

Computer Science

Computer[kəm'pjutər] **Science**['saɪəns], the scientific study of the possible uses of computer.

Computer Science, the study of computers, including their design, operation, and use in processing information. Computer science combines both theoretical and practical aspects of engineering, electronics, information theory, mathematics, logic, and human behaviour.

Computer science is a combination of theory, engineering, and experimentation. In some cases, a computer scientist develops a theory, then engineers a combination of computer hardware and software based on that theory, and experimentally tests it. An example of such a theory-driven approach is the development of new software engineering tools that are then evaluated in actual use. In other cases, experimentation may result in new theory, such as the discovery that an artificial neural network exhibits behavior similar to neurons in the brain, leading to a new theory in neurophysiology. Experimentation and the traditional scientific method are thus key parts of computer science.

(Перевод Google)

Компьютерная наука научное исследование возможных вариантов использования компьютера.

Компьютерные науки, изучение компьютеров, в том числе их проектирования, эксплуатации и использования при обработке информации. Информатика сочетает в себе как теоретические, так и практические аспекты, электроники, теории информации, математики, логики и человеческого поведения.

Информатика представляет собой сочетание теории, инженерии и экспериментов. В некоторых случаях, ученый развивает теорию, то инженеры комбинацией компьютерного оборудования и программного обеспечения на основе этой теории, так и экспериментально проверяет его. Примером такого подхода теории управляемой является разработка новых технических средств, программного обеспечения, которые затем оцениваются в реальных условиях эксплуатации. В других случаях, экспериментирование может привести к новой теории, таким как открытие того, что искусственную нейронную сеть проявляет поведение, подобную нейронов в головном мозге, что приводит к новой теории в нейрофизиологии. Экспериментирование и традиционный научный метод, таким образом, ключевыми элементами компьютерной науки.

□ Perfect Posture

Small assaults on your muscles and tendons can take a big toll over time. How to stay limber :

- Sit at arm's length from monitor;
- Screen should be low enough so you don't have to flex your neck;
- Minimize reflection on screen;
- Keep elbows above keyboard;
- Keep wrists, hands, thighs parallel to floor.

Computer measurement units

Baud	max. number of state transitions per second	2400-56,600
Bps	Bits per second	Modems speed (14,400 – 56,000)
Cps	Characters per second	Printers speed (25- ...)
Dpi	Dots per inch	Printer resolution (300-1200)
MHz	Megahertz (millions of cycles per second)	CPU (Bas) speeds (12-1000 (GHz))
MIPS	Million Instructions Per Second	speed with which computer programs are ran
ms	Milliseconds	The average access time of HDD – the speed at which a hard drive finds data (9 - 14)
pin		A stiff prong in an electronic connector(port) (9 or 25)
pixel	Smallest element on the screen	Resolution of display, number of horizontal and vertical dots (640*480 ÷ 1,280*1,024)
Ppm	Pages per minute	Printers speed (4-20)
RPM	revolution per minute	speed of rotation(newer HD rotate at 7200 RPM)
W	Watts (joule per second) the rate at which electrical power being consumed	Speakers build - in amplifier (Watts=volts * amperes)

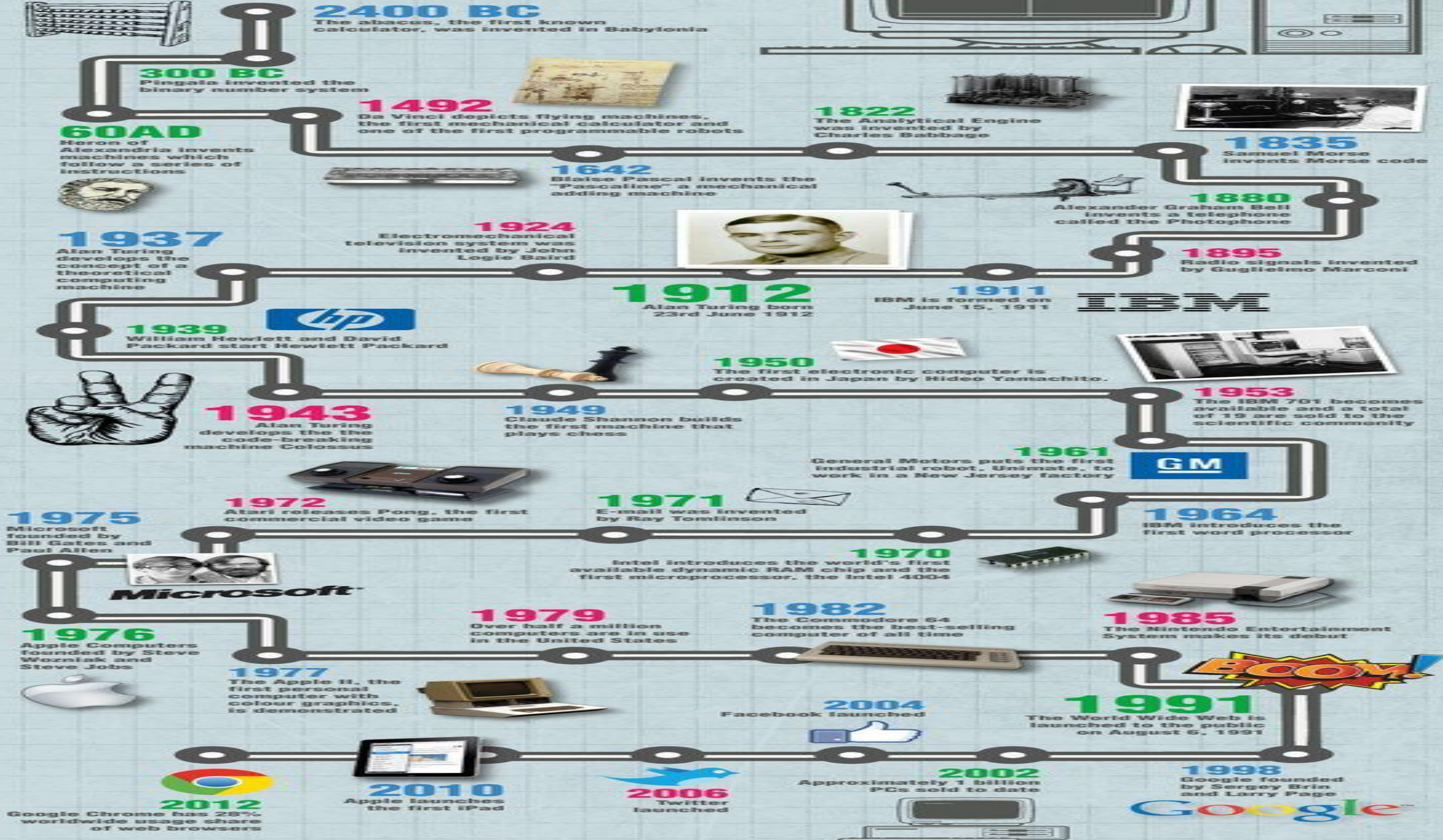
3D	TREE-DIMENTIONAL GRAPHICS
A.D.	Anno Domini (since Christ was born)
a.k.a.	Also known as
AI	ARTIFICIAL INTELIGENCE
AIFF	Audio Interchange File Format.
AMI	American Megatrends, Inc.
ANSI	American National Standards Institute
APL	A Programming Language
app	APPLICATION PROGRAM
ASCII	American Standards Code for Information Interchange
AT	Advanced Technology
ATM	1.Adobe Type Manager. 2.asynchronous transfer mode
AVI	Audio-Video Interleave
AWT	Abstract Window Toolkit (in Java)
B.C.	Before Christ
BBS	Bulletin Board System
BCC	Blind Carbon Copy.
BIOS	Basic Input Output System.
BMP	Bitmap
Bps	Bits per second
CAD	Computer-Aided Design
Cc	Carbon Copy
CGA	Color Graphics Adapter
CGM	Computer Graphics Metafile
CIO	Chief Information Officer
CORBA	Common Object Request Broker Architecture
DAC	DIGITAL-TO-ANALOG CONVERTER
DARPA	Defense Advanced Research Projects Agency
DIMM	dual inline memory module
DNS	Domain Name Server.
EDORAM	Extended dada out random access memory
EGA	Enhanced Graphics Adapter
EOF	End of fail
EOL	End of line
FAT	File Allocation Table
FAQ	Frequently asked questions
FTP	File Transfer Protocol.
GIF	Graphics Interchange Format

•Atto $10^{-18} = 0,00000000000000000001$

Nonillion $10^{30} = 10000000000000000000000000000000$

Numeric Prefixes

Prefix		Abbreviation	Pronounce	Multiplier	Prefix		Abbreviation	Pronounce	Multiplier		
	Atto	A	At-to	10^{-18}		Deca	Da	Dek-a	10^1		
	Femto	F	Fem-to	10^{-15}		Hecta	H	Hek-to	10^2		
	Pico	P	Pe-ko	10^{-12}		Kilo	K	Kil-o	10^3		
	Nano	N	Nan-o	10^{-9}		Mega	M	Meg-a	10^6		
	Micro	μ	Mi-kro	10^{-6}		Giga	G	Ji-ga	10^9		
	Milli	M	Mil-l	10^{-3}		Tera	T	Ter-a	10^{12}		
	Centi	C	Sent-ti	10^{-2}		Peta	P	Pe-ta	10^{15}		
	Deci	D	Des-l	10^{-1}		Exa	E	Ex-a	10^{18}		
		Multiplier In PC		Prefix	Abbreviation			Zetta	Z	sextillion	10^{21}
								Yotta	Y	septillion	10^{24}
										octillion	10^{27}
										nonillion	10^{30}
		2^{10}	Kilo(byte)	K(b)							
		2^{20}	Mega	M							
		2^{30}	Giga	G							
		2^{40}	Tera	T							
		2^{50}	Peta	P							



SPEAKERS

Used to produce audio output.

MONITOR

An output device that lets you see your work as you go.

CD/DVD DRIVE

Reads CD/DVD discs.

SYSTEM UNIT

The case that contains the CPU, memory, the power supply, disk drives, and all other hardware— such as a modem— that are in an internal format.

PRINTER

Produces printed copies of computer output.

MICROPHONE

Used to get spoken input.

FLOPPY DISKS

Used for storing small amounts of data for backup or to transport data to another PC.

KEYBOARD

The principal input device; used to type instructions into the computer.

CD/DVD DISCS

Commonly used to deliver programs and store large multimedia files.

MOUSE

A pointing device used to make on-screen selections.

FLOPPY DISK DRIVE

Reads from and writes to floppy disks.

HARD DRIVE

Located inside the system unit and used to store programs and most data.

FLASH MEMORY CARD READER

Used to read flash memory cards.

Power Connector

PS/2 Keyboard Connector

Serial COM Port

VGA Port

USB Ports

Microphone Jack
Audio Out Jack

Line In Jack

56Kbps RJ-11
Modem Ports

Power Supply Cooling Fan

PS/2 Mouse Connector

Thumb Screw
Parallel Port

Case Cooling Fan

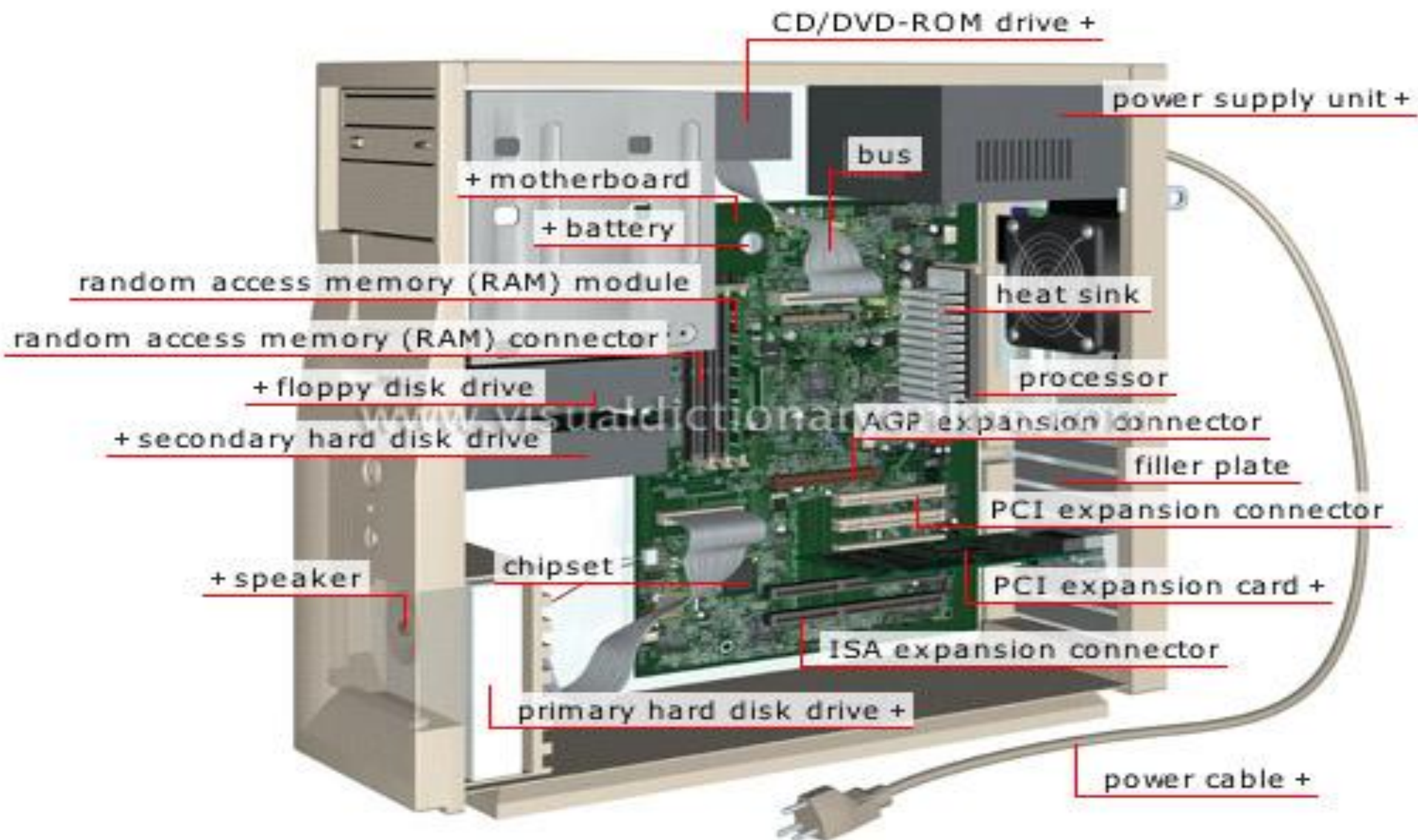
10/100Mbps RJ-45
Ethernet LAN Port

Security Lock Port

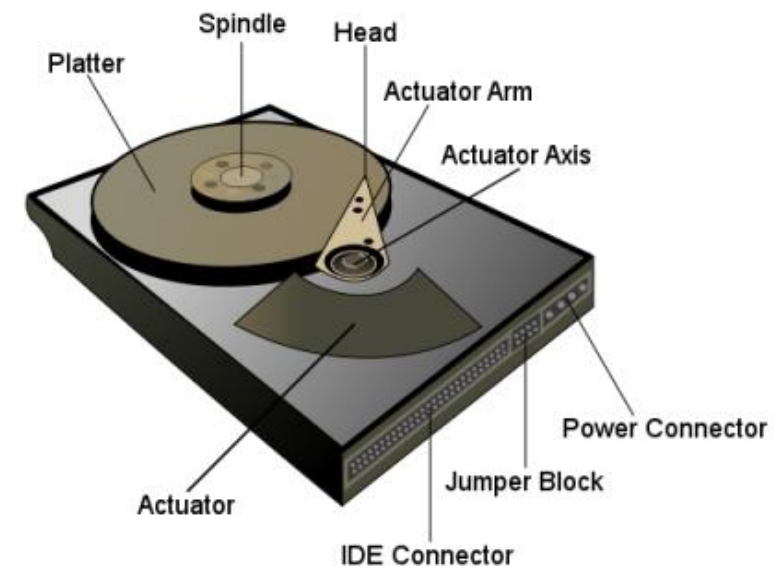
Expansion Slots

Thumb Screw





Ports



MOTHERBOARD ANA LOVHO



Multimedia Navigation Buttons
Function Keys
Calculator Button
Sleep Button

Web Navigation Buttons

Number Lock LED

Caps Lock LED

Scroll Lock LED

Scroll Wheel



Cut Button

Paste Button

Editing/Navigation Keys

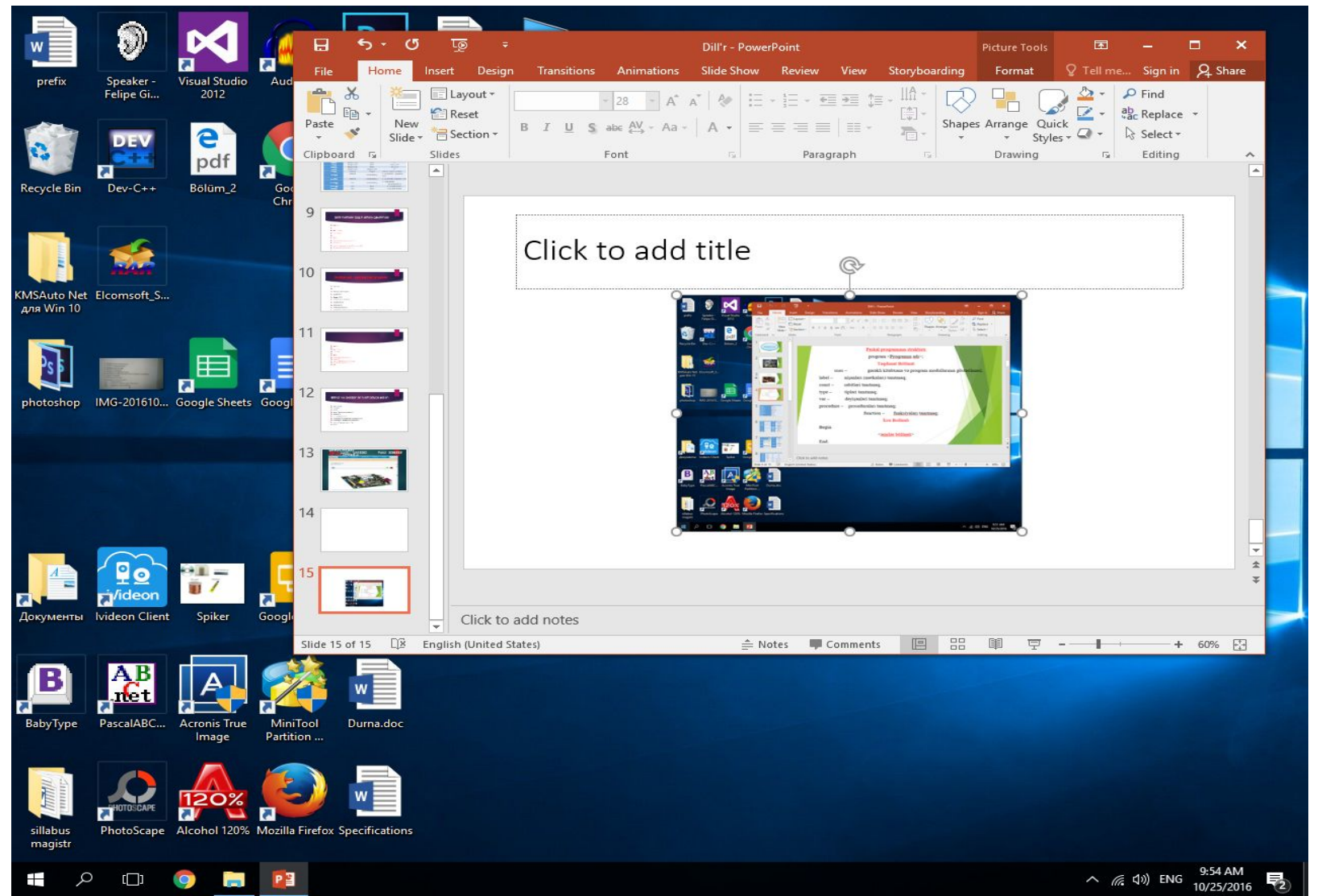
Numeric Keypad

Buttons

Print Screen düyməsinin funksiyası:

Cari ekranın təsvirini Clipboard-a,
bufer yaddaşa yerləşdirir

Scrool Lock – skrolling (ekranı
fırlatmaq) prosesini dayandırır



PROQRAM TƏMİNATI

Sistem PROQRAM TƏMİNATI

ƏS

Əməliyyat mühiti

Şəbəkə ƏS

Utilitlər

Antivirus proqramları

Tədbiqi ROQRAM TƏMİNATI

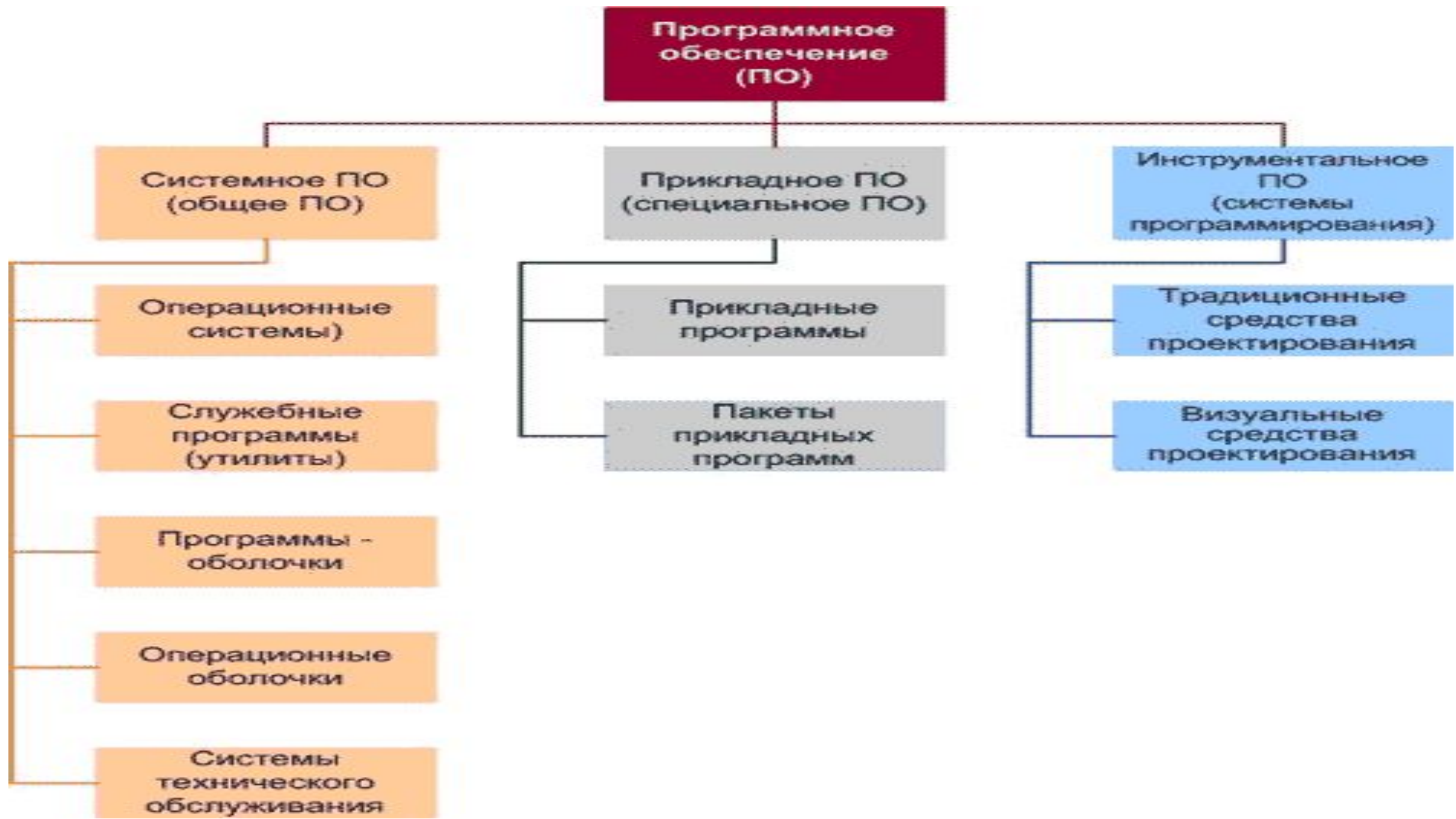
Tədbiqi Proqram dəstləri

Tədbiqi ROQRAMlar

İnstrumental PROQRAM TƏMİNATI

Ənənəvi proqramlaşdırma vasitələri

Vizual proqramlaşdırma vasitələri



SİSTEM PROQRAM TƏMİNATI

Baza PROQRAM TƏMİNATI

Xidməti (servis) PROQRAM TƏMİNATI

- ƏS
- Əməliyyat mühiti
- Şəbəkə ƏS

- Diagnostika proqramları
- Antivirus proqramları
- Qurğulara xidmət proqramları
Drayverlər
- Arxivləşdirmə proqramları
- Şəbəkə xidməti proqramları

Əməliyyat Sistemləri

DOS

DOS sözü ingiliscə Disk Operating System olub Azərbaycan dilinə tərcüməsi “Disk əməliyyat sistemi”. DOS komputerlər üçün nəzərdə tutulmuş kiçik və sadə bir əməliyyat sistemi növü olub, əsas vəzifəsi disket və sabit disk kimi saxlama mühitlərinin idarə edilməsidir. Komputerlərin digər funksiyaları: qrafika, səs, yazma, şəbəkədə gəzilmə, yaddaşa nəzarət, çoxlu istifadəçi və çoxlu iş xüsusiyyətləri DOS tərəfindən yerinə yetirilə bilər. DOS sistemləri 90-cı illərin ortalarına qədər demək olar ki, hər personal komputerdə öz vəzifəsini yerinə yetirmişdir. Bu gün computer dünyasının çox sahəsində qrafik əməliyyat sistemlərindən istifadə olunsa da, DOS sadə və kiçik olmaq üstünlüyü ilə müxtəlif əməllər və nəzarət etmə sistemlərində öz həyatına davam etməkdədir. DOS-un tarixi komputerlərin tarixi ilə başlayır. DOS-da qrafik bir istifadəçi pəncərəsi olmadığından hər bir əməliyyat əməllərlə və az sayda olan parametrlərlə həyata keçirilir.

Windows XP

Windows XP Maykrosoft şirkətinin personal komputerlər və server sistemləri üçün Windows NT-nin kernel sistemi üzərində qurulmuş olan 6-cı əsas distributividir. Windows XP 25 oktyabr 2001-ci il tarixində satışa təqdim edilmişdir. XP adı ingilis sözü olan “experience” sözündən gəlir və Azərbaycan dilində mənası “təcrübə” deməkdir. Windows XP hal-hazırda dünyada ən çox istifadə edilən əməliyyat sistemidir. Windows XP əvvəlki distributivlərindən fərqli olaraq tamamilə 32 bitlik Windows NT və Windows 2000 kernel sistemi üzərində qurulmuşdur. Onu da qeyd etmək lazımdır ki, Windows əməliyyat sisteminin əvvəlki versiyaları aşağıdakılardır:

- Windows 1.0
- Windows 2.0
- Windows 3.0
- Windows 95
- Windows 98
- Windows NT

Bu kernel system 16 və 32 bitlik tətbiqləri işlədə bilir və göy ekran səhvlərini azaldır. Windows əməliyyat sistemi ailəsinin Cairo, Nashville, Neptune, Odyssey kimi versiyaları satışa çıxarılmayaraq ləğv edilmişdir. Maykrosoft şirkəti Windows XP-in satışını 30 iyun 2008-ci il tarixindən etibarən dayandırmışdır: amma bəzi mini noutbuklarda hələ də qurulu olaraq XP gəlməkdədir. Windows XP Service Pack 2-nin də satışı 2010-cu ildə bitmişdir. Windows XP Service Pack 3 isə 2014-cü ilə qədər satılacaqdır.

Macintosh

Macintosh qısaca Mac olaraq tanınır və adını Macintosh alma növündən alır. Mac personal komputerlər istehsal edən Apple Computer Inc. bir məhsulu olan əməliyyat sistemidir. Macintosh-un istehsalına 1984-cü ildə başlanılmışdır. Onu da qeyd etmək ki, Apple şirkəti siçan və qrafik interfeysdən ilk dəfə istifadə edən şirkətlərdəndir. Macintosh-dan əvvəl Apple şirkətinin Lisa, Apple II, Apple III kimi sistemləri olsada 1986-cı ildə onların fəaliyyəti dayandırılmışdır və bütün məhsullar Macintosh adı altında tanınmışdır. Macintosh-un PowerPC arxitekturasına istifadə etməsi 1994-2005-ci illəri əhatə edən x86 sinifli komputerlərdən əvvəlki məhsullar

UNIX

UNIX 1969-cu ildə Ken Tompson və Denis Riçi tərəfindən Bell Laboratori-yalarında yaradılmış olan çox istifadəçili və çox vəzifəli quruluşu dəstəkləyən bir əməliyyat sistemidir. UNIX törəməli əməliyyat sistemləri çox prosessorlu bahalı kompüterlərdən bir prosessorlu sadə ev kompüterlərinə qədər bir çox cihaz üzərində işləyə bilən test edilmiş strukturu ilə çox prosessorlu serverlərdə adətən standart halına gəlmişdir. UNIX-in əsası 1965-ci ildə MIT, AT&T Bell Labs və GE-nin birlikdə yaratdıqları MULTICS (Multiplexed Operating and Computing System) layihəsi ilə atılmışdır. MULTICS layihəsinin əsas hədəfi çox istifadəçili kompüter sistemlərinə icazə verərək eyni vaxtlı məlumat paylaşmasını təmin edə bilməkdir. 1969-cü ildə layihə qarışıq bir şəkil almağa başlamış və AT&T Bell Labs layihədən çəkilmişdir. Bell şirkətində araşdırmaçı olaraq işləyən Ken Tompson MULTICS proqramını stimulyasiya edən bir fayl sistemini kodlamaqla UNIX-in ilk distributivi olan UNICS-i (Uniplexed Operating and Computing) meydana çıxardı. Bu versiya Assembler ilə yazılmışdı. 1973-cü ildə Ken Tompson C dilinin yaradıcısı olan Denis Riçi ilə birlikdə əməliyyat sisteminin kernel sistemini C dili ilə təkrar kodlayaraq UNIX 5.0 versiyasını yaratdılar. Beləliklə də, UNIX C dilinin sayəsində müxtəlif avadanlıqlara uyğun olaraq təkrar dəyişdirilə bilən kodlardan ibarət bir əməliyyat sisteminə çevrilmişdir.

Linux


Linux UNIX-ə texniki mənada bənzəyən redaktə qabiliyyətli müstəqil bir əməliyyat sistemidir. Kernelin kodları GNU Ümumi Cəmiyyət Lisenziyası ərçivəsində müstəqil olaraq dəyişdirilə istifadə edilə bilər. Linux hər hansı bir kompüter sistemində problemsiz işləmə qabiliyyətinə malikdir. Çox geniş bir təchizat dəstəyinə malik olan Linux netbuk, noutbuk, server kompüterləri, is stansiyaları, ağıllı telefon, stolüstü kompüterlər kimi hər bir platformada tam bir uyğunlaşma içərisində işləmək qabiliyyətinə malikdir. Linux adətən server və iş stansiyalarında istifadə olunsa da onu şəxsi kompüterlərdə də istifadə edənlər çoxluq təşkil edir. Təbii ki, bu halda redaktə edilən kodların və azad proqram anlayışının təsiri böyükdür. Windows XP və Mac OS sistemlərinə nəzərən daha rahat bir struktura malik olan Linux xüsusilə server satışlarında rəqibləri ilə müqayisədə daha üstündür. Mint, Ubuntu, openSUSE, Pardus, Mandriva kimi son versiyaları ilə Linux əməliyyat sisteminin istifadəçi faizi gündən-günə artmaqdadır.

Sun OS

Sun OS əməliyyat sistemi əvvəllər iş stansiyaları və server kompüter sistemləri üçün Sun Microsystems şirkəti tərəfindən istehsal olunan UNIX əməliyyat sisteminin bir versiyası idi. Sun OS adı adətən Sun OS əməliyyat sisteminin 1.0 və 4.1.4 versiyalarına istinad edilir. Bu versiyalar açıq redaktə edilə bilinən kodlu idi. Sun OS sistemi 5-ci versiyasından başlayaraq Solaris brendi altında satışa çıxarılmaya başlandı. Sun OS 1 və 2 versiyası Sun-2 sistemindən, Sun OS 3 versiyası Sun-2 və 3 sistemindən, Sun OS 4 versiyası isə Sun-3, Sun-386i, Sun-4 sistemindən daha doğrusu arxitekturalarından istifadə edir. Bu əməliyyat sisteminə yeni SPARC prosessor faylı əlavə edildikdən sonra birinci Sun-4m arxitekturalı multiprosessorlu kompüterlər (SPARCserver 600MP seriyası) meydana gəlməyə başladı. Sun OS 4-ün axıncı versiyası 4.1.4 olmuşdur və 1994-cü ildə buraxılmışdır. 2003-cü ildən bəri isə Solaris şirkətinin ən məşhur məhsulu Sun OS 5 əməliyyat sistemidir və bu sistem sadəcə olaraq Solaris adı ilə dünyada tanınır.

Palm OS

Garnet OS kimi də tanınan Palm mobil əməliyyat sistemi 1996-cı ildə PDA-lar üçün Palm Inc şirkəti tərəfindən istehsal edilməyə başlanılmışdır. Palm OS əllə toxunula bilən (touchscreen) qrafik istifadəçi interfeysi ilə asan istifadə olunması üçün istehsal edilmişdir. Bu, şəxsi məlumatların idarə edilməsi üçün əsas proqramlar dəsti ilə təmin edilir. Palm əməliyyat sisteminin son versiyaları smartfonlarda istifadə üçün nəzərdə tutulmuşdur. 2007-ci ildə Palm əməliyyat sistemi ticarət markası aldıqdan sonra Garnet OS deyə yenidən adlandırıldı. Garnet OS sisteminin varisi olan ACCESS 2009-cu ildən Access Linux Platform deyə tanınmaya başladı və gələcəkdə istehsal olunacaq cihazlar üçün Palm OS ilə Web OS sistemləri arasında keçidlər etməyə başladı. Palm OS rəsmi olaraq Cef Hokinsin rəhbərliyi altında Palm Computing şirkəti tərəfindən istehsal olunur. Palm Computing şirkəti sonralar 3Com-a çevrilən US Robotics Corp tərəfindən Palm Inc-ə çevrildi.



**PROGRAM
LANGUAGES**

PROQGRAM DILLƏRI

Kompyüterdə ixtiyari problemi həll etmək üçün proqramlaşdırma dillərindən istifadə edilir. Məsələ həll edərkən əvvəlcə yerinə yetiriləcək əməliyyatların *alqoritmi* tərtib edilir. Proqramlaşdırmada məsələni alqoritmləşdirməkdən qabaq aşağıdakı addımlar yerinə yetirilməlidir:

- ❖ **Məsələnin riyazi qoyuluşu:**
- ❖ **Nə verilir, ilkin verilənlərin sadalanması;**
- ❖ **Nə tələb olunur, nəticələrin sadalanması ;**
- ❖ **İlkin verilənlərin məhdudiyət şərtləri.**
- ❖ **Riyazi model: nəticələri almaq üçün lazım olan bütün qayda və qanunlar.**
- ❖ **Həll metodu: riyazi modelin optimal istifadə olunması.**

Alqoritm

Latınca qayda-qanun deməkdir.

Alqoritm 783- 850-ci illərdə Xorezmdə yaşamış IX əsr məşhur riyaziyyatçı Məhəmməd İbn Musa əl-Xarəzminin (yəni Xarəzmi Musa oğlu Məhəmməd) adının latın hərflərilə olan yazılışıyla bağlıdır. Əl-Xarəzminin yazdığı traktatın XII əsrdə latın dilinə tərcümə olunması sayəsində avropalılar **mövqeli say sistemi ilə tanış olmuş, onluq say sistemini** və onun hesab qaydalarını alqoritm adlandırmışlar.

Ümumiyyətlə, alqoritm-verilmiş məsələnin həlli üçün lazım olan əməliyyatları müəyyən edən və onların hansı ardıcılıqla yerinə yetirilməsini göstərən formal yazılışdır.

1 Alqoritm xassələri

2 Alqoritm təsvir üsulları

3 Alqoritm tipləri

Alqoritmin növləri

Xətti alqoritm



Alqoritmin bütün mərhələləri verildiği ardıcılıqla yerinə yetirilir

Budaqlanan alqoritm

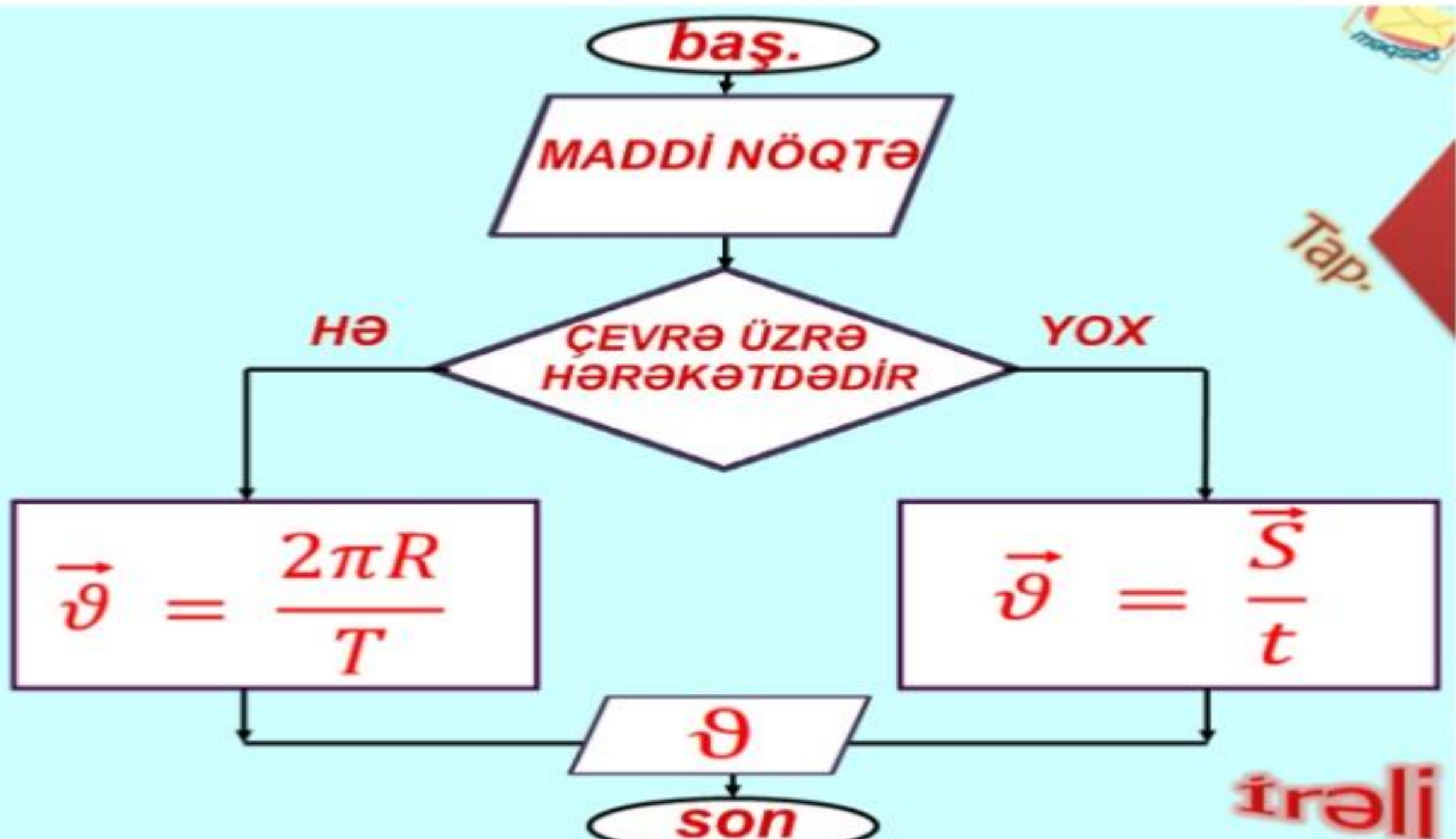


Alqoritm həll variantlarından birini seçməyə imkan verir

Dövri alqoritm



Alqoritmde müəyyən addımlar dəfələrlə təkrarlanır



Alqoritmin xassələri

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

Alqoritm sonlu sayda mərhələdən sonra qurtarmalıdır. Buna, alqoritmin **sonluluq (nəciləlilik)** xassəsi deyilir.

Alqoritmin hər bir addımı dəqiq və birqiymətli təyin olunmalıdır. Bu alqoritmin **müəyyənlik** xassəsidir.

Alqoritmin müəyyən sayda giriş qiymətləri (məsələnin başlanğıc şərtləri) olmalıdır. Bu şərtlər proqram icra olunmamış və ya olunduqca maşına daxil edilə bilər. Alqoritmin yerinə yetirilməsi nəticəsində giriş qiymətlərindən asılı olan bir və ya bir neçə çıxış qiymətləri alınmalıdır.

Alqoritm sadə və səmərəli olmalıdır, yəni alqoritmin nəticəsi (cavabı) mümkün qədər sadə əməliyyatlar vasitəsilə və ən qısa yolla alınmalıdır.

Alqoritm ümumi olmalıdır, yəni müəyyən məsələ üçün tərtib olunmuş alqoritm, həmin tipdən (sinifdən) olan bütün məsələlər üçün yararlı olmalıdır. Bu alqoritmin **kütləvilik** xassəsidir. Riyaziyyatda və informatikada məsələnin həllinin alqoritmi yerinə yetirilibsə, məsələ qismən həll edilmiş sayılır.

Düzgün tətbiq edilən alqoritm aşağıdakı əsas xüsusiyyətlərə malik olmalıdır:

- *Nəticəlilik;*
- *Müəyyənlik;*
- *Diskretlilik;*
- *Kütləvilik.*

Nəticəlilik əməliyyatların sonlu sayının mümkünlüyünü göstərir ki, onların yerinə yetirilməsi axtarılan nəticəyə gətirib çıxara bilsin.

Müəyyənlik

istifadəçidən və tətbiq edilən texniki vasitələrdən asılı olmayaraq alınan nəticələrin üst-üstə düşdüyünü göstərir.

Diskretlilik

alqoritmin təsvir etdiyi proses ayrı-ayrı addımlar ardıcılığına bölünməli və bu bölgü elə olmalıdır ki, alqoritmin yazılışı bir-birindən dəqiq ayrılmış göstərişlər şəklində olsun. Məsələn, “yolun kənarı ilə gedin” və yaxud “kənar şəxslərin daxil olması qadağandır” kimi göstərişləri kəsilməz xarakterli olduqları üçün alqoritm deyil, lakin resept əsasında dərman hazırlamaq və yaxud bankomatdan pul çıxartmaq və bu kimi qaydalar diskret xarakterli olduqları üçün alqoritmdir.

Kütləvilik ilkin məlumatların konkret qiymətləriylə fərqlənən eyni tipli məsələlərin bütöv sinifinə alqoritmin tətbiq edilmə imkanlarını nəzərdə tutur. Tənlik P_{r-1} ixtiyari əmsallar üçün korrekt cavab verməlidir.

Alqoritmin təsvir üsulları

- ❖ **Mətn şəklində (adi dildə);**
- ❖ **Qrafik blok-sxem;**
- ❖ **Psevdokod**
- ❖ **Cədvəl;**
- ❖ **Proqram (alqoritmik dil).**

Alqoritmin adi dildə təsviri (nəqli). Bu zaman əməliyyatlar, icra olunacaq hərəkətlərin nəqli şəkildə ardıcıl sadalanması kimi verilir. Məsələn, kofenin hazırlanmasını ifadə edən alqoritmin təsviri buna misal ola bilər.

Alqoritmin blok-sxem təsviri. Mürəkkəb alqoritmlərin təsviri zamanı blok-sxemlərdən istifadə olunması daha geniş yayılmışdır, çünki bu halda alqoritmin blok-sxem şəklində təsviri daha əyani olur. Bu zaman, adətən alqoritmin bir addımına bir blok uyğun olur. Lakin bir blokda bir neçə eyni tipli mərhələ və ya bir mərhələ bir neçə blokda təsvir oluna bilər. Bloklar standart işarələr şəklində ifadə olunur və bir-birləri ilə şaquli və ya üfüqi xətlərlə birləşdirilir. Birləşdirici xətlərin uclarında istiqaməti göstərən ox işarəsi qoyulur. Alqoritm ayrı-ayrı ədədlərlə yox, verilmiş hər hansı obyektlərlə işləyir.

Alqoritmin tipləri

EHM-də müxtəlif tipli məsələləri həll edərkən əsasən üç tipli alqoritmlərdən istifadə olunur: xətti (düz), budaqlanan və dövri.

Xətti alqoritmlər sadə hesablama prosesini ifadə edən bir neçə ardıcıl əməliyyatlardan ibarət olur və onlar yazıldığı ardıcılıqla da icra olunur.

Budaqlanan alqoritmlərin tərkibində bir və ya bir neçə məntiq mərhələsi olur. Bu mərhələdə müəyyən kəmiyyətlərin hər hansı bir şərti ödəyib-ödəmədiyi yoxlanılır və ona uyğun olaraq sonrakı gedişin istiqaməti seçilir. Yəni nəzərdə tutulan şərt ödənilirsə, məsələnin həli prosesi bir istiqamətə, həmin şərt ödənilmirsə, başqa istiqamətə doğru hərəkət edilir. Beləliklə, alqoritmə budaqlanma baş verir.

Dövri alqoritm. Proqramlaşdırmada tez-tez eyni əməliyyatlar qrupunun çoxlu sayda təkrar olunması lazım gəlir. Bu halda dövr alqoritmindən istifadə olunur. Dövrələr sadə və mürəkkəb olur. Sadə dövrlü alqoritmin bir dövrü olur. Əgər hər hansı bir alqoritmə bir neçə daxili dövr iştirak edirsə, onda belə dövrlərə mürəkkəb dövr deyilir. Mürəkkəb dövrləri əmələ gətirən sadə dövrlər kəsişə bilməz.

Kompilyasiya və interpretasiya

Hesablama maşınlarının əsas fərqləndirici xüsusiyyəti onun *proqramla* idarə olunmasıdır. Yəni, istər sadə, istərsə də mürəkkəb məsələni maşının həll etməsi üçün proqram tərtib edilməlidir

əməliyyatlar hər-hansı alqoritm (*proqramlaşdırma*) dilində *əmlər* şəklində yazılır. Tərtib olunmuş proqram *translyator* proqramlar vasitəsilə *maşın koduna* çevrilir və *işləndikdən* sonra yerinə yetirilir.

Translyasiyanın iki qaydası var: interpretasiya və kompilyasiya. Interpretasiya 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 qabaq proqramın bütün tərcüməsi yığılır. Interpretasiya böyük çevikliyə malik olmaqla asan realizə olunur. Kompilyasiya isə daha effektiv proqram yaradır.

Proqramlaşdırmanın əsas obyektı dəyişəndir. Məsələn, x adlı dəyişənə 25 qiymətinin mənimsənilməsini belə müəyyən etmək olar:

$x := 25$ yazılır və $x = 25$ olur.

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

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

Proqram dillərinin nəsilləri

Nəsillər	Proqram dilləri	Xarakteristika
I	Maşın dilləri	Konkret maşın üçün, mənimsənməsi çətin, arxitekturanı bilmək tələbli
II	Assemblerlər, makroassemblerlər	Maşından asılı olsa da istifadəsi rahat
III	Yüksək səviyyəli Proqram dilləri	Öyrənilməsi asan, mobil, insana yönəli
IV	Q/Prosedur, Obyektyönümlü, Sorğular dilləri, Parallel	q/professionallara hesablanıb. Parallel arxitektura üçündür
V	Süni intellekt, ekspert sistemləri və biliklər bazası, təbii dillər	Dilli interfeysə, maşının artan intellektinə yönəli

Proqram dillərinin sinifləşdirilməsi

Faktor	Xarakteristika	Qruplar	Nümunələr
Proqram dillərinin səviyyəsi	Proqram dillərinin PK arxitekturasına yaxınlıq dərəcəsi	Aşağı	Avtokod, Assembler
		Yüksək	Fortran, Pascal, ADA, Basic, C və s.
		Çox yüksək	Setl
Proqram dillərinin İxtisaslaşdırılması	Potensial və ya real tətbiq sahəsi	Ümumi təyinatlı (universal)	Fortran (mühəndis hesablamaları), Cobol (İqtisadi məsələlər), Refal, Lisp(simvol emalı), Modula, ADA(real zaman miqyasında proqramlaşdırma)
		İxtisaslaşdırılmış	
Alqoritmlilik (Prosedurluq)	Alqoritmin xırdalıqlarından mücərrədləşməyə doğrudur. Алгоритмичность тем выше, чем точнее приходится планировать порядок выполняемых действий	Prosedur	Assembler, Fortran, Basic, Pascal, ADA
		Q/Prosedur	Prolog, Langin

Source code . Mənbə kod - Başlanğıc proqram – proqramın mətni

Adətən, aşağıdakı bəndlərdən ibarət olur: (ilk dəfə Cobol dili üçün təyin olunub):

İdentifikasiya- proqramın adı, eləcə də proqramçı və istifadəçi üçün əlavə məlumat saxlanan hissə;

Əlaqələr- əgər proqram digərindən çağırılıbsa və deməli başqa proqramdan nəticələri almalıdırsa mübadilə olunan verilənlərin (proqramın parametrləri adlanır) təsvir olunduğu proqram fragmentidir. фрагмент текста, описывающий внешние переменные, передаваемые вызывающей программой (если таковая имеется), т.е. ту часть исходных данных, которая обязательно поступает на вход программы при ее запуске. Эти переменные часто называют параметрами программы;

Avadanlıq(mühit) – kompüterin proqramın icra olunması baxımından əhəmiyyətli olan prosesor göstəriciləri, tipi, əməli və xarici yaddaşa tələblər.

Məsələn, ondadır ki, hətta eyni tipli kompüterlər arasında maşın komandaları arasında fərqlər ola bilər.

Verilənlər- proqramda istifadə olunan verilənlərin tipləri, adları (deklorasiya, elan; təsvir). Verilənlərin tiplərinin göstərilməsi əməliyyat gedişində verilənlər arasındakı uyğunsuzluqları və mümkün olmayan çevirmələri üzə çıxarmağa imkan verir.

prosedurlar- proqramın əsas hissəsidir ki, verilənlərin emalı proseslərinin təsvirindən ibarətdir. Prosedurların elementləri proqram dilinin operatorlar və standart funksiyalarıdır.

Bu bəndlər müxtəlif dillər üçün fərqli ola bilər

PROQRAMLASHDIRMA DILLƏRİ

- **Ada** - Ada proqramlaşdırma dilinin yaranma tarixi 1974-1980-ci illərə təsadüf edir.
- **Assembler** – bu proqram vasitəsilə effektiv və kompakt proqramlar yaradılır.
- **Basic** – Bu dil üçün kompilyator və interpretatorlar mövcuddur. 60-cı illərdə yaradılmışdır və öyrənilməsi sadədir.
- **C** – Bell laboratoriyasında yaradılmışdır və assembler dilini əvəz etmək üçün nəzərdə tutulmuşdur. Assemblerdən fərqli olaraq konkret tip prosessordan asılı deyil.
- **C++** – 1980-ci ildə Byörn Straustrup tərəfindən yaradılmışdır. Proqramın imkanları programcının işinin məhsuldarlığını artırmış olur.

- **LISP** - List Processing (siyahıların emalı) sözlərindən götürülmüşdür. 1958-ci ildə yaradılmışdır
- **Lua**- 1993-cü ildə yaradılmış açıq qaynaq kodlu dildir.
- **Objective-C** - yuxarı səviyyəli, Obyekt yönümlü proqramlaşdırma dilidir.
- **Pascal**– 70-ci illərdə Niklaus Birt tərəfindən yaradılıb, Alqol dilinə daha çox oxşayır.
- **Perl** - 1987-ci ildə Lary Wall tərəfindən yaradılmış dinamik proqramlaşdırma dilidir.
- **PHP**- (Hypertext Preprocessor) dinamik veb səhifələr yaratmaq üçün 1995-ci ildə yaradılıb.
- **Prolog** - Proloq məntiqi proqramlaşdırma dilidir ki Süni intellekt və Hesablama dilçilik ilə əlaqədardır.
- **Python** - 1991-ci ildə Guido van Rossum tərəfindən yaradılmışdır. Dilinin sintaksisi çox aydın və anlaşılıqdır.
- **Ruby** - Yukihiko Matsumoto tərəfindən Perl, Smalltalk və Eiffel dillərindən

Maşın dili. Kompüterin bilavasitə “başə düşdüyü” yeganə dil – sadəcə, ədədlər yığınınından ibarət olan maşın dilidir [machine language].

Ədədlərlə işlədiyindən prosessor üçün komandaları kodlaşdırmaq, məsələn, ədədlərlə ifadə etmək lazımdır

01 və 02 xanalarında yerləşmiş iki ədədin ədədi ortasını tapan program aşağıdakı şəkildə olacaq:

01 0101 0102 0103
03 0103 0002 0103

Hər bir verilən, kompüterin yaddaşının xanalarında yerləşir. Bütün yaddaş xanalara bölünmüşdür və hər xananın öz nömrəsi – ünvanı olur. Beləliklə, hər bir operand iki parametrlə – əvvəlməti və vaddasda olan veri ilə təyin olunur

Maşın dili

-konkret kompüterin əmrlər sistemindən ibarət olub, bilavəsitə həmin maşın tərəfindən həyata keçirilir. Maşın dilində proqram tərtib etdikdə hər şeydən əvvəl dəyişənlər və konstantlar üçün maşının yaddaşında yer ayrılır. Maşın dilində proqram maşın əmrləri ardıcılığından və dəyişənlər, konstantlar üçün yaddaşda təyin edilmiş müəyyən sahələrdən ibarətdir. Maşın proqramının strukturu qəti müəyyən edilmədiyindən, dəyişənlər, konstantlar və əmrlər proqramda ixtiyari ardıcılıqda yerləşirlər. Yaddaşın əmrlər, konstantlar və dəyişənlər yerləşən oyuqları arasında heç bir fərq yoxdur. Belə ki, əmr üçün ayrılmış yaddaş elementi dəyişən yaxud konstant üçün də istifadə olunan bilən maşın əmrləri vasitələ çox sadə əməllər həyata keçirilir. Misal üçün yaddaşın bir oyuqda yerləşən informasiyanı digərinə keçiriməli;iki oyuqun daxilində yerləşən kəmiyyətləri toplamalı və s. Bununla belə, maşın əmrləri vasitəsilə proqram tərtib etmək böyük əmək sərfi tələb edir.Maşın dilində tərtib edilmiş proqram sazlamaq da mürəkkəb məsələdir, çünki proqrama yalnız bir əmirin əlavə edilməsi belə yaddaşda çox sayda ünvanın dəyişməsinə səbəb ola bilər.Maşın dili kompüterin qurğularından tam istifadə etməyə imkan verir. Onun vasitəsilə kifayət qədər effektiv olan proqram tərtib etmək mümkündür.

Bu dil proqramçıdan yüksək ixtisaslılıq tələb edir.

Bununla belə, bu dil əsasında yüksək əmək məhsuldarlığına nail olmaq olduqca çətindir.



Assemblerlər səviyyəli dillər

-konkret kompüterlərin əmrlər sisteminə uyğun gələn maşinyönlü dillərdir. Buna baxmayaraq, onlar proqramı istifadəçi üçün daha rahat olan formada tərtib etməyə imkan verirlər. Assembler dilinin üstün cəhəti ondadır ki, dildə əmrlərə, konstantlara və dəyişənlərə müəyyən adlar mənsub edilir və bu adlar vasitəsilə onların özlərinə müraciət etmək imkanı yaranır. Bundan əlavə, dilə bir neçə proqram vahid proqram şəklində birləşdirməyə və səhvlərə nəzarət etməyə imkan verən vasitələr də daxil edilmişdi. Assembler dilində kompüterin bütün imkanlarından tam istifadə etməyə imkan verən effektiv proqramlar yazılır.

Çatışmayan cəhət proqramın həddindən artıq təfərrüfatı ilə yazılmasıdır.

ASSEMBLERİN YARANMASI



1952-ci ildə amerikalı qadın **Qreys Hopper** dünyada ilk mnemonik proqramlaşdırma dili olan Assembler dilini yaratdı. Onun adı "assemble" ingilis sözündən götürülmüşdür, "yığmaq", "quraşdırmaq" mənasını verir. O, özündə mnemonik komandalar sistemini, prosedurlar kitabxanasını və proqram mətnlərini maşın koduna çevirmək üçün xüsusi proqramı birləşdirirdi. MAşın kodunun alınmasının belə proseduru kompilyasiya (ing. *compile* – "tərtib etmək", "yığmaq"), onu həyata keçirən proqram isə **kompilyator** adlanır. Bunları da Qreys

Assembler dilinin *sistem proqramlaşdırma*ya aid olan hissəsi isə, həm mürəkkəbliyinə, həm də həcminə görə istifadəçi proqramlaşdırmadan qat-qat böyükdür. İşin problem tərəfi odur ki, sistem proqram kodları istifadəçi proqramları kimi istədiyimiz vaxt kompilyasiya və icra edə bilmərik. Sistem proqramları kodu əməliyyatlar sisteminin proqram kodudur və yalnız bir dəfə kompüter yüklənəndə yüklənir və kompüterin bütün fəaliyyəti boyu icra olunur, kompüteri, onun resurslarını, avadanlıqlarını, fiziki yaddaşı, istifadəçi proqramlarını və s. idarə edir. Sistem proqram kodlarının işinə misal olaraq, prosessorun rejimlərini dəyişmək, kompüterin portlarına məlumat yazmaq (oxumaq), avadanlıqları sistemə tanıdıb konfigurasiya etmək, avadanlıqların işini idarə etmək, kəsilmələrə cavab vermək və s. göstərmək olar.

Assembler

— aşağı səviyyəli proqramlaşdırma dili olub maşın kimandalarının işləmək üçün rahatlaşdırılmış yazı formasından ibarətdir.

Əmrləri CPU komandalarına uyğundur,

Rahat simvolik yazı forması var

(мнемокод) команд и их аргументов. Также язык ассемблера обеспечивает базовые программные абстракции: связывание частей программы и данных через метки с символьными именами и директивы.

Verilənlər bloklarını proqrama daxil etmək imkanı var(описанные явно или считанные из файла); повторить определённый фрагмент указанное число раз; компилировать фрагмент по условию; задавать адрес исполнения фрагмента, менять значения меток в процессе компиляции; использовать макроопределения с параметрами и др.

Каждая модель процессора, в принципе, имеет свой набор команд и соответствующий ему язык (или диалект) ассемблера.

Üstünlük və zəif cəhətləri

- İzafi kodun minimal olması(использование меньшего количества команд и обращений в память). Как следствие — большая скорость и меньший размер программы
- большие объемы кода, большое число дополнительных мелких задач
- плохая читабельность кода, трудность поддержки (отладка, добавление возможностей)
- трудность реализации парадигм программирования и любых других сколько-нибудь сложных конвенций, сложность совместной разработки
- меньшее количество доступных библиотек, их малая совместимость
- непосредственный доступ к аппаратуре: портам ввода-вывода, особым регистрам процессора
- возможность написания самомодифицирующегося кода (т.е. метапрограммирования, причем без необходимости программного интерпретатора)
- максимальная «подгонка» для нужной платформы (использование специальных инструкций, технических особенностей «железа»)

Sintaksis

Общепринятого стандарта для синтаксиса языков ассемблера не существует. Однако, существуют стандарты де-факто — традиционные подходы, которых придерживаются большинство разработчиков языков ассемблера. Основными такими стандартами являются Intel-синтаксис и AT&T-синтаксис.

Общий формат записи инструкций одинаков для обоих стандартов:

`[метка:] опкод [операнды] [;комментарий]`

Опкод — непосредственно мнемоника инструкции процессору. К ней могут быть добавлены префиксы (повторения, изменения типа адресации и пр.). В качестве операндов могут выступать константы, названия регистров, адреса в оперативной памяти и пр.. Различия между стандартами Intel и AT&T касаются, в основном, порядка перечисления операндов и их синтаксиса при различных методах адресации.

Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры или семейства архитектур (среди широко известных — мнемоники процессоров и контроллеров Motorola, ARM, x86). Они описываются в спецификации процессоров.

Например, процессор Zilog Z80 наследовал систему команд Intel i8080, расширил ее и поменял мнемоники (и обозначения регистров) на свой лад. Например, сменил интеловские mov на ld. Процессоры Motorola Fireball наследовали систему команд Z80, несколько её урезав. Вместе с тем, Motorola официально вернулась к мнемоникам Intel. и в данный момент половина ассемблеров для Fireball работает с интеловскими мнемониками, а половина с мнемониками Zilog.

Директивы

Кроме инструкций, программа может содержать директивы: команды, не переводящиеся непосредственно в машинные инструкции, а управляющие работой компилятора. Набор и синтаксис их значительно разнятся и зависят не от аппаратной платформы, а от используемого компилятора (порождая диалекты языков в пределах одного семейства архитектур). В качестве набора директив можно выделить:

•определение данных (констант и переменных)

•управление организацией программы в памяти и параметрами выходного файла

•задание режима работы компилятора

•всевозможные абстракции (т.е. элементы языков высокого уровня) — от оформления процедур и функций (для упрощения реализации парадигмы процедурного программирования) до условных конструкций и циклов (для парадигмы структурного программирования)

•макросы

Ассемблер для процессора Intel 8080

Происхождение и критика термина «язык ассемблера»

Данный тип языков получил свое название от названия транслятора (компилятора) с этих языков — ассемблера (англ. assembler — сборщик). Название последнего обусловлено тем, что на первых компьютерах не существовало языков более высокого уровня, и единственной альтернативой созданию программ с помощью ассемблера было программирование непосредственно в кодах.

Язык ассемблера в русском языке часто называют «ассемблером» (а что-то связанное с ним — «ассемблерный»), что, согласно английскому переводу слова, неправильно, но вписывается в правила русского языка. Однако, сам ассемблер (программу)

тоже называют просто «ассемблером», а не «компилятором языка ассемблера» и т. п.

Использование термина «язык ассемблера» также может вызвать ошибочное мнение о существовании единого языка низкого уровня, или хотя бы стандарта на такие языки. При именовании языка, на котором написана конкретная программа, желательно уточнять, для какой архитектуры она предназначена и на каком диалекте языка написана.

Assemblerdә

İKİ ƏDƏDİN TOPLANMASI PROQRAMI

Программа складывает два числа, и проверяет результат. Если сумма равна 0 – выводится одно сообщение, если нет – другое.

В данной задаче самое интересное – это работа с метками, командами `jz`, `jmp` и `test`.

`test` – это операция логического сравнения двух операндов, размерностью байт, слово или двойное слово. В процессе выполняется операцию логического умножения: бит результата равен 1, если соответствующие биты операндов равны 1, в остальных случаях бит результата равен 0. Затем устанавливаются флаги, в том числе флаг `ZF` (zero flag), который равен 1, если результат логического умножения равен нулю.

Флаг `ZF` в дальнейшем используется для анализа результата.

`jnz` – выполняет переход по указанной метке, если не установлен флаг `ZF`. Данная команда обычно используется с операциями сравнения, которые влияют на состояние флага `ZF`.

Например, `test` и `cmp`.

`jz` – выполняет переход по указанной метке, если установлен флаг `ZF`. Данная команда обычно используется с операциями сравнения, которые влияют на состояние флага `ZF`.

Например, `test` и `cmp`.

`jmp` – выполняет безусловный переход по указанной метке.

В данной задаче самое интересное – это работа с метками, командами `jz`, `jmp` и `test`.

`test` – это операция логического сравнения двух операндов, размерностью байт, слово или двойное слово. В процессе выполняется операцию логического умножения: бит результата равен 1, если соответствующие биты операндов равны 1, в остальных случаях бит результата равен 0. Затем устанавливаются флаги, в том числе флаг `ZF` (zero flag), который равен 1, если результат логического умножения равен нулю.

Флаг `ZF` в дальнейшем используется для анализа результата.

`jnz` – выполняет переход по указанной метке, если не установлен флаг `ZF`. Данная команда обычно используется с операциями сравнения, которые влияют на состояние флага `ZF`. Например, `test` и `cmp`.

`jz` – выполняет переход по указанной метке, если установлен флаг `ZF`. Данная команда обычно используется с операциями сравнения, которые влияют на состояние флага `ZF`.

Например, `test` и `cmp`.

`jmp` – выполняет безусловный переход по указанной метке.

```
1  .486
2  .model flat, stdcall
3  option casemap: none
4  include /masm32/include/windows.inc
5  include /masm32/include/user32.inc
6  include /masm32/include/kernel32.inc
7  includelib /masm32/lib/user32.lib
8  includelib /masm32/lib/kernel32.lib
9
10 include /masm32/macros/macros.asm
11 uselib masm32, comctl32, ws2_32
12
13 .data
14
15 .code
16 start:
17
18 mov eax, 123
19 mov ebx, -90
20 add eax, ebx
21
22 test eax, eax
23
24 jz zero
25 invoke MessageBox, 0, chr$("B eax не 0!"), chr$("Info"), 0
26 jmp lexit
27
28 zero:
29 invoke MessageBox, 0, chr$("B eax 0!"), chr$("Info"), 0
30
31 lexit:
32 invoke ExitProcess, 0
33
34 end start
35
36
```

təkrarlamalı” və s.

Belə problemləri aradan qaldırmaq üçün yeni proqramlaşdırma dilinə ehtiyac vardı. Assembler dilindən fərqli olaraq, həmin dilin yaradılmasından əsas məqsəd onu maşının anlaması deyil, onunla işləməyin insan üçün əlverişliyi idi. Beləliklə, insan (proqramçı) üçün daha anlaşılıqlı olan və proqramlaşdırma prosesini asanlaşdıran yeni dillər yaradılmağa başladı. Proqramlaşdırma dilinin hər bir yaradıcısı insan-maşın əlaqələri haqqında öz təsəvvürlərini gerçəkləşdirir - di yindən qısa müddət ərzində yüzlərlə yeni dil meydana çıxdı (1993-cü ilin hesablamalarına görə, 1950-ci ildən həmin vaxta kimi 1000-dən artıq proqramlaşdırma dili yaradılıb). Təbii, yüksək səviyyəli [high-level] dillər adlandırılan dillərin az bir qismi öz yerini tapıb inkişaf etdi və möhkəmləndi. Bunun əksinə olaraq, assembler dili aşağı səviyyəli [low-level] dil hesab olunur, çünki o, maşın dilinə daha yaxındır və kompüterin qurğuları ilə işləyir.

Yüksək səviyyəli dillərin öz müsbət və mənfi cəhətləri var. Yüksək səviyyəli dilin assemblerdən başlıca üstünlüyü onun öyrənmək və istifadə üçün çox-çox asan olmasıdır. Yüksək səviyyəli dildə yazılmış proqram assemblerdəkinə nisbətən daha yığcam və anlaşılıqlıdır. Onlar, əsasən, daşınabiləndir, yəni müxtəlif prosessorlu kompüterlərdə eyni cür işləyir. Bu isə o deməkdir ki, proqramı yazarkən onun işləyəcəyi kompüterin arxitekturasının incəliklərini öyrənməyə ehtiyac qalmır. Əlbəttə, bu halda hər bir prosessorun öz kompilyatoru olmalıdır və onun yaratdığı icra faylı yalnız həmin prosessor üçün yararlı olacaqdır. Digər tərəfdən, assemblerdə yazılmış proqram, onun yüksək səviyyəli dildə yazılmış bənzərindən praktik olaraq həmişə səmərəlidir. Yəni yüksək səviyyəli dilin kompilyatorunun yaratdığı icra faylı funksional cəhətdən eyni olan assembler proqramından daha çox yer tutur və yavaş işləyir. Doğrudur, son illər mikroprosessorların inkişafı nəticəsində kompilyatorlar daha optimal kodlar generasiya edir

İndi dünyada ***bir neçə min proqramlaşdırma dili*** mövcuddur. Daha münasib dili necə seçməli?

Mühəndislərin, bankirlərin, hərbiçilərin qarşısında cürbəcür məsələlər durur. Onlar özləri üçün müxtəlif proqramlaşdırma dilləri seçirlər. Mühəndislər FORTRAN dilinə üstünlük verir, bankirlər çox zaman COBOL (“kobol”) dilindən istifadə edir, hərbiçilər isə döyüş əməliyyatlarını planlaşdırmağı və qoşunun idarə olunmasını ADA dilində yazılmış proqramların köməyi ilə həyata keçirirlər. Süni intellekt sahəsində çalışanlar üçün PROLOG, yaxud LISP dili daha münasibdir. İnternet üçün proqram yazan proqramçılar, adətən, JAVA dilinə üstünlük verirlər.

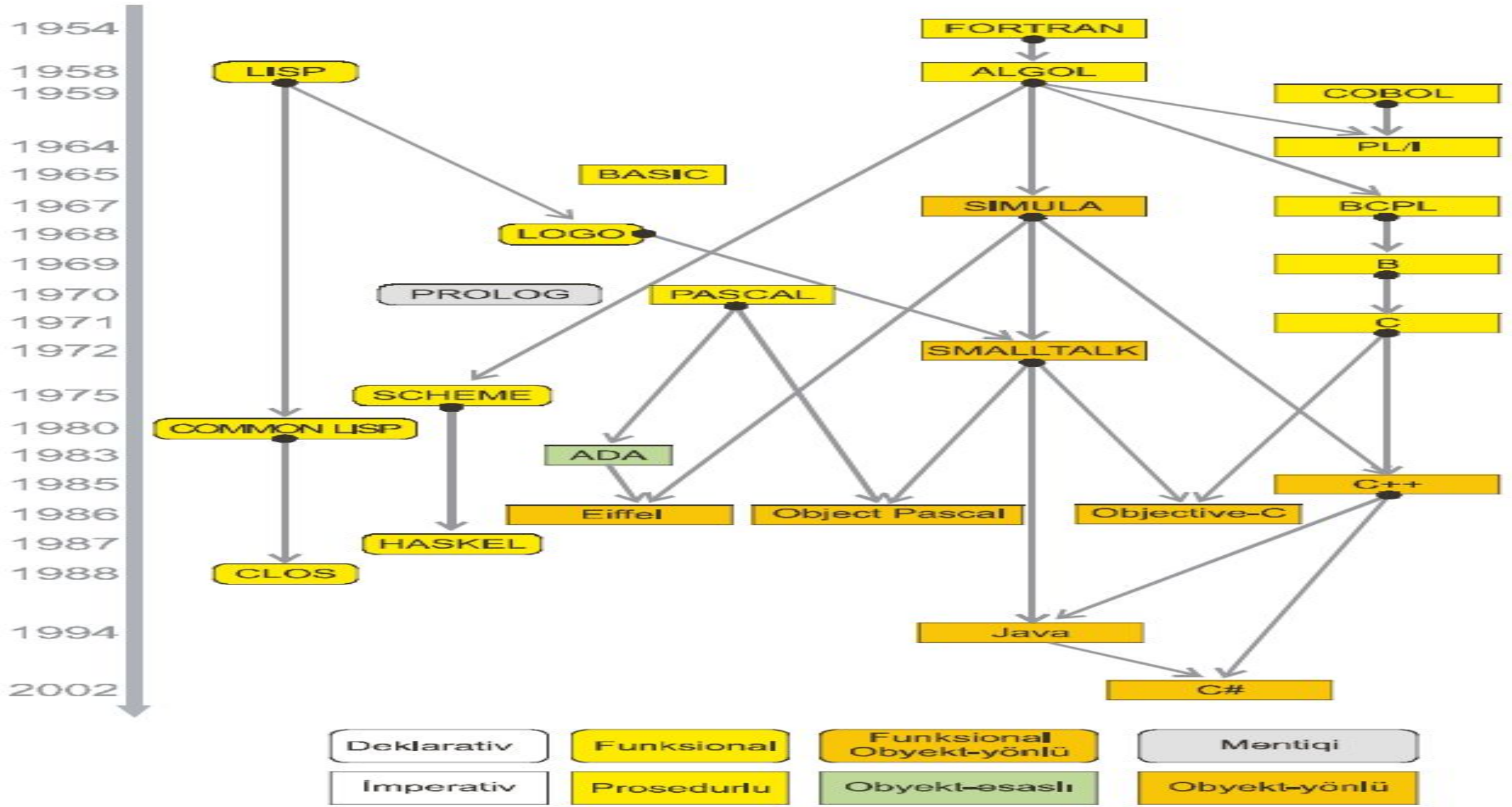
Bu sadaladığımız proqramlaşdırma dillərinin hamısı xüsusi dillərdir. Bu dillərin hər birində elə operatorlar var ki, onlar vasitəsilə xüsusi məsələləri daha asanlıqla həll etmək olur. Adətən, həmin dilləri iş prosesində ehtiyac olduqca öyrənirlər, belə ki, onları “qabaqcadan” öyrənməyin mənası yoxdur. Xüsusi dillərdən savayı, universal proqramlaşdırma dilləri də mövcuddur. Onların köməyi ilə, demək olar ki, istənilən məsələni həll etmək mümkündür. Belə dilləri “qabaqcadan” öyrənmək olar – onlar hər zaman gərəyiniz ola bilər. Belə dillərin üçü daha çox populyardır:

- Basic
- Pascal
- C++

Yüksək səviyyəli dillərin köməyi ilə istənilən sahəni proqramlaşdırmaq olar. Ancaq elə dillər də var ki, onlar xüsusi olaraq müəyyən sahələr üçün nəzərdə tutulub:

- ALGOL – riyazi məsələlər üçün;
- CHILL – telekommunikasiya sistemləri üçün;
- COBOL – iqtisadi məsələlər üçün;

FORTRAN – riyazi hesablamalar üçün;



Yüksək səviyyəli dillərin təkamül sxemi

PASCAL



Pascal dili (Paskal) – 1971-ci ildə professor Niklaus Vurt (Niclaus Wirth) şərəfindən yaradılmışdır. Dil fransız riyaziyyatçısı və filosofu Blez Paskalın (Blaise Paskal) şərəfinə adlandırılmışdır və qısa zaman içində universitetlərdə istifadə edilən proqramlaşdırma dili halına gəlmişdir.

Paskal proqramının strukturu

program <Proqramın adı>;

Təqdimat Bölümü

- uses – gərəkli kitabxana və proqram modullarının göstərilməsi;
- label – nişanları (metkaları) tanımaq;
- const – sabitləri tanımaq;
- type – tipləri tanımaq;
- var – dəyişənləri tanımaq;
- procedure – proseduraları tanımaq;
- function – funksiyaları tanımaq;

İcra Bölümü

Begin

<əmrilər bölümü>

End.

XİDMƏTİ SÖZLƏR

and – və	mod – qalıq
array – massiv	nil – sıfır
begin – başlanğıc	not – yox (xeyr)
case – seçim	of – dan (başlayaraq)
const – konstantlar	or – və ya
div – qalıqsız bölmə	packed – sıxlaşmış
do – icra et	procedure – prosedura
downto – aşağı qədər	program – proqram
else – əks halda	record – yazı
end – son	repeat – təkrar et
file – fayl	set – çoxluq
for – üçün (dövr üçün)	then – onda
function – funksiya	to – qədər
goto – keç	type – tip
if – əgər	until – hələlik
in – daxilində	var – dəyişənlər
label – nişan	while – hələ ki (nə qədər ki)

İfadə	Qiymət	Nümunə	Qiymət
not true	false	$-1 < 0$	true
true and true	true	$0 > -1$	true
true and false	false	$16 \text{ div } 5 = 3$	false
false and true	false	$16 \text{ mod } 5 = 1$	true
false and false	false	$\text{not}(b > 4)$	true
not false	true	$(-3 > -1) \text{ or } (3 > 5)$	false
true or true	true	$(10 \text{ div } 5=1) \text{ and } (3=3)$	false
true or false	true	$(17 * 3 = -51) \text{ or } (1>0)$	true
false or true	true		
false or false	false		

Əməl	Əməlin təyinatı
+, - , not	işarələmə, məntiqi inkar
*, / div, mod and	vurma, bölmə- tam ədəd əməlləri məntiqi hasil
+, - or, xor	toplama, çıxma məntiqi cəm
=, <>, <, >, <=, >=	müqayisə

Tipin adı	Qiymət diapazonu
byte	0..255
shortint	-128..+127
Word	0..65535
integer	-32768..+32767
longint	-2147483648..+214748

inc(x, y)	integer	integer	x- $\dot{\iota}$ y qədər böy-r
inc(x)	integer, char	integer , char	x- $\dot{\iota}$ 1 vahid böy-r
dec(x, y)	integer	integer	x- $\dot{\iota}$ y qədər az-r
dec(x)	integer, char	integer , char	x- $\dot{\iota}$ 1 vahid az-r

Tipin adı	Qiymət diapazonu
single	1.5*10 ⁻⁴⁵ .. 3.4*10 ³⁸
real	2.9*10 ⁻³⁹ .. 1.7*10 ³⁸
double	5.0*10 ⁻³²⁴ .. 1.7*10 ³⁰⁸
extended	3.4*10 ⁻⁴⁹⁵¹ .. 1.1*10 ⁴⁹³²

Funksiya	Arqumentin tipi	Nəticənin Tipi	Riyyazi yazılış
t	integer, real	integer, real	$ x $
arctan(x)	integer, real	Real	$\arctg(x)$
cos(x)	integer, real	Real	$\cos(x)$
sin(x)	integer, real	Real	$\sin(x)$
exp(x)	integer, real	Real	
ln(x)	integer, real	Real	$\ln(x), x>0$
sqrt(x)	integer, real	Real	
sqr(x)	integer, real	integer, real	
ord(x)	ordered	integer	ASCII –simvol kodları
pred(x)	ordered	nizamlanmış	x –in əvvəlki qiymətini verir
round(x)	real	nizamlanmış	x -i tam ədədə yuvarlaqlaşdırır
trunc(x)	real	Real	x –in tam hissəsi
frac(x)	real	Real	x–in kəsr hissəsi
odd(x)	integer	Boolean	true (x –tək) false(x –cüt) çevirir

Sətri hərflərin baş hərflərə çevrilməsi

- **Uses** Crt;
- **Var** S : String;
- I : Integer;
- **Begin**
- Write('Sətri Hərflərlə mətn daxil et! : ');
- ReadLn(S);
- **For** I:=1 **To** lenght[S] **do** S[I]:=upcase(S[I]);
- WriteLn('Baş hərflər sətri: ', S);
- **End.**

PASKAL-da psixoloji test proqramı

- **uses** crt;
- **var** a,b,c,d,m:longint;
- g,r,y,k:text;
- **begin** clrscr;
- a:=0; b:=0; c:=0; d:=0;
- //textmode(2);
- textcolor(3);
- textbackground(1);
- write('Her teqdim olunan nomredeki xarakterik xususiyyetden size en cox uygun olanini');
- Readln;
- write('secib qeyd edin.');
- Readln;
- write('I '):

Birinci və axırdan iki hərfi baş hərflərdən program

- **Var** S,n:String;
- i:Integer;
- **Begin**
- n:=' ';
- Write ('Hərfləri Sətiri daxil et');
- Readln(S);
- S[1]:=UpperCase(S[1]);
- **For** i:=1 **to** Length(S) **Do**
- **If** S[i]=n **then**
- S[i+1]:=UpperCase(S[i+1]);
- Writeln('Duzgun yazılış: ',S);
- **End.**

Yazılan mətnin bir hərfini baş digərini sətiri edən program.

- **Var** S,n:String;
- I:Integer;
- **Begin**
- Readln(S);
- Begin
- **For** I:=1 **to** Length(S) **Do**
- S[I]:=UpperCase(S[I]);
- **End;**
- **Begin**
- **For** I:=1 **to** Length(S) **Do**
- **If** I mod 2=0 **then**
- S[I]:=LowerCase(S[I]);
- **End;**
- Writeln(S);
- **End.**

EVKLİD məsələsi

- **program** evklid;
- **var** x,y1,y2,n:longint; //dəyişənlərin təsviri,longint -2147483648...+2147483647,integer -32768...+32767
- **begin** {programın govdəsinin başlanğıcı}
- **writeln**('n-i daxil edin!'); {məlumatın ekrana çıxarılması}
- **readln**(n); {n-in oxuması }
- **for** x:=1 **to** n **do** //for operatoru
- **y1:=2*X-3;** //y1 ve y2 daxil edilməsi
- **Y2:=X+2;**
- **if** y1=y2 **then** **writeln**('y1=',y1,' y2=',y2,' y1=y2'); //if şərt operatoru
- **if** y1>y2 **then** **writeln**('kicik eded daxil edin!');
- **if** y1<y2 **then** **writeln**('Daha boyuk eded daxil edin!');
- **end.**

Fibonaçi ədədləri

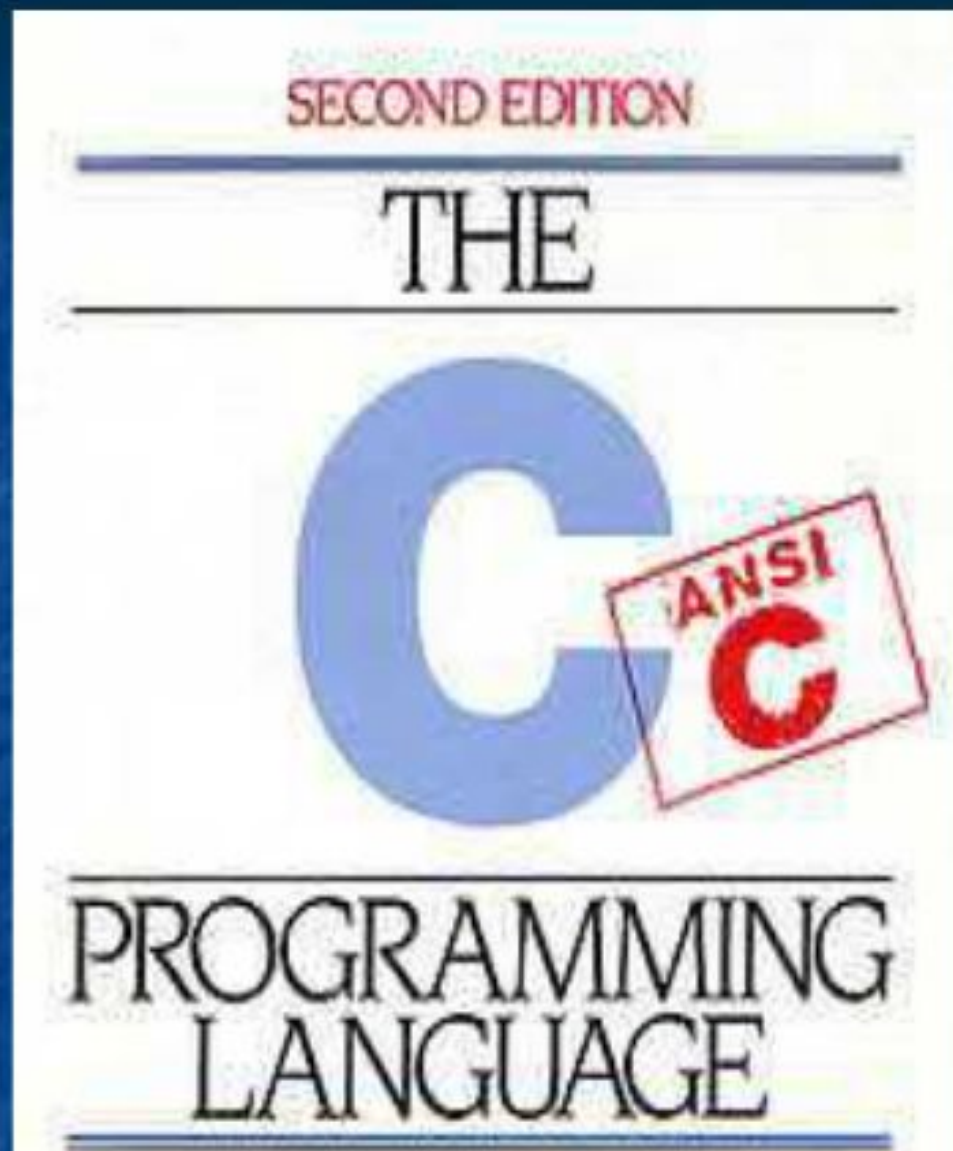
- **program** fibonaci; {program adı}
- **var** n,i,x1,x2,summa:integer; {dəyişənlərin təsviri}
- **label** 1; {nişanın təsviri }
- **begin** {programın gövdəsinin başlanğıcı}
- 1: {goto operatorunun keçidi}
- x1:=0; x2:=1; {sabitlər}
- write('Nece element lazımdır? '); {məlumatın ekrana çıxarılması}
- readln(n); {n-in oxuması }
- **if** n>0 **then** {şərt operatoru}
- **begin** {2}
- writeln ('1-ci element= ',x2);
- **for** i:=2 **to** n **do** {for operatoru }
- **begin** {3}
- summa:= x1+x2 ;
- writeln(i,'-ci element= ',summa);
- x1:=x2; x2:= summa ;
- **end**; {3}
- **end** {2}
- **else**
- Writeln ('Sehv! Novbeti defe musbet eded goturun!');
- **goto** 1; {kecid operatoru}
- **end.** {gövdənin sonu}

C proqramlaşdırma dili 1970-ci illərin əvvəllərində Denis Ritçi və Ken Tompson

tərəfindən UNIX əməliyyat sistemi üçün yaradılmış proqramlaşdırma dil. C proqramlaşdırma dili "A" və "B" dilinin inkişafı məqsədilə yaradılmışdır. Müasir dövrdə bu dil sistem proqramlaşdırması üçün ən güclü proqramlaşdırma dillərindən biri hesab olunur.

Windows, Linux, Unix, FreeBSD və

əməliyyat sistemləri məhz C da



C++

Bu dildə proqramı yazarkən adətən, kompilyator olaraq **Ms Visual Studio** proqramlaşdırma sistemindən istifadə edilir. Ms Visual Studio kompilyatorunun quraşdırma faylını <http://www.microsoft.com/express/Downloads> keçidindən kompüterinizə endirə bilərsiniz.

İlk test:

Ms Visual Studio proqramını yükləyin. Daha sonra **File -> New -> Project** Seçimini edin. **New Project** pəncərəsi açılacaq. Project types: panelindən **Win32**, **Templates:** panelindən isə **Win32 Console Application** seçimini edin.

Daha sonra Name: pəncərəsindən yeni yaradacağımız proqramın adını daxil edirik. Bu pəncərəyə **prog1** yazıb Ok düyməsini basırıq. Açılan yeni pəncərədə Finish düyməsini basırıq. Bu əməliyyatlar ilk proqramımızı tərtib etmək üçün bütün lazımı faylları yaradacaq və proqramın mətn faylı redaktoru pəncərəsi - **prog1.cpp** aktivləşdirəcək. Bu fayl aşağıdakına bənzər formada olur:

```
// prog1.cpp : Defines the entry point for the console // application. #include "stdafx.h" int _tmain(int argc, _TCHAR* argv[]) { return 0; }
```

Bu mətn kompilyator tərəfindən avtomatik yaradılıb. Proqramın mətn faylında aşağıdakı kimi dəyişikliklər edirik. **#include "stdafx.h"** sətirindən sonra **#include**, { mötərəzəsindən sonra isə **std::cout<<"Salam dunya \n";**, **return 0;** sətirindən əvvəl isə **int x; std::cin>>x;** sətrlərini daxil edirik. Aşağıdakı kimi:

```
// prog1.cpp : Defines the entry point for the console // application. #include "stdafx.h" #include int _tmain(int argc, _TCHAR* argv[]) { std::cout<<"Salam dunya \n"; int; std::cin>>x; return 0; }
```

Etdiyimiz dəyişiklikləri saxlamaq üçün File -> Save all düyməsini basırıq. Artıq proqramımızın mətn faylı hazırdır və biz onu kompilyasiya edə bilərik. Kompilyasiya nəticəsində kompilyator bizim mətn faylından prosessor tərəfindən icra oluna bilən ikilik proqram alacaq. Proqramı kompilyasiya etmək üçün Build -> Build Solution əmrini daxil edirik. Proqramımızın kompilyasiyası başlayacaq və Output pəncərəsinə ötürüləcək. Əgər sonda Build: 1 Succeeded ... sətiri çap olunursa deməli proqramımız uğurla kompilyasiya olunmuşdur. İndi isə proqramımızı yerinə yetirək. Bunun üçün Debug -> Start Debugging düyməsini basırıq. Nəticədə kansole pəncərəsi açılacaq və *Salam dunya* mətni çap olunacaq. Proqramı söndürmək üçün klaviaturadan hər-hansı simvol daxil edib enter düyməsinə basmağımız kifayətdir.

Daxil edilən məlumatın növünə və yaddaşda tutduğu yerin həcminə görə dəyişənlər tiplərə ayrılır. Misal üçün *tam ədədlər* tipi – int (4 bayt), *kəsr ədədlər* tipi – double (8 bayt) , *simvol* tipi – char (1 bayt), *sətir* tipi – char [], char * v.s.

Bu tiplərə standart tiplər deyilir. Bundan əlavə C++ dilində *ünvan dəyişənləri tip, struktur tiplər, siyahılar və siniflər (klasslar)* da çox geniş istifadə olunur. Dəyişənlərə istədiyimiz kimi ad verə bilərik yalnız və yalnız həriflərdən (ingilis əlifbasının) , '_' simvolundan və rəqəmlərdən istifadə etməklə. Dəyişənin adı mütləq hərflə başlamalıdır və operator, tip v.s. adlarından da dəyişən adı kimi istifadə etmək olmaz. Operatorlarla gələn mövzularda tanış olacağıq. Beləliklə, cəm proqramında hər iki toplananı və onların cəmini yerləşdirmək üçün biz tam tipli 3 dəyişən təyin etməliyik. Gəlin bu dəyişənləri uyğun olaraq top1, top2 və cem kimi adlandıraraq. Bu dəyişənləri təyin etmək üçün proqram kodu aşağıdakı kimi olacaq. int top1; int top2; int cem; Qeyd edək ki, eyni tiptən olan dəyişənləri vergüllə ayırmaqla bir sətirdə də elan edə bilərik. Aşağıdakı kimi :

```
int top1,top2,cem;  
// prog1.cpp : Defines the entry point for the console application.  
#include "stdafx.h" #include int _tmain(int argc, _TCHAR* argv[])  
{ int top1, top2, cem; top1 = 4; top2 = 6;  
cem = top1 + top2; std::cout<<" 4 ile 6 -nin cemi ="<>x;  
return 0; }
```


Tək ədəd daxil edirik. 10-dan > və ya < şərtinə görə C++ da budaqlanma:

İF OPERATORU

```
#include <iostream> using namespace std;
int main() { setlocale(0, ""); double num;
cout << «İxtiyari ədəd daxil et: "; cin >> num;
if (num < 10) { // 10-dan kiçik olarsa. cout << «Bu ədəd 10-dan kiçikdir." << endl; } else
{ // əks halda cout << «Ədəd 10-dan böyük ya da =-dir." << endl; }
return 0;
}
```

10-dan > və ya < şərtinə görə C++ proqramı müvafiq mətn verəcək **if** operatorunda C++ - da şərt həmişə mötərizədə verilir. Fiqur mötərizədə şərtin **bodisi** olur. Şərt ödəndikdə fiqur mötərizə arasındakı əməllər icra olunur.

Budaqlanmaya nümunə:

```
if (num < 10) { // ədəd 10-dan kiçik olsa. cout << "Bu ədəd 10-dan kiçikdir." << endl; } else { // əks halda cout << «Bu ədəd
10-dan böyük və ya =-dir." << endl; }
```

num dəyişəninin 10-a = halı üçün əlavələr edək:

```
if (num < 10) { // Əgər daxil edilmiş ədəd 10-dan kiçikdirsə. cout << "Bu ədəd 10-dan kiçikdir." << endl; } else
if (num == 10) { cout << "Bu ədəd 10-a bərabərdir." << endl; } else
{ // əks halda cout << "Bu ədəd 10-dan böyükdür." << endl; }
```

3 şərt yoxlanır:

- İ —10-dan kiçik
- İİ — ədəd 10-a = olduqda
- İİİ—10-dan böyük

İf şərtində bərabərlik şərt operatoru `==` istifadə olunur, bu mənimləmə deyil!

`num == 10` yazılışı dəyişəni 10 ədədiylə müqaisə edir.

• Bir `=` -lik, yəni mənimləmə yazəlsaydı dəyişənin qiymətini yeniləşdirərdi (10 edərdi).

Hər `if` operatorunun 1 *else operatoru olur*. Yuxarıdakı programı belə yazmaq olar:

```
#include <iostream> using namespace std; int main() { setlocale(0, ""); double num; cout << "Введите произвольное число: "; cin >> num; if (num < 10) // Если введенное число меньше 10. cout << "Это число меньше 10." << endl; else if (num == 10) cout << "Это число равно 10." << endl; else // иначе cout << "Это число больше 10." << endl; return 0; }
```

Bu üsul daha yığcamdır. Şərt ödəndikdə 1-dən çox əmr yerinə yetirilirsə fiqur mötərizələrdəni istifadə etmək zəruridir.

Nümunə:

```
#include <iostream> using namespace std; int main() { setlocale(0, ""); double num; int k; cout << "Введите произвольное число: "; cin >> num; if (num < 10) { // Если введенное число меньше 10. cout << "Это число меньше 10." << endl; k = 1; } else if (num == 10) { cout << "Это число равно 10." << endl; k = 2; } else { // иначе cout << "Это число больше 10." << endl; k = 3; } cout << "k = " << k << endl; return 0; }
```

Bu proq. `num`. dəyişəninin qiymətini yoxlayır. 10-dan kiçik olarsa, `k` dəyişəninə 1 verir. Əgər `num` 10-a `==`-dirsə, `k` dəyişəninə iki qiyməti verilir. Əks halda, 3 qiyməti verilir. Budaqlanmadan sonra `k` -nın qiyməti ekrana çıxarılır.

1 dən 1000-ə qədər ədədlərin cəmi proqramı:

```
#include <iostream> using namespace std;
int main()
{ int i; // dövr sayğacı
  int sum = 0; // cəm 1 -dəm 1000 - ə qədər.
  setlocale(0, "");
  for (i = 1; i <= 1000; i++) // ilkin qiymət 1, son 1000 dövrün addımı 1.
  { sum = sum + i; }
  cout << "1 dən 1000-ə qədər ədədlərin cəmi = " << sum << endl;
  return 0;
}
```

Proqramın cavabı : **500500-dür.**

Yash, boy, chəki nisbətinin analizi proqramı

```
#include<iostream>
using namespace std;
main (){
    int boy,cheki,ferq , yash;// int tipli "boy,cheki,ferq" deyisenlerini daxil edirik
    int d,d1,netice; // int tipli "d,d1" deyisenlerini daxil edirik
    string ad; // string tipli "ad" deyiseni daxil edirik
    string needek; // string tipli "needek" deyiseni daxil edirik
    int orta=0;// int tipli "orta" deyiseni daxil edib 0 reqemine menimsedirik
    cout << "Adamlarin sayi :";// 'Adamlarin sayi :' ekrana yazilir
    cin >> d1 ;// klavisdən d1 deyisenine istenilen edede menimsede bilerik
    for (d=0;d<d1;d++)// sonra ise d-e 0 menimsedirik , adamlarin sayi yeni d1 boyuk olmalidi d-den ve d adamlarin sayina qeder catanda dovr diyansin
    {
        cout << "Adinizi daxil edin :";// 'Adinizi daxil edin :' ekrana yazilir
        cin >> ad;// klavışden ad deyisenine istenilen adi menimsede bilerik
        cout << "Boy ededi daxil edin : "; // 'Boy ededi daxil edin :' ekrana yazilir
        cin >> boy;// klavisdən boy deyisenine istenilen edede menimsede bilerik
        cout << "Cheki. Ededi daxil edin :";// `Cheki. Ededi daxil edin : ' ekrana yazilir
        cin >> cheki;// klavisdən cheki deyisenine istenilen edede menimsede bilerik
        cout << "Yaşinizi daxil edin :";// Yaşinizi daxil edin : ' ekrana yazilir
        cin >> yash;// klavisdən cheki deyisenine istenilen ededi menimsədə bilərik
        // bu hissede ise ferq deyiseninə boy - 100 - cheki - ni menimsedirik
            ferq= boy - 100 - cheki;
        // bu hissede int tipinde verilmiş orta deyisenine 0 menimsədiyimiz halda funksiyamız orta=orta+ferq (yeni 0+(yuxarıdaki)ferqin cavabi)
        orta = orta +ferq ;
        //bu hissede int tipinde verilmiş netice deyiseni menimsedirik yash deyiseni- 18 / 3
        netice = (yash - 18)/3 ;
        //burada ise ferq deyiseni berabərdir (ferq= boy - 100 - cheki;) burda alınan ferq + (netice = yash - 18/3 ;) burada alınan netice
        ferq = ferq + netice ;
    }
}
```

```

    if(ferq==0 ) needek="  Ideal ceki !   Yasiniz ucun ela neticedi !" ;// bu shertimizde ise ferq deyiseninin cavabi 0 olarsa ekrana needek
deyisenine daxil etdiyimiz Ideal ceki fikri cixir
    else // shertimiz odemesse eks halda asagidaki sherte baxiriq
    {
        if (ferq<0 ) needek = "Siz Ariqlamalisiniz " ;// bu sertimizde ise ferq deyiseninin cavabi 0 dan kicik olarsa ekrana needek deyisenine
daxil etdiyimiz Ariqlamali fikri cixir
        else // shertimiz odemesse eks halda ekrana needek deyisenine daxil etdiyimiz Kokelmeli fikri cixir
        needek = "Siz Kokelmelisiniz" ;
    }
// sonra ise ekrana ad deyiseninin neticesi 'ucun : ' ferq deyiseninin neticesi ve nehayet needek deyiseninin neticesi cixir
cout << ad << " ucun : " << needek << " " << ferq << " kq" ;
cout << "\n";
cout << "-----" << "\n";
}
// burada ise orta deyiseninin (orta = orta +ferq ;)cavabi bolunsuz yeni daxil etdiyimiz adamlarin sayi
orta = orta / d1;
// burada ise ekrana 'Orta beraberdir :' ve orta deyisenin son neticesi (orta = orta / d1) cixir
cout << "Cekiye orta teleb : " << orta << "\n" ;
return 0; // bu kod ise neticemizi gorene qeder ekrani sonmeye imkan vermir. Klavisdən ne ise daxil etdikde kompilyasiya sona catir
}

```


C# proqramlaşdırma dili

C # proqramlaşdırma dili (si şarp şəklində tələffüz edilir), Microsoftun inkişaf etdirmiş olduğu yeni nəsil proqramlaşdırma dilidir. Microsoft tərəfindən yaradılmış .NET Texnologiyası üçün yazılmışdır.

Microsoft məhsulu olsa da ecmain və ISO standartları altına alınmışdır.

C proqramlaşdırma dilində N tam ədəd dəyişənini 1 vahid atırmaq üçün N++ istifadə edilir. C ++ dili adını, C diliylə obyekt yönümlü Proqramlaşdırma üçün əlavələr (C With Classes) almışdır. Bənzər şəkildə C ++ dilinə yeni əlavələr edilərək ((C ++) ++) bir addım daha irəliyə götürülmüş və tamamilə obyektə istiqamətli hazırlanmış C # dilinin isimlendirilmesində, + xarakterlərinin bir-birlərinə yaxınlaşmış halı və bir melodiya açarı olan C # Major istifadə edilmişdir.

Bu dilin hazırlanması Pascal, Delphi derleyicilər və J ++ proqramlaşdırma dilinin dizaynlarıyla bilinən Anders Hejlsberg liderlik etmişdir.

Bir çox sahədə Java'yı özünə nümunə götürər və C # da java kimi C və C ++ kod sözdizimine bənzər bir kod quruluşundadır. .NET Kitabxanalarını istifadə məqsədiylə yazılan proqramların çalışdığı kompüterlərdə uyğun bir kitabxananın və yorumlayıcının olması lazımlıdır. Bu, Microsoftun .Net Framework'u ola biləcəyi kimi ecmain standartlarına uyğun hər hansı bir kitabxana və şərh da ola bilər. Məşhur digər kitabxanalara nümunə olaraq Portable.Net və Mono verilə bilər.

Xüsusilə obyekt yönümlü proqramlaşdırma anlayışının inkişafına kömək olan ən aktiv proqramlaşdırma dillərindən biridir .NET platformasının ana dili olduğu bəzi seqmentlər tərəfindən qəbul edilsə də bəzi mütəxəssislər bunun doğru olmadığını göstərirlər.

C #, .NET orta səviyyəli proqramlaşdırma dillerindəndir. Yəni həm maşın dilinə həm də insan qəbuluna bərabər səviyyədədir. Buradakı orta ifadəsi dilin gücünü deyil maşın dili ilə gündəlik danışmaq dilinə olan məsafəsini göstərir. Məsələn; Visual Basic .NET (VB.NET) yüksək səviyyəli bir dildir desək bu, dilin insanların gündəlik həyatlarında danışma formasına yaxın şəkildə yazıldığını ifadə etməkdədir. Bu səbəbdən VB.NET, C # .NET'ten daha güclü bir dildir deyə bilmərik. Proqramın çalışması istənən kompüterlərdə framework heyəti olması lazımlıdır. (Windows 7 və Windows Vistada .NET Framework quruludur)

C# da Yazılmış program

```
int x, y, z;
Console.WriteLine("Birinci ededi daxil edin : ");
x = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Ikinci ededi daxil edin : ");
y = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Hesabinizi secin - \n 1.Toplama \n 2.Cixma \n 3.Vurma \n 4.Qismet \n ");
z = Convert.ToInt32(Console.ReadLine());
switch (z)
{
    case 1:
    {
        Console.WriteLine("Toplama Emeliyyatinizin Neticesi : ");
        Console.WriteLine(x + y);
        break;}
    case 2:
    {
        Console.WriteLine("Cixma Emeliyyattinizin Neticesi : ");
        Console.WriteLine(x - y);
        break;}
    case 3:
    {
        Console.WriteLine("Vurma Emeliyyatinizin Neticesi : ");
        Console.WriteLine(x * y);
        break;}
    case 4:
    {
        Console.WriteLine("Qismet Emeliyyatinizin Neticesi : ");
        Console.WriteLine(x / y);
        break;}
    default :{Console.WriteLine("Bu cur emeliyyat tetbiq edilmeyib");
        break;}
}
Console.ReadLine();
```

C++ da Yazılmış program

```
#include <iostream>
using namespace std;
main (){
int x, y, z;
cout << "Birinci ededi daxil edin : ";
cin >> x;
cout <<"Ikinci ededi daxil edin : ";
cin >> y;
cout << "Hesabinizi secin - \n 1.Toplama \n 2.Cixma \n 3.Vurma \n 4.Qismet \n ";
cin >> z;
    switch (z)
    {
        case 1:
        {
            cout << "Toplama Emeliyyatinizin Neticesi : ";
            cout << x + y;
            break;}
        case 2:
        {
            cout << "Cixma Emeliyyattinizin Neticesi : ";
            cout << x - y;
            break;}
        case 3:
        {
            cout << "Vurma Emeliyyatinizin Neticesi : ";
            cout << x * y;
            break;}
        case 4:
        {
            cout << "Qismet Emeliyyatinizin Neticesi : ";
            cout << x / y;
            break;}
        default :{cout << "Bu cur emeliyyat tetbiq edilmeyib";
            break;}
    }
return 0;
}
```

C#

Yash, boy, chəki nisbətinin analizi proqramı

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace C_sharp_insanin_cekisi_ve_yasi_gosder
{
    class Program
    {
        static void Main(string[] args)
        {
            int boy, cheki, ferq, yash; // int tipli "boy,cheki,ferq" deyisenlerini daxil edirik
            int d, d1, netice; // int tipli "d,d1" deyisenlerini daxil edirik
            string ad; // string tipli "ad" deyiseni daxil edirik
            string needek; // string tipli "needek" deyiseni daxil edirik
            int orta = 0; // int tipli "orta" deyiseni daxil edib 0 reqemine menimsedirik
            Console.WriteLine("Adamlarin sayini daxil edin : "); // `Adamlarin sayini daxil edin : ' ekrana yazilir
            d1 = Convert.ToInt32(Console.ReadLine()); // klaviaturadan d1 deyisenine istenilen ededi menimsedirik
            for (d = 0; d < d1; d++) // d-e 0 menimsedirik , adamlarin sayi yeni d1 boyuk olmalidi d-den ve d adamlarin sayina qeder
                catanda dövür diyansin
            {
                Console.WriteLine("Adinizi daxil edin : "); // 'Adinizi daxil edin :' ekrana çıxarılır
            }
        }
    }
}
```

```

ad = Convert.ToString(Console.ReadLine()); // klavisdən ad deyiseninə istenilən adı mənimsədə bilərik
Console.WriteLine("Boy ededi daxil edin :");// 'Boy ededi daxil edin :' ekrana yazılır
boy = Convert.ToInt32(Console.ReadLine());// klavisdən boy deyiseninə istenilən ədədə mənimsədə bilərik

Console.WriteLine("Ceki eded daxil edin :");// 'Cheki ededi daxil edin :' ekrana yazılır
cheki = Convert.ToInt32(Console.ReadLine());// klavisdən cheki deyiseninə istenilən ədədə mənimsədə bilərik

Console.WriteLine("Yasinizi daxil edin :");// 'Yasinizi daxil edin :' ekrana yazılır
yash = Convert.ToInt32(Console.ReadLine());// klavisdən yash deyiseni istenilən ədədə mənimsədə bilərik
// bu hissədə isə fərq deyisenini boy - 100 - cheki - e mənimsədirik
// yeni klavisdən boy-a mənimsədiyimiz ədəd - 100 - cheki-e mənimsədiyimiz ədəd
ferq = boy - 100 - cheki;
// bu hissədə isə int tipində verilmiş orta deyiseninə 0 mənimsədiyimiz halda funksiyamız orta=orta+ferq (yeni 0+(yuxarıdakı)ferqin cavabı)
orta = orta + ferq;
//bu hissədə int tipində verilmiş nəticə deyiseni mənimsədirik yash deyiseni- 18 / 3
netice = yash - 18 / 3;
//burada fərq deyiseni bərabərdir (ferq= boy - 100 - cheki);. alınan fərq + nəticə yəni (yash - 18/3 ;) hesablamalıyıq:
ferq = ferq + netice;
if (ferq == 0) needek = " Ideal Ceki !   Yasiniz ucun ela neticedi !"; // bu şərtimizdə isə fərq deyiseninin cavabı 0 olarsa ekrana needek deyiseninə daxil etdiyimiz Ideal ceki fikri çıxır
else // şərtimiz ödənməzsə aşağıdakı şərtə baxırıq
{
    if (ferq < 0) needek = "Siz Ariqlamalısınız "; // fərq deyiseninin cavabı 0 dan kiçik olarsa ekrana needek deyiseninə daxil etdiyimiz Ariqlamalı fikri çıxmalıdır
    else // əks halda
        needek = "Siz Kokelməlisiniz"; // needek deyiseninə Siz Kokelməlisiniz mətni daxil edilir}
// sonra isə ekrana ad deyiseninin nəticəsi, 'ucun : ', fərq deyiseninin nəticəsi və needek deyiseninin nəticəsi çıxır:
Console.WriteLine(ad + " ucun : " + ferq + " " + needek+"\n");
Console.WriteLine("-----");
}
// orta deyiseninin (orta = orta +ferq ;)cavabı bölünsüz yeni daxil etdiyimiz adamların sayı
orta = orta / d1;
// ekrana 'Orta bərabərdir :' və orta deyisenin son nəticəsi (orta = orta / d1; əsasən) çıxır
Console.WriteLine("Orta Bərabərdir : " + orta+"\n");
Console.ReadLine(); // klaviatürdən nə isə daxil etdiyiniz halda cari proqramın icra pəncərəsi bağlanır
}
}
}

```


JAVA



Java – Sun Microsystems şirkəti tərəfindən təqdim olunan obyekt-orientə edilmiş proqramlaşdırma dilinə və texnologiyalara verilən ümumi addır. Java proqram dili struktur və sintaksisinə görə C proqram dilinə çox yaxındır. Rəsmi olaraq 23 may 1995-ci ildə təqdim olunmuşdur.

Başlanğıcda Oak ("palıd") adlandırılan bu dil Ceyms Qoslinq (ing. James Gosling) tərəfindən məişət cihazlarının proqramlaşdırılması üçün yaradılmışdı. Daha sonra bu ad Java ilə əvəz olundu və server proqram təminatının yazılması üçün istifadə olunmağa başladı. Bu proqramlaşdırma dili Java kofe markasının şəərəfinə adlandırılmışdı və buna görə də onun rəsmi embleminin üzərində fincanda buxarlanan kofe təsvir olunmuşdur.

JavaScript və Java arasında heç bir qohumluq yoxdur.

Java dilində dəyişənlər

Dəyişən istənilən vaxt dəyişdirilə bilən qiymətdir. Java dilində əsas 7 baza tipi var:

int, long, float, double, char, String, boolean. Əlavə tiplər : byte, short, BigInteger və object.

Javada dəyişənin təsviri: dəyişənin_tipi dəyişənin_adı başlanğıc_qiymət; .

Javada dəyişənin adı: rəgəm, +, *, #, %, ^, &, !, @, (,), [,], ~, \, /, | simvolları ilə başlaya bilməz!

Əsas tiplərin təsvirinə aid proqram:

```
public class main
{
public static void main(String[] args)
{
int deyisen1; int deyisen2 =18; //deyisene baslangic qiymet 18 menimsedilib
long deyisen3; long deyisen4 = 13000; //deyisene baslangic qiymet 13000 menimsedilib
float deyisen5; float deyisen6 = -3.14f; //deyisene baslangic qiymet -3.14 menimsedilib
double deyisen7; double deyisen8 = 2.7; //deyisene baslangic qiymet 2.7 menimsedilib
char deyisen9; char deyisen10 = 'a'; //deyisene baslangic qiymet a simvolu menimsedilib
String deyisen11; String deyisen12 = «Koroğlu Rəhimov küçəsi»; //deyisene baslangic qiymet - mətn menimsedilib
boolean deyisen13; boolean deyisen14 = true; //deyisene baslangic qiymet kimi true menimsedilib
}
}
```

JAVADA TIPLƏR

byte tipi - 8 bitli tam tipdir. Minimum qiymət $-128(-2^7)$, maksimum qiymət $127(2^7 - 1)$. Məsələn: byte a = -50; byte b = 100;

short tipi - 16 bitli tam tipdir. Minimum qiymət $-32,768(-2^{15})$, maksimum qiymət $32,767(2^{15} - 1)$. Məsələn: short a = -3350; short b = 10001;

int tipi - 32 bitli tam tipdir. Minimum qiymət $-2,147,483,648.(-2^{31})$, maksimum qiymət $2,147,483,647.(2^{31} - 1)$. Məsələn: int a = -9837350; int b = 10804901;

long tipi - 64 bitli tam tipdir. Minimum qiymət $-9,223,372,036,854,775,808.(-2^{63})$, maksimum qiymət $9,223,372,036,854,775,807.(2^{63} - 1)$. Məsələn: long a = 100000L; long b = -200000L;

float tipi - 32 bitli, sürüşən vergüllü həqiqi tipdir. Məsələn: float deyisen = 345.3453f;

double tipi - 64 bitli, sürüşən vergüllü, ikiqat dəqiqli həqiqi tipdir. Məsələn: double deyisen = 345.3453;

boolean tipi - məntiqi tipdir. true/false qiymətlərdən birini alır. Susmaya görə false qiymət verir. Məsələn: boolean bool = true;

char tipi - simvol tipidir. Min. '\u0000' (yaxud 0), Max. '\uffff' (yaxud 65,535). Məsələn: char letter = 'A';

NÜMUNƏ

```
package javadersleri.eu.pn;
```

```
public class DataTypes {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Byte type");
```

```
        byte a = -99;
```

```
        byte b = 120;
```

```
        System.out.println("a = "+a+" b = "+b);
```

```
        System.out.println(Byte.MIN_VALUE);
```

```
        System.out.println(Byte.MAX_VALUE);
```

```
System.out.println("Short type");
    short c = -919;
    short d = 27120;
    System.out.println("c = "+c+" d = "+d);
    System.out.println(Short.MIN_VALUE);
    System.out.println(Short.MAX_VALUE);
System.out.println("Integer type");
    int e = -999999999;
    int f = 1287438790;
    System.out.println("e = "+e+" f = "+f);
    System.out.println(Integer.MIN_VALUE);
    System.out.println(Integer.MAX_VALUE);

    System.out.println("Long type");
    long g = -92372036854775808L;
    long h = 1257758437589599850L;
    System.out.println("g = "+g+" h = "+h);
    System.out.println(Long.MIN_VALUE);
    System.out.println(Long.MAX_VALUE);
```

```
System.out.println("Float type");
float k = -34.443535f;
float l = 44.444545f;
System.out.println("k = "+k+" l = "+l);
System.out.println(Float.MIN_VALUE);
System.out.println(Float.MAX_VALUE);

System.out.println("Double type");
double m = -374.4485;
double n = 7587483.487;
System.out.println("m = "+m+" n = "+n);
System.out.println(Double.MIN_VALUE);
System.out.println(Double.MAX_VALUE);

System.out.println("Char type");
char letter = 'A';
System.out.println("letter = "+letter);

System.out.println("Boolean type");
boolean bul1 = false;
boolean bul2 = true;
System.out.println("bul1 = "+bul1+" bul2 = "+bul2);
}
}
```

Java programında dövrlərin qurulması

Dövrlər program içərisində dəfələrlə icra olunan komandalardan ibarət fraqmentlərdir. Java dilində 2 növ fraqmentlərdən istifadə olunur:

1. “n dəfə” dövrlər

2. “nə qədər ki” dövrlər

Adından göründüyü kimi 1 tip verilmiş əməliyyatın tələb olunan sayda icrası üçündür. Məsələn “n” faktorialda olduğu kimi yəni eyni ədəd “n” dəfə 1-dən n-ə qədər ardıcıl ədədlərə vurulur.

“n” dəfə dövr tipi (for operatoru)

```
For operatorunun 3 parametri var. 1 İnizializasiya 2 təkrarlanma şərti 3 İterasiya For (inizializasiya; şərt ;İterasiya) {  
//dövrün bədəni yəni dövrü təkrar olunan hərəkət  
}
```

Birinci parametrdə adətən bir dəyişən seçilir ki, onun köməyiylə dövrün təkrarlanmaları sayılır. Bu dəyişənə sayğac deyilir.

Sayğaca tələb olunan ilkin qiymət verilir.

İkinci parametrdə sayğacın məhdudlaşdırılması şərti göstərilir. Yəni sayğacın hansı qiymətə dəyişəcəyi təyin edilir.

Üçüncü parametrdə dövrün hər addımından sonra sayğacı dəyişdirən ifadə göstərilir . Bir qayda olaraq iterasiya inkrement və ya dekrement olur . yəni bir bir artır və ya bir -bir azalır. Amma sayğaca zəruri qiymət mənimsədən ixtiyari ifadədən istifadə etmək olar.

Dövrün birinci addımından qabaq sayğaca ilkin qiymət mənimsədir. Buna inisializasiya deyilir. Bu əməliyyat yalnız bir dəfə icra olunur.

İnizializasiyadan sonra dövrün hər bir addımından qabaq təkrarlanma şərti yoxlanılır. Əgər şərt doğrudursa dövrün bədəni növbəti dəfə icra olunur. Bu zaman dövrün bədəni bir dəfədə icra olunmaya bilər. Bunun üçün birinci yoxlamada şərt yalan olmalıdır. Dövrün hər bir addımı qutardıqdan sonra təkrarlanma şərti yoxlanmaqdan qabaq iterasiya icra olunur.

Aşağıdaki program 1-dən 100-ə qədər ədədləri ekrana çıxarır:

```
For(int i=1;i<=100;i++) {  
System.out.print(i + " ");
```

7-dən məni 7-ə qədər ədədləri ekrana çıxaran program

```
For( int s =7; s> -8 ; s--) {  
System.out.print(s + " ");
```

Aşağıdaki program 2-dən 100-ə qədər cüt ədədlərin cəmini və hasilini hesablayır:

```
Double prod =1;    // sum cəm, prod hasil üçün  
İnt sum=0  
For (int j=2 j<=100; j=j+2)  
{ sum=sum+j}  
System.out.println(sum);  
System.out.println(prod);  
Prod=prod * j
```

1 dövr daxilində bir neçə sayğac istifadə etmək olar. Bu zaman iterasiya və inisializasuya daxilində olan ifadələr vergüllə ayrılır. Təkrarlanmaq şərti yalnız bir dənə olur. Amma o da özündə bir neçə sayğac saxlıyan ifadədən ibarət ola bilər.

Maaşı hesablayan sadə proqram:

```
import java.util.Scanner;
public class maas {

public static void main(String[] args) {

double yekunmaas;
int maas;
double cem = 0;
System.out.println("n-adamlarin sayi");
Scanner m = new Scanner(System.in);
int n = m.nextInt();
System.out.println("adamlarin sayi n="+n);
double reis = 0;
System.out.println("reisin payi-");
Scanner r = new Scanner(System.in);
reis = r.nextInt();//reise ayrilmis faiz
int s;

//emekdaslarin sayi,iw gunleri
//tutulan faiz
```

// DAVAMI

```
int saatsay;

int i=0 ;
while (i < n)
{ //fiquer moterize icerisinde tekrarlanan emeliyatlar verilir
System.out.println("isci");
Scanner a = new Scanner(System.in);
String isci = a.nextLine();

System.out.println("gunluk maas:");
Scanner gunluk = new Scanner(System.in);
int gunm = gunluk.nextInt();

System.out.println("is gunlerinin sayi:");
Scanner gun = new Scanner(System.in);
int gunsayi = gun.nextInt();//is gunlerin sayin daxil edir
System.out.println("Vergi faizi:");
Scanner ver = new Scanner(System.in);
int vergi = ver.nextInt();//her adamin verdiyi verginin faizi
```

```

System.out.println( "iscinin Adi:" +isci + " vergi faizi = " +vergi+ " reisin faizi = " + reis + " gunsayi= " +gunsayi+ "
gunluk maas= " +gunm );
maas =gunsayi*gunm;//novbeti adamin maasi hesablanir
System.out.println(" ayliq maas= " + maas);
yekunmaas=maas-maas/100*vergi;
System.out.println("yekun maas= "+yekunmaas);
double rhaqqi=yekunmaas/100*reis;
yekunmaas=yekunmaas-maas/100*reis;
System.out.println(" cari adamin " + isci+ " yekunmaas=" + yekunmaas);
    cem=cem+=yekunmaas;
i++;
double reiscem = reiscem+rhaqqi;
// TODO Auto-generated method stub
}
double orta=cem/n;
System.out.println("reisin adi nedir?");
Scanner a = new Scanner(System.in);
    String reisad = a.nextLine();
    System.out.println(" Son! *** adamlarin sayi - i= " +i+ " cem = " +cem + " orta = " +orta + " "+ reisad + " reisin
haqqi= " +reis );
System.out.println("birinci maashinizi tebrik edirem!" );
}
} }

```

Aşağıdakı ifadə verilmiş ardıcılığı ekranda yaradacaq: “0 -1 -4 -9 -16 -25”

```
For(int a=3 i=1 i<=10 a=2*a-2 i++) { system.out.print(a+ “ “); }
```

Nəqədəki tipli dövr (while və do...while)

While operatoru onun parametric doğru qiymət aldığı sürəcdə verilmiş əməliyyatı icra edir. məsələn aşağıdakı dövr 5 dəfə icra olunaraq “ 1 2 3 4 5 “ ədədlərini çıxarır.

```
İnt i=1;
```

```
While (i<6)
```

```
{system.out.print(i+””); i++}
```

Aşağıdakı dövr dayanmadan təkrarlanaraq ekrana” 1*2*3*4*5*6*7.....”

```
İnt i=1
```

```
While (true)
```

```
{ system.out.print(i + “ *”);
```

```
i++;
```

```
}
```

Göründüyü kimi şərt birincidən qabaq bütün addımlarda yoxlanılır (öncə yoxlamaq) şərtin əməliyyatın icrasından sonra yoxlanılması üçün do...while operatorundan istifadə olunur. (sonra yoxlamaq)

Asağıdakı dövr 4 dəfə icra olunaraq ekrana”2/3/4/5/” çıxarır:

```
İnt i=1
```

```
Do{
```

```
    i++;
```

```
System.out.print(i + “/”);
```

```
} while (i<5);
```

Aşağıdakı program 1-10 intervalında hər hansı tam ədəd götürüb istifadəçiyə tapmağı təklif edir. İstifadəçi klaviaturada program axtarılan ədədin istifadəçinin daxil etdiyindən böyük və ya kiçik olduğunu göstərir:

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) { // prog — число созданное программой user — число введённое пользователем
        int prog, user;
        // Генерируем случайное целое число от 1 до 10
        prog = (int)(Math.random() * 10) + 1;
        System.out.println("Я загадала число от 1 до 10, отгадайте его.");
        System.out.print("Вводите ваше число: ");
        Scanner input = new Scanner(System.in);
        // Проверяем, есть ли в потоке ввода целое число
        if( input.hasNextInt() ) {
            do { // tam ədədi daxil edirik
                user = input.nextInt();
                if(user == prog) {
                    System.out.println("Вы угадали!");
                } else {
                    // Проверяем, входит ли число в отрезок [1;10]
                    if (user > 0 && user <= 10) {
                        System.out.print("Вы не угадали! ");
                        // Если число загаданное программой меньше...
                        if( prog < user ) {
                            System.out.println("Моё число меньше.");
                        } else {
                            System.out.println("Моё число больше.");
                        }
                    } else {
                        System.out.println("Ваше число вообще не из нужного отрезка!");
                    }
                }
            } while( user != prog );
        } else {
            System.out.println("Ошибка. Вы не ввели целое число!");
        }
        System.out.println("До свиданья!");
    }
}

```


Əməliyyat

&&

||

!

^

Əməliyyat

>

>=

<

<=

==

!=

Əməliyyat

+

-

*

/

%

İstifadə

op1 && op2

op1 || op2

!op

op1 ^ op2

İstifadə

op1 > op2

op1 >= op2

op1 < op2

op1 <= op2

op1 == op2

op1 != op2

İstifadə

op1 + op2

op1 - op2

op1 * op2

op1 / op2

op1 % op2

True-olur. Əgər:

op1 и op2 оба истины (конъюнкция)

один из op1 или op2 истинен (дизъюнкция)

op — ложь (отрицание)

op1 и op2 различны (исключающее или)

True-olur. Əgər:

op1 больше чем op2

op1 больше или равен op2

op1 меньше op2

op1 меньше или равно op2

op1 и op2 равны

op1 и op2 не равны

Funksiyası

Складывает op1 и op2

Вычитает op1 из op2

Умножает op1 на op2

Делит op1 на op2

Вычисляет остаток от деления op1 на op2

Hər bir ikilik əməliyyat üçün məxsusi mənimsetmə operatoru var.

```
x += 8; // то же самое, что x = x + 8 (x увеличивается на 8)
*=, -=, /=, %=, &= ^= və s..
x *= 3; // то же самое, что x = x * 3 (x увеличивается в 3 раза)
b1 ^= b2; // то же самое, что b1 = b1 ^ b2
```

Tapşırıq

2 tam ədəd elan edib onlara qiymət mənimlədin. Cəm və hasilini və ədədi ortanə tapın! Ekranı çıxarın!

İnkrement və Dekrement

Operasiyalar && və || fərqlənirlər, çünki onlar lazımsız hesablamırlar ikinci [operandı](#). Nəzərə alın, && hesablamırlar birinci operandı və, əgər o yanlışdır, dərhal `false` qaytarır, `||` qaytarır `true` dərhal, əgər görünür, ki, birinci operand – həqiqət. Java-da eyni fəaliyyətə operasiyalar & və |, onlar hesablamırlar hər iki operandın qiymətini, əvvəlcədən onlara fəaliyyət etməyə imkan verirlər.

Mənimləmə

Sonra, necə ki, dəyişən təsvir edilmişdir, onla işləmək mümkündür. Xüsusi olaraq, onlara qiymət təyin edilə bilər. Bundan sonra, bu dəyişəni istifadə edərkən, onlar həmişə bu qiyməti istifadə edirlər.

Qiymət əlaqələndirilir dəyişənlə [operator](#) istifadə edilərək. Java-da bu işləmə bərabərlik işarəsi ilə edilir:

```
deyişən = ifadə;
```

Sol tərəfdən operator istifadə edilərkən, həmişə dəyişən göstərilir. [İfadə](#) sağda olmalıdır, dəyişənə uyğun. Onlar yalnız [litaral](#) (məsələn, rəqəm və ya simvol) ola bilər:

```
x = 7; // dəyişən x əlaqələndirilir qiymət letter = 'Q'; // dəyişən letter əlaqələndirilir qiymət 'Q'
```

Ümumiyyətlə, ifadə — bu, nə olursa olsun hesablanmışdır (məsələn, rəqəm, rəqəm hesablamı, rəqəm, qaytarılan bəzi metodlar):

```
a = 7.5 + 2.4; // dəyişən a əlaqələndirilir 9.9 kimi rəqəm hesablamı
```

İfadədə, litarallarla yanaşı, digər dəyişənlər də istifadə edilə bilər. Onlar yalnız onların hazırkı qiymətini istifadə edirlər. Nəticədə, komanda:

```
b = a + 1;
```

deyişən b qiymətini 10.9 qaytarır.

Beləliklə, operator istifadə edilərkən, əvvəlcə hesablamırlar ifadənin qiymətini sağda, sonra isə hesablanmış rəqəm dəyişənlə əlaqələndirilir, göstərilən sol tərəfdə.

Bəzən belə bir vəziyyət mümkündür:

```
x = x + 4;
```

Bu komanda rəqəm qiymətini artırır. Beləliklə, x qiymətini 4 artırır.

Aşağıdakı komandalər yanlışdır və işləməyəcəklər:

```
5 = x + 7; // sola qoymalı deyil dəyişən + 3 = 14; // sola qoymalı deyil bir dəyişən = 4.5; // dəyişən yalnız rəqəm qiymətini qəbul edə bilər
```

Eclipse sizə xəbərdarlıq göstərəcək, çünki bu komandalər hələ də işləməyəcəklər, lakin sizə xəbərdarlıq göstərəcək.

JAVAda riyazi funksiyalar

`Math.abs(n)` — **modul**. возвращает модуль числа n .

`Math.round(n)` — **Yuvarlaqlaşdırma**. возвращает целое число, ближайшее к вещественному числу n (округляет n).

`Math.ceil(n)` — **Yuvarlaqlaşdırma**. возвращает ближайшее к числу n справа число с нулевой дробной частью (например, `Math.ceil(3.4)` в результате вернёт 4.0).

`Math.cos(n)`, `Math.sin(n)`, `Math.tan(n)` — **Triqonometrik**. тригонометрические функции \sin , \cos и \tan от аргумента n , указанного в радианах.

`Math.acos(n)`, `Math.asin(n)`, `Math.atan(n)` — **Əks Triqonometrik** обратные тригонометрические функции, возвращают угол в радианах.

`Math.toDegrees(n)` — **Bucaq**. возвращает градусную меру угла в n радианов.

`Math.toRadians(n)` — **Radian**. возвращает радианную меру угла в n градусов.

`Math.sqrt(n)` — **Kvadrat Kök**. возвращает квадратный корень из n .

`Math.pow(n, b)` — **Kvadrat Kök**. возвращает значение степенной функции n в степени b , основание и показатель степени могут быть вещественными.

`Math.log(n)` — **Loqaritm**. возвращает значение натурального логарифма числа n .

`Math.log10(n)` — **Onluq Loqaritm**. возвращает значение десятичного логарифма числа n .

`Math.random()` — **Təsadüfi ədədlər**. возвращает псевдослучайное вещественное число из промежутка $[0;1)$.

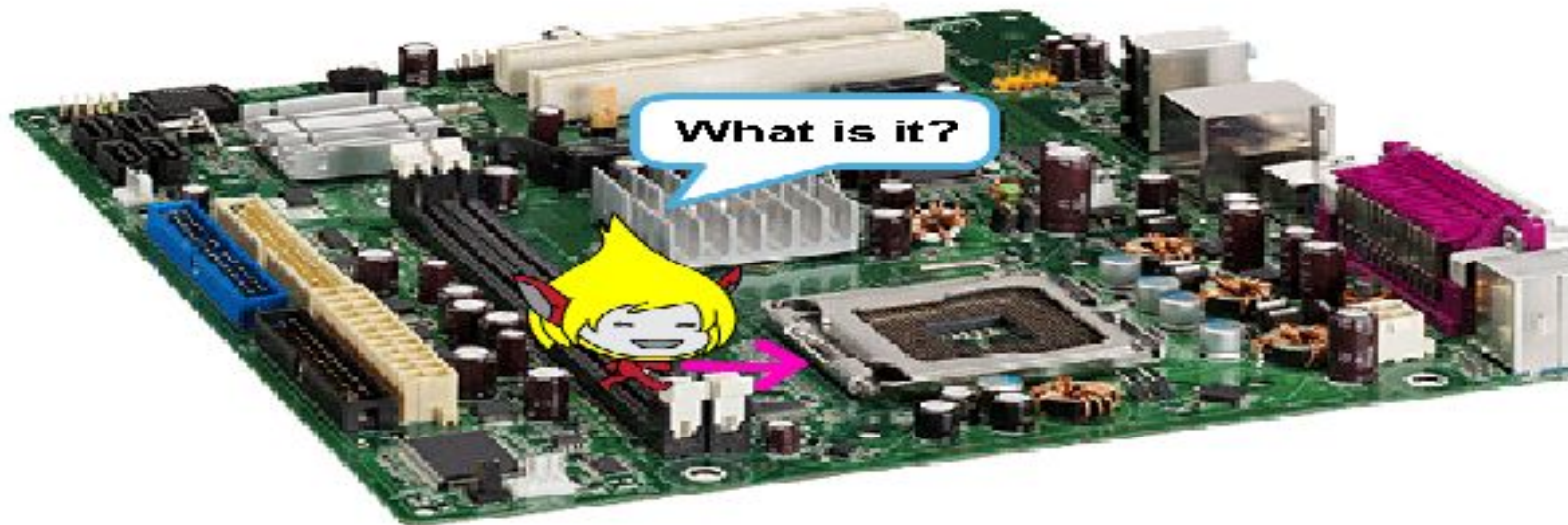
This project is not shared -- so only you can see it. Click share to let everyone see it

Untitled-3

by Tahir5282



v450.1



What is it?

Untitled-3
v461 by Tahir5282 (unshared)

What is it?

x: 240 y: -85

Sprites

New sprite: [] [] [] []

Stage
4 backdrops

New backdrop: [] [] [] []

Giga2

Scripts Costumes Sounds

Motion

- move 10 steps
- turn 15 degrees
- turn 15 degrees
- point in direction 90
- point towards mouse-pointer
- go to x: -69 y: -11
- go to mouse-pointer
- glide 1 secs to x: -69 y: -11
- change x by 10
- set x to 0
- change y by 10
- set y to 0
- if on edge, bounce
- set rotation style left-right
- x position
- y position
- direction

Events

- Control
- Sensing
- Operators
- More Blocks

when green flag clicked

- hide
- switch backdrop to welcome
- wait 1 secs
- show
- switch costume to catherine-c
- wait 1 secs
- say Salam
- wait 1 secs
- say Hello!
- go to x: -13 y: 9
- wait 1 secs
- switch costume to Giga walk4
- wait 1 secs
- glide 1 secs to x: -202 y: -30
- wait 1 secs
- ask What's your name? and wait
- say Good!!!
- wait 1 secs
- switch backdrop to motherboard-0
- wait 1 secs
- switch costume to giga-b
- wait 1 secs
- ask What is it? and wait
- repeat until answer = motherboard
 - switch costume to giga-d
 - say mistake
 - wait 1 secs
 - switch costume to giga-a2
 - ask Give the right answer and wait
- switch costume to giga-c
- say you are quite right

x: -202 y: -30

Scratch programinin mәtni

```
when green flag clicked
hide
switch backdrop to welcome
wait 1 secs
show
switch costume to catherine-c
wait 1 secs
say Salam
wait 1 secs
say Hello!
go to x: -13 y: 9
wait 1 secs
switch costume to Giga walk4
wait 1 secs
glide 1 secs to x: -202 y: -30
wait 1 secs
ask What's your name? and wait
say Good!!!
wait 1 secs
switch backdrop to motherboard-0
wait 1 secs
switch costume to giga-b
wait 1 secs
ask What is it? and wait
repeat until answer = motherboard
switch costume to giga-d
```

```
say mistake
wait 1 secs
switch costume to giga-a2
ask Give the right answer and wait

switch costume to giga-c
say you are quite right
wait 1 secs
switch costume to giga-b
wait 1 secs
glide 1 secs to x: 1 y: -39
ask What is it? and wait
repeat until answer = cpu socket
switch costume to giga-d
say mistake
wait 1 secs
switch costume to giga-a2
ask Give the right answer and wait

switch costume to giga-c
say you are quite right
wait 1 secs
switch costume to giga-b
ask That can be set here? and wait
repeat until answer = Cpu
switch costume to giga-d
say mistake
wait 1 secs
```

```
switch costume to giga-a2
ask Give the right answer and wait

glide 1 secs to x: -69 y: -11
ask What can be set here? and wait
repeat until answer = ram
switch costume to giga-d
say mistake
wait 1 secs
switch costume to giga-a2
ask Give the right answer and wait

switch costume to giga-c
say you are quite right
wait 1 secs
glide 1 secs to x: -171 y: -110
say Fine
forever
switch costume to catherine-c
wait 1 secs
switch costume to catherine-b
wait 1 secs
switch costume to catherine-d
```