
# کمپیوټري ژبې



کمپیوټرستان (۱)

# کمپیوټري ژبې

~~~~~~~~~~***~~~~~~~~~~

## کمپیوټرستان(۱)

**ژبـــاړه:** رحمت شاه فراز
~~~~~~~~~~

# د کتاب پېژندنه

# کمپیوټري ژبې

**لیکوال:**

هېري هېنډرسن

**ژباړه:**

رحمت شاه فراز



اړیکه:

078 063 48 38

بریښنالیک:

rsfaraz4@gmail.com

# لیکلر

# سریزه

کمپیوټرپوهنه که په افغانستان کې نوی علم نه وي، نو ډېر زوړ هم نه دی، او لویه نښه یې دا ده چې په دې برخه کې تر اوسه پورې په پښتو ژبه یو باوري او معتبر اثر هم نه لرو. په دې برخه کې تحقیق، څېړنه، د لویو پوستغالو پنځول او د سختغالي په برخه کې نوښت او ابتکار کول خو په ډېر لرې راتلونکي کې هم نه تر سترګو کېږي.

که څه هم نن ورځ، یاده څانګه شاوخوا د هېواد په ټولو دولتي پوهنتونو کې ښوول کېږي او شخصي پوهنتونونو هم د اوسني عصر د غوښتنو له مخې کمپیوټرپوهنې ته تر ډېره پاملرنه کړې؛ او همدا رنګه دا هم له پامه نه شو غورځولای چې د کمپیوټرپوهنې په ځینو برخو لکه د ډېسکټاپ او موبایل کاریالونو او ویبسایټ جوړونې په ډګر کې ډېرې شخصي کمپنۍ او سافټویر کورونه په کار بوخت دي؛ ترڅنګ یې لوی شمېر دولتي اورګانونو هم خپلې چارې کمپیوټرې کړې، او لا هم د یوه (کمپیوټر شوي افغانستان) په هیله او موخه خپلې هڅې او زحمتونه کوي؛ خو څومره چې باید د یوه علم د پرمختګ، ارتقا او ودې لپاره کار بویه، له بده مرغه په دې ډګر کې هغه کچه کار او لاسته راوړنه نه لرو او لاهم له لویو تشو سره مخ یو. مګر د دې ټولو یادو ټکو ترڅنګ دا د نېکمرغۍ او خوشالۍ ځای دی چې یادې څانګې ته زمور خلکو د اړتیا وړ لېوالتیا ښودلې او که لاسته راوړنې و هم نه لرو، خو د دې څانګې پر وړاندې کوم مانع او خنډ هم نه ترسترګو

کېږي، او د هېواد اکثریت وګړو د دې برخې اړتیا د یوه هېواد د پرمختګ او ودې لپاره درک کړې او په پرانیسته سینه ورته ولاړ دي.

همدې اړتیا او تشې ته په کتو پر دې کتاب لاس پورې شوی. د کمپیوټري ژبو چې پروګرامي[1] ژبې هم ورته وایي، په برخه کې ښایي دا لومړنی کتاب وي، چې د دې څانګې مینه والو ته په پښتو ژبه وړاندې کېږي. د «کمپیوټري ژبې» کتاب له دوو کتابونو څخه را ژباړل شوې مقالې دي: لومړی، «د کمپیوټرپوهنې او ټکنالوژۍ پوهنغونډ» چې د کمپیوټرپوهنې د نړۍ تکړه او د ډېرو اثارو څښتن لیکوال «هېري هېنډرسن» لیکلی او که څوک غواړي چې د کمپیوټرپوهنې له هر کونج څخه ځان خبر کړي او په دې ناپایه کاینات کې قدمونه ووهي، نو زما تر سترګو له دې نه زیات علمي، مستدل او ځېرنیز کتاب نه دی تېر شوی. دوهم کتاب د «پي. کې. سینا»، «د کمپیوټر اساسات» دی چې د کمپیوټر په بنسټې مفاهیمو او مسایلو د پوهېدو لپاره یو لنډ او غونډ اثر دی چې په خپل ځای یې یو ارزښتناک او ګټور اثر ګڼلی شو.

له لومړي کتاب څخه کمپیوټري ژبې او له وروستي کتاب څخه د کمپیوټري ژبو اساسات او پېژندنه را ژباړل شوي. په دې ترتیب، له دوهم کتاب څخه را ژباړل شوې محتوا د «کمپیوټري ژبې » تر عنوان لاندې پیل کېږي او له لومړي کتاب څخه را ژباړل شوې محتوا د «اېډا» په عنوان پیل او د کتاب تر پایه غځېږي.

[1] Programming Language

د کتاب د ژباړې په هکله به هم په دې ځای کې ځینې ټکو ته اشاره ښه وي. دا چې په دې برخه کې تر دې دمخه هیڅ کتاب نه وو ژباړل شوی (یا زما ترسترګو نه دی تېر شوی)؛ او د کمپیوټرپوهنې د اصطلاحاتو لپاره پښتو بدیلونه نه لرو؛ او که یې لرو، نو هغه هم یوازې د معارف په کتابونو کې څه نا څه لیدل کېږي او بیا د کمپیوټري ژبو موضوع په هکله خو دا برخه له نشت سره مخ ده. له همدې امله، د کتاب د ژباړې په اوږدو کې تل دوه وېرې راسره وې: یوه وېره دا وه چې که چېرته د کمپیوټرپوهنې اصطلاحات را ونه ژباړم، نو عادي لوستونکي به خندی کوي چې دا نو څنګه ژباړه ده؟ یوازې انګلیسي متن په پښتو تورو لیکل شوی. او بل خوا که د ژباړې هڅه یې وکړم او د کمپیوټرپوهنې د علمي اصطلاحاتو لپاره پښتو بدیلونو راوړم، نو د دوه ډله کسانو تر نیوکو لاندې به راشم. د کمپیوټرپوهنې زده کوونکي او زده کړیالان به نیوکه کوي چې هیڅ نه وی ژباړل شوی، ښه به وه، ځکه ژباړه یې تر انګلیسي هم سخته ده او یا د اروابنباد شپون په خبره چې عبدالحی حبیبي صېب به مجاور احمد زیار ته ویل چې ما ډېر نوي او زاړه متنونه لوستي دي او وربانديې پوه شوی یم، خو ستا په لیکنو نه پوهېږم. دوهمه ډله د ژبې ماهرین دي چې نیوکه به یې دا وي چې د ژباړې اصول او معیارونه پکې له پامه غورځول شوي.

په دې کرلو او ربیلو کې مې زړه پر دې پریکړې اوبه وخښلې چې هغه اصطلاحات چې په اسانه د ژباړلو وړ وي او له ژباړې وروسته به د کمپیوټرپوهنې لوستونکي هم په اسانه وربانديې پوهېدای شي او ورسره د ژباړې

اصول او معیارونه هم پکې مراعت شوي وي، هغه به وژباړم؛ او کوم چې پاتې شوو، په هغه برخه کې به د دې علم له لوستونکو، زده کوونکو، زده کړیالانو، ښوونکو او ماهرینو نه هیله وکړم چې وړ او معیاري بدیلونه ورته ومومي او د یاد علم په بډاینه کې چې د یوه تن د وس کار نه دی، وندې واخلي. د کتاب په دریم مخ کې مې خپله اړیکه او بریښنالیک همدې ته په کتو زیات کړي.

خبرې اوږدې هم شوې او پاتې هم دي؛ ښایي لامل یې دا وي چې په دې برخه کې به خبرې زیاتې وي. خو نورې خبرې به بیا وشي، ځکه دا د «کمپیوټرستان» د زنځیر لومړۍ کړۍ ده، نو په دې خاطر خپلې خبرې به بیا په نورو کړیو کې کړۍ کړۍ کړم.

ستاسو

رحمت شاه فراز

# د «کمپیوټر اساسات»

# له کتاب څخه

**لیکوال:** پي.کې. سینا

**ژباړه:** رحمت شاه فراز

# کمپیوټري ژبې

## له طبیعي ژبو سره یې ورته والی

ژبه د پوهېدلو او پوهولو (افهام او تفهیم) یو سیستم دی. په خپلو طبیعي ژبو لکه په پښتو کې نور وګړي په خپلو احساساتو او نظریاتو پوهوو. په همدې ډول، د کمپیوټر ژبه د افهام او تفهیم هغه وسیله ده چې د انسانانو او کمپیوټر ترمنځ د اړیکو ټینګولو لپاره پکارېږي. د یوې کمپیوټري ژبې په مرسته، یو پروګرام لیکونکی[1] کمپیوټر ته وایي چې دی څه غواړي. هره طبیعي ژبه (لکه انګلیسي، فرانسوي، جرمني، او داسې نور) د افهام او تفهیم د موخې لپاره د معیاري سمبولونو یوه ټولګه په کار اچوي. پر دې سمبولونو د هغې ژبې ټول ټکټه اخیستونکي پوهېږي. مور په عامه توګه د هغې ځانګړې ژبې د سمبولونو ټولګې ته ویېپانګه وایو. د مثال په ډول، هغه کلمات چې مور یې په انګلیسي کې کاروو، د انګلیسي ژبې سمبولونه دي چې د دې ژبې ویېپانګه جوړوي. هره کلمه ځانګړې معنا لري چې په سیند (قاموس) کې موندل کېدای شي. په ورته ډول، د کمپیوټر ټولې ژبې خپله ځانګړې ویېپانګه لري. د دې ویېپانګې هر سمبول ځانګړې او څرګنده معنا لري چې د هغې ژبې په خپل لارښود کتاب کې لیدل کېدای شي. په دې اساس، د کمپیوټري ژبې هر سمبول کمپیوټر ته د یوه ځانګړي کار لارښوونه کوي. د طبیعي او کمپیوټري ژبو ترمنځ اصلي توپیر دا دی چې

1 Programmar

طبیعي ژبې د کلماتو پراخه زېرمه لري، خو ډېرې کمپیوټري ژبې یوازې له یوې محدودې او ټاکلې ویبپانګې څخه ګټه پورته کوي. د دې اصلي لامل دا دی چې پروګرامي ژبه [1] د خپل طبیعت او هدف له مخې د ډېرو خبرو کولو ته اړتیا نه لري. هر ډول ستونزه چې یو کمپیوټر یې باید حل کړي، پکار ده چې په ټاکلو (ساده او بېلو) برخو او منطقي پړاوونو ووېشل شي چې په اساسي ډول په څلورو عملیو[2] بنا دي: ورکړیز او راکړیز عملیات[3]، ریاضیکي عملیات، د مرکزي پروسېس کوونکي ترمنځ د معلوماتو د حرکت عملیات، او منطقي یا مقایسوي عملیات[4].

هره طبیعي ژبه د خپلو سمبولونو د کارونې لپاره یو سیستماتیک میتود لري. په انګلیسي کې، دغه میتود د ګرامري اصولو په نامه وړاندې کېږي. په اصل کې همدا اصول مور ته وایي چې کوم کلمات وکاروو او په څه ډول یې وکاروو. په ورته ډول، د یوې ځانګړې کمپیوټري ژبې سمبولونه هم باید د خپل ټاکلو اصولو له مخې چې د نحوي ترکیب[5] په نوم پېژندل کېږي، وکارول شي. د طبیعي ژبې په برخه کې که له ناسمو کلماتو او غلط ګرامر څخه هم ګټه واخیستل شي، نو بیا هم کولای شو چې د یو بل په خبرو پوه شو. خو، کمپیوټر چې یو ماشین دی، یوازې هغې کره ویبپانګې ته ځواب ورکوي چې په سمه توګه او د ژبې له نحوي

1 Programming Language
2 Operations
3 Input/Output Operations
4 Logical or Comparison Operatoins
5 Syntax

اصولو سره سم وکارول شي. په دې اساس، په کمپیوټري ژبو کې، که چېرته غواړو چې د کمپیوټر په خبرو پوه شو، نو باید د هغې ژبې کړه اصولو ته ژمن پاتې شو. که نه نو، هېڅ کمپیوټر د دې وړتیا نه لري چې ناسمې لارښوونې [1] سمې کړي او یا معنا ترې واخلي. کمپیوټري ژبې د طبیعي ژبو په پرتله کوچنۍ او ساده دي، خو باید په ډېر دقت وکارول شي.

تر څو پورې چې یو پروګرام لیکونکی د یوې پروګرامي ژبې نحوي اصولو ته ژمن نه پاتې کېږي، تر دې چې که لیکنښې هم په سمه توګه ونه کاروي، نو کمپیوټر به د هغه په لارښوونو د پوهېدو جوګه نه شي.

پروګرامي ژبو هم د کلونو په اوږدو کې هماغسې پرمختګ کړی، لکه څنګه چې د کمپیوټر هارډوېر ارتقا کړې. ژبو له ماشینواله ژبو څخه [2] چې بایټري متنکرښې [3] (صفر او یو) کاروي، تر ستونزواله ژبو [4] پورې تحول کړی چې د ریاضي او/یا انګلیسي ژبې له اصطلاحاتو څخه ګټه اخلي. د کمپیوټر ټولې ژبې په لاندې درې لویو برخو ډلبندي کېدای شي:

۱. ماشینواله ژبه (Machine Language)

۲. منځنۍ ژبه (Assembly Language)

---

1 Instructions
2 Machine Languages
3 Binary strings
4 Problem-oriented

۳. عالي ژبه (High-level Language)

اوس به د هرې ژبې طبیعت او تحول وڅېړو.

## ماشینواله ژبه

که څه هم کمپیوټرونه په بېلا بېلو کمپیوټري ژبو باندې د پوهېدو لپاره پروګرام کېدای شي، خو یوازې یوه داسې ژبه شتون لري چې یو کمپیوټر پرې له کوم ژباړونکي پرته پوهېږي. دا ژبه، ماشینواله ژبه یا د کمپیوټر ماشینواله کوډ بلل کېږي. ماشینواله کوډ د یوه کمپیوټر بنیادي ژبه ده او عموما د بایټري متنکرښو (صفر او یو) په شکل لیکل کېږي. د کمپیوټر سرکټ په داسې ډول اوبدل شوی چې سمدلاسه ماشینواله ژبه پېژني او هغو برېښنایي زېګنالونو ته یې اړوي چې د کمپیوټر د ځغلولو[1] لپاره ورته اړتیا ده.

هر لارښود چې په ماشینواله ژبه کې لیکل شوی وي، دوه برخیز فارمټ[2] لري، لکه په لاندې چوکاټ کې چې لیدل کېږي. لومړنۍ برخه یې امر یا عملیه ده، او کمپیوټر ته امر کوي چې کومه کړنه ترسره کړي. هر کمپیوټر د خپلو کړنو لپاره یو عملیاتي کوډ (opcode) [3] لري. د لارښود دوهمه برخه معمول[4] بلل کېږي، او دا

1 Running
2 Two-part format
3 Operation Code
4 Operand

برخه کمپیوټر ته ښیي چې ډېټا او یا نورې لارښوونې چې باید لاسوهنه[1] ورکې وشي، په کوم ځای کې ولټوي او یا یې ذخیره کړي. په دې اساس، هره لارښوونه د مرکزي پروسېس کوونکي د کنټرول واحد ته لارښوونه کوي چې څه وکړي او د هغو ډېټاساحو[2] اوږدوالی او ځای ور په ګوته کوي، چې په عملیه کې ګډون لري. په عادي عملیو کې لوستل[3]، جمع کول، تفریق کول، لیکل[4] او داسې نور راځي.

| OPERAND<br>(ادرس/ځای) | OPCODE<br>(عملیاتي کوډ) |
|---|---|

مور پوهېږو چې ټول کمپیوټرونه د داخلي عملیو د ترسره کولو لپاره له بایزري ارقامو (صفرونو او یوونو) څخه ګټه پورته کوي. په دې ترتیب، د ډېرې کمپیوټرونو ماشینواله ژبې د بایزري شمېرو متنکرښې لري او همدا هغه یوازینۍ ژبه ده چې مرکزي پروسېس کوونکی په مستقیم ډول وربانديې پوهېږي. کله چې د کمپیوټر دننه زېرمه شي، نو هغه سمبولونه چې د ماشینواله ژبې پروګرام جوړوي، له صفرونو او یوونو څخه جوړېږي. د مثال په ډول، ښایي د هغه عادي پروګرام لارښوونه چې یو عدد په پرنټر چاپ کړي، په لاندې ډول وي:

---

1 Manipulation
2 Data Fields
3 read
4 Write

101100111111101001101100

هغه پروګرام چې په حافظه کې دوه عددونه جمع کوي او بیا یې پایله چاپوي، ښایي هغسې حالت ولري چې لاندې لیدل کېږي:

010000000000110011101

011000110000010000000

011000000011100101110

101000111101110010111

000000000000000000000

ښکاره ده چې دا ژبه دومره اسانه نه ده چې زده شي، یو څه له دې امله چې لوستل او پرې پوهېدل ستونزمن دي او یو څه سبب یې دا دی چې دا د اعدادو په هغه سیسټم کې لیکل شوي چې موږ ورسره بلدتیا نه لرو. خو دا به د حیرانتیا ځای وي چې ځینې لومړني پروګرام لیکونکي چې پر ځینو لومړنیو کمپیوټرونو یې کار کاوه، په اصل کې خپل پروګرامونه په بایزي حالت کې لیکل لکه پورته چې تېر شوو.

دا چې انساني پروګرام لیکونکي د لسو له عددي سیسټم سره ډېر بلد دي، نو له هغو څخه زیاترو دا غوره وګڼله چې د کمپیوټر هدایات[1] د لسو په سیسټم کې

1 Instructions

ولیکي. بیا به یې دغه هدایات ورکړیز واحد[1] ته وسپارل چې باینري حالت ته یې واړوي. په حقیقت کې، له ډېرو هڅو پرته، یو کمپیوټر داسې اوبدل کېدای شي چې د صفرونو او یوونو د اوږدو متنکرښو پرځای، له هغه اعشاري سیستم څخه پکې کار واخیستل شي چې زموږ بلدتیا ورسره ډېره ده. په دې بدلون سره، پورتنی پروګرام لاندینۍ بڼه خپلوي:

10001471

14002041

30003456

50773456

00000000

د لارښود کوډونو[2] دا ټولګه، که په باینري سیستم کې وي او که په اعشاري سیستم کې، چې د یوه کمپیوټر مرکزي پروسېس کوونکی په مخامخ ډول ورباندې پوه شي، ماشینواله کوډ یا ماشینواله ژبه نومېږي. په دې اساس، د ماشینواله ژبو پروګرام ضرور نه ده چې د باینري ارقامو (صفرونو او یوونو) په بڼه کوډ شي؛ بلکې د لسو د ارقامو په کارولو هم لیکل کېدای شي که چېرته تر کار لاندې د کمپیوټر سرکټ د دې اجازه وکړي.

---

1 Input Unit
2 Instruction code

## د ماشینواله ژبې ګټې او محدودیتونه

هغه پروګرامونه چې په ماشینواله ژبه لیکل شوي وي، د کمپیوټر له لوري خورا چټک اجرا[1] کېدای شي. د دې اصلي لامل دا دی چې مرکزي پروسېس کوونکی د ماشین په لاربنوونو په مستقیم ډول پوهېږي او د پروګرام ژباړلو ته هېڅ اړتیا نه پېښېږي. خو پر دې سربېره، په ماشینواله ژبه کې د یوه پروګرام لیکل ګڼ زیانونه هم لري چې لاندې پرې رڼا اچول شوې:

۱. **ماشین-اړونده2 ژبه**: دا چې د هر کمپیوټر داخلي ډیزاین له بل هر ډول کمپیوټر سره توپیر لري او د فعالیت لپاره مختلفو بربښنایي زیګنالونو ته اړتیا لري، په همدې ډول ماشینواله ژبه هم له یوه کمپیوټر نه بل ته توپیر کوي. دا د ریاضیکي و منطقي واحد او کنټرول واحد د اصلي جوړښت او ډیزاین او د زېرمې په واحد کې د کلماتو د اوږدوالي[3] د ظرفیت له مخې ټاکل کېږي. په دې اساس، فرض کړئ چې د یوه خاص کمپیوټر د ماشینواله کوډ په برخه کې له مهارت ترلاسه کولو وروسته، یوه کمپنۍ پرېکړه کوي چې خپل کمپیوټر بدل کړي، نو کېدای شي پروګرام لیکونکی مجبور شي چې یوه نوې ماشینواله ژبه زده کړي او هغه ټول شته پروګرامونه یو ځل بیا ولیکي.

---

1 Execute
2 Machine Dependent
3 Word length

**۲. د پروګرام کولو پېچلتیا:** که څه هم یو ماشین په اسانه توګه استفاده تړې کولای شي، خو په ماشینواله ژبه پروګرام لیکل یو پېچلی او دروند کار دی. پروګرام لیکونکی اړ دی چې د ماشین د لارښوونو په ډله کې د اوامرو[1] لپاره لسهاوو کوډونه په یادو زده کړي او یا پرلپسې له مراجعه کارت[2] څخه ګټه واخلي. دغه راز یو پروګرام جوړوونکی مجبور دی چې په زېرمه کې د ډېټا او لارښوونو موقعیت هم په ذهن کې وساتي. د دې ترڅنګ، د ماشینواله ژبې پروګرام لیکونکی باید هغه ماهر شخص وي چې د کمپیوټر د سختغالي جوړښت په اړه هم بشپړ پوهاوی ولري.

**۳. د تېروتنو زیاتوالی:** په ماشینواله ژبه کې د پروګرامونو د لیکلو لپاره، دا چې یو پروګرام لیکونکی باید عملیاتي کوډونه په یاد ولري او ورسره باید په زېرمه کې د ډېټا او لارښوونو موقعیتونه هم په ذهن کې وساتي، نو دا شی نوموړي ته دا ستونزمنوي چې خپل ټول تمرکز د ستونزې پر منطق [3] را ټول کړي. چې په حتمي توګه په پروګرام کې د تېروتنو په رامنځته کېدو پای ته رسېږي. له دې کبله، د ماشینواله کوډ د کارونې پرمهال ډېر په اسانه تېروتنې رامنځته کېږي.

1 Commands
2 Reference Card
3 Logic

**٤. د بدلون راوستلو پېچلتیا:** د ماشینواله ژبې پروګرامونو اصلاح [1] او یا تغیر پکې راوستل سخت وي. د ټپروتنو د موندلو په موخه د ماشینواله لاربنوونو کتنه هماغومره ستونزمنه وي څومره چې یې په پیل کې لیکل سخت وي. په همدې ډول، یو څه وخت وروسته د ماشینواله ژبې په پروګرام کې بدلون راوستل هم ستونزمن وي چې په دې ځای کې به ډېرې پروګرام لیکونکي دا ښه وبولي چې په زاړه پروګرام کې د اړینو بدلونونو پر ځای، له پیل څخه نوی منطق کوډ کړي.

په لنډو ټکو کې ویلای شو چې په ماشینواله ژبه کې د یوه پروګرام لیکل دومره ستونزمن او وخت غواړي چې نن ورځ ترې استفاده له نشت سره برابره ده.

1 Maintain

## منځنۍ ژبه

د پروګرام تیارونې د پروسې په وده کې له لومړنیو پړاوونو څخه یو دا وو چې د ماشینواله ژبې د عددي عملیاتي کوډونو[1] پر ځای حروفي سمبولونه – نیمونیکس – رامنځته شوو. یو نیمونیک (یا د زېرمې مرسته) هر هغه ذهني چل او طریقې ته ویل کېږي چې له مور سره د یو څه په یادولو کې مرسته کوي. نیمونیکونه په بېلابېلو شکلونو او اندازو کې راځي او هر یو یې په خپل ځای کې ګټور دي. د مثال په ډول، یو کمپیوټر کېدای شي د دې لپاره ډیزاین شوی وي څو ۱۱۱۱ (بایزري) او یا ۱۵ (اعشاري) ماشینواله کوډ، د «تفریق» د عملیې په توګه وژباړي[2]، خو د یوه انسان لپاره دا اسانه ده چې دا ټوله پروسه د SUB په ډول زده کړي.

## د عددي عملیاتي کوډونو پر ځای د سمبولونو کارونه

ټول کمپیوټرونه د دې جوګه دي چې عددونه او توري دواړه سمبال کړي. په دې توګه، یوه کمپیوټر ته دا ور ښوولی شو چې د تورو او یا عددونو یو خاص ترکیب وپېژني او یا دا زده کړه ورکولای شو (د یوه پروګرام په مرسته) چې هر ځل چې کمپیوټر د ADD کلمه وینې، نو د هغه پر ځای 14 راولي، هر ځل چې SUB وي، نو ځای یې 15 ته ورکړي، او همداسې ادامه وکړي. په دې ډول، کمپیوټر

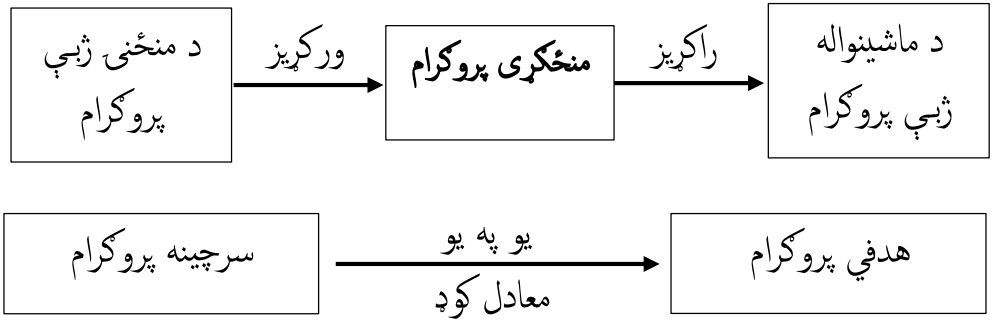
1 Numeric Operation code/opcode
2 Translate

د یوه داسې پروګرام د ژباړلو لپاره روزلی شو چې د کمپیوټر یو پروګرام په ماشینواله ژبه کې د عددونو پر ځای د سمبولونو په کارولو ولیکل شي. بیا موږ کولای شو چې د کمپیوټر لپاره د عددونو پر ځای د سمبولونو په کارولو یو پروګرام ولیکو، او کمپیوټر دې ته چمتو کړو چې پخپله یې ژباړه وکړای شي. دا د پروګرام لیکونکي لپاره اسانتیا رامنځته کوي، ځکه نوموړی کولای شي چې د خپل پروګرام د لیکلو لپاره د عددونو پر ځای له نیمونیکونو، سمبولونو او تورو څخه ګټه پورته کړي. د مثال په ډول، پورتنی پروګرام چې په ماشینواله ژبه کې لیکل شوی وو داسې چې دوه عددونه جمع کوي او بیا یې د جمعي حاصل پرنټ کوي، کولای شو په لاندې ډول ولیکو:

```
CLA  A
ADD  B
STA  C
TYP  C

HLT
```

چې معنا به یې دا وي چې A درواخله په B کې یې جمع کړه او حاصل یې په C کې ذخیره کړه، C (دلته د پروګرام د ټرلو لپاره، ژباړن) ولیکه او پروګرام به وټرل شي. کمپیوټر د یوه ژباړونکي پروګرام په مرسته، د دې پروګرام هره کرښه د ماشینواله ژبې د پروګرام معادل کوډ ته ژباړي.




په دې ځای کې باید ځینې نور اصطلاحات هم زده کړو. هغه ژبه چې د ماشینواله ژبې په پروګرام کې د عددونو پر ځای توري او سمبولونه ځای پر ځای کوي، منځنۍ یا سمبولیکه[1] ژبه بلل کېږي. هغه پروګرام چې په سمبولیکه ژبه کې لیکل شوی وي او د عددونو پر ځای له سمبولونو څخه ګټه اخلي، منځنی کوډ یا سمبولیک پروګرام بلل کېږي. ژباړونکی پروګرام چې منځنی کوډ د کمپیوټر ماشینواله کوډ ته ژباړي، منځګړی پروګرام[2] بلل کېږي. منځګړی پروګرام یو سیستمي پوستغالی دی چې د کمپیوټر د تولیدوونکو له لوري چمتو کېږي. دا پروګرام د سیستمي پروګرام لیکونکو له خوا په ډېر احتیاط لیکل کېږي. دې ته له دې امله منځګړی پروګرام ویل کېږي چې نوموړی د دې ترڅنګ چې منځنی کوډ، ماشینواله کوډ ته ژباړي، د کمپیوټر په اصلي زېرمه کې ماشینواله کوډ هم ځای پر ځای کوي، او د اجرا چاره یې اسانه کوي. یو سمبولیک پروګرام چې د

1 Assembly
2 Assembler

پروګرام لیکونکي له لوړي په منځنۍ ژبه کې لیکل شوی وي، سرچینه پروګرام[1] بلل کېږي. وروسته له دې چې پروګرام د منځګړي په مرسته ماشین کوډ ته واوړي، نو بیا د هدفي پروګرام[2] په نوم یادېږي. څنګه چې په پورتني شکل کې لیدل کېږي، د منځګړي ورکړیز یو سرچینه پروګرام دی چې په منځنۍ ژبه کې لیکل کېږي. دا چې منځګړی پروګرام د منځنۍ ژبې هره لارښوونه خپل معادل د ماشینواله ژبې یوې لارښوونې ته ژباړي، نو له همدې امله د سرچینه پروګرام د منځنیو لارښوونو[3] او د هدفي پروګرام د ماشینواله لارښوونو ترمنځ یو-په-یو اړیکه ده.

تر اوسه پورې باید لوستونکو ته دا روښانه شوې وي چې کله موږ په سمبولیکه ژبه کې یو پروګرام لیکو، نو لومړی منځګړی (پروګرام) ځغلوو څو سمبولیک پروګرام ماشینواله ژبې ته واړوي، او ورپسې د ځواب د ترلاسه کولو لپاره د ماشینواله ژبې پروګرام ځغلوو. ستاسو به پام وي چې د د دې معنا دا ده چې تالیفوونکی[4] ډېر زیات وخت مصرفوي. نوموړی د ځواب د ترلاسه کولو لپاره نه یوازې دا چې اصلي پروګرام وځغلوي، بلکې لومړی باید اصلي سمبولیک پروګرام ماشینواله ژبې ته وژباړي. خو دا چې سمبولیکه پروګراملیکنه د پروګرام لیکونکي

1 Source Program
2 Ojbect Program
3 Assembly instructions
4 Compiler

پربمانه وخت او هڅې سپموي، نو که کمپیوټر ډېر وخت هم تېروي، پروا نه کوي.

که غواړئ وکورئ چې سمبولیکه پروګراملیکنه څه ډول کار کوي، راځئ لومړی د ماشینواله ژبې یو کوچنی پروګرام ولیکو او بیا وکورو چې همدا پروګرام په منځنۍ ژبه کې څنګه لیکل کېږي. د دې کار لپاره، راځئ فرض کړو چې کمپیوټر د عملیاتي کوډونو لپاره چې مخامخ ورته لیکل شوي، له لاندې نیمونیکونو څخه ګټه اخلي.

| نیمونیک | عملیاتي کوډ | معنا |
|---|---|---|
| HLT | 00 | هالټ د پروګرام د ټرلو لپاره په پای کې راځي. |
| CLA | 10 | د رجسټر پاکول یا ور اضافه کول. |
| ADD | 14 | د راجستر له موادو سره جمع کېږي. |
| SUB | 15 | د یوه رجسټر له موادو څخه تفریق کوي. |
| STA | 30 | یو رجسټر ذخیره کوي. |

د آسانیتا په پار، په دې ځای کې موږ یوازې پنځه عملیاتي کوډونه په پام کې نیولي چې زموږ د پروګرام په لیکلو کې به کارېږي. همدې ته ورته، د یوه ځانګړي پروګرام لپاره په سلهاوو نور عملیاتي کوډونه هم کېدای شي.

هغه پروګرام چې موږ یې لیکو ډېر ساده دی: دوه عدده جمع کول او حاصل یې زېرمه کول. د دې لپاره به د ماشینواله ژبې عادي پروګرام په لاندې ډول وي:

| د زېرمې موقعیت | محتوا | | کتنې |
|---|---|---|---|
| | عملیاتي کوډ | ادرس | |
| 0000 | 10 | 1000 | A رجسټر پاک او لومړنی عدد ور زیات کړه. |
| 0001 | 14 | 1001 | د A رجسټر محتوا ته دوهم عدد ور جمع کړه. |
| 0002 | 30 | 1002 | له A رجسټر څخه ځواب زېرمه کړه. |
| 0003 | 00 | | پروګرام وتړه. |
| . | | | |
| . | | | |
| . | | | |

| | | | |
|---|---|---|---|
| 1000 | | | د لومړني نمبر لپاره نیول شوی. |
| 1001 | | | د دوهم نمبر لپاره نیول شوی. |
| 1002 | | | د ځواب لپاره نیول شوی. |

په دې ځای کې دا فرض شوي چې کمپیوټر د بائنري عددونو پر ځای د اعشاري اعدادو د سمبالښت وړتیا هم لري. هغه دوه عددونه چې باید جمع شي، د زېرمې په 1000 او 1001 موقعیتونو کې زېرمه کېږي، او د دواړو عددونو له جمع څخه چې کوم حاصل لاسته راځي هغه به په 1002 موقعیت کې ذخیره کېږي. په 1000 موقعیت کې لومړنۍ لارښوونه د A رجسټر پاکوي (صفر کوي یې) او د 1000 د موقعیت محتوا (لومړنی عدد) پکې ځای په ځای کوي. دوهمه ماشینواله لارښوونه په 1000 موقعیت کې، د 1001 موقعیت محتوا (دوهم عدد) د A رجسټر له محتوا (لومړنی عدد) سره جمع کوي او حاصل یې په A رجسټر کې ذخیره کوي. په 0002 موقعیت کې دریمه لارښوونه له A رجسټر څخه ځواب را اخلي او د زېرمې په 1002 موقعیت کې یې ذخیره کوي. په پای کې، څلورمه لارښوونه چې 1003 موقعیت کې راغلې، د پروګرام اجرا ته د پای ټکی وردي.

اوس به وګورو چې همدا پروګرام په منځنۍ ژبه کې څنګه لیکلای شو. ډېر په اسانه کولای شو چې په پورتنیو لارښوونو کې عملیاتي کوډ د هغو په معادلو نیمونیکونو بدل کړو او پورتنی پروګرام په لاندې ډول ولیکو؛ او نور کار کمپیوټر ته وسپارو چې سمبال یې کړي. منځګړی پروګرام به CLA په 10، ADD په

14، STA په 30 او HLT په 00 ترجمه[1] کړي، او په دې ډول به د ماشینواله ژبې پروګرام تولید شي.

| د زېرمې موقعیت | محتوا | |
|---|---|---|
| | نیمونیک | ادرس |
| 0000 | CLA | 1000 |
| 0001 | ADD | 0001 |
| 1002 | STA | 0002 |
| | HLT | 0003 |

## د ادرسونو پر ځای د سمبولونو کارول

خو اړتیا نشته چې په همدې ځای تم شو؛ اخر، د یوه کمپیوټر لپاره د عملیاتي کوډونو په خاطر د یو څو عددونو په یادو زده کول دومره لوی کار نه دی او د عملیاتي کوډونو پر ځای د سمبولونو کارول پر کمپیوټر دومره بوج نه دی. خو، مور کولای شو چې په کمپیوټر باندې تر دې هم د کار برخه ور زیاته کړو.

[1] Interpret

د ماشینواله ژبې په پروګراملیکنه کې له لویو ستونزو څخه یوه هم د ادرسونو په ذهن کې ساتل دي. هر ځل چې موږ یو لوی پروګرام لیکو، نو اړتیا ده چې په خپل څنګ کې یوه پاڼه راسره ولرو چې په هغه کې یو لیست ولیکو چې کوم عددونه باید چېرته ذخیره شي. هر ځل چې موږ یو عدد غواړو، نو باید په هغه لیست کې یې ادرس وګورو. دا کار وخت غواړي او که چېرته احتیاط ونه شي، تېروتنه تر ې رامنځته کېږي. د دې پر ځای ولې کمپیوټر ته دا واک نه ورکوو چې زموږ پر ځای هم کار وکړي؟ موږ د منځنۍ ژبې د پروګرام د یوې برخې په توګه، په پروګرام کې یوه داسې برخه زیاتولای شو چې بل هېڅ کار هم ونه کړي، پرته له دې چې د عددونو لپاره د ادرسونو یو لیست له ځان سره وساتي.

په پورتني پروګرام کې د دوو عددونو د جمع کولو لپاره، ښایي د خپل سمبولیک پروګرام په مرسته کمپیوټر ته داسې امر وکړو:

«له دې وروسته 1000 ادرس به په FRST یادېږي، 1001 ادرس به په SCND یادېږي، او 1002 به په ANSR یادېږي.»

د دې په غبرګون کې منځګړی پروګرام د کمپیوټر د زېرمې په یوه برخه کې یو جدول جوړوي چې ښایي لاندې حالت ولري:

FRST = 1000

SCND = 1001

ANSR = 1002

له دې وروسته به یوازې په خپلو نومونه ور غږ کوو. منځګړی پروګرام به په جدول کې هغه نوم ګوري او دقیق ادرس به راته چمتو کوي او په دې ډول به مو له ډېر کار څخه ځان ساتلی وي. دا په دې معنا چې مونږ خپل پروګرام په لاندې ډول لیکلی شو:

| د زېرمې موقعیت | محتوا | |
|---|---|---|
| | نیمونیک | ادرس |
| 0000 | CLA | FRST |
| 0001 | ADD | SCND |
| 1002 | STA | ANSR |
| 1003 | HLT | |

اوس منځګړی پروګرام د هر عدد لپاره ادرس لټوي، هغه نیمونیک عملیاتي کوډ، عددي عملیاتي کوډ ته ترجمه کوي او په پای کې د ماشینواله ژبې یو پروګرام راته چمتو کوي. د مثال په ډول، په لومړنۍ لاربنوونه کې CLA FRST، منځګړی پروګرام CLA د 10 عملیاتي کوډ ته ژباړي او بیا په جدول کې د FRST ادرس لټوي، منځګړی دا ادرس د 1000 په ډول مومي، او په دې ډول د ماشینواله ژبې لومړنۍ لاربنوونه د 1000 10 په بڼه ځای په ځای کوي. کټ مټ په همدې ډول،

د ADD SCND لارښوونه 1001 14 ته ژباړي، او همداسې دوام کوي، او پراو په پراو د ماشینواله ژبې پروګرام چمتو کوي.

خو زمور د برخې یو بل کار هم پاتې دی چې کولای شو هغه هم کمپیوټر ته ور له غاړې کړو. مور اړتیا نه لرو چې کمپیوتر ته ووایو چې یو عدد چېرته ځای په ځای کړي او یوه لارښوونه چېرته ځای کړي، لکه تر دې دمه مو چې همداسې کول. بلکې د دې پر ځای ورته ویلی شو چې «FRST په 1000 کې ځای په ځای کړه، SCND په 1001 کې ځای کړه، او ANSR په 1002 کې.» مور باید کمپیوټر ته یوازې دومره ووایو چې د CLA FRST لارښوونه 0000 ته ځي، ADD SCND په 0001 پورې اړوند ده او همداسې نور. دا ځل مور یوازې کمپیوټر ته دومره وایو چې د 0000 په موقعیت پروګرام پیل کړي. په دې ترتیب، سمبولیک پروګرام به په لاندې ډول یوه بڼه ولري:

```
START PROGRAM AT 0000 AND START DATA AT 1000
SET ASIDE AN ADDRESS FOR SCND
SET ASIDE AND ADDRESS FOR ANSR
CLA FRST
ADD SCND
STA ANSR
HLT
```

## د پروګرام پښتو ژباړه:

پروګرام په 0000 پیل کړه او ډېټا په 1000 پیل کړه.
د دوهم عدد لپاره یو ادرس ټاکل کړه.
د ځواب لپاره یو ادرس ټاکل کړه.
رجسټر پاک او لومړنی عدد ور زیات کړه.
دوهم عدد ور زیات کړه.
په A رجسټر کې ځواب ذخیره کړه.
پروګرام وتړه.

ورپسې په دې سمولیک پروګرام باندې د کار کولو لپاره منځګړی پروګرام پیلوو. موږ وینو چې لومړني څلور پراوونه د اصلي پروګرام چې د دوو عددونو جمع کول دي، حقیقي برخه نه دي، بلکې یوازې هغه لارښوونې دي چې منځګړي پروګرام ته وایي چې څه وکړي، او دې لارښوونو ته جعلي لارښوونې[1] ویل کېږي. د pseudo کلمه له یوناني ژبې څخه راغلې چې د غلط، جعلي او فرض شوي په معنا ده، او دا په ډاګه کوي چې: دا هغه لارښوونې دي چې له اصلي پروګرام سره هېڅ اړیکه نه لري، بلکې یوازې منځګړي ته معلومات چمتو کوي او ورته وایي چې موږ په څه ډول غواړو چې زموږ پروګرام اجرا شي.

[1] Pseudo Instructions

د دې لپاره چې وګورو منځګړی دا سمبولیک پروګرام څنګه ماشینواله ژبې ته ژباړي، راځئ چې پراو په پراو یې نندارې ته کښنو.

لومړنی پراو منځګړي ته وایي چې پروګرام باید د 1000 په ادرس پیل شي او هره راتلونکې لاربنوونه باید په همدې ادرس کې وي، او بله دا چې ډېټا (چې په دې مثال کې FRST او SCND عددونه او ورسره ANSR دي)، هم باید د زېرمې په 1000 موقعیت کې پیل شي.

بل پراو منځګړي ته وایي چې د FRST لپاره یو ادرس ټاکل کړي. په دې ډول منځګړی پروګرام له هغې ډلې څخه چې د ډېټا لپاره ټاکل شوی وي، لومړنی تش ادرس ور اوچتوي، چې 1000 دی او له دې وروسته به یې په FRST یادوو. پام مو وي چې له دې امله چې موږ هغه ادرس ته د FRST په سمبول اشاره کوو، نو موږ په دې اړه هېڅ اندېښنه نه کوو چې هغه ادرس په دقیق ډول څه شی دی، هغه که 1000 وي یا 1001 یا 4253 وي، هېڅ توپیر نه کوي، ځکه چې منځګړي په خپله حافظه کې زېرمه کړی دی.

بل پراو منځګړي ته وایي چې د SCND لپاره یو ادرس ټاکل کړي. له 1000 نه ووروسته بل تش ادرس 1001 دی (ځکه چې 1000 دمخه لا FRST نیولی دی)، نو منځګړی هر هغه ځل چې د SCND سمبول کاروو، د 1001 ادرس برابروي.

په همدې ډول، راتلونکی پړاو منځګړي ته وایي چې د ANSR لپاره یو ادرس پیل کړي، چې دا ځل 1002 دی. دا وروستۍ جعلي لارښوونه ده، او راتلونکې لارښوونه په خپله اصلي پروګرام پیلوي.

ورپسې لارښوونه CLA FRST ده، چې کمپیوټر یې 1000 10 ته ژباړي، داسې چې CLA په 10 ترجمه کوي او په خپل جدول کې د FRST ادرس لټوي چې 1000 دی. په ورته ډول، منځګړی د ADD SCND په 1001 14 ترجمه کوي او STA ANSR په 1002 30 ترجمه کوي. په پای کې منځګړی HLT په 00 ترجمه کوي، او په دې ډول د منځنۍ ژبې له پروګرام سره سم په بشپړ ډول د ماشینواله ژبې پروګرام چمتو کوي.

## پر ماشینواله ژبه د منځنۍ ژبې ګټې

منځنۍ ژبه پر ماشینواله ژبه لاندینۍ ګټې لري:

**۱. د کارونې او پوهېدنې لپاره اسانه:** منځنۍ ژبې د کارولو او وړاندې پوهېدلو په اساس اسانه دي، په دې چې د عددي عملیاتي کوډونو پر ځای له نیمونیکونو څخه ګټه اخلي او د ډېټا لپاره مناسب نومونه کاروي. له نیمونیکونو څخه ګټه اخیستل په دې معنا ده چې نظرونو[1] ته معمولا اړتیا نه وي، بلکې پخپله پروګرام د پوهاوي وړ وي. د دې ترڅنګ، سمبولیکه پروګراملیکنه د پروګرام لیکونکي

[1] Comments

پربمانه وخت او زحمت هم سپموي، په دې چې د ماشینواله ژبې د پروګرامونو په پرتله په اسانه لیکل کېږي.

**۲. د تېروتنو موندلو او سمولو اسانتیا:** کله چې په منځنۍ ژبه کې پروګرام لیکل کېږي، د تېروتنه شمېر کم وي، او کومې تېروتنې چې شوې وي د هغو موندل او سمول د نیمکونیکونه او ساحو لپاره د سمبولیکو نومونو له کبله اسانه وي. سربېره پر دې، منځګړي پروګرامونه داسې ډیزاین شوي وي چې په خپلکاره[1] ډول تېروتنې په ګوته کوي. که چېرته موږ داسې نیمونیک یا نوم وکاروو چې هېڅکله نه وي تعریف شوی، نو منځګړی پروګرام به د تېروتنې په ډول د دا ډول نوم شتون ته نغوته وکړي. د مثال په ډول، فرض کړئ چې په سمبولیک پروګرام کې یوه لاربښوونه ADD AREA لوستل کېږي، او له موږ نه دا هېر شوي وي چې AREA تعریف کړو، نو منځګړی پروګرام به په خپل جدول کې په AREA پسې سترګې وغړوي؛ نو که یې ونه مومي، تېروتنه به را برسېره کړي.

**۳. د بدلون راوستو اسانتیا:** د پروګرام لیکونکو لپاره د ماشینواله ژبو د پروګرامونو په پرتله د منځنۍ ژبې په پروګرامونو کې بدلون او اصلاح راوستل اسانه دي. د دې اصلي لامل دا دی چې په دې ژبه پوهېدل اسانه دي، په دې اساس د لاربښوونو بدلون، سمون او تېروتنو په نسبه کول هم اسانه وي. تر دې ورهاخوا، له پروګرام څخه د ځانګړو لاربښوونو د درج او ایستلو لپاره د دې اړتیا هم نه وي

[1] Automatic

چې د پروګرام د هغه برخې په لارښوونو پورې اړوند د ادرس برخه تغیر کړای شي. دې کار ته یوازې په ماشینواله ژبه کې اړتیا پېښېږي.

**٤. د ادرسونو په تړاو د اندېښنو نشتوالی:** د منځنۍ ژبې یو له لویو ګټو څخه دا هم ده چې د ډېټا او لارښوونو په ادرسونو پورې تړلې اندېښنې له منځه وړي. دا تر هغه ډېر مهم دی څومره چې په لومړني نظر ښکاري. فرض کړئ چې موږ یو اوږد ماشینواله پروګرام لیکلی چې څو پراوونه لري او د پروګرام دننه یې بېرته خپل ځان ته څو ځله مراجعه هم کړې، لکه څرخ[1] کارول، یا د ادرسونو بدلون، او داسې نور. خو په پای کې مو ناڅاپه پام شي چې د پروګرام په منځ کې یوه لارښوونه را څخه پاتې شوې. که چېرته موږ هغه لارښوونه درج کوو، نو باید ټولې لارښوونې مو په یادو زده وي، او ټول پروګرام ټکی په ټکی را وګورو تر څو هغه ځای پیدا کړو چې له هغه ځایه نورو پراوونو ته مراجعه شوې وي. دا یو سخت کار دی. خو که چېرته موږ همدا پروګرام په سمبولیکه ژبه ولیکو، نو موږ یوازې یوه اضافي لارښوونه ور زیاتوو، او منځګړی پروګرام په خپلکاره ډول هغه ټول پراوونه سرته ورسوي.

**٥. په اسانه د ادرس بدلول:** فرض کړئ چې د منځنۍ ژبې پروګرام د 1000 په ادرس پیلېږي او موږ ناڅاپه پوهېږو چې موږ یو بل پروګرام هم باید له دغه پروګرام سره استعمال کړو او هغه بل پروګرام هم د 1000 په موقعیت پیل کېږي.

---

[1] Loop

د دې لپاره چې دواړه پروګرامونه ولیکل شي، نو ښکاره ده چې یو باید له هغه ځایه لرې کړای شي. په ماشینواله ژبه کې دا کار ډېر پیچلی دی. خو په منځنۍ ژبه کې یوازې باید لومړنۍ لارښوونه بدله کړو. د مثال په ډول لاندې لومړۍ لارښوونه:

START PROGRAM AT 1000 START DATA AT 2000

په دې لارښوونه بدلوو:

START PROGRAM AT 3000 AND START DATA AT 4000

او د منځکړي په مرسته سمبولیک پروګرام یو ځل بیا ځغلوو. د دې لارښوونو معادل ماشینواله پروګرام به خپل پروګرام د 1000 په ځای له 3000 څخه پیل کړي، او په دې ډول به له بل پروګرام سره هېڅ ټکر نه خوري. په بل عبارت، سمبولیکه ژبه په ډېرې اسانۍ پروګرامونه د حافظې له یوې برخې څخه بلې هغې ته لېږدولی شي؛ په دې اساس موږ ویلای شو چې د منځنۍ ژبې پروګرامونه په اسانۍ د انتقال وړ دي ځکه چې د دوی موقعیت یوازې د لومړنۍ لارښوونې په بدلون، په اسانه بدلېږي. دا کار په ماشینواله پروګراملیکنه کې دومره په اسانه سرته نه رسېږي.

**٦. د ماشینواله ژبې اغېزمنتیا:** د هغو ګټو ترڅنګ چې پورته یادې شوې، د منځنۍ ژبې پروګرام د خپل معادل ماشیني کوډ له اغېزمنتیا او ګټو څخه هم برخمن دی، ځکه د یوې منځنۍ ژبې د لارښوونو او د هغه د معادل ماشینواله ژبې د

لارښوونو ترمنځ مطابقت، د یو په یو اړیکې په اساس وي. له جعلي لارښوونو پرته چې یوازې د منځګړي پروګرام ځانګړې لارښوونې دي، د منځنۍ ژبې بله هره لارښوونه د ماشینواله ژبې یوازې یوې لارښوونې ته ترجمه کېږي. د ماشینواله ژبې د هرې لارښوونې لپاره یوازې یوه معادله سمبولیکه لارښوونه شتون لري او هره سمبولیکه لارښوونه (له سوډو لارښوونو پرته)، یوازې یو ماشینواله معادل لري. په بل عبارت، یو سمبولیک پروګرام یوازې هغومره حجم لري، څومره یې چې د هغه معادل ماشینواله ژبه لري. په دې اساس که چېرې هغه وخت چې منځګړی پروګرام د ژباړې لپاره ورته اړتیا لري له منځه وایستل شي، نو د ماشینواله پروګرام او منځني پروګرام د اجرا اصلي وخت به سره یو شان وي. پر دې ټکي باندې د ټینګار لامل دا دی چې داسې ژبې (میکرو ژبې) هم شته چې ښایي یوازې یوه لارښوونه یې د ماشینواله ژبې د لارښوونو په یوې بشپړې لړۍ وژباړل شي. منځنۍ ژبه، د خپل اصلي فطرت له مخې د دې ژبو له ډلې څخه نه ده، او په دې ژبه کې د سمبولیکې او ماشینواله ژبې ترمنځ یوازې یو په یو اړیکه ده.

## د منځنۍ ژبې محدودیتونه

د ماشینواله ژبې لاندینې زیانونه د منځنۍ ژبې په کارولو نه شي هوارېدلای:

۱. **ماشین-اړونده:** دا چې د سمبولیکې ژبې هره لارښوونه د ماشینواله ژبې یوازې یوې لارښوونې ته ژباړل کېږي، په دې سبب منځنۍ ژبې هم ماشین-

اړونده دي. دا په دې معنا چې نوموړې ژبه د یوه ځانګړي جوړښت او ماډل د پروسیس کوونکي واحد لپاره ډیزاین شوې وي. که چېرته د کمپیوټر د بدلولو پرېکړه وشي، نو معمولا د یوې نوې ژبې زده کړې او د ټولو شته پروګرامونو اړولو ته اړتیا وي چې دا اقدام د لوړې بیې غوښتونکی دی.

**۲. هارډوېر پوهنې ته اړتیا:** دا چې منځنۍ ژبې ماشین-اړونده دي، نو پروګرام لیکونکی اړ دی چې د پروګرام د لیکلو پر مهال، د یوه ځانګړي ماشین له ځانګړنو او شرایطو څخه خبر وي. په منځنۍ ژبه کې پروګرام لیکونکی باید پوه وي چې د هغه ماشین څنګه کار کوي او د خپل کمپیوټر د منطقي جوړښت په هکله پوره پوهاوی ولري تر څو وکولای شي چې یو ښه پروګرام ولیکي.

**۳. د ماشینواله کچې کوډ[1] لیکل:** په منځنۍ ژبه کې هم لارښوونې د ماشینواله کچې په کوډ لیکل کېږي – په دې معنا چې د ماشینواله کوډ یوه لارښوونه خپل ځای د منځنۍ ژبې یوې لارښوونې ته پرېږدي.

ماشینواله او منځنۍ ژبې چې ماشین-اړونده ژبې دي، د ټیټو ژبو[2] په نوم هم یادېږي.

## د مېکرو لارښوونو منځنۍ ژبه

[1] Machine-level code
[2] Low-level languages

په عموم کې منځنۍ ژبې د خپل طبیعت له مخې «یو په یو» ژبو په توګه پېژندل کېږي، یعنې، د منځنۍ ژبې هره لارښوونه د ماشینواله ژبې یوازې یوه لارښوونه پنځوي. که څه هم، ډېر کله د ماشینواله یا منځنۍ ژبې د لارښوونو یوه ځانګړې ټولګه څو څو ځله باید وکارول شي. د مثال په ډول، کېدای شي په یوه کمپیوټر باندې د یوه نمبر د پرنټ کولو لپاره درې یو په بل پسې لارښوونو ته اړتیا وي. دغه درې لارښوونې چې تل په یوه ترتیب کې وي، کېدای شي په هماغه یوه پروګرام کې بیا بیا ځلې وکارول شي. هر ځل چې یو پروګرام لیکونکی غواړي چې یو عدد پرنټ کړي، نو د دې پر ځای چې د درې لارښوونو یوه ټولګه ولیکي، ښایي یو منځګړی (پروګرام) په داسې ډول ډیزاین کړای شي چې دغه درې واړه لارښوونې سمبال کړي. د مثال په ډول، هر ځل چې پروګرام لیکونکی د PRINT لارښوونه لیکي، نو منځګړی د یوې لارښوونې پر ځای نوموړې لارښونه د ماشینواله ژبې په درې لارښوونو ترجمه کوي، او په دې ډول به د پرنټ لپاره د اړتیا وړ لارښوونو پوره ټولګه چمتو کړي.

هره لارښوونه، لکه PRINT، چې د ماشینواله ژبې څو لارښوونو ته ژباړل کېږي، مېکرو لارښوونه بلل کېږي. یو ځانګړی منځګړی پروګرام کولای شي چې څو مېکرو لارښوونو ته اجازه ورکړي. په دې اساس، د کوډ کولو د پروسې د کړندي کولو لپاره داسې منځګړي پروګرامونه جوړ شوو چې د منځنۍ ژبو د پروګرام د مېکرو لارښوونو لپاره یې د ماشینواله ژبې د لارښوونو یو بدیل مقدار تولیدولای شوی.

مېکرو لارښوونې پر کمپیوټر باندې د کار بوج زیاتوي، په دې چې د ژباړې پروسه یوازې د هرې کلمې یوه عدد ته له بدلولو څخه زیاتېږي. منځګړی باید د دې وړتیا ولري چې پاتې پراوونه هم چمتو کړي، خو دا د پروګرام لیکونکي له لوی کاري بوج څخه د ژغورلو په معنا ده. په دې ډول پروګرام لیکونکی د هرې عملیې لپاره د لارښوونو د لیکلو له دندې څخه بې غمه وي. او دا کار د نوموړي د پروګرام حجم راکموي، د تېروتنو شمېر یې ټیټوي او پروګراملیکنه ساده کوي.

د مېکرو لارښوونو ځواک و وړتیا د کمپیوټري ژبو د تحول په لومړنیو وختونو کې رامنځته شوه. په ار کې د مېکرو لارښوونو له یوې لارښوونې څخه د ځنې شمېر ماشینواله لارښوونو جوړولو همدا مفهوم دی چې له مخې یې د نن ورځې ماشین-خپلواکه عالي ژبې ډیزاین شوې دي.

# عالي ژبې

مور دمخه ولیدل چې په ماشینواله یا منځنۍ ژبه کې د پروګرامونو لیکل د کمپیوټر د داخلي جوړښت په اړه ژورې پوهې ته اړتیا لري. حال دا چې په دې ژبو کې د پروګرام لیکلو لپاره پروګرام لیکونکی باید د کمپیوټر ټول عملیاتي کوډونه (عددي یا نیمونیک) په یادو زده کړي او په پوره تفصیل په دې پوه وي چې هر کوډ څه کار کوي او څنګه د کمپیوټر بېلا بېل رجسټرونه اغیزمنوي. که څه هم، مور پورته ولوستل چې د دې لپاره چې یو ښه کمپیوټري پروګرام ولیکل شي، نو پروګرام لیکونکی باید د هغه پروګرام پر منطق تمرکز وکړي، نه دا چې د کمپیوټر د داخلي جوړښت په ټراو اندښنه وکړي. د دې لپاره چې پروګرام لیکونکي هوسا کړای شي تر څو د کمپیوټر په داخلي جوړښت باندې له پوهېدو پرته کمپیوټر استفاده کړي، عالي ژبې رامنځته شوې.

عالي ژبو، د دې پر ځای چې ماشین-اړونده اوسي، ډېره ډډه د ستونزې په حل کولو اچولې. دغه ژبې پروګرام لیکونکی د دې جوګه کړځوي چې د انګلیسي کلماتو او د عادي ریاضیکي سمبولونو په کارولو خپلې لارښوونې ولیکي. په دې ډول د نوموړي لپاره دا اسانه کېږي چې د پروګراملیکنې په تفصیلاتو کې د سرګردانه کېدو پر ځای د ستونزې منطق ته متوجه شي. ښکاره ده چې هغه دوه-برخیز فارمټ چې په ماشینواله یا منځنۍ ژبه کې د لارښوونو لیکلو لپاره ورته اړتیا وه، په عالي ژبو کې د لارښوونو لیکلو لپاره ضرور نه ده. د مثال په ډول، راځئ د دوو عددونو د (FRST & SCND) جمع کولو او حاصل یې په ANSR کې د

ذخیره کولو هماغه ستونزه فرض کړو. موږ دمخه ولیدل چې د دې کار د سرته رسولو لپاره دري د ټیټې کچې (منځنۍ/ماشینواله) لارښوونو ته اړتیا ده. خو، که چېرته د دې کار د ترسره کولو لپاره له عالي ژبو څخه ګټه واخلو، د مثال په ډول فورټران[1] ته ور ګرځو، نو یوازې یوې لارښوونې لیکلو ته اړتیا ده:

$$ANSR = FRST + SCND$$

ښکاره ده چې د پورتنۍ لارښوونې لیکل او وړاندې پوهېدل ډېر اسانه دي، په دې چې دا د الجبر له هغې معادلې سره شباهت لري چې د دوو عددونو د جمع کولو لپاره استفاده کېږي: a = b + c

عالي ژبې په اساسي ډول هغه ژبې دي چې د نیمونیکو کوډونو پر ځای د انګلیسي له کلماتو او/یا ریاضیکي نښو څخه ګټه اخلي. په بل عبارت، عالي ژبه یوه سمبولیکه ژبه ده چې هېڅ مېکرو-لارښوونې نه لري. هره لارښوونه چې په یوه عالي ژبه کې د پروګرام لیکونکي له خوا لیکل کېږي، د ماشینواله ژبې څو لارښوونو ته ژباړل کېږي. دا په اصل کې «یو په څو» ژباړه ده او د منځنۍ ژبې په څېر «یو په یو» اړیکه نه لري. د همدې علت له مخې دې ژبو ته عالي ژبې ویل کېږي.

[1] Fortran

عالي ژبې د ستونزواله ژبو[1] په نوم هم یادېږي، په دې چې مېکرو لارښوونې یوازې د دې لپاره تر کار لاندې راځي څو د یو خاص ډول ستونزو د حل کولو لپاره کټورې پرېوځي. په دې اساس هره ژبه د ستونزو د یوې ځانګړې ټولګې د حل کولو لپاره کټوره پرېوځي، او کېدای شي د ستونزو د نورو ډولونو لپاره په کامله توګه بې کټې اوسي. د مثال په ډول، که چېرته یوه عالي ژبه د کاروبار-ډوله کاریالونو لپاره چې د لوړ حجم ورکړیز، نسبتاً لږ پروسېس او د لوړ حجم راکړیز غواړي، نو دا ډول ژبه، یوه کاروبار-واله ژبه ده. بل خوا، هغه ژبې چې د پېچلو محاسبو لپاره غوره وي، خو د لویو ډېټا فایلونو د سمبالښت توان په ځان کې ونه لري، ریاضي-واله ژبه ده. په دې اساس، یوه ستونزواله ژبه داسې ډیزاین کېږي چې لارښوونې یې د هغې ستونزې د ژبې په توګه ولیکل شي. د مثال په ډول، یو ساینسپوه د یوې ساینسي ژبې په کارولو له ساینټیفیکو فورمولونو څخه کټه اخیستلای شي، او یو سوداګر د یوې کاروباري ژبې په مرسته له کاروباري اصطلاحاتو څخه کټه اخیستلای شي. په دې اساس، عالي ژبې په عموم کې په اسانه زده او لیکل کېږي.

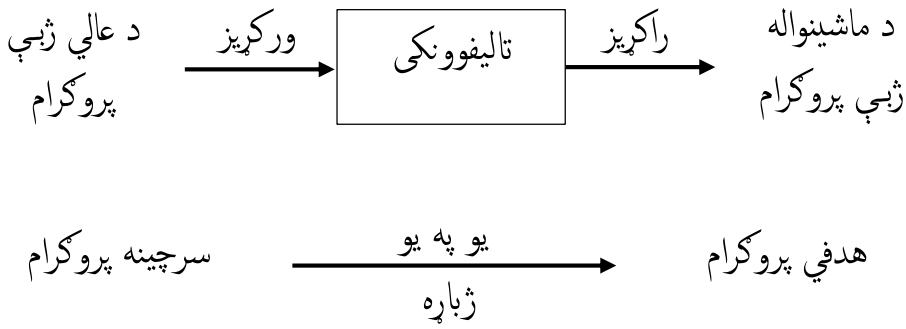
[1] Problem-oriented

## تالیفوونکي[1]

دا چې کمپیوټر یوازې د ماشین د کچې په لاربنوونو باندې د پوهېدو وړتیا لري، نو د دې لپاره چې یو پروګرام اجرا کړای شي، اړتیا ده چې په عالي ژبه کې لیکل شوې لاربنوونې لومړی ماشینواله لاربنوونو ته واوړي. مور ولیدل چې منځنۍ ژبې د اوړون پروسېس[2] د سرته رسولو لپاره له منځګړي پروګرام څخه ګټه اخلي. په یوه عالي ژبه کې، همدا چاره د تالیفوونکي له لوري پرمخ وړل کېږي. په دې اساس، تالیفوونکی(چې له یوه منځګړي پروګرام څخه ډېر پېچلی وي) د ژباړې هغه پروګرام دی، چې د عالي ژبې لاربنوونې ماشینواله ژبې ته ترجمه کوي. دې پروګرام ته په دې تالیفوونکی ویل کېږي چې د عالي ژبې د پروګرام د یوې لاربنوونې لپاره د ماشینواله ژبې د لاربنوونو یوه ټولګه تالیف کوي. هغه پروګرام چې پروګرام لیکونکي په عالي ژبه کې لیکلی وي، سرچینه پروګرام بلل کېږي. کله چې دا سرچینه پروګرام د تالیفوونکي له خوا ماشینواله ژبې ته واوړي، له هغه وروسته هدفي پروګرام ورته ویل کېږي. څنګه چې په لاندې شکل کې لیدل کېږي، د تالیفوونکي (پروګرام) ورکړیز هغه سرچینه پروګرام دی چې په عالي ژبه کې لیکل شوی وي او راکړیز یې هغه هدفي پروګرام دی چې د ماشینواله ژبو له لاربنوونو څخه جوړ وي. پام مو وي چې سرچینه پروګرام او هدفي پروګرام دواړه یو دي، مګر د تحول په ېلابېلو پړاوونو کې قرار لري.

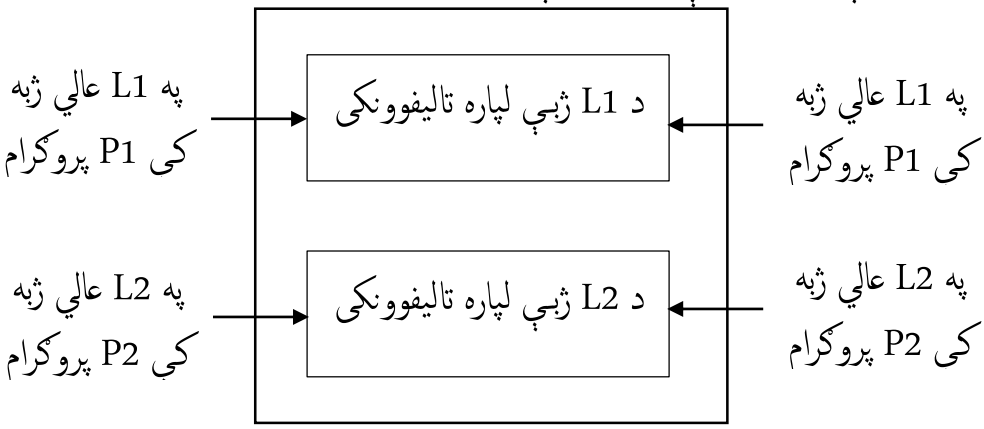
---

[1] Compilers
[2] Conversion process

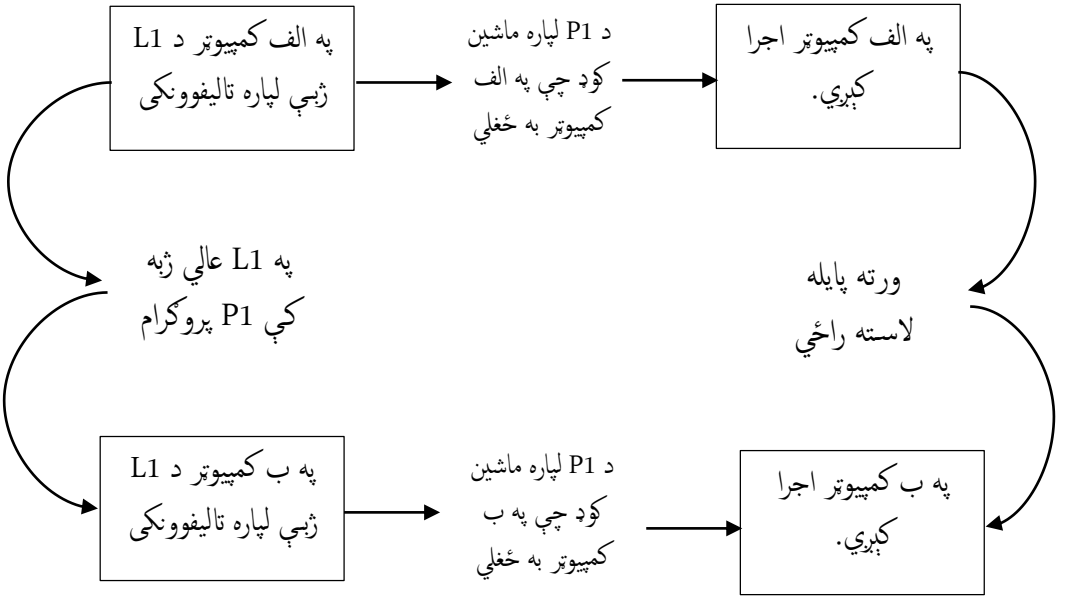



پورتنی انځور د تالیفوونکي د ژباړې پروسه تشریح کوي.

یو تالیفوونکی یوازې هغه سرچینه پروګرامونه تالیف کولای شي چې په هغه ژبه کې لیکل شوي وي چې هغه تالیفوونکی ورته ډیزاین شوی وي. د مثال په ډول، د فورټران تالیفوونکی یوازې د هغو سرچینه پروګرامونو د تالیف وړتیا لري چې په فورټران کې لیکل شوی وي، او له همدې امله هر ماشین د هرې عالي ژبې لپاره یوه بېل تالیفوونکي ته اړتیا لري چې هغه ماشین یې ملاتړ کوي. دا موضوع په لاندې انځورونو کې سپړل شوې.



انځور: د هرې عالي ژبې لپاره چې یو کمپیوټر یې ملاتړ کولای شي ، د یوه بېل تالیفوونکو شرط تشریح کوي.




د عالي ژبو د ماشین-خپلواکۍ ځانګړنه تشریح کوي. په مختلفو کمپیوټرونو باندې د یوې ژبې د ځغلولو لپاره مختلف تالیفوونکي پکار دي.

د دې ترڅنګ، دا چې د یوه ماشین لپاره لیکل شوی هدفي پروګرام له هغه هدفي پروګرام سره یو شان نه وي چې د بل ماشین لپاره لیکل شوی وي، په دې اساس ارټیا ده چې هر ماشین باید د یوې ځانګړې ژبې لکه L1، لپاره خپل تالیفوونکی ولري. پورتنی انځور دا تشریح کوي چې د مختلفو تالیفوونکو په مټ

په څه ډول د ماشین-خپلواکې ځانګړنه ترلاسه کېږي تر څو یوه عالي ژبه د مختلفو کمپیوټرونو لپاره ماشینواله ژبو ته ترجمه کړي.

ژباړونکي هغه ستر پروګرامونه دي چې په تلپاتې ډول په مرستندویه زېرمه کې پراته وي. کله چې د یوه پروګرام ژباړه سرته رسېږي، نو هغه لاربنوونې د کمپیوټر اصلي زېرمې ته کاپي کېږي. تالیفوونکی چې پخپله یو پروګرام دی، په مرکزي پروسېس کوونکي واحد کې اجرا کېږي. کله چې یو پروګرام ژباړل کېږي، نو تالیفوونکی د سرچینه پروګرام هر عبارت تجزیه کوي او د ماشینواله لاربنوونو یوه لړۍ تولیدوي، چې کله هغه اجرا شي، نو هماغه محاسبه سرته رسېږي چې په هغو لاربنوونو کې تعریف شوې وي. کوم وخت چې تالیفوونکی هر عبارت تجزیه کوي، له هماغه سره سم تالیفوونکی په پروګرام کې شته تېروتنې هم را برسېره کوي. دغه تېروتنو ته تشخیصي تېروتنې (diagnostic erros) ویل کېږي. تالیفوونکی کولای شي چې په یوه سرچینه پروګرام کې لاندې ډول تېروتنې تشخیص کړي:

الف) ناقانونه توري[1]

ب) د تورو ناقانونه ترکیب[2]

[1] Illegal characters
[2] Illegal combination of charachters

ج) په پروګرام کې د لارښوونو نامناسب اوډون (ترتیب)[1]

هغه سرچینه پروګرام چې تالیفوونکي یوه ټپروتنه پکې تشخیص کړې وي، هدفي پروګرام ته نه تالیف کېږي. په دې ځای کې تالیفوونکی د یوه مناسب پیغام په مرسته هغه ټپروتنه په ګوته کوي، او ورسره د هغو کوډ شوو ټپروتنو پیغامونه هم ښکاره کوي چې خاص ډول ټپروتنو ته اشاره کوي. د ټپروتنې تشخیص له پروګرام لیکونکي سره د لوړ ارزښت مرسته ده.

خو پام مو وي چې یو تالیفوونکی منطقي ټپروتنې نه شي تشخیص کولای او په پروګرام کې یوازې ګرامري (د نحوي ترکیب) ټپروتنې تشخیصوي. نوموړی د انسان په نیتونو نه شي پوهېدلی. د مثال په ډول، که یو چا په ټپروتنه کې د یوه تن عمر 25- درج کړ، حال دا چې هدف یې 25+ وو، نو تالیفوونکی دا ټپروتنه نه شي تشخیصولای. هغه پروګرامونه چې دا ډول ټپروتنې لري، په پوره بریالیتوب تالیف کېږي او هدفي کوډ د ټپروتنې له پیغام پرته لاسته راځي. دا ښله خبره ده چې دا ډول پروګرامونه چې کله اجرا کېږي، نو سم ځوابونه نه شي وړاندې کولای. نو منطقي ټپروتنې یوازې هغه مهال په ګوتو راځي چې کله پروګرام اجرا شي او پایله یې له مطلوبې پایلې سره سمون ونه خوري. په دې ډول، دا اړینه ده چې د یوه پروګرام په لیکلو کې دقیق اوسو او ډېرو کوچنیو مسایلو ته هم بشپړه پاملرنه وکړو.

[1] Improper sequencing of program instructions

# ترجمان[1]

ترجمان، د ژبو د ژباړونکي یو بل ډول دی چې عالي ژبې ماشینواله کوډ ته ترجمه کوي. دا ترجمان د عالي ژبې یو عبارت ور اوچتوي او ماشینواله لارښوونې ته یې ژباړي چې ورپسې په سملاسي ډول یې اجرا هم ترسره کېږي. نوموړی ترجمان چې کله د عالي ژبو له عبارت سره مخ شي، نو د ژباړې او اجرا پروسې یو په بل پسې ترسره کېږي. په بل عبارت، یو ترجمان یوه لارښوونه ژباړي، او د کنټرول واحد پر هماغه لاسته راغلي ماشینواله کوډ اجراات ترسره کوي، بیا بله لارښوونه ژباړل کېږي، او د کنټرول واحد پر هغه ماشینواله کوډ اجراات ترسره کوي، او همداسې دوام کوي. دا پروګرام له تالیفوونکي څخه توپیر لري چې یوازې ټول سرچینه پروګرام یوه هدفي پروګرام ته ژباړي او د اجرا عملیه پکې شامله نه وي. د تالیفوونکو په خوا کې ټول سرچینه پروګرام د هغه معادل ماشینواله پروګرام ته ژباړل کېږي. ورپسې هغه هدفي کوډ په راتلونکي کې د استفادې لپاره په تلپاتې ډول زېرمه کېږي او د پروګرم په هر ځل اجرا کې استفاده ورڅخه کېږي. په دې اساس د یوه پروګرام د مکرر اجرا کېدو لپاره مکرر تالیف ته اړتیا نه لري. خو، د ترجمان په صورت کې هېڅ ډول کوډ د راتلونکي استفادې لپاره نه زېرمه کېږي، په دې چې د ژباړې او اجرا پروسې یو په بل پسې صورت نیسي. بل ځل که هغې لارښوونې ته اړتیا شي، نو لومړی باید یو ځل بیا ترجمه او ماشینواله ژبې

[1] Interpreter

ته وژباړل شي. د مثال په ډول، د یوه څرخ[1] دننه د لاربنوونو د مکرر پروسېس لپاره، د څرخ هره لاربنوونه باید هر ځل چې څرخ اجرا کېږي، ترجمه شي. ترجمان پروګرامونه تر ډېره له مایکرو کمپیوټرونو (کوچنیو کمپیوټرونو) سره کار راځي. پر تالیفوونکي باندې د ترجمان کټه په سرچینه پروګرام کې د بدلونونو پر وړاندې چټک غبرګون ښودل دي. ترجمان هغه اړتیا له منځه وړي چې هر ځل چې د پروګرام د ځانګړنو زیاتولو یا ټېروتنو سمولو لپاره تغیرات راوستل کېږي، نو لومړی باید تالیف شي. د دې ترڅنګ، تالیفوونکی د یوه ترجمان په پرتله یو پېچلی پروګرام دی. ترجمان پروګرامونه په اسانه لیکل کېږي او په کمپیوټر کې د لویې حافظې شتوالي ته هم اړتیا نه لري. خو بل خوا، ترجمان د ژباړې یو وخت نیوونکی میتود دی، په دې چې هر عبارت، هر ځل چې له سرچینه پروګرام څخه اجرا کېږي، باید وژباړل شي. په دې توګه، تالیف شوی ماشینواله پروګرام د ترجمه شوي پروګرام په پرتله څو چنده چټک ځغلي.

منځګړي، تالیفوونکي او ژباړونکي هغه سیسټمي پوستغالي دي چې د کټې اخیستونکي له لوري لیکل شوی سرچینه پروګرام، یوه داسې هدفي پروګرام ته ژباړي چې د کمپیوټر د سختغالي لپاره معنا ولري. دې ژباړونکو ته ژبنې پروسېس کوونکي[2] هم ویل کېږي، ځکه چې دوی یوه ځانګړې ژبه پروسېس کوي.

---

[1] Loop

[2] Language Processors

د پورتنیو خبرو پر بنسټ، موږ دې نتیجې ته رسېږو چې په عموم کې، یوه پروګرامي ژبه باید له لاندې ځانګړنو څخه برخمنه وي څو یوه عالي ژبه وګڼل شي:

۱. ژبه باید له یوه ټاکلي کمپیوټري سیستم څخه نسبتاً خپلواکه وي. په دې معنا چې د دې پر ځای چې ماشین-اړونده وي، پکار ده چې د ستونزې د حل پر لور یې تمایل ډېر وي.

۲. د ژبې هر عبارت باید یو داسې مېکرو لاربښوونه اوسي چې د ماشینواله ژبې څو لاربښوونو ته ژباړل کېدای وشي.

۳. ژبه باید پروګرام لیکونکو ته دا وړتیا ور وبښي چې د عادي کلماتو او ریاضیکي نښو په کارولو لاربښوونې ولیکي. ژبه باید طبیعي او له هغو لنډیزونو او کلماتو څخه ګټه واخلي چې په ورځنۍ محاوره کې استفاده کېږي.

۴. ژبه باید له تالیفوونکي او ترجمان پرته، د ماشینواله ژبې له لاربښوونو او د سیستمي پوستغالي له نورو برخو څخه خپلواکه وي.

۵. ژبه باید په خپل فطرت کې تجربي نه وي او باید له یوه څخه په زیاتو سیستمونو کې موجوده وي.

## د عالي ژبو ګټې

عالي ژبې د منځنۍ او ماشینواله ژبو په پرتله له لاندې ګټو څخه برخمنې دي:

**۱. ماشین-خپلواکه:** عالي ژبې ماشین-خپلواکه دي. دا یې یوه ډېره ارزښتناکه ګټه ده، ځکه دا په دې معنا ده چې یوه کمپنۍ چې خپل کمپیوټرونه تغیروي – یا حتی یوه ېبل تولیدوونکي ته انتقال کېږي، نو هېڅ اړتیا نشته چې ټول هغه پروګرامونه بیا ځلې ولیکل شي چې همدا اوس تر کار لاندې دي. حتی هغه پروګرامونه چې په عالي ژبو کې لیکل شوي وي، یوازې ځینو بدلونونو ته اړتیا لري، خو دغه بدلونونه نسبتاً واړه او له ډېر زیار پرته په اسانۍ ترسره کېدای شي. په بل عبارت، هغه پروګرام چې په عالي ژبه کې لیکل شوی وي په څو مختلفو کمپیوټرونو کې په ډېر لږ زحمت او یا په عملي کچه له هېڅ ډول زحمت پرته هم ځغلول کېدای شي.

**۲. اسانه زده کړه او کارونه:** دغه ژبې له هغو ژبو سره ډېر زیات نږدېوالی لري چې موږ یې په خپل ورځني ژوند کې کاروو. په دې اساس، د دې ژبو زده کړه او کارونه اسانه ده. پروګرام لیکونکی اړتیا نه لري چې د خپل کمپیوټر په هکله په هر څه پوه شي. دی اړتیا نه لري چې په دې اړه اندېښنه وکړي چې عددونه په څه ډول په کمپیوټر کې ذخیره کړي، چېرته یې ذخیره کړي، څه ورسره وکړي، او داسې نور. په دې معنا چې پروګرام لیکونکي ته هېڅ پکار نه ده چې په ماشینواله لارښوونو، د ډېټا په فارمټ، او داسې نورو موضوعاتو پوه شي. که څه هم، دا ډول پوهه یو څه مطلوبه وي ځکه چې پروګرام لیکونکي ته اجازه ورکوي چې له سیستم څخه په ډېر اغېزمن ډول ګټه پورته کړي.

۳. **کمې تېروتنې**: د عالي ژبو په برخه کې، دا چې پروګرام لیکونکی اړتیا نه لري چې هغه ټول کوچني پراوونه ولیکي چې کمپیوټر یې پرمخ وړي، نو د تېروتنو کولو امکان پکې ډېر کم دی. کمپیوټر په خپله وړو مسایلو ته اوږه ورکوي، او تر هغو په خپل سر یوه تېروتنه نه زېږوي تر څو چې کوم شی ویجاړ نه شي. د دې ترڅنګ، تالیفوونکي په داسې ډول ډیزاین شوي وي چې په خپلکاره ډول د پروګرام لیکونکي تېروتنې په نښه کولای او نیولای شي. په دې اساس، که پروګرام یوه نیمه تشخیصي تېروتنه ولري، نو پروګرام لیکونکی یې په اسانه ډول تشخیص کولای او سمولای شي.

٤. **ټیټ لګښت**: په عالي ژبو کې د پروګرامونو لیکل لږ وخت او کم زحمت غواړي، چې په طبیعي ډول د پروګراملیکنې ټیټ لګښت ته لاره برابروي. عموماً، د پروګرام د چمتو کولو د ټولو فازونو (کوډ کول، ډي بګ کول، ازمایښت کول، او داسې نورو) لګښت د منځنۍ او ماشینواله ژبو په پرتله ټیټ راځي.

۵. **غوره لارښودلیکنه**[1]: عالي ژبې په داسې ډول ډیزاین کېږي چې ټولې لارښوونې یې د ستونزې د ژبې په ډول لیکل کېدای شي. په دې اساس، په عالي ژبو کې د لیکل شوي پروګرام پر عباراتو له ستونزې سره بلد کس په اسانه

[1] Documentaion

پوهېدلای شي. د دا ډول پروګرامونو د لاربنودلیکنې لپاره، ډېر کم او یا په عملي ډګر کې له نشت سره برابر، نظرعبارتونه[1] پکار وي.

**٦. اسانه بدلون:** په عالي ژبه کې لیکل شوي پروګرامونه د منځنۍ او ماشینواله ژبو د پروګرامونو په پرتله په اسانه اصلاح کېږي. د دې اصلي لامل دا دی چې دا ژبې په اسانه د پوهېدو وړ دي او په دې ترتیب د تېروتنو موندل او سمول او په لاربنوونو کې بدلونونه راوستل هم اسانه وي. له یوه پروګرام څخه د ځینو لاربنوونو درج یا ایستنه له کوم تالیف پرته شونتیا مومي. په دې اساس، لوی بدلونونه په ډېر کم زحمت د پلي کېدو وړ وي.

## د عالي ژبو محدودیتونه

د عالي ژبو دوه زیانونه په لاندې ډول دي:

**١. د اغېزمنتیا کمښت:** عموماً، هغه پروګرام چې په منځنۍ یا ماشینواله ژبه کې لیکل شوی وي، د هغه پروګرام په پرتله ډېر اغېزمن راځي چې په عالي ژبه کې شوی وي. په دې معنا چې، د عالي ژبو پروګرامونه د ځغاستې لپاره ډېر وخت او پراخه اصلي زېرمې ته اړتیا لري.

**٢. د انعطاف کمښت:** له دې امله چې د عالي ژبو خپلکاره ځانګړنې تل واقع کېږي او د پروګرام لیکونکي تر کنټرول لاندې نه وي، نو په دې سبب د منځنۍ ژبو

[1] comment statements

څخه يې د انعطاف کچه کمه وي. منځنۍ ژبه، پروګرام ليکونکي ته د ماشين ټولې ځانګړې په لاس کې ورکوي چې د نوموړي په استعمال کې وي. د عملياتو ځينې داسې خاص ډولونه شته چې د منځنۍ ژبې په مرسته په اسانه توګه پروګرام کېدای شي، خو په عالي ژبو کې د عملي کېدو وړ نه دي. د انعطاف دا کمښت په دې معنا دی چې ځينې کارونه په عالي ژبو کې نه شي ترسره کېدای، او يا په ډېر ستونزمن ډول ترسره کېږي.

په زياتره حالاتو کې د عالي ژبو کتې د هغو پر زيانونو ور اوړي. ډېرې کمپيوټري سيسټمونه د زياتره پروګرامونو لپاره عالي ژبه کاروي او منځنۍ ژبې يوازې د هغو خاصو چارو لپاره کاروي چې په بله لاره په اسانۍ نه شي ترسره کېدای.

## ځينې عالي ژبې

د لومړنۍ عالي ژبې د جوړونې درناوی معمولا د ډاکټر ګريس هوپر په برخه کېږي چې د ۱۹۵۲ څخه وړاندې يې د تاليفوونکي او د هغه د ژبې نظريه مطرح کړه. د ډاکټر هوپر تر مديريت لاندې دوه ژبې رامنځته شوې: فلومېټيک[1] يوه تجارتي او کاروباري ژبه وه چې له بهيرچارټونو[2] څخه يې په اسانه متحويات را ټولولای

[1] Flowmatic
[2] Flowcharts

شوی؛ په داسې حال کې چې ماتیمټیک[1] یوه ریاضیکي ژبه وه. دغه دواړه ژبې په مختلفو ډګرونو – کاروبار، سوداګري، ساینس او ریاضي – کې د عالي ژبو لومړنۍ بېلګې دي. له همدې مهال را په دېخوا، پرېمانه عالي ژبې جوړې شوې. نن ورځ له ۲۰۰۰ څخه زیاتې عالي ژبې شتون لري. که څه هم، په دې کې زیاتره د خاصو اهدافو او یا په یوه ځانګړي ډګر کې د ستونزو د حل لپاره ډیزاین شوې. له عالي ژبو څخه یو څو ډېرې عامې ژبې لاندې تشریح شوې. د دې اساسي موخه د دې ژبو په اړه یو عمومي پوهاوی وړاندې کول ده، نه دا چې داسې هر اړخیزه پوهه وړاندې کړو چې د پروګرامونو د لیکلو لپاره اړینه وي.

[1] Mathematic

# «د کمپیوټرپوهنې او ټکنالوژۍ له پوهنغونډ»

# څخه

لیکوال: هېري هېنډرسن

ژباړه: رحمت شاه فراز

# (۱)

## اډا(Ada)

له ۱۹۶۰ز کلونو را په دې خوا، د امریکې د دفاع وزارت د سافټویر جوړونې په هڅو کې ورځ تر بله له زیاتېدونکو ګډوډیو سره مخامخ وو. هر کله به چې یو نوی سافټویر یا کاریال لکه د اړیکو کنټرولوونکی ایجاد شو، نو په عامه توګه به په یوه ځانګړې پروګرامي ژبه کې لیکل کېده. په دې وخت کې له ۲۰۰۰ زیاتې ژبې د سافټویر جوړونې لپاره کارول کېدې، نو په دې اساس د دومره لوی شمېر ناهمغږه[1] سیسټمونو رغونه او اپګریډ کول ستونزمن او ګرانه پرېوته. په ۱۹۷۷ز کې، د دفاع وزارت یوې کاري ډلې د یوې عام-هدفه[2] پروګرامي ژبې د جوړولو طرحه وړاندیز کړه چې د وسلو له کنترول او لارښود سیسټمونو نیولې بیا په ګودامونو کې د بار-کوډ سکېن کوونکو[3] پورې د سافټویرونو د جوړولو لپاره د کارولو وړ وي. په دې لړ کې هغه طرحه چې بریالۍ شوه، اډا نوم ورباندې کېښودل شو چې د نولسمې پېړۍ د کمپیوټرپوهنې د سرلارې اډا لاولېس په نوم پسې نومول شوې وه. د نومورې ژبې د ځانګړنو له بیاکتنې او تصحیح

[1] incompatible
[2] general-purpose
[3] scanners

وروسته، د امریکې د ملي معیارونو اداري[1] په ۱۹۸۳ز کې په رسمي توګه اډا یوه معیاري ژبه اعلان کړه. د دې ژبې لومړنۍ نسخه[2] په اډا-۸۳ هم یادیږي.

## د ژبې ځانګړنې

د اډا په ډیزاین کولو کې، د ژبې جوړوونکو د دودیزو اصولو پر اساس د ژبې هغه بنیادي عناصر را خپل کړل چې په هغو ژبو کې پلي شوي وو چې د ۱۹۶۰ او ۱۹۷۰ کلونو په اوږدو کې جوړې شوې وې. دغه عناصر په خپل ځان کې په غوره ډول تعریف شوي کنټرول جوړښتونه[3] را نغاړي او د ناڅاپي جمپ او یا «goto» له کارولو څخه مخنیوی کوي.

اډا (د کنټرول جوړښتونو او فرعي پروګرامونو[4] په ګډون) د معیاري ژبو له ځانګړنو څخه برخمنه ژبه ده او دغه راز د ګټې اخیستونکي له خوا د تعریف شوو ډېټا ډولونو[5] پېکجونه هم په ځان کې رانغاړي چې هغو کلاسونو ته ورته دي چې وروسته په سي پلس پلس او نورو ژبو کې وکارول شوو. څنګه چې په دې ساده مثال کې لیدل کېږي، د اډا یو پروګرام هغه ډول عمومي بڼه لري چې په

[1] ANSI – Americal National Standards Institute
[2] version
[3] control structures
[4] subprograms
[5] data types

پاسکال ژبه کې کارېدله. (پام مو وي چې په پرېړمخ[1] لیکل شوي لغتونه د ژبې اړین -ویوکي[2] دي.)

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Get_Name is
Name : String (1..80);
Length : Integer;

begin
Put ("What is your first name?");
Get_Line (Name, Length);
New_Line;
Put ("Nice to meet you,");
Put (Name (1..Length));
end Get_Name
```

د پروګرام لومړنۍ کرښه دا مشخصوي چې په پروګرام کې به کوم پېکجونه کارول کېږي. پېکجونه هغه جوړښتونه دي چې له ډېټا ډولونو او اړونده فنکشنونو څخه جوړ وي؛ د مثال په ډول هغه فنکشنونه چې د متن د بدولو او را اخیستلو لپاره کارول کېږي. د مثال په ډول، د Ada.Text.IO پېکج په خپل ځان کې لاندې مشخصات[3] را ټولوي:

---

[1] boldface
[2] keywords
[3] Specifications

```
package Text_IO is
type File_Type is limited private;
type   File_Mode   is   (In_File,   Out_File,
Append_File);
procedure Create (File : in out File_Type;
Mode : in File_Mode := Out_File;
Name : in String := "");
procedure Close (File : in out File_Type);
procedure Put_Line (File : in File_Type;
Item : in String);
procedure Put_Line (Item : in String);
end Text_IO;
```

پورتنی پېکج د فایلونو لپاره د یوه ډېټا ډول په تعینولو پیل کېږي او بیا د یوه فایل د جوړولو او ټړلو، او په فایل کې د متن د ځای پر ځای کولو لپاره یو فنکشن تعریفوي. د سي پلس پلس د کلاسونو په څېر، له ډېرو عمومي پېکجونو څخه پربمانه ځانګړي پېکجونه هم مشتق کېدای شي.

په اصلي پروګرام کې **Begin** د ډېټا اصلي پروسېس پیلوي چې په دې ځای کې د Put فنکشن په مرسته Ada.Text.IO فنکشن یو پیغام څرګندوي او د Get_Line د فنکشن په وسیله د ګټې اخیستونکي ځواب[1] ترلاسه کېږي، او ورپسې یو ځل بیا د Put فنکشن په کارولو هغه متن پر اړیکځای[2] ښودل کېږي چې سملاسي لیکل شوی وي.

---

[1] user response
[2] interface

ابډا د لویو او پیچلو سافټویرې پروژو لپاره یوه مناسبه ژبه ده، ځکه چې د پیکجونو کارول په یوه ډېټا ډول باندې د کار کولو او پلي کولو تفصیلات له نورو کټه اخیستونکو څخه پټوي او ساتنه یې کوي. هغه پروګرام لیکونکی چې په پروګرام کې یې یو داسې پیکج کارول شوی وي چې دا مشخصه کړي چې کوم پارامترونه[1] باید له هر فنکشن سره وکارول شي، نو په دې توګه نوموړی پروګرام لیکونکی له ښکاربدوني اړیکځای[2] څخه کټه نه شي پورته کولای. د ابډا تالیفوونکي په ډېر غوره ډول قانون بندي شوي[3] تالیفوونکي دي څو د ډېټا د بېلا بېلو ډولونو د پروسېس لپاره له ټاکلو ځانګړنو سره سمون وخوري. همدا رنګه، یاده ژبه پیاوړې لیکنې[4] ژبه ده، په دې معنا چې هر ډېټا ډول باید په مشخص ډول تعریف شي، د سي ژبې پر خلاف، چې کله ډېټا ډولونه په خپلواک ډول له یو ډول څخه بل ډول ته اوړي تر څو ترمنځ یې همغږي را پیدا شي، نو کوچنۍ او ګونګې تېروتنې تري پیدا کېږي.

له دې امله چې دا ژبه په مختلطو[5] سیسټمونو او د ریښتین-مهاله[6] فعالیتونو لپاره کارېږي، په دې اساس ابډا کې کټ شمېر داسې ځانګړنې ډیزاین شوې څو ماشینواله کوډ په اغېزمن ډول وربانديې ولیکل شي؛ بله دا چې نوموړې ژبه د دې

[1] Parameter
[2] visible interface
[3] validated
[4] strongly typed
[5] embedded
[6] real-time

ترڅنګ په منځنیو او نورو عالي ژبو کې د روټینونو[1] د لیکلو لپاره هم اسانتیاوې وړاندې کوي. د دې ژبې تازه بڼه، اډا-۹۵، د موازي پروګرام جوړونې[2] په ملاتړ هم ټینګار کوي. که څه هم د اډا راتلونکی ناڅرګند دی، ځکه چې د دفاع وزارت نور دې ته اړتیا نه لري چې یاده ژبه په دولتي ټرونونو کې وکاروي.

خو په ځانګړې ډول د پراخو استازواله ژبو[3] د ځانګړنو په ګډون چې د ګڼشمېر توارث[4] د اړیکځایونو لپاره ملاتړ چمتو کوي، د متنکربنو[5] د غوره سمبالښت او نورو ډېټا ډولونو او فایلونو په برخه کې د اډا پرمختګ تر اوسه پورې دوام لري؛ او دغه راز د ربښتین-مهاله پروسېس او عددي پروسېس[6] په برخه کې هم کار ورته روان دی.

## اضافي لوست:

1- “Ada 95 Lovelace Tutorial.” Available online. URL: http://www.adahome.com/Tutorials/Lovelace/lovelace.htm. Accessed April 18, 2008.

2- Ada 95 On-line Reference Manual (hypertext) Available online. URL: http://www.adahome.com/Resources/refs/rm95.html. Accessed April 18, 2008.

---

[1] د لارښوونو هغه ټولګه چې د یوه ځانګړي کار د ترسره کولو لپاره ډیزاین Routines شوي وي. -

[2] parallel programming

[3] object oriented language

[4] multiple inheritance

[5] string

[6] Numeric Process

3- Barnes, John. *Programming in Ada 2005 with CD.* New York: Pearson Education, 2006.

4- Dale, Nell, and John W. McCormick. *Ada Plus Data Structures: An Object-Oriented Approach.* 2nd ed. Sudbury, Mass.: Jones and Bartlett, 2006.

***

# (۲)

## الګول (Algol)

په ۱۹۵۰ او ۱۹۶۰ کلونو کې د کمپیوټر دوو عالي ژبو سر را پورته کړ چې په لویه کچه تر استعمال لاندې وې. لومړنۍ ژبه د دې لپاره ډیزاین شوې وه څو د ساینسي محاسبو لپاره یوه اغیزمنه ژبه واوسي او دوهمه ژبه بیا د کاروباري فعالیتونه لپاره ډیزاین شوې وه، چې ډېر ټینګار یې په ډېټا پروسېس باندې وو. خو بیا هم ډېرې پروګرامونه په ټیټو ژبو کې لیکل کېدل چې دا ډول ژبې د یوه خاص ماشین د سختغالي له ځانګړنو څخه د ګټو اخیستو لپاره ډیزاین شوې وې.

د دې لپاره چې د محاسبې او شمېر میتودونه په اسانه ډول شریک او څرګند کړای شي، مخکښو پروګرام لیکونکو د یوې داسې پروګرامي ژبې په جوړولو لاس پورې کړ چې یوازې د یوه ځانګړي کار او یا هارډویري دریځ لپاره نه وي ډیزاین شوې. له ۱۹۵۷ز نه مخکې، د جرمني GAMM[1] او د امریکې ACM[2] یا د کمپیوټري ماشینرۍ شرکت ادارو خپل ځواکونه سره یو ځای کړل تر څو د داسې یوې ژبې لپاره مشخصات رامنځته کړي. د دې هڅو پایله د Zurich

[1] Gesellshaft fur angewandte Mathematic und Mechanik
[2] Association for Computer Machinery

Report یا الګول-۵۸ په بڼه مخې ته راغله او الګول-۶۰ د دې ژبې هغه نسخه شوه چې تر ټولو لومړی په لویه کچه استعمال او تطبیق[1] کړای شوه.

## د ژبې ځانګړنې

الګول یوه بلاک-جوړښته[2] او پروسوي[3] ژبه ده. په نوموړې ژبه کې هر متحول[4] د ډېټا د یو کوچني شمېر ډولونو پورې د اړوند کېدو لپاره تعریفېږي د بېلګې په ډول کامل اعداد[5]، حقیقي اعداد او یا له دواړو څخه د یو ډول اعدادو لرې. په داسې حال کې چې د ډېټا ډولونو شمېر پکې محدود دی او دا چې د نوو ډولونو د تعریفولو لپاره اصلا اسانتیا پکې نشته، نو د تالیفوونکي ډول کنټه[6] (دا باوري کول چې د ډېټا توک[7] د تعریف شوي متحول له ډول سره سمون خوري او که نه) تر یوې کچې داسې خوندیتوب چمتو کاوه چې په پخوانیو ژبو کې نه تر سترګو کېده.

د الګول یو پروګرام په لوی شمېر بېلا بېلې پروسې[8] لرلای شي او یا هم په خارج کې له تعریف شوو پروسو څخه ګټه پکې پورته کېدای شي. د دې ترڅنګ، د

---

[1] Implement
[2] block structured
[3] procedural
[4] variable
[5] integer
[6] type checking
[7] item
[8] procedure

ورته نومونو متحولونه چې په بېلا بېلو پروسوي بلاکونو کې تعریف شوي وي، پخپلو کې له یو بل سره تکر نه کوي. یوه پروسه کولای شي چې بېرته خپل ځان را وبولي او یا یې کال[1] کړي. نوموړې ژبه له معیاري کنټرول جوړښتونو څخه هم برخمنه ده.

لاندینی د الګول ساده پروګرام له ۱ څخه تر ۱۰ پورې اعداد په یوه کتار[2] کې زېرمه او جمع کوي، او بیا یې پایله پر صفحه څرګندوي:

```
begin
integer array ints[1:10];
integer counter, total;
total := 0;
for counter :=1 step 1 until counter > 10
do
begin
ints [counter] := counter;
total := total + ints[counter];
end;
printstring "The total is:";
printint (total);
end
```

[1] call
[2] array

## د الګول ترکه

د الګول په هغه نسخه کې چې په الګول-۶۸ پټژندل کېږي، د (بولین[1] او سم/ناسم وېلیوکانو[2] په کڅون) د ډېټا ډولونو شمېر ډېر کړای شوی او د ګټې اخیستونکي له خوا تعریف شوي ډولونه او سټرکټونه[3] هم ور زیات شوي. سټرکټونه هغه ریکارډونه دي چې د ډېټا د مختلفو ډولونو لپاره خپلې ساحې لري. نښه کوونکي[4] چې وېلیوکانو ته مراجعه کوي، هم پکې پلي شوي، او پارامترونه هم له داسې اسانتیاوو څخه برخمن دي چې فنکشنونو ته ور تېرېدای[5] او یا ېرته تېې اخیستل کېدای شي.

که څه هم الګول د کمپیوټر په ځینو مرکزونو (په ځانګړي ډول په اروپا) کې د تولیدي ژبې[6] په توګه کارول کېده، خو د نوموړې ژبې نسبي پېچلتیا او له دې ژبې سره نابلدتیا یې مقبولیت یو څه ټکنی کړی، د دې ترڅنګ په دې جریان کې د فورټران او په ځانګړي ډول کوبال په څېر سیالو ژبو هم پراخه ونډه لرلې ده. الګول خپل تر ټولو ستر بریالیتوب په دوو برخو کې ترلاسه کړ: د یو څه وخت لپاره نوموړې ژبه د کمپیوټر پوهانو لپاره د نوو الګوریتمونو د تشریح کولو

---

[1] boolean
[2] values
[3] structs
[4] pointers
[5] pass
[6] Production language

لپاره یو غوره انتخاب وو او دوهم دا چې د دې ژبې جوړښتي ځانګړنې په هغو پروسوي ژبو کې هم وکارول شوې چې د ۱۹۷۰ ز کلونو په شاوخوا کې رامنځته شوې.

## اضافي لوست:

1- “Algol 68 Home Page.” URL: http://www.algol68.org. Accessed April 10, 2007.

2- Backus, J. W., and others. “Revised Report on the Algorithmic Language Algol 60.” Originally published in *Numerische Mathematik,* the *Communications of the ACM,* and the *Journal of the British Computer Society.* Available online. URL: http://www.masswerk.at/algol60/report.htm. Accessed April 10, 2007.

***

## (۳)

## اې پي ایل[1] (APL)

دغه پروګرامي ژبه د هارورډ (او وروسته د آی بي اېم شرکت[2]) د یوه څېړونکي کښنیت اي. ایورسن له خوا په ۱۹۶۰ کلونو کې جوړه شوه، تر څو ریاضیکي عملیې د کمپیوټر د استعمال لپاره په ښکاره او منظم ډول څرګندې کړای شي. د نوموړې ژبې په غونډ[3] شکل د ریاضیکي عملیو د وړاندې کولو ځواک د ډېرې ګټې اخیستونکو پام را جلب کړ او اې پي ایل په لنډ وخت کې یوه عام-هدفه کمپیوټري ژبه وګرځېده.

د بېسیک[4] د ډېرو نسخو په څېر، اې پي ایل یوه ترجمه کېدونکې ژبه ده، په دې معنا چې د پروګرام لیکونکي ورکړیز په ورته مهال کې ارزول کېږي او د دې ځانګړنې له مخې تعاملي[5] ځواب شونتیا مومي. د بېسیک او فورټران پر خلاف، اې پي ایل د کتارونو او میټریکسونو په کډون د ټولو ریاضیکي عملیو لپاره ځواکمن او مستقیم ملاتړ چمتو کوي.

---

[1] A programming language – یوه پروګرامي ژبه
[2] IBM – International business machine
[3] Compact
[4] Basic
[5] Interactive

اې پي اېل له سلو زیات خپل ځاني[1] عاملونه[2] لري چې اولیه[3] نومېږي. یوازې د یوه یا دوو عاملونو په کارولو پروګرام لیکونکی کولای شي چې پیچلي کارونه لکه د عددي او مثلثي عملیو استخراجول، د اعدادو ترتیبول[4]، او یا د کتارونو او میټریکسونو بیاترتیتول ترسره کړي. (په اصل کې، د اې پي اېل تر ټولو ستر ځواک دا دی چې د څرګند-څرخونو[5] او یا خارجي لایبریري فنکشنونو[6] له ترتیب کولو پرته کولای شي چې په مستقیم ډول پر میټریکسونو عملیې ترسره کړي.

د یوه ساده مثال په توګه، د اې پي اېل د کوډ لاندې کرښه:

$$X\,[\Delta\,X]$$

د X کتار ترتیبوي. په ډېری پروګرامي ژبو کې دا کار باید د یوه داسې ترتیبي الګوریتم[7] د کوډ لیکلو له لارې ترسره شي چې د هغه په لسهاوو کرښو کوډ کې ځاله-څرخونه[8] او لنډمهاله متحولین کارول شوي وي.

بل خوا، اې پي اېل د زیاتره پروګرام لیکونکو لپاره ګټې نیمګړتیاوې او عیبونه هم درلودل. په دې چې نوموړې ژبه د ډېری عاملونو لپاره یوناني توري کاروي، چې

---

[1] built-in
[2] Operators
[3] primitive
[4] sorting
[5] explicit loops
[6] Library functions
[7] Sorting algorithm
[8] nested loops

د دې تورو لپاره یوه ځانګړي فونټ ته اړتیا وه چې هغه مهال له آی بي ایم شرکت پرته په نورو سیسټمونو کې نه موندل کېده. د دې لپاره چې ژبه ساده او پراخه کړای شي، د J په نوم یوه لهجه هم په کار واچول شوه چې یوازې معیاري اسکي کرکټرونه[1] کاروي. د ډېرې پروګرام لیکونکو لپاره د اې پي ایل ریاضیکي معادلات ګونګ رمز بندي شوي[2] وي چې په دې ډول یې د پروګرامونو بیاکتنه او اصلاح ورته ستونزمنه کړي. خو پر دې سربېره هم، د اې پي ایل د مینوالو ځانګړې ډله په سترو کمپیوټري ټولنو کې لا هم له دې ژبې سره پراخه لېوالتیا ښیي.

## اضافي لوست:

1- ACM Special Interest Group for APL and J Languages. Available online. URL: http://www.acm.org/sigapl/. Accessed April 12, 2007.
2- "APL Frequently Asked Questions." Available from various sites including URL: http://home.earthlink.net/~swsirlin/apl.faq.html. Accessed May 8, 2007.
3- Gilman, Leonard, and Allen J. Rose. *APL: An Interactive Approach.* 3rd ed. (reprint). Malabar, Fla.: Krieger, 1992.
4- "Why APL?" Available online. URL: http://www.acm.org/sigapl/whyapl.htm. Accessed.

***

[1] ASCII Characters
[2] cryptic

# (٤)

# آک(Awk)

آک یوه متنواله ژبه[1] ده چې د «آلفرېډ وي. آهو»، «بریان ډبلیو»، «کرنیګان» او «پیټر جي. وینبرګر» له خوا د یونیکس د عامل سیستم لپاره په ۱۹۷۷ کې جوړه شوه. (د آک کلمه د پورتنیو کسانو د تخلص د لومړنیو تورو یو لنډیز دی.) نوموړې ژبه د عامل سیستم د مرستندویه پوستغالو[2] سره سمون لرونکو نقشو[3] پر اساس جوړه شوې او په ابتدایي ډول له فایلونو څخه د ډېټا د راویستلو او راپور لیکلو لپاره ډیزاین شوې. د نورو عامل سیستمونو لکه ډاس[4] لپاره د آک ژبې کڼ بدیلونه لیکل شوي دي.

د نورو متنواله ژبو په څېر، د آک یو پروګرام د هغو کمانډونو په یوه لیکلړ مشتمل وي چې د آک ژباړونکي په مرسته له یوه فایل څخه لوستل کېږي. د مثال په ډول د یونیکس لاندینۍ امر کړئه:

```
awk – f Myprogram > Report
```

---

[1] scripting language
[2] utility
[3] pattern
[4] DOS - Disk Operating System

د MyProgram له فایل څخه د آک عبارات[1] د آک ژباړونکي ته ور سپاري او بیا د پروګرام راکړیز Report نومي فایل ته استوي.

## د ژبې ځانګړنې

د آک یو عبارت په خپل ځان کې دوه شیان رانغاړي: یوه داسې نقشه[2] چې له پایلې سره سمون وخوري او یو عمل چې ترسره شي (که چېرته جوړښت ته اړتیا نه وه، نو له منځه هم وړل کېدای شي). لاندې یې څو مثالونه دي:

```
{print $1} # prints the first field of every
     # line of input (since no pattern
     # is specified)
/debit/ {print $2} # print the second field
of
     # every line that contains the
     # word “debit”
if ( Code == 2 ) # if Code equals 2,
print $3 # print third field
     # of each line
```

د نقشو ورته والی یو شمېر منظم معادلات کاروي چې د یونیکس کټه اخیستونکي ورسره بلد دي او عملونه د محدود شمېر خو بسندویه کنټرول جوړښتونو په کارولو مشخص کېدای شي چې دا ډول کنټرول جوړښتونه په سي ژبه کې هم موندل کېږي. د دې ترڅنګ په دې ژبه کې خپل ځاني متحولونه (د ساحو او

[1] Statements
[2] pattern

کړنو د شمېر لپاره د شمېرونکو[1] په کډون)، ریاضیکي عملیې، له ساحو[2] څخه د متن د استخراجولو لپاره ګټور متنکړنیزه فنکشنونه او ریاضیکي او نسبتي عاملونه هم شته. دغه راز، کولای شو چې د څو اړخیزه print فنکشن له لارې راکړیز ته هم ډول ډول فارمټونه ورکړو چې د سي ژبې پروګرام لیکونکي ورسره بلدتیا لري.

آک په یونیکس سیسټمونو کې له فایلونو او ساده ډېټابېسونو څخه د معلوماتو د راویستلو لپاره پراخه شهرت وموند. د لا پېچلو سافټوېرونو لپاره نوموړې ژبې خپل ځای Perl ته پرېښود چې د ډېټابېسونو لپاره د مشخصاتو یوه لویه ټولګه وړاندې کوي.

## اضافي لوست:

1- Aho, Alfred V., Brian Kernighan, and Peter J. Weinberger. *The Awk Programming Language.* Reading, Mass.: Addison-Wesley,
1998.

2- Goebel, Greg. “An Awk Primer.” Available online. URL: http:// www.vectorsite.net/tsawk.html. Accessed May 22, 2007.

[1] counters
[2] field

***

## (۵)

## بېسیک(BASIC)

بېسیک یا «د مبتدي ټول-هدفه سمبولیک لارښود کوډ»[1] ژبه په ډارټماؤټ کالج کې په ۱۹۶۴ کې د «جې. کېمیني» او «تي. کرتز» له خوا جوړه شوه. هغه مهال د کالج ټول کېمپس له ټرمینلونو سره ټړل شوو په مهال-شریکه[2] کمپیوټرونو باندې سمبال وو چې دا کار په داسې وخت کې یو لوی نوښت وو چې ډېرې کمپیوټرونه به یوازې له یوه ځای څخه د پنچکارتونو په وسیله پروګرام کېدل. جان. جې. کېمیني او تامس کرتز غوښتل چې یوه داسې ژبه جوړه کړي چې په اسانۍ سره زده او وکولای شي چې د کیبورډ له لارې لیکل شوي اوامر سمدلاسه تالیف او ځواب ورته ووایي. او په دې ډول د خپلو سیستمونو له تعاملي[3] ځانګړنې څخه ګټه پورته کړي. د دوی دا فکر د هغه مهال له ژبو لکه کوبال، الګول او فورټران سره له یوې مخې په تضاد کې وو، چې په دې ژبو کې به پروګرامونه لومړی په بشپړ ډول لیکل کېدل او بیا به یې ازمایښت ترسره کېده.

---

[1] Basic All-Purpose Symbolic Instruction Codee
[2] time-shared
[3] interactivity

د هغو ژړو ژبو په خلاف چې پنچکارتونه به يې کارول، د ښسيک پروګرامونو کې اړتيا نه وه چې اړين-ويوکي په مشخصو ليکو کې وليکل شي. بلکې، عبارات د انګليسي ژبو د کړښو په څېر ليکل کېدای شوی، پرته له دې چې ليکنښو او د واټن[1] په اړه ځانګړي اصول مراعت شي. په عمومي ډول، د پرېکړيزو[2] او کنټرول جوړښتونو لپاره نحوي ترکيب[3] د نورو ژبو په پرتله ساده دی. د مثال په ډول، يو for څرخ چې له ۱ څخه تر ۱۰ پورې اعداد شمېري، په لاندې ډول ښکاري:

```
for (i = 1; i <= 10; i++)
       printf("%d", i);
The same loop in BASIC reads as follows:
for i = 1 to 10
       print i
next i
```

## ښسيک ژبه او کوچنې کمپيوټرونه[4]

د ۱۹۶۰ او ۱۹۷۰ کلونو په اوږدو کې ښسيک ورځ په ورځ زياتېدونکو مهال-شريکه کمپيوټرونو کې کارول کېدله. د نوموړې ژبې د ساده والي او په اسانۍ د استعمال کېدو ځانګړتيا دا ژبه د لنډو مرستندويه پوستغالو د ليکلو او په ځانګړي ډول د کمپيوټرپوهنې له پوهنځيو پرته په نورو پوهنځيو کې د کمپيوټر د بنسټيز مفاهيمو د ښوونې لپاره کټوره کړځولې وه. د ۱۹۸۰ کلونو په لومړيو کې هغه

---

[1] spacing
[2] Decision
[3] Syntax
[4] Microcomputers

مهال چې لومړني شخصي کمپیوټرونه د پراخه استعمال لپاره منځته راغلل، د دې کمپیوټرونو د زېرمې ظرفیت عموما له ۸ کیلو بایټ څخه تر ۶۴ کیلو بایټ پورې وو، چې د یوه ایډیټور، تالیفوونکي او نورو مرستندویه پوستغالو د ځغلولو لپاره کافي نه وو چې د سي په څېر د یوې ژبې لپاره ورته اړتیا وه. خو بیا هم، د ښسیک د ژباړونکي یوه ساده نسخه د تش-لوسته زېرمې[1] په چیپ کې ځای په ځای کېدلای شوی، هغسې چې په اپیل-۲ کمپیوټرونو ، لومړنیو آی. بي. اېم شخصي کمپیوټرونو او لسګونو نورو شخصي کمپیوټرونو کې دود وو. د ښسیک نورې پرمختللې نسخې (د تالیفوونکو په ټېپ[2] کې زېرمه کېدای شوی. (بیل ګېټس نومي ځوان سوداګر چې لومړی ځل یې دا ډول تولیدات خرڅلاو ته وړاندې کړل).

کله چې ښسیک د مختلفو کوچنیو کمپیوټرونو لپاره ومنل شوه، نو په نتیجه کې یې د نوموړې ژبې ګڼې لهجې[3] منځته راغلې. د ګرافیکونو د جوړولو او په مستقیم ډول پر سختغالو او زېرمه – له زېرمې څخه د وېلیو اخیستل[4] او یا تغیر

[1] ROM – Read-Only-Memory
[2] Tape
[3] Dialects
[4] Peek

پکې راوستل[1] - د عملیو د ترسره کولو لپاره د اوامرو شتون د بېسیک پروګرامونه د ځانګړي دریځ[2] لپاره د کارول کېدو وړ وګرځول.

ورو په ورو، څنګه چې د کوچنیو کمپیوټرونو د زېرمې ظرفیت او د پروسېس ځواک لوړېده، بېسک هم خپل ځای د کوچنیو کمپیوټرونو لپاره د پېچلو پوستغالو د جوړولو لپاره د پاسکال (په ځانګړي ډول د جوړونې-مختلط-چاپېریال[3] چې په سانډیاګو کې د کالیفورنیا په پوهنتون کې جوړ شو) او سي (د یونیکس له ټولنې څخه) په څېر ژبو ته پرېښودل پیل کړل.

## تنقید او لرلید

د بېسک زیاتره نسخو له کرښه-ګڼو[4] (د هغو لومړنیو متن اډیټ کوونکو یو میراث چې کرښه په کرښه به یې کار کاوه) څخه ګټه اخیستله او همدا رنګه د دې لپاره چې د پروګرام کنټرول یوې ځانګړې کرښې ته ټوپ کړي، د goto عبارت هم پکې کارول کېدای شوی. که څه هم په نومورې ژبه کې ساده فرعي پروګرامونه ول (چې د gosub عبارت په مټ ورته لاسرسی کېده)، خو بیا هم د پاسکال او سي ژبو په څېر یوه فنکشن ته د متحول د ور تېرولو له وړتیا څخه بې برخې

[1] poke
[2] Specfic Platform
[3] integrated development environment
[4] Line numbers

وو. په اصل کې به ټول متحولین نړیوال[1] وو، په دې معنا چې متحولینو ته به د پروګرام له هرې برخې څخه لاسرسی کېدای شوی، چې د دې خطر یې رامنځته کاوه چې په بې پامۍ سره یې وېلیوکانې تغیر شي.

څنګه چې د جوړښتواله پروګرام جوړونې[2] اصولو وده وکړه، د ښسیک جوړښتواله نیمګړتیاوو دا ژبه د هغو کمپیوټر پوهانو له پامه وغورځوله چې پاسکال ته یې د پروګراملیکنې د ښوونې او سي ژبې ته یې د سیستمي پروګراملیکنې لپاره لومړیتوب ورکاوه. په ۱۹۸۴ کې، د ښسیک اصلي جوړوونکو هغو ستونزو ته ځواب ووایه چې دوی د «د کوڅې ښسیک» په نوم پېژندل او «رښتینې ښسیک» یې بازار ته وړاندې کړه، چې د نوموړې ژبې یوه عصري او ښه رغېدلې نسخه وه او د ښسیک- ۱۹۸۸- ANSI نسخه د ژبو معیاري ځانګړې هم ور خپلې کړې. دې هڅو یوازې محدود اغېز درلود. مایکروسافټ کمپنۍ د سیستمي پروګراملیکنې لپاره د ښسیک نوې نسخې رامنځته کړې (چټک ښسک[3] په ۱۹۸۰ کلونو او عیني ښسیک[4] په ۱۹۹۰ کلونو کې) چې له پرمختللو کنټرول جوړښتونو او ډېټا ډولونو څخه برخمنې وې او دغه راز د بې ګټو ګربنه-ګټو اړتیا یې هم له منځه وړې وه. عیني ښسک په ځانګړې توګه د پام وړ بریالیتوب ترلاسه کړ چې د دودیز ښسیک تعاملي وړتیا او له وړاندې څخه د

[1] Global
[2] Structured Programming
[3] quick basic
[4] visual basic

جوړ شوو ځواکمنو کنټرولونو پېکج ته د لاسرسي له اسانتیاوو څخه ترکیب وو. دغو پېکجونو مینوګاني[1]، مکالموي چوکاټونه[2] او د ویندوز د اړیکځای نور ځانګړي مشخصات چمتو کول. د عیني بېسیک اوسنیو بڼو تر ډېره استازواله[3] حالت ور خپل کړی او د سي پلس پلس په څېر له کلاسونو څخه ګټه اخلي.

که څه هم د بېسیک نوي حالتونه اغېزمن راتلونکی لري، خو له دې خبرې سره دا اختلاف کېدای شي چې د اصلي بېسیک زیاتره ځانګړنې (چټکتیا او د پروګراملیکنې تعاملي تګلاره[4]) چې دا ژبه یې له نورو بېلوله، دمګړۍ نه تر سترګو کېږي. نن ورځ لنډ مرستندویه پروګرامونه تر ډېره په یوه نه یوه متنواله ژبه کې لیکل کېږي.

**اضافي لوست:**

1- Brin, David. “Why Johnny Can’t Code,” September 14, 2006. Available online. URL: http://www.salon.com/tech/feature/2006/09/14/basic/print.html. Accessed April 24, 2007.

2- Kemeny, J. G., and Thomas E. Kurtz. *Back to Basic: The History, Corruption, and Future of the Language.* Reading, Mass.: Addison- Wesley, 1985.

3- Lomax, Paul, and Ron Petrusha. *VB and VBA in a Nutshell: The*

---

[1] Menus
[2] Dialog Box
[3] object oriented
[4] Interactive approach to programming

*Languages.* Sebastopol, Calif.: O'Reilly, 1998.

4- Neuberg, Matt. *REALbasic: The Definitive Guide.* 2nd ed. Sebastapol, Calif.: O'Reilly, 2001. Sempf, Bill. *Visual Basic 2008 for Dummies.* Hoboken, N.J.: Wiley, 2008.

***

# (٦)

## سي(C)

سي پروګرامي ژبه د ۱۹۷۰ کلونو په لومړيو وختونو کې د «ډېنيس ريچي» له خوا جوړه شوه چې اساس يې په لومړنيو ژبو لکه بي. سي. پي. اېل.[1] او بي.[2] ژبو ايښودل شوی وو. سي د لومړي ځل لپاره په DEC PDP-11 کمپيوټرونو کې وکارول شوه چې په همدې وروستيو وختونو کې جوړ شوی يونيکس عامل سيستم باندې ځغلېده، په دې ځای کې سي ژبې د ګڼو مرستندويه پوستغالو د جوړولو لپاره د PDP منځنۍ ژبې لپاره د عالي ژبې بديل وړاندې کړ چې دې کار يونيکس ته انعطاف ور په برخه کړ. له ۱۹۸۰ کلونو را په دېخوا، سي او د

[1] BCPL
[2] B

هغې بچونې سي پلس پلس، تر ټولو زیاتې کارېدونکې پروګرامي ژبې ګڼېدلې دي.

## د ژبې ځانګړنې

د لومړنۍ الګول او تر یو څه بریده پاسکال په څېر، سي یوه پروسوي ژبه[1] ده چې د هغه پروګراملیکنې فلسفه انعکاس کوي چې د ۱۹۷۰ کلونو په اوږدو کې یې ورو په ورو بڼه ومونده. په عمومي ډول، د سي ژبې تګلاره داسې بیانېدای شي چې د کمپیوټر د رښتیني فطرت لپاره اړینې ځانګړتیاوې په غوند او اغېزمن حالت کې وړاندې کوي. نوموړې ژبه له اساسي کنټرول جوړښتونو لکه if، switch، while، do او for څخه برخمنه ده او دغه راز د کاملو عددونو لپاره د « int, short, long»؛ د اعشاري عددونو لپاره د «float او double» او د تورو لپاره د «char» خپل ځانې ډېټا ډولونه وړاندې کوي. د هر ډول ډېټا لپاره یو کتار[2] تعریف کېدای شي او یوه متنکرښه د تورو د یوه کتار په څېر پکې پلي کېږي.

نښه کوونکي[3] (چې زېرمه ځایونو ته اشاره کوي) د بېلا بېلو موخو لپاره کارول کېږي، مثلا په یوه کتار کې د ډېټا د ترتیبولو او یا اخذ کولو لپاره. په داسې حال کې چې په نښه کوونکو باندې پوهېدل د مبتدیانو لپاره یو څه ستونزمن پرېوتی شي، خو په اصل کې دا ځانګړنه د سي ژبې هغه ټینګار انتقالوي چې یاده ژبه د

[1] Procedural language
[2] array
[3] pointers

سیستمي پروګراملیکنې ژبه ده چې کولای شي په زېرمه باندې د عملیو د ترسره کولو لپاره له پوستغالي سره نږدې اړیکې وساتي.

د سټرکټ نومي ریکارډ کې د ډېټا بېلابېل ډولونه زېرمه کېدای شي. د مثال په ډول:

```
struct Employee_Record {
     char [10] First_Name;
     char [1] Middle_Initial;
     char [20] Last_Name;
     int Employee_Number;
} ;
```

(یو بل ډول یووالی هم شته، چې په اصل کې سټرکټ دی چېرته چې همدا پورتنی جوړښت کولای شي له دوو مختلفو ډېټا توکونو څخه یو په ځان کې ولري.)

د معیاري ریاضي او منطقي مقایسې عاملونه هم پکې شته. یاده ژبه له نا اشنا اړخونو څخه هم برخمنه ده: د مساوي لپاره د مقایسې عامل == دی ، حال دا چې د مساوي یوه علامه = ټاکونکی عامل[1] دی. دا شی په ځینو وختونو کې یوه سرګرداني هم جوړولی شي، لاندې شرط

```
if (Total = 10)
  printf ("Finished!");
```

[1] Assignment operation

تل Finished ښکاروي، ځکه چې Total = 10 د تل لپاره د ۱۰ وېلیو راوباسي (او څنګه چې دا وېلیو صفر نه ده، نو پایله یې صحیح ده او په دې ډول شرط هم صحیح دی.)

سي د زیاتونې[1] ++ او کمونې[2] -- عاملونو هم لري، چې په یوه شمېرونکي څرخ[3] کې د یو په واحد د یوه متحول د زیاتولو او کمولو د عملیې لپاره یوه اسانه لاره ده. په سي کې لاندې عبارات د یو بل پر ځای کارول کېدای شي:

```
Total = Total + 1;
Total += 1;
Total ++;
```

د پاسکال د دوو مختلفو پروسو پر خلاف چې (فنک یا فنکشن، چې یوه وېلیو واپس کوي، او پروک یا پروسیجر، چې وېلیو نه واپس کوي)، سي یوازې فنکشنونه لري. آرګومېنټونه، فنکشنونو ته د وېلیوګانو په وسیله ور تېرېږي، او د نښه کوونکي په کارولو کولای شو چې د ریفرنس له لارې یې هم ور تېر کړو.

## د پروګرام بېلګه

لاندې د یوه لنډ پروګرام بېلګه وړاندې شوې:

```
#include <stdio.h>
float Average (void);
```

---

[1] Increment
[2] Decrement
[3] Counting loop

```
main () {
printf ("The average is: %f", Average() );
}
float Average (void) {
int NumbersRead = 0;
int Number;
int Total = 0;
while (scanf("%d\n", &Number) == 1)
{
Total = Total + Number;
NumbersRead = NumbersRead + 1;
}
return (Total / NumbersRead);
}
}
```

د پروګرام د پیل عبارتونه چې په # نښه پیل کېږي د پروسېس لارښودونه[1] بلل کېږي. دا لارښودونه تر تالیف کېدو مخکې په سرچینه کوډ کې تغیرات راولي. د #include لارښود پروګرام ته ځانګړی اصلي فایل ور اضافه کوي. د نورو ژبو برعکس، سي ژبه پخپل سر ډېرې اساسي فنکشونه نه لري، لکه ورکړیز/راکړیز یا I/O عبارتونه. د دې پر ځای، دا ټول فنکشنونه په سټاندرد لایبرېري کې پراته وي. (د دې ترتیب موخه دا ده چې پخپله ژبه ساده او سپکه شي، او که چېرته نومورې پروګرام یوه بل درېځ ته هم وړل کېږي، نو کار ورکړي.) د stdio.h فایل، په دې ځای کې یو سرلیک فایل[2] دی چې I/O فنکشنونه تعریفوي، لکه

[1] processor directive
[2] header file

printf() (چې فارمټ شوي[1] معلومات پرنټ کوي) او scanf() (چې په پروګرام ډېټا لولي او فارمټ ورکوي.)

د پروګرام بله برخه هر هغه فنکشن تعریفوي چې وروسته به په پروګرام کې تعریف او کارپرې (په دې ځای کې، یوازې یو فنکشن لرو، Average). فنکشن تعریفول په هغه ډېټا ډول پیل کېږي چې د فنکشن په واسطه به رابلونکي عبارت[2] ته واپس کېږي چې په دې ځای کې اعشاري وېلیو ده. د فنکشن له نوم وروسته د هغو پارامترونو تعریفول راځي چې د بلونکي په مرسته به فنکشن ته ور تېرېږي. دا چې Average فنکشن به خپله ډېټا د دې پر ځای چې له بلونکي عبارت څخه یې واخلي، د ځټه اخیستونکي له ورکړیز څخه به یې اخلي، نو په دې ځای کې د پارامتر په توګه د (void) وېلیو کارول شوې.

د Average فنکشن له تعریفولو وروسته د main() فنکشن نوبت راځي. هر سي پروګرام باید یو اصلي فنکشن ولري. مېن هغه فنکشن ده چې هغه وخت ځغلي چې کله پروګرام اجرا (execute) کېږي. عموما، مېن د اړینو کارونو د ترسره کولو لپاره له نورو فنکشنونو څخه یې یو شمېر ته بلنه ورکوي. په دې ځای کې مېن د printf عبارت دننه د Average فنکشن ته بلنه ورکوي، چې هغه

[1] formatted
[2] calling statement

اوسط به پرنټ کړي چې د هغه فنکشن له خوا واپس شوی وي. (د نورو عبارتونو دننه یوه فنکشن ته بلنه ورکول د سي د غښتلي نحوي ترکیب یوه بېلګه ده.)

په پای کې، د Average فنکشن تعریفېږي. دا فنکشن د ډېټا نمبرونو د لوستلو لپاره څرخ کاروي، چې جمع کېږي او بیا د اوسط لپاره تقسیم کېږي، او ورپسې د return عبارت په مټ بېرته بلونکي عبارت ته لېږدول کېږي.

یو پروګرام لیکونکی کولای شي چې د یوه اډیټور لکه vi په مرسته، همدا کوډ په یوه اصلي فایل یا سرچینه فایل (چې په دې ځای کې test.c دی) کې ولیکي او پورتنی پروګرام په یونیکس سیسټم باندې هم جوړ کړي. همدا سرچینه کوډ بیا د سي تالیفوونکي (په دې ځای کې gcc) ته ورکول کېږي. سرچینه کوډ تالیف کېږي، اړیکه مومي او بیا د اجرایي پروګرام فایل a.out ترې چمتو کېږي. له هغه وروسته که همدا نوم په کمانډ پرمپټ کې ولیکل شي نو پروګرام ځغلي، چې لومړی اعداد اخلي او ورپسې یې اوسط راوباسي.

```
% gcc test.c
% a.out
5
7
9
.
The average is: 7.000000
```

## بریا او بدلون

له رامنځته کېدو درې لسیزې وروسته، سي په تاریخ کې یوه بریالۍ پروګرامي ژبه وګرځېده. د دې ترڅنګ چې نوموړې ژبه د یونیکس پروګراملیکنې لپاره د خوښې وړ ژبه وه، نو څنګه چې کوچنۍ کمپیوټرونه د دې جوګه شول چې عالي ژبې وغځلوي، سي ژبه د مایکروسافټ ډاس، وینډوز او مېکنټاش پروګرامونو د لیکلو لپاره د خوښې وړ انتخاب وګرځېد. د بېلګې په ډول، د وینډوز لپاره د کاریال پروګرامي اړیکځای یا API[1] د سلګونو په شمېر د سي فنکشنونه، جوړښتونه او تعریفونه[2] لري.

خو پر دې سربېره، سي د کمپیوټر پوهانو ترمنځ له نیوکو او انتقاد څخه پټه نه ده پاتې شوې. د هغو اصطلاحاتو ترڅنګ چې د مرموز کوډ لیکنې[3] لپاره یو څوک هڅولای شي، د سي اصلي بڼه (چې کرنیګن او ریچي په The C Programming Language کتاب کې تعریف کړې) پارامترونه نه ګوري تر څو دا باوري کړي چې د فنکشن په تعریف کې له وضع شویو ډېټا ډولونو سره سمون خوري او که نه. دا ستونزه په پروګرام کې د لوی شمېر داسې ټېروتنو لامل ګرځي چې موندل یې ستونزمن وي. خو د امریکې د ملي معیارونو اداري له خوا معیاري سي ژبه چې په پېچلو شرطونو ولاړه ده، په تالیفوونکي کې د ډول-کتنې د ځانګړنې دا ستونزه تر ډېره له منځه وړې. تقریباً په همدې وخت کې،

[1] Application Programming Interface
[2] definitions
[3] cryptic coding

سي پلس پلس د سي ژبې د يوې زياتونې[1] او سمونې په توګه يوه استازواله ژبه وګرځېده. په داسې حال کې چې سي ژبې خپل ځای د نوو پروګرامونو د جوړولو لپاره سي پلس پلس او جاوا ته پرې ايښی، خو د سي پروګرام ليکونکو لپاره د نوو ژبو زده کړه نبستاً اسانه وي او همدا رنګه د سي ژبې د کوډ پراخه ميراث به تر راتلونکو کلونو پورې خپل استعمال له لاسه ور نه کړي.

## اضافي لوست:

1-m Kernighan, B. W., and D. M. Ritchie. *The C Programming Language*, 2nd ed. Upper Saddle River, N.J.: Prentice-Hall, 1988.

2- Prata, Stephen. *C Primer Plus.* 5th ed. Indianapolis: SAMS, 2004.
Ritchie, D. M. “The Development of the C Language,” in *History of Programming Languages II*, ed. T. J. Bergin and R. G. Gibson, 678–698. Reading, Mass.: Addison-Wesley, 1995

***

[1] extension

# (۷)

## سي شارپ (C#)

په ۲۰۰۲ز کې منځته راغلې ژبه C# چې په سي شارپ تلفظ کېږي، سي پلس پلس ته ورته پروګرامي ژبه ده خو له ښلا ښلو اړخونو څخه ساده شوې او د مایکروسافټ له اوسني پروګرامي درېځ سره د کارېدو لپاره خاصه شوې ژبه ده. سي شارپ یوه عام-هدفه او په بشپړ ډول استازواله ژبه ده – چې ټول فنکشنونه باید د یوه کلاس یا سټرکټ د غړو په توګه تعریف شوي وي، تر دې چې اساسي ډېټا ټایپونه هم له System.Object کلاس څخه مشتق کېږي.

د سي پلس پلس په پرتله، په سي شارپ کې د ډېټا ډولونو د کارونې او اړونې[1] لپاره سخت قوانین وضع شوي دي، چې ډېرې مبهمو اوړېدنو[2] ته اجازه نه ورکوي (د ښلګې په ډول له عددي ډول څخه خپل استازي integer ته). د سي پلس پلس پر خلاف، سي شارپ کښمېر توارث ته اجازه نه ورکوي (چېرته چې یو ډول له دوو یا زیاتو اساسي ډولونو څخه مشتق کېږي)، او په دې ډول په لویو پوستغالي پروژو کې د کلاس د اړیکو په پېچلتیا کې د یوه اضافي پوړ څخه

[1] Conversion
[2] implicit conversion

مخنیوي کوي. (که څه هم، د ګڼشمېر اړیکځایونو[1] په تعریفولو او یا یوه کلاس ته د لاسرسي د بېلابېلو طریقو په مټ کولای شو چې دا کار ترسره کړو.)

د جاوا برعکس (خو د سي پلس پلس په څېر)، سي شارپ له نښه کوونکو (او په ډېر خوندي ډول چې delegates بلل کېږي)، عددي ډېټا ډولونو[2]، ستړکټونو (چې کوچنیو کلاسونو ته ورته دي)، او امبارونې[3] (د عاملونو لپاره ګڼشمېر تعریفونه) څخه برخمنه ژبه ده. د نومورې ژبې نوې بڼه، C# 3.0 (چې په ۲۰۰۷ کې منځته راغله)، د لیست پروسېس او فنکشنل پروګرام لیکنې لپاره اضافي ځانګړتیاوې هم چمتو کوي.

د «Hello World!» معیاري پروګرام په سي شارپ ژبه کې دا ډول برېښي:

```
using System;
// A "Hello World!" program in C#
namespace HelloWorld
{
class Hello
{
static void Main()
{
System.Console.WriteLine("Hello World!");
}
}
```

[1] Multiple Interfaces
[2] enum types
[3] overloading

```
}
```

په اصل کې د پروګرام ټول رغښتونه باید د یوه کلاس برخه اوسي. لومړنی عبارت په System کلاس کې پیل کېږي، او له همدې څخه بیا د اړیکځای اساسي میتودونه را مشتق کېږي. یو پروګرام کولای شي چې یو یا زیات نومځایونه[1] ولري، چې د کلاسونو او جوړښتونو د منظم کولو لپاره کارېږي ترڅو د کډوډۍ مخه ونیول شي. پورتنی پروګرام یوازې یو کلاس (Hello) لري، چې په هغه کې یو مېن فنکشن دی چې هر پروګرام باید یو دانه ولري او یوازې یو. دا فنکشن د System نومي کلاس، Console غړی رابولي، او له هغه وروسته د متن د ښودلو لپاره له WriteLine میتود څخه ګټه اخلي.

## سي پلس پلس او د مایکروسافټ پرمختګ

سي شارپ د سي پلس پلس، جې شارپ [چې د جاوا یوه بدیله نسخه ده]، او ویژوال بېسیک ډاټ نېټ[2] په ګډون د ژبو د کورنۍ، یوه برخه ده. دا ټولې ژبې یوې عمومي منځنۍ ژبې یا IL[3] ته تالیف کېږي. د کلاس عمومي چارچوکاټ[4]

[1] Namespaces
[2] Visual Basic.Net
[3] Intermediate Language
[4] framework

مایکروسافټ ډاټ نټ، د ویندوز پروګرامليکنې او د عصري وېب جوړونې[1] لپاره د نورو پخوانیو چارچوکاټونو ځای نیولی دی.

که څه هم په ابتدایي ډول یاده ژبه د مایکروسافټ په سافټوېرونو او ویندوز پورې تړلې وه، خو د مونو او «ډاټ جي. اېن. یو.» پروژې هم سي شارپ وړاندې کوي، او دغه راز د لینکس او یونیکس چاپېریال لپاره د CLI یا عامه ژبې په پېښبنسټ[2] او د ډاټ نټ په زیاترو (نه ټولو) لایبرېریانو کې هم سي شارپ پلي کېږي.

**اضافي لوست:**

1- “The C# Language.” MSDN. Available online. URL: http://msdn2. microsoft.com/en-us/vcsharp/aa336809.aspx. Accessed April 28. 2007.

2- Davis, Stephen Randy. *C# for Dummies.* New York: Hungry Minds, 2002.

3- Hejlsberg, Andres, Scott Wiltamuth, and Peter Golde. *The C# Programming Language.* 2nd ed. Upper Saddle River, N.J.: Addison-Wesley, 2006.

***

[1] Web-development

[2] Common Language Infrastucture

# (۸)

## سي پلس پلس (C++)

سي پلس پلس په مرې هیل، نیو جرسي کې په AT&T لابراتوار کې د بیارنې سټراؤسټرپ[1] له خوا ډیزاین شوه چې په ۱۹۷۹ز کې یې کار ورباندې پیل کړی وو. تر دې مهاله سي ژبه د سیستمي پروګراملیکنې لپاره د یوې ځواکمنې وسیلې په توګه کارول کېده. که څه هم سټراؤسټرپ (او نور) په دې باور وو چې د سي محدود ډېټا جوړښتونه او د فنکشنو میکانیزم په لویو سافټویرې پروژو کې چې د استازو[2] ترمنځ یې اړیکې پیچلې وي، د خپلمنځي اړیکو د څرګندولو لپاره د سي ژبه ناکافي وه.

د یوه ساده استازي مثال ته وګورئ: هغه ډېرۍ[3] چې پر هغه نور عددونه هم ور اینبودل کېدای او یا له هغه څخه اخیستل کېدای شي، په سي ژبه کې یوه ډېرۍ باید د یوه سټرکټ په توګه پلي شي ترڅو د ډېرۍ ډېټا او نښه کوونکی سمبال کړي، او یوه ډله فنکشونه په جلا ډول تعریف شي چې د ډېرۍ ډېټا جوړښت ته لاسرسی وکولای شي، تر څو ډېرۍ ته یو عدد ور اضافه او یا د ډېرۍ له سر څخه یو عدد پورته کړای شي. په دې طرحه کې د استازي د ډېټا او فنکشن

---

1 Bjarne Stroustrup
2 Objects
3 stack

ترمنځ هېڅ ډول مستقیمه او د اجرا وړ اړیکه نه لیدل کېږي. دا په دې معنا ده چې د یوه پروګرام ځینې برخې د یوه استازي په داخلي جوړښت باندې یا تکیه کولای شي او یا په مستقیمه توګه دا ډول داخلي ډېټا ته لاسرسی او تغیر پکې راوستلای شي. په لویو سافټویري پروژو کې چې ډېر پروګرام لیکونکي په کوډ باندې کار کوي، نو دا ډول حالت یو څه ګډوډي ته بلنه ورکوي.

د دې لپاره یوه بدیله تګلاره هم وه چې په ځینو نوو ژبو کې وړاندې شوې وه. دغه ژبې دا اجازه ورکوي چې ډېټا او فنکشن دواړه یو ځای د استازو (یا کلاسونو) په بڼه جوړښت بندي شي. د سي ژبې د یوه سټرکټ پر خلاف، یو کلاس کولای شي چې د یوه استازي د تعریفولو لپاره اړینه ډېټا او پر هغې ډېټا باندې د عملیو د ترسره کولو لپاره فنکشنونه دواړه یو ځای ولري. یو کلاس خپله شخصي ډېټا «پوښي» او ساتنه یې کوي، او له پاتې پروګرام سره د هغو بلنو[1] په وسیله اړیکه نیسي چې تعریف شوو فنکشنونو ته شوې وي.

پر دې سربېره، په استازواله ژبو کې، د توارث اصول له ډېر عمومي او انتزاعي استازي څخه ځانګړو نسخو ته ادامه کولای شي چې د ځانګړو کارونو لپاره مناسب وي، په داسې حال کې چې هر استازي به خپل عمومي مشخصات هم ساتلي وي، بیاکتنه هم پرې کېدای شي او یا زیاتونه او کمونه هم پکې کېدای

[1] calls

شي. په دې اساس، د يوه عمومي کلاس له لست څخه ډېر ډوله ځانګړي لستونه مشتق کېدای شي (لکه غبرګ-تړلی لست[1]).

په داسې حال کې چې د استازواله تګلارې ګټو خپل لوري ته ور مايل کړی وو، سټراؤسټرپ غوښتل چې په څرګند ډول د ماشين د چلند کنټرولولو ځانګړنه چې سي ژبې لرله، هغه په خپل ځای پاتې شي چې په سيسټمواله پروګرام ليکنه کې پکارېږي. په دې اساس، نوموړي غوښتل چې يوه داسې ژبه جوړه کړي چې نحوي ترکيب يې سي ته ورته وي او د استازوالو ژبو ځانګړنې هم ورسره وي. سټراؤسټرپ د انګلستان په کېمبرج پوهنتون کې د دوکتورا د تيزس په توګه د سي پلس پلس لومړنۍ نسخه وليکله چې ( C له کلاسونو سره[2]) نومېدله. همدا ژبه د ۱۹۸۰ کلونو په شاوخوا کې په C++ بدله شوه.

## د سي پلس پلس ځانګړنې

د سي پلس پلس اساسي جوړښتي بلاګونه، کلاسونه دي. کلاس د دې لپاره استفاده کېږي څو د خپل ډول او نوعيت استازي جوړ کړي. هر استازی د ډېټا يوه ټولګه لري او کولای شي چې ټاکلي فنکشنونه هم ترسره کړي چې کله د پروګرام له خوا وبلل شي. د مثال په ډول، لاندې کلاس د اعدادو يو کتار تعريفوي

[1] doubly-linked list
[2] C with classes

او ورسره پر دې کتار باندې د عملیو ترسره کولو لپاره فنکشنونه هم تعریفوي. عموما، دا ټول په یوه سرلیک فایل کې ځای پر ځای کېږي (لکه stack.h):

```
const int Max_size=20; // maximum elements
     in Stack
class Stack { // Declare the Stack class
public: // These functions are available
     outside
Stack(); // Constructor to create Stack
     objects
void push (int); // push int on Stack
int pop(); // remove top element
private: // This data can only be used in
     class
int index;
int Data[Max_size];
};
```

په دې پسې، د stack د کلاس غړي فنکشنونه تعریف کېږي. د فنکشنونو تعریفونه کولای شو چې د Stack.cpp په نوم یوه سرچینه فایل کې ځای پر ځای کړو:

```
#include "Stack.h" // bring in the
declarations
Stack::Stack() { index=0;} // set zero for
new stack
void Stack::push (int item) { // put a
number
on stack
Data[index++] = item;
```

```
}
int Stack::pop(){ // remove top number
return Data [index-];
}
```

اوس یو دوهم (Stacktest.cpp) سرچینه فایل هم لیکل کېدای شي چې په دې فایل کې به یو مېن فنکشن وي چې د Stack استازی جوړوي او د کلاس فنکشنونه کوري:

```
#include "Stack.cpp" // include the Stack
     class
#include <iostream.h> // include standard
I/O
     library
main() {
Stack S; // Create a Stack object called S
int index;
for (index = 1; index <= 5; index++)
     S.push(index); // put numbers 1-5 on
stack
for (index = 1; index <=5; index++)
     cout < S.pop(); // print the stack
}
```

د ډېرې پلي کول د هر هغه پروګرام له کوډ څخه په جلا ډول ترسره کېږي چې د ډېرې له استازو څخه ګټه اخلي. په دې اساس، پروګرام لیکونکی کولای شي چې د ډېرې کلاس بیا راوکوري (بنایي د یوه پرمختللي الګوریتم په کارولو او یا عمومیت ورکولو تر څو له مختلفو ډېټا ډولونو سره کار وکولای شي). تر څو پورې

چې د غړو فنکشنونو لپاره د اړتیا وړ پارامترونه نه وي بدل شوي، هغه پروګرامونه چې د ډیرې له استازو څخه ګټه اخلي،اړتیا به نه وي چې بدل شي.
د کلاسونو او توارث ترڅنګ، سي پلس پلس ځینې نورې مهمې ځانګړتیاوې هم لري. د فنکشن د پارامترونو لپاره ډېټا ډولونه په بشپړ ډول تعریفېدای شي، او د ډېټا ډولونه په خپلواک ډول کتل کېږي (که څه هم پروګرام لیکونکی کولای شي چې دا ډول کتنه ودروي که چېرته په ربنتیا غواړي او یا اړتیا ورته لري). د خاصو عامل-فنکشونو[1] د تعریفولو په مټ کولای شو چې یوه کلاس ته نوي عاملونه هم ور اضافه کړو، او کله چې مختلف ډېټا ډولونه تر کار لاندې وي نو یو ډول عاملونو ته مختلفې معناوې هم ورکول کېدای شي (دې ته امبارونه یا overloading ویل کېږي). په دې ترتیب، د + عامل د string له کلاس سره تعریف کېدای شي تر څو دوه متنکربنې سره یو ځای کړي. خو که همدا عامل له عددي ډېټا سره وکارول شي، نو بیا به د جمعي معنا ورکوي.
یو انتزاعي[2] استازی (هغه چې اطلاق ونه لري) د مجازي[3] فنکشنونو لپاره د اساس په توګه کارېدلای شي. دغه فنکشنونه په هر مشتق شوي استازي کې بیاځلې تعریف کېدای شي څو هر کله چې د هغه نوعیت یو استازی را وړاندې شي، نو تالیفوونکی په خپلواک ډول له پلرني کلاس څخه لاندې لور ته په اشتقاقي کلاسونو کې لټون پیل کړي.

---

[1] operator function
[2] abstract
[3] Virtual

د سي پلس پلس راوسته بڼې د template په نوم یو ټرلی مفهوم لري. ټمپلېټ هغه انتزاعي خصوصیت ده چې د هغو ډېټا ډولونو لپاره د کلاس تعریفونو د جوړولو لپاره کارېدای شي چې دې کلاس ته ور ټپرېږي. په دې اساس، یوه لیست ټمپلېټ ته یو وکټور او دوه-بعدي کتار ور ټپرېدای شي او دا ټمپلېټ به د هر ډېټا ډول لپاره د لیست کلاس تعریفونه جوړ کړي. ټمپلېټونه عموما هغه مهال کارېږي چې کله د ډېټا ډولونو ترمنځ د هایرارکي توارث هېڅ ډول اړیکه شتون ونه لري (په دې صورت کې مجازي پلرنی کلاس یوه غوره تکلاره ده).

سي پلس پلس د معیاري لایبرېریانو لپاره استازواله بدیلونه هم وړاندې کوي. د مثال په ډول، راکړېز/ورکړېز یو اصلي ماډل[1] کاروي، او د I/O عاملونه د دې لپاره چې له نوو کلاسونو سره کار وکړي، امبارېدلای هم شي. د دې ترڅنګ، په نوموړې ژبه کې د مناسبو استازو د کارونې له لارې د ټپروتنې د سمالښت پرمختللی میکانیزم هم شته.

## د سي پلس پلس پراختیا

د ۱۹۸۰ او ۱۹۹۰ کلونو په شاوخوا کې، سي پلس پلس له سیسټمي پروګرامليکنې څخه نیولې بیا تر کاروباري کاریالونو او ټیمونو پورې د مختلف ډوله سافټوېرونو لپاره یوه نومیالۍ ژبه وګرځېده. د یادې ژبې وده د ځواکمنو شخصي کمپیوټرونو له پرمختګ او ورسره د مایکروسافټ، بورلېنډ او نورو له خوا د ارزانه، اسانه کارېدونکي خو د جوړونې د ځواکمنو چاپېریالونو له منځته راتګ

[1] stream model

سره هم مهاله وه. څنګه چې دې تالیفوونکو د دودیزې سي ژبې کوډ هم سمبالولای شوی، پروګرام لیکونکو کولای شوی چې شته کوډ واخلي او د استازواله تخنیکونو له زده کولو وروسته له هغو ځنې ګټه پورته کړي. د ۱۹۹۰ په وروستیو کلونو کې، که څه هم سي پلس پلس په ډېرو ساحو کې یوه مشهوره ژبه وه، خو د جاوا له خوا وننګول شوه، چې د سي پلس پلس ځینې ډېرې پېچلې ځانګړنې یې ساده کړې او په ځانګړي ډول د هغو پوستغالو د لیکلو لپاره ډیزاین شوې وه چې په وېب سروونو او پرانیستونکو باندې ځغلېدل. د سافټوېر جوړونې د یوې بدیل ټکلارې په توګه د سي ډوله ژبې د لا منظمې بڼې لپاره «سي شارپ» وګورئ.

## اضافي لوست:


1- “C++ Archive.” Available online. URL: http://www.austinlinks.com/CPlusPlus/. Accessed May 24, 2007.

2- “Complete C++ Language Tutorial.” Available online. URL: http://www.cplusplus.com/_doc/tutorial/. Accessed May 24, 2007.

3- Prata, Stephen. *C++ Primer Plus.* 5th ed. Indianapolis: SAMS, 2004.

4- Stroustrup, Bjarne. “A History of C++: 1979–1991.” In *History of Programming Languages II,* edited by Thomas J. Bergin, Jr., and Richard G. Gibson, Jr. New York: ACM Press; Reading, Mass.: Addison-Wesley, 1996, 699–755.

5- ———. *The C++ Programming Language.* Special 3rd ed. Reading, Mass.: Addison-Wesley, 2000.

***

# (۹)

## کوبال (COBOL)

عامه کاروباروالە ژبه[1] په ۱۹۵۹ز کې د دفاع وزارت په نوښت تر دې انګېزې لاندې جوړه شوه چې د کاروباري کاریالونو لپاره یوه داسې عمومي ژبه جوړه کړي چې له فایلونو څخه د ډېټا پر پروسېس تمرکز ولري. (اخر پوځ هم یو کاروبار وو، چې د ګودامونو کنټرول او محاسبوي اړتیاوې یې له لویو شرکتونو څخه هم زیاتې او لوړې وې. هغه مهال، د مېنفریم کمپیوټرونو لپاره اساسي کاروباروالە ژبه فلومېټیک[2] وه، نوموړې ژبه د ګرېس هوپر د ډلې له خوا په ریمینګټن رینډ یونیواک کې جوړه او د همدې کمپنۍ تر کمپیوټرونو پورې محدوده وه. د کوبال لومړنۍ تالیفوونکي په ۱۹۶۸ کې منځته راغلل او د امریکې د ملي معیارونو اداري ANSI په ۱۹۶۸ کې د دې ژبې لپاره معیاري مشخصات وضع کړل. دا معیارونه په ۱۹۷۴ او ۱۹۸۵ کې لا زیات پراخه شول (COBOL-74 او COBOL-85) او په ۲۰۰۲ کال کې یې نوی معیار را وایست.

هغه ډله کسان چې په هغه ژبه یې کار کاوه چې وروسته کوبال ونومول شوه، د هغو تمرکز پر دې وو چې د پروګرام عبارتونه یې د انګلیسي ژبې له بیانیه جملو سره ورته والی ولري، نه دا چې د پروګرامي ژبې حالت ولري. د کوبال اوډونکرانو

[1] Common Business-Oriented Language
[2] Flow-Matic

هیله لرله چې محاسبین، مدیران او نور کاروباري مسلکیان به ډېر ژر دا ژبه زده کړي، چې د پروګرام لیکونکو اړتیا که ختمه نه کړي، نو کمه خو به یې خامخا کړي. («پروګراملیکنه له پروګرام لیکونکو پرته» پورتنۍ طرحه د RPG او BASIC په څېر ژبو او ځنې ډېټابېس سیسټمونو پر مهال هم په ذهنونو کې تازه شوه خو د تل په څېر یې محدوده بریا په برخه وه.)

## د پروګرام جوړښت

د کوبال پروګرام په ټوله کې هغه سوداګریز فورم ته ورته بڼه لري چې په ځانګړو برخو وېشل شوی وي، او هره برخه یې اختیاري او د اړتیا وړ توکونه لري.

د پېژندنې برخه[1] یوازې ګرامر مشخصوي او د پروګرام په هکله یو څه معلومات وړاندې کوي:

```
IDENTIFICATION DIVISION.
PROGRAM-ID WEEKLY REPORT.
AUTHOR JAMES BRADLEY.
DATE-WRITTEN DECEMBER 10, 2000.
DATE-COMPILED DECEMBER 12, 2000.
REMARKS THIS IS AN EXAMPLE PROGRAM.
```

د چاپېریال برخه[2] د چاپېریال (سختغالي) په تړاو هغه خصوصیات په ځان کې رانغاړي، د کومو لپاره چې پروګرام تالیف کېږي. په ځینو حالتونو کې (د مثال په

[1] identification division
[2] Environment Division

ډول، د کوچنیو کمپیوټرونو لپاره د کوبال په نسخو کې) کېدای شي دې برخې ته اړتیا نه وي. په نورو حالاتو کې، ښایي پروګرام د پیکر-بندي[1] برخه هم ولري چې کارېدونکی ماشین مشخص کړي.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER IBM-370.
OBJECT-COMPUTER IBM-370.
```

د سرچینه یې او هدفي دوو بېلو کمپیوټرونو د شتوالي علت دا دی چې کله نا کله به پروګرامونه په یوه کمپیوټر کې تالیف کېدل او کارول کېدل به په بل کمپیوټر کې، چې زیاتره به یې کوچني کمپیوټرونه وو.)

په ځینو حالاتو کې، د چاپېریال برخه د راکړېز/ورکړېز برخه هم باید ولري چې هغه وسایل او فایلونه مشخص کړي چې د پروګرام په وسیله به استفاده کېږي. د مثال په ډول:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT STUDENT-FILE ASSIGN TO READER
SELECT STUDENT-LISTING ASSIGN TO LOCALPRINTER
```

د ډېټا برخه د ډېټا ریکارډونو او نورو توکونو په اړه تفصیلات وړاندې کوي چې د پروګرام له خوا به پروسېس کېږي چې تر یو څه بریده کولای شو چې دا برخه د

[1] Configuration

پاسکال، سي او بېسیک په څېر ژبو کې د متحولینو له تعریفولو سره پرتله کړو. دا چې کوبال د فایلونو د ریکارډونو په پروسېس او د راپورونو په فارمټ باندې تمرکز کوي، په دې اساس په دې ژبه کې د نورو ژبو په پرتله د ډېټا ډولونو شمېر کم دی، خو د نوموړې ژبې دا ځانګړنه د هغو بېلا بېلو ډېټا جوړښتونو تعریفول یو څه اسانه کوي چې عموما په کاروباري کاریالونو کې استعمالېږي. د مثال په ډول، دا اسانه کار دی چې د درجه نمبرونو په کارولو هغه ریکاډونه تعریف کړای شي چې خپلې اصلي او فرعي ساحې ولري تر څو د هغو ترمنځ اړیکه په ګوته شي:

```
DATA DIVISION.
FILE SECTION.
FD INFILE
     LABEL RECORDS ARE OMITTED.
01 STUDENT-DATA.

02 STUDENT-ID PIC 999999.
02 STUDENT-NAME.
     03 LAST-NAME PIC X(15).
     03 INITIAL PIC X.
     03 FIRST-NAME PIC X(10).
02 GPA PIC 9.99
```

د Pic یا انځور عبارت د ډېټا ډول (د عددونو لپاره تر نهو اعداد او اعشاري ټکی؛ او د متن لپاره X) او اوږدوالی مشخصوي. د ورکړیز ریکارډونو د مشخصولو ترڅنګ، د ډېټا برخه ډېری وختونه هغه توکونه هم په ځان کې رانغاړي چې د هغه راکړیز د کړښو بڼه ټاکي چې د پایلې په توګه پرنټ کېږي.

د پروسې[1] برخه هغه عبارتونه چمتو کوي چې پر ډېټا اصلي عملیې ترسره کوي. پروسه کولای شو چې په فرعي پروګرامونو ترتیب کړو (چې تر یو څه بریده د نورو ژبو له پروسو او فنکشونو سره ورته والی لري). د پروسې د عبارتونو بېلګه په لاندې ډول ده:

```
READ STUDENT-DATA INTO STUDENT-WORK-RECORD
AT END MOVE 'E' TO PROC-FLAG-ST
GO TO EXIT-PRINT
ADD 1 TO TOTAL-STUDENT-RECORDS
```

ریاضیکي معادلات د compute عبارت په کارولو محاسبه کېدای شي:

```
COMPUTE GPA = TOTAL-GRADES / CLASSES
```

په کوبال ژبه کې څانګیز if[2] عبارتونه هم شتون لري، او څرخونه د Perform عبارت په وسیله ترسره کېږي. د مثال په ډول:

```
PERFORM 100-PRINT-LINE
UNTIL LINES-FL IS EQUAL TO 'E'
```

(د ښسیک د نورو نسخو په څېر، فرعي پروګرامونو ته شمېره هم ورکول کېږي.)

## اغېزاولرلید

له ۱۹۶۰ څخه تر ۱۹۸۰ کلونو پورې، کوبال په مېنفریم او منځنیو کمپیوټرونو کې د کاروباري پوستغالو د جوړولو لپاره یوه اساسي ژبه وګرځېده او تر نن ورځې

[1] procedure
[2] Branching if

پورې په پراخه توګه کارول کېږي. (د روانې پېړۍ په پای کې د هغو ممکنه ستونزو په هکله اندېښنې تر ډېره په هغو زړو پروګرامونو پورې اړوند وې چې په کوبال کې لیکل شوي وو). د پروګرامي ژبو د ودې کرښه له کوبال څخه واوښته او همداسې پر الګول (د کوبال همزمانه)، پاسکال، سي او نورو بلاک-جوړښته ژبو ورتېره شوه.

د کوبال ځینو اوسنیو نسخو د جوړښتواله پروګراملیکنې وروستي پرمختګونه هم په کې ځان کې راټول کړي (لکه modularization) او حتی استازواله ډیزاین. کوبال د مایکروسافټ ډاټ نېټ او ورسره د XML ډېټا د پروسېس کولو په ګډون د جوړونې د اوسنیو چارچوکاټونو د خپلولو لپاره هم د پام وړ تغیرات له ځانه سنبولي. خو بیا هم، د کوبال استعمال ورځ تر بله د ځوړتیا پر لور درومي، په دې چې پروګرام لیکونکو ورځ په ورځ خپل مخ د سي پلس پلس په څېر ژبو، متنواله ژبو، او د ډېټابېس جوړونې سیسټمونو خوا ته کړی دی.

## اضافي لوست:

1- Bivar de Oliveria, Rui. *The Power of COBOL: For Systems Developers of the 21st Century.* Charleston, S.C.: BookSurge, 2006.

2- COBOL Portal. Available online. URL: http://www.cobolportal.com. Accessed June 8, 2007.

3- Murcah, Mike, Anne Prince, and Raul Menendez. *Murach's Mainframe COBOL.* Fresno, Calif.: Murach and Associates, 2004.

4- Sammet, J. E. "The Early History of COBOL," in *History of Programming Languages.* Wexelblat, R. L., ed., 199–276. New York: Academic Press, 1985.

***

# (۱۰)

# آیفل (Eiffel)

آیفل یوه په زړه پورې ژبه ده چې د برټرانډ میېر او د هغه د کمپنۍ آیفل سافټویر له خوا په ۱۹۸۰ کلونو کې جوړه شوه. نوموړې ژبه د ګوستاف آیفل په نوم پسې ونومول شوه – هغه معمار چې په پاریس کې یې نومیالی منار ډیزاین کړ. نوموړې ژبې او د هغې میتودولوژي د پوستغال انجینرۍ په کنفرانسونو کې د پام وړ توجه را جلب کړه.

آیفل د پروګراملیکنې د هغو ټولو مفاهیمو (او په ځینو برخو کې یې لا سرلاړې ده) ملاتړ کوي چې د نن ورځ په نوموتو او ډېر کارېدونکو ژبو کې تر سترګو کېږي. د نحوي ترکیب له مخې، آیفل په ساده والي او بیا کارېدونکو تعریفونو چې په پروګرام پوهېدل اسانه کوي، باندې ټینګار کوي او هڅه کوي چې له ټیټې کچې او مبهم کوډ لکه د تالیفوونکي آپټیمایزېشن څخه مخنیوی وکړي.

## د پروګرام جوړښت

په آیفل ژبه کې لیکل شوی پروګرام سیستم بلل کېږي، چې د هغو کلاسونو د ټولګې په جوړښت ټینګار کوي چې د رښتیني نړۍ د ډېټا د پروسېس کولو لپاره پکارېږي. یو ساده کلاس یې ښایي داسې جوړښت ولري:

```
class
```

```
     COUNTER
feature–access counter value
     total: INTEGER
feature–manipulate counter value
     increment is–increase counter by one
     do
          total :- total + 1
     end
decrement is–decrease counter by one
     do
          total := total - 1
     end
reset is–reset counter to zero
     do
          total := 0
     end
end
```

په دې لیستواله ژبه[1] کې، اړین-ویوکي په پرېږ او د کټه اخیستونکي له خوا تعریف شوي استازي په کاږه لیک لیکل شوي. کله چې کټه اخیستونکی متن درج کړي، نو یاد فارمټ په خپلواک ډول صورت نیسي. کله چې یو کلاس تعریف شي، له هغه وروسته د استازي جوړول ډېر اسانه وي:

```
my_counter COUNTER
create my_counter
```

[1] Listing language

د آیفل تالیفوونکی یو کوډ، خپلې منځنۍ ژبې یعنې بایټ کوډ ته تالیف کوي، چې په وروستي پړاو کې سي ژبې ته تالیف کېږي، او په دې ځای کې د سي له آپټیمایز شوو تالیفوونکو له وړاندې څخه د شتوالي ګټه پورته کوي.

د آیفل یوه خاصه ځانګړنه دا ده چې قراردادونه منظم کولای شي چې دا مشخصوي چې کلاسونه به په څه ډول له یو بل سره اړیکه ټینګوي. (همدا کار د پارامترونو او ډېټا ډولونو د تعریفولو په برخه کې هم په ښه ډول پلي کېږي.) د مثال په ډول، د Counter کلاس سره یو ثابت هم تعریفولای شو لکه total >=0. په دې معنا چې له هر ډول توپیر او بدلون پرته، دا شرط باید د تل لپاره صحیح پاتې شي. کله چې هغه فنکشن پروسېس شي او خپل بلونکي ته ډېټا واپس کړي، نو دا باوري کولای شي چې یو خاص شرط صحیح دی. د دې مشخصاتو مهم ټکی دا دی چې د کوډ له هر واحد څخه چې څه ترلاسه کېږي او څه چې واپس کوي، هغه واضح کوي. او په دې ډول د پروګرام لاربښودلیکنې[1] ته هم وده ورکولای شي.

## تطبیق او استعمال

د آیفل ملاتړ کوونکي وایي چې نوموړې ژبه له یوې ژبې څخه ډېره ده: نوموړې ژبه د دې لپاره ډیزاین شوې تر څو د سافټوېر جوړونې په ټول جریان کې د پروګرام بېلا بېلې برخې بیاځلې وکتل او وکارول شي. د آیفل ژبې اوسنی تطبیق په ټولو دریځونو کې موجود دی او د سي، سي پلس لس او نورو ژبو لپاره هم

[1] documentation

اړیکځایونه لري. دا ځانګړنه آیفل ژبې ته اجازه ورکوي چې یو داسې چارچوکاټ ډیزاین او جوړ کړي چې د سافټویر بېلابېلې شته برخې په نورو ژبو کې وکارول شي. همدا رنګه، د آیفل ثابت استازواله ډیزاین هم دا ژبه د سافټویري پروژو د ډیزاین او لاربنودلیکنې لپاره ګټوره ثابتوي.

آیفل له سي پلس پلس ژبې سره په ورته مهال کې رامنځته شوه. که څه هم، آیفل له شک پرته د سي پلس پلس په پرتله په ډیزاین کې غوره او ستره ژبه ده. خو سي پلس پلس ته دوو ځانګړنو پر آیفل غوره والی ور پرخه کړی: د ارزانه او وړیا تالیفوونکو شتون او د هغو زرګونو پروګرام لیکونکو شتون چې له وړاندې څخه یې لا سي ژبه کارولې. آیفل د سافټویر د ډیزاین د تدریس لپاره د یوې نوې ژبې او د آیفل ستوډیو د پروګرامي چاپېریال په مرسته د محدود شمېر سوداګریزو کاریالونو جوړولو لپاره وکارول شوه.

آیفل د استازواله سافټویر ډیزاین د جوړولو په ډګر کې د پراخې برخې اخیستنې په سبب پرېمانه شهرت لري او په دې وروستیو کې د کمپیوټري ماشینرۍ د ټولنې له خوا د سافټویر پر کیفیت باندې د اغېز جایزه چې د ۲۰۰۶ د سافټویر سیستم جایزه نومېږي، هم آیفل ژبې ته ورکول شوه.

## اضافي لوست:

1- “Eiffel in a Nutshell.” Available online. URL: http://archive.eiffel.com/eiffel/nutshell.html. Accessed September 19, 2007.

2- Eiffel Zone. Available online. URL: http://eiffelzone.com/. Accessed

September 19, 2007.

3- “Introduction to Eiffel” [Flash presentation]. Available online. URL: http://www.eiffel.com/developers/presentations/eiffel_introduction/player.html?slide=. Accessed September 19, 2007.

4- Meyer, Bertrand. *Eiffel: The Language.* Englewood Cliffs, N.J.: Prentice Hall, 1991.

5- ———. *Object-Oriented Software Construction.* 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2000.

6- Wiener, Richard. *An Object-Oriented Introduction to Computer Science Using Eiffel.* Upper Saddle River, N.J.: Prentice Hall, 1996.

***

# (۱۱)

## فورت (Forth)

فورت ژبه ډېر ساده جوړښت لري. د فورت سیستم د کلماتو یوه ټولګه لري. هره کلمه د عملیاتو یوه لړۍ ده (چې نور کلمات هم پکې موجود وي). د مثال په ډول، DUP کلمه د ډېټا وېلیو یوه کاپي جوړوي. په دې ژبه کې ډېټا په یوه ډېرۍ کې ځای پر ځای کېږي. د مثال په ډول، د ریاضي یوه معادله چې په ډېرو ژبو کې 3 + 2 لیکل کېږي، په فورت کې 3 2 + لیکل کېږي. کله چې د + عامل (چې په فورت ژبه کې له وړاندې څخه تعریف شوې کلمه ده) اجرا کېږي، نو هغه دوه عددونه سره جمع کوي چې ترڅنګ یې وي، او بیا یې جمع په یوه ډېرۍ کې ځای پر ځای کوي (چېرته چې په خپل نوبت همدا نتیجه په پروګرام کې بلې کلمې ته د راتلونکي پروسېس لپاره وړاندې کېږي.) دا ډول ښودنه[1]، وروستاړیزه ښودنه[2] بلل کېږي او د ساینټیفیک ماشین-حساب ډېرى ګټه اخیستونکي ورسره بلد دي.

په قاموس کې ټول کلمات له یو بل سره ټرلي او نښتي وي داسې چې هره کلمه د راتلونکې کلمې ادرس په ځان کې خوندي کړی وي. د فورت ژباړونکی یو ساده څرخ ځغلوي چېرته چې ژباړونکی، راتلونکی ټوکن (یو یا ډېر توري چې

---

1

[2] Postfix Notation

فاصله ونه لري) را پورته کوي او بيا ډکشنري لولي تر څو وګوري چې آيا نومورې کلمه له کومې تعريف شوې کلمې يا متحول سره سر خوري او که نه. که چېرته يوه کلمه ومومندل شي، نو په کوډ کې شته کلمه اجرا کېږي. که چېرته کلمه ونه مومندل شي، نو ژباړونکی هغه ټوکن يوه عددي ثابت ته ژباړي، ډېرې ته يې ور پورته کوي، او بيا راتلونکي لغت ته ور درومي.

د فورت يوه کليدي ځانګړنه د دې ژبې د پراختيامونديې خصوصيت دی. کله مو چې يوه کلمه تعريف کړه، نو نوې کلمه هم کټ مټ په هماغه ډول کارول کېدای شي لکه له وړاندې څخه تعريف شوې کلمه چې کارېږي. د کلمو د تعريفولو لپاره ګڼ حالتونه د دې اجازه ورکوي چې پر دې باندې کنټرول ولري چې څه وخت يوه نوې کلمه جوړېږي او يا وروسته هغه کلمه اجرا کېږي. (په ډېرو برخو کې فورت د استازواله پروګراملیکنې اصول په ځان کې را ټول کړي دي چې کنسټرکټونه او ميتودونه لري او کلمات يې د استازو په توګه کار ورکوي. د فورت يو منظم پروګرام د اوليه عملياتو او عالي ژبو له کلماتو څخه جوړ وي، چې پخپله پروګرام هم د عالي ژبو يوه کلمه ده.)

فورت تل د هغو لېواله پروګرام ليکونکو پام ځان ته اړولی وو چې په ماشين کې د ډېټا له جريان سره له نږدې اړيکې او په څرګند ډول پروګرام سره د کار کولو وړتيا ته په ښه سترګه ګوري. نومورې ژبه په بشپړ ډول تعاملي ژبه ده، ځکه هره کلمه په کيبورډ ليکل کېدای شي، تر څو اجرا شي او پايله يې په صفحه ښکاره کړي.

فورت د کوچنیو کمپیوټرونو په ورځو کې هم یوه راښکونکې ژبه وه ځکه چې فورت کوم پیچلي ژباړونکي یا تالیفوونکي ته اړتیا نه لرله، او دا په دې معنا وه چې فورت سیسټمونه په هغو سیسټمونو په ارامتیا ځغلېدای شي چې د ناڅاپي-لاسرسي زېرمې ظرفیت به یې له ۱۶ کیلو بایټ څخه ۶۴ کیلو بایټه وو.

فورت هېڅکله د پروګرام لیکونکو لپاره یوه عمومي ژبه نه شوه. له نورو ژبو د یادې ژبې ښلوالی او غیرعادي ذهني چوکاټ د هغو کسانو شمېر محدوداوه چې د دې ژبې زده کړې ته یې زړه ښه کاوه. په داسې حال کې چې د فورت پروګرامونه په څرګند ډول ترتیب کېدای شي، خو هغه پروګرامونه چې په ناوړه ډول لیکل شوي وي، د هغو لوستل شاوخوا ناممکنه وي. که څه هم، ځینې وختونه فورت په ځینو برخو کې لوړ دریځ خپل کړی (لکه، پسمتنواله[1] ژبه فورت ته ورته ده) او نوموړې ژبه تر اوسه هم د هارډوېر د کنټرولوونکو وسایلو په ډیزاین کې د پام وړ پلویان لري.

## اضافي لوست:

1- Brodie, L. *Starting FORTH.* 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1987.

2- ———. *Thinking FORTH.* 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1994.

[1] Postscripting

3- Forth Interest Group. Available online. URL: http://www.forth.org. Accessed August 14, 2007.

4- Rather, Elizabeth D. *Forth Application Techniques.* Hawthorne, Calif.: FORTH, Inc., 2006.

***

# (۱۲)

## فورټران (Fortran)

کله چې د ۱۹۵۰ کلونو په شاوخوا کې کمپیوټرونه منځته راغلل، نو د داسې یوې ژبې اړتیا هم احساس شوه چې د کمپیوټر کړنې په داسې ژبه کې څرګندې کړای شي چې د انسان لپاره د لوستو وړ وي. په یوه عالي ژبه کې، پروګرام لیکونکي متحولونه تعریفوي، او پر هغو باندې د عملیو د ترسره کولو لپاره عبارتونه او فنکشنونه لیکي. په دې ډول پروګرام لیکونکی نور په دې اړه اندېښنه نه کوي تر څو په مفصل ډول د کمپیوټر زېرمه ورته مشخصه کړي او له کمپیوټر څخه بایزري ډېټا ېرته ترلاسه کړي؛ دغه راز د پروګرام د جوړښت او د سمو الګوریتمونو له تطبیق څخه هم ېغمه وي.

فورټران (FORmula TRANslator) لومړنۍ تر ټولو زیاته کارېدونکې عالي ژبه وه. فورټروان د آی بي اېم شرکت د یوه څېړونکي تر مشرۍ لاندې د یوې ډلې له خوا په هغه پروژه کې جوړه شوه چې په ۱۹۵۴ کې پیل شوې وه. د پروژې موخه دا وه چې یوه داسې ژبه جوړه کړي چې ریاضي پوهانو، ساینسپوهانو او انجینرانو ته دا اجازه ورکړي تر څو محاسبې تر یو څه بریده په دودیزه ښودنه کې بیان کړي. په همدې وخت کې باید یو تالیفوونکی هم په بشپړ احتیاط ډیزاین شوی وي، تر څو د اجرا وړ ماشینواله کوډ تولید کړي چې د هغه کوډ په اندازه اغېزمن وي چې د منځنۍ ژبې په وسیله د یوې نغښتې پروسې له لارې تولیدېږي.

د نوموړې ژبې لومړنۍ نسخه« فورټران-۱ » په ۱۹۵۷ کې د آی. بي. اېم د مېنفریم کمپیوټرونو لپاره د تالیفوونکو په توګه منځته راغله. بله پرمختللې نسخه یې (چې ټېروتنې یې هم نیولې)، ډېر ژر ورپسې راووتله. فورټروان-۴ (۱۹۶۳) د ډېټا ډولونو شمېر زیات کړ، «عامه» ډېټا زېرمتون ور زیات شوی، او د DATA عبارت پکې رامنځته شو، چې متحولینو ته د لیټرل[1] عددي وېلیوګانو ور پورته کول یې اسانه کړل. د فورټروان دې پرمختللې نسخې ته د زیاتره هغو ساینسپوهانو او انجینرانو له لورې هرکلی وشو، چې د هغو محاسبو لپاره یې د کوډ سترې لایبرېریانې جوړې کړې چې عموما د دوی په کار کې ورته اړتیا وه.

له ۱۹۷۰ کلونو وړاندې، د جوړښتواله پروګراملیکنې غورځنګ هم په لاره وو. د پروګراملیکنې دې مکتب پر دې ټینګار کاوه چې پروګرامونه باید پر ځان بسیا پروسو ووېشل شي چې هغو ته ډېټا ور ټېرېدای، همالته پروسېس او بېرته واپس کېدای وشي. هغه ناشرطي څانګیز[2] عبارت (goto) چې په فورټران کې عام وو، له پامه وغورځېده. د فورټران ژبې یوه نوې نسخه، فورټران-۷۷ (یا F77)، ډېرې جوړښتواله ځانګړنې پخپله لمنه کې راپورته کړې. په بله نسخه فورټران-۹۰ (F90) کې د راګرځېدنې (recursion) فیچر اضافه شو. راګرځېدنه د ډول ډول ستونزو د کوډ کولو لپاره یو مهم تخنیک دی. د ریاضي اړوند لایبرېریانې هم له عصر سره برابرې شوې. فورټران ۲۰۰۳ د پروګراملیکنې د اوسنیو جوړښتونو

[1] Literal
[2] Unconditional branching statement

لپاره د ملاتړ او په سي ژبه کې د لیکل شوو پروګرامونو سره د هوسا تعامل کولو د وړتیا په کډون، یو شمېر نوي فیچرونه هم لري. فورټران ۲۰۰۸ تر یو څه بریده د لا پرمختللي نسخې لپاره لا هم یو تجربي نوم دی.

## د پروګرام بېلګه

لاندې ساده بېلګه د دودیز فورټران پروګرام ځینې ځانګړنې واضح کوي:

```
INTEGER INTARRAY(10)
INTEGER ITEMS, COUNTER, SUM, AVG
SUM = 0
READ *, ITEMS
DO 10 COUNTER = 1, ITEMS
     READ *, INTARRAY(COUNTER)
     SUM = SUM + INTARRAY(COUNTER)
10 CONTINUE
     AVG = SUM / ITEMS
     PRINT 'SUM OF ITEMS IS: ', SUM
     PRINT 'AVERAGE IS: ', AVG
STOP
END
```

پورتنی پروګرام یو داسې کتار جوړوي چې تر لسو پورې عددونه په ځان کې ذخیره کوي. لومړنی عدد چې دی یې لولي د هغو توکونو شمېر دی چې ور زیاتېږي. او دی دا شمېر په ITEM متحول کې زېرمه کوي. د DO څرخ، راتلونکي دوه عبارتونه د هر عدد لپاره له ۱ څخه د توکونو تر شمېر پورې یو ځل تکراروي. دا عبارتونه له کتار څخه راتلونکي عددونه لولي او بیا یې په SUM کې

ور جمع کوي. په پای کې، اوسط محاسبه کېږي، او د جمع حاصل او اوسط يې پرنټ کېږي.

د خپلې همعصره، کوبال ژبې، په څېر فورټران هم د زياتره اوسنيو پروګرام ليکونکو له لوري د يوې ګډې وډې او مهالټېره[1] ژبې په توګه پېژندل کېږي (او د دې لامل د مثال په ډول، د ريفرنس کړنبنه-ګنيو کارول دي.) خو بيا هم، فورټران د ازمايل شوو او باوري کوډونو له يوې پراخه زېرمې او د رياضي له ځواکمنو لايبرېريانو څخه برخمنه ژبه ده. ( د مثال په ډول، د فورټروان يو پروګرام کولای شي چې د لايبرېري روټينونو ته بلنه ورکړي تر څو په چټکه توګه د جمع حاصل او يا د مټريکس يا کوم کتار د طرفينو د ضرب حاصل ترلاسه کړي). دا ځانګړنې اوس هم دا باور لېږدولای شي چې فورټران لا هم د هغو ګټه اخيستونکو لپاره هغه انګېزه او ګټورتيا په ځان کې لري چې ډېره اندېښنه يې د پروګراملیکنې د بنايسته سبک پر ځای د چټکو او دقيقو پايلو ترلاسه کولو ته وي.

## اضافي لوست:

1- Chapman, Stephen J. *Fortran 95/2003 for Scientists & Engineers.* 3rd ed. New York: McGraw-Hill, 2007.

2- Chivers, Ian, and Jane Sleightholme. *Introduction to Programming with Fortran: With Coverage of Fortran 90, 95, 2003 and 77.*

[1] Anachronistic

New York: Springer, 2005.

3- Page, Clive. “Clive Page’s List of Fortran Resources.” Available online. URL: http://www.star.le.ac.uk/~cgp/fortran.html. Accessed August 4, 2007.

4- Reid, John. “The Future of Fortran.” Available online. URL: http://www.ieeexplore.ieee.org/iel5/5992/27213/01208645.pdf. Accessed August 4, 2007.

***

# (۱۳)

## جاوا (Java)

جاوا، د جوړښت له مخې سي پلس پلس ته ورته یوه کمپیوټري ژبه ده. که څه هم جاوا، یوه عام-هدفه پروګراموالە ژبه ده، خو تر ډېره د هغو کاریالونو د جوړولو لپاره کارېږي چې په انټرنېټ لکه وېب سرورونو باندې ځغلي. د جاوا د پروګرام یو ځانګړی ډول applet بلل کېږي چې له وېبپاڼو[1] سره وصل کېدای او د ګټې اخیستونکي په وېبپرانیستونکي[2] باندې ځغلول کېدای شي.

د یوې استازوالە ژبې په توګه، جاوا له هغو کلاسونو څخه ګټه اخلي چې د اړتیا وړ عادي فنکشنونه چمتو کوي، او له هغې ډلې څخه د اړیکځای د استازو[3] جوړول لکه د عامل سیستم کړکۍ[4] او بټنونه یادولی شو. د ډول ډول کلاسونو یوه ټولګه (د کلاس چارچوکاټونه) هم پکې شتون لري، لکه AWT یا وینډوزوالە انتزاعي ابزاربکس[5].

---

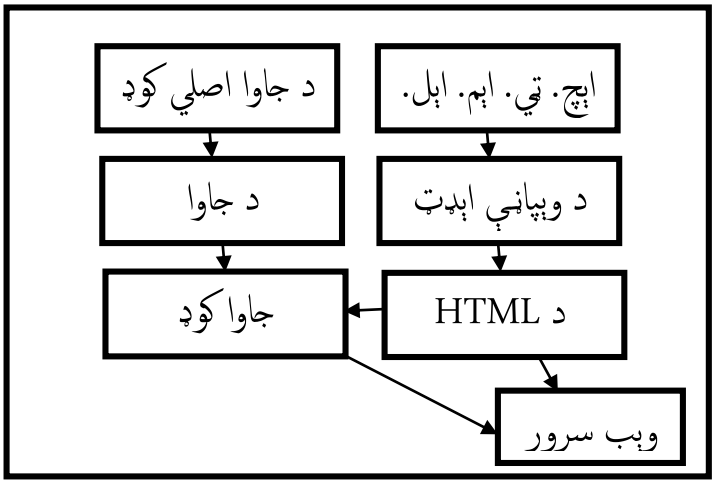
[1] Webpages
[2] Web browsers
[3] user interface object
[4] windows
[5] Abstract Windowing Toolkit

## د پروګرام جوړښت

د جاوا پروګرام د کلاسونو په راوړلو او بیا د هغو په وسیله د پروګرام د فنکشنونو لپاره د اړتیا وړ استازو په جوړولو پیل کېږي. ورپسې د کوډ عبارتونه مطلوب راکړیز تولیدوي او یا له استازو سره یې اړیکه پیلېږي. لکه د یوه انځور رسمول او یا په یوه کړکۍ کې د متن ځای په ځای کول. په دې ځای کې د جاوا اپلټ یوه ساده پروګرام ته تم کېږو:

```
import java.applet.Applet;
import java.awt.Graphics;
public class HelloWorld extends Applet {
  public void paint(Graphics g) {
    g.drawString("Hello world!", 50, 25);}}
```

لومړنۍ دوه کرښې پروګرام ته معیاري کلاسونه را کوزوي. اپلټ کلاس هغه بنسټ دی چې پر هغه اپلټ پروګرامونه بنا وي. وینډوزواله انټزاعي ابزاریکس د هغو کلاسونو ټولګه ده چې ګرافیکي اړیکځای (GUI) چمتو کوي.



کله چې د جاوا یو مختلط پروګرام (چې اپلټ بلل کېږي) کمپایل شي ، نو د هغه اجرایي فایل (جاوا کوډ) او د وېبپاڼې لپاره د HTML فایل چې له پروګرام سره وصل وي ، دواړه یو ځای په ګډه په وېب سرور کې زېرمه کېږي.

ورپسې پروګرام یو نوی کلاس Hello World تعریفوي او دا مشخصوي چې نوموړی کلاس د اپلټ کلاس نور هم غځوي او یا پر اپلټ کلاس بنا دی.

په دې پسې د میتود (د یو څه د کولو پروسه) تعریف راځي چې paint بلل کېږي. نوموړی میتود له ګرافیکي استازي څخه استفاده کوي چې په هغه کې د پردې پر مخ د شیانو د رسمولو لپاره ډول ډول وړتیاوې شتون لري. په پای کې، پروګرام د ګرافیکي استازو، تعریف کړی drawstring میتود کاروي تر څو یوه متنکرښه ورباندې رسم کړي.

د دې پروګرام د جوړولو لپاره، پروګرام لیکونکی نوموړی کوډ په جاوا کمپایلر کې تالیف کوي. ورپسې پروګرام لیکونکی د html پاڼه جوړوي چې یو ټېک لري چې دا مشخصوي چې دا کوډ به هغه مهال ځغلي چې کله د html لینک فعاله کېږي.

## د جاوا رامنځته کول

جاوا د جېمز ګوسلینګ (۱۹۵۵ - ) له لوري جوړه شوې. نوموړې ژبه په سن مایکروسیسم کمپنۍ کې د یوې کورنۍ پروژې په توګه پیل شوه څو یوه داسې ژبه

ډیزاین کړای شي چې د «ځیرکو» مصرفي وسایلو د پروګرام کولو لپاره کار وکړي، لکه ګړېدونکی ټلویزون. کله چې دا پروژه پرېښودل شوه، نو ګوسلینګ، بیل جوای، او نورو جوړوونکو احساس کړه چې نوموړې ژبه ورځ تر بله وده کوونکي انټرنېټ لپاره په خدمت کې ګمارل کېدای شي. د وېبپاڼو جوړوونکو یوې داسې اسانه لارې ته اړتیا لرله تر څو داسې پروګرامونه جوړ کړي چې هغه مهال وځغلي چې کله د ګټې اخیستونکي له لوري هغې وېبـپاڼې ته لاسرسی وشي. په وېبپاڼو باندې د ګټې اخیستونکو د کنټرول پلې کولو له لارې، اوډونکرانو د وېب ګټې اخیستونکو ته هغه وړتیا په لاس کې ورکولای شوی چې په آنلاین ډول کټ مټ هماغسې د ارتباط نیولو وړتیا ولري لکه څنګه یې چې د مېکنټاش او یا د شخصي کمپیوټرونو د وینډوز پر پرده له استازو (objects) سره ټینګوي.

## ګټې

جاوا، د وېب ډوبلپرانو لپاره خپله پورته یاده ژمنه په لویه کچه پوره کړې. د سي پلس پلس پروګرام لیکونکي په ډېر اسانه کولای شي چې جاوا ژبه زده کړي، ځکه چې دواړه ژبې ډېر زیات ورته نحوي ترکیب لري او په ورته ډول له کلاسونو او د استاوزاله پروګراملیکنې له ځانګړنو څخه ګټه اخلي. بل خوا، هغه پروګرام لیکونکي چې په سي پلس پلس نه پوهېږي، د هغو ګټه په دې کې ده چې جاوا د سي پلس پلس په پرتله ډېره پرمختللې او غښتلې ژبه ده. د مثال په ډول، جاوا د نښه کوونکو اړتیا له منځه وړې او له کلاسونو څخه د پروګرام د جوړښت لپاره د ثابتو بنسټې تیرو په توګه ګټه پورته کوي. سافټوېر جوړوونکې اداري لکه

مایکروسافټ (تر پرونه پورې) او آی بي اېم شرکت د جاوا د پرمختک لپاره له سن سره لاس یو کړی دی.

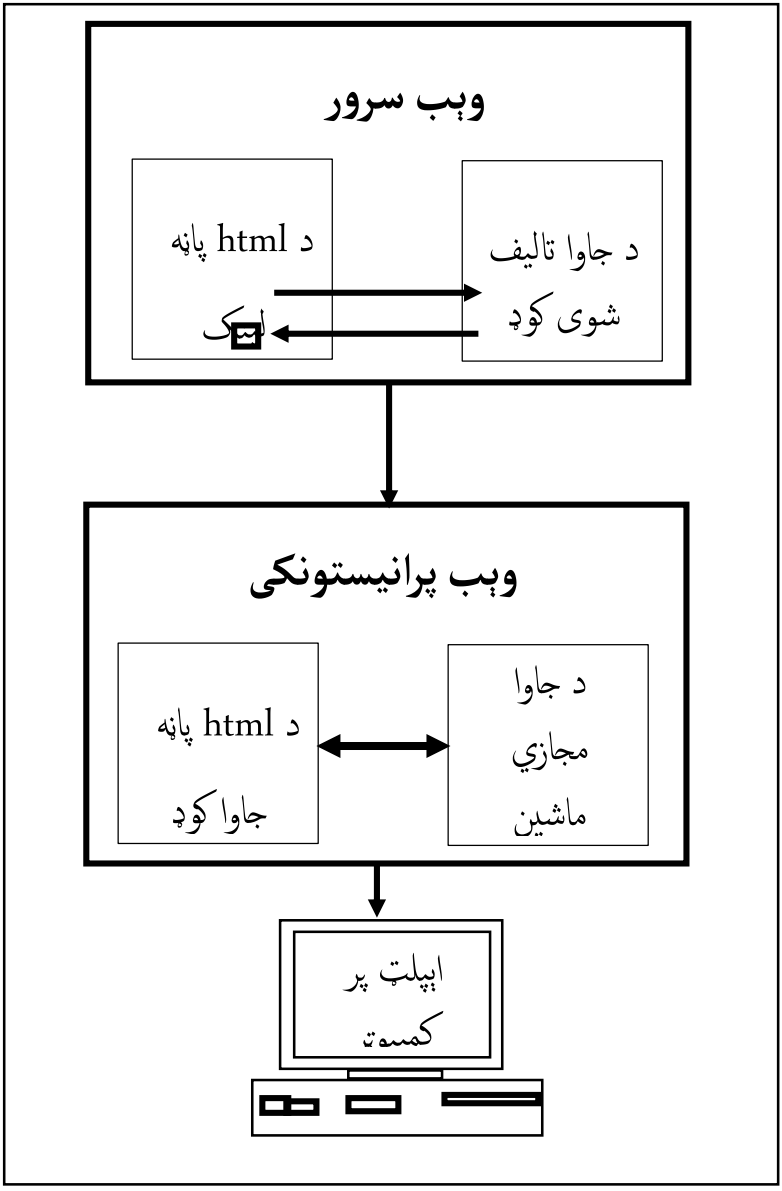
د جاوا یوه بله ستره ځانګړنه دا ده چې جاوا خپلواک-دریځه ژبه ده. پخپله ژبه د کنټرول عامل سیسټمونو له دریځونو څخه بېله او پردې ده. د هر دریځ لپاره، د جاوا مجازي ماشین (Jave Virtual Machine) جوړېږي چې د جاوا د تالیفوونکو لپاره تولید شوی کوډ کمپایل یا ژباړي څو په یاد دریځ وځغلېدای شي.

د خوندیتوب لپاره، د جاوا اپلټونه د یوه شکلن-ډبلي یا sandbox او یا محدود چاپېریال دننه ځغلي تر څو کټه اخیستونکی یې د جاوا له زیانمنو پروګرامونو څخه خوندي وي. (د مثال په ډول، پروګرامونو ته دا اجازه نه ورکول کېږي چې د کټه اخیستونکي زېرمتون ته لاسرسی ولري او یا د کټه اخیستونکي ماشین له بل وېبسایټ سره وصل کاندي.) همدا رنګه، وېبپرانیستونکي هم د جاوا اپلټونو د مخنیوي لپاره غیرفعاله کېدای شي.

## بالغه ټکنالوژي

د سن کمپنۍ جاوا په دوو ذایقو کې راځي: د جاوا معیاري اډیشن (J2SE) د مایکروسافټ د وینډوز، د سن د سولاریس او لینکس لپاره؛ او د جاوا سوداګریز اډیشن (JSEE) چې له هغو ځانګړنو څخه برخمن ده چې په سترو او پېچلو چاپېریالونو کې ورته اړتیا وي. مایکروسافټ د وینډوز لپاره د جاوا خپله لهجه هم جوړه کړې، خو د سن د قانوني اقدام په پایله کې یې لاس ترې واخیست.

(کمپینو ته دا اجازه ده چې د بېلا بېلو درېځونو لپاره د جاوا تطبیقي نسخې جوړې کړي، خو په دې شرط چې دوی د سن له ټاکلې قانوني پروسې څخه وانه وړي.)



د دې لپاره چې جاوا اپلټ وځغلي ، ګټه اخیستونکی ، وېب پرانیستونکي ته لینک شوې پاڼه را پورته کوي. په دې پسې اپلټ ښایي په خپلواک ډول وځغلي ، او یا کېدای شي چې له یوه ځانګړي لینک او یا کنټرول لکه بټن پورې وصل اوسي. یو ځل چې فعاله شي ، نو اپلټ وېب پرانیستونکي ته ډاونلوډ کېږي ، چې وروسته د خپل جاوا مجازي ماشین په نوم د یو ماډول (module) په کارولو خپل جاوا-کوډ ځغلوي. د هر کمپیوټري سیسټم لپاره د جاوا بېل مجازي ماشین (JVM) شتون لري.

جاوا، د سافټوېر بیا کارېدونکو اجزاوو ته ځانګړې پاملرنه کړې. جاوا-لوبیا یا JavaBeans یو شمېر ټرلي-کلاسونه په یوه واحد کې را ټولوي چې د یو ډله معیاري میتودونو په وسیله لاسرسی ورته کېدای او په خپلواک ډول د خپلو موادو په هکله د معلوماتو لپاره چمتو کېدای شي.

نن ورځ له وېب خدماتو سره د کار کولو لپاره د جاوا ځواکمن او په ښه ډول ترتیب شوي د جاوا پروګرامي اړیکځایونه هم شتون لري. په داسې حال کې چې د کلاینټ لوری، جاوا اپلټونه پر وېب پرانیستونکي ځغلوي، خو د جاوا سرور پاڼې (JSP) همدا کوډ د HTML په یوه پاڼه کې ور ځایوي او نومورې کوډ د سرور لوري کاریال «سرولېټ»[1] په مرسته تالیف کېږي. ورپسې XML پروسېس او د XML لپاره ډېټابېس ته لاسرسی د جاوا د API له خوا چمتو کېږي.

[1] servlet

جاوا خپله هغه ژمنه تر ډېره پورې پوره کړې چې د اصلي تګلوري[1] استاوازله پروګراملیکنه په لوی شمېر دریځونو باندې د پلي کېدو وړ وګرځوي. یاده ژبه نن ورځ د سي یا سي پلس پلس پر ځای د لومړنۍ ژبې په توګه تدریس کېږي. خو، نن ورځ د سافټوېر جوړونې د نړۍ په چټک پرمختګ کې، د نوموړې ژبې په هکله نور دا نظر د اطلاق وړ نه بریښي چې همدا یوازینۍ مشهوره او غوره ژبه ده.

په ۲۰۰۶ کې سن مایکروسیستم شرکت اعلان وکړ چې د جاوا سرچینه کوډ به په وړیا ډول د لاسرسي وړ وګرځوي. تر یو څه بریده کېدای شي دا د دې لپاره یوه هڅه اوسي چې جاوا ژبه د پروګرام لیکونکو ترمنځ پاتې شي، چې ځینو پروګرام لیکونکو اوس خپل پام له جاوا څخه د مایکروسافټ د سي پلس پلس نسخې ته اړولی. د جاوا لپاره تر ټولو ستره ننګونه د وېب پروګراملیکنې په ډګر کې ده، چېرته چې نوموړې ژبه ورځ تر بله له ځیرکو او ټیزو ژبو سره مخ کېږي (د مثال په ډول وګورئ، Ruby). دغه راز متنواله ژبې هم د جاوا لپاره یوه لویه ننګونه ده چې ښایي د زده کړې لپاره اسانه او د زیاتره کاریالونو لپاره یې استعمالول چټک وي.

[1] Streamline

## اضافي لوست:

1- Arnold, Ken, James Golsing, and David Holmes. *The Java Programming Language.* 4th ed. Upper Saddle River, N.J.: Prentice Hall, 2005.

2- Burd, Barry. *Beginning Programming with Java for Dummies.* 2nd ed. Hoboken, NJ: Wiley, 2005.

3- Gosling, James. "Is Java Getting Better with Age?" [interview]. Cnet News. Available online. URL: http://news.com. com/2008-7345_3-6022062.html. Accessed April 10, 2007.

4- Krill, Paul. "Java Facing Pressure from Dynamic Languages." *Info-World,* March 25, 2006. Available online. URL: http://www. infoworld.com/article/06/03/25/76803_HNjavapressure_ 1.html. Accessed April 10, 2007.

5- McGovern, James, et al. *Java Web Services Architecture.* San Francisco: Morgan Kaufmann, 2003.

6- Schildt, Herbert. *Swing: A Beginner's Guide.* New York: McGraw-Hill, 2007.

***

# (۱٤)

## جاواسکرپټ (JavaScript)

جاواسکرپټ له هغو نوموتو ژبو څخه ده چې وېبپاڼې د دې وړ ګرځوي چې له ګټې اخیستونکو سره په چټک او اغېزمن ډول ارتباط ټینګ کړي. نوموړې ژبه د لومړي ځل لپاره د ۱۹۹۰ کلونو په منځ کې د نېټسکېپ-۲[1] په پرانیستونکي کې د لایوسکرپټ[2] تر نوم لاندې راڅرګنده شوه. له تخنیکي پلوه، جاواسکرپټ د سن مایکروسیستم شرکت یو تولید دی چې د خپل یوه معیار چې EMAScript بلل کېږي، د پلي کولو لپاره یې جوړه کړه. سره له دې چې د جاوا او جاواسکرپټ نومونه سره نږدې دي، خو جاواسکرپټ د جاوا له پروګرامي ژبې سره په مستقیم ډول کومه اړیکه نه لري.

په لومړنیو کلونو کې جاواسکرپټ پخپله د ځان د بریالیتوب پر وړاندې لوی خنډ وه. نوموړې د کارولو لپاره یوه نسبتاً اسانه متنواله ژبه وه چې پخوانیو مبتذلو وېب چوکاټونو ته یې د ځینو مشخصاتو لکه درې بعدي بټنونو او پاپ-اپ کړکیو، د ور زیاتولو لپاره اسانه لاره چمتو کوله. خو څنګه چې پخوانیو نسلونو د ګڼشمېر فونټونو له کارولو سره لېوالتیا لرله، نو په دې اساس د جاواسکرپټ د لومړنیو پروګرام لیکونکو وېبپاڼې به له سرګردانوونکو او بې ګټو ګډوډیو څخه ډکې وې.

---

[1] Netscape-2
[2] Livescript

د دې ترڅنګ چې ځینې وختونه به یې کټه ا خیستونکي هم ځوروله، د لومړیو وختونو جاواسکرېپټ له هغو توپیرونو څخه هم کړېده چې یاده ژبه به په لویو پرانیستونکو کې په څه ډول پلي کېږي. په پایله کې به د نېټسکېپ د کټې اخیستونکو کار کله ناکله د هغه جاواسکرېپټ له امله ټکنی کېده چې د مایکروسافټ په انټرنېټي پرانیستونکي کې به لیکل شوي وو او یا به په جاواسکرېپټ کې د مایکروسافټ لپاره لیکل شوي وو. په پای کې د پرانیستونکو نیمګړتیاوو کله نا کله جاواسکرېپټ ته اجازه ورکوله چې په وایرس ککړ شوو د پرانیستونکو مرستیالانو[1] د نصبولو له لارې د وېبپاڼو پر امنیت د له منځه وړلو لپاره وکارول شي. په پایله کې، د امنیت ډېرې ماهرینو دا وړاندیزونه پیل کړل چې کټه اخیستونکي دې په خپلو پرانیستونکو کې د جاواسکرېپټ اجرا، غیرفعاله کړي.

## د جاواسکرېپټ کارول

د جاواسکرېپټ نحوي ترکیب او ژبني رغښتونه سي ژبې ته ورته دي او ورسره د استازواله ژبو له ځانګړنو څخه هم برخمنه ده. نوموړې ژبه په خپله پر چاپېریال (لکه ورکړیز/راکړیز) باندې د عملیو د ترسره کولو هېڅ وړتیا نه لري. د دې پرځای، جاواسکرېپټ یوه ماشین[2] ته بلنه ورکوي، چې د هر کوربه چاپېریال (عموما

[1] browser helpers
[2] engine

یو وېب پرانیستونکي) لپاره لیکل شوی وي. یاد ماشین هغه ځانګړنې تطبیق کوي چې د دې لپاره ډیزاین شوې وي څو دا کنټرول کړي چې یوه وېبپاڼه له ګټه اخیستونکي سره په څه ډول تعامل کوي، لکه د کړکیو او کنټرولونو لکه مینوګانو، بټنونو او یا ابزارپټو[1] څرګندول. جاواسکرپټ تر دې دمخه چې سرور ته تسلیم شي، په پرانیستونکي کې د یوه وېب چوکاټ[2] د قانون بندۍ لپاره هم کارېږي. عموماً، د پرانیستونکي لوري[3] جاواسکرپټ پروسېس پر وېب سروروونو باندې بوج کموي او په ورته وخت کې پاڼه چټک غبرګون ته چمتو کوي، د بېلګې په ډول، کله چې ګټه اخیستونکی د پاڼې په یوه برخه د ماؤس نښه کوونکی ودروي، نو ورسره ګرافیک هم بدل شي.

د جاواسکرپټ او html د پاڼو ترمنځ اساسي اړیکځای، ډاکومنټ آبجکټ ماډل دی. د نړیوال جال ائتلاف[4] د DOM معیاري فنکشنونه تعریف کړي او ډېرې پرانیستونکي په ثابت ډول د دې معیارونو لومړني او دوهمې پوړونو څخه ګټه اخلي. که څه هم داسې ګټه اخیستونکي شته چې د جاواسکرپټ معیارونه نه شي ځغلولی، لکه په ستركو راندې ګټه اخیستونکي، هغه ګټه اخیستونکي چې د موبایلونو پرانیستونکي کاروي (لکه ډیجیټل مرستیال (PDA) یا ځیرک ټلیفونونه)، او یا هغه ګټه اخیستونکي چې د امنیتي ستونزو له امله یې جاواسکرپټ غیرفعاله

[1] Toolbars
[2] Web frame
[3] browser side
[4] World Wide Web Consortium

کړی وي. له همدې کبله، کله چې جاواسکرپټ د ویبپاڼې د اساسي فنکشنونو لپاره کارول کېږي (لکه د فورم پروسېس)، نو ډوېلپر باید د ګټې اخیستونکي لپاره یوه بدیل لاره چمتو کړي تر څو هغه اړوند کار ترسره کړي. (د معیوبو ګټې اخیستونکو په صورت کې، کېدای شي دا یوه قانوني غوښتنه اوسي.)

په دودیز ډول، د جاواسکرپټ کوډ په مستقیم ډول د ټېګونو په کارولو د html پاڼې سره یو ځای کېږي، لکه په لاندې مثال کې:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html dir="ltr" lang="en">
<head>
<title>JavaScript Example</title>
<body>
<script type="text/JavaScript">
          var Name = prompt ("Enter your
          name","");
     alert(Name);
</script>
</body>
</html>
```

کله چې هغه پرانیستونکی چې جاواسکرپټ پکې فعاله وي له دې کوډ سره مخ شي، نو یو متنچوکاټ له کټه اخیستونکي څخه د نوم غوښتنه کوي، چې د Name په متحول کې زېرمه کېږي او وروسته بیا په خبرتیایي-چوکاټ[1] کې څرګندېږي.

د XHTML د معیارونو په مرسته د وېبپاڼو په عصري ډیزاین کې څنګه چې د فارمټ ورکونې معلومات په یوه بېل فایل کې ایښودل کېږي، په همدې ډول د جاواسکرپټ کوډ هم په یوه بېل فایل کې ساتل کېږي او ډېر په ساده توګه د html له فایل سره وصل کېږي:

```
<script type="text/javascript"
src="mainscript.js"></script>
```

جاواسکرپټ یوازې د معلوماتو له څرګندولو او یا فورمونو له پروسېس څخه ډېر نور کارونه هم کولای شي. جاواسکرپټ بېلابېلو وېب خدمتونو (لکه ډېټابېسونو او لټون ماشینونو) ته لاسرسی موندلی او د کټه اخیستونکي د کړنو په غبرګون کې لاسي پانې هم جوړولای شي. جاواسکرپټ له وېبپرانیستونکو پرته په نورو کاریالونو کې هم اغېز کېدای شي: د مثال په ډول، اډوب اکروبات او لوستونکی یې او حتی د عامل-سیستم متنواله ژبه کې د مایکروسافټ له Jscript او Jscript.NET سره یو ځای کېدای شي. که څه هم وروستي کلونه داسې برېښي

[1] alert-box

چې خپل لوری یې نورو ژبو لکه PHP ته اړولی وي، خو جاواسکرپټ تر اوسه پورې د وېب-ډیزاین لپاره یوه ځواکمنه او په لویه کچه کارېدونکې وسیله ده.

## اضافي لوست:

1- Flanagan, David. *JavaScript: The Definitive Guide.* 5th ed. Sebastapol, Calif.: O'Reilly, 2006.
2- Heilmann, Christian. *Beginning JavaScript with DOM Scripting and Ajax.* Berkeley, Calif.: APress, 2005.
3- JavaScript at Webdeveloper.com. Available online. URL: http://www.webdeveloper.com/javascript/. Accessed September 24, 2007.
4- JavaScript.com: The Definitive JavaScript Resource. Available online. URL: http://www.javascript.com/. Accessed September 24, 2007.
5- Wilton, Paul, and Jeremy McPeak. *Beginning JavaScript.* 3rd ed. Indianapolis: Wiley, 2007.

***

# (۱۵)

# لیسپ (LISP)

کله چې د ۱۹۵۰ کلونو په لومړیو وختونو کې له مصنوعي ځیرکتیا[1] سره لېوالتیا وده وکړه، نو څېړونکي د هغه مهال د کمپیوټر له ټیټو ژبو (low-leve) څخه ډېر ژر ستومانه شول، چې په شمېرنه او پر اعدادو باندې د نورو عملیو پر ترسره کولو یې ټینګار کاوه، د دې پر ځای چې سمبولیکه ډېټا، پروسېسس کړي.

په ۱۹۵۶ کال کې پر مصنوعي ځیرکتیا د ډارټماؤت د اوړي د څېړنیزې پروژې پر مهال، چې د دې ډګر لومړني مخکښان یې را یو ځای کړل، جان میکارتي د یوه بېل ډول کمپیوټري ژبې په هکله خپل فکر وړاندې کړ. نوموړی په دې باور وو چې دا ډول یوه ژبه باید د دې وړ وي چې په خپلو اصطلاحاتو کې له ریاضیکي فنکشنونو سره تعامل وکړي – او پر سمبولونو عملیې ترسره کړي، نه دا چې یوازې اعداد محاسبه کړي.

له ماروین مینسکي سره یو ځای، میکارتي د یوې ژبې پلي کول پیل کړل چې LISP یا لیست پروسېسر[2] نومېدله. خنګه چې یې له نومه ښکاري، نوموړې ژبه د ډېټا د زېرمه کولو لپاره له لیستونو څخه ګټه اخلي او د لیست په عناصرو باندې

[1] Artificial Intelligence
[2] List Processor

د عملیو د ترسره کولو لپاره کټ فنکشنونه چمتو کوي. لیست کولای شي چې مفرد عناصر ولري (چې آټوم بلل کېږي)، لکه دلته:

```
(A B C D)
```

خو لیستونه کولای شي چې نور لیستونه هم ولري، لکه لاندې:

```
(A (B C) D)
```

د لیست هر توک د نوډ[1] په توګه زېرمه کېږي چې د ډېټا وېلیوګانو لپاره هم یو نښه کوونکی لري او په همدې ټرلي لیست کې راتلونکي توک لپاره هم نښه کوونکی لري. د لیست سیسټم عموما له ذخیره شوو فنکشنونو څخه جوړ وي لکه ګاربېج کلېکشن[2]، چې په دې کې د لیست له توکونو څخه تشه شوې زېرمه، وروسته د تخصیص لپاره د تش-لوسته زېرمې برخې ته واپس کېږي.

د لیسپ پروګرامونه په لومړي نظر ټرلي ښکاري، ځکه نوموړي پروګرامونه کټ شمېر په یو بل کې اوبدلي قوسونه لري. خو افادې او فنکشنونه په اصل کې د ډېرې نورو ژبو په پرتله په ساده ډول رغېدلي وي. پېچلي اوړون ته له اړتیا پرته، د لیسپ ژباړونکی (چې ایول یا eval بلل کېږي ځکه دی راکړیز ارزوي[3]) د ډېټا مسیر ته ګوري او لومړی دا پوښتنه کوي چې آیا راتلونکی توک ثابت دی (لکه عدد، یا په لیندیو کې لیکلی سمبول، متنکرښه، په لیندیو کې لیکلی لیست، او

[1] node
[2] garbage collection
[3] evaluate

یا اړین-ویوکی). که چېرته همداسې وي، نو وېلیو یې ېرته واپس کېږي. که نه، ژباړونکی ګوري چې آیا نوموړی توک له وړاندې څخه تعریف شوی متحول دی، او که همداسې وي، نو د هغه وېلیو واپس کوي. په پایله کې، ژباړونکی ګوري چې آیا کوم لیست شته. که چېرته هو، نو له یاد لیست سره د یوې فنکشني بلنې[1] په څېر چلند کېږي چې ترځنګ یې ارګومنټونه راغلي وي. په دې اساس فنکشن ته بلنه ورکول کېږي، ډېټا ورکول کېږي، او پایله یې واپس کېږي.

**د لیسپ د توکونو مثالونه**

| د توکو ډول | مثال | ارزونه |
|---|---|---|
| عدد | ۲۴ | ۲۴ |
| اعشاري | ۵,۵ | ۵,۵ |
| نسبت | ۳/۴ | ۰,۷۵ |
| اساسي لغت | Defun | فنکشن تعریفوي |
| لیندۍ لرونکی لیست | (۵ ۱ ۴ ۱ ۳) | (۶ ۱ ۴ ۱ ۳) |
| بولین | تش | غلط |
| فنکشن بلنه | + ۴ ۲ | ۶ |
| متحول | A | وېلیو یې |
| لیندۍ لرونکی متحول | ‘a | A |

[1] function call

الګول، پاسکال او سي ته ورته ژبې په عبارتونو او پروسو ټینګار کوي. خو بل خوا، لیسپ بیا لومړنۍ فنکشني[1] ژبه وه. د لیسپ د یوه پروګرام غوښنه برخه هغه فنکشنونه جوړوي چې له خپلو ارګومنټونو سره یوځای ارزول کېږي. په لیسپ ژبه کې ځاني، او یا اولیه فنکشنونه شتون لري. د ریاضي د عادي عملیو سربېره، د بنسټیز لیست پروسېس لپاره اولیه فنکشنونه لري. د مثال په ډول، لیست فنکشن له خپلو ارګومنټونو څخه یو لیست جوړوي: (لیست ۱ ۲ ۳) په خپله همدا لیست (۱ ۲ ۳) واپس کوي، په داسې حال کې چې cons فنکشن د لیست په سر کې یو اټوم ځای په ځای کوي، او append فنکشن یې بېرته بېخ ته ټپل وهي. پروګرام لیکونکي د خپلو فنکشنونو د تعریفولو لپاره د defun له اړین-ویوکي څخه ګټه پورته کوي.

لیسپ دوه نورې ځانګړنې هم لري چې له دې ژبې څخه یوه ځواکمنه او انعطاف منونکي ژبه جوړوي تر څو پر سمبولونو او ډېټا باندې، ډول ډول عملیات ترسره کړي. لیسپ راګرځنده (recursive) فنکشنونو ته هم اجازه ورکوي. د مثال په ډول، لاندې فنکشن د x یو متحول د y توان ته پورته کوي:

```
(defun power (x y)
(if (= y 0) 1
(* x (power x (1- y)))))
```

[1] Functional

دلته د if افاده دا ګوري چې آیا y صفر دی او که نه. که نه وي، نو دوهمه افاده د power فنکشن را پاروي[1]، چې همدا ازمایښت یو ځل بیا ترسره کوي. پایله یې دا چې فنکشن بیا بیا خپل ځان ته بلنه ورکوي، لنډمهاله وېلیوکانې ذخیره کوي، تر هغو چې y صفر شي. له هغه وروسته دی ځان بېرته ور ټولوي، او x له خپل ځان سره y ځله ضربوي.

خو ښایي د لیسپ تر ټولو خوندوره ځانګړنه دا وي چې نوموړې ژبه د پروګرامونو (فنکشنونو) او ډېټا ترمنځ هېڅ توپیر نه کوي. دا چې یوه فنکشني بلنه او د هغه آرګومنټونه پخپله یو لیست جوړوي، نو یو فنکشن په نورو فنکشنونو باندې د ډېټا په توګه لوستل کېدای شي. په دې ډول د هغو پروګرامونو لیکنه اسانېږي چې په خپلو عملیو کې بدلون راولي.

## د ژبې تحول

لیسپ ژبې په لنډ وخت کې د مصنوعي ځیرکتیا د څېړونکو پام ځانته را واړاوه، او هغه ښه چې Lisp 1.5 نومېده، د لویې کچې د کاریالونو لیکلو لپاره یوه ګړندۍ ژبه وګڼل شوه. په داسې حال کې چې مخکښو کمپیوټرپوهانو ډېر کله د الګوریتمونو د بیانولو لپاره الګول او د هغه راتلونکي نسلونه د نړیوالې[2] ژبې په توګه کارول، خو لیسپ د مصنوعي ځیرکتیا د خلکو ترمنځ مشترکه نړیواله ژبه شوه.

---

[1] Invoke

[2] Universal

د لیسپ ژبې ځنې ورژنونه لکه Mac-LISP ولیکل شوو تر څو د نوي هارډویر ملاتړ وکولای شي او د LMI او سمبولیکس په څیر شرکتونو له خوا پرمختګ ورکول شي. په داسې حال کې چې څېړونکو د ژباړل شوې لیسپ ژبې تعاملي فطرت خوښاوه (چېرته چې فنکشنونه تعریفېدای او په سملاسي توګه په کیبورډ عملي کېدای شوی)، خو عملي کاریالانو داسې تالیف ته اړتیا لرله چې کوډ د ماشین ژبې ته تالیف کړي څو د کفایت وړ چټکتیا ترلاسه کړي.

د لیسپ تر ټولو زیات کارېدونکی ورژن سکیم[1] نومېږي، چې د ۱۹۷۰ کلونو په منځنیو وختونو کې د MIT[2] د څېړونکو له خوا جوړه شوه. Scheme د لیسپ نحوي ترکیب ساده کړ (چې اصلي روح یې هم پخپل ځای پاتې وو) او په ورته وخت کې فنکشنونو ته دا اجازه ورکوي چې د ډېټا-ټولګو[3] وړتیاوې ولري او په دې ډول ژبې ته عمومیت وبښي. په دې معنا چې فنکشنونه، نورو فنکشنونو ته د پارامترونو په توګه ور تېرېدای، د وېلیوګانو په توګه واپس کېدای، او د لیستونو او متحولینو په توګه ټاکل کېدای یا اساین کېدای شي.

تر ۱۹۸۰ کلونو دمخه، شخصي کمپیوټرونه د لیسپ د ځغلولو لپاره په کافي کچه ځواکمن وو، او د لیسپ د ېلا ېلو نسخو په مختلفو دریځونو باندې د ځغلېدلو ځانګړتیا د معیار ورکونې[4] غورځنګ ته لاره هواره کړه چې په پایله کې یې په

[1] scheme
[2] Massachusetts Institute of Technology – د ماسوچوسېټ د ټکنالوژۍ مؤسسه
[3] data entities
[4] Standrandization

۱۹۸۴ کې کامن لیسپ[1] رامنځته شوه. کامن لیسپ د شته لهجو زیاتره ځانګړنې په ځان کې راټولوي، او د ډیټا له غني ډولونو څخه برخمنه ده، او همدا رنګه د تاکیدي، او ترتیبواله پروګراملیکنې تګلارو کارولو ته هم پراخه اجازه ورکوي لکه سي ته ورته ژبو کې. په دې ډول نوموړې ژبه د پروګراملیکنې لپاره ګڼ سبکونه وړاندې کوي. لیسپ نن ورځ د سوداګریزو او اشتراکي پوستغالو (shareware) په نسخو کې په پراخه توګه شتون لري.

سیمور پیپرټ لیسپ ته ورته ژبه چې لوګو (Logo) نومېدله جوړه کړه، چې زلمیو زده کوونکو ته د کمپیوټرپوهنې د مفاهیمو د ښوونې لپاره کارول کېده.

## اضافي لوست:

1- Dybvig, R. Kent. *The SCHEME Programming Language.* 3rd ed. Cambridge, Mass.: MIT Press, 2003.

2- “Lisp Information and Resources.” Available online. URL: http:// www.lispmachine.net/. Accessed August 13, 2007.

3- Queinnec, Christian. *Lisp in Small Pieces.* New York: Cambridge University Press, 1996.

4- Seibel, Peter. *Practical Common Lisp.* Berkeley, Calif.: Apress, 2005.

***

[1] common lisp

# (۱۶)

## لوګو(Logo)

لوګو د لیسپ ژبې یو مشتق دی چې د هماغې ژبې د لیست پروسېس او سمبولیکي لاسوهنې[1] ځواک یې تر ډېره پورې ساتلی دی، په داسې حال کې چې نحوي ترکیب یې د نوموړې ژبې په پرتله ساده، تعامل یې اسانه، او ګرافیکي وړتیاوې یې د ځوانانو لپاره زړه راښکونکې دي. لوګو ډېر کله د لومړنیو او منځنیو ښوونځیو زده کوونکو ته د لومړنۍ کمپیوټري ژبې په توګه ښودل کېږي. د هارولډ ابلسن په ۱۹۸۲ کې د ایپل لوګو په رسامي کې پام شوو چې «لوګو د ښوونیزې فلسفې او د ورځ تر بله وده کوونکو پروګرامي ژبو د کورنۍ لپاره یو نوم دی چې د نوموړې ژبې په احساس کې مرسته کوي.»

لوګو د یوه ښوونکي سېمور پېپرټ او د هغه د ملګرو له خوا د بولټ، برنېک او نیومن په شرکت کې جوړه شوه چې کار یې په ۱۹۶۷ کې پیل شو. ریاضي پوه او د مصنوعي ځیرکتیا د مخکښ سېمورپېپرټ زړه ته، د ودې او پرمختګ له اړخونه جان پیاجټ سره له کار کولو وروسته د یوې ښوونیزې کمپیوټري ژبې د جوړولو لپوالتیا ور ولوېده. پېپرټ، د پیاجټ پر رغښتواله[2] باندې ټینګار ته ځانګړې پاملرنه کوله. رغښتواله هغه نظریه ده چې خلک په شته قالبونو کې هغه

---

[1] symbolic manipulation
[2] constructivism

چې د ورځني ژوند له تجربو څخه رغېدلي وي، د نوو مفاهيمو د ځايولو له لارې زده کړه کوي. پېپرټ په دې باوري شو چې د يوې انتزاعي کمپيوټري ژبې لکه فورټران او يا حتى بېسک زده کړه د ماشومانو لپاره سخته ده، ځکه چې د دې ژبو الجبري فورمولونو او نحوي ترکيب د هغوی له ورځنيو فعاليتونو لکه قدم وهلو، لوبو کولو، رسامي کولو، او يا د شيانو له جوړولو سره ډېر کم سمون درلود.

د مثال په ډول، ډېرى کمپيوټري ژبې د هغو عبارتونو په کارولو ګرافيک پلي کوي چې د کارتيزن محورونو (X,Y) په مرسته د پردې ټکي مشخصوي. د مثال په ډول، يوه مربع کېداى شي چې د لاندې عبارتونو په مرسته رسم شي:

```
PLOT 100, 100
LINETO 150, 100
LINETO 150, 150
LINETO 100, 150
LINETO 100, 100
```

په داسې حال کې چې د محورونو له سيستم سره بلدتيا په پاى کې يو څوک د دې جوګه کوي چې دا عمليه په ذهن کې ترسيم کړي، او دا له درک څخه ډېر لرې خبره ده.

پېپرټ په خپله لوګو ژبه کې يو کيشپ راوړي. کيشپ په اصل کې يو ربنتينى روبات وي چې شاوخوا ته د حرکت کولو لپاره پروګرام کېداى شي؛ نن ورځ په ډېرى سيستمونو کې نوموړى کيشپ د يوه کرسر په بڼه وړاندې کېږي. ځنګه

چې کیشپ حرکت کوي، نو د یوه قلم په مرسته تر شا یوه «لیکه» پریږدي، چې همدا بیا ګرافیک رسموي.

د کیشپ د کمانډونو په مرسته، یوه مربع په لاندې ډول انځوریدای شي:

```
FD 50 (that is, forward 50)
RT 90 (turn right 90 degrees)
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
```

په دې ځای کې، مبتدي پروګرام لیکونکی په اسانۍ سره قدم وهل او تاوېدل په ذهن کې مجسم کولای شي تر هغو چې پروګرام لیکونکی/ې ېرته د پیل ټکي ته راګرځي. د پیاجټ تیوریانو ته په پام سره، زده کړه زمور له فزیکي نړۍ او ورځینو فعالیتونو سره اوړه په اوړه مخته ځي.

په لوګو کې نورو ژبو ته ورته کنټرول جوړښتونه هم شته، نو پورتنی پروګرام په ډېره اسانه په لاندې ډول لیکل کېدای شي:

```
REPEAT 4 [FD 50 RT 90]
```

لوګو یوازې د رسامۍ له ځینو کمانډونو څخه هاخوا ډېر نور کارونه هم کولای شي. زده کوونکي د لیست پروسېس کمانډونو په مرسته کولای شي چې له کمپیوټري شاعرۍ څخه نیولې تر حیرانوونکو ټیمونو پورې هر څه جوړ کړي. د لیسپ په

خلاف چې کمانډونه یې په ګونګ ډول نومول شوي وو لکه car او cdr؛ د لوګو لیست کمانډونه په ډېره اسانۍ د پوهېدو وړ دي. د مثال په ډول، first په لیست کې لومړنی توک راګرځوي، حال دا چې butfirst له لومړني توک پرته د لیست ټول توکونه واپس کوي.

د لوګو پروسې د اړین-ویوکو په مرسته جوړېږي، ګواکې یو پروګرام لیکونکی چې کمپیوټر ته دا ور زده کوي چې یو څه، څنګه وکړي. د مثال په ډول، یوه پروسه چې د پیل له ټکو څخه او د یوه متحول په اندازه یوه مربع رسموي، په لاندې ډول ښکاري:

```
to square :X :Y :Size
setxy :X :Y
repeat 4 [fd :Size rt 90]
end
```

لوګو ژبه د کلونو په اوښتو په پرېمانه ډول پراخه شوې، او اوس نه یوازې د ریاضي د فنکشنونو یوه بشپړه ټولګه لري، بلکې ډېرې ورځنونه یې د ویندوز او مېکنټاش لپاره ځانګړي غږونه، ګرافیک او ملټي-میډیا هم لري. د ۱۹۸۰ کلونو په منځنیو وختونو کې لوګو له نومیالي لیګو (Lego) لوبتوکي سره یو ځای کړای شوه تر څو لیګو لوګو (LEGO Logo) جوړه شي. دا نومیالی لوبتوکی زده کوونکي د دې جوګه کوي چې ډول ډول روباټونه او نور میخانکي وسایل جوړ او کنټرول کړي.

په ۱۹۹۰ کلونو کې، لوګو پر ښوونکو باندې د هغه فشار له کبله یوه نومورکي ژبه وګرځېده چې باید د سي پلس پلس او جاوا د ژبو په مرسته د «ربښتیني نړۍ»

د پروګراملیکنې مهارتونه وپنځوي. خو سربیره پر دې هم، لوګو د اروپا، جاپان، او لاتیني امریکا په ځینو برخو کې ښه وځلیده. لوګو په لنډو وختونو کې د دوو نوو ورژنونو په جوړولو هم پراخه انرژي واخیسته. مایکرو-ورلډز لوګو د میکنټاش له اړیکځای څخه کټه واخیسته تر څو د ملتي میډیا لپاره له بشپړو ځانګړنو برخمن کاري چاپیریال وړاندې کړي، او وروسته د وینډوز سیسټمونو له لوري هم په کار واچول شوه. د لوګو بله نسخه، StarLogo د موازي پروسېس په مفاهیمو باندې ټینګار کوي او د دې وړتیا لري چې په زرهاوو بېلا بېل کیشپونه کنټرول کړي چې د مرغانو د غونډ او یا ترافیکي هجوم د چلند د تقلید لپاره پروګرام کېدای شي. څنګه چې د بریان هاروې کتابونه څرګندوي، د لوګو د لاسرسي وړ او تعاملي فطرت زلمیو زده کوونکو ته هم د کمپیوټرپوهنې د ښوونې لپاره نوموړې ژبه یو غوره انتخاب کړځوي.

## اضافي لوست:

1- Harvey, Brian. *Computer Science Logo Style.* Vols. 1–3, 2nd ed. Cambridge, Mass.: MIT Press, 1997.

2- LCSI Microworlds. Available online. URL: http://www.microworlds.com/. Accessed August 13, 2007.

3- Logo Foundation (MIT). Available online. URL: http://el.media.mit.edu/Logo-foundation/. Accessed August 13, 2007.

4- Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, 1993.

5- “StarLogo on the Web.” MIT Dept. of Education. Available online. URL: http://education.mit.edu/starlogo/. Accessed August 13, 2007.

6- “Welcome to MSWLogo.” Available online. URL: http://www.softronix.com/logo.html. Accessed August 13, 2007.

***

# (۱۷)

## لووا (Lua)

لووا یوه متنواله[1] ژبه ده چې د برازیل په هېواد کې د «ریو ډي. جانېریو» له لوري په پوهنتون کې اختراع شوه. (لووا په پرتګالي ژبه کې «سپورې» ته ویل کېږي). نوموړې ژبې تر یو څه حده د خلکو پام ځان ته راواړه، په ځانګړي ډول د ګېمونو او انټرنېټي پروګرام لیکونکو.

لووا ساده نحوي ترکیب لري او د دودیزې (تاکیدي) پروګراملیکنې او فنکشني پروګراملیکنې دواړو لپاره کاري بستر وړاندې کوي او ملاتړ یې کوي. ډېرې هغه ځانګړنې لکه توارث او نومځایونه چې په استازواله ژبو کې جوړېږي، په لووا ژبه کې نه موندل کېږي، خو کولای شو چې د ژبې د پراختیایي میتودونو[2] په مرسته یې جوړ کړو.

د لووا یو ساده فنکشن په لاندې ډول جوړښت لري:

```
function factorial(n)
if n = 0 then
return 1
end
return n * factorial(n - 1)
```

---

[1] scripting
[2] extension methods

```
end
```

د سیمیکالن نشتون ته مو پام وي!

د ځاني ډیټا ډولونو سربېره، لووا د پېچلو او د ګټې اخیستونکي له خوا تعریف شوو ډیټا ډولونو د جوړولو لپاره له جدولونو څخه هم ګټه اخلي. په جدولونو کې د وېلیوګانو او کیلیانو جوړې شتون لري. کیلي کېدای شي چې یو عدد اوسي (چې په نورو ژبو کې د کتار له جوړولو سره معادل دی)، او یا متنکرښه وي. د مثال په ډول، لاندې جدول دري وېلیوګاني لري چې د متنکرښو په مرسته واټن پکې ایجاد شوی:

```
coin = { quarter = 25, dime = 10, nickel =
5,
penny = 1 }
printer (coin[“dime”]) -- prints 10
```

په جدولونو کې د فنکشنونو او اړتیا وړ ډیټا په اضافه کولو، د استازواله ژبو په څېر په دې ژبه کې هم کلاسونه جوړولای شو.

## عملي ډګر او کارونه

د لووا پروګرامونه یوه منځني حالت (بایټ کوډ) ته تالیف کېږي چې د هر دریځ لپاره د لووا په خپل مجازي ماشین ځغلي. لووا د سي له پروګرامونو سره له نږدې د کار کولو په موخه جوړه شوې، او خپله ډیټا د ډېرې په وسیله انتقالوي.

په ژبه کې د ځغلمهاله[1] پېکجونو له امله، لووا زیاتره وختونه کارپالي پوستغالو ته ور اضافه کېږي تر څو د پوستغالي د پراختیا او سمون لپاره اړیکځای چمتو کړي. دا خبره په ځانګړي ډول ټپمونو لکه د جنګ نړۍ[2]، کې صدق کوي.

## اضافي لوست:

1- Ierusalimschy, Roberto. *Programming in Lua.* 2nd ed. Rio de Janeiro, Brazil: Lua.org, 2006.

2- Jung, Kurt, and Aaron Brown. *Beginning Lua Programming.* Indianapolis: Wiley, 2007.

3- Lua.org. Available online. URL: http://www.lua.org. Accessed September 26, 2007.

***

[1] runtime
[2] The world of Warcraft

# (۱۸)

## پاسکال (Pascal)

د ۱۹۶۰ کلونو په لومړیو وختونو کې د کمپیوټر پوهانو لویه اندېښنه د داسې لارو موندل وو چې خپل پروګرامونه په غوره ډول منظم او ورغوي. په اصل کې تر یو څه بریده داسې یوه ژبه (وګورئ: الګول) جوړه شوې هم وه چې د پروګراملیکنې روغ اصول یې چمتو او پلي کوي، چې د کنټرول جوړښتونو کارونه هم پکې یادولی شو. خو الګول د ډېټا ډولونو او نورو هغو ځانګړنو له کمښت سره مخ وه چې د عملي پروګراملیکنې لپاره ورته اړتیا وه، په ورته وخت کې له شک پرته د یوې نښې ښوونیزې ژبې لپاره خورا پېچلې او بې ثباته ژبه هم وه.

نیکولس ویرت په ETH یا د سویس د ټکنالوژۍ په فدرالي اداره کې د ۱۹۶۰ کلونو په اوږدو کې له یوې کمېټې سره کار کاوه چې په دې هڅه کې وه چې یوه داسې ژبه رامنځته کړي چې د الګول پر ستونزو برلاسي وي او نومورې ژبه د کمپیوټر جوړوونکو او ګټې اخیستونکو لپاره راښکونکې او عملي ژبه اوسي. خو له یو څه وخت وروسته، ورت د کمېټې له مبهمو پایلو څخه ناخوښه شو، او د یوې نوې ژبې، پاسکال، په جوړولو یې لاس پورې کړ، او په ۱۹۷۰ کې یې د ژبې خصوصیات اعلان کړل.

د پاسکال اصلي تکلوری پر الګول ورغلی وو او ورسره یې دا ژبه پراخه کړې هم وه. د تورو، بولین، او د ډېټا ډولونو لپاره د ملاتړ په چمتو کولو، پاسکال کټه اخیستونکو ته دا اجازه هم ورکوله چې د ځانې ډېټا ډولونو په ترکیبولو نوي ډېټا ډولونه جوړ کړي. دا ځانګړنه په ځانګړې توګه د «ریکارډ» ډېټا ډول جوړولو لپاره ګټوره ده چې، د مثال په ډول، کېدای شي د یوه کارمند نوم او د دندې عنوان (توري)، پېژندګنه[1] (اوږد عدد)، او معاش (یو اعشاري عدد) سره یو ځای او ترکیب کړي. په دقیق ډول د ډېټا ډولونو کارونه د پروسو له رابللو او تعریفولو تر روش پورې هم ادامه کولای شي.

پاسکال د غوره تعریف شوې ژبې په وړاندې کولو چې الګوریتمونه پکې په څرګند ډول تشریح کېدای شوی، په لویه کچه د کمپیوټر پوهانو او ښوونکو پام ځان ته را وکاږه. د پاسکال په مقبولیت کې د نوموړې ژبې د تالیفوونکو نوښتګر ډیزاین هم لوی لاس درلود. د هغه وخت د ماشین-اړونده[2] تالیفوونکو پر خلاف، د پاسکال تالیفوونکي په مستقیم ډول ماشین کوډ نه رامنځته کاوه. بلکې، د نوموړي راکړیز «P-code» وو، چې یو ډول انتزاعي کمپیوټري ژبه وه. یو ځغلمهاله سیستم چې د هر کمپیوټر لپاره ېپل لیکل شوی وي، P-code ژباړي او ورپسې د ماشین مناسب هدایات اجرا کوي. دا په دې معنا وه چې د پاسکال تالیفوونکی د یوه ځانګړي ماډل له کمپیوټر څخه ډېر په اسانه د بل ماشین لپاره د P-code

1 ID number
2 machine-specific

ترجمان په لیکلو، هغه بل ماشین ته وړل کېدای شوی. همدا ستراتیژي دوه لسیزې وروسته د وېب کارپالانونو لپاره د نومیالۍ ژبې د جوړوونکو له خوا وکارول شوه. (وګورئ: جاوا)

## د پاسکال د پروګرام جوړښت

لاندینی ساده پروګرام په پاسکال ژبه کې د یوه پروګرام اساسي جوړښت تشریح کوي. (په پرېړمخ لیکل شوي کلمات، اړین-ویوکي دي چې د پروګرام د جوړښت لپاره کارېږي). پروګرام په Type برخه پیل کېږي چې د ګټې اخیستونکي ډېټا ډولونه وضع کوي. په دې برخه کې کتارونه، ټولګي، ریکارډونه (مرکب ډولونه چې د ډېټا ېلا ېل اساسي ډولونه لرلای شي) لري. په دې ځای کې تر لسو پورې د (کاملو اعدادو) یو کتار تعریفېږي چې IntList بلل کېږي.

د Var یا متحول برخه د پروګرام ځانګړي متحولونه وضع کوي. متحولونه کولای شو چې یا د ژبې د ځاني ډېټا ډولونو په مرسته تعریف کړو (لکه عدد) او یا یې د هغو ډولونو په مرسته تعریف کړو چې ترمخه په Type برخه کې تعریف شوي دي. د پاسکال یوه مهمه ځانګړنه دا ده چې مخکې له دې چې په متحولونو کې وکارول شي، باید د ګټې اخیستونکي ډېټا ډولونه تعریف شي. په همدې ډول متحولونه باید وړاندې له دې چې په پروګرام کې وکارول شي، باید وضع شي. ځینې پروګرام لیکونکي دا پېچلتیا یو څه محدودوونکې ګڼي، خو په اصل کې دا محدودیت، د مثال په ډول، له یوې ټایپوګرافیکي (چاپي) تېروتنې مخه نیسي،

لکه یوه متحول ته د اړتیا په ځای کې د یوه بل ناتعریف شوي متحول شتون. نن ورځ ډیری ژبې تر استعمال دمخه د متحولونو په وضع کولو ټینګار کوي.

*begin* کلمه کوډ، د پروګرام اجرایي برخې ته ور پیژني. د څرخ لپاره د اړتیا وړ متحولونه لومړی د صفر د یوې وېلیو په ټاکنه پیل[1] کېږي. پام مو وي چې په پاسکال کې := (کالن او د مساوي نښه) د وېلیوګانو د ټاکلو لپاره استعمالېږي. باندینی if عبارت د دې لپاره استعمال شوی چې کټه اخیستونکی د توکونو ناسم شمېر درج نه کړي. ورپسې د for څرخ ورکړیزه وېلیو لولي، په کتار کې یې په خپل ځای، ځای پر ځای کوي، او ټولټال ذخیره کوي. هماغه ټولټال حاصل، بیا د اوسط د محاسبه کولو لپاره کارېږي، چې په پای کې د writeln عبارت په مرسته راکړیز راوباسي.

```
program FindAvg (input, output);
     type IntList = array [1 . . 10] of
     integer;
     var
        Ints: IntList;
        Items, Count, Total, Average: integer;
begin
     Average := 0;
     Total := 0;
     Readln (Items);
If ((Items > 0) and (Items <= 10)) then
     begin
```

[1] initialize

```
    for Count := 1 to Items do
      begin
      readln (Ints [Count]);
      Total := Total + Ints [Count]
      end;
    Average := Total / Items;
    Writeln ('The average of the items is:',
    Average)
    end
else
    Writeln ('Error: Number of items must be
    between 1 and 10')
end.
```

## د پاسکال اغیز

پاسکال په سوداګریزه برخه کې په منځنۍ کچه بریالۍ ژبه وه. د پي-کوډ فکر د UCSD پي-سیستم له خوا چې په کالیفورنیا پوهنتون، سانډیاګامو کې رامنځته شو، راخپل کړای شو. د ۱۹۷۰ کلونو په پای او د ۱۹۸۰ کلونو په پیل کې، پي-سیستم د پاسکال د جوړښتواله پروګراملیکنې ګټې د کمپیوټرونو ګټه اخیستونکو ته په لاس کې ورکړې لکه ایپل-۲، چې د دې کسانو لپاره لومړنۍ بدیلونه یوازې د ماشین ژبه او یا د بېسیک یو نارغېدلی ورژن وو. د ۱۹۸۰ کلونو په را وروسته وختونو کې بورلېنډ انټرنېشنل له ټربو پاسکال سره صحنې ته راووت. دې تالیفوونکي د پي-کوډ پر ځای له مستقیم تالیف څخه ګټه اخیسته چې په دې ډول پاسکال خپله چټکیتا او اغېزمنتیا له لاسه ورکړه. نوموړي تالیفوونکي د پروګراملیکنې مختلط کاري چاپېریال درلود چې د سافټوېر جوړول

یې د پخوانیو درنو او ګرانبیه تالیفوونکو لکه د مایکروسافټ تالیفوونکی، په پرتله ارزانه او اسانه کړل. ټربـــو ډېر زیات شهرت وموند او په لنډ وخت کې یې هغه ژبنۍ زیاتونې په خپل ځان کې را پورته کړې چې د استازواله پروګراملیکنې ملاتړ یې کاوه. خو پاسکال تر ټولو زیات نوم له خپل هغه رول څخه ترلاسه کړ چې د پروګراملیکنې د ښوونې او الګوریتمونو د څرګندولو لپاره د لومړنۍ ژبې په توګه وپېژندل شوه.

خو په ۱۹۹۰ کال کې د بریا اوج د سي او سي پلس پلس په برخه شو. دغو ژبو د پاسکال په پرتله له ډېر کریپټک[1] نحوي ترکیب څخه ګټه پورته کوله او د ډېټا ډولونو په اړه د پاسکال له میکانیزم څخه هم تشه وه. سیسټمي پروګرام لیکونکو سي ژبې ته ځانګړی لومړیتوب ورکاوه چې کولای یې شوای په دې ژبه کې له ماشین سره له نږدې کار ترسره کړي او د ډېټا ډولونو له تریفولو پرته یې په حافظه کې د لاسوهنې وړتیا په لاس کې ورکوله. سي له دې امله چټک پرمختګ وکړ چې لومړني ډوبلپران یې د یونیکس د مهمو ډوبلپرانو له ډلې څخه وو، چې هغه مهال د کمپیوټري چاپېریالونو په کښپس کې یو ډېر نوموتی عامل سیسټم وو.

د ۱۹۹۰ کلونو په اوږدو کې، سي، سي پلس پلس او جاوا د کمپیوټرپوهنې د تدریس لپاره د پاسکال ځای نیول پیل کړل. پاسکال د جوړښتواله پروګراملیکنې د مفاهیمو په رابرسېره کولو او د کمپیوټر پوهانو د یوه نسل په ښوونه کې په مرسته

[1] Cryptic

سربېره، وکولای شو چې د کمپیوټرپوهنې په ډګر کې یو اوږدمهاله اغېز پرېږدي. ورت د ماډولا-۲ او اوبیرون په رامنځته کولو خپل کار ته دوام ورکړ، خو دا دواړه ورژنونه په اساسي ډول تر علمي نړۍ پورې محدود وو. پاسکال د اپډا په جوړولو کې هم پرېمانه اغېز درلود، هغه ژبه چې د امریکې د فدرالي دولت له لوري هم ومنل شوه او جوړښتواله پروګراملیکنه یې د استازواله پروګرملکنې له ځانګړنو سره ترکیب کړې وه، او د کوچنیو پروګرامونو د پراخه پیکجونو د مدیریت لوبه وړتیا یې لرله.

## اضافي لوست:

1- Free Online Pascal and Delphi Tutorials and Documentation. Available online. URL: http://www.thefreecountry.com/ documentation/onlinepascal.shtml. Accessed August 17, 2007.
Free Pascal Compiler. http://www.freepascal.org/

2- Jensen, Kathleen, Niklaus Wirth, and A. Mickel. *Pascal User Manual and Report: ISO Pascal Standard.* 4th ed. New York: Springer-Verlag, 1991.
3- Koffman, Elliot B. *Turbo Pascal.* 5th update ed. Reading, Mass.: Addison-Wesley, 1997.

4- Rachele, Warren. *Learn Object Pascal with Delphi.* Plano, Tex.: Wordware Publishing, 2000.

5- Wirth, Niklaus. *Programming in Modula-2.* 3rd, corr. ed. New York: Springer-Verlag, 1985.

***

# (۱۹)

# پرل (Perl)

د نړیوال جال چاودېدونکي پرمختګ پروګرام لیکونکي له دې اړتیا سره مخامخ کړل چې داسې لارې ولټوي چې ډېټابېسونه او نورې شته سرچینې له ویبسایټونو سره وصل کاندي. د دې ډول اتصال خصوصیات د عامه-ګېټوې په اړیکځای[1] کې موندل کېږي. خو، د CGI متونو د لیکلو لپاره لومړنۍ اسانتیاوې بېکاره او د کارولو په برخه کې ستومانوونکې وې.

په ۱۹۸۶ کې د یونیکس جوړوونکي «لېري وال» یوه ژبه جوړه کړه چې پرل (د عملي استخراج او راپور ژبه)[2] نومېدله. د ساده ډېټا پروسېس د متنونو د لیکلو لپاره هغه مهال هم لارې وې او دغه راز په نقشو کې د لاسوهنې، لاسي ژبو هم شتون درلود. خو، وال غوښتل چې د ډېټا د فارمټ کولو، استخراجولو او لټون لپاره په لوی شمېر ډول ډول فنکشنونه او تخنیکونه چمتو کړي. پرل د یونیکس د ټولنې دننه د ډېرو کسانو پام ځان راته ټول کړ. څنګه چې د ۱۹۹۰ کلونو په منځ او پای کې ډېرې انټرنېټي پوستغالي د یونیکس سیسټمونو په اساس جوړېدل، نو دا طبیعي وه چې د وېب ماهرین او د کاریالونو جوړوونکي د خپل CGI متنونو د لیکلو لپاره پرل ته راوګرځي.

---

[1] Common Gateway Interface

2 Practical Extraction and Report Language

د یونیکس د ډیزی متنواله ژبو په څیر، د پرل نحوي ترکیب هم سي ژبې ته ورته وو. خو د سي ژبې تر شا فلسفه دا وه چې بنیادي ژبه یې له دا ډول کړنو او فعالیتونو څخه برخمنه وه چې د معیاري او اضافي[1] پروګرامونو له لارې سمبالېده. بل خوا، پرل، خپلې زیاتره کړنچارې[2] د یونیکس له مرستیالو پوستغالو څخه پیلوي، لکه sed یا اصلي ایډیټور، سي شېل، آک، او هغه منظمې افادې چې د یونیکس ګټه اخیستونکي ورسره بلدتیا لري. په دې ژبه کې د «hash» په نوم ډېټا ډول (د جوړه کیلیانو او وېلیوګانو یوه ټولګه) شتون لري چې په پروګرام کې د تغیراتو راوستل اسانه کوي او د انټرنېټ د هوستونو او د هغوی د انټرنېټي پروټوکولونو ادرسونه ګوري.

وال به د پرل ګټه اخیستونکو ته د هغوی د غبرګون ځواب د نوو ځانګړنو او فنکشنونو په اضافه کولو ورکاوه او د ګټه اخیستونکو له دې نیوکو څخه یې استفاده کوله. د وال تګلاره دا وه چې له پروګرام لیکونکو سره د امکان تر بریده عملي مرسته وکړي، نه دا چې په دې اړه اندېښنه وکړي چې ژبه غوره تعریف شوې او ثابته اوسي، او یا دا چې د زده کړې لپاره اسانه ده او که نه. د مثال په ډول، په ډېرو ژبو کې، د دې لپاره چې یو کار ترسره شي، که چېرته یو شرط صحیح نه وي، نو یو نفر داسې لیکي:

```
If ! (test for valid data)
```

[1] add-in
[2] funcationality

```
Print Error-Msg;
Else Process_Data;
```

خو په پرل کې کولای شي چې د (! if) پر ځای د (unless) عبارت وکاروي، چې هغه به په دې ډول وي:

```
Unless (Test for invalid data) {
Process_Data;
}
```

د نحوي ترکیب له مخې، د unless عبارت له دې پرته بل څه نه کوي چې یو if او else یې په کې کوي، او دا د یوه مختلف جوړښت زده کړه درکوي. د دې یوه عملي کټه دا وه چې پروګرام یې د لوستلو لپاره یو څه اسانه کړ چې دلته ډېر ټینګار په دې وو چې پروګرام د کوم کار کولو شیمه لري او که نه، نه دا چې پروګرام کې خو به کومه تېروتنه نه وي. په ورته ډول، پرل د until څرخ هم وړاندیزوي:

```
Until (Condition is met) {
Do something;
}
In C, one would have to say
While (Condition is not met) {
Do something;
}
```

د ژبې ځانګړنو ته د دې «هرکاره» تګلارې زیاتولو نوموړې ژبه د ځینو کمپیوټر پوهانو له خوا تر نیوکې لاندې راوستله، چې دا تګلاره نامنظمې او نه تصدیق

کېدونکې پروګراملیکنې ته لاره هواروي. خو د پرل ډېرې مینوال دې ژبې ته د یوې هر اړخیزې ژبې او د ویب پروګراملیکنې د ننګونکې نړۍ لپاره د یوې اساسي ابزارټولګې په سترګه ګوري. څومره چې د ۱۹۹۰ کلونو په وروستیو کې ژبې وده کوله، په همغه ډول د استازواله پروګراملیکنې ځانګړنې هم ور زیاتېدې.

## د پرل د پروګرام بېلګه

لاندې ډېر ساده کوډ د پرل داسې پروګرام تشریح کوي چې له یوه فایل څخه د ډېټا یو څو کرښې ور اخلي او بیا یې پرنټ کوي. لومړۍ کرښه یونیکس ته وایي چې د پرل ژباړونکی اجرا کړي. د فایل نوم data.txt د متنکرښې متحول یا string variable چې $file دی، ته اساین کوي. ورپسې فایل پرانیستل کېږي او د INFO متحول ته اساین کېږي. یوازې یو عبارت (نه یو څرخ) د دې لپاره بسنه کوي چې د فایل ټولې کرښې د @lines کتار ته ور اساین شي. د «foreach» عبارت د for څرخ یو غوندې حالت دی چې د کتار هره کرښه د متنکرښې متحول $line ته ور اساین کوي او بیا یې د HTML په بڼه د پردې پر مخ پرنټ کوي.

```
#!/usr/local/bin/perl
$file = 'data.txt';
open(INFO, "<$file" ) ;
@lines = <INFO> ;
foreach $line (@lines)
{
print "\n <P> $line </P>" ;
```

```
}
# DONE
```

که څه هم اوسمهال د انټرنټي پوستغالو جوړولو لپاره د نورو متنواله ژبو له خوا یو څه کم رنګه شوې، خو پرل یوه بالغه ټکنالوژي ده چې اوس هم په لویه کچه کارول کېږي، په ځانګړي ډول د ډېټا د استخراج، اړولو، او لاسوهنې لپاره. د پرل د ټولیز ارشیف شبکه (CPAN)[1] له ۱۲۰۰۰ څخه زیات ماژولونه لري چې په وړیا توګه د پروګرام لیکونکو لپاره د لاسرسي وړ دي.

**اضافي لوست:**

1- Comprehensive Perl Archive Network (CPAN). Available online. URL: http://www.cpan.org/. Accessed August 17, 2007.

2- Conway, Damian. *Perl Best Practices.* Sebastapol, Calif.: O'Reilly, 2005.

3- Lee, James. *Beginning Perl.* 2nd ed. Berkeley, Calif.: Apress, 2004.

4- Schwartz, Randall L., Tom Phoenix, and Brian D. Foy. *Learning Perl.* 4th ed. Sebastapol, Calif.: O'Reilly, 2005.

5- Wall, Larry, Tom Christiansen, and Jon Orwant. *Programming Perl.* 3rd ed. Sebastapol, Calif.: O'Reilly, 2000.

***

1 Comprehensive Perl Archive Network

# (۲۰)

## پي ايچ پي (PHP)

پي ايچ پي له پراخ شهرت برخمنه متنواله ژبه ده چې په بنسټيز ډول د خوځنده[1] ويبپاڼو د جوړولو لپاره استفاده کېږي. پي ايچ پي په ۱۹۹۴ کې د ډنمارکي پروګرام ليکونکي راسموس لرډورف له خوا رامنځته شوه تر څو د پرل د يو ډله متنونو پر ځای د خپلې ويبپاڼې د مديريت لپاره کټه تړې واخلي – او نوم يې هم له همدې ځايه سرچينه اخيستې، «شخصي ويبپاڼه يا personal home page». لرډورف په ۱۹۹۵ کې د ژبې لومړنۍ ورژن له «قالب ژباړونکي يا form interpreter» سره يو ځای را وايست. په ۱۹۹۷ کې د ژبې تجزيه کوونکی[2]، دوو اسرائيلي ډوبلپرانو زيف سوراسکي او آنډي ګوتمان وليکه چې په ۱۹۹۸ کې يې PHP3 بازار ته وړاندې کړه؛ او تر دې مهاله پورې د PHP لومړني توري په خپله بېرته د PHP معنا وړاندې کوله: Hypertext Processor يا د مافوق متن پروسېس کوونکی. په ۲۰۰۴ کې اوسنۍ نسخه يعنې PHP5، رامنځته شوه. څومره چې ژبې وده کوله، هماغومره يې د استازو لپاره ملاتړ غوره والی مونده او همدارنګه له MySQL او نورو ډېټابېسونو او د انټرنېټي کاريالونو همغږه ټکنالوژيو سره يې اړيکو او ارتباط موندنې هم پرمختګ کاوه.

---

[1] dynamic
[2] parser

عموما پي ايچ پي پر يوه ويب سرور ځغلي او د پي ايچ پي کوډ پروسس کوي چې ډېر کله له يوې ويب پاڼې سره مختلط شوی وي. د Hello World يا سلام نړۍ کلاسيک پروګرام په پي ايچ پي کې داسې ښکاري:

```
<? php
echo "Hello, World";
?>
```

د پي ايچ پي پروسس کوونکی پورتنی کوډ يوازې د محدودوونکو <? او ?> دننه تجزيه کوي. د دې محدوودونکو معادل بيا په لاندې ډول دی:

```
<script language ='php'></script>
```

سربېره پر دې چې کوډ د HTML په پاڼو کې مختلط وي، پي ايچ پي په کمانډکرښه (command line) کې په تعاملي ډول هم کارول کېدای شي، چېرته چې دې ژبې د آک، پرل او شېل په څېر متنواله زړو ژبو ځای نيولی دی. پي ايچ پي د ګټې اخيستونکي د اړيکځای له لايبرېريانو سره هم وصل کېدای شي (لکه GTK+ د لېنکس/يونيکس لپاره) څو داسې پوستغالي جوړ کړو چې د سرور پر ځای د کلاينټ په ماشين باندې وځغلي.

پي ايچ پي د بنسټيزو ډېټا ډولونو يوه مجموعه او ورسره يو بل ډېټا ډول چې «resource» نومېږي، لري، چې هغه ډېټا وړاندې کوي چې د هغو ځانګړو فنکشنونو په مرسته پروسس شوي وي چې انځورونه، متني فايلونو، د ډېټا ريکارډونه او داسې نور ېرته راګرځوي. د دې ترڅنګ، پي ايچ پي-۵ د شخصي

او خوندي، غړو متحولونو، کنسټرکټرونو او ډیکنسټرکټرونو او د نورو هغو ځانګړنو په ګډون چې په سي پلس پلس او نورو ژبو کې تر سترګو کېږي، د استازو لپاره هم بشپړ ملاتړ چمتو کوي.

د پرانیستو سرچینو د استازو او فنکشنونو کڼ شمېر لایبریانې په پي ایچ پي کې شتون لري چې د پي ایچ پي متنونو ته اجازه ورکوي تر څو عادي انټرنټې فعالیتونه ترسره کړي، لکه، د ډېټابېس سرورونو (لکه MySQL) ته لاسرسی، او همدا رنګه د مشهورو انټرنټي فارمټونو د سمبالښت لپاره ژبنۍ زیاتونې چمتو کوي لکه د اډوب فلش خوځښتونه او داسې نور. پروګرام لیکونکي د PEAR[1] (د پي ایچ پي د زیاتونو او کاریالونو زېرمه) له لارې د پي ایچ پي لوی شمېر سرچینو ته لاسرسی موندلی شي.

د اساسي ځانګړنو او په اسانه د متنوالۍ تعامل پي ایچ پي ژبه د ډېرې هغو وېب ډوېلپرانو لپاره د خوښې وړ ژبه ګرځولې چې د ټکنالوژیو د ډلې چې LAMP نومېږي، د یوه غړي په توګه ګټه تري پورته کوي. LAMP په اصل کې د لینکس، اپاشې (وېب سرور)، MySQL (ډېټابېس) او پي ایچ پي لنډیز دی.

[1] PHP Extension and Application Repository

## اضافي لوست:

1- Achour, Mehdi, et al. *PHP Manual.* Available online. URL: http://www.php.net/manual/en/. Accessed November 7, 2007.

2- Lerdorf, Rasmus, Kevin Tatroe, and Peter MacIntyre. *Programming PHP*. 2nd ed. Sebastapol, Calif.: O'Reilly Media, 2006.

3- PEAR-PHP Extension and Applications Repository. Available online. URL: http://pear.php.net/. Accessed November 7, 2007.

4- PHP [official Web site]. Available online. URL: http://php.net/. Accessed November 7, 2007.

5- Zandstra, Matt. *PHP Objects, Patterns, and Practice.* Berkeley, Calif.: Apress, 2004.

***

# (۲۱)

## پي اېل-۱ (PL/1)

له ۱۹۶۰ کلونو دمخه، دوه پروګرامي ژبې تر ټولو زياتې تر استعمال لاندې وې: فورټران د ساينسي او انجينري کاريالونو لپاره او کوبال د کاروباري محاسبو لپاره. خو کاريالونو ورځ تر بله پراختيا موندله او لا زيات پيچلي کېدل، او په دې اساس د لوی شمېر ډول ډول وړتياوو غوښتنه يې کوله. د مثال په ډول، ساينټيفيک پروګرام ليکونکو اړتيا لرله چې د محاسبې ترڅنګ د ډېټا پروسېس او راپور جوړونې وړتياوې چمتو کړي. بل خوا، کاروباري پروګرام ليکونکو، ډېره زياته اړتيا لرله چې له فورمولونو او احصائيو سره کار وکړي او همدا رنګه اعشاري عددونو او د عددونو نورو ډولونو ته يې اړتيا ليدله.

په دې اساس د ژبو ډوبلپرانو د يوې داسې عام-هدفه ژبې جوړولو ته مټې راونغښتې چې د کلماتو، شمېرو او ډېټا فايلونو لپاره مشترک بستر چمتو کړي. په ورته وخت کې، آی بي اېم شرکت د کاروباري او ساينسي فعاليتونو لپاره د پلو کمپيوټري سيسټمونو ځای هر اړخيزه کمپيوټرونو «سيستم/۳۶۰» ته ورکړ. آی بي اېم شرکت او د دوی د کټه اخيستونکو يوه ډله Share سره يو ځای شول تر څو د دې نوي ماشين لپاره نوې ژبه جوړه کړي.

په لومړيو کې ډيزاين کوونکو فکر وکړچې د غوره متني او ډېټا پروسېس د وړتيا لپاره فورټران ژبې ته پراختيا او وده ورکړي، نو دوی فورټران-۴ د نوې ژبې د

رامنځته کولو لپاره تر کار لاندې ونیوله. خو ډېر ژر د دوی ټول تمرکز په بشپړ ډول د نوې ژبې ډیزاین کولو ته واوښت، چې تر ۱۹۶۵ز پورې د «نوې پروګرامي ژبې» یا NPL[1] په نوم یاده شوه. دا چې دا لنډیز له وړاندې څخه لا د برتانیې د ملي فزیکي لابراتوار (National Physical Laboratory) لپاره استفاده کېده، نو د ژبې نوم په PL/1 یا «لومړنۍ پروګرامي ژبه» بدل شو.

## د ژبې ځانګړنې

پي ایل/۱ د «هرکاره» ژبې توګه پېژندل کېږي، ځکه چې د پروګرام لیکونکو لپاره ډېر زیات داسې مشخصات چمتو کوي چې له بېلا بېلو سرچینو څخه اخیستل شوي وو. اساسي بلاک جوړښتونه او کنټرول جوړښتونه له الګول څخه اخیستل شوي، چې یوه نستاً کوچنۍ ژبه وه چې د کمپیوټر پوهانو له خوا د جوړښتواله پروګراملیکنې د یوه ماډل په توګه وړاندیز شوې وه، او پاسکال ژبې ته ورته وه (وګورئ: پاسکال). بلاکونه په یو بل کې اغږل کېدای، او هغه متحولونه چې د یوه بلاک دننه وضع شوي وي، یوازې د هماغه بلاک دننه او د نورو ټرلو بلاکونو دننه د لاسرسي وړ وي، که نه بله لاره یې دا ده چې په نړیوال ډول وضع شي.

پي ایل/۱ د ډېټا ډولونو پربمانه شمېر په ځان کې رانغاړي، تر دې چې د عددي ډېټا لپاره د یوه عدد د رقمونو شمېر هم مشخصولای شي. د PICTURE عبارت کوبال ژبې ته ورته د یوې ټاکلې اډانې د تعینولو لپاره استفاده کېدای شي. خو

1 New Programming Language

نوموړې ژبه د الګول او پاسکال په پرتله ډېر پراګماتیک (واقع بینانه) چلند کوي؛ یعنې اړتیا نشته چې ډېټا یو ځل وضع شي، بلکې عادي ځانګړنې د متن له مخې ورته غوره کېږي. راکړېز/ورکړېز (I/O) په خارجي لایبرېري کې د چمتو کېدو پر ځای د ژبې دننه جوړېږي، او د ژبې انعطاف منونکي آپشنونه توري، د تورو کرښې، بلاکونه او د ترتیبي او یا ناڅاپي لاسرسي ریکارډونه دي.

په عموم کې، پي اېل/۱ د الګول یا وروستنیو ژبو لکه د سي په پرتله د ماشین د ټیټې کچې په فعالیتونو باندې هم کنټرول په لاس کې درکوي. د مثال په ډول، دا چې متحولونه په څه ډول ذخیره کېږي، له ستاټیک (چې د پروګرام تر پایه پورې موجود وي) څخه نیولې تر اتوماټیک (د بلاکونو له درج او ایستلو سره په مشترک ډول، یا ځای نیسي او یا ځای پرېږدي) او Controlled پورې (چېرته چې حافظه باید په ښکاره ډول ونیول شي او یا تشه کړای شي)؛ په دې ټولو برخو کې غیرعادي کنټرول چمتو کوي. نښه کوونکي د حافظې موقعیتونه د دې جوګه کوي چې په مستقیم ډول لاسوهنه پکې وشي. پي اېل/۱ د استثناټېروتنو (exceptions) د سمالښت لپاره پرېمانه اسانتیاوې لري چې دا ډول تېروتنې د هارډویر له شرایطو، ریاضیکي معادلو، د فایلونو له سمبالښت او یا نورو شرایطو څخه را ټوکېدلای شي.

## د پروګرام بېلګه

لاندې پروګرام یو Do څرخ اجرا کوي او له یو څخه نیولې تر ټاکلو توکونو پورې اعداد شمېري. ورپسې د اعدادوو د جمعې حاصل او اوسط را وباسي.

```
COUNTEM: PROCEDURE OPTIONS (MAIN);
DECLARE (ITEMS, COUNTER, SUM, AVG) FIXED;
ITEMS = 10;
SUM = 0;
DO COUNTER = 1 TO ITEMS;
     SUM = SUM + COUNTER;
END;
AVG = SUM / ITEMS;
PUT SKIP LIST ("TOTAL OF ");
PUT ITEMS;
PUT ("ITEMS IS ");
PUT TOTAL;
PUT SKIP LIST ("THE AVERAGE IS: ");
PUT AVG;
END COUNTEM;
```

## د ژبې اغېز

د خپلو عملي ځانګړنو او د آی بي اېم شرکت د نومیالیو ۳۶۰ مېن فریم کمپیوټرونو لپاره د موجودیت له کبله، پي اېل/۱ د ۱۹۶۰ او ۱۹۷۰ کلونو په پای کې د پام وړ بری تجربه کړ. وروسته یاده ژبه ډېری لوړو درېځونو او عامل سیسټمونو ته هم انتقال شوه. کله چې شخصي کمپیوټرونه منځته راغلل، پي اېل/۱ د آی بي اېم شرکت د OS/2 عامل سیسټم او دغه راز د مایکروسافټ د ډاس او وینډوز

لپاره هم موجوده وه، خو په دې چاپېریالونو کې د نوموړې ژبې رښتینی بری په برخه نه شو.

د جوړښتواله پروګراملیکنې د سرلاري اډسکر ډیټکسټرا په څېر کمپیوټر پوهانو پې اېل/۱ د واضح او ښه-رغېدلي جوړښت د نشتون له کبله رسوا کړه. نوموړي په ۱۹۷۲ کې د ټیورینګ جایزې د ترلاسه کولو پر مهال په خپل لېکچر کې وویل چې: «زه خو یې اصلاً نه وینم چې موږ به د خپل ذهني چوکاټ دننه د خپلو پروګرامونو وده په خپل ځای وساتلای شو، کله چې زموږ پروګرامي ژبه له دومره زیاتو ځانګړنو سره – ګوزاره مې وکړه! دمخه لا زموږ له ذهني کنټرول څخه په ټښته کې ده.»

په عملي ډګر کې په یاده ژبه کې د ګڼ شمېر ځانګړنو شتون په دې معنا وو چې د نوموړې ژبې سمه زده کړه، یوه اوږده پروسه ده. بل خوا د سي په څېر یوه ساده ژبه ده چې زده کړه یې اسانه ده، که څه هم هر اړخیزه نه ده. پې اېل/۱ هڅه وکړه چې د آی بي اېم شرکت په کمپیوټرونو کې خپل ځای له لاسه ور نه کړي، په داسې حال کې چې سي، د کوچنیو کمپیوټرونو او یونیکس ټولنې په نړۍ کې وده وکړه او ځان یې د شخصي کمپیوټرونو لپاره ډېر وړ ثابت کړ. پې اېل/۱ ډېرې ښېلګې چمتو کړې چې د ژبو ډیزاین کوونکي یې د غوره ډیزاین د تطبیق لپاره استفاده تري کولای شي.

## اضافي لوست:

*1- The Essentials of PL/I Programming Language.* Piscataway, N.J.: Research and Education Association, 1993.

2- Hughes, Joan Kirby. *PL/I Structured Programming.* 3rd ed. New York: Wiley, 1986.

3- "PL/I Frequently Asked Questions (FAQ)." Available online. URL: http://www.faqs.org/faqs/computer-lang/pli-faq/. Accessed August 17, 2007.

4- "The PL/I Language." Available online. URL: http://home.nycap.rr.com/pflass/pli.htm. Accessed February 9, 2008.

5- Sebesta, Robert W. *Concepts of Programming Languages.* 8th ed. Boston: Pearson Addison-Wesley, 2007.

***

# (۲۲)

## پرولوګ (Prolog)

له ۱۹۵۰ کلونو را په دېخوا، څېړونکي د عقلي چلند د اټوماټیک کولو امکان ته هوسېدل چې د بېلګې په ډول منطقي استنباط تري یادولای شو. ځکه شمېر بېلګیز پروګرامونه ولیکل شوو تر څو له قضیو نیولې تر فرضیو پورې تیورۍ په ثبوت ورسوي. په ۱۹۷۲ کې فرانسوي څېړونکي «الېن کولمیراور» او «رابرټ کولاسکي» په اېډنبرګ پوهنتون کې یوه منطقي پروګرامي ژبه جوړه کړه چې پرولوګ بلل کېږي (چې د منطقي پروګراملیکنې لنډیز وو) څو د اټوماټیک استدلال او پوهنبودنې[1] لپاره یوه عمومي لاره شتون ولري.

یو عادي پروسوي پروګرام د بېلا بېلو ډېټا توکونو په تعریفولو پیل کېږي، چې ورپسې په دې ډېټا کې د لاسوهنې لپاره د پروسو (فنکشنونو) یوه ډله راځي تر څو مطلوب نتیجه تري ترلاسه شي. بل خوا، د پرولوګ پروګرام، د حقایقو (فرضیو) په یوه ټولګه پېلېږي چې صحیح انګېرل کېږي. (دې ته کله کله بیانیه پروګراملیکنه هم ویل کېږي.)

د مثال په ډول، دا حقیقت چې جو د بیل پلار دی، په لاندې ډول لیکل کېږي:

```
Father (Joe, Bill).
```

[1] knowledge representation

په دې پسې پروګرام لیکونکی منطقي اصول تعریفوي چې پر دې حقایقو پلي کېږي. د مثال په ډول:

```
father (X, Y) :- parent (X, Y), is male (X)
grandfather (X, Y) :- father (X, Z), parent
(Z, Y)
```

په دې ځای کې لومړنۍ څرګندونه وایي چې x نومي یو تن د y پلار دی که چېرته هغه د Y له والدینو څخه وي او نر وي. دوهمه څرګندونه وایي چې X د Y نیکه دی، که چېرې هغه د Z په نوم د یوه تن پلار وي چې بېرته د Y پلار دی.

کله چې یو پروګرام ځغلي، نو همدا ډول ګروېرنې[1] یا څرګندونې چې د ربنتینولۍ ثبوت یې غوښتل شوی وي، پروسېس کوي. د unification په نوم د یوې پروسې په مرسته، د پرولوګ سیستم هغو حقایقو او اصولو ته ګوري چې پر ګروېرنو پلي کېږي او هغه کوي چې یو داسې منطقي زنځیر جوړ کړي چې د هغې ګروېرنې په اثبات رسولو ته لاره هواره کړي. که چېرې زنځیر مات شي (په دې چې هېڅ سمون لرونکی حقیقت یا اصول ونه موندل شي)، سیستم بېرته په شا راګرځي او په یوه بل مطابقت کوونکي حقیقت یا اصول پسې ګوري او له دې ځایه د یو بل زنځیر لپاره هڅه کوي.

پرولوګ د مصنوعي ځیرکتیا د څېړونکو ترمنځ د پام وړ مینوال پیدا کړل چې د اتوماتیک استدلال لپاره د دودیزو پروګرامي ژبو پر ځای د یوه واکمن بدیل د

[1] query

جوړولو په هیله وو. دا مینه په ۱۹۸۰ کلونو کې د جاپان د پنځم نسل په کمپیوټري پروګرام کې را وټوکېده چې د منطقي زبر کمپیوټرونو[1] د جوړولو په لټه کې وو او پرولوګ یې د خپلې خوښې وړ ژبه وټاکله. که څه هم ځینې ماشینونه جوړ شول، خو د دې نظریې هېڅکله رښتینی بری په برخه نه شو. بورلېنډ انټرنېشنل (د بریالي ټربو پاسکال جوړوونکي) ټربو پرولوګ را وایستله چې یاده ژبه یې د هغو زده کوونکو لپاره د لاسرسي وړ وګرځوله چې له شخصي کمپیوټرونو څخه ګټه پورته کوي، که څه هم په دې ژبه کې ځینې غیرمعیاري ژبنۍ زیاتونې استفاده کېدې.

که څه هم په سوداګریز ډګر کې یې بریا محدوده وه، خو پرولوګ د مصنوعي ځیرکتیا د څېړنو په لوی شمېر ډګرونو کې کارول شوې. د نوموړې پر اصولو ولاړ جوړښت د ځیرکو سیسټمونو، پوهنښتونو[2]، او طبیعي ژبې پروسېس لپاره مناسبه ژبه وه. یاده ژبه د هغو نمونوي (پروټایپي) سیسټمونو د ډیزاین لپاره هم کارول کېدای شي چې وروسته د چټکتیا او اغېزمنتیا لپاره په دودیزو ژبو کې بیاځلې لیکل (کوډ) کېدای شي.

**اضافي لوست:**

1- Bratck, Ivan. *Prolog Programming for Artificial Intelligence.* 3rd ed. Reading, Mass.: Addison-Wesley, 2000.

---

[1] supercomputer
[2] knowledge bases

2- Clocksin, W. F., and C. S. Mellish. *Programming in Prolog: Using the ISO Standard.* 5th ed. New York: Springer, 2003.

3- Fisher, J. R. "Prolog: Tutorial." Available online. URL: http://www.csupomona.edu/~jrfisher/www/prolog_tutorial /contents.html. Accessed August 17, 2007.

4- Sterling, Leon, and Ehud Shapiro. *The Art of Prolog: Advanced Programming Techniques.* 2nd ed. Cambridge, Mass.: MIT Press, 1994.

5- SWI-Prolog [free Prolog for Windows, Linux, and Mac OS]. Available online. URL: http://www.swi-prolog.org/. Accessed August 17, 2007.

***

# (۲۳)

# پایتان (Python)

د ګایډو وان روسم له خوا جوړه شوې او په لومړي ځل په ۱۹۹۰ کې وړاندې شوې ژبه، پایتان، یو څه اسانه خو ځواکمنه متنواله ژبه ده. پایتان نوم د بریتانیې د کومیډي له نومیالې ډلې «مونټي پایتان»[1] څخه اخیستل شوی.

پایتان په ځانګړي ډول د سیسټمي مدیریت کوونکو، وېب ماهرینو او نورو هغو خلکو لپاره ګټوره ژبه ده چې باید خپل بېلا بېل فایلونه، ډېټا سرچینې، او پروګرامونه د خپلو ورځینو چارو د ترسره کولو لپاره سره یو ځای کړي. نومورې ژبه اوسمهال لږ شمېر خو زیاتېدونکي (او ډېر لېواله) پلویان لري.

پایتان تر ډېره پورې دودیز نحوي ترکیب له منځه وړی چې د سي ژبو په کورنۍ کې کارېده. د مثال په ډول، لاندې کوچنی پروګرام د فارن هایټ درجه په خپل سلیوس معادل باندې اړوي:

```
temp = input("Farenheit temperature:")
print (temp-32.0) *5.0/9.0
```

د هغو سیمیکالنو او لیندیو د نشتون له کبله چې په سي او اړونده ژبو کې تر سترګو کېږي، پایتان تر ډېره ښسیک ژبې ته ورته بربښي. همدا رنګه دې ټکي

[1] Monty Python

ته مو هم پام وي چې اړتیا نشته د ورکړیزې ډېټا ډول بیان شي. ځغلمهاله میکانیزم په خپله له هغې افادې څخه پوهېږي چې د print په عبارت کې کارول شوي چې ډېټا یو عددي ډول دی. په دې ترتیب، د پایتان پروګرامونه د سي، جاوا او حتی پرل پروګرامونو په پرتله لنډ او ساده جوړښټ لري. ساده نحوي ترکیب او د ډېټا ډول نه درج کول په دې معنا نه ده چې پایتان یوه «جدي» ژبه نه ده، بلکې پایتان د استازواله پروګراملیکنې ټولې اسانتیاوې په خپل ځان کې را غونډې کړې دي.

د پایتان پروګرامونه په تعاملي ډول د کمانډونو په درج کولو په اسانه او چټک ډول لیکل کېدای او ورپسې نوموړی متن بایټ کوډ ته اړول کېدای شي. بایټ کوډ هغه خپلواک-ماشینه ښودنه ده چې د هر ډول ماشین د چاپېریال لپاره په ډیزاین شوي ترجمان ځغلېدای شي. د دې د بدیل په توګه، داسې ترجمان پروګرامونه هم شته چې د پایتان متن د سي ژبې سرچینه فایل ته اړوي چې وروسته په لوړ سرعت تالیف (کمپایل) کېدای شي.

پرل تر دې دمه د یونیکس او ویب اړونده کاریالونو لپاره یوه مشهوره متنواله ژبه ده. پرل له ځواکمنو ځانې افادو او pattern-matching میکانیزمونو څخه برخمنه ژبه ده او دغه راز ډېر نور فنکشنونو هم لري چې د عملي متنوالې لپاره ګټور پرېوځي. بل خوا پایتان، ډېره عمومي او په څرګند جوړښت رغېدلي ژبه ده چې د لوی شمېر کاریالونو لپاره یوه وړ ژبه او د لویو او پېچلو پوستغالو لپاره په اسانۍ سره د پراختیا وندنې له لویو وړتیاوو څخه برخمنه ده.

## اضافي لوست:

1- Lutz, Mark. *Learning Python.* 3rd. ed. Sebastapol, Calif.: O'Reilly, 2007.

2- ———. *Programming Python.* 3rd ed. Sebastapol, Calif.: O'Reilly, 2006.

3- Python Programming Language—Official Web site. Available online. URL: http://www.python.org/. Accessed August 17, 2007.

4- Zelle, John M. *Python Programming: An Introduction to Computer Science.* Wilsonville, Ore.: Franklin, Beedle & Associates, 2004.

***

# (۲٤)

# آرپي جي (RPG)

ډېرى کاروباري پروګرامونه چې د مېن فرېم کمپیوټرونو لپاره لیکل شوي وو، داسې وو چې له فایلونو څخه به یې ډېټا لوستله، نسبتاً ساده پروسې به یې ترسره کولې او یا به یې پرنټ شوي راپورونه د راکړېز په ډول برابرول. د ۱۹۶۰ کلونو په اوږدو کې ځینې خلک په دې باوري وو چې کوبال یوه عام-هدفه (خو کاروبارواله) کمپیوټري ژبه ده او د هغو کسانو لپاره اسانه ژبه ده چې مسلکي پروګرام لیکونکي نه دي. که څه هم د خبرې حقیقت په دې ډول نه وو، خو آی بي اېم شرکت د آر پي جي یا (راپور تولیدوونکي) په جوړولو بریالی شوو، هغه ژبه چې د پروګرام لیکونکو (د مبتدیانو په ګډون) لپاره په دې موخه جوړه شوې وه چې د کاروباري راپورونو تولیدول ورته اسانه کړي.

ډېرى کوبال پروګرامونه ډېټا لولي، ازموینې او محاسبې ترسره کوي او پایلې یې پرنټ کوي. «آر پي جي» چې د نوو مېن فرېم کمپیوټرونو، سیستم/۳۶۰ او کوچنیو کمپیوټرونو، سیستم/۳۶۰ لپاره جوړه شوه، د لومړي ځل لپاره په ۱۹۶۴ کې د استعمال لپاره وړاندې شوه. نومورې ژبه د کوبال پورته یاده پروسه ساده کوي او د پروګرام د کوډ ډېرى عبارات له منځه وړي.

د آر پي جي یو کلاسیک پروګرام د آر پي جي په دایروي جریان کې جوړېږي، چې درې پړاوونه لري. په ورکړېز پړاو کې، ورکړېز واحد(ونه)، فایل ډول، د

لاسرسي خصوصیات، او د ډیټا ریکارډ، جوړښت مخشص کېږي. (دا خصوصیات ښه زیات څرګند لیکل کېدای شي.) د پروګرام زړه هغه محاسبې مشخصوي چې باید په ډول ډول ډیټاساحو سرته ورسېږي، په داسې حال کې چې د راکړیز برخه دا ټاکي چې د پروګرام پایله به د راپور په ښه څنګه رامنځته کېږي چې د سرلیکونو، پالیکونو او نورو برخو لرونکې به وي.

د آر پي جي په راتلونکو نسخو کې ځانګړنې لا ډېرې شوې دي. آر پي جي-۴ په ۱۹۹۴ کې راووته، په دې ژبه کې د مثال په ډول د فرعي پروګرامونو د تعریفولو وړتیا هم شته . آی بي ایم شرکت هم ویژوال-ایج آر پي جي را وباسله چې د مایکروسافټ ویندوز په کاري چاپېریال کې د آر پي جي پروګرامونو د جوړولو او ځغلولو اسانتیا برابروي. د آر پي جي له پروګرامونو سره د اړیکو ټینګولو کې وسایل شتون لري چې ډول ډول ډیټابېس سیستمونه ترې یادولای شو؛ دغه راز کولای شو چې آر پي جي د انټرنېټي پروګرامونو (CGI) د لیکلو لپاره هم استفاده کړو.

## اضافي لوست:

1- Cozzi, Robert. *The Modern RPG IV Language.* 4th ed. Lewisville, Tex.: MC Press Online, 2006.
2- Martin, Jim. *Free-Format RPG IV: How to Bring Your RPG Programs into the 21st Century.* Lewisville, Tex.: MC Press Online, 2005.
3- Meyers, Bryan, and Jef Sutherland. *VisualAge for RPG by Example.* Loveland, Colo.: Duke Press, 1998.

***

# (۲۵)

## روبي (Ruby)

روبي يوه هر اړخيزه خو په ورته وخت کې ثابته پروګرامي ژبه ده چې په وروستيو کلونو کې يې ښه نوم پيدا کړی، په ځانګړي ډول د ويب جوړونې په برخه کې. نوموړې ژبه د يوکيرو ماځوموټو له خوا ډيزاين او په ۱۹۹۵ کې د استعمال لپاره راووته؛ روبي، غوټه نحوي ترکيب لري چې د پرل او نورو متنواله ژبو ځټه اخيستونکي ورسره بلدتيا لري، او د متحول د ډول بيانولو ته اړتيا نه پيدا کوي. روبي په ټوله کې يوه استازواله ژبه ده چې يو څه سمالټاک (Smalltalk) ته ورته ده. ماځوموټو پر دې ټينګار کړی چې د ژبې ډيزاين په داسې شکل وکړي چې د پروګرام ليکونکي لپاره طبيعي او خوندوره اوسي، نه دا چې د پروګرام ليکونکی د ماشين پر اړتياوو خپله انرژي راټوله کړي.

### جوړښت

په روبي کې هر ډېټا ډول يو استازی دی، تر دې چې هغه ډولونه هم چې په نورو ژبو کې د اوليه ډولونو په توګه تعريف شوي، لکه عددي يا بولين ډېټا ډولونه. که څه هم يو څوک کولای شي چې د متحولونو د ډيزاين کولو لپاره د دوديزو پروسوي ميتودونو څخه ګټه پورته کړي او بيا پر هغو خپل کارونه ترسره کړي،

خو بیا هم له دا ډول متحولونو سره مبهم چلند کېږي او د Self په نوم د بېخن استازي یوه برخه ګڼل کېږي.

هر فنکشن یو میتود دی چې په یو نه یو کلاس پورې اړه لري. په دې ډول - 5.abs پر 5- د مطلقې وېلیو میتود پلي کوي او بېرته 5 راګرځوي. په ورته ډول wireless wombat.length د همدې متنکرښې اوږدوالی راګرځوي چې ۱۵ کېږي. روبي له ډېټا جوړښتونو لکه کتارونه او هېشونو[1] سره د کار کولو لپاره ګڼ شمېر ځاني میتودونه هم لري، همدا رنګه ډېرې دودیزې لایبرېریانې او کاریالونه هم ورکې د لاسرسي وړ دي.

د ټولو لویو عامل سیسټمونو لپاره روبي ترجمانان لري. له یوه فایل څخه د پروګرام د اجرا کولو او یا لوستلو تر څنګ چې په ډېرو متنواله ژبو کې ترسره کېږي، روبي د ازمایښتي عبارتونو لپاره هم په تعاملي ډول کارول کولای شي:

```
% ruby eval.rb
ruby> puts “Hello, world.”
Hello, world.
nil
ruby> exit
```

په دې ځای کې د روبي ترجمان ته ویل کېږي چې eval.rb وځغلوي چې یو خاص پروګرام دی چې په تعاملي ډول عبارتونه او افادې ارزوي. د puts کمانډ

[1] Hashes

متنکربنه Hello world درج کوي. ورپسې ارزوونکی داسې راپور چمتو کوي چې: puts میتود تشه وېلیو (nil) راکړئولې ده.

که څه هم روبي په دودیز ډول یوه ترجمه کېدونکې ژبه ده، خو یوه نسخه یې چې د مجازي ماشین (جاوا ته ورته) لپاره به بایټ کوډ تولیدوي، د جوړېدو په حال کې ده، او د یوه لا مستقیم تالیفوونکي امکان هم لیدل کېږي.

### روبي پر اورګاډي (Ruby on Rails)

د روبي د پروګراملیکنې لپاره ډېر نومیالی کاري چاپېریال روپي پر اورګاډي دی، چې یو پرانیستي سرچینه کاریالي چارچوکات دی چې خاص په دې موخه جوړ شوی چې داسې پروګرامونه ورباندې ولیکل شي چې وېبسایټونه له ډېټابېسونو سره وصل کړي. نوموړی چارچوکات د ماډل-لید کنټرولوونکې[1] تګلارې باندې بنا دی چې ډېټا ته لاسرسی او منطق د کټه اخیستونکي له اړیکځای څخه بېلوي؛ همدا رنګه داسې یو چوکاټ (scaffolding) هم لري چې په ټیزې سره ډکېدای شي څو له اساسي فعالیت څخه برخمن داسې وېبسایټونه چمتو کړي چې د ډېټا په اساس حرکت وکړي. ډوبلپران کولای شي چې د روبي په مرسته نښتي کاریالونه[2] او شته پېکجونه هم پراخه کړي.

[1] Model-view controller
[2] plug-ins

## اضافي لوست:

1- Baird, Kevin. *Ruby by Example: Concepts and Code.* San Francisco: No Starch Press, 2007.

2- Burd, Barry. *Ruby on Rails for Dummies.* Hoboken, N.J.: Wiley, 2007.

3- Cooper, Peter. *Beginning Ruby: From Novice to Professional.* Berkeley, Calif.: Apress, 2007.

4- "Ruby: A Programmer's Best Friend." Available online. URL: http:// www.ruby-lang.org/en/. Accessed November 13, 2007.

5- Slagell, Mark. "Ruby User's Guide." Available online. URL: http:// www.mentalpointer.com/ruby/index.html. Accessed November 13, 2007.

6- Stewart, Bruce. "An Interview with the Creator of Ruby." O'Reilly Linux devcenter, November 29, 2001. Available online. URL: http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ ruby.html. Accessed November 13, 2007.

7- Thomas, Dave. *Programming Ruby: The Pragmatic Programmer's Guide.* 2nd ed. Raleigh, N.C.: Pragmatic Bookshelf, 2004.

***

## (۲۶)

## سیمولا (Simula)

د کمپیوټر له ډېرو په زړه پورې ګټو څخه یوه د هغو سیسټمونو تقلید دی چې په هغو کې ډېر شمېر بېلا بېل کارونه او پېښې په ورته وخت کې پېښېږي. د ۱۹۵۰ کلونو په اوږدو کې، نارویژي کمپیوټرپوه کریسټن نېګیارډ د تقلیدونو د ډیزاین او تشریح کولو لپاره یوه رسمي لاره رامنځته کړه. یو عادي تقلید یو شمېر استازي په ځان کې رانغاړي لکه په ترافیکي ګڼه ګوڼه کې موټرونه او یا بانک ته په لیکه کې ولاړ مشتریان. د بانک په تقلید کې، د مثال په ډول، استازي (مشتریان) له ځانګړو خدماتي استازو (د کړکۍ له صرافانو) څخه د خدمت تقاضا کوي. دوی په یوه لیکه کې حرکت کوي او د دوی حرکت د وخت په بېلا بېلو نقطو کې ثبتېږي.

نېګیارډ له خپلو نظریاتو څخه ګټه واخیسته تر څو داسې سمبولونه او جریان ښوودونکي ډیاګرامونه جوړ کړي چې په یوه تقلید کې روانې پېښې په ډاګه کړای شي. شته کمپیوټري ژبې لکه الګول-۶۰ داسې ډیزاین شوې وې چې په یوه ترتیب او په یوه وخت کې یوه پروسه سرته ورسوي، نه په هم مهاله توګه. او دې

شي دا کار سخت کړی وو چې داسې یو پروګرام ولیکل شي چې یو داسې حالت وکاږي په کوم کې چې ډېر مشتریان یا موټرې په یوه وخت کې د حرکت په حال کې وي.

د ۱۹۶۰ کلونو په لومړیو کې نېګیارډ له اولې-یوهان ډال سره یو ځای شو چې په سیستمي پروګراملیکنه او د کمپیوټري ژبو په ډیزاین کولو کې یې ډېره تجربه لرله. دوی دواړو په ګډه کار وکړ او یوه نوې ژبه یې جوړه کړه چې سیمولا نوم یې پرې کېښود، او پر تقلیدي پروګراملیکنه[1] د دوی د ټینګار لیدلوری یې را غبرګاوه (انعکاس کاوه). د سیمولا د ډیزاین پر مهال، دواړه لیکوالان د داسې ډېټا جوړښت[2] د جوړولو په هڅه کې وو چې د هم مهاله کارونو یا پېښو لپاره غوره اوسي. د مثال په ډول، د موټرونو د ګڼه ګوڼې په پېښه کې، هر موټر یو استازی دی چې خپله ډېټا لکه ځای او چټکتیا لري او دغه موټر ځینې کارونه او وړتیاوې لري لکه د چټکتیا او لوري بدلول. د هر استازي ډېټا باید په ټول ډول پاتې شي او په پرلپسې توګه باید اپډېټ شي.

الګول ۶۰ ترمخکې لا د کوډ بلاګونو د تعریفولو لپاره یوه لاره لرله چې خپله ځانې سیمه ییزه ډېټا[3] او کارونه (actions) یې په ځان کې ذخیره کولای شوی. پر دې سربېره، دا ډول بلاګونه په تکراري ډول بلل کېدای شي داسې چې په ورته

[1] simulation programming
[2] data structure
[3] local data

وخت کې يې ډېرې کاپيانې پرانيستلی شي. خو دې بلنو بيا هم ترتيبي بڼه لرله او هم مهاله نه وې. په نوې سيمولا-۱ ژبه کې (چې ۱۹۶۵ کې راغله)، ډال او نېکيارډ داسې لاره وپنځوله چې د هم مهاله پروسېس تقليد وکولای شي. که څه هم کمپيوټر ښايي يوازې يو پروسېس کوونکی ولري داسې چې په يوه وخت کې د کوډ د بلاک يوازې يوه کاپي اجرا کېدای شي، خو سيمولا ځانګړي متحولونه وضع کړل تر څو تقليدي مهال تعقيب کړي. کنټرول به د يوه بلاک له استازي څخه بل استازي ته «ټوپ» کړل، ګواکې ټول بلاکونه، د مثال په ډول، د 20:15 وخت لپاره خپل کارونه لري چې اجرا شي، او ورپسې به د ۲۰:۱۶ د وخت کارونه اجرا کېږي او همداسې به دوام کوي. په يوه ليست کې ټولې پروسې د خپل وخت په اساس خوندي وي. په دې ترتيب، سيمولا-۱ د الګول ټولې ځانګړنې په ځان کې لرلې، خو د هم مهاله پېښو د ماډل کولو لپاره يې ډېره ښه استفاده تري وکړه.

سيمولا-۱ خورا بريالی تقليدي ژبه وه، خو ليکوالانو يې ډېر ژر احساس کړه چې د يوې پروسې په بېل ډول د اجرا کېدو وړتيا، تر څو انفرادي استازي رامنځته کړي، د تقليد په پرتله په کاريالونو کې د ډېټا ښودنې لپاره لا ډېره عمومي ګټه لري. د سيمولا-۶۷ (د نوموړې ژبې هغه نسخه چې تر ننه پورې کارېږي) په جوړولو کې دوی د کلاس رسمي مفهوم د يوه داسې خصوصيت په توګه رامنځته کړ چې د هماغه ډول استازو د جوړولو لپاره کارېدای شي. همدا رنګه دوی د توارث اساسي نظريه پکې رامنځته کړه (چېرته چې يو کلاس له يوه مخکيني کلاس څخه

مشتق کېدای شي)، او دغه راز داسې یوه لاره یې هم وموندله چې یو مشتق شوی کلاس هغه پروسه بیاځلې تعریفولای وشي چې له اصلي (پلرني) کلاس یې څخه په ارث اخیستې وي.

که څه هم سیمولا-۶۷ به د یوې عام-هدفه پروګرامي ژبې پر ځای په اساسي ډول د تقلیدونو لپاره وکارېږي، خو د نوموړې ژبې استازواله نظریات ډېر زیات اغېزمن ثابت شوو. د سمالتاک او ابډا دیزاین کوونکو، سیمولا ته د هغې د جوړښتواله نظریاتو له مخې ستركې ور اړولې، په همدې وخت کې نومیالۍ سي پلس پلس ژبه د «سي له کلاسونو سره» د جوړولو په هڅه را پیل شوه چې په جوړښت کې یې د سیمولا کربنې هم ګرځېدې.

**اضافي لوست:**

1- Holmevik, Jan-Rune. “Compiling Simula: a Study in the History of Computing and the Construction of the SIMULA Programming Languages.” *STS Report* (Trondheim). Available online. URL: http://staff.um.edu.mt/jskl1/simula.html. Accessed August 21, 2007.

2- Nygaard, K., and O.-J. Dahl. “The Development of the Simula Languages” in *The History of Programming Languages*, R. L. Wexelblat, ed. New York: Academic Press, 1981.

3- Pooley, R. *Introduction to Programming with Simula.* Oxfordshire, U.K.: Alfred Waller, 1987.

4- Sebesta, Robert W. *Concepts of Programming Languages.* 7th ed. Boston: Addison-Wesley, 2006.

5- Sklenar, J. “Introduction to OOP in Simula.” Available online. URL: http://staff.um.edu.mt/jskl1/talk.html. Accessed August 21, 2007.

***

## (۲۷)

## سمالټاک (Smalltalk) یا وړې خبرې

په ۱۹۷۰ کلونو کې په زیراکس پالو آلټو څېړنیز لابراتوار (PARC) کې د کار کولو په جریان کې کمپیوټرپوه اېلن کای ګډ داسې نظریات او وسایل وپنځول چې د نن ورځ شخصي کمپیوټرونو ته یې لار کړې ده. د یوه وړاندیز شوي لېپټاپ د ډیزاین کولو پر مهال چې Dynabook نومېده، کای پرېکړه وکړه چې د نوموړي د عامل سیستم د جوړولو لپاره له نوې ټکلارې څخه ګټه پورته کړي.

د سمالټاک په جوړولو کې کای دوه مهمې مفکورې په ذهن کې نیولې وې: لومړی دا چې خلک به وکولای شي چې د کمپیوټر پر ځواک په ډېر اسانه ډول برلاسي شي، د دې پر ځای چې د دودیز تالیف د ستومانوونکې پروسې باندې ځان سرګردانه کړي چې پروګرامونه په تعاملي شکل جوړ، امتحان او تصحیح کوي. سېمور پېپرټ دمخه لا لوګو جوړه کړې وه چې یوه تعاملي او له ګرافیکي اړخه غني ژبه وه او ماشومانو ته د کمپیوټر د ډېرو پېچلو مفاهیمو د تدریس لپاره یوه غوره ژبه ثابته شوې وه. د سمالټاک نوم پخپله هم د دې خبرې انعکاس دی.

په څه ډول چې د دې ژبې لومړنۍ نسخه ډیزاین شوې وه، هغه پخپله هم ساده او ماشوم-دوسته ژبه وه.

بل اساسي فکر چې په سمالټاک کې وکارول شو، هغه استازواله پروګراملیکنه وه چې د لومړي ځل لپاره په سیمولا-۶۷ ژبه کې رامنځته شو. د ژبو شته ځانګړنو ته یوازې د ساده کلاسونو او استازو د ور زیاتولو پر ځای، کای سمالټاک داسې ډیزاین کړه چې له خپل بنیاد څخه استازواله ژبه وي. تر دې چې ډېټا ډولونه (لکه عددونه او توري) چې په دودیزو ژبو کې د متحولونو د بیانولو لپاره استعمالېږي، په سمالټاک کې استازي تړې جوړ شوو. ګټه اخیستونکو نوي کلاسونه هم جوړولای شوی چې د «ځاني» کلاسونو چلند به ورسره کېده. دې خبرې ته د اندېښنې کولو هېڅ اړتیا نه وه چې وړاندې له دې چې یو خاص ډول متحولونه وکارول شي، هغه لومړی باید بیان شي؛ په سمالټاک کې متحولونه په هر ډول استازي پورې تړل کېدای شي.

د دې لپاره چې یو پروګرام د یوه کار ترسره کولو ته وهڅول شي، یوه استازي ته پیغام لېږل کېږي چې هغه د هغه استازي تعریف شوې وړتیاوې (میتودونه) را وپاروي. د یوه ډېر ساده مثال په توګه، د ښسیک ژبې ټاکنیز[1] عبارت ته وګورئ:

```
Total = Total + 1
```

[1] Assignment

د بېسیک په څېر په یوه دودیزه ژبه کې له پورتني عبارت څخه داسې مفهوم اخیستل کېږي چې «۱ له هغه وېلیو سره جمع کړه چې د Total تر عنوان لاندې موقعیت کې ذخیره ده او حاصل یې بېرته په هماغه موقعیت کې ذخیره کړه.» خو په استازواله سمالټاک ژبه کې د همدې معادل عبارت په لاندې ډول دی:

```
Total <- Total + 1
```

دا په دې معنا ده چې « 1 + پیغام هغه استازي ته ولېږه چې د Total نومي متحول په مرسته مراجعه ورته شوې.». دا پیغام د + میتود ته مراجعه کوي، چې یو له هغو میتودونو څخه دی چې عددي استازي وربانديې پوهېږي. له همدې کبله، هغه استازی په خپله وېلیو کې ۱ جمع کوي او بیا هغه وېلیو د نوي استازي په توګه واپس کوي، چې اوس د Total متحول په وسیله ورته مراجعه کېږي.

په سمالټاک کې یو پروګرام په ساده ډول له وړتیاوو څخه برخمنه د استازو یوه ټولګه ده چې هره اړینه پروسه سرته رسوي. استازي او په هغوی پوري تړلي متحولونه، یوه «کاري ساحه» جوړوي، چې د ډیسک پر مخ په دوراني شکل ذخیره کېږي.

د سمالټاک د پروګرام لیکونکي لپاره د سمالټاک د عامل سیستم او د کوربه د کمپیوټر د عامل سیستم ترمنځ هېڅ توپیر نشته. د عامل سیستم وړتیاوې (لکه د فایلونو سمبالښت) له مخکې څخه د تعریف شوو استازو په توګه د سمالټاک سیستم دننه پراته وي. کای، سمالټاک په خپل ذهن کې د یوه داسې بشپړ کاري

چاپېریال په توګه مجسم کړي وه چې د هغو پروګرام لیکونکو له خوا هم پراختیا وموندلی شي چې د پروګراملیکنې تجربه نه لري، او نوموړي خپل مخکښ ګرافیکي اړیکځای د داسې روش په توګه ډیزاین کړ چې د ګټې اخیستونکو لپاره دا کار اسانه کړي څو له سیستم سره نېغ په نېغه کار وکولای شي.

سمالتاک یو «مجازي ماشین» لري چې هدایات یې د هر لوی کمپیوټري سیستم د ډول لپاره په ځانګړي کوډ کې تطبیق کېدای شي. د سمالتاک د ثابت جوړښت او له استازو څخه د هر شي د جوړولو د وړتیا له امله، د سمالتاک شاوخوا ټول سیستم پخپله سمالتاک کې لیکل کېږي، چې دا کار اسانه کوي چې کله یو ځل ماشین-اړونده تفصیلات چمتو شي، هغه بل کمپیوټر ته (له تغیر پرته) ولېږدول شي.

د نوموړې ژبې د سبک، ثبات او په شخصي کمپیوټرونو کې د شتون له امله، سمالتاک له ۱۹۸۰ کلونو څخه وړاندې د پام وړ توجه را جلب کړه. یاده ژبه د اصلي تګلوري په کاریالانو کې په پراخه توګه نه ده کارول شوې، په دې چې د کلاسونو او توارث د میتودونو د تعقیبولو لپاره د اړتیا وړ میکانیزمونه په هغومره اغېزمن ډول پکې پلي کول ستونزمن وو، څومره ساده میکانیزمونه چې دودیزو ژبو کارول. د اغېزمنتیا او د زده کړې په برخه کې د اسانتیا له کبله په شته ژبه باندې د استازواله ځانګړنو د ور زیاتولو تګلارې (څنګه چې سي پلس پلس له سي څخه جوړه شوه) د ډېرو کسانو زړونه خوارې کړل.

د سمالټاک معنوي ځواک دا ژبه، د مصنوعي ځیرکتیا او پیچلي تقلید په ډګر کې د ځانګړو پروژو لپاره یوه زړه راښکونکې ژبه ګرځولې، او دا د هغو کسانو لپاره یو لوی انعام دی چې پر استازواله روش بنا یوه کره ژبه غواړي چېرته یو کاریال په واضح ډول د ربنتینې نړۍ حالت انځورولای شي. سمالټاک ماشومانو (او نورو) ته د پروګراملیکنې د تدریس لپاره لا هم یو غوره انتخاب دی. د سمالټاک یوه نسخه چې squeak بلل کېږي، د ګرافیک او نورو فنکشنونو لپاره یو غني کاري چاپېریال برابروي. Squeak او د سمالټاک یو شمېر نور تطبیقات د مختلفو کمپیوټري سیسټمونو لپاره په وړیا ډول د ډاونلوډ کېدو وړ دي.

## اضافي لوست:

1- Ducasse, Stephane. *Squeak: Learn Programming with Robots.* Berkeley, Calif.: Apress, 2005.

2- Klimas, Edward J., Suzanne Skublics, and David A. Thomas. *Smalltalk with Style.* Englewood Cliffs, N.J.: Prentice Hall, 1996.

3- Lewis, Simon. *The Art and Science of Smalltalk.* Englewood Cliffs, N.J.: Prentice Hall, 1995.

4- Smalltalk. Available online. URL: http://smalltalk.org. Accessed August 21, 2007.

5- Squeak. Available online. URL: http://www.squeak.org/. Accessed August 21, 2007.

***

# (۲۸)

## ټي سي اېل 1(TCL)

په ۱۹۸۸ کې د جان آؤسټرهاؤټ له خوا جوړه شوې، ټي سي اېل ژبه د متنوالی، پروټوټایپ، اړیکځایونو د کتنې او مختلطو کاریالانو د جوړولو لپاره استفاده کېږي.

ټي سي اېل یو ډېر غیرمعمولي ساده او ثابت نحوي ترکیب لري. د ژبې متن د کمانډونو او د هغو د آرګومنټونو (پارامترونو) یوه ساده لړۍ وي. (چې یا ځانې وي او یا د کټې اخیستونکي له خوا تعریف شوي وي). پخپله یو کمانډ کېدای شي چې یو آرګومنټ وي او یا یو بل کمانډ، چې په نورو ژبو کې د فنکشني بلنې معادل وي.

د مثال په ډول، د یوه متحول وېلیو ټاکل د set کمانډ په مرسته ترسره کېږي:

```
set total 0
```

د متحول وېلیو اوس له $total څخه اخیستل کېدای شي.

کنټرول جوړښتونه هغه کمانډونه دي چې نور کمانډونه ځغلوي. While څرخ، د مثال په ډول، هغه کمانډ یا افاده لري چې یوه پرتلنه سرته رسوي، او کوم کمانډونه یې ترڅنګ راغلي وي، چې هغه هر ځل «صحیح» شي، نو اجرا شي.

```
while { MoreInFile } {
```

1 Tool command language

```
GetData
DisplayData
}
```

په عمل کې، ډېرى کمانډونه د عامل سیستم چې معمولا یونیکس یا لینکس وي، مرستندویه پوستغالي دي چې استعمالېږي. تي سي اېل یو شمېر ګټور ډېټا جوړښتونه هم لري لکه اشتراکي کتارونه[1] چې د ډېټا توکونو جوړې پکې راځي، لکه:

```
set abbr (California)
```

## زیاتونې اوګټې

تي سي اېل یو شمېر زیاتونې هم لري چې د مثال په ډول، د MySQL په څېر د ډېټابېس مشهورو فارمټونو ته لاسرسی چمتو کوي او له نورو پروګرامي ژبو لکه سي پلس پلس او جاوا سره ارتباط ټینګولای شي. تر ټولو زیاته کارېدونکې زیاتونه یې Tk ده، چې د بېلا بېلو عامل سیستمونو او ژبو لکه پرل، پایتان او روبي لپاره د ګټه اخیستونکي د اړیکځایونو جوړولو لپاره یوه لایبرېري وړاندې کوي.

تي سي اېل د شته کاریالونو د نښلوونکو لپاره د «سریښ» په توګه پېژندل کېږي. نوموړې ژبه نسبتاً اسانه او یو متن په تعاملي شکل ګوري (چې ډېر پر کمانډ کرښه ترسره کېږي) او بیا دا متن د یوه کاریال کوډ ته ور درج کوي. کله چې

[1] associative array

کاریال ځغلي، د تي سي اېل ترجمان هغه متن ځغلوي، چې وروسته یې راکړیز د اصلي کاریال له خوا استفاده کېدای شي.

## اضافي لوست:

1- Foster-Johnson, Eric. *Graphical Applications with Tcl & Tk.* 2nd ed. New York: Hungry Minds, 1997.

2- Wall, Kurt. *Tcl and Tk Programming for the Absolute Beginner.* Boston:Course Technology, 2007.

3- Welch, Brent B., and Ken Jones. *Practical Programming in Tcl and Tk.* 4th ed. Upper Saddle River, N.J.: Prentice Hall, 2003.

***

# (۲۹)

## وي بي سکرپټ (VBScript)

وي بي سکرپټ، هغه ژبه چې د زوکړې مهال يې د ۱۹۹۰ کلونو منځينو وختونو ته رسېږي، يوه متنواله ژبه ده چې مايکروسافټ کمپنۍ جوړه کړې او بنسټ يې په خپل مشهور ويژوال بېسيک پروګرامي ژبې اينبودل شوی. نوموړې ژبه د هغه تحول يوه برخه ده چې مايکروسافټ «فعاله متنواله» باله، او په هغو اجزاوو ولاړه وه چې د پوستغالي وړتياوو ته له بهر څخه د لاسرسي اجازه ورکوي. هغه کوربه چاپېريال چې متنونه پکې ځغلي د وينډوز له خوا چمتو کېږي (لکه د وينډوز متني کوربه[1] او يا د مايکروسافټ انټرنټي اېکسپلورر دننه اجرا کېږي.)

د کلاينټ-لوري پروسېس لپاره، وي بي سکرپټ د هغو متنونو د ليکلو لپاره استعمالېدای شي چې په html پاڼو کې مختلط وي، چې له معياري ډاکومنټ آبجکټ ماډل سره د نورو وېبي متنواله ژبو په څېر اړيکه نيسي (په ځانګړي ډول، وګورئ: جاواسکرپټ). که څه هم، د مايکروسافټ له پرانيستونکو پرته په نورو پرانيستونکو لکه فايرفاکس يا اوپېرا کې د وي بي سکرپټ ملاتړ نه کېږي، نو د دې پر ځای ډوېلپران بايد له انطباق پذيره جاواسکرپټ څخه ګټه پورته کړي. وي

[1] Windows Script Host

بي سکرپټ په ویب سرور باندې د پروسېس کولو لپاره هم کارول کېدای شي، په ځانګړي ډول د مایکروسافټ له ویب سرورونو سره د ارتباط لپاره.

د ویندوز د نسخو له کبله چې په ویندوز-۹۸ را پیل شوې، ټولې نسخې یې د ویندوز متني کوربه لري، خو وي بي سکرپټونه په مستقیم ډول تر ویندوز لاندې هم لیکل کېدای شي. د وي بي سکرپټ یوه خواشینوونکې پایله د بې پامه ګټه اخیستونکو لپاره هغه متنونه دي چې چینجي[1] به یې لرل (لکه د «زه درسره مینه لرم»[2] چینجی) او یا نور په ویروس ککړ پوستغالي (malware) یا په برېښنالیک کې راغلي ویرسونه.

## مثالونه

د وي بي سکرپټ له کوډ سره به د ویژوال بېسیک ګټه اخیستونکي ښه بلد وي او عموماً استازواله ژبو ته ورته نحوي جوړښت پالي. د «Hello World» عادي پروګرام په لاندې ډول لیکل کېږي:

```
WScript.Echo "Hello World!"
```

په دې ځای کې WScript هغه استازی دی چې د کوربه متن استازیتوب کوي.

[1] worms
[2] I love you

په متنچوکاټ کې له کټه اخیستونکي څخه د ورکړیز اخیستو لپاره، پروګرام لیکونکی په لاندې شکل کوډ لیکلای شي:

```
option explicit
dim userInput
userInput = InputBox("What is your name?:",
"Greetings")
if userInput = "" then
Msgbox "You did not write anything or you
pressed cancel!"
else
MsgBox "Hello, " & userInput & ".",
vbInformation
end if
```

له شک پرته وي بي سکرپټ داسې لایبرېرۍ او اړیکټایونه لري چې د دې جوړه یې کړئوي چې لا ډېرې پېچلې کړنې سرته ورسوي لکه د ډېټابېسونو برابرول او د وینډوز د مدیریتي آلاتو (WMI)[1] او فعاله خدماتي اړیکټای (ADSI)[2] په مرسته د وینډوز سیسټمونو د نورو برخو تنظیمات ترسره کول.

که څه هم نوموړې ژبه (او کوډ به یې) تر راتلونکو څو کلونو پورې تر استعمال لاندې وي، خو مایکروسافټ په اوس وخت کې وي بي سکرپټ ته نور پرمختګ نه ورکوي، ځکه دا مهال یې د یوه نوي پروګرامي چارچوکاټ په لور ګام پورته

1 Windows Management Instrumentaion
2 Active Directory Services Interface

کړی او ټول پام یې د ویژوال بېسیک ډاټ نېټ (visual basic.net) په څېر ژبو ته دی.

## اضافي لوست:

1- Jones, Don. *VBScript, WMI, and ADSI Unleashed.* Indianapolis: Sams, 2007.

2- VBScript Sample Scripts. Available online. URL: http://cwashington. netreach.net/depo/default.asp?topic=repository&scriptty pe =vbscript. Accessed December 2, 2007.

3- VBScript Tutorial. W3Schools. Available online. URL: http://www. w3schools.com/vbscript/default.asp. Accessed December 2, 2007.

4- VBScript User's Guide. Microsoft Developer Network. Available online. URL: http://msdn2.microsoft.com/en-us/library/ sx7b3k7y.aspx. Accessed December 2, 2007.

5- Wilson, Ed. *Microsoft VBScript Step by Step.* Redmond, Wash.: Microsoft Press, 2007.

***

## پای

# ماخذونه:

1- Hend-erson, Harry, (1951–), Encyclopedia of Computer Science and Technology – Rev. ed. Facts on File – 132 West 31st StreetNew York NY 10001, 2009.

2- Sinha, P.K., Computer Fundamentals, PBP Publications, B- 14, Connaught Place, New Delhi-110001, 2003.