsələn, .

*Dəyişən = ifadə* dəyişənə ifadənin qiymətini mənimsətmək deməkdir. Bu za­man­ əvvəlcə ifadənin qiyməti hesablanır və bu qiymət sol tə­­rəf­dəki dəyişənin qiyməti olur.

Yadda saxlamaq lazımdır ki, “=” işarəsi mənim­sət­mə işa­rəsidir, bərabərlik işarəsi deyil. Mənimsətmə ifadə olur­. Bu qaydaya görə *i=(j=k)* da ifadə olur, belə ki, *j=k* ifa­­dədir və kon­kret qiymətə malikdir. Mötərizəni yazmamaq olar, onda *i=j=k* ifadəsi k qiymətini i və j dəyişənlərinə mənimsədir*, i=j=k=12* ifadəsi isə 12 qiymətini hər üç dəyişənə mənimsədir.

**Əməllər**

İfadələr qurularkən tətbiq edilən əməllərin növləri bunlardır:

* Hesabi,
* Münasibət,
* Məntiqi və mərtəbə,
* Inkrement və dekrement,
* Mənimsətmə.

**1)** **Hesabi əməllər***:*

\*- vurma;

/ - bölmə;

+- toplama;

- - çıxma;

%- tam bölmədə qalığı götürmə.

Bölmədən başqa bütün əməllər ənənəvi qaydada yerinə ye­ti­­­rilir. Bölmə əməlinin xüsusiyyəti ondadır ki, əgər hər iki­­ ope­rand tamdırsa, onda nəticə tam qiymət olur. Həqiqi qiy­­­mət al­maq üçün heç olmazsa bir operand həqiqi tipə ma­lik olma­lı­dır. Belə ki, 9/5=1; 9/5.0=1.8; 9%5=4.

Hesabi operatorların üstünlük dərəcəsi belədir:

a) yüksək: - (unar minus),

b) orta: \*, /, % (vurma, bölmə, qalığı götürmə),

c) aşağı: +, - (toplama, çıxma).

Yadda saxlamaq lazımdır ki, proqramlaşdırma dili­n­də bü­­tün əməl­ləri aşkar şəkildə göstərmək lazımdır. Bu, əsa­sən vur­ma əmə­l­inə aiddir, belə ki, riyaziyyatda onu yaz­ma­maq olar, proq­ram­laşdırmada isə yox! Məsələn, *ab* yazılışı proqramlaşdırmada iden­tfi­ka­tor ki­mi başa dü­şü­lür. Hesabi operatorlar binar opera­tor­lar­dır, çünki onlar iki operand üzərində əməliyyatı yerinə yetirirlər.

Aşağıda riyazi ifadənin C/C++ dilində yazılışına aid nü­mu­nə verilmişdir:

Riyazi ifadə: ;



C/C++ dilində yazılışı:



**2) Münasibət əməlləri** müqayisə üçün istifadə olunur:

< - kiçik;

<= - kiçik və ya bərabər;

> - böyük;

>= - böyük və ya bərabər;

= = - bərabər;

!= - bərabər deyil.

Məsələn,

*#include <stdio.h>*

*#include <conio.h>*

*main(){*

*int b;*

*b=5>2; // b doğru qiymət alır;*

*printf("b=%i\n",b);*

*b=7<4; // b yalan qiymət alır;*

*printf("b=%i\n",b);*

*getch();*

*}*

Münasibət, məntiqi və bitlər (mərtəbə) üzərində əməl­lə­rin nə­ti­­cəsi məntiqi tip olur. Lakin **C/C++** dilində *true* ­və *false* qiymətləri alan standart məntiqi tip yoxdur. Burada mən­­tiqi qiy­mətlər olaraq tamqiymətli qiymətlər istifadə olu­nur, belə ki, sı­fır yalan kimi, sıfırdan fərqli istənilən qiymət isə - doğru ki­mi başa düşülür.

Qeyd edək ki, “= =” münasibət işarəsini “=” mənimsətmə işarəsi ilə qarışdırmaq olmaz, belə ki, bu səhvə gətirib çıxara bilər.

**3) Məntiqi əməllərə** aşağıdakılar daxildir:

məntiqi *“və” (&&),*

məntiqi *“ və ya”* (||),

məntiqi inkar (!) .

Məntiqi əməllər bir neçə şərti ifadələri birləşdirdikdə isti­fadə olunur.

Məntiqi əməllər aşağıdakı xassələrə malikdirlər;

U1 və U2 – iki şərti ifadələrdirsə, onda:

* əgər hər iki ifadə eyni zamanda doğrudursa,  *U1&&U2* - doğ­ru­dur;

*6>2 && 7>5* - doğrudur;

*6>2 && 7<5*  - yalandır;

* əgər ifadələrdən heç olmasa biri doğ­ru­dur­sa, U1 || U2 -doğrudur;

*6>2 || 7>5* - doğrudur;

*6>2 || 7<5* - doğrudur;

*6<2 || 7<5* - yalandır;

* U1- yalan olduqda, !U1 - doğrudur;
* U1 - doğru olduqda, !U1 - yalandır;

*! (6>2)-* yalandır;

*! (6<2)-* doğrudur.

Mərtəbələr (bitlər) üzərində əməllər *float* və *double* tip­ dəyişənlərə tətbiq oluna bilməz. Belə əməllərə aşağıdakılar aiddir:

“ və” (&),

“ və ya “ (|),

mərtəbəli inversiya (-),

“ yoxetmə və ya “ (^),

sola yerdəyişmə (<<),

sağa yerdəyişmə (>>).

**Misal 3.2.**



**C/C++** dilində riyaziyyatda rast gəlinməyən iki əməl var:

inkrement (++) və dekrement (--). İnkrement əməliyyatı ope­ran­da 1 əlavə edir, dekrement isə operanddan 1 çıxır:

i++ ⬄ i=i+1;

i - - ⬄ i=i-1.

j=i++

yazısı

j=i; i++

yazısına ekvivalentdir;

j=++i

yazısı isə

i=i+1; j=i;

yazısına ekvivalentdir.

Məsələn***,*** *S=a + b ++;* yazısı o deməkdir ki, *a+b-* ni tapıb nə­ti­cə­ni S-ə mənimsətməli və onu 1 vahid artırmalı.

*S=a+++b;* yazısı o deməkdir ki, b-ni 1 vahid artırmalı, a və b – ni toplamalı, nəticəni S - ə mənimsətməli.

**C/C++** dilində mənimsətmə əməli adi haldan əlavə qey­­ri ənə­nə­vi göstərilişə malikdir. Məsələn, *x=x+15* əməlinin yeri­nə ye­ti­rilməsi x dəyişəninin ünvanının 2 dəfə hesablanmasına gətirir: il­kin qiymətin tapılması üçün və alınan nəticənin köçürülməsi üçün.

Bu çatışmazlığı x+=15 yazmaqla aradan qaldırmaq olar.

“+ =” simvolu da mənimsətmə əməli adlanır və o, kom­pil­ya­­­­tora x - ə 15 qiymətini əlavə edib nəticəni x-də yadda sax­la­maq göstərişi verir.

Mənimsətmə əməlinin tam yığımı:

\*=, /=, +=, -=, %=, <<=, >>=, &=, ^=, \=.

**Tiplərin çevrilməsi**

İfadələrdə eyni tipli operandlardan istifadə etmək əlve­riş­­lidir. La­kin **C/C++** dili tiplərin çevrilməsini mümkün sa­yır, yəni əgər operatorun operandları müxtəlif tiplərə ma­lik­­­­dirsə, onda onlar müəyyən ümumi bir tipə gətirilə bilər.

Gətirilmə qaydaları bunlardır:

1. avtomatik olaraq yalnız qiymətlər diapazonu kiçik olan­ operandları qiymətləri böyük olan operandlara çevirən tə­ləb­­lər yerinə yetirilir, belə ki, bu heç bir informasiyanın itmə­­məsi daxilində baş verir. Məsələn:

*int i;*

*float f, summa;*

*- - - - - - - - - - -*

*i = 5; f=2.5;*

*summa=i+f;*

*- -- - - - - - - - - -*

1. mənası olmayan ifadələr (məsələn, sürüşkən vergüllü ədəd­­­lər indeks rolunda) translyator tərəfindən hətta trans­lya­si­ya mərhələsində olduqda belə buraxılmırlar. Məsələn,

*float f;*

*- - - - - -*

*mas [f]=25; //* translyasiya səhvi verir;

1. informasiyanın itə biləcəyi ifadələr (məsələn, uzun tam­­la­­rın daha qısa tamlara mənimsədilməsində), xəbər­dar­lıq tö­rə­də bilərlər (warning), lakin onlar mümkündürlər. Məsə­lən,

*int i;*

*float f;*

*f=3.2;*

*i=f;*

İstənilən ifadə üçün tipin gətirilməsi adlanan unar opera­to­run­dan istifadə edərək aşkar şəkildə onun tipinin çev­­­­rilməsini gös­tərmək olar. Məsələn,

*(Tipin adı) ifadə*

şəklində konstruksiya ifadəni yuxarıda sadalanan qaydalar üz­rə göstərilən tipə gətirir. Məsələn,

(int) i= 2

**Mövzu № 13.**

**Şərt operatoru**

Ümumi formatı belədir:

1. əgər şərtin (ifadənin) d

*if (ifadə) operator;*

1. əgər belə operatorlar bir neçə­dir­sə:

*if (ifadə) { operator 1;*

*operator2;*

*- - - - - - - - - - - -*

*operator n;* }

Bu halda *“ifadə*” qiyməti tam qiymətə gətirilə bilən ixti­ya­ri ifa­də­dir. Əgər onun qiyməti sıfırdan fərqlidirsə (doğru­dur­sa), on­da *“ operator*” yerinə yetirilir.

İfadəni müəyyən qiymətlə müqayisə edərkən münasibət ope­ratorlarından (== və ya !=) istifadə etmək lazımdır.

*if (ifadə = = qiymət) operator;*

və ya

*if (ifadə != qiymət) operator;*

Burada operator sadə və ya mürəkkəb ola bilər.

***Qeyd 1.*** *if (ifadə !=0) operator;*

operatoru

*if (ifadə) operator****;***

operatoruna ekvivalentdir.

***Qeyd 2****. if (ifadə=qiymət) operator;*yazmaq olmaz, çünki = işa­rə­si mənimsətmə əməliyyatını göstərir.

Şərti göstərən ifadə kiçik mötərizədə yazılır. *if* opera­to­ru iki bu­da­ğa malikdir, onlardan biri alternativ olur. Belə sxe­m­ “şaxələnmə” adlanır. “Şaxələnmə” strukturunun təşkili üçün *else* operatoru istifadə olunur. O, *if* operatorunun genişlənməsi və ya onun alternativ hissəsidir:

* əgər *if* və *else* – dən sonra sadə (tək) operatorlar gəlir­sə:

*if(ifadə) operator1;*

*else operator2;*

* əgər operatorlar mürəkkəbdirsə:

*if (ifadə) {operator 1 ;*

*operator 2;*

*}*

*else {operator 3;*

*operator 4;*

*}*

*if* operatorunun daxili operatorları olaraq istənilən ope­ra­tor­lar, həmçinin şərt operatorları istifadə olunur:

*if(ifadə 1)*

*operator 1;*

*else if (ifadə 2)*

*operator 2;*

*else if (ifadə 3)*

*operator 3;*

*. . . .*

*else if (ifadə n)*

*operator n;*

*else // məcburi olmayan hissə*

*operator\_ susmaya\_ görə;*

Bu cür konstruksiyalarda *else* hissəsi əvvəlki ən yaxın *if* -lə əlaqələndirilir ( *else* hissəsi olmayan).

***Qeyd 3*.** *if – else* operatorunu onun funksional qiy­mə­ti­ni­ də­­­yiş­­­məyərək daha kompakt yazmaq olar. Bu halda o, şərt­ ifa­də­si adlanır:

*ifadə 1? ifadə 2; ifadə 3;*

Proqram *ifadə 1* – i hesablayır: əgər o, yalandırsa, *ifadə 2* ­ye­rinə yetirilir; əgər o da yalandırsa, *ifadə 3* yerinə yetirilir. Bu isə tamamilə

*if (ifadə1) ifadə 2*

*else ifadə 3*

sintaksisinə analojidir. Aşağıdakı proqram fraqmentləri ekvi­va­lent­dir.

. . . . . . . . . . . . . . . . .

*if (meas== “y”) value­=(meas==”y”)?­100:;*

*value= 100; . . . . . . . . .*

*else*

*value=0;*

*. . . . . . . .*

Lakin aydınlıq (əya­ni­lik) nöqteyi-nəzərindən şərt ope­ra­to­ru daha əlverişlidir.

**Misal .** Kvadrat tənliyin həlli üçün proqram.

*#include <stdio.h>*

*#include <stdlib.h>*

*# include "math.h"*

*int main()*

*{ float a,b,c, x1,x2, d;*

*printf("a=");*

*scanf (" %f", &a);*

*printf("b=");*

*scanf("%f",&b);*

*printf("c=");*

*scanf("%f",&c);*

*d=b\*b-4\*a\*c;*

*if (d<0) printf( " heqiqi helli yoxdur \n ");*

*else {*

*x1=(-b+sqrt(d)) / (2\*a);*

*x2=(-b-sqrt(d)) / (2\*a);*

*printf (" kokler: x1=%3.2f\n x2=%3.2f\n",x1,x2);*

*} printf ("\n <Enter>\n " );*

*return 0;*

*}*

**Seçmə (variant) operatoru**

Əgər proqramda çoxlu sayda variantdan birini seçmək la­zım­ gələrsə, onda mürəkkəb *if* əvəzinə *switch* seçmə opera­to­run­dan isifadə etmək məqsədəuyğundur. Onun sintak­sisi:

*switch (ifadə){*

*case qiymət1: operator 1;*

*break;*

*case qiymət 2: operator 2;*

*break;*

*case qiymət 3: operator 3;*

*break;*

*default :* }

**Seçmə (variant) operatoru**

Əgər proqramda çoxlu sayda variantdan birini seçmək la­zım­ gələrsə, onda mürəkkəb *if* əvəzinə *switch* seçmə opera­to­run­dan isifadə etmək məqsədəuyğundur. Onun sintak­sisi:

*switch (ifadə){*

*case qiymət1: operator 1;*

*break;*

*case qiymət 2: operator 2;*

*break;*

*case qiymət 3: operator 3;*

*break;*

*default :* }

Burada qiyməti ifadənin qiyməti ilə üst-üstə düşən vari­ant (operatorlar qrupu) yerinə yetirilir. Həm ifadə, həm də nişan (qiy­mət) tam və ya simvol tip olmalıdırlar.

*break* operatoru hər bir *case* blokunda *switch* opera­to­run­dan çıxışı təmin edir.

Əgər *break* operatoru yoxdursa, yəni *switch* operatoru

*switch (ifadə) {*

*case qiymət 1: operator 1;*

*case qiymət 2: operator 2;*

*case qiymət 3: operator 3;*

*} // end switch (ifadə),*

şəklində yazılarsa, onda nəticə “doğru” qiyməti verən ilk­ mü­qa­yisə halı olar.

**Misal** Ilin fəsillərini cap edən proqram:

*scanf("%d", &ay);*

*switch (ay) {*

*case 3:*

*case 4:*

*case 5: printf("yaz"); break;*

*case 6:*

*case 7:*

*case 8: printf("yay"); break;*

*case 9:*

*case 10:*

*case 11: printf("payız"); break;*

*case 12:*

*case 1:*

*case 2: printf("qış"); break;*

*}*

Qeydedəkki*, switch* operatoru əvəzinə  *if* operatoruisti­fa­­də edilə bilər. Ancaq *switch* operatoru proqramı daha oxu­naq­lı etdiyi üçün lazım olduğu hallarda istifadə edil­mə­si məslə­hət­dir.

***goto* operatoru**

*goto* operatoru nə şərt, nə də dövr operatorlarına aiddir, la­kin o da operatorların yerinə yetirilimə ardıcıllığının dəyişə və idarəetməni nişanla qeyd olumuş operatora verə bilər. *goto* opera­to­run­dan çox istifadə etməmək məsləhətdir!

*- - - -*

*goto nn;*

*- - -*

*nn: a[i] ==b[j]*

**Proqramın *exit* operatoru ilə dayandırılması**

Proqramı dayandırmaq üçün *exit()* funkiyasından istifadə etmək olar.

*exit (0)-* operatorunun yerinə yetirilməsi proqramı dərhal dayandrır, bütün açıq faylları bağlayır və bəzi digər yekun­laş­dı­rıcı əməlləri yerinə yetirir. () - mötərizəsi daxilində olan qiy­mət DOS - a və ya digər proqrama qaytarılır.

**Mövzu № 14.**

Proqramlşdırmada dövr dedikdə müəyyən əməllərin (ope­ra­­torların) avtomatik olaraq çoxlu sayda yerinə yetirilməsi pro­­­se­si başa düşülür.

C/C++ - da dövr – hər hansı bir şərtin yalan qiymət alana qə­­dər kodun yerinə yetirilməsini təkrarlayan strukturdur.

**C/C++**  dilində 3 növ dövr operatorundan istifadə olu­nur:­ önşərtli, son şərtli və parametrli.

Dövrlərin işləmə sayını bilmədikdə önşərtli və sonşərtli dövr operatorlarından istifadə etmək məqsədəuyğundur.

**Önşərtli dövr operatoru**

Bu operator təkrarolunan əməlləri o vaxta qədər yerinə ye­­ti­rir ki, verilmiş şərt doğru olsun. O, aşağıdakı şəkildədir:

*while (ifadə) operator;*

və ya

*while (ifadə){*

*operator1;*

*. . .*

*operator n;*

*} // end while*

Məsələn,

*int num=2; //başlanğıc qiymət*

*while (num<=1000)*

*num=2\*num;*

Əvvəlcə, *num* dəyişəninə 2 qiyməti mənimsədilir, dövr da­xi­lin­də bu dəyişən 2-yə vurulur və ardıcıl olaraq 4, 8, 16, 32, 64, 128, 256, 512, 1024 qiymətlərini alır. *num=1024* olduqda *while* strukturundakı *(num<=1000)* şərti yalan qiymət alır və dövr sona çatır. Son nəticə *num=1024* olur.

*while* operatorunun məntiqi strukturu Şəkil 5.1 blok-sxe­min­də təsvir olunmuşdur.

true

false

*while* dövr operatorunun blok-sxemi

**Sonşərtli dövr operatoru**

Bu operatorun sintaksisi belədir:

*do* və ya *do {*

*{operator;} operator 1;*

*. . .*

*while (ifadə); operator n;*

*} while (ifadə);*

Operatorun təyin etdiyi əməl *ifadə* yalan və ya sıfır ola­na­­dək davam etdirilir. Onun *while* operatorundan əsas fərqi on­dan ibarətdir ki, burada dövrün gövdəsi ən azı bir dəfə ye­ri­nə yetirilir. Çünki davam şərtinin yoxlanması döv­rün göv­də­si­ni əmələ gətirən operatorlardan sonra həyata ke­çi­rilir. Aşa­ğı­dakı proqram fraqmentində 1-dən 10-a qədər ədədlərin çap olun­ması təsvir olunmuşdur. Burada dövrün sonunu yoxlayan şərt­də prefiks formada inkrement istifadə olunmuşdur.

*int counter=1;*

*do {*

*cout<<counter<<” “;*

*} while (++counter<=10);*

*do/ while* operatorunun struktur blok-sxemi Şəkil 5.2-də göstərilmişdir.

true

false

*do/while* dövr operatorunun blok-sxemi

**Parametrli dövr operatoru**

*for* dövr operatoru – dövrün təkrarolunma sayı məlum ol­duq­da tətbiq olunur və ümumi sintaksisi aşağıdakı kimi­dir:

*for (ifadə1; ifadə 2; ifadə 3)*

*{ operator1;*

*operator2;*

*. . . . . . . . . . . .*

*operator n;*

*}*

*ifadə 1* –dövrün dəyişəninə başlanğıc qiymət verir, şərtin qiyməti təyin ediləndən əvvəl bir dəfə yeri­nə yetirilir, *ifadə2* dövrün davam etməsi üçün şərtdir. Əgər şərt lap başlanğıcdan yalandırsa, onda *ifadə2*(son ifa­də­) hesablanmaya da bilər.

*ifadə 3* – dövrün parametrinin dəyişmə qanununu təyin edir, ancaq onun olması vacib deyil. *for* operatoru C/C++ - da klassik operator deyil, yəni o tam olmayan da ola bilər. *for* operatorunu *while* operatoru ilə əvəz etdikdə aşağıdakı for­ma­da təsvir oluna bilər:

*ifadə 1;*

*while (ifadə 2) {*

*operator*

*ifadə 3;*

*}*

***for* operatoruna aid qeydlər**

***Qeyd 1.*** Elə etmək olar ki, hər hansı bir parametrin qiy­mə­ti ədədi yox, həndəsi silsilə ilə artsın, məsələn,

*for ( d=1; d<15; d=d\*2)*

*printf (“ d=%. 2f.\n” , d);*

***Qeyd 2****.* *for* dövründə bir və ya daha çox ifadəni yazmamaq olar (lakin bu zaman “;” işarəsini buraxmaq olmaz). Ancaq bu vaxt döv­rün gövdəsində onun işini sona çatdıracaq bir ne­çə ope­ra­to­ru yazmaq zəruridir. Məsələn,

*a=2;*

*for (n=3; a<=25;) a=a\*n;*

*//dövrün gövdəsi*

*for (;;)*

*printf (“ biz sonsuz dövr yazdiq \n”);*

sonsuz sayda yerinə yetiriləcək, çünki boş şərt həmişə doğru hesab olunur.

***Qeyd 3.*** Əgər dövrdə hər üç komponent boşdursa, onda belə dövr açıq dövr adlanır.

***Qeyd 4.*** *for* dövr operatorunda cəbri ifadələrdən də isti­fa­də etmək olar. Məsələn, fərz edək ki, x=2 və y=10. Əgər x və y dövr daxilində dəyişmirsə, onda

*for* ***(****int j=x; j<=4\*x\*y; j+=y / x )*

operatoru

*for (int j=2; j<=80; j+=5)*

operatoru ilə ekvivalentdir.

*for* operatorunun blok-sxemi *while*  operatorunun blok-sxemi ilə eynidir. Məsələn,

*for (int counter=1; counter<=10; counter++)*

*cout <<counter<<endl;*

operatoru üçün blok-sxem

Şəkil 5.3.-də verilmişdir.

counter=1

counter<=100

false

true

cout<<counter<<endl;

counter++

*for* dövr operatorunun blok-sxemi

**Misal** *Y=x sinx* funksiyasının qiymətini [*0, pi*] par­ça­­sında *pi/20* addımı ilə hesablamalı.

*#include <stdio.h>*

*#include <math.h>*

*#include <conio.h>*

*main() {*

*double x, dx = M\_PI/20;*

*printf ("y=x\*sin(x)\n x y\n --------------\n");*

*for (x=0; x<=M\_PI; x+=dx)*

*printf("\n%7.3f | %7.3f " ,x, x\*sin(x));*

*getch();*

*return 0;*

*}*

**Misal .**‘A’– dan ‘Z’-ə qədər dəyişən *simvol* para­met­rinin qiymətini çap etməli.

*#include <stdio.h>*

*main()*

*{ char simvol;*

*printf ( "elifbani cap etmeli\n");*

*simvol ='A';*

*while(simvol <='Z')*

*{ printf ("%c ", simvol);*

*simvol ++; }*

*getch();*

*return 0;*

*}*

**Misal .** Yuxarıdakı misalı sonşərtli dövr opratoru ilə yazaq:

*#include <stdio.h>*

*main()*

*{ char simvol;*

*printf("elifba\n ");*

*simvol='A';*

*do{*

*simvol++; printf("%c ",simvol-*1*);*

*} while(simvol<='Z');*

*getch();*

*return 0;*

*}*

**Misal 5.4*.*** Vurma cədvəli (iç - içə dövrlər).

*#include<stdio.h>*

*void main()*

*{ int i, j;*

*for (i=1; i<=10; i++) {*

*for (j =1; j<=10; j++)*

*printf ("%4.0d",i\*j);*

*printf("\n");*

*}*

*getch();*

*}*

***break* operatoru**

*Switch*  operatorunun bir atributu olan *break* operatoru proqramı dayandırmaq üçün müstəqil də işləyə bilər. *break* operatoru cari dövr qurtardıqdan sonra proqramın işinin davam etdirilməsinə xidmət edir və əgər dövrdən çıxdıqdan sonra proqramın qalan hisəsini yerinə yetirmək zəruridirsə, onda istifadə edilir.

Məsələn:

*for (;;) { …*

*if (ifadə)*

*break }*

Idarəedici ifadələr olmadıqda *for* operatoru “`addımla” yerinə yetirilir, ancaq dövrü olduqda *if* operatorundakı ifadə *break* operatorunu yerinə yetirilməyə məcbur edir və o dövrün “ sonsuz ” icrasını dayandırır.

*break*  açar sözü C++ dilinin bir hissəsi olsa da onu ehtiyatla istifadə etmək lazımdır. Belə ki, şərt yalan olduqda for dövrü bir çıxışa malik olur. Lakin bir və ya bir neçə *break* operatoru istifadə etdikdə, for dövrünün bir neçə çıxışı olur. Bu isə kodu daha qarışıq edir və məntiqi səhvlərə gətirə bilər.

***continue* operatoru**

*for* dövründə *continue* açar sözü istifadə etmək olar. *continue* – dövrün cari iterasiyasını qurtarmaq və dövrün növbəti iterasiyasına keçid üçün təyin olunub.

*continue*  açar sözünün C++ dilinin bir hissəsi olmasına baxmayaraq, onu da *break*  kimi ehtiyatla istifadə etmək lazımdır. Adətən *for* dövrünün hər iterasiyası bir çıxışa malik olur. Lakin *continue*  operatorundan istifadə etdikdə hər iterasiya bir neçə mümkün çıxışa malik olacaq. Bu isə kodu daha qarışıq edir və məntiqi səhvlərə gətirib çıxara bilər.

Məsələn: 10 müsbət elementin cəminin hesablanması

*continue* ilə *continue* olmadan

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

*int s= 0, i, a; int s= 0, i, a;*

*. . . . . . . . . . . . . . . . . . . . . . . .*

*for (i=0; i<0; i++) for (i=0; i<0; i++)*

*{ scanf (“ % d” , & a); { scanf (“ % d” , & a);*

*if (a<=0) continue; s+=a >0? a : 0;*

*s+=a; }*

*}*

**Misal** *break* və *continue*  operatorlarından birgə istifadə olunması:

*#include<stdio.h>*

*void main()*

*{ int i;*

*for(i = 1 ; i < 10 ; i++)*

*{*

*if (i == 5)*

*break;*

*printf ("%d\n", i);*

*}*

*for(i = 1 ; i < 10 ; i++)*

*{*

*if (i == 5)*

*continue;*

*printf("%d ", i);*

*}getch();*

*}*

**Misal.** 1-dən100-ə qədər ədədlər içərisində cüt ədədləri çapa verən proqram.

*#include <stdio.h>*

*main()*

*{*

*int x ;*

*for ( x=0 ; x<=100 ; x++) {*

*if (x%2) continue;*

*printf("%4d" , x);*

*}getch();*

}

**Mövzu № 15.**

**Massivlər**

Mənfi olmayan tam ədədlərlə indekslənmiş eyni tipli ele­ment­lər küllisi massiv adlanır. Massivlər birölçülü və çox­öl­çü­lü olurlar.

**Birölçülü massivlər**

Birölçülü massiv və ya vektor- eyni bir addan istifadə edən bir neçə eynitipli dəyişən olub hər bir dəyişənə girişi onun sıra nömrəsi ilə həyata keçirilir. Birölçülü massivi belə elan edirlər:

<*Elementlərin\_tipi>< massivin adı> [<elementlərin\_ sa­yı>];*

Bu zaman elementlərin sayı aşkar şəkildə verilməlidir, belə ki, kompilyatorun növbəti kompilyasiyası zamanı və sonra onu dəyişmək mümkün olmur. Məsələn,

*int i\_ arc [10];*

*char liter [80];*

*double d\_mas [100];*

Massivi elan edərkən yadda saxlamaq lazımdır ki, birinci elementin indeksi həmişə sıfra bərabərdir.

İndeksləri mümkün olmayan massivlərin elementlərinə müraciət olunan zaman massivə aid olmayan yaddaş sahəsinə müraciətlə əlaqədar səhvlər yaranır.

Birölçülü massivlərin emalı zamanı bir sıra tipik alqoritmlər istifadə olunur. Onlardan bir neçəsini verək.

**Massivin elementlərinin daxil edilməsi**:

*const int m=10 ;*

*void main( )*

*{ float x[m];*

*for ( int i=0; i< m; i++)*

*{ printf ( “ \ nx [%d] = “ , i);*

*scanf ( “ % f “, & x [ i ]);*

*}*

Massivin elementlərinin adlandırılması onun başlanğıc qiy­mətlərinin böyük mötərizədə göstərilməsi ilə həyata keçirilir.

*int digits [ ]= { 0,1, 2, 3, 4, 5, 6, 7, 8, 9}*

elanı avtomatik olaraq massivin ölçüsünü hesablayır və 10 elementdən ibarət massiv yaradır.

**Massivin adlandırılmas**ı:

*const int m=10;*

*void main()*

*{ double x[m]= { 12.2, 3.7, 4.2, 8.0,3, 1, 6.7, 9.1, 0.4, 5.5}*

Əgər adlandırma zamanı massivin az sayda başlanğıc qiy­­mətləri verilibsə (onların ümumi sayına nisbətən), onda qalanlar sıfır qiymətlərlə adlandırılırlar. Başqa sözlə,

*int digits [m]= {0, 1, 2, 3, 4};*

elanı 10 tam elementdən ibarət massiv yaradır ki, onlardan ilk beşi ≠0, qalan beşi isə =0 olur. Lakin kvadrat mötərizədə verilən saydan çox qiymət versək kompilyator səhv haqqında məlumat verəcək.

**Qeyd***.* Zəruri hallarda massiv-sabiti aşağıdakı kimi vermək olar:

*cons int const arr [ ]= {5,4,3,2,1};*

Onda massivin heç bir elementini dəyişmək olmaz.

**Misal .** Fibonaççi ədədi ardıcıllığının ilk 20 həddini hesab­la­yan proqram yazın.

*#include <stdio.h>*

*main()*

*{*

*int fib[20];*

*int i;*

*fib[0] = 0;*

*fib[1] = 1;*

*for (i=2; i<=19; i++)*

*fib[i] = fib[i-1] + fib[i-2];*

*clrscr();*

*for (i=0; i<=19; i++)*

*printf("%d . = %d\n", i, fib[i]);*

*}*

**Nəticə:**

*0. = 0*

*1.= 1*

*2. = 1*

*...*

*19. = 4181*

**Misal** . Massivlərdə ikilik axtarış alqoritminin proq­ra­mı.

*#include<iostream.h>*

*#include<iomanip.h>*

*#include<conio.h>*

*int ikilikaxtar(int [],int,int,int,int);*

*void printmassiv(int);*

*void printsetir(int[],int,int,int,int);*

*int main(int argc, char\* argv[])*

*{*

*const int massiv=15;*

*int m[massiv],key,netice;*

*for(int i=0;i<massiv;i++) //verilenlerin massivdə*

*//yerləşdirilməsi*

*m[i]=2\*i;*

*cout<<"0 - 28 arasinda axtarilan ededi daxil edin:- " ;*

*cin >> key;*

*printmassiv(massiv);*

*netice=ikilikaxtar(m,key,0,massiv-1,massiv);*

*if (netice !=-1)*

*cout<<endl<<key<<" Element " <<netice<<" nomreli indeksde tapilmishdir " <<endl;*

*else*

*cout<<endl<<key<<" tapilmamishdir "<<endl;*

*getch();*

*return 0;*

*}*

*//Ikilik axtarma*

*int ikilikaxtar(int b[],int seachkey,int low,int high,int size)*

*{*

*int middle;*

*while (low<=high){*

*middle=(low+high) /2;*

*printsetir(b,low,middle,high,size);*

*if(seachkey==b[middle])*

*return middle;*

*else if (seachkey<b[middle])*

*high=middle-1;*

*else*

*low=middle+1;*

*}*

*return -1;*

*}*

*void printmassiv(int size)*

*{*

*cout<<endl<<"Indeksler"<<endl;*

*for (int i=0;i<size;i++)*

*cout<<setw(3)<<i<<' ';*

*cout<<endl;*

*cout << "-------------------------------------------------------";*

*cout << endl;*

*}*

*void printsetir(int b[],int low,int mid,int high,int size)*

*{*

*for (int i = 0; i < size; i++)*

*if (i<low ||i>high)*

*cout<<" ";*

*else if (i==mid)*

*cout<<setw(3)<<b[i]<<'\*';*

*else*

*cout<<setw(3)<<b[i]<<' ';*

*cout<<endl;*

*}*

**İkiölçülü massivlər**

İkiölçülü massivlər ( matrislər) – vektorlar massividir və kvad­rat mötərizədə iki ədədlə verilirlər.

*Elementin\_ tipi massivin\_adı [ölçü 1] [ölçü 2];*

Burada *ölçü 1* – sətirlərin sayı, *ölçü 2* – sütunların sayı­dır.

Üçölçülü massivlər:

*Elementin\_tipi massivin\_adı [ölçü 1] [ ölçü 2][ölçü 3];*

Məsələn:

*double matre [100][10];*

*int i\_matrix [10] [20];*

*char cube [10][20][3];*

Kompüterin yaddaşında matrislər aşağıdakı ardıcıllıqla yer­ləş­dirilmiş oyuqları tuturlar: əvvəlcə I sətir, sonra II sətir və s.

Çoxölçülü massivlərin emalında bir-birinə daxil olan dövr­­­lərdən istifadə edilir. Bu vaxt ən yavaş sol kənar indeks, ən tez isə sağ kənar indeks dəyişir.

Çoxölçülü massivlərlə işləyərkən ən daxili dövr sağ kə­nar­, ən xarici dövr isə sol kənar indeksə uyğun gəlir.

Ikiölçülü massivlərin də emalı zamanı bir sıra tipik al­qo­ritm­­lər istifadə olunur. Onlardan bir neçəsini verək.

**Klaviaturadan tamqiymətli matrisin daxil edil­məsi**:

*const int max\_row=5;*

*const int max\_col=4;*

*. . . . . . . . . . .*

*for (i=0; i<max\_row; i++)*

*{ for ( j=0; j< max\_ col; j++)*

*{ printf (“ \ n x [%d][%d]”, i, j);*

*scanf (“%d, &x[i][j]); }*

*}*

**İkiölçülü massivin ekrana çıxarılması:**

*const int max\_row=5;*

*const int max\_col=4;*

*. . . . . . . . . . .*

*for (i=0; i<max\_row; i++)*

*{ for ( j=0; j< max\_ col; j++)*

*printf (“ %4d”, x [i] [j]” );*

*printf (“ \ n” ); }*

*}*

**Massivin maksimal elementinin hesablanması**:

*. . . . . . . . . .*

*int arr [n][k], max;*

*. . . . . . . . . . .*

*max=arr [0][0];*

*for (i=0; i<n; i++)*

*for (j=0; j<k; j++)*

*{ if (max< arr [i][j]) max=arr [i][j]; }*

**Matrisin hər sətrinin maksimal elementinin tapılması**:

*int arr [n][k], int max[n];*

*. . . . . . . . . . .*

*for (i=0; i<n; i++)*

*{ max[i]=arr [i][0];*

*for (j=0; j<k; j++)*

*if (max[i]< arr [i][j]) max[i]=arr [i][j]; }*

*}*

**Misal .** Matrisdə sıfırdan kiçik qiymətli ilk elementin yerini təyin edən proqram yazın.

*#include <stdlib.h>*

*main()*

*{*

*int a[10][10];*

*int i, j;*

*randomize();*

*for (i = 0; i<10; i++)*

*for (j = 0; j<10; j++)*

*a[i][j] = (rand() % 100) - 20;*

*for (i = 0; i<10; i++)*

*for (j = 0; j<10; j++)*

*if (a[i][j] < 0)*

*goto bul;*

*printf("Sıfırdan kiçik element yoxdur\n");*

*goto son;*

*bul:*

*printf("elementin mövqeyi və qiyməti (%d, %d) = %d \n",i, j, a[i][j]);*

*son:*

*return 0;*

*}*

**Misal . “Qabarcıq” üsulu ilə massivlərin nizam­lan­ması**

*#include<stdio.h>*

*#define n 1000*

*int main() {*

*int n, i, j;*

*int a[n];*

*scanf("%d", &n);*

*for(i = 0 ; i < n; i++) {*

*scanf("%d", &a[i]);*

*}*

*for(i = 0 ; i < n ; i++) {*

*//iki qonşu elementin müqayisəsi*

*for(j = 0 ; j < n - i - 1 ; j++) {*

*if(a[j] > a[j+1]) {*

*// əgər şərt doğrudursa,onda*

*// onların yerləri dəyişdirilir.*

*int tmp = a[j]; a[j] = a[j+1] ; a[j+1] = tmp;*

*}*

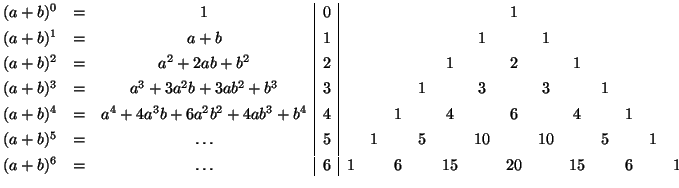
*}*

*}*

*}*

n elementli massivlərin nizamlanması üçün n-1 sayda əmə­liy­yat lazımdır.

**Misal.** **Binomial əmsalların hesablanması (Paskal üç­­bucağı)**



*#include <stdio.h>*

*#define N 1000*

*long c[N];*

*int main () {*

*long n, i, j;*

*scanf ("%ld",&n);*

*for(i = 1; i <= n ; i++) c[i] =0;*

*c[0] = 1;*

*for(j = 1 ; j <= n; j++)*

*for(i = j; i >= 1 ; i--)*

*c[i] = c[i-1] + c[i];*

*for(i = 0; i <= n ; i++)*

*printf ("%ld ", c[i]);*

*return 0;*

*}*

**Mövzu № 16.**

|  |
| --- |
| Siyahıda verilmiş elementlərin verilmiş ardıcıllıqda yerləşdirilməsi **çeşidləmə**(sorting)  adlanır. |

Çeşidləmənin bir neçə növü var (əlifbaya görə, tarixlərə görə və s.). Bütün bunlar bir- birindən elementlərin müqayisə proseduraları ilə fərqlənir. Ən sadə çeşidləmə üsulunu - massiv elementlərinin artmaya görə çeşidlənməsini nəzərdən keçirək.

### Hava qabarcığı üsulu

Bu üsulun adı məlum fiziki hadisədən götürülmüşdür: suda hava qabarcıqları yuxarı qalxır. Bu üsulda əvvəlcə massivin ən kiçik (“yüngül”) elementi “yuxarı” (massivin əvvəlinə) qalxır, sonra növbəti element və s.

Əvvəlcə axırıncı elementi axırıncıdan əvvəlki elementlə müqayisə edirik. Əgər onlar düzgün dayanmayıblarsa, onda onların yerlərini dəyişirik. Sonra növbəti cütlüyü götürüb həmin ardıcıllığı təkrar edirik. Axırıncı cütlüyü, yəni A[0] və A[1] elementlərini emal etdikdən sonra minimal element A[0]-ın yerində dayanacaqdır. Növbəti addımlarda bu elementi nəzərəalmırıq.

**Misal “Qabarcıq” üsulu ilə massivlərin nizam­lan­ması**

*#include<stdio.h>*

*#define n 1000*

*int main() {*

*int n, i, j;*

*int a[n];*

*scanf("%d", &n);*

*for(i = 0 ; i < n; i++) {*

*scanf("%d", &a[i]);*

*}*

*for(i = 0 ; i < n ; i++) {*

*//iki qonşu elementin müqayisəsi*

*for(j = 0 ; j < n - i - 1 ; j++) {*

*if(a[j] > a[j+1]) {*

*// əgər şərt doğrudursa,onda*

*// onların yerləri dəyişdirilir.*

*int tmp = a[j]; a[j] = a[j+1] ; a[j+1] = tmp;*

*}*

*}*

*}*

*}*

n elementli massivlərin nizamlanması üçün n-1 sayda əmə­liy­yat lazımdır.

Hava qabarcığı üsulu böyük massivlər üçün çox ləng işləyir. Massivin ölçüləri 10 dəfə artdıqda proqramın yerinə yetirilmə müddəti 100 dəfə artır (N2 dəfə). Təəssüf ki, bu ən sadə çeşidləmə üsullarınaaiddir.

### Minimal elementin seçilməsi üsulu

Hava qabarcığı üsulun çatışmayan cəhətlərdən biri yanaşı elementlərin tez-tez yerdəyişməsidir. Bu problemi həll etmək üçün *minimal elementin seçilməsi* üsulundan istifadə etmək lazımdır. Bu üsul aşağıdakından ibarətdir: massivdə minimal elementi təyin edib onu birinci yerə qoyuruq. Sonra qalan elementlər içindən minimal elementi tapıb, onu növbəti yerə qoyuruq və s.

Hava qabarcığı üsuldan fərqli olaraq burada yerdəyişmələrin sayı kifayət qədər azdır

(ən pis halda **N-1**). Proqramın icrası müddəti azalır.

|  |
| --- |
| #include <stdio.h> const int N = 10; main ()  {  int i, j, nMin, A[N], c;  // Burada A massivi daxil edilməlidir |

|  |
| --- |
| for ( i=0; i<N-1; i++)  {  nMin= i; // A[i]-dən başlayaraq minimal elementitəyinedirikfor ( j = i+1; j<N; j++)  if ( A[j] < A[nMin] ) nMin = j;  if ( nMin!=i) // əgər minimal element öz yerində dayanmayıbsa,…  {  c = A[i]; A[i] = A[nMin]; A[nMin]=c; // onu lazımı yerəqoyuruq  }  } |
| printf ( "\n Cheshidlenmish massiv: \n" );  for ( i=0; i<N-1; i++ )  printf ( "%d ", A[i] );  } |

**Mövzu № 17.**

İkiölçülü massivlər ( matrislər) – vektorlar massividir və kvad­rat mötərizədə iki ədədlə verilirlər.

*Elementin\_ tipi massivin\_adı [ölçü 1] [ölçü 2];*

Burada *ölçü 1* – sətirlərin sayı, *ölçü 2* – sütunların sayı­dır.

Üçölçülü massivlər:

*Elementin\_tipi massivin\_adı [ölçü 1] [ ölçü 2][ölçü 3];*

Məsələn:

*double matre [100][10];*

*int i\_matrix [10] [20];*

*char cube [10][20][3];*

Kompüterin yaddaşında matrislər aşağıdakı ardıcıllıqla yer­ləş­dirilmiş oyuqları tuturlar: əvvəlcə I sətir, sonra II sətir və s.

Çoxölçülü massivlərin emalında bir-birinə daxil olan dövr­­­lərdən istifadə edilir. Bu vaxt ən yavaş sol kənar indeks, ən tez isə sağ kənar indeks dəyişir.

Çoxölçülü massivlərlə işləyərkən ən daxili dövr sağ kə­nar­, ən xarici dövr isə sol kənar indeksə uyğun gəlir.

Ikiölçülü massivlərin də emalı zamanı bir sıra tipik al­qo­ritm­­lər istifadə olunur. Onlardan bir neçəsini verək.

**Klaviaturadan tamqiymətli matrisin daxil edil­məsi**:

*const int max\_row=5;*

*const int max\_col=4;*

*. . . . . . . . . . .*

*for (i=0; i<max\_row; i++)*

*{ for ( j=0; j< max\_ col; j++)*

*{ printf (“ \ n x [%d][%d]”, i, j);*

*scanf (“%d, &x[i][j]); }*

*}*

**İkiölçülü massivin ekrana çıxarılması:**

*const int max\_row=5;*

*const int max\_col=4;*

*. . . . . . . . . . .*

*for (i=0; i<max\_row; i++)*

*{ for ( j=0; j< max\_ col; j++)*

*printf (“ %4d”, x [i] [j]” );*

*printf (“ \ n” ); }*

*}*

**Massivin maksimal elementinin hesablanması**:

*. . . . . . . . . .*

*int arr [n][k], max;*

*. . . . . . . . . . .*

*max=arr [0][0];*

*for (i=0; i<n; i++)*

*for (j=0; j<k; j++)*

*{ if (max< arr [i][j]) max=arr [i][j]; }*

**Matrisin hər sətrinin maksimal elementinin tapılması**:

*int arr [n][k], int max[n];*

*. . . . . . . . . . .*

*for (i=0; i<n; i++)*

*{ max[i]=arr [i][0];*

*for (j=0; j<k; j++)*

*if (max[i]< arr [i][j]) max[i]=arr [i][j]; }*

*}*

**Misal.** Matrisdə sıfırdan kiçik qiymətli ilk elementin yerini təyin edən proqram yazın.

*#include <stdlib.h>*

*main()*

*{*

*int a[10][10];*

*int i, j;*

*randomize();*

*for (i = 0; i<10; i++)*

*for (j = 0; j<10; j++)*

*a[i][j] = (rand() % 100) - 20;*

*for (i = 0; i<10; i++)*

*for (j = 0; j<10; j++)*

*if (a[i][j] < 0)*

*goto bul;*

*printf("Sıfırdan kiçik element yoxdur\n");*

*goto son;*

*bul:*

*printf("elementin mövqeyi və qiyməti (%d, %d) = %d \n",i, j, a[i][j]);*

*son:*

*return 0;*

*}*

**Mövzu № 18.**

Təsəvvür edin ki, qar yağır. Tutaq ki, bu mənzərənin bir anını çəkirik. “Növbəti qar dənəsi hara düşəcək” suala dəqiq cavab vermək olarmı? Xeyr, çünki bu çoxlu sayda səbəblərdən asılıdır: hansı qar dənəciyi yerə daha yaxındır, külək necə olacaqdır və s. Demək olar ki, qar dənəciyi *təsadüfi yerə*düşəcəkdir, yəni onun yerini qabaqcadan söyləmək olmaz.

Kompüterdə təsadüfi proseslərin (qar dənəciklərin hərəkəti, molekulların Broun

hərəkəti) modelləşdirilməsi üçün **təsadüfi ədədlərdən** istifadə edirlər.

|  |
| --- |
| Əvvəlki ədədləri bilərək növbəti ədədi təyin etməyə mümkün olmayan ədədlər ardıcıllığı  **təsadüfi ədədlər** adlanır. |

Kompüterdə təsadüfi ədədləri almağa çox çətindir. Bəzən, bunun üçün radio küylərin müxtəlif mənbələrindən istifadə edirlər. Lakin riyaziyyatçılar daha universal və rahat üsuldan - **psevdo təsadüfi ədədlərdən** istifadə edirlər.

|  |
| --- |
| Təsadüfi ədədlərə yaxın xassələrə malik olan ədədlər ardıcıllığı **psevdo təsadüfi ədədlər** adlanır. Burada növbəti ədəd müəyyən riyazi düstur əsasında əvvəlki ədədlər əsasında hesablanır. |

Beləliklə, bu ardıcıllıq özünü təsadüfi ədədlər ardıcıllığı kimi aparır, baxmayaraq ki, düsturu

bilərək, ardıcıllığın növbəti ədədini hesablamaq mümkündür.

## Təsadüfi ədədlərlə işləyən funksiyalar

Təsadüfi ədədlərlə işləmək üçün C dilində bir neçə funksiya vardır. Onlar **stdlib.h** başlıq faylda təsvir olunublar. Bu o deməkdir ki, proqramın əvvəlində bu faylı qoşmaq lazımdır. Funksiyalar aşağıdakılardır:

**n =rand();** [0, **RAND\_MAX)** intervalında tamtəsadüfi

ədədin alınması ( RAND\_MAX=32767).

**srand (m);** təsadüfi ardıcıllığın başlanğıc qiymətini**m**-ə

bərabər etmək.

## Verilmiş intervalda təsadüfi ədədlərin alınması

Praktiki məsələlər üçün təsadüfi ədədləri verilmiş [a, b] intervalından almağa lazım gəlir. Əgər interval sıfırdan başlayırsa (a=0), onda qalığın tapılması əməliyyatından istifadə etmək olar: N-ə bölmə qalığı həmişə N-dən kiçikdir, yəni [0, N-1] intervalında yerləşir. Belə bir funksiya yazmaq olar:

|  |
| --- |
| int random ( int N )  {  return rand()%N; // [0, N-1] intervalında təsadüfiədəd  } |

Bu funksiya vasitəsi ilə [0, N-1] intervalında bərabər paylanmış təsadüfi ədədləri almaq mümkündür.

İndi isə, həmin funksiyanı [a,b] intervalı üçün istifadə edək. Aydındır ki, aşağıdakı düstur:

k = random (N) + a;

**[ a, a+N-1]** intervalında yerləşən təsadüfi ədədləri hesablayır. Belə ki, bizə**[a,b]** intervalı lazımdır, onda **b=a+N-1** və**N=b-a+1**. Ona görə də,

|  |
| --- |
| **[a,b]** intervalında bərabər paylanmış təsadüfi tam ədədlərin alınması üçün aşağıdakı düsturdan  istifadə etmək lazımdır:  k = random (b-a+1) + a; |

Daha mürəkkəb məsələ - təsadüfi həqiqi ədədlərin alınmasıdır. Əgər **rand()** funksiyasını **RAND\_MAX** ədədinə bölsək, onda nəticədə [0, 1) intervalında dəyişən təsadüfi həqiqi ədəd alacayıq:

### x = (float) rand() / RAND\_MAX;

Burada ədədlərin birini həqiqi tipə gətirmək lazımdır, çünki iki tam ədədin bir-birinə bölünməsi nəticəsində sıfır alınır.

**[0, 1)** intervalın uzunluğu 1-ə bərabərdir, bizdə isə intervalın uzunluğu **b-a** olmalıdır.

Əgər ədədi **b-a**-ya vursaq və nəticəyə**a** əlavə etsək, onda lazım intervalı alacayıq.

|  |
| --- |
| [a,b) intervalında bərabər paylanmış təsadüfi həqiqi ədədlərin əldə edilməsi üçün aşağıdakı düsturdan istifadə etmək lazımdır:  **x = rand()\*(b-a)/RAND\_MAX + a;** |

Bərabər paylanmış təsadüfi ədədlərdən qeyri-bərabər paylanmış ədədləri necə almaq olar? Riyazi statistikadan məlumdur ki, bir neçə bərabər paylanmış təsadüfi prosesin toplanması nəticəsindəədədlərin qeyri-bərabər (**normal vəya Qaus**) paylanması alınır.

**Mövzu № 19.**

Bu qrupa tam ədədlərlə işləyən klassik alqoritmlər daxildir.

### Evklid alqoritmi (I)

Evklid (3 əsir e.ə.) alqoritmi iki natural ədədin ən böyük ortaq bölənin (ƏBOB) hesablanması üçün nəzərdə tutulmuşdur. Orijinal Evklid alqoritmi aşağıdakı bərabərliyəəsaslanır:

Mənfi ədədlərlə işləməmək üçün alqoritmi aşağıdakı kimi ifadə etmək olar:

|  |
| --- |
| **Evklid alqoritmi**. İki ədədin ən böyüyünü böyük və kiçik ədədlərin fərqi iləəvəz edirik. Bu proseduranı iki ədədin bir-birinə bərabər olanına kimi aparırıq. Nəticədə alınmış ədəd – ƏBOB-dur. |

ƏBOB-ü hesablayan funksiyanı iki variantda yazmaq olar:

rekursiv qeyri-rekursiv

int EBOB( int a, int b)

{

if (a == b) return a; if ( a < b)

return EBOB (a, b-a); else

return EBOB (a-b, b);

}

int EBOB (int a, int b)

{

while ( a !=b ) {

if (a > b ) a -=b; else b -=a;

}

return a;

}

Qeyri-rekursiv alqoritm daha üstündür, çünki o, daha sürətlə işləyir və yaddaşdan (stekdən) az istifadəedir.

### Evklid alqoritmi (II)

Ədədlər bir-birindən çox fərqlənəndə birinci Evklid alqoritmi yavaş işləyir ( məsələn, ƏBOB(2, 1998) hesablanması üçün 998 addım lazım olacaqdır). Buna görə də, adətən modifikasiya olunmuş Evklid alqoritmindən istifadə edilir. Bu alqoritm aşağıdakı bərabərliyindən istifadə edir:

|  |
| --- |
| **Modifikasiya olunmuş Evklid alqoritmi**. İki ədədin ən böyüyünü böyük ədədin kiçik ədədə bölmə qalığı iləəvəz edirik. Bu proseduranı kiçik ədədin sıfıra bərabər olunana kimi aparırıq. Nəticədə alınmış ikinci ədəd – ƏBOB-dur. |

|  |
| --- |
| int EBOB (int a, int b)  {  while ((a !=0 ) && (b !=0)) { if (a>b) a =a%b;  else b=b%a;  }  return a+b;  } |

C dilində rekursiya olmayan variantı tərtib edək:

### Ən kiçik ortaq bölünən

|  |
| --- |
| Hər iki ədədə qalıqsız bölünən ən kiçik ədədə İki ədədin **ən kiçik ortaq bölünəni (ƏKOB)**  deyilir. |

### Eratosfen xəlbiri

Qədim Yunan riyaziyyatçılar tərəfindən həll olunan digər məsələ - **(1, N)** intervalında yerləşən bütün sadəədədlərin təyini ilə bağlıdır. İndiyənədək ən sürətlə işləyən *alqoritm Eratosfena* (275-195 il e.ə.) məxsusdur. Əfsanəyə görə, Eratosfen bütün natural ədədləri papirusda sıraya düzüb, hər ikinci ədədi deşirdi (yəni 2-ə bölünəni). Sonra o, hər üçüncü, dördüncü və s. ədədləri deşmişdi. Bu proseduradan sonra qalan “deşilməmiş” ədədlər sadəədədlərdir, yəni onlar yalnız özlərinə və 1-əbölünürlər.

Papirusunəvəzinə **A** massivindən istifadə edək. Bu massivdə**A[i]**elementi (

) iki mümükün olan qiymət ala bilir:

**1**, ədəd “deşilməyib” və sadəədədə namizəddir;

|  |
| --- |
| Yaddaşı qənaət etmək üçün tam tipli dəyişənlər əvəzinə (onlar yaddaşda 4 bayt yer tuturlar) simvol tipli (məsələn, unsigned char 1 bayt yer tutur) dəyişənlərdən istifadə etmək olar. |

**0**, ədəd “deşilib” və ona daha baxılmır.

**2**-dən √ -ə kimi bütün ədədləri nəzərdən keçirərək **2k, 3k, ...** və s., yəni **k**-ya bölünən ədədləri “aradan götürək”. Bu alqoritmi realizə edən proqram aşağıda göstərilmişdir. Qeyd edək ki, yaddaşda **N+1** elementə yerin ayrılması daha rahatdır, onda **A[1] ÷A[N]** elementlərdən istifadə etmək olur. **A[0]** elementi istifadə olunmur.

|  |
| --- |
| #include <stdio.h> |

|  |
| --- |
| main()  {  unsigned char \*A; int i, k, N;  printf( "Maksimal ededi daxil edin "); scanf( "%d", &N); |
| A = newunsignedchar[N+1]; // yaddaşda massivə yerayrılması  if ( A == NULL)return1; // səhv olduğu haldaçıxış |
| for ( i=1; i<=N; i++) A[i]=1;  for (k=2; k\*k <=N; k++ ) if ( A[k] !=0 )  for ( i=k+k; I <=N; i+=k) A[i]=0; |
| for (i=1; i<=N; i++)  if ( A[i] == 1 ) printf ( "%d\n", i);} |

Alqoritmin əsas **üstünlüyü -** onun yüksək sürətlə işləməsi, əsas **qüsuru** isə - yaddaşın böyük həcmindən istifadə edilməsidir. Bu alqoritm proqramlaşdırmanın əsas problemlərindən birini nümayiş edir: sürətin artırılması yaddaşın həcminin artırılmasına gətirib çıxardır vəəksinə, yaddaşın qənaəti sürətin azaldılmasına gətirib çıxardır. Real şəraitdə çox vaxt güzəştlərə gedirlər.

**Mövzu № 20.**

## Çoxmərtəbəli tam ədədlər

**int** tipli yaddaş xanasının birinə yerləşməyən çoxmərtəbəli tam ədədlərlə C dilində necə işləmək olar? Yəni bu ədədlər modula görə 2147483647-dan böyükdürlər. Şübhəsiz ki, beləədədlərə yaddaşda bir neçə xana ayrılmalıdır, lakin bu əməliyyatları əl ilə realizə etməli olacayıq. Burada 2 yanaşma var:

* + 1. ədəd simvol massivi şəkilində saxlanılır. Massivin hər elementi ədədin bir rəqəmini (09) təsviredir;
    2. ədəd tam tipli massivdə saxlanılır. Massivin hər elementi bir və ya bir neçə rəqəmi təsviredir.

İkinci yanaşmaya daha ətraflı baxaq. Yaddaşın qənaəti üçün bir xanada mümkün qədər çox rəqəm saxlamaq məqsədə uyğundur. Özü də nəzarət etmək lazımdır ki, aralıq əməliyyatların nəticələri seçilmiş verilənlər tipinin maksimal qiymətlərini keçməsin.

Misal üçün 12345678901234567890 ədədi nəzərdən keçirək. Bu ədədi aşağıdakı

kimi yazmaq olar:

Bu yazılış əsası **d=1000000** olan say sisteminin yazılışına uyğundur. Bu say sistemində rəqəmlər çoxmərtəbəli onluq ədədlərdir. Verilmiş ədədin yazılışı üçün **4 int** tipli yaddaş xanası lazım olacaqdır.

### 100! hesablanması

100 faktorialın qiymətini hesablayaq:

Faktorialı təqribən qiymətləndirib demək olar ki, onun qiyməti -dən kiçikdir, ona görə də onun yazılışında **201**-dən çox olmayan rəqəm var. Bundan əlavə, demək olar ki, ədədin sonunda **24** sıfır olacaqdır, çünki sıfırlar 10-a bölünən ədədlərə vurmaqla (10 dənə sıfır), 2 və 5 ilə bitən ədədlərin vurulması nəticəsində (10 dənə sıfır), 25, 50, 75 və 100 ədədlərinin cüt ədədlərə vurulması nəticəsində (4 əlavə sıfır)alınacaqdır.

Belə ki, 10! **8-**lə bitir (axırda yerləşən sıfırları nəzərə almasaq), onda 100! 4-lə bitir (810 ədədinin sonuncu rəqəmi 4-dür, çünki 82 4-lə, 85 8-ləbitir).

Say sisteminin əsasının düzgün seçilməsini yoxlayaq. Əgər **d=1000000**-dirsə, onda bu ədəd **int** tipli xanaya yerləşəcəkdir. Vurma zamanı ən böyük alınan ədəd 1000000·100 ola bilər, o da tam ədədlərə ayrılmış diapazona **(-231... 231-1**) yerləşir.

|  |  |  |
| --- | --- | --- |
| const int d = 1000000; int A[40] = { 1 };  int i, k, len = 1, r, s;  for ( k=2; k<=100; k++) i =0;  r = 0; | { | // d – say sisteminin əsasıdır  // A[0]=1, qalanlari A[i]=0  // len - ədədin uzunluğudur, r - qalıqdır  // 2,3, …, 100-ə vururuq  // ən kiçik (0) mərtəbədən başlayırıq  // əvvəlcə növbəti mərtəbəyə keçirilən ədəd r=0 |
| while ( i<len | // hələ ki bütün mərtəbələrə baxılmayıb | |

|  |
| --- |
| ||r>0) { // və ya növbəti mərtəbəyə keçirilən ədədvarsa  s=A[i]\*k+r; // mərtəbəni vururuq və üzərinə keçirilən ədəditoplayırıq  A[i]=s%d; // mərtəbədə qalanədəd  r= s/d; // növbəti mərtəbəyəkeçənədədi++; // növbəti mərtəbəyəkeçid  } |
| len=i; // ədədinuzunluğu  }  for (i=len-1; i>=0;i--) // ekranaxaricetmə  if (i==len-1)printf("%d", A[i]); // 123 ->123  else printf("%.6d", A[i]); // 123 ->000123 |

Qeyd edək ki, ən böyük mərtəbədən başqa uzun ədədin bütün mərtəbələri “%.6d” formatla ekrana çıxarılır. Bu format onu göstərir ki, ekrana mütləq 6 simvol çıxarılmalıdır vəəgər çıxarılan ədədin uzunluğu 6 simvoldan azdırsa, onda o, soldan sıfırlarla tamamlanır. Sağ tərəfdən olan sıfırlar həmişə yazılır.

**Mövzu № 21**

## Çoxhədlilər

### Qorner sxemi

verilmiş **x** üçün funksiyanın qiymətini hesablamaq üçün cəmi **n** toplama və **n** vurma əməliyyatı tələb olunur. Bu alqoritm **Qorner alqoritmi** adlanır və o, daha sürətlə işləyir. Aşağıda Qorner sxemini realizə edən və C dilində yazılmış proqram göstərilmişdir:

|  |
| --- |
| float Gorner (float x, float a[ ], int n)  {  float v=0.; |
| for ( int i=n; i>=1; i--) v=(v+a[i])\*x;  v +=a[0]; |
| return v;  } |

### Ardıcıllıqlar

|  |
| --- |
| Müəyyən qaydalara uyğun yerləşdirilmiş elementlər yığımına **ardıcıllıq** deyilir. |

Ədədlər ardıcıllığına baxaq. Ardıcıllığın bütün elementlərinə 1-dən başlayaraq nömrə verilir. Ardıcıllığı təsvir etmək üçün 2 üsuldan istifadə edirlər:

1. *Rekurent düstur*. Bu düstur əsasında **n** nömrəli elementi hesablamaq üçün əvvəlki elementlərin qiymətlərini bilmək lazımdır. Məsələn, 1,3,5,7,... ardıcıllığı

rekurent düsturla vermək olar.

1. *n-ci elementin düsturu*. Bu üsulda ardıcıllığın **n**-ci elementini hesablamaq üçün düstur verilir. Əvvəlki ardıcıllıq üçün . Düsturun sağ tərəfində yazılmış ifadə yalnız **n**-dən asılıdır. Məsələn, **ədədi silsilədə**n-ci element aşağıdakı düstur əsasındahesablanır:

Burada a1 – birinci element, d – fərqdir. **Həndəsi silsilə**üçün . Burada q –

silsilənin əmsalıdır.

Aşağıdakı cədvəldə ardıcıllıqların nümunələri göstərilmişdir:

|  |  |  |
| --- | --- | --- |
| **№** | **ardıcıllıq** |  |
| 1 | 1,3,5,7,... | 2n-1 |
| 2 | 1,3,7,15,... | 2n-1 |
| 3 | 2,9,28,65,... | n3+1 |
| 4 |  |  |
| 5 |  |  |

Ardıcıllıqlarla bağlı ən çox rast gəlinən məsələlər bunlardır:

* Verilmiş qiymətdən böyük olan azalan ardıcıllığın elementlərin sayını təyinedin;
* Müəyyən şərtə uyğun olan ardıcıllığın birinci elementini təyinedin;
* Ardıcıllığın müəyyən elementlərinin cəmini və ya müəyyən şərtə uyğun olan

elementlərin cəmini təyin edin.

|  |
| --- |
| **Misal**. Cədvəldə verilmiş 5-ci ardıcıllıq üçün:   1. 10-5 –dən böyük olan elementlərin cəmini təyinedin; 2. Cəmə daxil olan elementlərin sayını təyinedin; 3. 10-5-dən kiçik olan ən birinci elementi təyinedin. |

Bu cürə məsələlərin həlli zamanı alqoritmin səmərəliliyinə və dəqiqliyinə diqqət yetirmək lazımdır. Əgər hər addımda ədədləri bir-birinə vurmaq və 2n hesablamaq lazım gəlirsə, onda bu cürə proqram səmərəli olmayacaqdır.

Bunu başqa cürə etmək olar: 2\*n-1 və2n ifadə edən iki əlavə**u** və**d** dəyişənləri daxil edək. Hər addımda birinci dəyişən **2**-yə, ikinci dəyişən isə**2** dəfə artacaqdır. Bu məsələnin

həlli üçün aşağıda iki proqram göstərilmişdir. Onlardan biri digərindən 3 dəfə daha tez

işləyir. Hər iki proqramda cavab **s, n** və**a** dəyişənlərdə saxlanılır.

|  |
| --- |
| int n; float a, s; n=0; s=0;  while (1) { n++; |
| a=(2\*n-1)/(pow(2,n)-1); |
| if ( a < 1.e-5 ) break; s += a;  } |

|  |
| --- |
| int n;  float a, s, u, d; n=0; s=0; u=1;d=2;  while (1) { n++; |
| a = u/(d-1.); |
| if (a < 1.e-5) break; |
| u +=2; d \*=2; |
| } |

**Mövzu № 22**

### Rekurent ardıcıllıqlar

1202-ci ildə Fibonaççi adı ilə tanınan italyan riyaziyyatçı Leonardo Pizanskiy aşağıdakı məsələni təklif elədi:

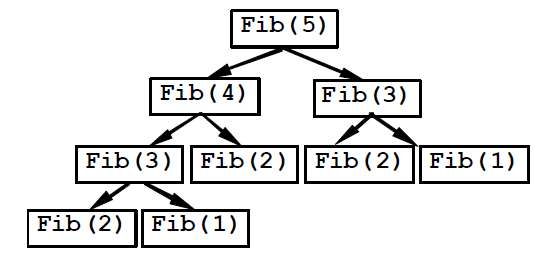
|  |
| --- |
| **Fibonaççi məsələsi**. Dovşanların bir cütü hər ay iki bala verir – bir erkək və bir dişi, hansılar ki, 2 aydan sonra yenə də iki bala verirlər və s. Soruşulur, bir ildən sonra dovşan cütlüklərin sayı nə qədər olacaqdır, əgər hal-hazırda 1 cütlük var? |

Hər ay ərzində dovşanların sayı aşağıdakı ardıcıllıq əsasında dəyişir:

1,1,2,3,5,8,13,21,34,...

Bu ardıcıllıq **Fibonaççi ardıcıllığı** adlanır. Bu ardıcıllıq **n-**ci elementi rekurent düsturla təyin edir. Düsturda n-ci element əvvəlki 2 element əsasında hesablanır. Hesabatı başlamaq üçün başlanğıc qiymətlər təyinolunmalıdır:

,

Bu cürə ardıcıllıq rekursiv funksiya vasitəsi ilə çox asanlıqla realizə olunur:

int Fib (int n)

{

if ( n < 3 ) return 1; else

return Fib(n-1) + Fib(n-2);

}

Lakin bu proqram səmərəli deyil, çünki, məsələn, **Fib(5)**-in hesablanması üçün funksiya 9 dəfə çağırılmalıdır, bu da səmərəsizdir.

Vəziyyətdən çıxış yolu elə funksiyanın yazılmasıdır ki, rekursiyadan yox, iterasiyadan (dövrdən) istifadə etsin. Qeyd edək ki, an elementinin hesablanması üçün **an-1** və**an-2** elementlərin qiymətləri lazımdır, yəni onları yadda saxlamaq lazımdır (proqramda onlar **a1** və**a2** kimi işarə edilmişdir). **a1** və**a2**-nin başlanğıc qiymətləri kimi, uyğun olaraq, **0** və**1** götürülmüşdür.

**Fibonaççi məsələsi**. Dovşanların bir cütü hər ay iki bala verir – bir erkək və bir dişi,

hansılar ki, 2 aydan sonra yenə də iki bala verirlər və s. Soruşulur, bir ildən sonra dovşan

cütlüklərin sayı nə qədər olacaqdır, əgər hal-hazırda 1 cütlük var?

Hər ay ərzində dovşanların sayı aşağıdakı ardıcıllıq əsasında dəyişir:

1,1,2,3,5,8,13,21,34,...

Bu ardıcıllıq **Fibonaççi ardıcıllığı** adlanır. Bu ardıcıllıq **n-**ci elementi rekurent

düsturla təyin edir. Düsturda n-ci element əvvəlki 2 element əsasında hesablanır. Hesabatı

başlamaq üçün başlanğıc qiymətlər təyin olunmalıdır:

,

Bu cürə ardıcıllıq rekursiv funksiya vasitəsi ilə çox asanlıqla realizə olunur:

Lakin bu proqram səmərəli deyil, çünki,

məsələn, **Fib(5)**-in hesablanması üçün

funksiya 9 dəfə çağırılmalıdır, bu da

səmərəsizdir.

Vəziyyətdən çıxış yolu elə funksiyanın yazılmasıdır ki, rekursiyadan yox, iterasiyadan

(dövrdən) istifadə etsin. Qeyd edək ki, an elementinin hesablanması üçün **an-1** və **an-2**

elementlərin qiymətləri lazımdır, yəni onları yadda saxlamaq lazımdır (proqramda onlar **a1**

və **a2** kimi işarə edilmişdir). **a1** və **a2**-nin başlanğıc qiymətləri kimi, uyğun olaraq, **0** və **1**

götürülmüşdür.

int n;

float a, s;

n=0; s=0;

while (1) {

n++;

a=(2\*n-1)/(pow(2,n)-1);

if ( a < 1.e-5 ) break;

s += a;

}

int Fib (int n)

{

if ( n < 3 ) return 1;

else

return Fib(n-1) + Fib(n-2);

}

int Fib ( int n )

{

int a2 = 1, a1 = 0, a, i;

for ( i=1; i<=n; i++)

{

a = a1 + a2; // növbəti elementi hesablayırıq

a2 = a1; a1 = a; // qabağa irəlləmə

}

return a;

}

}

while ( fabs(a) > 1.e-6 );

return s;

}

Fibonaççi ədədlər ardıcıllığının yaranması 1202-ci ildə italyan riyaziyyatçısı Fibonaççinin irəli sürdüyü məşhur bir

əyləncəli məsələnin həlli ilə bağlıdır[9,10]. Bu məsələ belə qoyulmuşdu:  
 "Doğulduqdan bir ay sonra dovşanlar cütləşir və daha bir ay sonra, yəni ikinci ayın tamamında bir cüt bala verməklə artırlar, dovşan ailəsi bir ilin sonunda neçə cüt canlıdan ibarət olacaq? Hesab olunur ki, bala dovşanlar da valideyinlərinə analoji artma qanunauyğunluğuna malikdir."

Bu məsələni iki üsulla həll etmək olar: 1) qrafik, 2) rekursiv funksiya üsuludur. Qrafik üsulla həllə baxaq:

Birinci cüt dovşan ikinci ayın sonunda bir cüt bala verir və o zaman dovşanların sayı iki cüt olur. Onlardan birinci cüt dovşanlar 3-cü ayın sonunda yenə bir cüt bala verir, deməli, üçüncü ayın sonunda 3 cüt dovşan olur. Sonrakı ay iki cüt dovşan doğulur ( hərəsi bir cüt ) və toplam 5 cüt dovşan olur. Bu proses şəkil 3.2-dəki sxemdə aşkar görünür. Şəkil 3.2-də k ilə kiçik, b ilə böyük cütlər təsvir olunub.

İndi isə Fibonaççi məsələnin riyazi həllinə baxaq.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | 0 | 1 | 2 | 3 | 4 | 5 | ... | 12 |
| F(n) | 0 | 1 | 1 | 2 | 3 | 5 | ... | 144 |

s qəbul etsək, Fibonaççi məsələsini həll edə bilərik:,...,F(12)=144

Bu məsələ sadə olsa da, çoxlu tətbiq sahələri vardır. Sonralar bu məsələ halı əlavə olunmaqla ümumiləşdirilmişdir. (3.11) düsturu Fibonaççi ədədlərini əks istiqamətdə qurmaq üçün nəzərdə tutulub.

*Fibonaççi ədədlərinin n-in müsbət və mənfi qiymətlərində düzülüşü*

|  |  |
| --- | --- |
| n | -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 ... 12 |
| F(n) | -8 5 -3 2 -1 1 0 1 1 2 3 5 ... 144 |

Fibonaççi ədədlərini həndəsi şərh etmək üçün müstəvidə tərəflərinin uzunluğu 1,1,2,3,5,8, və s. ədədlərinə bərabər olan kvadratları ardıcıl olaraq, saat əqrəbi istiqamətində (yaxud əks istiqamətdə) ortaq tərəfləri olmaqla qurub, bunların təpə nöqtələrini hamar əyrilərlə birləşdirsək, spiral fiquru alınar.

**Rekursiv obyektlər**

Rekursiyanın bir nümunəsi – öz təsvirini özündə saxlayan şəkildir.

Bir obyektin özü özündən təyin edilməsi **rekursiya** adlanır.

Riyaziyyatda rekursiya vasitəsi ilə çoxlu sayda sonsuz çoxluqları, məsələn, natural

ədədlərin çoxluğunu təyin edirlər. Həqiqətən, bunu belə etmək olardı:

**Natural ədəd:**

 1 – natural ədəddir.

 Natural ədədin ardınca gələn ədəd – natural ədəddir.

Faktorial anlayışını da (**n!=1****2****3****...****(n-1)** **n** )rekursiya vasitəsi ilə vermək olar:

**Faktorial:**

 0! = 1 (qəbul olunub).

 n! = n\*(n-1)!

**Rekursiv prosedura və funksiyalar**

Müasir proqramlaşdırma dillərində prosedura və funksiyalar özü özlərini çağıra

bilərlər.

Özü özünü çağıran prosedura və funksiyalar rekursiv adlanırlar.

Məsələn, faktorialı hesablayan funksiyanı aşağıdakı kimi yazmaq olardı:

int Factorial ( int n )

{

if ( n<=0 ) return 1; // 1 qaytarmaq

else return n\*Factorial(n-1); // rekursiv çağırış

}

Diqqət yetirsək, görmək olar ki, əgər n>0-dan, onda **Factorial** funksiyası özü özünü

çağırır. Bu məsələnin həlli üçün funksiyadan yox, rekursiv proseduradan da istifadə etmək

olardı. Rekursiv prosedura nəticəni yalnız istinad parametri vasitəsi ilə qaytara bilər (bu

parametrin qabağında **&** işarəsi dayanmalıdır). Proseduranın rekursiv çağırışı zamanı bu

parametr öz qiymətini dəyişəcəkdir.

void Factorial ( int n, int &fact )

{

if ( n == 0 ) fact = 1; // rekursiya bitdi

else

{ Factorial( n-1, fact); // rekursiv çağırış, (n-1)! hesablanır

fact \*=n; // n! =n\*(n-1)!

}

}

Funksiyadan fərqli olaraq prosedura istinadla göndərilən bir neçə parametrin

qiymətini dəyişə bilər.

**Dolayısı rekursiya**

Bəzən daha mürəkkəb konstruksiyadan istifadə edilir. Burada prosedura özünü

bilavasitə yox, digər prosedura və ya funksiya vasitəsi özünü çağırır. Bu konstruksiya

aşağıdakı sxemlə təsvir olunur:

Bu cürə rekursiya **dolayısı rekursiya** adlanır.

**Sonsuz rekursiya**

Rekursiv prosedura və funksiyalardan istifadəsi zamanı çox böyük təhlükə yaranır:

rekursiv çağırışlar sonsuz davam edə bilər. Ona görə də, belə funksiyalarda hər addımda

yoxlanılan şərti nəzərə almaq lazımdır. Əgər şərt yerinə yetirilmirsə, onda funksiyanın

çağırışları dayanır.

Faktorialı hesablayan funksiya üçün belə şərt **n<=0** – dır. Sübut edək ki, yuxarıda

göstərilmiş prosedurada rekursiya sonlu olacaqdır.

1. **n** sıfıra bərabər olduğu halda rekursiv çağırışlar dayandırılır.

2. Hər yeni rekursiv çağırışda **n**-nin qiyməti **1** vahid azalır (bunu proseduranın

çağırışından görmək olur: **Factorial (n-1, ...)**).

3. Ona görə də, əgər əvvəl **n>0** idisə, onda n-in qiyməti tədricən azalaraq, **0**-ra

çatacaq və rekursiya bitəcəkdir.

Rekursiv prosedura və ya funksiya rekursiyanı bitirən şərti özündə saxlamalıdır.

**Rekursiyadan nə vaxt istifadə etmək lazım deyil**

Hər yeni rekursiv çağırış zamanı kompüter aşağıdakı əməliyyatları yerinə yetirir:

1. Hər addımda hesablamaların vəziyyəti qeyd olunur.

2. Stekdə (yaddaşın xüsusi sahəsi) lokal dəyişənlərin **yeni** yığımı yaranır.

Belə ki, funksiyanın hər çağırışında yeni yaddaş ayrılır və ona vaxt sərf olunur,

rekursiyadan istifadə zamanı aşağıdakıları nəzərə almaq lazımdır:

**Rekursiyanın dərinliyi** (çağırışların sayı) kifayət qədər az olmalıdır.

Dərinliyi çox olan rekursiyadan istifadə edən proqram daha gec yerinə yetiriləcəkdir

və bu isə **stekin dolmasına** gətirib çıxara bilər. Ona görə də,

əgər rekursiyadan istifadə etməməklə məsələni asanlıqla həll etmək olursa, onda bu halda

rekursiyadan istifadə etmək məqsədə uyğun deyil.

Məsələn, faktorialın hesablanması məsələsi **for** dövrü vasitəsi ilə asanlıqla həll

olunur (bu cürə həll etmə **iterativ** və ya **dövrlü** adlanır):

int Factorial ( int n)

{

int i, fact = 1;

for ( i=2; i<=n; i++ )

fact \*=I;

return fact;

}

Yuxarıdakı funksiya rekursiv funksiyadan daha sürətli işləyir.

Sübut olunub ki, istənilən rekursiv proqram rekursiyadan istifadə etmədən yazıla bilər,

lakin bu realizasiya çox mürəkkəb ola bilər.

**Misal**. Fibonaçi ədədlərini ( *fi* ) hesablayan funksiyanı tərtib etmək lazımdır. Ədədlər

aşağıdakı kimi hesablanır:

1.

2.

Rekursiyadan istifadə edərək aşağıdakı funksiyanı yazmaq olar:

int Fib ( int n )

{

if ( n == 0 ) return 0;

if ( n == 1 ) return 1;

return Fib(n-1) + Fib(n-2);

}

Qeyd edək ki, hər rekursiya çağırışı **n>1** olduğu halda funksiyanın 2 çağırışına gətirib

çıxardır. Bəzi ifadələr, təkrar olaraq bir neçə dəfə hesablanır. Ona görə də, böyük **n**-lər

**17**üçün bu alqoritm faydalı deyil. **Fib(5)**-in hesablama sxemi ağac şəkilində aşağıdagöstərilmişdir.

Qeyd edək ki, növbəti Fibonaçi ədədinin qiyməti əvvəlki iki ədəddən asılıdır. Bu

qiymətləri **f1** və **f2** dəyişənlərdə saxlayaq. Əvvəlcən **f1=1** və **f2=0** qəbul etsək, sonrakı

Fibonaçi ədədini x-a yazaq. Növbəti addımda **f2**-nin qiyməti artıq lazım deyil və onun

yerinə **f1**, **x**-in qiymətini isə **f1**-ə köçürürük.

int Fib2 (int n )

{

int i, f1 = 1, f2 = 0, x;

for ( i=2; i<=n; i ++) {

x = f1 + f2; // növbəti ədəd

f2 = f1; f1 = x; // qiymətlərin sürüşdürülməsi

}

return x;

}

Bu cürə yazılmış funksiya böyük **n**-lər üçün (**>20** ) rekursiv funksiyadan xeyli sürətli

işləyir. Nəticə:

əgər rekursiyadan istifadə etmədən daha yaxşı nəticə almaq mümkündürsə, onda

rekursiyadan istifadə etmək gərək deyil.

**Rekursiyalı axtarış**

**Misal**. Təqdim edilmiş cümlədə verilmiş sözün neçə dəfə təkrar olunduğunu təyin etmək

lazımdır.

Rekursiv proseduradan istifadə etməklə, alqoritmi aşağıdakı kimi təsvir etmək olar:

1. **strstr** funksiyası vasitəsi ilə sözün birinci daxil olmasını axtarırıq. Əgər tapılmasa,

onda stop;

2. sözlərin sayı = 1 + cümlənin qalan hissəsində sözlərin sayı.**118**

int HowMany (char \*s, char \*word)

{

char \*p = strstr(s, word); // sözün birinci daxil olmasını axtarırıq

if ( !p ) return 0; // əgər söz tapılmırsa, onda nəticə - 0

return 1+ HowMany( p+strlen(word), word); // söz bir dəfə tapılıbsa, onda axtarışı davam edirik

}

Funksiya qısa və aydındır, lakin sürətə görə çox da səmərəli deyil.

**Mühazirə 23**

Müasir dövrdə informasiya cəmiyyəti inkişaf edərək getdikcə daha böyük vüsət alır. Kompyuterlər cəmiyyətdəki bütün proseslərə o cümlədən elmi−tədqiqat işlərinə, iqtisadiyyata təsir göstərərək, ümumilikdə insanın fəaliyyət formasını dəyişdirir və praktikanın yeni sahələrinə daxil olur. İnsanların yeni informasiya texnologiyalarını (kompyuterlər, proqram təminatı, kütləvi informasiya vasitələrinin yeni növləri) öyrənməsi və onu müxtəlif sahələrə tətbiq etməsi müasir yeni sistemlərin və proqramlaşdırma dillərinin yaranmasına və inkişafına səbəb olur.

Proqramçının bir proqramlaşdırma dili üzrə möhkəm biliyi olduqda, digərini öyrənmək ona çətin olmur. Bundan başqa, proqramçı çox vaxt əldə edilmiş biliklər sayəsində proqramın strukturunu daha asan başa düşür. Onu da qeyd etmək lazımdır ki, uzun müddət riyaziyyatın bir hissəsi hesab olunmuş informatikanın öyrənilməsi də proqramlaşdırmaya ən azı ona görə əsaslanır ki, konkret məsələnin həlli üçün kompyuterlərdə tətbiq edilməsinə məhz proqram imkan verir.

Hər proqramlaşdırma dili hərflər yığımı − əlifba ilə, qaydalar yığımı − sintaksis ilə xarakterizə edilir. Daha dəqiq desək proqramlaşdırma dili alqoritmin müəyyən qaydalar və vasitələrlə kompyuter üçün münasib formada təsviridir. Əksər proqramlaşdırma dilində latın əlifbasının hərflərindən, ərəb rəqəmlərindən, mötərizələrdən, hesabi əməliyyat işarələrindən, vergüldən və s. istifadə edilir. Proqram bu simvolların köməyilə yazılır.Proqramlaşdırma özündə fundamental elmləri, hər şeydən əvvəl informatikanı, riyaziyyatı və proqram mühəndisliyini birləşdirir. Başqa sözlə desək, proqramlaşdırma dili proqram yazmaq üçün istifadə olunan simvollar və işarələr sistemidir.

Hər proqramlaşdırma dilinin leksik, sintaksis, semantik qaydaları vardır. O proqramçıya kompyuterin hadisələrə necə reaksiya verəcəyini bilməyə imkan verir.

Proqramlaşdırmaya aşağıdakı mərhələlər aiddir:

* Layihələndirmə − kompleks şəkildə alqoritmin işlənməsi;
* Kodlaşdırma və translyasiya – proqramın ilkin mətninin yazılması və onun translyatorlar vasitəsilə yerinə yetirilən koda çevrilməsi;
* Testləşdirmə və sazlama – səhvlərin təyin edilməsi və aradan qaldırılması;
* Müşayiət .

Proqram məhsulları daima təkmilləşdirilir: yeni funksiyalar əlavə olunur, istifadəçi interfeysi

dəyişdirilir və s.

**Tеxnоlоgiya** sözü yunanca «tеchnе» (ustalıq, bacarıq) və «lоgоs» (öyrənmə, idrak) sözlərinin birləşməsindən yaranmışdır və istеhsal prоsеslərinin yеrinə yеtirilməsi üçün üsullar və vasitələr haqqında biliklər tоplusunun və həmin prоsеslərin özlərini ifadə еdir.

**Infоrmasiya tеxnоldоgiyası**- vеrilənlərin tоplanması, ötürülməsi və еmalı üçün mеtоd və vasitələrdən istifadə еtməklə tədqiq оlunan оbyеktin, prоsеsin, hadisənin vəziyyəti haqqında infоrmasiyanın (infоrmasiya məhsulunun) alınması prоsеsidir.

Matеrial istеhsalı tеxnоlоgiyasında məqsəd insanın və ya sistеmin tələblərinə cavab vеrən məhsulun istеhsalıdır. Infоrmasiya tеxnоlоgiyasında isə məqsəd insan tərəfindən analiz еdilmək və оnun əsasında qərar qəbul еtmək üçün infоrmasiya istеhsalıdır.

Məlumdur ki, еyni matеrial rеsurslarına müxtəlif tеxnоlоgiyaları tətbiq еtməklə müxtəlif məmulat və məhsul almaq оlar.Bu dеyilən infоrmasiyanın еmalı tеxnоlоgiyasına da aiddir.

**Infоrmasiya sistеmi** kоmpütеrlərdən, kоmpütеr şəbəkələrindən, prоqram məhsullarından, vеrilənlər bazalarından, insanlardan, müxtəlif növ rabitə vasitələrindən və s. ibarət оlan muhitdir. Infоrmasiya sistеmi, «insan-kоmpütеr» tipli infоrmasiya еmalı sistеmidir və burada əsas məqsəd infоrmasiyanın saxlanması, sоrğulara görə axtarıĢı və sеçilən isnfоrmasiyanı lazımi fоrmaya salıb, istifadəçiyə çatdırılmasıdır.

Infоrmasiya sistеminin funksiyalarının rеallaĢdırılması оna yönəlmiĢ infоrmasiya tеxnоlоgiyasını bilmədən mümkün dеyil. Infоrmasiya tеxnоlоgiyası isə infоrmasiya sistеmindən kənar оlaraq rеallaĢdırıla bilər.

Bеləliklə, infоrmasiya tеxnоlоgiyası infоrmasiya cəmiyyətində infоrmasiyanın çеvrilmə prоsеsləri haqqında müasir təsəvvürü ifadə еdən daha gеniş anlayışdır. Infоrmasiya sistеminin uğurla qurulmasının və fəaliyyətinin təminatı isə infоrmasiya və idarəеtmə tеxnоlоgiyalarından birgə və bacarıqla istifadə оlunmasıdır.