

کمپیوٹری ژپی

Ketabton.com

کمپیوٹرستان (۱)

کمپیوٲری ژبې

~~~~~\*\*\*~~~~~

کمپیوٲرستان (۱)

ژباړه: رحمت شاه فراز

## د کتاب پیژندنه

# کامپیوٲری ژبې

## لیکوال:

هېري هېنډرسن

## ژباړه:

رحمت شاه فراز

د کتاب ټول حقونه له ژباړن سره خوندي دي.

اړیکه:

078 063 48 38

برېښنالیک:

rsfaraz4@gmail.com

## لیکلر

|         |                 |
|---------|-----------------|
| 7.....  | سریزه           |
| 12..... | کمپیوٹری ژبې    |
| 15..... | ماشینواله ژبه   |
| 22..... | منحنی ژبه       |
| 43..... | عالی ژبې        |
| 46..... | تالیفونکی       |
| 51..... | ترجمان          |
| 60..... | اډا (Ada)       |
| 67..... | الګول (Algol)   |
| 72..... | اې پی اېل (APL) |
| 75..... | آک (Awk)        |
| 78..... | بېسیک (BASIC)   |
| 84..... | سی (C)          |

---

|          |                          |
|----------|--------------------------|
| 94.....  | سي شارپ (C#)             |
| 98.....  | سي پلس پلس (C++)         |
| 107..... | ڪوبال (COBOL)            |
| 114..... | آيفل (Eiffel)            |
| 119..... | فورت (Forth)             |
| 123..... | فورٽران (Fortran)        |
| 128..... | جاوا (Java)              |
| 137..... | جاوا اسڪرپٽ (JavaScript) |
| 143..... | ليسيپ (LISP)             |
| 150..... | لوگو (Logo)              |
| 156..... | لووا (Lua)               |
| 159..... | پاسڪال (Pascal)          |
| 166..... | پرل (Perl)               |
| 171..... | پي اڇ پي (PHP)           |
| 175..... | پي ايل-۱ (PL/1)          |

---

|          |                                 |
|----------|---------------------------------|
| 181..... | پرولوڪ (Prolog)                 |
| 185..... | پايتان (Python)                 |
| 188..... | آرپي جي (RPG)                   |
| 190..... | روبي (Ruby)                     |
| 194..... | سيمولنا ( Simula)               |
| 198..... | سمالتاڪ (Smalltalk) يا وري خبري |
| 204..... | تي سي ايل (TCL)                 |
| 207..... | وي بي سڪرپٽ (VBScript)          |

## سريزه

کمپیوټر پوهنه 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 پښخول او د سختوالي په برخه کې نوښت او ابتکار کول خو په ډېر لرې راتلونکي کې هم نه تر سترگو 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 کمپیوټر پوهنې ته تر ډېره پاملرنه کړې؛ او همدا رنگه دا هم له پامه نه شو غورځولای چې د کمپیوټر پوهنې په ځینو برخو لکه د ډېسکټاپ او موبایل کار یالونو او وېبسایټ جوړونې په ډګر کې ډېری شخصي کمپنۍ او سافټوېر 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 ښکمرغۍ او خوشالی ځای دی چې یادې څانګې ته زموږ 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 ترسترگو

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

همدې اړتیا او تشې ته په کتو پر دې کتاب لاس پورې شوی. د کمپیوټري ژبو چې پروګرامي<sup>1</sup> ژبې هم ورته وایي، په برخه کې بنایي 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 کمپیوټر پوهنې او ټکنالوژۍ پوهنغونډ» چې د کمپیوټر پوهنې د نړۍ تکره او د ډېرو اثارو څښتن لیکوال «هېډري هېنډرسن» لیکلی او که څوک غواړي چې د کمپیوټر پوهنې له هر کونج څخه ځان خبر کړي او په دې ناپایه کاینات کې قدمونه ووهي، نو زما تر سترگو له دې نه زیات علمي، مستدل او څېړنیز کتاب نه دی تېر شوی. دوهم کتاب د «پي. کې. سینا»، «د کمپیوټر اساسات» دی چې د کمپیوټر په بنسټي مفاهیمو او مسایلو د پوهېدو لپاره یو لنډ او غونډ اثر دی چې په خپل ځای یې یو ارزښتناک او کتور 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 «ابدا» په عنوان پیل او د کتاب تر پایه غځېږي.

---

<sup>1</sup> Programming Language



د کتاب د ژباړې په هکله به هم په دې ځای کې ځینې ټکو ته اشاره بڼه وي. دا چې په دې برخه کې تر دې دمخه هېڅ کتاب نه وو ژباړل شوی (یا زما ترسترگو 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 اوردو کې تل دوه وېرې راسره وې: یوه وېره دا وه چې که چرته د کمپیوټرپوهنې اصطلاحات را ونه ژباړم، نو عادي لوستونکي به خندی کوي چې دا نو څنگه 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 هېڅ نه وی ژباړل شوی، بڼه به وه، ځکه ژباړه یې تر انګلیسي هم سخته ده او یا د ارواښاد شپون په خبره چې عبدالحی حبیبی صېب به مجاور احمد زیار ته ویل چې ما ډېر نوي او زاړه متنونه لوستي دي او ورباندې 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 رېلو کې مې زړه پر دې پرېکړې اوبه وڅښلې چې هغه اصطلاحات چې په اسانه د ژباړلو وړ وي او له ژباړې وروسته به د کمپیوټرپوهنې لوستونکي هم په اسانه ورباندې پوهېدای شي او ورسره د ژباړې

اصول او معيارونه 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 کړيالانو، ښوونکو او ماهرينو نه هيله وکړم چې وړ او معياري بدلونه ورته ومومي او د یاد علم په بډاینه کې چې د يوه تن د وس کار نه دی، ونډه واخلي. د کتاب په دريم مخ کې مې خپله اړیکه او برېښنالیک همدې ته په کتو زیات کړي.

خبرې اوږدې هم شوې او پاتې هم دي؛ ښايي لامل يې دا وي چې په دې برخه کې به خبرې زياتې وي. خو نورې خبرې به بيا وشي، ځکه دا د «کمپیوټرستان» د زنجير لومړۍ کړۍ ده، نو په دې خاطر خپلې خبرې به بيا په نورو کړيو کې کړۍ کړۍ کړم.

ستاسو

رحمت شاه فراز

# د «ڪمپيوٽر اساسات»

له ڪتاب ٽيڪهه

ليڪوال: پي. ڪي. سينا

ٽيڪهه: رحمت شاه فراز

## کمپیوټري ژبې

### له طبیعي ژبوسره 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<sup>1</sup> 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 ګټه اخیستونکي پوهېږي. مور په عامه توګه د هغې ځانګړې ژبې د سمبولونو ټولګې ته وپېانګه وایو. د مثال په ډول، هغه کلمات چې مور یې په انګلیسي کې کاروو، د انګلیسي ژبې سمبولونه دي چې د دې ژبې وپېانګه جوړوي. هره کلمه ځانګړې معنا لري چې په سیند (قاموس) کې موندل کېدای شي. په ورته ډول، د کمپیوټر ټولې ژبې خپله ځانګړې وپېانګه لري. د دې وپېانګې 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

طبیعي ژبې د کلماتو پراخه زېرمه لري، خو ډېری کمپیوټري ژبې یوازې له یوې محدودې او ټاکلې ویبپانګې څخه ګټه پورته کوي. د دې اصلي لامل دا دی چې پروګرامي ژبه <sup>1</sup> 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<sup>2</sup> بنا دي: ورکړیز او راګړیز عملیات<sup>3</sup>، ریاضیکي عملیات، د مرکزي پروسېس کونکي ترمنځ د معلوماتو د حرکت عملیات، او منطقي یا مقایسوي عملیات<sup>4</sup>.

هره طبیعي ژبه د خپلو سمبولونو د کارونې لپاره یو سیستاتیک میتود لري. په انګلیسي کې، دغه میتود د ګرامري اصولو په بڼه وړاندې کېږي. په اصل کې همدا اصول مور ته وايي 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<sup>5</sup> 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 کره اصولو ته ژمن پاتې شو. که نه نو، هېڅ کمپیوټر د دې وړتیا نه لري چې ناسمې لارښوونې<sup>1</sup> 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<sup>2</sup> چې بايزري متنګرښې<sup>3</sup> (صفر او یو) کاروي، تر ستونزواله ژبو<sup>4</sup> 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 طبيعت او تحول وڅېړو.

#### ماشينواله ژبه

که څه هم کمپیوټرونه په بېلا بېلو کمپیوټري ژبو باندې د پوهېدو لپاره پروگرام کېدای شي، خو یوازې یوه داسې ژبه شتون لري چې یو کمپیوټر پرې له کوم ژباړونکي پرته پوهېږي. دا ژبه، ماشينواله ژبه یا د کمپیوټر ماشينواله کوډ بلل کېږي. ماشينواله کوډ د یوه کمپیوټر بنيادي ژبه ده او عموماً د بايزي متنکړينو (صفر او يو) په شکل لیکل کېږي. د کمپیوټر سرکټ په داسې ډول او بدل شوی چې سمډلاسه ماشينواله ژبه پېژني او هغو برېښنايي زیګنالونو ته یې اړوي چې د کمپیوټر د ځغلولو<sup>1</sup> لپاره ورته اړتیا ده.

هر لارښود چې په ماشينواله ژبه کې لیکل شوی وي، دوه برخيز فارمت<sup>2</sup> 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)<sup>3</sup> لري. د لارښود دوهمه برخه معمول<sup>4</sup> بلل کېږي، او دا

---

1 Running

2 Two-part format

3 Operation Code

4 Operand

برخه کمپیوټر ته بنیې چې ډېټا او یا نورې لارښوونې چې باید لاسوهنه<sup>1</sup> ورکې وشي، په کوم ځای کې ولټوي او یا یې ذخیره کړي. په دې اساس، هره لارښوونه د مرکزي پروسېس کونکي د کنټرول واحد ته لارښوونه کوي چې څه وکړي او د هغو ډېټاساحو<sup>2</sup> اوږدوالی او ځای ور په گوته کوي، چې په عملیه کې گډون لري. په عادي عملیو کې لوستل<sup>3</sup>، جمع کول، تفریق کول، لیکل<sup>4</sup> او داسې نور راځي.

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

مور پوهېږو چې ټول کمپیوټرونه د داخلي عملیو د ترسره کولو لپاره له بايزي ارقامو (صفرنو او یوونو) څخه گټه پورته کوي. په دې ترتیب، د ډېری کمپیوټرونو ماشینواله ژبې د بايزي شمېرو متنکړنې لري او همدا هغه یوازینی ژبه ده چې مرکزي پروسېس کونکي په مستقیم ډول ورباندې پوهېږي. کله چې د کمپیوټر دننه زېرمه شي، نو هغه سمبولونه چې د ماشینواله ژبې پروگرام جوړوي، له صفرونو او یوونو څخه جوړېږي. د مثال په ډول، بنایي د هغه عادي پروگرام لارښوونه چې یو عدد په پرنټر چاپ کړي، په لاندې ډول وي:

---

1 Manipulation

2 Data Fields

3 read

4 Write



10110011111101001101100

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

0100000000000110011101

0110001100000010000000

0110000000011100101110

1010001111101110010111

0000000000000000000000

بنکاره 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 سیستم کې لیکل شوي چې مور ورسره بلدتیا نه لرو. خو دا به د حیرانتیا  
ځای وي چې ځینې لومړني پروگرام لیکونکي چې پر ځینو لومړنیو کمپیوټرونو یې  
کار کاوه، په اصل کې خپل پروگرامونه په بايزي حالت کې لیکل لکه پورته چې  
تېر شوو.

دا چې انساني پروگرام لیکونکي د لسو له عددي سیستم سره ډېر بلد دي، نو له  
هغو څخه زیاترو دا غوره وگڼله چې د کمپیوټر هدايات<sup>1</sup> د لسو په سیستم کې

ولیکي. بیا به یې دغه هدايات ورکړيز واحد<sup>1</sup> ته وسپارل چې بايزي حالت ته یې واړوي. په حقیقت کې، له ډېرو هڅو پرته، یو کمپیوټر داسې اوډل کېدای شي چې د صفرونو او یوونو د اوږدو متنکړینو پرځای، له هغه اعشاري سیستم څخه پکې کار واخیستل شي چې زموږ بلدتیا ورسره ډېره ده. په دې بدلون سره، پورتنی پروگرام لاندینی بڼه خپلوي:

10001471

14002041

30003456

50773456

00000000

د لارښود کوډونو<sup>2</sup> دا ټولګه، که په بايزي 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 پروگرام ضرور نه ده چې د بايزي 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 کټې او محدودیتونه

هغه پروګرامونه چې په ماشینواله ژبه لیکل شوي وي، د کمپیوټر له لوري خورا چټک اجرا<sup>1</sup> کېدای شي. د دې اصلي لامل دا دی چې مرکزي پروسېس کوونکی د ماشین په لارښوونو 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-اړونده ژبه:** دا چې د هر کمپیوټر داخلي ډیزاین له بل هر ډول کمپیوټر سره توپیر لري او د فعالیت لپاره مختلفو برېښنايي 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 اوردوالي<sup>3</sup> 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. پروگرام کولو پېچلتيا:** که څه هم يو ماشين په اسانه توگه استفاده ترې کولای شي، خو په ماشينواله ژبه پروگرام ليکل يو پېچلی او دروند کار دی. پروگرام ليکونکی اړ دی چې د ماشين د لارښوونو په ډله کې د اوامرو<sup>1</sup> لپاره لسه او کوډونه په يادو زده کړي او يا پرلپسې له مراجعه کارت<sup>2</sup> څخه گټه واخلي. دغه راز يو پروگرام جوړوونکی مجبور دی چې په زېرمه کې د ډېټا او لارښوونو موقعيت هم په ذهن کې وساتي. د دې ترڅنګ، د ماشينواله ژبې پروگرام ليکونکی بايد هغه ماهر شخص وي چې د کمپيوټر د سختوالي جوړښت په اړه هم بشپړ پوهاوی ولري.

**۳. د تېروتنو زياتوالی:** په ماشينواله ژبه کې د پروگرامونو د ليکلو لپاره، دا چې يو پروگرام ليکونکی بايد عملياتي کوډونه په ياد ولري او ورسره بايد په زېرمه کې د ډېټا او لارښوونو موقعيتونه هم په ذهن کې وساتي، نو دا شی نوموړي ته دا ستونزمنوي چې خپل ټول تمرکز د ستونزې پر منطق<sup>3</sup> را ټول کړي. چې په حتمي توگه په پروگرام کې د تېروتنو په رامنځته کېدو پای ته رسېږي. له دې کبله، د ماشينواله کوډ د کارونې پرمهال ډېر په اسانه تېروتنې رامنځته کېږي.

---

1 Commands

2 Reference Card

3 Logic

۴. **د بدلون راوستلو پېچلتیا:** د ماشینواله ژبې پروگرامونو اصلاح<sup>1</sup> او یا تغیر پکې راوستل سخت وي. د تېروتنو د موندلو په موخه د ماشینواله لارښوونو کتنه هماغومره ستونزمنه وي څومره چې یې په پیل کې لیکل سخت وي. په همدې ډول، یو څه وخت وروسته د ماشینواله ژبې په پروگرام کې بدلون راوستل هم ستونزمن وي چې په دې ځای کې به ډېری پروگرام لیکونکي دا ښه وپولي چې په زاړه پروگرام 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 پروگرام لیکل دومره ستونزمن او وخت غواړي چې نن ورځ ترې استفاده له نشت سره برابره ده.

## منځنۍ ژبه

د پروگرام تیارونې د پروسې په وده کې له لومړنیو پړاوونو څخه یو دا وو چې د ماشینواله ژبې د عددي عملیاتي کوډونو<sup>1</sup> پر ځای حروفی 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 گټور دي. د مثال په ډول، یو کمپیوټر کېدای شي د دې لپاره ډیزاین شوی وي خو ۱۱۱۱ (بايزي) او یا ۱۵ (اعشاري) ماشینواله کوډ، د «تفریق» د عملیې په توګه وژباړي<sup>2</sup>، 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 ویني، نو د هغه پر ځای ۱۴ راوړي، هر ځل چې SUB وي، نو ځای یې ۱۵ ته ورکړي، او همداسې ادامه وکړي. په دې ډول، کمپیوټر

1 Numeric Operation code/opcode

2 Translate

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

CLA A

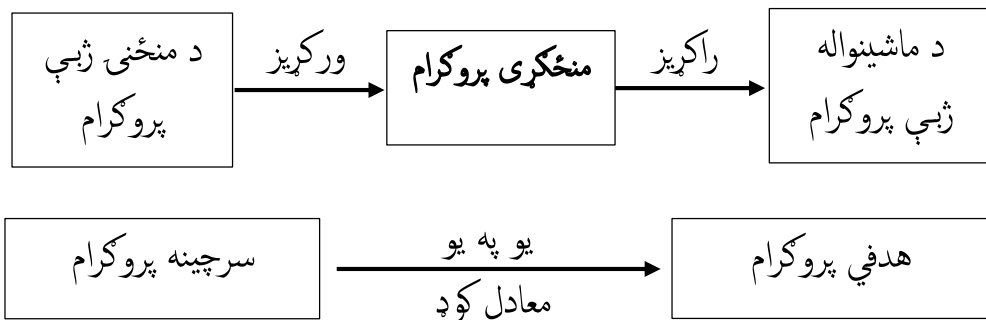
ADD B

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 پروگرام د تړلو لپاره، ژباړن) ولیکه او پروگرام به وتړل شي. کمپیوټر د یوه ژباړونکي پروگرام په مرسته، د دې پروگرام هره کرښه د ماشینواله ژبې د پروگرام معادل کوډ ته ژباړي.



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

---

1 Assembly

2 Assembler



پروگرام لیکونکي له لوري په منځنۍ ژبه کې لیکل شوی وي، سرچینه پروگرام<sup>1</sup> بلل کېږي. وروسته له دې چې پروگرام د منځګړي په مرسته ماشین کوډ ته واوړي، نو بیا د هدفي پروگرام<sup>2</sup> په نوم یادېږي. څنګه چې په پورتنی شکل کې لیدل کېږي، د منځګړي ورکړیز یو سرچینه پروگرام دی چې په منځنۍ ژبه کې لیکل کېږي. دا چې منځګړی پروگرام د منځنۍ ژبې هره لارښوونه خپل معادل د ماشینواله ژبې یوې لارښوونې ته ژباړي، نو له همدې امله د سرچینه پروگرام د منځنیو لارښوونو<sup>3</sup> او د هدفي پروگرام 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 پروگرام لیکو، نو لومړی منځګړی (پروگرام) ځغلوو څو سمبولیک پروگرام ماشینواله ژبې ته واوړي، او ورپسې د ځواب د ترلاسه کولو لپاره د ماشینواله ژبې پروگرام ځغلوو. ستاسو به پام وي چې د دې معنا دا ده چې تالیفونکی<sup>4</sup> ډېر زیات وخت مصرفوي. نوموړی د ځواب د ترلاسه کولو لپاره نه یوازې دا چې اصلي پروگرام وځغلوي، بلکې لومړی باید اصلي سمبولیک پروگرام ماشینواله ژبې ته وژباړي. خو دا چې سمبولیکه پروگراملیکنه د پروگرام لیکونکي

---

1 Source Program

2 Object Program

3 Assembly instructions

4 Compiler

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

که غواړئ وگورئ چې سمبولیکه پروگراملیکنه څه ډول کار کوي، راځئ لومړی د ماشینواله ژبې یو کوچنی پروگرام ولیکو او بیا وگورو چې همدا پروگرام 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          | هالت د پروگرام د تړلو لپاره په پای کې راځي. |
| CLA     | 10          | د رجسټر پاکول یا ور اضافه کول.              |
| ADD     | 14          | د رجسټر 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 مور یوازې پنځه عملیاتي کوډونه په پام کې نیولي چې زموږ د پروگرام په لیکلو کې به کارېږي. همدې ته ورته، د یوه ځانګړي پروگرام لپاره په سلهاوو نور عملیاتي کوډونه هم کېدای شي.

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

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

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

په دې ځای کې دا فرض شوي چې کمپیوټر د باینري عددونو پر ځای د اعشاري اعدادو د سمبالنټ 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 لومړنی لارښوونه د A رجسټر پاکوي (صفر کوي يې) او د 1000 د موقعیت محتوا (لومړنی عدد) پکې ځای په ځای کوي. دوهمه ماشينواله لارښوونه په 1000 موقعیت کې، د 1001 موقعیت محتوا (دوهم عدد) د A رجسټر له محتوا (لومړنی عدد) سره جمع کوي او حاصل يې 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 يې ذخیره کوي. په پای کې، څلورمه لارښوونه چې 1003 موقعیت کې راغلي، د پروگرام اجرا ته د پای ټکی ږدي.

اوس به وگورو چې همدا پروگرام 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 پروگرام په لاندې ډول ولیکو؛ او نور کار کمپیوټر ته وسپارو چې سمبال يې کړي. منځګړی پروگرام به CLA په 10، ADD په

14، STA په 30 او HLT په 00 ترجمه<sup>1</sup> کړي، او په دې ډول به د ماشینواله ژبې پروگرام تولید شي.

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

### 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.

<sup>1</sup> Interpret

د ماشينواله ژبې په پروگرامليکنه کې له لويو ستونزو څخه يوه هم د ادرسونو په ذهن کې ساتل دي. هر ځل چې مور يو لوی پروگرام ليکو، نو اړتيا ده چې په خپل څنگ کې يوه پاڼه راسره ولرو چې په هغه کې يو ليست وليکو چې کوم عددونه بايد چېرته ذخيره شي. هر ځل چې مور يو عدد غواړو، نو بايد په هغه ليست کې يې ادرس وگورو. دا کار وخت غواړي او که چېرته احتياط ونه شي، تېروتنه ترې رامنځته کېږي. د دې پر ځای ولې کمپيوټر ته دا واک نه ورکوو چې زموږ پر ځای هم کار وکړي؟ مور د منځنۍ ژبې د پروگرام د يوې برخې په توگه، په پروگرام کې يوه داسې برخه زياتولای شو چې بل هېڅ کار هم ونه کړي، پرته له دې چې د عددونو لپاره د ادرسونو يو ليست له ځان سره وساتي.

په پورتنی پروگرام کې د دوو عددونو د جمع کولو لپاره، بنايي د خپل سمبولیک پروگرام په مرسته کمپيوټر ته داسې امر وکړو:

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

د دې په غبرگون کې منځگړی پروگرام د کمپيوټر د زېرمې په يوه برخه کې يو جدول جوړوي چې بنايي لاندې حالت ولري:

FRST = 1000

SCND = 1001

ANSR = 1002

له دې وروسته به یوازې په خپلو نومونه ور غیر کوو. منځګری 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 موږ خپل پروګرام په لاندې ډول لیکلی شو:

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

اوس منځګری 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 لارښوونه کې CLA FRST، منځګری پروګرام CLA د 10 عملیاتي کوډ ته ژباړي او بیا په جدول کې د FRST ادرس لټوي، منځګری دا ادرس د 1000 په ډول مومي، او په دې ډول د ماشینواله ژبې لومړنۍ لارښوونه د 10 1000 په بڼه ځای په ځای کوي. کټ مټ په همدې ډول،

د ADD SCND لارښوونه 14 1001 ته ژباړي، او همداسې دوام کوي، او پړاو په پړاو د ماشينواله ژبې پروگرام چمتو کوي.

خو زموږ د برخې يو بل کار هم پاتې دی چې کولای شو هغه هم کمپیوټر ته ور له غاړې کړو. موږ اړتيا نه لرو چې کمپیوټر ته ووايو چې يو عدد چېرته ځای په ځای کړي او يوه لارښوونه چېرته ځای کړي، لکه تر دې دمه مو چې همداسې کول. بلکې د دې پر ځای ورته ويلی شو چې «FRST په 1000 کې ځای په ځای کړه، SCND په 1001 کې ځای کړه، او ANSR په 1002 کې». موږ بايد کمپیوټر ته يوازې دومره ووايو چې د CLA FRST لارښوونه 0000 ته ځي، ADD SCND په 0001 پورې اړوند ده او همداسې نور. دا ځل موږ يوازې کمپیوټر ته دومره وايو چې د 0000 په موقعيت پروگرام پيل کړي. په دې ترتيب، سمبولیک پروگرام به په لاندې ډول يوه 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 پروگرام پښتو ژباړه:

پروگرام په 0000 پیل کړه او ډېټا په 1000 پیل کړه.  
 د دوهم عدد لپاره یو ادرس بیل کړه.  
 د ځواب لپاره یو ادرس بیل کړه.  
 رجسټر پاک او لومړنی عدد ور زیات کړه.  
 دوهم عدد ور زیات کړه.  
 په A رجسټر کې ځواب ذخیره کړه.  
 پروگرام وټړه.

ورپسې په دې سمولیک پروگرام باندې د کار کولو لپاره منځګړی پروگرام پیلوو. مور وینو چې لومړني څلور پړاوونه د اصلي پروگرام چې د دوو عددونو جمع کول دي، حقیقي برخه نه دي، بلکې یوازې هغه لارښوونې دي چې منځګړي پروگرام ته وایي چې څه وکړي، او دې لارښوونو ته جعلي لارښوونې<sup>1</sup> 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 پروگرام سره هېڅ اړیکه نه لري، بلکې یوازې منځګړي ته معلومات چمتو کوي او ورته وایي چې مور په څه ډول غواړو چې زموږ پروگرام اجرا شي.

---

<sup>1</sup> Pseudo Instructions

د دې لپاره چې وگورو منځگړی دا سمبولیک پروگرام څنگه ماشینواله ژبې ته ژباړي، راځئ چې پراو په پراو یې نندارې ته کښو.

لومړنی پراو منځگړي ته وايي چې پروگرام باید د 1000 په ادرس پیل شي او هره راتلونکې لارښوونه 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 منځگړي ته وايي چې د FRST لپاره یو ادرس پیل کړي. په دې ډول منځگړی پروگرام له هغې ډلې څخه چې د ډېټا لپاره پیل 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 مور هغه ادرس ته د FRST په سمبول اشاره کوو، نو مور 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 منځگړي په خپله حافظه کې زېرمه کړی دی.

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

په همدې ډول، راتلونکي پړاو منځگړي ته وايي چې د ANSR لپاره يو ادرس بېل کړي، چې دا ځل 1002 دی. دا وروستی جعلي لارښوونه ده، او راتلونکي لارښوونه په خپله اصلي پروگرام پيلوي.

ورپسې لارښوونه CLA FRST ده، چې کمپيوټر يې 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، منځگړی د ADD SCND په 14 1001 ترجمه کوي او STA ANSR په 1002 30 ترجمه کوي. په پای کې منځگړی HLT په 00 ترجمه کوي، او په دې ډول د منځنۍ ژبې له پروگرام سره سم په بشپړ ډول د ماشينواله ژبې پروگرام چمتو کوي.

## پر ماشينواله ژبه د منځنۍ ژبې گټې

منځنۍ ژبه پر ماشينواله ژبه لاندینۍ گټې لري:

۱. **د کارونې او پوهېدنې لپاره اسانه:** منځنۍ ژبې د کارولو او ورباندې پوهېدلو په اساس اسانه دي، په دې چې د عددي عملياتي کوډونو پر ځای له نيمونیکونو څخه گټه اخلي او د ډېټا لپاره مناسب نومونه کاروي. له نيمونیکونو څخه گټه اخیستل په دې معنا ده چې نظرونو<sup>1</sup> ته معمولا اړتیا نه وي، بلکې پخپله پروگرام د پوهاوي وړ وي. د دې ترڅنګ، سمبولیکه پروگرامليکنه د پروگرام ليکونکي

---

<sup>1</sup> Comments

پرېمانه وخت او زحمت هم سپموي، په دې چې د ماشينواله ژبې د پروگرامونو په پرتله په اسانه ليکل کېږي.

**۲. د تېروتنو موندلو او سمولو اسانتيا:** کله چې په منځنۍ ژبه کې پروگرام ليکل کېږي، د تېروتنه شمېر کم وي، او کومې تېروتنې چې شوې وي د هغو موندل او سمول د نيمکونیکونه او ساحو لپاره د سمبولیکو نومونو له کبله اسانه وي. سرپرته پر دې، منځګړي پروگرامونه داسې ډيزاين شوي وي چې په خپلکاره<sup>1</sup> ډول تېروتنې په ګوته کوي. که چېرته مور داسې نيمونیک يا نوم وکاروو چې هېڅکله نه وي تعريف شوی، نو منځګړی پروگرام به د تېروتنې په ډول د دا ډول نوم شتون ته نغوته وکړي. د مثال په ډول، فرض کړئ چې په سمبولیک پروگرام کې يوه لارښوونه ADD AREA لوستل کېږي، او له مور نه دا هېر شوي وي چې AREA تعريف کړو، نو منځګړی پروگرام به په خپل جدول کې په AREA پسې سترګې وغړوي؛ نو که يې ونه مومي، تېروتنه به را برسېره کړي.

**۳. د بدلون راوستو اسانتيا:** د پروگرام ليکونکو لپاره د ماشينواله ژبو د پروگرامونو په پرتله د منځنۍ ژبې په پروگرامونو کې بدلون او اصلاح راوستل اسانه دي. د دې اصلي لامل دا دی چې په دې ژبه پوهېدل اسانه دي، په دې اساس د لارښوونو بدلون، سمون او تېروتنو په نښه کول هم اسانه وي. تر دې ورهاخوا، له پروگرام څخه د ځانګړو لارښوونو د درج او ايستلو لپاره د دې اړتيا هم نه وي

---

<sup>1</sup> Automatic

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

**۴. د ادرسونو په تړاو د اندېښنو نشتوالی:** د منځنۍ ژبې یو له لویو گڼو 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 مور یو اوږد ماشینواله پروگرام لیکلی چې څو پړاوونه لري او د پروگرام دننه یې بېرته خپل ځان ته څو ځله مراجعه هم کړې، لکه څرخ<sup>1</sup> کارول، یا د ادرسونو بدلون، او داسې نور. خو په پای کې مو ناڅاپه پام شي چې د پروگرام په منځ کې یوه لارښوونه را څخه پاتې شوې. که چېرته مور هغه لارښوونه درج کوو، نو باید ټولې لارښوونې مو په یادو زده وي، او ټول پروگرام ټکی په ټکی را وگورو تر څو هغه ځای پیدا کړو چې له هغه ځایه نورو پړاوونو ته مراجعه شوې وي. دا یو سخت کار دی. خو که چېرته مور همدا پروگرام په سمبولیکه ژبه ولیکو، نو مور یوازې یوه اضافي لارښوونه ور زیاتوو، او منځګړی پروگرام په خپلکاره ډول هغه ټول پړاوونه سرته ورسوي.

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

---

<sup>1</sup> Loop

د دې لپاره چې دواړه پروگرامونه وليکل شي، نو ښکاره ده چې يو بايد له هغه ځايه لرې کړای شي. په ماشينواله ژبه کې دا کار ډېر پېچلی دی. خو په منځنۍ ژبه کې يوازې بايد 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 منځګړي په مرسته سمبولیک پروگرام يو ځل بيا ځغلوو. د دې لارښوونو معادل ماشينواله پروگرام به خپل پروگرام د 1000 په ځای له 3000 څخه پيل کړي، او په دې ډول به له بل پروگرام سره هېڅ تکر نه خوري. په بل عبارت، سمبولیکه ژبه په ډېرې اسانۍ پروگرامونه د حافظې له يوې برخې څخه بلې هغې ته لېږدولی شي؛ په دې اساس موږ ويلای شو چې د منځنۍ ژبې پروگرامونه په اسانۍ د انتقال وړ دي ځکه چې د دوی موقعيت يوازې د لومړنۍ لارښوونې په بدلون، په اسانه بدلېږي. دا کار په ماشينواله پروگرامليکنه کې دومره په اسانه سرته نه رسېږي.

**٦. د ماشينواله ژبې اغېزمنتيا:** د هغو گټو ترڅنګ چې پورته يادې شوې، د منځنۍ ژبې پروگرام د خپل معادل ماشيني کوډ له اغېزمنتيا او گټو څخه هم برخمن دی، ځکه د يوې منځنۍ ژبې د لارښوونو او د هغه د معادل ماشينواله 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، خومره 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 (مېکرو ژبې) هم شته چې بنایي 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 لاندیني 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 يوه ځانگړي جوړښت او ماډل د پروسېس کونکي واحد لپاره ډيزاين شوې وي. که چرته د کمپيوټر د بدلولو پرېکړه وشي، نو معمولا د يوې نوې ژبې زده کړې او د ټولو شته پروگرامونو اړولو ته اړتيا وي چې دا اقدام د لوړې بېبې غوښتونکي دی.

**۲. هارډوېر پوهنې ته اړتيا:** دا چې منځنۍ ژبې ماشين-اړونده دي، نو پروگرام ليکونکي اړ دی چې د پروگرام د ليکلو پر مهال، د يوه ځانگړي ماشين له ځانگړنو او شرايطو څخه خبر وي. په منځنۍ ژبه کې پروگرام ليکونکي بايد پوه وي چې د هغه ماشين څنگه کار کوي او د خپل کمپيوټر د منطقي جوړښت په هکله پوره پوهاوی ولري تر څو وکولای شي چې يو ښه پروگرام وليکي.

**۳. د ماشينواله کچې کوډ<sup>1</sup> ليکل:** په منځنۍ ژبه کې هم لارښوونې د ماشينواله کچې په کوډ ليکل کېږي – په دې معنا چې د ماشينواله کوډ يوه لارښوونه خپل ځای د منځنۍ ژبې يوې لارښوونې ته پرېږدي.

ماشينواله او منځنۍ ژبې چې ماشين-اړونده ژبې دي، د ټينو ژبو<sup>2</sup> په نوم هم يادېږي.

## د مېکرو لارښوونو منځنۍ ژبه

<sup>1</sup> Machine-level code

<sup>2</sup> 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 خو 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، بنایي 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 لارښوونه 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 ګوډ کولو د پروسې د ګړندي 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 پړاوونه 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 اړ کې د مېکرو لارښوونو له یوې لارښوونې څخه د ګڼ 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 وليدل چې په ماشينواله يا منځنۍ ژبه کې د پروگرامونو ليکل د کمپيوټر د داخلي جوړښت په اړه ژورې پوهې ته اړتيا لري. حال دا چې په دې ژبو کې د پروگرام ليکلو لپاره پروگرام ليکونکی بايد د کمپيوټر ټول عملياتي کوډونه (عددي يا نيمونیک) په يادو زده کړي او په پوره تفصيل په دې پوه وي چې هر کوډ څه کار کوي او څنگه د کمپيوټر بېلا بېل رجسټرونه اغېزمنوي. که څه هم، مور پورته ولوستل چې د دې لپاره چې يو ښه کمپيوټري پروگرام وليکل شي، نو پروگرام ليکونکی بايد د هغه پروگرام پر منطق تمرکز وکړي، نه دا چې د کمپيوټر د داخلي جوړښت په تړاو اندېښنه وکړي. د دې لپاره چې پروگرام ليکونکي هوسا کړای شي تر څو د کمپيوټر په داخلي جوړښت باندې له پوهېدو پرته کمپيوټر استفاده کړي، عالي ژبې رامنځته شوې.

عالي ژبو، د دې پر ځای چې ماشين-اړونده اوسي، ډېره ډډه د ستونزې په حل کولو اچولې. دغه ژبې پروگرام ليکونکی د دې جوگه گرځوي چې د انگليسي کلماتو او د عادي رياضيکي سمبولونو په کارولو خپلې لارښوونې وليکي. په دې ډول د نوموړي لپاره دا اسانه کېږي چې د پروگرامليکنې په تفصيلاتو کې د سرگردانه کېدو پر ځای د ستونزې منطق ته متوجه شي. ښکاره ده چې هغه دوه-برخيز فارمټ چې په ماشينواله يا منځنۍ ژبه کې د لارښوونو ليکلو لپاره ورته اړتيا وه، په عالي ژبو کې د لارښوونو ليکلو لپاره ضرور نه ده. د مثال په ډول، راځئ د دوو عددونو د (FRST & SCND) جمع کولو او حاصل يې په ANSR کې د

ذخیره کولو هماغه ستونزه فرض کړو. موږ دمخه ولیدل چې د دې کار د سرته رسولو لپاره درې 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<sup>1</sup> ته ور ګرځو، نو یوازې یوې لارښوونې لیکلو ته اړتیا ده:

$$\text{ANSR} = \text{FRST} + \text{SCND}$$

ښکاره ده چې د پورتنۍ لارښوونې لیکل او ورباندې 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.

---

<sup>1</sup> Fortran

عالي ژبې د ستونزواله ژبو<sup>1</sup> په نوم هم يادېږي، په دې چې مېکرو لارښوونې يوازې د دې لپاره تر کار لاندې راځي خو د يو خاص ډول ستونزو د حل کولو لپاره گټورې پرېوځي. په دې اساس هره ژبه د ستونزو د يوې ځانگړې ټولگې د حل کولو لپاره گټوره پرېوځي، او کېدای شي د ستونزو د نورو ډولونو لپاره په کامله توگه بې گټې اوسي. د مثال په ډول، که چېرته يوه عالي ژبه د کاروبار-ډوله کاريالونو لپاره چې د لوړ حجم ورکړيز، نسبتاً لږ پروسېس او د لوړ حجم راکړيز غواړي، نو دا ډول ژبه، يوه کاروبار-واله ژبه ده. بل خوا، هغه ژبې چې د پېچلو محاسبو لپاره غوره وي، خو د لويو ډېټا فایلونو د سمبالښت توان په ځان کې ونه لري، رياضي-واله ژبه ده. په دې اساس، يوه ستونزواله ژبه داسې ډيزاين کېږي چې لارښوونې يې د هغې ستونزې د ژبې په توگه وليکل شي. د مثال په ډول، يو ساينسپوه د يوې ساينسي ژبې په کارولو له ساينټيفيکو فورمولونو څخه گټه اخيستلای شي، او يو سوداگر د يوې کاروباري ژبې په مرسته له کاروباري اصطلاحاتو څخه گټه اخيستلای شي. په دې اساس، عالي ژبې په عموم کې په اسانه زده او ليکل کېږي.

---

<sup>1</sup> Problem-oriented

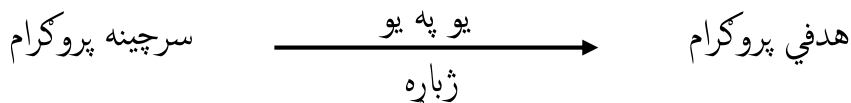
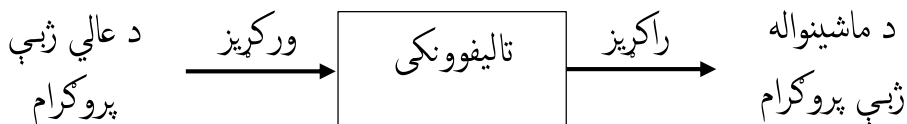
## تالیفونکي<sup>1</sup>

دا چې کمپیوټر یوازې د ماشین د کچې په لارښوونو باندې د پوهېدو وړتیا لري، نو د دې لپاره چې یو پروګرام اجرا کړای شي، اړتیا ده چې په عالي ژبه کې لیکل شوې لارښوونې لومړی ماشینواله لارښوونو ته واوړي. مور ولیدل چې منځنۍ ژبې د اوړون پروسېس<sup>2</sup> د سرته رسولو لپاره له منځګړي پروګرام څخه ګټه اخلي. په یوه عالي ژبه کې، همدا چاره د تالیفونکي له لوري پرمخ وړل کېږي. په دې اساس، تالیفونکي (چې له یوه منځګړي پروګرام څخه ډېر پېچلی وي) د ژباړې هغه پروګرام دی، چې د عالي ژبې لارښوونې ماشینواله ژبې ته ترجمه کوي. دې پروګرام ته په دې تالیفونکي ویل کېږي چې د عالي ژبې د پروګرام د یوې لارښوونې لپاره د ماشینواله ژبې د لارښوونو 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 تالیفونکي له خوا ماشینواله ژبې ته واوړي، له هغه وروسته هدفي پروګرام ورته ویل کېږي. څنګه چې په لاندې شکل کې لیدل کېږي، د تالیفونکي (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 لارښوونو څخه جوړ وي. پام مو وي چې سرچینه پروګرام او هدفي پروګرام دواړه یو دي، مګر د تحول په بېلابېلو پړاوونو کې قرار لري.

---

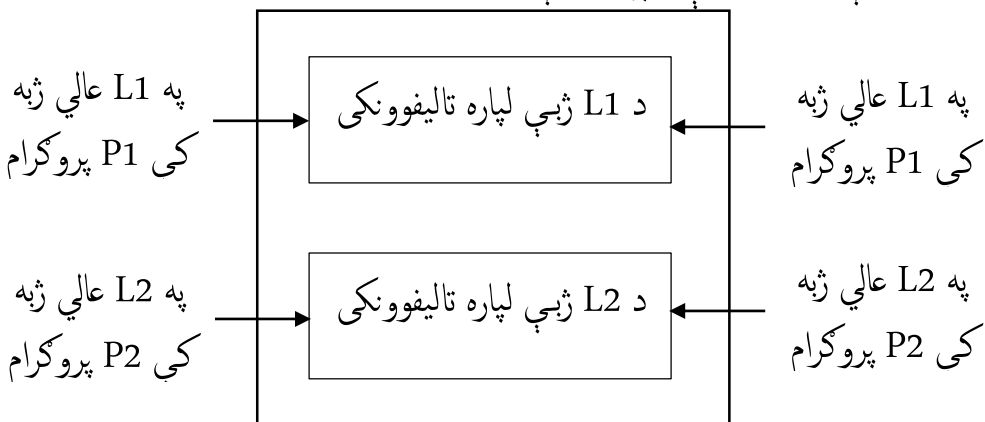
<sup>1</sup> Compilers

<sup>2</sup> Conversion process

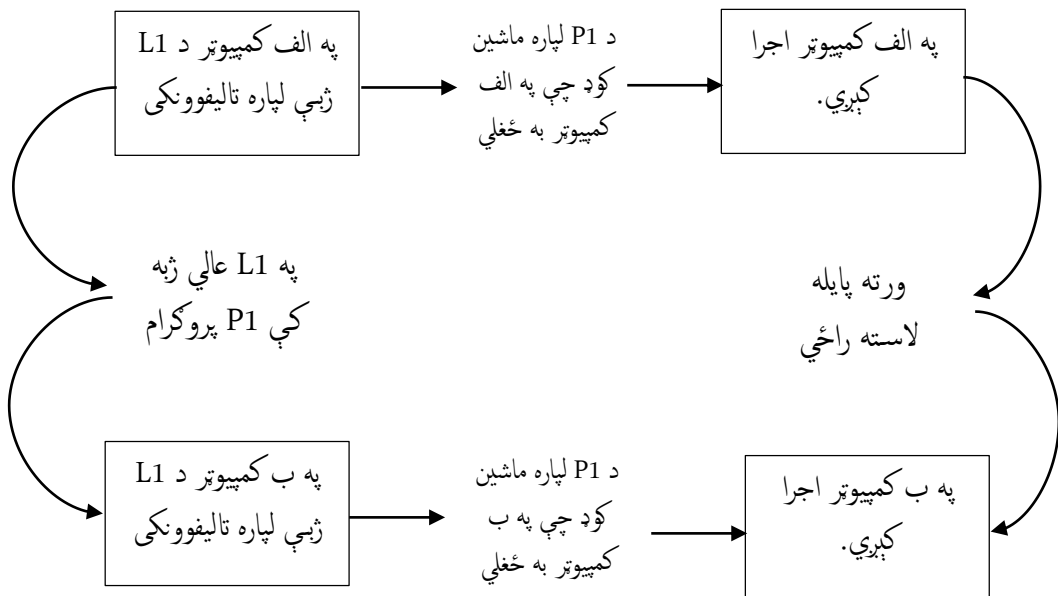


پورتنی انځور د تالیفونکي د ژباړې پروسه تشریح کوي.

یو تالیفونکی یوازې هغه سرچینه پروگرامونه تالیف کولای شي چې په هغه ژبه کې لیکل شوي وي چې هغه تالیفونکی ورته ډیزاین شوی وي. د مثال په ډول، د فورټران تالیفونکی یوازې د هغو سرچینه پروگرامونو د تالیف وړتیا لري چې په فورټران کې لیکل شوی وي، او له همدې امله هر ماشین د هرې عالي ژبې لپاره یوه بېل تالیفونکي 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 تالیفونکو شرط تشریح کوي .



د عالي ژبو د ماشین-خپلواکۍ ځانګړنه تشریح کوي. په مختلفو کمپیوټرونو باندې د یوې ژبې د ځغلولو لپاره مختلف تالیفونکي 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 اړتیا ده چې هر ماشین باید د یوې ځانګړې ژبې لکه L1 ، لپاره خپل تالیفونکی ولري. پورتنی انځور دا تشریح کوي چې د مختلفو تالیفونکو په مټ



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

ژباړونکي هغه ستر پروگرامونه دي چې په تلپاتې ډول په مرستندویه زېرمه کې پراته وي. کله چې د یوه پروگرام ژباړه سرته رسېږي، نو هغه لارښوونې د کمپیوټر اصلي زېرمې ته کاپي کېږي. تالیفونکي چې پخپله یو پروگرام دی، په مرکزي پروسیس کونکي واحد کې اجرا کېږي. کله چې یو پروگرام ژباړل کېږي، نو تالیفونکي د سرچینه پروگرام هر عبارت تجزیه کوي او د ماشینواله لارښوونو یوه لړۍ تولیدوي، چې کله هغه اجرا شي، نو هماغه محاسبه سرته رسېږي چې په هغو لارښوونو کې تعریف شوې وي. کوم وخت چې تالیفونکي هر عبارت تجزیه کوي، له هماغه سره سم تالیفونکي په پروگرام کې شته تېروتنې هم را برسېره کوي. دغه تېروتنو ته تشخیصی تېروتنې (diagnostic errors) ویل کېږي. تالیفونکي کولای شي چې په یوه سرچینه پروگرام کې لاندې ډول تېروتنې تشخیص کړي:

(الف) ناقانونه توري<sup>1</sup>

(ب) د تورو ناقانونه ترکیب<sup>2</sup>

<sup>1</sup> Illegal characters

<sup>2</sup> Illegal combination of characters

ج) په پروگرام کې د لارښوونو نامناسب اوډون (ترتیب)<sup>1</sup>

هغه سرچینه پروگرام چې تالیفونکي یوه تېروتنه پکې تشخیص کړې وي، هډفي پروگرام ته نه تالیف کېږي. په دې ځای کې تالیفونکي د یوه مناسب پیغام په مرسته هغه تېروتنه په گوته کوي، او ورسره د هغو کوډ شوو تېروتنو پیغامونه هم ښکاره کوي چې خاص ډول تېروتنو ته اشاره کوي. د تېروتنې تشخیص له پروگرام لیکونکي سره د لوړ ارزښت مرسته ده.

خو پام مو وي چې یو تالیفونکي منطقي تېروتنې نه شي تشخیص کولای او په پروگرام کې یوازې ګرامري (د نحوي ترکیب) تېروتنې تشخیصوي. نوموړی د انسان په نیتونو نه شي پوهېدلی. د مثال په ډول، که یو چا په تېروتنه کې د یوه تن عمر 25- درج کړ، حال دا چې هدف یې 25+ وو، نو تالیفونکي دا تېروتنه نه شي تشخیصولای. هغه پروګرامونه چې دا ډول تېروتنې لري، په پوره بریالیتوب تالیف کېږي او هډفي کوډ د تېروتنې له پیغام پرته لاسته راځي. دا بېله خبره ده چې دا ډول پروګرامونه چې کله اجرا کېږي، نو سم ځوابونه نه شي وړاندې کولای. نو منطقي تېروتنې 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.

<sup>1</sup> Improper sequencing of program instructions

## ترجمان<sup>1</sup>

ترجمان، د ژبو د ژباړونکي يو بل ډول دی چې عالي ژبې ماشينواله کوډ ته ترجمه کوي. دا ترجمان د عالي ژبې يو عبارت ور اوچتوي او ماشينواله لارښوونې ته يې ژباړې چې ورپسې په سملاسي ډول يې اجرا هم ترسره کېږي. نوموړی ترجمان چې کله د عالي ژبو له عبارت سره مخ شي، نو د ژباړې او اجرا پروسې يو په بل پسې ترسره کېږي. په بل عبارت، يو ترجمان يوه لارښوونه ژباړي، او د کنټرول واحد پر هماغه لاسته راغلي ماشينواله کوډ اجرات ترسره کوي، بيا بله لارښوونه ژباړل کېږي، او د کنټرول واحد پر هغه ماشينواله کوډ اجرات ترسره کوي، او همداسې دوام کوي. دا پروگرام له تالیفونکي څخه توپير لري چې يوازې ټول سرچينه پروگرام يوه هدفې پروگرام ته ژباړي او د اجرا عمليه پکې شامله نه وي. د تالیفونکو په خوا کې ټول سرچينه پروگرام د هغه معادل ماشينواله پروگرام ته ژباړل کېږي. ورپسې هغه هدفې کوډ په راتلونکي کې د استفادې لپاره په تلپاتې ډول زېرمه کېږي او د پروگرام په هر ځل اجرا کې استفاده ورڅخه کېږي. په دې اساس د يوه پروگرام د مکرر اجرا کېدو لپاره مکرر تالیف ته اړتيا 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 يو په بل پسې صورت نيسي. بل ځل که هغې لارښوونې ته اړتيا شي، نو لومړی بايد يو ځل بيا ترجمه او ماشينواله ژبې

---

<sup>1</sup> Interpreter

ته وژباړل شي. د مثال په ډول، د يوه څرخ<sup>1</sup> دننه د لارښوونو د مکرر پروسېس لپاره، د څرخ هره لارښوونه بايد هر ځل چې څرخ اجرا کېږي، ترجمه شي. ترجمان پروگرامونه تر ډېره له مايکرو کمپیوټرونو (کوچنیو کمپیوټرونو) سره ګډ راځي. پر تالیفونکي باندې د ترجمان ګټه 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 تېروتنو سمولو لپاره تغیرات راوستل کېږي، نو لومړی باید تالیف شي. د دې ترڅنګ، تالیفونکی د يوه 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، تالیفونکي او ژباړونکي هغه سیستمي پوستغالي دي چې د ګټه اخیستونکي له لوري لیکل شوی سرچینه پروګرام، یوه داسې هدفي پروګرام ته ژباړي چې د کمپیوټر د سختغالي لپاره معنا ولري. دې ژباړونکو ته ژبني پروسېس کوونکي<sup>2</sup> هم ویل کېږي، ځکه چې دوی یوه ځانګړې ژبه پروسېس کوي.

<sup>1</sup> Loop<sup>2</sup> Language Processors

د پورتنیو خبرو پر بنسټ، مور دې نتیجې ته رسېږو چې په عموم کې، یوه پروګرامي ژبه باید له لاندې ځانګړنو څخه برخمنه وي خو 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 لارښوونه اوسي چې د ماشینواله ژبې څو لارښوونو ته ژباړل کېدای وشي.

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

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

۵. ژبه باید په خپل فطرت کې تجربې 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 يې يوه ډېره ارزښتناکه گټه ده، ځکه دا په دې معنا ده چې يوه کمپنی چې خپل کمپیوټرونه تغیروي - يا حتی يوه پېل تولیدوونکي ته انتقال کېږي، نو هېڅ اړتیا نشته چې ټول هغه پروگرامونه بیا ځلې وليکل شي چې همدا اوس تر کار لاندې دي. حتی هغه پروگرامونه چې په عالي ژبو کې ليکل شوي وي، يوازې ځينو بدلونونو ته اړتیا لري، خو دغه بدلونونه نسبتاً واره او له ډېر زیار پرته په اسانۍ ترسره کېدای شي. په بل عبارت، هغه پروگرام چې په عالي ژبه کې ليکل شوی وي په څو مختلفو کمپیوټرونو کې په ډېر لږ زحمت او يا 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 مور يې په خپل ورځني ژوند کې کاروو. په دې اساس، د دې ژبو زده کړه او کارونه اسانه ده. پروگرام ليکونکي 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 يې ذخیره کړي، څه ورسره وکړي، او داسې نور. په دې معنا چې پروگرام ليکونکي ته هېڅ پکار نه ده چې په ماشینواله لارښوونو، د ډېټا په فارمت، او داسې نورو موضوعاتو پوه شي. که څه هم، دا ډول پوهه يو څه مطلوبه وي ځکه چې پروگرام ليکونکي ته اجازه ورکوي چې له سیستم څخه په ډېر اغېزمن ډول گټه پورته کړي.

**۳. کمپټروټنې:** د عالي ژبو په برخه کې، دا چې پروگرام لیکونکی اړتیا نه لري چې هغه ټول کوچني پړاوونه 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 زیروي تر څو چې کوم شی ویجاړ نه شي. د دې ترڅنګ، تالیفونکي په داسې ډول ډیراین شوي وي چې په خپلکاره ډول د پروگرام لیکونکي تېروتنې په نښه کولای او نیولای شي. په دې اساس، که پروگرام یوه نیمه تشخیصی تېروتنه ولري، نو پروگرام لیکونکی یې په اسانه ډول تشخیص کولای او سمولای شي.

**۴. ټیټ لگښت:** په عالي ژبو کې د پروگرامونو لیکل لږ وخت او کم زحمت غواړي، چې په طبیعي ډول د پروگراملیکنې ټیټ لگښت ته لاره برابروي. عموماً، د پروگرام د چمتو کولو د ټولو فازونو (کوډ کول، ډي بگ کول، ازمايښت کول، او داسې نورو) لگښت د منځنۍ او ماشینواله ژبو په پرتله ټیټ راځي.

**۵. غوره لارښودلیکنه<sup>1</sup>:** 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 پروگرام پر عباراتو له ستونزې سره بلد کس په اسانه

---

<sup>1</sup> Documentaion

پوهېدلای شي. د دا ډول پروگرامونو د لارښودلیکنې لپاره، ډېر کم او یا په عملي ډگر کې له نشت سره برابر، نظر عبارتونه<sup>1</sup> پکار وي.

**۶. اسانه بدلون:** په عالي ژبه کې لیکل شوي پروگرامونه د منځنۍ او ماشینواله ژبو د پروگرامونو 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 لارښوونو کې بدلونونه راوستل هم اسانه وي. له یوه پروگرام څخه د ځینو لارښوونو 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:

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

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

---

<sup>1</sup> comment statements



څخه يې د انعطاف کچه کمه وي. منځنۍ ژبه، پروگرام ليکونکي ته د ماشين ټولې ځانگړنې 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 توگه پروگرام 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 کتنې د هغو پر زيانونو وړ اوږي. ډېری کمپیوټري سيستمونه د زياتره پروگرامونو 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 گريس 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<sup>1</sup> يوه تجارتي او کاروباري ژبه وه چې له بهيرچارتونو<sup>2</sup> څخه يې په اسانه متحويات را ټولولای

<sup>1</sup> Flowmatic

<sup>2</sup> Flowcharts

شوی؛ په داسې حال کې چې ماتیمېټیک<sup>1</sup> یوه ریاضیکی 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.

---

<sup>1</sup> Mathematic

# « ډ کمپیوٲر پوهنې او ٲکنالوژۍ له پوهنغونډ »

## څخه

لیکوال: هېري هېنډرسن

ژباړه: رحمت شاه فراز

(۱)

## اېډا (Ada)

له ۱۹۶۰ز کلونو را په دې خوا، د امریکې د دفاع وزارت د سافټوېر جوړونې په هڅو کې ورځ تر بله له زیاتېدونکو ګډوډیو سره مخامخ وو. هر کله به چې یو نوی سافټوېر یا کارپال لکه د اړیکو کنټرولونکي ایجاد شو، نو په عامه توګه به په یوه ځانګړې پروګرامي ژبه کې لیکل کېده. په دې وخت کې له ۲۰۰۰ زیاتې ژبې د سافټوېر جوړونې لپاره کارول کېدې، نو په دې اساس د دومره لوی شمېر ناهمغږه<sup>۱</sup> سیستمونو رغونه او اپګرېډ کول ستونزمن او ګرانه پرېوته. په ۱۹۷۷ز کې، د دفاع وزارت یوې کاري ډلې د یوې عام-هدفه<sup>۲</sup> پروګرامي ژبې د جوړولو طرحه وړاندیز کړه چې د وسلو له کنټرول او لارښود سیستمونو نیولې بیا په ګودامونو کې د بار-کوډ سکېن کونکو<sup>۳</sup> پورې د سافټوېرونو د جوړولو لپاره د کارولو وړ وي. په دې لړ کې هغه طرحه چې بریالۍ شوه، اېډا نوم ورباندې کېښودل شو چې د نولسمې پېړۍ د کمپیوټرپوهنې د سرلارې اېډا لاولېس په نوم پسې نومول شوې وه. د نوموړې ژبې د ځانګړنو له بیاکتنې او تصحیح

---

<sup>1</sup> incompatible

<sup>2</sup> general-purpose

<sup>3</sup> scanners

وروسته، د امریکې د ملي معیارونو ادارې<sup>1</sup> په ۱۹۸۳ز کې په رسمي توګه اېدا یوه معیاري ژبه اعلان کړه. د دې ژبې لومړنۍ نسخه<sup>2</sup> په اېدا-۸۳ هم یادېږي.

## د ژبې ځانګړنې

د اېدا 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<sup>3</sup> را نغاړي او د ناڅاپي جمپ او یا «goto» له کارولو څخه مخنیوی کوي.

اېدا (د کنټرول جوړښتونو او فرعي پروګرامونو<sup>4</sup> په ګډون) د معیاري ژبو له ځانګړنو څخه برخمنه ژبه ده او دغه راز د ګټه اخیستونکي له خوا د تعریف شوو ډېټا ډولونو<sup>5</sup> پېکجونه هم په ځان کې را نغاړي 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 اېدا یو پروګرام هغه ډول عمومي بڼه لري چې په

<sup>1</sup> ANSI – Americal National Standards Institute

<sup>2</sup> version

<sup>3</sup> control structures

<sup>4</sup> subprograms

<sup>5</sup> data types

پاسکال ژبه کې کارېدله. (پام مو وي چې په پرېرمخ<sup>1</sup> لیکل شوي لغتونه د ژبې اړین ویوکی<sup>2</sup> 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 پروگرام لومړنی کرښه دا مشخصوي چې په پروگرام 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 ښودلو او را اخیستلو لپاره کارول کېږي. د مثال په ډول، د Ada.Text.IO پېکج په خپل ځان کې لاندې مشخصات<sup>3</sup> را ټولوي:

---

<sup>1</sup> boldface

<sup>2</sup> keywords

<sup>3</sup> 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 تړلو، 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 پرېمانه ځانگړي پېکجونه هم مشتق کېدای شي.

په اصلي پروگرام کې **Begin** د ډېټا اصلي پروسېس پیلوي چې په دې ځای کې د Put فنکشن په مرسته Ada.Text.IO فنکشن یو پیغام څرگندوي او د Get\_Line د فنکشن په وسیله د گټه اخیستونکي ځواب<sup>1</sup> ترلاسه کېږي، او ورپسې یو ځل بیا د Put فنکشن په کارولو هغه متن پر اړیکځای<sup>2</sup> بنودل کېږي چې سملاسي لیکل شوی وي.

---

<sup>1</sup> user response

<sup>2</sup> interface

اېدا د لویو او پېچلو سافټوېري پروژو لپاره یوه مناسبه ژبه ده، ځکه چې د پېکجونو کارول په یوه ډېټا ډول باندې د کار کولو او پلي کولو تفصیلات له نورو ګټه اخیستونکو څخه پټوي او ساتنه یې کوي. هغه پروګرام لیکونکی چې په پروګرام کې یې یو داسې پېکج کارول شوی وي چې دا مشخصه کړي چې کوم پارامترونه<sup>1</sup> باید له هر فنکشن سره وکارول شي، نو په دې توګه نوموړی پروګرام لیکونکی له ښکارېدونې اړیکې<sup>2</sup> څخه ګټه نه شي پورته کولای. د اېدا تالیفونکي په ډېر غوره ډول قانون بندي شوي<sup>3</sup> تالیفونکي دي خو د ډېټا د بېلا بېلو ډولونو د پروسېس لپاره له ټاکلو ځانګړنو سره سمون وځوري. همدا رنگه، یاده ژبه بیاوړې لیکنې<sup>4</sup> 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 ترې پیدا کېږي.

له دې امله چې دا ژبه په مختلطو<sup>5</sup> سیسټمونو او د رښتین-مهاله<sup>6</sup> فعالیتونو لپاره کارېږي، په دې اساس اېدا کې ګڼ شمېر داسې ځانګړنې ډیزاین شوې خو ماشینواله کوډ په اغېزمن ډول ورباندې ولیکل شي؛ بله دا چې نوموړې ژبه د دې

<sup>1</sup> Parameter

<sup>2</sup> visible interface

<sup>3</sup> validated

<sup>4</sup> strongly typed

<sup>5</sup> embedded

<sup>6</sup> real-time



ترڅنگ په منځنيو او نورو عالي ژبو کې د روټينونو<sup>1</sup> د ليکلو لپاره هم اسانتياوې وړاندې کوي. د دې ژبې تازه بڼه، اېډا-۹۵، د موازي پروگرام جوړونې<sup>2</sup> په ملاتړ هم ټينگار کوي. که څه هم د اېډا راتلونکي ناڅرگند دي، ځکه چې د دفاع وزارت نور دې ته اړتيا نه لري چې ياده ژبه په دولتي تړونونو کې وکاروي. خو په ځانگړې ډول د پراخو استازواله ژبو<sup>3</sup> د ځانگړنو په گډون چې د گڼشمېر توارث<sup>4</sup> د اړيکځايونو لپاره ملاتړ چمتو کوي، د متنکرينو<sup>5</sup> د غوره سمبالښت او نورو ډېټا ډولونو او فايلونو په برخه کې د اېډا پرمختگ تر اوسه پورې دوام لري؛ او دغه راز د رښتين-مهاله پروسېس او عددي پروسېس<sup>6</sup> 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.

---

<sup>1</sup> Routines د لارښوونو هغه ټولگه چې د يوه ځانگړي کار د ترسره کولو لپاره ډيزاين شوي وي. -  
<sup>2</sup> parallel programming  
<sup>3</sup> object oriented language  
<sup>4</sup> multiple inheritance  
<sup>5</sup> string  
<sup>6</sup> 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 خو د ساینسي محاسبو لپاره یوه اغېزمنه 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 ډېری پروګرامونه په ټیټو ژبو کې لیکل کېدل چې دا ډول ژبې د یوه خاص ماشین د سختوالي 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 دريځ لپاره نه وي ډیزاین شوې. له ۱۹۵۷ز نه مخکې، د جرمني <sup>1</sup>GAMM او د امریکې <sup>2</sup>ACM یا د کمپیوټري ماشینري شرکت ادارو خپل ځواکونه سره یو ځای کړل تر څو د داسې یوې ژبې لپاره مشخصات رامنځته کړي. د دې هڅو پایله د Zurich

<sup>1</sup> Gesellschaft für angewandte Mathematik und Mechanik

<sup>2</sup> Association for Computer Machinery

Report يا الګول-۵۸ په بڼه مخې ته راغله او الګول-۶۰ د دې ژبې هغه نسخه شوه چې تر ټولو لومړی په لويه کچه استعمال او تطبيق<sup>1</sup> کړای شوه.

## د ژبې ځانګړنې

الګول يوه بلاک-جوړبنسته<sup>2</sup> او پروسوي<sup>3</sup> ژبه ده. په نوموړې ژبه کې هر متحول<sup>4</sup> د ډېټا د يو کوچني شمېر ډولونو پورې د اړوند کېدو لپاره تعريفېږي د بېلګې په ډول کامل اعداد<sup>5</sup>، حقيقي اعداد او يا له دواړو څخه د يو ډول اعدادو لړۍ. په داسې حال کې چې د ډېټا ډولونو شمېر پکې محدود دی او دا چې د نوو ډولونو د تعريفولو لپاره اصلا اسانتيا پکې نشته، نو د تاليفوونکي ډول کنټه<sup>6</sup> (دا باوري کول چې د ډېټا توک<sup>7</sup> د تعريف شوي متحول له ډول سره سمون خوري او که نه) تر يوې کچې داسې خونديتوب چمتو کاوه چې په پخوانيو ژبو کې نه تر سترگو کېده.

د الګول يو پروګرام په لوی شمېر بېلا بېلې پروسې<sup>8</sup> لرلای شي او يا هم په خارج کې له تعريف شوو پروسو څخه ګټه پکې پورته کېدای شي. د دې ترڅنګ، د

<sup>1</sup> Implement

<sup>2</sup> block structured

<sup>3</sup> procedural

<sup>4</sup> variable

<sup>5</sup> integer

<sup>6</sup> type checking

<sup>7</sup> item

<sup>8</sup> procedure

ورته نومونو متحولونه چې په بېلا بېلو پروسوي بلاکونو کې تعريف شوي وي، پخپلو کې له يو بل سره ټکر نه کوي. يوه پروسه کولای شي چې بېرته خپل ځان را ووبولي او يا يې کال<sup>1</sup> کړي. نوموړې ژبه له معياري کنټرول جوړښتونو څخه هم برخمنه ده.

لانديني د الګول ساده پروګرام له ۱ څخه تر ۱۰ پورې اعداد په يوه کتار<sup>2</sup> کې زېرمه او جمع کوي، او بيا يې پايله 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

```

---

<sup>1</sup> call

<sup>2</sup> array

## د الګول ترکه

د الګول په هغه نسخه کې چې په الګول-۶۸ پېژندل کېږي، د (بولین<sup>1</sup> او سم/ناسم وېلیوګانو<sup>2</sup> په ګډون) د ډېټا ډولونو شمېر ډېر کړای شوی او د ګټه اخیستونکي له خوا تعریف شوي ډولونه او سټرکټونه<sup>3</sup> هم ور زیات شوي. سټرکټونه هغه ریکارډونه دي چې د ډېټا د مختلفو ډولونو لپاره خپلې ساحې لري. نښه کوونکي<sup>4</sup> چې وېلیوګانو ته مراجعه کوي، هم پکې پلي شوي، او پارامترونه هم له داسې اسانتیاوو څخه برخمن دي چې فنکشنونو ته ور تېرېدای<sup>5</sup> او یا بېرته ترې اخیستل کېدای شي.

که څه هم الګول د کمپیوټر په ځینو مرکزونو (په ځانګړي ډول په اروپا) کې د تولیدي ژبې<sup>6</sup> 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

<sup>1</sup> boolean

<sup>2</sup> values

<sup>3</sup> structs

<sup>4</sup> pointers

<sup>5</sup> pass

<sup>6</sup> 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 اېل<sup>1</sup> (APL)

دغه پروگرامي ژبه د هارورډ (او وروسته د آی بي اېم شرکت<sup>2</sup>) د يوه څېړونکي کښت اي. ايورسن له خوا په ۱۹۶۰ کلونو کې جوړه شوه، تر څو رياضيکي عمليې د کمپيوټر د استعمال لپاره په بنسټيزه او منظم ډول څرگندې کړای شي. د نوموړې ژبې په غونډ۳ شکل د رياضيکي عمليو د وړاندې کولو ځواک د ډېرې کټه اخيستونکو پام را جلب کړ او اې پي اېل په لنډ وخت کې يوه عام-هدفه کمپيوټري ژبه وگرځېده.

د بېسيک<sup>4</sup> د ډېرو نسخو په څېر، اې پي اېل يوه ترجمه کېدونکې ژبه ده، په دې معنا چې د پروگرام ليکونکي ورکړيز په ورته مهال کې ارزول کېږي او د دې ځانگړنې له مخې تعاملې<sup>5</sup> ځواب شونتيا مومي. د بېسيک او فورټران پر خلاف، اې پي اېل د کتارونو او ميټريکسونو په کېدون د ټولو رياضيکي عمليو لپاره ځواکمن او مستقيم ملاتړ چمتو کوي.

<sup>1</sup> A programming language – يوه پروگرامي ژبه

<sup>2</sup> IBM – International business machine

<sup>3</sup> Compact

<sup>4</sup> Basic

<sup>5</sup> Interactive



اې پي اېل له سلو زيات خپل ځاني<sup>1</sup> عاملونه<sup>2</sup> لري چې اوليه<sup>3</sup> نومېږي. يوازې د يوه يا دوو عاملونو په کارولو پروگرام ليکونکي کولای شي چې پېچلي کارونه لکه د عددي او مثلي عمليو استخراجول، د اعدادو ترتيبول<sup>4</sup>، او يا د کتارونو او ميټريکسونو بياترنتول ترسره کړي. (په اصل کې، د اې پي اېل تر ټولو ستر ځواک دا دی چې د څرگند-څرخونو<sup>5</sup> او يا خارجي لايبرېري فنکشنونو<sup>6</sup> له ترتيب کولو پرته کولای شي چې په مستقيم ډول پر ميټريکسونو عمليې ترسره کړي.

د يوه ساده مثال په توگه، د اې پي اېل د کوډ لاندې کرښه:

$$X [\Delta X]$$

د X کتار ترتيبوي. په ډېری پروگرامي ژبو کې دا کار بايد د يوه داسې ترتيبې الگوريتم<sup>7</sup> د کوډ ليکلو له لارې ترسره شي چې د هغه په لسه اوو کرښو کوډ کې ځاله-څرخونه<sup>8</sup> او لنډمهاله متحولين کارول شوي وي.

بل خوا، اې پي اېل د زياتره پروگرام ليکونکو لپاره گڼې نيمگړتياوې او عيبونه هم درلودل. په دې چې نوموړې ژبه د ډېری عاملونو لپاره يوناني توري کاروي، چې

<sup>1</sup> built-in

<sup>2</sup> Operators

<sup>3</sup> primitive

<sup>4</sup> sorting

<sup>5</sup> explicit loops

<sup>6</sup> Library functions

<sup>7</sup> Sorting algorithm

<sup>8</sup> nested loops

د دې تورو لپاره يوه ځانگړي فونټ ته اړتيا وه چې هغه مهال له آی بي اېم شرکت پرته په نورو سيستمونو کې نه موندل کېده. د دې لپاره چې ژبه ساده او پراخه کړای شي، د J په نوم يوه لهجه هم په کار واچول شوه چې يوازې معياري اسکي کرکټرونه<sup>1</sup> کاروي. د ډېری پروگرام ليکونکو لپاره د اې پي اېل رياضيکي معادلات گونگ رمز بندي شوي<sup>2</sup> وي چې په دې ډول يې د پروگرامونو بياکتنه او اصلاح ورته ستونزمنه کړې. خو پر دې سربېره هم، د اې پي اېل د مينوالو ځانگړې ډله په سترو کمپيوټري ټولنو کې لا هم له دې ژبې سره پراخه لېوالتيا ښيي.

### 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.

\*\*\*

---

<sup>1</sup> ASCII Characters

<sup>2</sup> cryptic

(۴)

## آک (Awk)

آک یوه متنواله ژبه<sup>1</sup> ده چې د «آلفرېډ وي. آهو»، «بريان ډبلیو»، «کرنیگان» او «پیتر جې. وېنبرگر» 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<sup>2</sup> سره سمون لرونکو نقشو<sup>3</sup> 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 پاس<sup>4</sup> لپاره د آک ژبې گڼ بدیلونه لیکل شوي دي.

د نورو متنواله ژبو په څېر، د آک یو پروگرام د هغو کمانډونو په یوه لیکلېر مشتمل وي چې د آک ژباړونکي په مرسته له یوه فایل څخه لوستل کېږي. د مثال په ډول د یونیکس لاندینی امر کرښه:

```
awk - f Myprogram > Report
```

---

<sup>1</sup> scripting language

<sup>2</sup> utility

<sup>3</sup> pattern

<sup>4</sup> DOS - Disk Operating System

د MyProgram له فایل څخه د آگ عبارات<sup>1</sup> د آگ ژباړونکي ته ور سپاري او بیا د پروگرام راکړیز Report نومي فایل ته استوي.

## د ژبې ځانګړنې

د آگ یو عبارت په خپل ځان کې دوه شیان رانغاړي: یوه داسې نقشه<sup>2</sup> چې له پایلې سره سمون وځوري 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 ګڼه 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

<sup>1</sup> Statements

<sup>2</sup> pattern

کرنېو د شمېر لپاره د شمېرونکو<sup>1</sup> په ګډون، ریاضیکي عملیې، له ساحو<sup>2</sup> څخه د متن د استخراجولو لپاره ګټور متنکرنیزه فنکشنونه او ریاضیکي او نسبتی عاملونه هم شته. دغه راز، کولای شو چې د څو اړخیزه print فنکشن له لارې راکړیز ته هم ډول ډول فارمتونه ورکړو چې د سي ژبې پروګرام لیکونکي ورسره بلدتیا لري.

آګ په یونیکس سیستمونو 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.

---

<sup>1</sup> counters

<sup>2</sup> field

\*\*\*

(۵)

## بېسيک (BASIC)

بېسيک يا «د مبتدي ټول-هدفه سمبولیک لارښود کوډ»<sup>1</sup> ژبه په ډارټماؤت کالج کې په ۱۹۶۴ کې د «جې. کېمپني» او «ټي. کرتز» له خوا جوړه شوه. هغه مهال د کالج ټول کېمپس له ترمينلونو سره تړل شوو په مهال-شريکه<sup>2</sup> کمپيوټرونو باندې سمبال وو چې دا کار په داسې وخت کې يو لوی نوبت وو چې ډېری کمپيوټرونه به يوازې له يوه ځای څخه د پنچکارتونو په وسيله پروگرام کېدل. جان. جې. کېمپني او تامس کرتز غوښتل چې يوه داسې ژبه جوړه کړي چې په اسانۍ سره زده او وکولای شي چې د کيبورډ له لارې ليکل شوي اوامر سمدلاسه تاليف او ځواب ورته ووايي. او په دې ډول د خپلو سيستمونو له تعاملې<sup>3</sup> ځانگړنې څخه گټه پورته کړي. د دوی دا فکر د هغه مهال له ژبو لکه کوبال، الگول او فورټران سره له يوې مخې په تضاد کې وو، چې په دې ژبو کې به پروگرامونه لومړی په بشپړ ډول ليکل کېدل او بيا به يې ازمايښت ترسره کېده.

---

<sup>1</sup> Basic All-Purpose Symbolic Instruction Codee

<sup>2</sup> time-shared

<sup>3</sup> interactivity

د هغو زرو ژبو په خلاف چې پنچکارتونه به یې کارول، د بېسیک پروگرامونو کې اړتیا نه وه چې اړین-ویوکی په مشخصو لیکو کې ولیکل شي. بلکې، عبارات د انګلیسي ژبو د کړنو په څېر لیکل کېدای شوی، پرته له دې چې لیکنښو او د واټن<sup>1</sup> په اړه ځانګړي اصول مراعت شي. په عمومي ډول، د پرېکړیزو<sup>2</sup> او کنټرول جوړښتونو لپاره نحوي ترکیب<sup>3</sup> د نورو ژبو په پرتله ساده دی. د مثال په ډول، یو 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
```

## بېسیک ژبه او کوچني کمپیوټرونه<sup>4</sup>

د ۱۹۶۰ او ۱۹۷۰ کلونو په اوږدو کې بېسیک ورځ په ورځ زیاتېدونکو مهال-شریکه کمپیوټرونو کې کارول کېدله. د نوموړې ژبې د ساده والي او په اسانۍ د استعمال کېدو ځانګړتیا دا ژبه د لنډو مرستندویه پوستغالو د لیکلو او په ځانګړي ډول د کمپیوټر پوهنې له پوهنځیو پرته په نورو پوهنځیو کې د کمپیوټر د بنسټي مفاهیمو د ښوونې لپاره ګټوره ګرځولې وه. د ۱۹۸۰ کلونو په لومړیو کې هغه

<sup>1</sup> spacing

<sup>2</sup> Decision

<sup>3</sup> Syntax

<sup>4</sup> Microcomputers

مهال چې لومړني شخصي کمپیوټرونه د پراخه استعمال لپاره منځته راغلل، د دې کمپیوټرونو د زېرمې ظرفیت عموماً له ۸ کیلو بايت څخه تر ۶۴ کیلو بايت پورې وو، چې د یوه ایډیټور، تالیفونکي 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 بېسیک د ژباړونکي یوه ساده نسخه د تش-لوسته زېرمې<sup>1</sup> په چپ کې ځای په ځای کېدلای شوی، هغسې چې په اپیل-۲ کمپیوټرونو، لومړنیو آی. بی. اېم شخصي کمپیوټرونو او لسگونو نورو شخصي کمپیوټرونو کې دود وو. د بېسیک نورې پرمختللي نسخې (د تالیفونکو په ګډون) په ټپ<sup>2</sup> کې زېرمه کېدلای شوی. (بیل ګېټس نومي ځوان سوداګر چې لومړی ځل یې دا ډول تولیدات خرڅلاو ته وړاندې کړل).

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

---

<sup>1</sup> ROM – Read-Only-Memory

<sup>2</sup> Tape

<sup>3</sup> Dialects

<sup>4</sup> Peek



پکې راوستل<sup>1</sup> - د عملیو د ترسره کولو لپاره د اوامرو شتون د بېسیک پروگرامونه د ځانگړي دریخ<sup>2</sup> لپاره د کارول کېدو وړ وگرځول.

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 ځانگړي ډول د جوړونې-مختلط-چاپېریال<sup>3</sup> چې په سانډیاګو کې د کالیفورنیا په پوهنتون کې جوړ شو) او سي (د یونیکس له ټولني څخه) په څېر ژبو ته پرېښودل پیل کړل.

## تنقید او لرلید

د بېسک زیاتره نسخو له کرښه-ګڼو<sup>4</sup> (د هغو لومړنیو متن اېډیټ کوونکو یو میراث چې کرښه په کرښه به یې کار کاوه) څخه ګټه اخیستله او همدا رنگه د دې لپاره چې د پروگرام کنټرول یوې ځانگړې کرښې ته ټوپ کړي، د goto عبارت هم پکې کارول کېدای شوی. که څه هم په نوموړې ژبه کې ساده فرعي پروگرامونه 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

<sup>1</sup> poke

<sup>2</sup> Specfic Platform

<sup>3</sup> integrated development environment

<sup>4</sup> Line numbers

وو. په اصل کې به ټول متحولین نړیوال<sup>1</sup> وو، په دې معنا چې متحولینو ته به د پروگرام له هرې برخې څخه لاسرسی کېدای شوی، چې د دې خطر یې رامنځته کاوه چې په بې پامۍ سره یې وپلټول شي.

څنگه چې د جوړښتواله پروگرام جوړونې<sup>2</sup> اصولو وده وکړه، د بېسیک جوړښتواله نیمګړتیاوو دا ژبه د هغو کمپیوټر پوهانو له پامه وغورځوله چې پاسکال ته یې د پروگراملیکني د ښوونې او سي ژبې ته یې د سیستمي پروگراملیکني لپاره لومړیتوب ورکاوه. په ۱۹۸۴ کې، د بېسیک اصلي جوړوونکو هغو ستونزو ته ځواب ووايه چې دوی د «د کوڅې بېسیک» په نوم پېژندل او «رنښتینی بېسیک» یې بازار ته وړاندې کړه، چې د نوموړې ژبې یوه عصري او ښه رغېدلې نسخه وه او د بېسیک-۱۹۸۸-ANSI نسخه د ژبو معیاري ځانګړنې هم ور خپلې کړې. دې هڅو یوازې محدود اغېز درلود. مایکروسافت کمپنی د سیستمي پروگراملیکني لپاره د بېسیک نوې نسخې رامنځته کړې (چټک بېسک<sup>3</sup> په ۱۹۸۰ کلونو او عیني بېسیک<sup>4</sup> په ۱۹۹۰ کلونو کې) چې له پرمختللو کنټرول جوړښتونو او ډېټا ډولونو څخه برخمنې وې او دغه راز د بې ګټو کرښه-ګڼو اړتیا یې هم له منځه وړې وه. عیني بېسک په ځانګړې توګه د پام وړ بریالیتوب ترلاسه کړ چې د دودیز بېسیک تعاملې وړتیا او له وړاندې څخه د

---

<sup>1</sup> Global

<sup>2</sup> Structured Programming

<sup>3</sup> quick basic

<sup>4</sup> visual basic

جوړ شوو ځواکمنو کنټرولونو پیکج ته د لاسرسي له اسانتیاوو څخه ترکیب وو. دغو پیکجونو مېنوګانې<sup>1</sup>، مکالموي چوکاټونه<sup>2</sup> او د وینډوز د اړیکځای نور ګڼ مشخصات چمتو کول. د عیني بېسیک اوسنیو بڼو تر ډېره استازواله<sup>3</sup> حالت ور خپل کړی او د سي پلس پلس په څېر له کلاسونو څخه ګټه اخلې.

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 تعاملې تکلاره<sup>4</sup>) چې دا ژبه یې له نورو بېلوله، دمکړۍ 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 Petruscha. *VB and VBA in a Nutshell: The*

---

<sup>1</sup> Menus

<sup>2</sup> Dialog Box

<sup>3</sup> object oriented

<sup>4</sup> Interactive approach to programming

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

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

\*\*\*

(٦)

سي(C)

سي پروگرامي 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.<sup>1</sup> او بي.<sup>2</sup> ژبو ايښودل شوی وو. سي د لومړي ځل لپاره په DEC PDP-11 کمپیوټرونو کې وکارول شوه چې په همدې وروستيو وختونو کې جوړ شوی يونيکس عامل سيستم باندې ځغلبده، په دې ځای کې سي ژبې د گڼو مرستندويه پوستغالو د جوړولو لپاره د PDP منځنۍ ژبې لپاره د عالي ژبې بدیل وړاندې کړ چې دې کار يونيکس ته انعطاف ور په برخه کړ. له ۱۹۸۰ کلونو را په دېخوا، سي او د

---

<sup>1</sup> BCPL

<sup>2</sup> B

هغې بچونې سي پلس پلس، تر ټولو زياتې کارېدونکې پروگرامي ژبې گرځېدلې دي.

## د ژبې ځانگړنې

د لومړنۍ الگول او تر يو څه بريده پاسکال په څېر، سي يوه پروسوي ژبه<sup>1</sup> ده چې د هغه پروگرامليکني فلسفه انعکاس کوي چې د ۱۹۷۰ کلونو په اوږدو کې يې ورو په ورو بڼه وموندله. په عمومي ډول، د سي ژبې تگلاره داسې بيانېدای شي چې د کمپيوټر د رېسټيني فطرت لپاره اړينې ځانگړتياوې په غونډ او اغېزمن حالت کې وړاندې کوي. نوموړې ژبه له اساسي کنټرول جوړښتونو لکه `switch`، `if`، `do`، `while` 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 ځاني ډېټا ډولونه وړاندې کوي. د هر ډول ډېټا لپاره يو کتار<sup>2</sup> تعريف کېدای شي او يوه متنکرېنه د تورو د يوه کتار په څېر پکې پلي کېږي.

نښه کوونکي<sup>3</sup> (چې زېرمه ځايونو ته اشاره کوي) د بېلا بېلو موخو لپاره کارول کېږي، مثلاً په يوه کتار کې د ډېټا د ترتيبولو او يا اخذ کولو لپاره. په داسې حال کې چې په نښه کوونکو باندې پوهېدل د مبتديانو لپاره يو څه ستونزمن پرېوتی شي، خو په اصل کې دا ځانگړنه د سي ژبې هغه ټينگار انتقالوي چې ياده ژبه د

<sup>1</sup> Procedural language

<sup>2</sup> array

<sup>3</sup> pointers

سیستمي پروگراملیکنې 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<sup>1</sup> دی. دا شی په ځینو وختونو کې یوه سرگرداني هم جوړولی شي، لاندې شرط

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

---

<sup>1</sup> Assignment operation

تل Finished بنکاروي، 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<sup>1</sup> ++ او کمونې<sup>2</sup> -- عاملونو هم لري، چې په یوه شمېرونکي څرخ<sup>3</sup> 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. آرګومېنتونه، 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);
```

<sup>1</sup> Increment

<sup>2</sup> Decrement

<sup>3</sup> 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 پروگرام د پیل عبارتونه چې په # نښه پیل کېږي د پروسېس لارښودونه<sup>1</sup> بلل کېږي. دا لارښودونه تر تالیف کېدو مخکې په سرچینه کوډ کې تغیرات راولي. د #include لارښود پروگرام ته ځانګړی اصلي فایل ور اضافه کوي. د نورو ژبو برعکس، سي ژبه پخپل سر ډېری اساسي فنکشنونه نه لري، لکه ورکړیز/راکړیز یا I/O عبارتونه. د دې پر ځای، دا ټول فنکشنونه په ستانډر لایبرېري کې پراته وي. (د دې ترتیب موخه دا ده چې پخپله ژبه ساده او سپکه شي، او که چیرته نوموړی پروگرام یوه بل درېځ ته هم ورل کېږي، نو کار ورکړي.) د stdio.h فایل، په دې ځای کې یو سرلیک فایل<sup>2</sup> دی چې I/O فنکشنونه تعریفوي، لکه

---

<sup>1</sup> processor directive

<sup>2</sup> header file



`printf()` (چې فارمت شوي<sup>1</sup> معلومات پرنټ کوي) او `scanf()` (چې په پروگرام ډېټا لولي او فارمت ورکوي).

د پروگرام بله برخه هر هغه فنکشن تعریفوي چې وروسته به په پروگرام کې تعریف او کارېږي (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<sup>2</sup> 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 گټه 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 سې پروگرام باید یو اصلي فنکشن ولري. مېن هغه فنکشن ده چې هغه وخت ځغلي چې کله پروگرام اجرا (`execute`) کېږي. عموماً، 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

<sup>1</sup> formatted<sup>2</sup> 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 نحوې ترکیب یوه بېلگه ده.)

په پای کې، د Average فنکشن تعریفېږي. دا فنکشن د دېټا 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 پروگرام لیکونکی کولای شي چې د یوه اېډیټور 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 پروگرام په یونیکس سیستم باندې هم جوړ کړي. همدا سرچینه کوډ بیا د سي تالیفونکي (په دې ځای کې gcc) ته ورکول کېږي. سرچینه کوډ تالیف کېږي، اړیکه مومي او بیا د اجرائي پروگرام فایل a.out ترې چمتو کېږي. له هغه وروسته که همدا نوم په کمانډ پرمپټ کې ولیکل شي نو پروگرام ځغلي، چې لومړی اعداد اخلي او ورپسې یې اوسط راوباسي.

```
% gcc test.c
```

```
% a.out
```

```
5
```

```
7
```

```
9
```

```
.
```

```
The average is: 7.000000
```

**بریا اوبدلون**

له رامنځته کېدو درې لسيزې وروسته، سي په تاريخ کې يوه بريالۍ پروگرامي ژبه وگرځېده. د دې ترڅنګ چې نوموړې ژبه د يونيکس پروگرامليکني لپاره د خوښې وړ ژبه وه، نو څنګه چې کوچني کمپيوټرونه د دې جوګه شول چې عالي ژبې وځغوي، سي ژبه د مايکروسافټ داس، وينډوز او مېکنټاش پروگرامونو د ليکلو لپاره د خوښې وړ انتخاب وگرځېد. د بېلګې په ډول، د وينډوز لپاره د کاريال پروگرامي اړيکځای يا API<sup>1</sup> د سلګونو په شمېر د سي فنکشنونه، جوړښتونه او تعريفونه<sup>2</sup> لري.

خو پر دې سربېره، سي د کمپيوټر پوهانو ترمنځ له نيوکو او انتقاد څخه پټه نه ده پاتې شوې. د هغو اصطلاحاتو ترڅنګ چې د مرموز کوډ ليکنې<sup>3</sup> لپاره يو څوک هڅولای شي، د سي اصلي بڼه (چې کرنينګ او ريچي په The C Programming Language کتاب کې تعريف کړې) پارامترونه نه گوري تر څو دا باوري کړي چې د فنکشن په تعريف کې له وضع شويو ډېټا ډولونو سره سمون خوري او که نه. دا ستونزه په پروگرام کې د لوی شمېر داسې تېروتنو لامل گرځي چې موندل يې ستونزمن وي. خو د امريکې د ملي معيارونو ادارې له خوا معياري سي ژبه چې په پېچلو شرطونو ولاړه ده، په تاليفونکي کې د ډول-کتنې د ځانگړنې دا ستونزه تر ډېره له منځه وړې. تقريباً په همدې وخت کې،

<sup>1</sup> Application Programming Interface

<sup>2</sup> definitions

<sup>3</sup> cryptic coding

سي پلس پلس د سي ژبې د يوې زياتونې<sup>1</sup> او سمونې په توگه يوه استازواله ژبه وگرځېده. په داسې حال کې چې سي ژبې خپل ځای د نوو پروگرامونو د جوړولو لپاره سي پلس پلس او جاوا ته پرې ايښی، خو د سي پروگرام ليکونکو لپاره د نوو ژبو زده کړه نسبتاً اسانه وي او همدا رنگه د سي ژبې د کوچې 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

\*\*\*

---

<sup>1</sup> 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 بېلا بېلو اړخونو څخه ساده شوې او د مایکروسافت له اوسني پروګرامي دريځ سره د کارېدو لپاره خاصه شوې ژبه ده. سي شارپ يوه عام-هدفه او په بشپړ ډول استازواله ژبه ده – چې ټول فنکشنونه بايد د يوه کلاس يا سټرکټ د غړو په توګه تعريف شوي وي، تر دې چې اساسي ډېټا ټايبونه 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<sup>1</sup> لپاره سخت قوانين وضع شوي دي، چې ډېری مبهمو او پرېدونو<sup>2</sup> ته اجازه نه ورکوي (د بېلګې په ډول له عددي ډول څخه خپل استازي integer ته). د سي پلس پلس پر خلاف، سي شارپ کنټمبر توارث ته اجازه نه ورکوي (چېرته چې يو ډول له دوو يا زياتو اساسي ډولونو څخه مشتق کېږي)، او په دې ډول په لويو پوستغالي پروژو کې د کلاس د اړيکو په پېچلتيا کې د يوه اضافي پور څخه

---

<sup>1</sup> Conversion

<sup>2</sup> implicit conversion

مخنيوي کوي. (که څه هم، د گڼشمېر اړيکځايونو<sup>1</sup> په تعريفولو او يا يوه کلاس ته د لاسرسي د بېلابېلو طريقو په مت کولای شو چې دا کار ترسره کړو.)

د جاوا برعکس (خو د سي پلس پلس په څېر)، سي شارپ له نښه کونکو (او په ډېر خوندي ډول چې delegates بلل کېږي)، عددي ډېټا ډولونو<sup>2</sup>، سټرکټونو (چې کوچنيو کلاسونو ته ورته دي)، او امبارونې<sup>3</sup> (د عاملونو لپاره گڼشمېر تعريفونه) څخه برخمنه ژبه ده. د نوموړې ژبې نوې بڼه، C# 3.0 (چې په ۲۰۰۷ کې منځته راغله)، د لست پروسېس او فنکشنل پروگرام ليکنې لپاره اضافي ځانگړتياوې هم چمتو کوي.

د «Hello World» معياري پروگرام 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!");
}
}
```

---

<sup>1</sup> Multiple Interfaces

<sup>2</sup> enum types

<sup>3</sup> 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<sup>1</sup> ولري، چې د کلاسونو او جوړښتونو د منظم کولو لپاره کارېږي ترڅو د ګډوډۍ 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 بنودلو لپاره له WriteLine میتود څخه ګټه اخلي.

### سي پلس پلس او د مایکروسافټ پرمختګ

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

---

<sup>1</sup> Namespaces

<sup>2</sup> Visual Basic.Net

<sup>3</sup> Intermediate Language

<sup>4</sup> framework



مایکروسافت ډاټ نېټ، د وینډوز پروګراملیکنې او د عصري وېب جوړونې<sup>1</sup> لپاره د نورو پخوانیو چارچوکاټونو ځای نیولی دی.

که څه هم په ابتدايي ډول یاده ژبه د مایکروسافت په سافتوېرونو او وینډوز پورې تړلې وه، خو د مونو او «ډاټ جی. این. یو.» پروژې هم سې شارپ وړاندې کوي، او دغه راز د لېنکس او یونیکس چاپیریال لپاره د CLI یا عامه ژبې په پېڅښت<sup>2</sup> او د ډاټ نېټ په زیاترو (نه ټولو) لایبرېریانو کې هم سې 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.

\*\*\*

---

<sup>1</sup> Web-development

<sup>2</sup> Common Language Infrastructure

## (۸)

## سي پلس پلس (C++)

سي پلس پلس په مړې هيل، نيو جرسي کې په AT&T لابراتوار کې د بيارنې سټراؤسټرپ<sup>1</sup> له خوا ډيزاين شوه چې په ۱۹۷۹ز کې يې کار ورباندې پيل کړی وو. تر دې مهاله سي ژبه د سيستيمي پروگرامليکني لپاره د يوې ځواکمنې وسيلې په توگه کارول کېده. که څه هم سټراؤسټرپ (او نور) په دې باور وو چې د سي محدود ډېټا جوړښتونه او د فنکشنو ميکانيزم په لويو سافټوېري پروژو کې چې د استازو<sup>2</sup> ترمنځ يې اړيکې پېچلې وي، د خپلمنځي اړيکو د څرگندوې لپاره د سي ژبه ناکافي وه.

د يوه ساده استازي مثال ته وگورئ: هغه ډېري<sup>3</sup> چې پر هغه نور عددونه هم ور ايښودل کېدای او يا له هغه څخه اخيستل کېدای شي، په سي ژبه کې يوه ډېري بايد د يوه سټرکټ په توگه پلي شي ترڅو د ډېري ډېټا او نښه کونکي سمبال کړي، او يوه ډله فنکشنونه په جلا ډول تعريف شي چې د ډېري ډېټا جوړښت ته لاسرسی وکولای شي، تر څو ډېري ته يو عدد ور اضافه او يا د ډېري له سر څخه يو 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 سافټوېري پروژو کې چې ډېر پروګرام لیکونکي په ګوډ باندې کار کوي، نو دا ډول حالت یو څه ګډوډۍ 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<sup>1</sup> 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، بیاکنه هم پرې کېدای شي او یا زیاتونه او کمونه هم پکې کېدای

---

<sup>1</sup> calls

شي. په دې اساس، د يوه عمومي کلاس له لست څخه ډېر ډوله ځانگړي لستونه مشتق کېدای شي (لکه غبرگ-ترلی لست<sup>1</sup>).

په داسې حال کې چې د استازواله تگلارې گڼو خپل لوري ته ور مايل کړي وو، سټراوسټرپ غوښتل چې په څرگند ډول د ماشين د چلند کنټرولولو ځانگړنه چې سي ژبې لرله، هغه په خپل ځای پاتې شي چې په سيستمواله پروگرام ليکنه کې پکار پريږي. په دې اساس، نوموړي غوښتل چې يوه داسې ژبه جوړه کړي چې نحوي ترکيب يې سي ته ورته وي او د استازوالو ژبو ځانگړنې هم ورسره وي. سټراوسټرپ د انگلستان په کمبرج پوهنتون کې د دوکتورا د تيزس په توگه د سي پلس پلس لومړنۍ نسخه وليکله چې (C له کلاسونو سره<sup>2</sup>) نومېدله. همدا ژبه د ۱۹۸۰ کلونو په شاوخوا کې په ++C بدله شوه.

## د سي پلس پلس ځانگړنې

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

<sup>1</sup> doubly-linked list

<sup>2</sup> C with classes

او ورسره پر دې کتار باندې د عملیو ترسره کولو لپاره فنکشنونه هم تعریفوي. عموماً، 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 کېږو:

```
#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 گوري:

```

#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 پروگرام له کوډ څخه په جلا ډول ترسره کېږي چې د ډېرې له استازو څخه گټه اخلي. په دې اساس، پروگرام لیکونکی کولای شي چې د ډېرې کلاس بیا راوگوري (بنایي د یوه پرمختللي الگوریتم 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 پروگرامونه چې د ډېری له استازو څخه گټه 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 پروگرام لیکونکی کولای شي چې دا ډول کتنه ودروي که چېرته په رښتیا غواړي او یا اړتیا ورته لري). د خاصو عامل-فنکشنونو<sup>1</sup> 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 متنګرښې سره یو ځای کړي. خو که همدا عامل له عددي ډېټا سره وکارول شي، نو بیا به د جمعي معنا ورکوي.

یو انتزاعي<sup>2</sup> استازی (هغه چې اطلاق ونه لري) د مجازي<sup>3</sup> فنکشنونو لپاره د اساس په توګه کارېدلای شي. دغه فنکشنونه په هر مشتق شوي استازي کې بیاځلې تعریف کېدای شي خو هر کله چې د هغه نوعیت یو استازی را وړاندې شي، نو تالیفونکی په خپلواک ډول له پلرني کلاس څخه لاندې لور ته په اشتقايي کلاسونو کې لټون پیل کړي.

---

<sup>1</sup> operator function<sup>2</sup> abstract<sup>3</sup> Virtual

د سي پلس پلس راوسته بڼې د template په نوم يو تړلی مفهوم لري. ټمپلېټ هغه انتزاعي خصوصیت ده چې د هغو ډېټا ډولونو لپاره د کلاس تعريفونو د جوړولو لپاره کارېدای شي چې دې کلاس ته ور تېرېږي. په دې اساس، يوه لیست ټمپلېټ ته يو وکتور او دوه-بعدي کتار ور تېرېدای شي او دا ټمپلېټ به د هر ډېټا ډول لپاره د لیست کلاس تعريفونه جوړ کړي. ټمپلېټونه عموماً هغه مهال کارېږي چې کله د ډېټا ډولونو ترمنځ د هایدراکي توارث هېڅ ډول اړیکه شتون ونه لري (په دې صورت کې مجازي پلرنی کلاس يوه غوره تگلاره ده).

سي پلس پلس د معیاري لایبرېريانو لپاره استازواله بدیلونه هم وړاندې کوي. د مثال په ډول، راکړیز/ورکړیز يو اصلي ماډل<sup>1</sup> 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 تېروتنې د سمالښت پرمختللی میکانیزم هم شته.

### د سي پلس پلس پراختیا

د ۱۹۸۰ او ۱۹۹۰ کلونو په شاوخوا کې، سي پلس پلس له سیسټمي پروګراملیکنې څخه نیولې بیا تر کاروباري کارپالونو او ګېمونو پورې د مختلف ډوله سافتوېرونو لپاره يوه نوميالی 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

<sup>1</sup> stream model



سره هم مهاله وه. څنگه چې دې تالیفونکو د دودیزې سي ژبې کوډ هم سمبالولای شوی، پروگرام لیکونکو کولای شوی چې شته کوډ واخلي او د استازواله تخنیکونو له زده کولو وروسته له هغو ځنې کټه 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 سرورونو او پرانیستونکو باندې ځغېدل. د سافټوېر جوړونې د یوې بدیل تګلارې 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/](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 کاروبارواله ژبه<sup>1</sup> په ۱۹۵۹ز کې د دفاع وزارت په نوبت تر دې انگېزې لاندې جوړه شوه چې د کاروباري کارپالونو 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 مېنفرېم کمپیوټرونو لپاره اساسي کاروبارواله ژبه فلومېټیک<sup>2</sup> وه، نوموړې ژبه د ګرېس هوپر د ډلې له خوا په رېمینګټن رېنډ یونیواک کې جوړه او د همدې کمپنۍ تر کمپیوټرونو پورې محدوده وه. د کوبال لومړني تالیفونکي په ۱۹۶۸ کې منځته راغلل او د امریکې د ملي معیارونو ادارې 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 اوډونګرانو

<sup>1</sup> Common Business-Oriented Language

<sup>2</sup> Flow-Matic

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

### د پروگرام جوړښت

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

د پېژندنې برخه<sup>1</sup> یوازې گرامر مشخصوي او د پروگرام 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 چاپیریال برخه<sup>2</sup> د چاپیریال (سختغالي) په تړاو هغه خصوصیات په ځان کې رانغاړي، د کومو لپاره چې پروگرام تالیف کېږي. په ځینو حالتونو کې (د مثال په

<sup>1</sup> identification division

<sup>2</sup> 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 پروگرام د پیکر-بندي<sup>1</sup> 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 پروگرامونه 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 چاپیریال برخه د راکړیز/ورکړیز برخه هم باید ولري چې هغه وسایل او فایلونه مشخص کړي چې د پروگرام 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 پروگرام له خوا به پروسېس کېږي چې تر یو څه بریده کولای شو چې دا برخه د

---

<sup>1</sup> Configuration

پاسکال، سي او بېسيک په څېر ژبو کې د متحولينو له تعريفولو سره پرتله کړو. دا چې کوبال د فایلونو د ریکارډونو په پروسېس او د راپورونو په فارمت 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 تعريفول يو څه اسانه کوي چې عموماً په کاروباري کارپالونو کې استعمالېږي. د مثال په ډول، دا اسانه کار دی چې د درجه نمبرونو په کارولو هغه ریکارډونه تعريف 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 9999999.
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 يا انځور عبارت د ډېټا ډول (د عددونو لپاره تر نهو اعداد او اعشاري ټکي؛ او د متن لپاره X) او اوږدوالی مشخصوي. د ورکړيز ریکارډونو د مشخصولو ترڅنګ، د ډېټا برخه ډېری وختونه هغه توکونه هم په ځان کې رانغاړي چې د هغه راکړيز د کرښو بڼه ټاکي چې د پایلې په توګه پرنټ کېږي.

د پروسي<sup>1</sup> برخه هغه عبارتونه چمتو کوي چې پر دېټا اصلي عمليې ترسره کوي. پروسه کولای شو چې په فرعي پروگرامونو ترتيب کړو (چې تر يو څه بريده د نورو ژبو له پروسو او فنکشنونو سره ورته والی لري). د پروسي د عبارتونو بېلگه 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
```

رياضيکي معادلات د compute عبارت په کارولو محاسبه کېدای شي:

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

په کوبال ژبه کې ځانگيز if<sup>2</sup> عبارتونه هم شتون لري، او څرخونه د Perform عبارت په وسيله ترسره کېږي. د مثال په ډول:

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

(د بېسيک د نورو نسخو په څېر، فرعي پروگرامونو ته شمېره هم ورکول کېږي.)

## اغېز او لرليد

له ۱۹۶۰ څخه تر ۱۹۸۰ کلونو پورې، کوبال په مېنفرېم او منځنيو کمپيوټرونو کې د کاروباري پوستغالو د جوړولو لپاره يوه اساسي ژبه وگرځېده او تر نن ورځې

<sup>1</sup> procedure

<sup>2</sup> 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 ورته شوه.

د کوبال ځينو اوسنيو نسخو د جوړښتواله پروګرامليکني وروستي پرمختګونه هم په کې ځان کې راټول کړي (لکه modularization) او حتی استازواله ډيزاين. کوبال د مايکروسافټ ډاټ نېټ او ورسره د XML ډېټا د پروسېس کولو په ګډون د جوړونې د اوسنيو چارچوکاټونو د خپلولو لپاره هم د پام وړ تغيرات له ځانه ښودلي. خو بيا هم، د کوبال استعمال ورځ تر بله د ځورتيا پر لور درومي، په دې چې پروګرام ليکونکو ورځ په ورځ خپل مخ د سي پلس پلس په څېر ژبو، متنواله ژبو، او د ډېټابېس جوړونې سيستمونو 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 برتراند مییر او د هغه د کمپنی آیفل سافتویر له خوا په ۱۹۸۰ کلونو کې جوړه شوه. نوموړې ژبه د گوستاف 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 پروگراملیکنې د هغو ټولو مفاهیمو (او په ځینو برخو کې یې لا سرلارې ده) ملاتړ کوي چې د نن ورځ په نوموتو او ډېر کارېدونکو ژبو کې تر سترگو کېږي. د نحوي ترکیب له مخې، آیفل په ساده والي او بیا کارېدونکو تعریفونو چې په پروگرام پوهېدل اسانه کوي، باندې ټینګار کوي او هڅه کوي چې له ټیټې کچې او مبهم کوډ لکه د تالیفونو کې آپټیمایزېشن څخه مخنیوی وکړي.

## د پروگرام جوړښت

په آیفل ژبه کې لیکل شوی پروگرام 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<sup>1</sup> کې، اړین-ویوکی په پرېر او د گټه اخیستونکي له خوا تعریف شوي استازي په کاره لیک لیکل شوي. کله چې گټه اخیستونکی متن درج کړي، نو یاد فارمت 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
```

---

<sup>1</sup> Listing language

د آیفل تالیفونوکی یو کوډ، خپلې منځنۍ ژبې یعنې بایت کوډ ته تالیف کوي، چې په وروستي پړاو کې سي ژبې ته تالیف کېږي، او په دې ځای کې د سي له آپټیمایز شوو تالیفونوکو له وړاندې څخه د شتوالي کټه پورته کوي.

د آیفل یوه خاصه ځانگړنه 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 بڼه ډول پلي کېږي.) د مثال په ډول، د Counter کلاس سره یو ثابت هم تعریفولای شو لکه  $total \geq 0$ . 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 پروگرام لارښودلیکنې<sup>1</sup> ته هم وده ورکولای شي.

## تطبيق او استعمال

د آیفل ملاتړ کوونکي وايي 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 پروگرام بېلا بېلې برخې بیاځلې وکتل او وکارول شي. د آیفل ژبې اوسنی تطبيق په ټولو دريځونو کې موجود دی او د سي، سي پلس لس او نورو ژبو لپاره هم

<sup>1</sup> documentation

اړیکځایونه لري. دا ځانگړنه آیفل ژبې ته اجازه ورکوي چې یو داسې چارچوکاټ ډیزاین او جوړ کړي چې د سافټوېر بېلابېلې شته برخې په نورو ژبو کې وکارول شي. همدا رنگه، د آیفل ثابت استازواله ډیزاین هم دا ژبه د سافټوېري پروژو د ډیزاین او لارښودلیکنې لپاره گټوره 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 ځانگړنو پر آیفل غوره والی ور پرځه کړی: د ارزانه او وړیا تالیفونکو شتون او د هغو زرگونو پروگرام لیکونکو شتون چې له وړاندې څخه یې لا سي ژبه کاروله. آیفل د سافټوېر د ډیزاین د تدریس لپاره د یوې نوې ژبې او د آیفل سټوډیو د پروگرامي چاپیریال په مرسته د محدود شمېر سوداگريزو کارپالونو جوړولو لپاره وکارول شوه.

آیفل د استازواله سافټوېر ډیزاین د جوړولو په ډگر کې د پراخې برخې اخیستنې په سبب پرېمانه شهرت لري او په دې وروستیو کې د کمپیوټري ماشینري د ټولنې له خوا د سافټوېر پر کیفیت باندې د اغېز جایزه چې د ۲۰۰۶ د سافټوېر سیسټم 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=](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 ډېری کې ځای پر ځای کېږي. د مثال په ډول، د ریاضي یوه معادله چې په ډېرو ژبو کې  $2 + 3$  لیکل کېږي، په فورت کې  $2\ 3\ +$  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 ډېری 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<sup>1</sup>، وروستاریزه ښودنه<sup>2</sup> بلل کېږي او د ساینټیفیک ماشین-حساب ډېری ګټه اخیستونکي ورسره بلد دي.

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

1

<sup>2</sup> Postfix Notation

فاصله ونه لري) را پورته کوي او بيا دکشنري لولي تر خو وگوري چې آيا نوموړې کلمه له کومې تعريف شوې کلمې يا متحول سره سر خوري او که نه. که چېرته يوه کلمه وموندل شي، نو په کوډ کې شته کلمه اجرا کېږي. که چېرته کلمه ونه موندل شي، نو ژباړونکي هغه توکن يوه عددي ثابت ته ژباړي، ډېری ته يې ور پورته کوي، او بيا راتلونکي لغت ته ور درومي.

د فورت يوه کلیدي ځانگړنه د دې ژبې د پراختياموندنې خصوصیت 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 گڼ 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 پروگرامليکني اصول په ځان کې را ټول کړي دي چې کنسټرکټونه او ميتودونه لري او کلمات يې د استازو په توگه کار ورکوي. د فورت يو منظم پروگرام د اوليه عملياتو او عالي ژبو له کلماتو څخه جوړ وي، چې پخپله پروگرام هم د عالي ژبو يوه کلمه ده.)

فورت تل د هغو لېواله پروگرام ليکونکو پام ځان ته اړولی وو چې په ماشين کې د ډېټا له جريان سره له نږدې اړیکې او په څرگند ډول پروگرام سره د کار کولو وړتيا ته په بڼه سترگه گوري. نوموړې ژبه په بشپړ ډول تعاملې ژبه ده، ځکه هره کلمه په کيبورډ ليکل کېدای شي، تر څو اجرا شي او پایله يې په صفحه ښکاره کړي.



فورت د کوچنیو کمپیوټرونو په ورځو کې هم یوه رابنکونکې ژبه وه ځکه چې فورت کوم پیچلي ژباړونکي یا تالیفونکي ته اړتیا نه لرله، او دا په دې معنا وه چې فورت سیستمونه په هغو سیستمونو په ارامتیا ځغلبدای شي چې د ناڅاپي-لاسرسې زېرمې ظرفیت به یې له ۱۶ کیلو بایت څخه ۶۴ کیلو بایتو وو.

فورت هېڅکله د پروگرام لیکونکو لپاره یوه عمومي ژبه نه شوه. له نورو ژبو د یادې ژبې بېلوالی او غیرعادي ذهني چوکاټ د هغو کسانو شمېر محدوداوه چې د دې ژبې زده کړې ته یې زړه بڼه کاوه. په داسې حال کې چې د فورت پروگرامونه په څرګند ډول ترتیب کېدای شي، خو هغه پروگرامونه چې په ناوړه ډول لیکل شوي وي، د هغو لوستل شاوخوا ناممکنه وي. که څه هم، ځینې وختونه فورت په ځینو برخو کې لوړ دريځ خپل کړی (لکه، پسمتنواله<sup>1</sup> ژبه فورت ته ورته ده) او نوموړې ژبه تر اوسه هم د هارډوېر د کنټرولونکو 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.

---

<sup>1</sup> 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 څرگندې 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 لیکونکي نور په دې اړه اندېښنه نه کوي تر څو په مفصل ډول د کمپیوټر زېرمه ورته مشخصه کړي او له کمپیوټر څخه باینري ډېټا بېرته ترلاسه کړي؛ دغه راز د پروګرام د جوړښت او د سمو الګوریتمونو له تطبیق څخه هم بېغمه وي.

فورټران (FORMula TRANslator) لومړنی تر ټولو زیاته کارېدونکې عالي ژبه وه. فورټروان د آی بی 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 بڼه کې بیان کړي. په همدې وخت کې باید یو تالیفونکی 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 مېنفرېم کمپیوټرونو لپاره د تالیفونکو په توګه منځته راغله. بله پرمختللي نسخه يې (چې ټېروټنې يې هم نیولې)، ډېر ژر ورپسې راووتله. فورټروان-۴ (۱۹۶۳) د ډېټا ډولونو شمېر زیات کړ، «عامه» ډېټا زېرمټون ور زیات شوی، او د DATA عبارت پکې رامنځته شو، چې متحولینو ته د لیټرل<sup>۱</sup> عددي وېلیوګانو ور پورته کول يې اسانه کړل. د فورټروان دې پرمختللي نسخې ته د زیاتره هغو ساینسپوهانو او انجینرانو له لوري هرکلی وشو، چې د هغو محاسبو لپاره يې د کوډ سټري لایبرېرېانې جوړې کړې چې عموماً 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 تېرېدای، هماغلته پروسېس او بېرته واپس کېدای وشي. هغه ناشرطي ځانګیز<sup>۲</sup> عبارت (goto) چې په فورټران کې عام وو، له پامه وغورځېده. د فورټران ژبې یوه نوې نسخه، فورټران-۷۷ (یا F77)، ډېری 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 لایبرېرېانې هم له عصر سره برابرې شوې. فورټران ۲۰۰۳ د پروګراملیکنې د اوسنیو جوړښتونو

<sup>1</sup> Literal<sup>2</sup> Unconditional branching statement

لپاره د ملاتړ او په سي ژبه کې د ليکل شوو پروگرامونو سره د هوسا تعامل کولو د وړتيا په گډون، يو شمېر نوي فيچرونه هم لري. فورټران ۲۰۰۸ تر يو څه بريده د لا پرمختللي نسخې لپاره لا هم يو تجربې نوم دی.

## د پروگرام بېلگه

لاندې ساده بېلگه د دوديز فورټران پروگرام ځينې ځانگړنې 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 پروگرام يو داسې کتار جوړوي چې تر لسو پورې عددونه په ځان کې ذخيره کوي. لومړنی عدد چې دی يې لولي د هغو توکونو شمېر دی چې ور زياتېږي. 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 يو ځل تکراروي. دا عبارتونه له کتار څخه راتلونکي عددونه لولي او بيا يې په SUM کې

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

د خپلې همعصره، کوبال ژبې، په څېر فورټران هم د زیاتره اوسنیو پروگرام لیکونکو له لوري د یوې ګډې وډې او مهالټېره<sup>1</sup> ژبې په توګه پېژندل کېږي (او د دې لامل د مثال په ډول، د ریفرنس کرښه-ګڼو کارول دي). خو بیا هم، فورټران د ازمايل شوو او باوري کوډونو له یوې پراخه زېرمې او د ریاضي له ځواکمنو لایبرېریانو څخه برخمنه ژبه ده. (د مثال په ډول، د فورټروان یو پروگرام کولای شي چې د لایبرېري روټینونو ته بلنه ورکړي تر څو په چټکه توګه د جمع حاصل او یا د مټریکس یا کوم کتار د طرفینو د ضرب حاصل ترلاسه کړي). دا ځانګړنې اوس هم دا باور لېږدولای شي چې فورټران لا هم د هغو ګټه اخیستونکو لپاره هغه انګېزه او ګټورتیا په ځان کې لري چې ډېره اندېښنه یې د پروګراملیکني د بنایسته سبک پر ځای د چټکو او دقیقو پایلو 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*.

---

<sup>1</sup> 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 يوه کمپیوټري ژبه ده. که څه هم جاوا، يوه عام-هدفه پروگرامواله ژبه ده، خو تر ډېره د هغو کاريالونو د جوړولو لپاره کارېږي چې په انټرنېټ لکه وېب سرورونو باندې ځغلي. د جاوا د پروگرام يو ځانگړی ډول applet بلل کېږي چې له وېبپاڼو<sup>1</sup> سره وصل کېدای او د گټه اخیستونکي په وېبرائيسټونکي<sup>2</sup> باندې ځغلول کېدای شي.

د يوې استازواله ژبې په توگه، جاوا له هغو کلاسونو څخه گټه اخلي چې د اړتيا وړ عادي فنکشنونه چمتو کوي، او له هغې ډلې څخه د اړیکځای د استازو<sup>3</sup> جوړول لکه د عامل سيستم کړکۍ<sup>4</sup> او بڼونه يادولی شو. د ډول ډول کلاسونو يوه ټولگه (د کلاس چارچوکاټونه) هم پکې شتون لري، لکه AWT يا وينډوزواله انتزاعي اېزارېکس<sup>5</sup>.

---

<sup>1</sup> Webpages<sup>2</sup> Web browsers<sup>3</sup> user interface object<sup>4</sup> windows<sup>5</sup> Abstract Windowing Toolkit

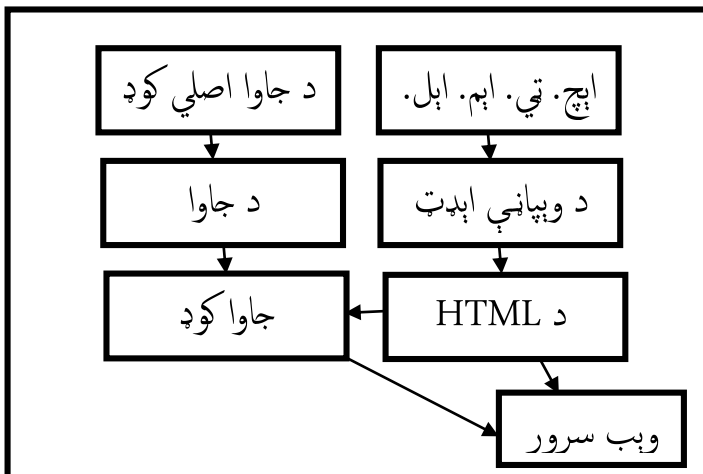


## د پروگرام جوړښت

د جاوا پروگرام د کلاسونو په راوړلو او بیا د هغو په وسیله د پروگرام 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 کرکې کې د متن ځای په ځای کول. په دې ځای کې د جاوا اپلټ یوه ساده پروگرام 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 پروگرام ته معیاري کلاسونه را کوزوي. اپلټ کلاس هغه بنسټ دی چې پر هغه اپلټ پروگرامونه بنا وي. ویندوزواله انتزاعي ابزاریکس د هغو کلاسونو ټولګه ده چې ګرافیکي اړیکځای (GUI) چمتو کوي.



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

ورپسې پروگرام 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، پروگرام د ګرافیکي استازو، تعریف کړی drawstring میتود کاروي تر څو یوه متنګرښه ورباندې رسم کړي.

د دې پروگرام د جوړولو لپاره، پروگرام لیکونکی نوموړی کوډ په جاوا کمپایلر کې تالیف کوي. ورپسې پروگرام لیکونکی د html پاڼه جوړوي چې یو ټېګ 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 مایکروسولیسیم کمپنی کې د یوې کورنۍ پروژې په توګه پیل شوه څو یوه داسې ژبه

ډیزاین کړای شي چې د «ځیرکو» مصرفي وسایلو د پروگرام کولو لپاره کار وکړي، لکه ګرېډونکی ټلويزون. کله چې دا پروژه پرېښودل شوه، نو ګوسلینګ، بیل جوی، 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 ترڅو داسې پروګرامونه جوړ کړي چې هغه مهال وځغلي چې کله د ګټه اخیستونکي له لوري هغې وېبپاڼې ته لاسرسی وشي. په وېبپاڼو باندې د ګټه اخیستونکو د کنټرول پلي کولو له لارې، اوډونګرانو د وېب ګټه اخیستونکو ته هغه وړتیا په لاس کې ورکولای شوی چې په آنلاین ډول کټ مټ هماعسې د ارتباط نیولو وړتیا ولري لکه څنګه یې چې د مېکنتاش او یا د شخصي کمپیوټرونو د وینډوز پر پرده له استازو (objects) سره ټینګوي.

## ګټې

جاوا، د وېب ډوېلپرانو 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 استازواله پروګراملیکني 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 پرمختللي او غښتلي ژبه ده. د مثال په ډول، جاوا د نښه کوونکو اړتیا له منځه وړي او له کلاسونو څخه د پروګرام د جوړښت لپاره د ثابتو بنسټي تیرو په توګه ګټه پورته کوي. سافټوېر جوړوونکي ادارې لکه

مایکروسافت (تر پرونه پورې) او آی بی اېم شرکت د جاوا د پرمختګ لپاره له سن سره لاس یو کړی دی.

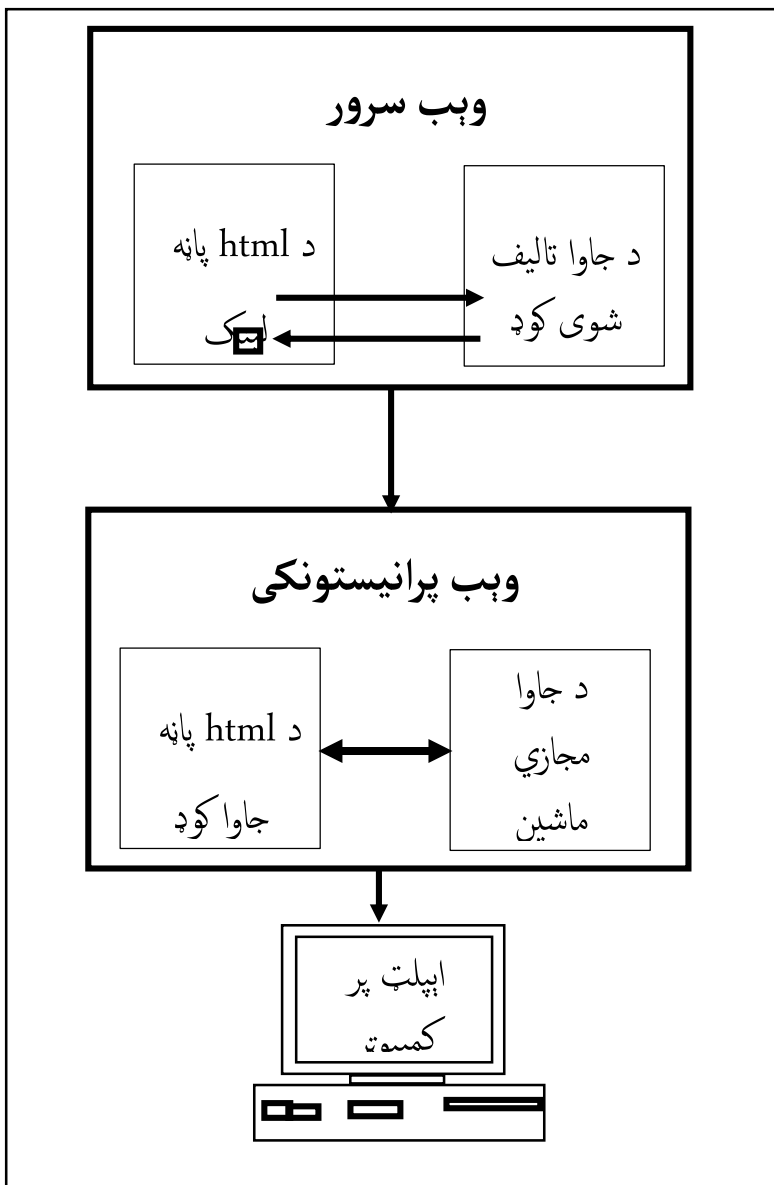
د جاوا یوه بله ستره ځانګړنه دا ده چې جاوا خپلواک-دریځه ژبه ده. پخپله ژبه د ګڼو عامل سیستمونو له دریځونو څخه بېله او پردی ده. د هر دریځ لپاره، د جاوا مجازي ماشین (Java Virtual Machine) جوړېږي چې د جاوا د تالیفونکو لپاره تولید شوی کوډ کمپایل یا ژباړي خو په یاد دریځ وځغېدای شي.

د خوندیتوب لپاره، د جاوا اېپلټونه د یوه شګلن-ډېلي یا sandbox او یا محدود چاپېریال دننه ځغلي تر څو ګټه اخیستونکی یې د جاوا له زیانمنو پروګرامونو څخه خوندي وي. (د مثال په ډول، پروګرامونو ته دا اجازه نه ورکول کېږي چې د ګټه اخیستونکي زېرمتون ته لاسرسی ولري او یا د ګټه اخیستونکي ماشین له بل ویبسایت سره وصل کاندې). همدا رنگه، وېپرانېستونکي هم د جاوا اېپلټونو د مخنیوي لپاره غیرفعاله کېدای شي.

## بالغه ټکنالوژي

د سن کمپنی جاوا په دوو ذایقو کې راځي: د جاوا معیاري اېډیشن (J2SE) د مایکروسافت د وینډوز، د سن د سولاریس او لینکس لپاره؛ او د جاوا سوداګریز اېډیشن (JSEE) چې له هغو ځانګړنو څخه برخمن ده چې په سترو او پېچلو چاپېریالونو کې ورته اړتیا وي. مایکروسافت 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 دريځونو لپاره د جاوا تطبیقي نسخې جوړې کړي، خو په دې شرط چې دوی د سن له ټاکلې قانوني پروسې څخه وانه وږي.)



د دې لپاره چې جاوا ایپلټ وځغلي ، گټه اخیستونکي ، وېب پرائیستونکي ته لینک شوې پاڼه را پورته کوي. په دې پسې ایپلټ ښايي په خپلواک ډول وځغلي ، او یا کېدای شي چې له یوه ځانگړي لینک او یا کنټرول لکه بتن پورې وصل اوسي. یو ځل چې فعاله شي ، نو ایپلټ وېب پرائیستونکي ته ډاونلوډ کېږي ، چې وروسته د خپل جاوا مجازي ماشین په نوم د یو ماژول (module) په کارولو خپل جاوا-کوډ وځغوي. د هر کمپیوټري سیستم لپاره د جاوا بیل مجازي ماشین (JVM) شتون لري.

جاوا، د سافټوېر بیا کارېدونکو اجزاوو ته ځانگړې پاملرنه کړې. جاوا-لوییا یا JavaBeans یو شمېر تړلي-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 پروگرامي اړیکځایونه هم شتون لري. په داسې حال کې چې د کلاينټ لوری، جاوا ایپلټونه پر وېب پرائیستونکي وځغوي، خو د جاوا سرور پانې (JSP) همدا کوډ د HTML په یوه پاڼه کې ور ځایوي او نوموړی کوډ د سرور لوري کاريال «سرولټ»<sup>1</sup> په مرسته تالیف کېږي. ورپسې XML پروسېس او د XML لپاره ډېټابېس ته لاسرسی د جاوا د API له خوا چمتو کېږي.

---

<sup>1</sup> servlet

جاوا خپله هغه ژمنه تر ډېره پورې پوره کړې چې د اصلي تگلوري<sup>1</sup> استاوازله پروگرامليکنه په لوی شمېر دريځونو باندې د پلي کېدو وړ وگرځوي. ياده ژبه نن ورځ د سي يا سي پلس پلس پر ځای د لومړنۍ ژبې په توګه تدريس 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 برېښي چې همدا يوازینی مشهوره او غوره ژبه ده.

په ۲۰۰۶ کې سن مایکروسيسټم شرکت اعلان وکړ چې د جاوا سرچينه ګوډ به په وړيا ډول د لاسرسي وړ وگرځوي. تر يو څه بريده کېدای شي دا د دې لپاره يوه هڅه اوسي چې جاوا ژبه د پروګرام ليکونکو ترمنځ پاتې شي، چې ځينو پروګرام ليکونکو اوس خپل پام له جاوا څخه د مایکروسافټ د سي پلس پلس نسخې ته اړولی. د جاوا لپاره تر ټولو ستره ننگونه د وېب پروګرامليکنې په ډګر کې ده، چېرته چې نوموړې ژبه ورځ تر بله له ځيرکو او تېزو ژبو سره مخ کېږي (د مثال په ډول وګورئ، Ruby). دغه راز متنواله ژبې هم د جاوا لپاره يوه لويه ننگونه ده چې ښايي د زده کړې لپاره اسانه او د زياتره کاريالونو لپاره يې استعمالول چټک وي.

---

<sup>1</sup> Streamline

## اضافی لوست:

- 1- Arnold, Ken, James Gosling, 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*. 2<sup>nd</sup> 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](http://news.com.com/2008-7345_3-6022062.html). Accessed April 10, 2007.
- 4- Krill, Paul. "Java Facing Pressure from Dynamic Languages." *InfoWorld*, March 25, 2006. Available online. URL: [http://www.infoworld.com/article/06/03/25/76803\\_HNjavapressure\\_1.html](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.

\*\*\*



(۱۴)

## جاواسکرپټ (JavaScript)

جاواسکرپټ له هغو نوموتو ژبو څخه ده چې وېبپاڼې د دې وړ گرځوي چې له گټه اخیستونکو سره په چټک او اغېزمن ډول ارتباط ټینګ کړي. نوموړې ژبه د لومړي ځل لپاره د ۱۹۹۰ کلونو په منځ کې د نېټسکېپ-۱۲ په پرائیستونکي کې د لایوسکرپټ<sup>۲</sup> تر نوم لاندې راڅرګنده شوه. له تخنیکي پلوه، جاواسکرپټ د سن مایکروسیسټم 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 مبتدلو وېب چوکاټونو ته یې د ځینو مشخصاتو لکه درې بعدي بټونونو او پاپ-اپ کړکیو، د ور زیاتولو لپاره اسانه لاره چمتو کوله. خو څنګه چې پخوانیو نسلونو د گېشمېر فوټونو له کارولو سره لېوالتیا لرله، نو په دې اساس د جاواسکرپټ د لومړنیو پروګرام لیکونکو وېبپاڼې به له سرګردانونکو او بې گټو ګیوډیو څخه ډکې وې.

---

<sup>1</sup> Netscape-2

<sup>2</sup> Livescript

د دې ترڅنګ چې ځینې وختونه به یې ګټه اخیستونکي هم ځورول، د لومړیو وختونو جاواسکرپټ له هغو توپيرونو څخه هم کړېده چې یاده ژبه به په لویو پرائیستونکو کې په څه ډول پلي کېږي. په پایله کې به د نېټسکېپ د ګټه اخیستونکو کار کله ناکله د هغه جاواسکرپټ له امله ټکنی کېده چې د مایکروسافت په انټرنېټي پرائیستونکي کې به لیکل شوي وو او یا به په جاواسکرپټ کې د مایکروسافت لپاره لیکل شوي وو. په پای کې د پرائیستونکو نیمګړتیاوو کله ناکله جاواسکرپټ ته اجازه ورکوله چې په وایرس ککړ شوو د پرائیستونکو مرستیالانو<sup>1</sup> د نصبولو له لارې د ویبپاڼو پر امنیت د له منځه وړلو لپاره وکارول شي. په پایله کې، د امنیت ډېری ماهرینو دا وړاندیزونه پیل کړل چې ګټه اخیستونکي دې په خپلو پرائیستونکو کې د جاواسکرپټ اجرا، غیرفعاله کړي.

## د جاواسکرپټ کارول

د جاواسکرپټ 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، جاواسکرپټ یوه ماشین<sup>2</sup> ته بلنه ورکوي، چې د هر کوربه چاپېریال (عموما

<sup>1</sup> browser helpers

<sup>2</sup> engine

يو وېب پرائيسټونکي) لپاره ليکل شوی وي. ياد ماشين هغه ځانگړنې تطبیق کوي چې د دې لپاره ډيزاين شوې وي خو دا کنټرول کړي چې يوه وېبپاڼه له گټه اخيستونکي سره په څه ډول تعامل کوي، لکه د کړکيو او کنټرولونو لکه مېنوگانو، بټنونو او يا ايزارېټو<sup>1</sup> څرگندول. جاواسکرپټ تر دې دمخه چې سرور ته تسليم شي، په پرائيسټونکي کې د يوه وېب چوکاټ<sup>2</sup> د قانون بندي لپاره هم کارېږي. عموماً، د پرائيسټونکي لوري<sup>3</sup> جاواسکرپټ پروسېس پر وېب سرورونو باندې بوج کموي او په ورته وخت کې پاڼه چټک غبرگون ته چمتو کوي، د بېلگې په ډول، کله چې گټه اخيستونکي د پاڼې په يوه برخه د ماؤس نښه کوونکي ودروي، نو ورسره گرافیک هم بدل شي.

د جاواسکرپټ او html د پاڼو ترمنځ اساسي اړيکځای، ډاکومنټ آبجکټ ماډل دی. د نړيوال جال ائتلاف<sup>4</sup> د DOM معياري فنکشنونه تعريف کړي او ډېری پرائيسټونکي په ثابت ډول د دې معيارونو لومړني او دوهمي پوړونو څخه گټه اخلي. که څه هم داسې گټه اخيستونکي شته چې د جاواسکرپټ معيارونه نه شي ځغلولی، لکه په سترگو رانده گټه اخيستونکي، هغه گټه اخيستونکي چې د موبایلونو پرائيسټونکي کاروي (لکه ډيجيټل مرستيال (PDA) يا ځيرک ټليفونونه)، او يا هغه گټه اخيستونکي چې د امنيتي ستونزو له امله يې جاواسکرپټ غیرفعال

---

<sup>1</sup> Toolbars

<sup>2</sup> Web frame

<sup>3</sup> browser side

<sup>4</sup> World Wide Web Consortium

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

په دودیز ډول، د جاوااسکریپټ کوډ په مستقیم ډول د تېګونو 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 ګټه اخیستونکي څخه د نوم غوښتنه کوي، چې د Name په متحول کې زېرمه کېږي او وروسته بیا په خبرتيايي-چوکاټ<sup>1</sup> 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 ګټه اخیستونکي د کړنو په غبرګون کې لاسي پانې هم جوړولای شي. جاواسکرپټ له وېبپرانيسټونکو پرته په نورو کاريالونو کې هم اغېز کېدای شي: د مثال په ډول، اډوب اکروباټ او لوستونکی یې او حتی د عامل-سیسټیم متنواله ژبه کې د مایکروسافټ له Jscript او Jscript.NET سره یو ځای کېدای شي. که څه هم وروستي کلونه داسې برېښي

---

<sup>1</sup> alert-box

چې خپل لوری یې نورو ژبو لکه PHP ته اړولی وي، خو جاوااسکرپټ 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<sup>۱</sup> سره لېوالتیا وده وکړه، نو څېړونکي د هغه مهال د کمپیوټر له ټیټو ژبو (low-level) 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، پروسېس کړي.

په ۱۹۵۶ کال کې پر مصنوعي ځیرکتیا د ډارټماؤټ د اوړي د څېړنیزې پروژې پر مهال، چې د دې ډګر لومړني مخکښان یې را یو ځای کړل، جان میکارټي د یوه پیل 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، میکارټي د یوې ژبې پلي کول پیل کړل چې LISP یا لیست پروسېسر<sup>۲</sup> نومېدله. څنګه چې یې له نومه بنکاري، نوموړې ژبه د ډېټا د زېرمه کولو لپاره له لیستونو څخه ګټه اخلي او د لیست په عناصرو باندې

<sup>۱</sup> Artificial Intelligence<sup>۲</sup> List Processor

د عملیو د ترسره کولو لپاره گڼ 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<sup>1</sup> په توگه زېرمه کېږي چې د ډېټا وېلیوگانو لپاره هم یو نښه کوونکی لري او په همدې تړلي لیست کې راتلونکي توک لپاره هم نښه کوونکی لري. د لیست سیستم عموماً له ذخیره شوو فنکشنونو څخه جوړ وي لکه کارېج کلېکشن<sup>2</sup>، 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 لیسټ پروگرامونه په لومړي نظر تړلي ښکاري، ځکه نوموړي پروگرامونه گڼ شمېر په یو بل کې اوبدلي قوسونه لري. خو افادې او فنکشنونه په اصل کې د ډېری نورو ژبو په پرتله په ساده ډول رغېدلي وي. پېچلي اوږون ته له اړتیا پرته، د لیسټ ژباړونکی (چې ایول یا eval بلل کېږي ځکه دی راکړیز ارزوي<sup>3</sup>) د ډېټا مسیر ته گوري او لومړی دا پوښتنه کوي چې آیا راتلونکی توک ثابت دی (لکه عدد، یا په لینډیو کې لیکلی سمبول، متنکرښه، په لینډیو کې لیکلی لیست، او

<sup>1</sup> node

<sup>2</sup> garbage collection

<sup>3</sup> evaluate



يا اړين-ويوکی). که چېرته همداسې وي، نو وېليو يې بېرته واپس کېږي. که نه، ژباړونکی گوري چې آيا نوموړی توک له وړاندې څخه تعريف شوی متحول دی، او که همداسې وي، نو د هغه وېليو واپس کوي. په پایله کې، ژباړونکی گوري چې آيا کوم لیست شته. که چېرته هو، نو له یاد لیست سره د يوې فنکشنې بلنې<sup>1</sup> په څېر چلند کېږي چې ترڅنګ يې ارگومنټونه راغلي وي. په دې اساس فنکشن ته بلنه ورکول کېږي، ډېټا ورکول کېږي، او پایله يې واپس کېږي.

### د لیسپ د توکو نومثالونه

| د توکو ډول         | مثال        | ارزونه        |
|--------------------|-------------|---------------|
| عدد                | ۲۴          | ۲۴            |
| اعشاري             | ۵,۵         | ۵,۵           |
| نسبت               | ۳/۴         | ۰,۷۵          |
| اساسي لغت          | Defun       | فنکشن تعريفوي |
| ليندۍ لرونکی لیست  | (۵ ۱ ۴ ۱ ۳) | (۶ ۱ ۴ ۱ ۳)   |
| بولین              | تش          | غلط           |
| فنکشن بلنه         | ۴ ۲ +       | ۶             |
| متحول              | A           | وېليو يې      |
| ليندۍ لرونکی متحول | 'a          | A             |

<sup>1</sup> function call

الګول، پاسکال او سي ته ورته ژبې په عبارتونو او پروسو ټينګار کوي. خو بل خوا، لیسپ بیا لومړنۍ فنکشنې<sup>1</sup> ژبه وه. د لیسپ د یوه پروګرام غوښنه برخه هغه فنکشنونه جوړوي چې له خپلو ارګومنتونو سره یوځای ارزول کېږي. په لیسپ ژبه کې ځاني، او یا اولیه فنکشنونه شتون لري. د ریاضي د عادي عملیو سرپرته، د بنسټیز لیست پروسېس لپاره اولیه فنکشنونه لري. د مثال په ډول، لیست فنکشن له خپلو ارګومنتونو 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 پېخ ته تېل 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)))))
```

---

<sup>1</sup> Functional

دلته د if افاده دا گوري چې آیا  $y$  صفر دی او که نه. که نه وي، نو دوهمه افاده د power فنکشن را پاروي<sup>1</sup>، چې همدا ازماينبت يو ځل بيا ترسره کوي. پایله يې دا چې فنکشن بيا بيا خپل ځان ته بلنه ورکوي، لنډمهاله وېليوکانې ذخيره 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 بنيابي د لیسپ تر ټولو خوندوره ځانگړنه دا وي چې نوموړې ژبه د پروگرامونو (فنکشنونو) او ډېټا ترمنځ هېڅ توپير نه کوي. دا چې يوه فنکشنې بلنه او د هغه آرگومنتونه پخپله يو لیست جوړوي، نو يو فنکشن په نورو فنکشنونو باندې د ډېټا په توگه لوستل کېدای شي. په دې ډول د هغو پروگرامونو ليکنه اسانېږي چې په خپلو عمليو کې بدلون راوړي.

## د ژبې تحول

لیسپ ژبې په لنډ وخت کې د مصنوعي ځيرکتيا د څېړونکو پام ځانته را واړاوه، او هغه بڼه چې Lisp 1.5 نومېده، د لويې کچې د کاريالونو ليکلو لپاره يوه گړندی ژبه وگڼل شوه. په داسې حال کې چې مخکښو کمپيوټرپوهانو ډېر کله د الگوريتمونو د بيانولو لپاره الگول او د هغه راتلونکي نسلونه د نړيوالې<sup>2</sup> ژبې په توگه کارول، خو لیسپ د مصنوعي ځيرکتيا د خلکو ترمنځ مشترکه نړيواله ژبه شوه.

<sup>1</sup> Invoke

<sup>2</sup> Universal

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

د لیسپ تر ټولو زیات کارېدونکی ورژن سکیم<sup>1</sup> نومېږي، چې د ۱۹۷۰ کلونو په منځنیو وختونو کې د MIT<sup>2</sup> 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<sup>3</sup> وړتیاوې ولري او په دې ډول ژبې ته عمومیت وېښي. 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 بېلا بېلو نسخو په مختلفو دريځونو باندې د ځغلېدلو ځانګړتیا د معیار ورکونې<sup>4</sup> غورځنګ ته لاره هواره کړه چې په پایله کې یې په

<sup>1</sup> scheme

<sup>2</sup> Massachusetts Institute of Technology – د ماسوچوسېټ د ټکنالوژۍ مؤسسه

<sup>3</sup> data entities

<sup>4</sup> Standardization

۱۹۸۴ کې کامن لیسپ<sup>۱</sup> رامنځته شوه. کامن لیسپ د شته لهجو زیاتره ځانگړنې په ځان کې راتولوي، او د ډېټا له غني ډولونو څخه برخمنه ده، او همدا رنگه د تاکیدي، او ترتیواله پروگراملیکنې تگلارو کارولو ته هم پراخه اجازه ورکوي لکه سي ته ورته ژبو کې. په دې ډول نوموړې ژبه د پروگراملیکنې لپاره گڼ سبکونه وړاندې کوي. لیسپ نن ورځ د سوداگریزو او اشتراکي پوستغالو (shareware) په نسخو کې په پراخه توگه شتون لري.

سیمور پیپرټ لیسپ ته ورته ژبه چې لوگو (Logo) نومېدله جوړه کړه، چې زلمیو زده کوونکو ته د کمپیوټر پوهنې د مفاهیمو د بنوونې 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.

\*\*\*

---

<sup>1</sup> common lisp

(۱۶)

## لوگو (Logo)

لوگو د لیسپ ژبې یو مشتق دی چې د هماغې ژبې د لیست پروسېس او سمبولیکي لاسوهنې<sup>1</sup> 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 کرافیکي وړتیاوې یې د ځوانانو لپاره زړه راښکونکې دي. لوگو ډېر کله د لومړنیو او منځنیو ښوونځیو زده کوونکو ته د لومړنۍ کمپیوټري ژبې په توګه ښودل کېده. د هارولډ ایبلسن په ۱۹۸۲ کې د اپیل لوگو په رسامی کې پام شوو چې «لوگو 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.»

لوگو 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 سېمور پېپرټ زړه ته، د ودې او پرمختګ له ارواپوه جان پیاچت سره له کار کولو وروسته د یوې ښوونیزې کمپیوټري ژبې د جوړولو لېوالتیا ور ولوېده. پېپرټ، د پیاچت پر رغښتواله<sup>2</sup> باندې ټینګار ته ځانګړې پاملرنه کوله. رغښتواله هغه نظریه ده چې خلک په شته قالبونو کې هغه

---

<sup>1</sup> symbolic manipulation

<sup>2</sup> 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 کمپیوټري ژبې لکه فورټران او يا حتی 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، ډېری کمپیوټري ژبې د هغو عبارتونو په کارولو گرافیک پلي کوي چې د کارټيزن محورونو (X,Y) په مرسته د پردې ټکي مشخصوي. د مثال په ډول، يوه مربع کېدای 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 پای کې يو څوک د دې جوگه کوي چې دا عمليه په ذهن کې ترسيم کړي، او دا له درک څخه ډېر لري خبره ده.

پېپرټ په خپله لوگو ژبه کې يو کيشپ راوړي. کيشپ په اصل کې يو رسنيتنی روبات وي چې شاوخوا ته د حرکت کولو لپاره پروگرام کېدای شي؛ نن ورځ په ډېری سيستمونو کې نوموړی کيشپ د يوه کرسر په بڼه وړاندې کېږي. څنگه

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

د کیشپ د کمانډونو په مرسته، یوه مربع په لاندې ډول انځورېدای 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 پروگرام لیکونکی په اسانۍ سره قدم وهل او تاوېدل په ذهن کې مجسم کولای شي تر هغو چې پروگرام لیکونکی /ې بېرته د پیل ټکي ته راگرځي. د بیاجت تیوریانو ته په پام سره، زده کړه زموږ له فزیکي نړۍ او ورځینو فعالیتونو سره اوره په اوره مخته ځي.

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

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

لوگو 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 کېمونو 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 پرماتنه ډول پراخه شوې، او اوس نه یوازې د ریاضي د فنکشنونو یوه بشپړه ټولګه لري، بلکې ډېری ورژنونه یې د وینډوز او مېکنتاش لپاره ځانګړي غبرونه، ګرافیک او ملتي-میډیا هم لري. د ۱۹۸۰ کلونو په منځنیو وختونو کې لوګو له نومیالي لیکو (Lego) لوبتوکي سره یو ځای کړای شوه تر څو لیکو لوګو (LEGO Logo) جوړه شي. دا نومیالی لوبتوکي زده کوونکي د دې جوګه کوي چې ډول ډول روباتونه 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 «رښتیني نړۍ»

د پروگراملیکنې مهارتونه وپنځوي. خو سربېره پر دې هم، لوگو د اروپا، جاپان، او لاتیني امریکا په ځینو برخو کې بڼه وځلېده. لوگو په لنډو وختونو کې د دوو نوو ورژنونو په جوړولو هم پراخه انرژي واخیستله. مایکرو-ورلډز لوگو د مېکنتاش له اړیکهای څخه ګټه واخیستله تر څو د ملتي میډیا لپاره له بشپړو ځانګړنو برخمن کاري چاپیریال وړاندې کړي، او وروسته د وینډوز سیستمونو له لوري هم په کار واچول شوه. د لوگو 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 لوگو د لاسرسي وړ او تعاملی فطرت زلمیو زده کوونکو ته هم د کمپیوټر پوهنې د بنوونې لپاره نوموړې ژبه یو غوره انتخاب ګرځوي.

### 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<sup>1</sup> ژبه ده چې د برازیل په هېواد کې د «ریو ډي. جانېریو» له لوري په پونتیفیکل پوهنتون کې اختراع شوه. (لووا په پرتګالي ژبه کې «سپورمی» 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 (ټاکیډي) پروګراملیکنې او فنکشنې پروګراملیکنې دواړو لپاره کاري بستر وړاندې کوي او ملاتړ یې کوي. ډېری 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<sup>2</sup> په مرسته یې جوړ کړو.

د لووا یو ساده فنکشن په لاندې ډول جوړېږي:

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

---

<sup>1</sup> scripting

<sup>2</sup> extension methods

end

د سیمیکالڼ نشتون ته مو پام وي!

د ځاني ډېټا ډولونو سرپرېه، لووا د پېچلو او د گټه اخیستونکي له خوا تعریف شوو ډېټا ډولونو د جوړولو لپاره له جدولونو څخه هم گټه اخلي. په جدولونو کې د وېلیوگانو 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 متنکړینه وي. د مثال په ډول، لاندې جدول درې وېلیوگانې لري چې د متنکړینو 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 ډېټا په اضافه کولو، د استازواله ژبو په څېر په دې ژبه کې هم کلاسونه جوړولای شو.

## عملي ډگر او کارونه

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

په ژبه کې د غلمهاله<sup>1</sup> پېکجونو له امله، لووا زیاتره وختونه کار یالی پوستغالو ته ور اضافه کېږي تر څو د پوستغالي د پراختیا او سمون لپاره اړیکځای چمتو کړي. دا خبره په ځانګړي ډول کېمونو لکه د جنګ نړۍ<sup>2</sup>، 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.

\*\*\*

---

<sup>1</sup> runtime

<sup>2</sup> The world of Warcraft

(۱۸)

## پاسکال (Pascal)

د ۱۹۶۰ کلونو په لومړيو وختونو کې د کمپیوټر پوهانو لويه اندېښنه د داسې لارو موندل وو چې خپل پروگرامونه په غوره ډول منظم او ورغوي. په اصل کې تر يو څه بريده داسې يوه ژبه (وګورئ: الګول) جوړه شوې هم وه چې د پروګرامليکني روغ اصول يې چمتو او پلي کوي، چې د کنټرول جوړښتونو کارونه هم پکې يادولی شو. خو الګول د ډېټا ډولونو او نورو هغو ځانګړنو له کمښت سره مخ وه چې د عملي پروګرامليکني لپاره ورته اړتيا وه، په ورته وخت کې له شک پرته د يوې ښې ښوونيزې ژبې لپاره خورا پېچلې او بې ثباته ژبه هم وه.

نيکولس ويرت په ETH يا د سويس د ټکنالوژۍ په فدرالي اداره کې د ۱۹۶۰ کلونو په اوږدو کې له يوې کمپټې سره کار کاوه چې په دې هڅه کې وه چې يوه داسې ژبه رامنځته کړي چې د الګول پر ستونزو برلاسي وي او نوموړې ژبه د کمپیوټر جوړوونکو او ګټه اخيستونکو لپاره رابښکونکې او عملي ژبه اوسي. خو له يو څه وخت وروسته، ورت د کمپټې له مبهمو پايلو څخه ناخوښه شو، او د يوې نوې ژبې، پاسکال، په جوړولو يې لاس پورې کړ، او په ۱۹۷۰ کې يې د ژبې خصوصيات 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 ګټه اخیستونکو ته دا اجازه هم ورکوله چې د ځاني ډېټا ډولونو په ترکیولو 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)، پېژندګڼه<sup>1</sup> (اورد 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 وکاره. د پاسکال په مقبولیت کې د نوموړې ژبې د تالیفونکو نوبتګر ډیزاین هم لوی لاس درلود. د هغه وخت د ماشین-اړونده<sup>2</sup> تالیفونکو پر خلاف، د پاسکال تالیفونکي په مستقیم ډول ماشین ګوډ 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 پیل لیکل شوی وي، P-code ژباړي او ورپسې د ماشین مناسب هدايات اجرا کوي. دا په دې معنا وه چې د پاسکال تالیفونکي 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 ورل کېدای شوی. همدا ستراتیژي دوه لسيزې وروسته د وېب کاريا لانونو لپاره د نوميالی ژبې د جوړوونکو له خوا وکارول شوه. (وگورئ: جاوا)

## د پاسکال د پروگرام جوړښت

لاندينی ساده پروگرام په پاسکال ژبه کې د يوه پروگرام اساسي جوړښت تشریح کوي. (په پرېمخ لیکل شوي کلمات، اړين-ويوکي دي چې د پروگرام د جوړښت لپاره کارېږي). پروگرام په Type برخه پيل کېږي چې د ګټه اخيستونکي ډېټا ډولونه وضع کوي. په دې برخه کې کتارونه، ټولګي، ريکارډونه (مرکب ډولونه چې د ډېټا بېلا بېل اساسي ډولونه لرلای شي) لري. په دې ځای کې تر لسو پورې د (کاملو اعدادو) يو کتار تعريفېږي چې IntList بلل کېږي.

د Var يا متحول برخه د پروگرام ځانګړي متحولونه وضع کوي. متحولونه کولای شو چې يا د ژبې د ځاني ډېټا ډولونو په مرسته تعريف کړو (لکه عدد) او يا يې د هغو ډولونو په مرسته تعريف کړو چې ترمخه په Type برخه کې تعريف شوي دي. د پاسکال يوه مهمه ځانګړنه دا ده چې مخکې له دې چې په متحولونو کې وکارول شي، بايد د ګټه اخيستونکي ډېټا ډولونه تعريف شي. په همدې ډول متحولونه بايد وړاندې له دې چې په پروگرام کې وکارول شي، بايد وضع شي. ځينې پروگرام ليکونکي دا پېچلتيا يو څه محدودوونکې ګڼي، خو په اصل کې دا محدوديت، د مثال په ډول، له يوې ټايپوګرافيکي (چاپي) تېروتنې مخه نيسي،

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

*begin* کلمه کوډ، د پروګرام اجرائي برخې ته ور پېژني. د څرخ لپاره د اړتیا وړ متحولونه لومړی د صفر د یوې وېلیو په ټاکنه پیل<sup>1</sup> کېږي. پام مو وي چې په پاسکال کې := (کالن او د مساوي نښه) د وېلیوگانو د ټاکلو لپاره استعمالېږي. باندینی *if* عبارت د دې لپاره استعمال شوی چې ګټه اخیستونکی د توکوونو 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

```

---

<sup>1</sup> 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 اغېز

پاسکال په سوداگریزه برخه کې په منځنۍ کچه بریالی 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 اپیل-۲، چې د دې کسانو لپاره لومړني بدیلونه یوازې د ماشین ژبه او یا د بېسیک یو نارغېدلی ورژن وو. د ۱۹۸۰ کلونو په را وروسته وختونو کې بورلېنډ انټرنېشنل له تریبو پاسکال سره صحنې ته راووت. دې تالیفونکي د پي-کوډ پر ځای له مستقیم تالیف څخه ګټه اخیسته چې په دې ډول پاسکال خپله چټکیتا او اغېزمنتیا له لاسه ورکړه. نوموړي تالیفونکي د پروګراملیکنې مختلط کاري چاپیریال درلود چې د سافټوېر جوړول

يې د پخوانيو درنو او گرانبيه تاليفوونکو لکه د مايکروسافت تاليفوونکي، په پرتله ارزانه او اسانه کړل. تېرو ډېر زيات شهرت وموند او په لنډ وخت کې يې هغه ژبني زياتونې په خپل ځان کې را پورته کړې چې د استازواله پروگرامليکني ملاتړ يې کاوه. خو پاسکال تر ټولو زيات نوم له خپل هغه رول څخه تر لاسه کړ چې د پروگرامليکني د بنوونې او الگوريتمونو د څرگندولو لپاره د لومړني ژبې په توگه وپېژندل شوه.

خو په ۱۹۹۰ کال کې د بريا اوج د سي او سي پلس پلس په برخه شو. دغو ژبو د پاسکال په پرتله له ډېر کريپټک<sup>1</sup> نحوې ترکيب څخه گټه پورته کوله او د ډېټا ډولونو په اړه د پاسکال له ميکانيزم څخه هم تشه وه. سيستيمي پراگرام ليکونکو سي ژبې ته ځانگړي لومړيتوب ورکاوه چې کولای يې شواى په دې ژبه کې له ماشين سره له نږدې کار ترسره کړي او د ډېټا ډولونو له تريفولو پرته يې په حافظه کې د لاسوهنې وړتيا په لاس کې ورکوله. سي له دې امله چټک پرمختگ وکړ چې لومړني ډوبلپران يې د يونيکس د مهمو ډوبلپرانو له ډلې څخه وو، چې هغه مهال د کمپيوټري چاپريالونو په کېمپس کې يو ډېر نوموتی عامل سيستيم وو.

د ۱۹۹۰ کلونو په اوږدو کې، سي، سي پلس پلس او جاوا د کمپيوټر پوهنې د تدريس لپاره د پاسکال ځای نيول پيل کړل. پاسکال د جوړښتواله پروگرامليکني د مفاهيمو په رابرسېره کولو او د کمپيوټر پوهانو د يوه نسل په بنوونه کې په مرسته

---

<sup>1</sup> Cryptic

سرپرته، وکولای شو چې د کمپیوټر پوهنې په ډگر 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 اېډا په جوړولو کې هم پرېمانه اغېز درلود، هغه ژبه چې د امریکې د فدرالي دولت له لوري هم ومنل شوه او جوړښتواله پروگراملیکنه یې د استازواله پروگراملکني له ځانگړنو سره ترکیب کړې وه، او د کوچنیو پروگرامونو د پراخه پیکجونو د مدیریت لویه 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 دېټابېسونه او نورې شته سرچینې له وېبسایټونو سره وصل کاندې. د دې ډول اتصال خصوصیات د عامه-ګټوې په اړیکځای<sup>1</sup> 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)<sup>2</sup> نومېدله. د ساده دېټا پروسېس د متونو 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 دېټا 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 ډېری انټرنېټي پوستغالي د یونیکس سیستمونو په اساس جوړېدل، نو دا طبعي وه چې د وېب ماهرین او د کارپالونو جوړوونکي د خپل CGI متونو د لیکلو لپاره پرل ته راوګرځي.

---

<sup>1</sup> Common Gateway Interface

<sup>2</sup> Practical Extraction and Report Language

د یونیکس د ډېری متنواله ژبو په څېر، د پرل نحوې ترکیب هم سي ژبې ته ورته وو. خو د سي ژبې تر شا فلسفه دا وه چې بنيادي ژبه يې له دا ډول کړنو او فعالیتونو څخه برخمنه وه چې د معیاري او اضافي<sup>1</sup> پروگرامونو له لارې سمبالده. بل خوا، پرل، خپلې زیاتره کړنچارې<sup>2</sup> د یونیکس له مرستیالو پوستغالو څخه پیلوي، لکه sed یا اصلي ایډیټور، سي شېل، آگ، او هغه منظمې افادې چې د یونیکس گټه اخیستونکي ورسره بلدتیا لري. په دې ژبه کې د «hash» په نوم ډېټا ډول (د جوړه کیلیانو او وېلیوگانو یوه ټولگه) شتون لري چې په پروگرام کې د تغیراتو راوستل اسانه کوي او د انټرنېټ د هوسټونو او د هغوی د انټرنېټي پروټوکولونو ادرسونه کوري.

وال به د پرل گټه اخیستونکو ته د هغوی د غبرگون ځواب د نوو ځانگړنو او فنکشنونو په اضافه کولو ورکاوه او د گټه اخیستونکو له دې نیوکو څخه یې استفاده کوله. د وال تگلاره دا وه چې له پروگرام 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)
```

---

<sup>1</sup> add-in

<sup>2</sup> functionality

```
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 ګډه کوي، او دا د یوه مختلف جوړښت زده کړه درکوي. د دې یوه عملي ګټه 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 پروگرامليکني ته لاره هواروي. خو د پرل ډېری مینوال دې ژبې ته د یوې هر اړخیزې ژبې او د وېب پروگرامليکني د ننګونکې نړۍ لپاره د یوې اساسي ابزارټولګې په سترګه کوري. خومره چې د ۱۹۹۰ کلونو په وروستيو کې ژبې وده کوله، په هماغه ډول د استازواله پروگرامليکني ځانګړنې هم ور زیاتېدې.

## د پرل ډېروګرام 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 پرنت کوي. لومړنۍ کرښه یونیکس ته وايي 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 غونډ 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 پرنت 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 انټرنېټي پوستغالو جوړولو لپاره د نورو متنواله ژبو له خوا يو څه کم رنگه شوې، خو پرل پوهه بالغه ټکنالوژي ده چې اوس هم په لويه کچه کارول کېږي، په ځانگړي ډول د ډېټا د استخراج، اړولو، او لاسوهنې لپاره. د پرل د ټوليز ارشيف شبکه (CPAN)<sup>1</sup> له ۱۲۰۰۰ څخه زيات ماژولونه لري چې په وړيا توگه د پروگرام ليکونکو 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.

\*\*\*

## (۲۰)

## پي اېچ پي (PHP)

پي اېچ پي له پراخ شهرت برخمنه متنواله ژبه ده چې په بنسټيز ډول د خوځنده<sup>1</sup> ويبپاڼو د جوړولو لپاره استفاده کېږي. پي اېچ پي په ۱۹۹۴ کې د ډنمارکي پروگرام ليکونکي راسموس لږډورف له خوا رامنځته شوه تر څو د پرل د يو ډله متنونو پر ځای د خپلې ويبپاڼې د مدیریت لپاره گټه ترې واخلي – او نوم يې هم له همدې ځايه سرچينه اخيستی، «شخصي ويبپاڼه يا personal home page». لږډورف په ۱۹۹۵ کې د ژبې لومړنی ورژن له «قالب ژباړونکي يا form interpreter» سره يو ځای را وايست. په ۱۹۹۷ کې د ژبې تجزيه کوونکی<sup>2</sup>، دوو اسراییلي ډولپلورانو زيف سوراسکي او آندې کوتمان وليکه چې په ۱۹۹۸ کې يې PHP3 بازار ته وړاندې کړه؛ او تر دې مهاله پورې د PHP لومړني توري په خپله بډرته 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 موند او همدارنگه له MySQL او نورو ډېټابېسونو او د انټرنېټي کارپالونو همغږه ټکنالوژيو سره يې اړيکو او ارتباط موندنې هم پرمختگ کاوه.

<sup>1</sup> dynamic<sup>2</sup> parser

عموما پې اېچ پي پر يوه وېب سرور ځغلي او د پي اېچ پي کوډ پروسېس کوي چې ډېر کله له يوې وېب پاڼې سره مختلط شوی وي. د Hello World يا سلام نړۍ کلاسيک پروگرام په پي اېچ پي کې داسې ښکاري:

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

د پي اېچ پي پروسېس کوونکی پورتنی کوډ يوازې د محدودونکو <? او ?> دننه تجزيه کوي. د دې محدودونکو معادل بيا په لاندې ډول دی:

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

سربره پر دې چې کوډ د HTML په پاڼو کې مختلط وي، پي اېچ پي په کمانډ کرښه (command line) کې په تعاملې ډول هم کارول کېدای شي، چېرته چې دې ژبې د آگ، پرل او شېل په څېر متنواله زرو ژبو ځای نیولی دی. پي اېچ پي د گټه اخيستونکي د اړیکو ځای له لایبرېريانو سره هم وصل کېدای شي (لکه GTK+ د لېنکس/يونیکس لپاره) خو داسې پوستغالي جوړ کړو چې د سرور پر ځای د کلاينټ په ماشين باندې وځغلي.

پي اېچ 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 ځانگړو فنکشنونو په مرسته پروسېس شوي وي چې انځورونه، متني فایلونو، د ډېټا ريکارډونه او داسې نور بېرته راگرځوي. د دې ترڅنګ، پي اېچ پي-5 د شخصي

او خوندي، غړو متحولونو، کنسټرکټرونو او ډيکنسټرکټرونو او د نورو هغو ځانگړنو په گډون چې په سي پلس پلس او نورو ژبو کې تر سترگو کېږي، د استازو لپاره هم بشپړ ملاتړ چمتو کوي.

د پرانیستو سرچینو د استازو او فنکشنونو گڼ شمېر لایبرېرېانې په پي ایچ پي کې شتون لري چې د پي ایچ پي متنونو ته اجازه ورکوي تر څو عادي انټرنېټي فعالیتونه ترسره کړي، لکه، د ډېټابېس سرورونو (لکه MySQL) ته لاسرسی، او همدا رنگه د مشهورو انټرنېټي فارمتونو د سمبالنټ لپاره ژبني زیاتونې چمتو کوي لکه د اډوب فلش خوځښتونه او داسې نور. پروگرام لیکونکي د PEAR<sup>1</sup> (د پي ایچ پي د زیاتونو او کارپالونو زېرمه) له لارې د پي ایچ پي لوی شمېر سرچینو ته لاسرسی موندلی شي.

د اساسي ځانگړنو او په اسانه د متنوالی تعامل پي ایچ پي ژبه د ډېری هغو وېب ډویلرانو لپاره د خوښې وړ ژبه گرځولې چې د ټکنالوژیو د ډلې چې LAMP نومېږي، د یوه غړي په توگه گټه ترې پورته کوي. LAMP په اصل کې د لینکس، اپاشي (وېب سرور)، MySQL (ډېټابېس) او پي ایچ پي لنډیز دی.

---

<sup>1</sup> 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 پروگرامي ژبې تر ټولو زیاتې تر استعمال لاندې وې: فورټران د ساینسي او انجینري کارپالونو لپاره او کوبال د کاروباري محاسبو لپاره. خو کارپالونو ورځ تر بله پراختیا موندله او لازیات پېچلي کېدل، او په دې اساس د لوی شمېر ډول ډول وړتیاوو غوښتنه یې کوله. د مثال په ډول، ساینټیفیک پروگرام لیکونکو اړتیا لرله چې د محاسبې ترڅنګ د ډېټا پروسېس او راپور جوړونې وړتیاوې چمتو کړي. بل خوا، کاروباري پروگرام لیکونکو، ډېره زیاته اړتیا لرله چې له فورمولونو او احصائیو سره کار وکړي او همدا رنگه اعشاري عددونو او د عددونو نورو ډولونو ته یې اړتیا لیدله.

په دې اساس د ژبو ډولپلورانو د یوې داسې عام-هدفه ژبې جوړولو ته متې راوغښتې چې د کلماتو، شمېرو او ډېټا فایلونو لپاره مشترک بستر چمتو کړي. په ورته وخت کې، آی بی اېم شرکت د کاروباري او ساینسي فعالیتونو لپاره د بېلو کمپیوټري سیسټمونو ځای هر اړخیزه کمپیوټرونو «سیسټم/۳۶۰» ته ورکړ. آی بی اېم شرکت او د دوی د ګټه اخیستونکو یوه ډله Share سره یو ځای شول ترڅو د دې نوي ماشین لپاره نوې ژبه جوړه کړي.

په لومړیو کې ډیزاین کوونکو فکر وکړ چې د غوره متني او ډېټا پروسېس د وړتیا لپاره فورټران ژبې ته پراختیا 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 اېل/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 نستاګونۍ 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 اغېز کېدای، او هغه متحولونه چې د یوه بلاک دننه وضع شوي وي، یوازې د هماغه بلاک دننه او د نورو تړلو بلاکونو دننه د لاسرسي وړ وي، که نه بله لاره یې دا ده چې په نړیوال ډول وضع شي.

پي اېل/۱ د ډېټا ډولونو پرېمانه 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



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. راکړیز/ورکړیز (I/O) په خارجي لایبرېري کې د چمتو کېدو پر ځای د ژبې دننه جوړېږي، او د ژبې انعطاف منونکي آپشنونه توري، د تورو کړنې، 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 اټوماتیک (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 بنکاره ډول ونيول 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 استثنا تېروتنو (exceptions) د سمالبنټ لپاره پرېمانه اسانتیاوې لري چې دا ډول تېروتنې د هارډویر له شرایطو، ریاضیکي معادلو، د فایلونو له سمالبنټ او یا نورو شرایطو څخه را ټوکېدلای شي.

## د پروگرام بېلگه

لاندې پروگرام یو Do څرخ اجرا کوي او له یو څخه نیولې تر ټاکلو توکونو پورې اعداد شمېري. ورپسې د اعدادو د جمعي 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 ځانگړنو او د آی بی اېم شرکت د نومبالیو ۳۶۰ مېن فرېم 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 لویو دريځونو او عامل سیستمونو ته هم انتقال شوه. کله چې شخصي کمپیوټرونه منځته راغلل، پي اېل/۱ د آی بی اېم شرکت د OS/2 عامل سیستم او دغه راز د مایکروسافټ د ډاس او وینډوز

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

د جوړښتواله پروگراملیکنې د سرلاري اېډسگر ډیتکسټرا په څېر کمپیوټر پوهانو پي اېل/۱ د واضح او ښه-رغېدلي جوړښت د نشتون له کبله رسوا کړه. نوموړي په ۱۹۷۲ کې د تیورینګ جایزې د ترلاسه کولو پر مهال په خپل لېکچر کې وویل چې: «زه خو یې اصلاً نه وینم چې مور به د خپل ذهني چوکاټ دننه د خپلو پروگرامونو وده په خپل ځای وساتلای شو، کله چې زموږ پروگرامي ژبه له دومره زیاتو ځانګړنو سره – گوزاره مې وکړه! دمخه لا زموږ له ذهني کنټرول څخه په تېښته 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. پي 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. پي اېل/۱ ډېرې بېلګې چمتو کړې چې د ژبو ډیزاین کوونکي یې د غوره ډیزاین د تطبیق لپاره استفاده ترې 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 اتوماتیک کولو امکان ته هوسپدل چې د بېلګې په ډول منطقي استنباط ترې یادولای شو. ګڼ 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 اتوماتیک استدلال او پوهنښودنې<sup>۱</sup> 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 ترې ترلاسه شي. بل خوا، د پرولوګ پروګرام، د حقایقو (فرضیو) په یوه ټولګه پیلېږي 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).

<sup>1</sup> knowledge representation

په دې پسې پروگرام لیکونکی منطقي اصول تعريفوي 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 وايي چې  $x$  نومي یو تن د  $y$  پلار دی که چېرته هغه د  $Y$  له والدينو څخه وي او نر وي. دوهمه څرګندونه وايي 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 پروگرام ځغلي، نو همدا ډول ګروپږنې<sup>1</sup> یا څرګندونې چې د رېسټینولۍ 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 ګروپږنو پلي کېږي او هڅه کوي چې یو داسې منطقي زنځیر جوړ کړي چې د هغې ګروپږنې 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 کونکي 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

---

<sup>1</sup> query

جوړولو په هیله وو. دا مینه په ۱۹۸۰ کلونو کې د جاپان د پنځم نسل په کمپیوټري پروگرام کې را وتوکېده چې د منطقي زبر کمپیوټرونو<sup>1</sup> د جوړولو په لټه کې وو او پرولوگ یې د خپلې خوښې وړ ژبه وټاکله. که څه هم ځینې ماشینونه جوړ شول، خو د دې نظریې هېڅکله رښتیني بری په برخه نه شو. یورلېنډ انټرنشنل (د بریالي تریو پاسکال جوړوونکي) تریو پرولوگ را وایستله چې یاده ژبه یې د هغو زده کوونکو لپاره د لاسرسي وړ وگرځوله چې له شخصي کمپیوټرونو څخه ګټه پورته کوي، که څه هم په دې ژبه کې ځینې غیرمعیاري ژبنی زیاتونې استفاده کېدې.

که څه هم په سوداګریز ډګر کې یې بریا محدوده وه، خو پرولوگ د مصنوعي ځیرکتیا د څېړنو په لوی شمېر ډګرونو کې کارول شوې. د نوموړې پر اصولو ولاړ جوړښت د ځیرکو سیستمونو، پوهنښتونو<sup>2</sup>، او طبیعي ژبې پروسپس لپاره مناسبه ژبه وه. یاده ژبه د هغو نمونوي (پروتایپي) سیستمونو 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.

<sup>1</sup> supercomputer

<sup>2</sup> 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](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 گایډو 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 کومیدي له نومیالی ډلې «مونتي پایتان»<sup>1</sup> څخه اخیستل شوی.

پایتان په ځانګړي ډول د سیسټمي مدیریت کونکو، 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. نوموړې 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 کورنی کې کارېده. د مثال په ډول، لاندې کوچنی پروګرام د فارن هایت درجه په خپل سلېوس معادل باندې اړوي:

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

د هغو سیمیکالانو او لینډیو د نشتون له کبله چې په سي او اړونده ژبو کې تر سترګو کېږي، پایتان تر ډېره بېسیک ژبې ته ورته برېښي. همدا رنگه دې ټکي

---

<sup>1</sup> Monty Python

ته مو هم پام وي چې اړتيا نشته د ورکړيزې ډېټا ډول بيان شي. ځغلمهاله ميکانيزم په خپله له هغې افادې څخه پوهېږي چې د print په عبارت کې کارول شوي چې ډېټا يو عددي ډول دی. په دې ترتيب، د پايتان پروگرامونه د سي، جاوا او حتی پرل پروگرامونو په پرتله لنډ او ساده جوړښت لري. ساده نحوي ترکيب او د ډېټا ډول نه درج کول په دې معنا نه ده چې پايتان يوه «جدي» ژبه نه ده، بلکې پايتان د استازواله پروگرامليکني ټولې اسانتياوې په خپل ځان کې را غونډې کړې دي.

د پايتان پروگرامونه په تعاملي ډول د کمانډونو په درج کولو په اسانه او چټک ډول ليکل کېدای او ورپسې نوموړی متن بايت کوډ ته اړول کېدای شي. بايت کوډ هغه خپلواک-ماشينه بنودنه ده چې د هر ډول ماشين د چاپېريال لپاره په ډيزاين شوي ترجمان ځغلمدای شي. د دې د بدیل په توگه، داسې ترجمان پروگرامونه هم شته چې د پايتان متن د سي ژبې سرچينه فايل ته اړوي چې وروسته په لوړ سرعت تاليف (کمپايل) کېدای شي.

پرل تر دې دمه د يونيکس او وېب اړونده کاريالونو لپاره يوه مشهوره متنواله ژبه ده. پرل له ځواکمنو ځاني افادو او pattern-matching ميکانيزمونو څخه برخمنه ژبه ده او دغه راز ډېر نور فنکشنونو هم لري چې د عملي متنوالی لپاره گټور پرېوځي. بل خوا پايتان، ډېره عمومي او په څرگند جوړښت رغېدلي ژبه ده چې د لوی شمېر کاريالونو لپاره يوه وړ ژبه او د لويو او پېچلو پوستغالو لپاره په اسانۍ سره د پراختيا موندنې له لويو وړتياوو څخه برخمنه ده.

## اضافی نوسٹ:

- 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.

\*\*\*

(۲۴)

## آرپي جي (RPG)

ڊپري کاروباري پروگرامونه چې د مڻن فرېم ڪمپيوٽرونو لپاره ليڪل شوي وو، داسي وو چې له فائيلونو څخه به يي ڊپٽا لوستل، نسبتاً ساده پروسې به يي ترسره ڪولې او يا به يي پرنٽ شوي راپورونه د راکريز په ډول برابرول. د ۱۹۶۰ کلونو په اوږدو کې ځيني خلک په دې باوري وو چې کوبال يوه عام-هدفه (خو کاروبارواله) ڪمپيوٽري ژبه ده او د هغو ڪسانو لپاره اسانه ژبه ده چې مسلڪي پروگرام ليکونکي نه دي. که څه هم د خبرې حقيقت په دې ډول نه وو، خو آي بي ايم شرکت د آر پي جي يا (راپور توليدونکي) په جوړولو بريالي شو، هغه ژبه چې د پروگرام ليکونکو (د مبتديانو په گډون) لپاره په دې موخه جوړه شوې وه چې د کاروباري راپورونو توليدول ورته اسانه ڪري.

ڊپري کوبال پروگرامونه ڊپٽا لولي، ازمويني او محاسبې ترسره ڪوي او پايلې يي پرنٽ ڪوي. «آر پي جي» چې د نوو مڻن فرېم ڪمپيوٽرونو، سيسٽم/۳۶۰ او کوچنيو ڪمپيوٽرونو، سيسٽم/۳۶۰ لپاره جوړه شوه، د لومړي ځل لپاره په ۱۹۶۴ کې د استعمال لپاره وړاندې شوه. نوموړې ژبه د کوبال پورته ياده پروسه ساده ڪوي او د پروگرام د کوڊ ڊپري عبارات له منځه وړي.

د آر پي جي يو ڪلاسيڪ پروگرام د آر پي جي په دائروي جريان کې جوړېږي، چې درې پړاوونه لري. په ورکريز پړاو کې، ورکريز واحد(ونه)، فائل ډول، د

لاسرسی خصوصیات، او د ډېټا ریکارډ، جوړښت مشخص کېږي. (دا خصوصیات ښه زیات څرګند لیکل کېدای شي.) د پروګرام زړه هغه محاسبې مشخصوي چې باید په ډول ډول ډېټاساحو 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 اېم شرکت هم ویشوال-اېچ آر پي جي را وباسله چې د مایکروسافټ وینډوز 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 ګڼ وسایل شتون لري چې ډول ډول ډېټابېس 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 پروگرامي ژبه ده چې په وروستيو کلونو کې يې ښه نوم پيدا کړی، په ځانگړي ډول د وېب جوړونې په برخه کې. نوموړې ژبه د يوکيرو ماخوموتو له خوا ډيزاين او په ۱۹۹۵ کې د استعمال لپاره راووته؛ روبي، غونډلې نځوي ترکيب لري چې د پرل او نورو متنواله ژبو گټه 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 سمالتاک (Smalltalk) ته ورته ده. ماخوموتو پر دې ټينگار کړی چې د ژبې ډيزاين په داسې شکل وکړي چې د پروگرام ليکونکي لپاره طبيعي او خوندوره اوسي، نه دا چې د پروگرام ليکونکي د ماشين پر اړتياوو خپله انرژي راتوله 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 توگه 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 میتودونو څخه گټه 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 گڼل کېږي.

هر فنکشن یو میتود دی چې په یو نه یو کلاس پورې اړه لري. په دې ډول -  
 5.abs پر 5- د مطلقي وېلیو میتود پلي کوي او بېرته 5 راگرځوي. په ورته ډول  
 wireless wombat.length د همدې متنکرنيې اوږدوالی راگرځوي چې ۱۵  
 کېږي. روبي له ډېټا جوړښتونو لکه کتارونه او هېشونو<sup>1</sup> سره د کار کولو لپاره گڼ  
 شمېر ځاني میتودونه هم لري، همدا رنگه ډېری دودیزې لایبرېرېانې او کاريالونه  
 هم ورکې د لاسرسي وړ دي.

د ټولو لویو عامل سیستمونو لپاره روبي ترجمانان لري. له یوه فایل څخه د پروگرام  
 د اجرا کولو او یا لوستلو تر څنګ چې په ډېرو متنواله ژبو کې ترسره کېږي،  
 روبي د ازماينېتي 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 وځغوي چې یو  
 خاص پروگرام دی چې په تعاملي ډول عبارتونه او افادې ارزوي. د puts کمانډ

---

<sup>1</sup> Hashes

منتکرښه Hello world درج کوي. ورپسې ارزوونکي داسې راپور چمتو کوي چې: puts میتود تشه وپلویو (nil) راگرځولي 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 بایت کوډ تولیدوي، د جوړېدو په حال کې ده، او د یوه لا مستقیم تالیفوونکي امکان هم لیدل کېږي.

### روبي پر اورگاډي (Ruby on Rails)

د روبي د پروگراملیکنې لپاره ډېر نومیالی کاري چاپیریال روبي پر اورگاډي دی، چې یو پرانیستي سرچینه کارپالی چارچوکات دی چې خاص په دې موخه جوړ شوی چې داسې پروگرامونه ورباندې ولیکل شي چې وبسایتونه له ډېتابېسونو سره وصل کړي. نوموړی چارچوکات د ماډل-لید کنټرولونکې<sup>1</sup> تگلارې باندې بنا دی چې ډېتا ته لاسرسی او منطق د ګټه اخیستونکي له اړیکځای څخه بېلوي؛ همدا رنگه داسې یو چوکات (scaffolding) هم لري چې په تېزۍ سره ډکېدای شي خو له اساسي فعالیت څخه برخمن داسې وبسایتونه چمتو کړي چې د ډېتا په اساس حرکت وکړي. ډوېلېران کولای شي چې د روبي په مرسته نسبتې کارپالونه<sup>2</sup> او شته پیکجونه هم پراخه کړي.

<sup>1</sup> Model-view controller

<sup>2</sup> plug-ins



## اضافی 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/](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](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](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 ډبرو په زړه پورې گټو څخه یوه د هغو سیستمونو 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 کریستن نېگیارد 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 گڼه گڼه کې موټرونه او یا بانک ته په لیکه کې ولاړ مشتریان. د بانک په تقلید کې، د مثال په ډول، استازي (مشتریان) له ځانگړو 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.

نېگیارد له خپلو نظریاتو څخه گټه واخیستله تر څو داسې سمبولونه او جریان بنوودونکي ډیاگرامونه جوړ کړي چې په یوه تقلید کې روانې پېښې په ډاگه 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 پروگرام ولیکل شي چې یو داسې حالت وکارې په کوم کې چې ډېر مشتریان یا موټرې په یوه وخت کې د حرکت په حال کې وي.

د ۱۹۶۰ کلونو په لومړیو کې نېکیارډ له اولې-یوهان ډال سره یو ځای شو چې په سیستمي پروگراملیکنه او د کمپیوټري ژبو په ډیزاین کولو کې یې ډېره تجربه لرله. دوی دواړو په گډه کار وکړ او یوه نوې ژبه یې جوړه کړه چې سیمولا نوم یې پرې کېښود، او پر تقلیدي پروگراملیکنه<sup>1</sup> د دوی د ټینکار لیدلوری یې را غبرگاو (انعکاس کاوه). د سیمولا د ډیزاین پر مهال، دواړه لیکوالان د داسې ډېټا جوړښت<sup>2</sup> د جوړولو په هڅه کې وو چې د هم مهاله کارونو یا پېښو لپاره غوره اوسي. د مثال په ډول، د موټرونو د گڼه گونې 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 بېل ډول پاتې شي او په پرلپسې توگه باید اډېټ شي.

الکول ۶۰ ترمخکې لا د کوډ بلاکونو د تعریفولو لپاره یوه لاره لرله چې خپله ځاني سیمه بیزه ډېټا<sup>3</sup> او کارونه (actions) یې په ځان کې ذخیره کولای شوی. پر دې سربېره، دا ډول بلاکونه په تکراري ډول بلل کېدای شي داسې چې په ورته

---

<sup>1</sup> simulation programming

<sup>2</sup> data structure

<sup>3</sup> local data

وخت کې يې ډېرې کاپيې پرايستي شې. خو دې بلنو بيا هم ترتيبې بڼه لرله او هم مهاله نه وې. په نوې سيمولا-۱ ژبه کې (چې ۱۹۶۵ کې راغله)، ډال او نېکيار داسې لاره وپنځوله چې د هم مهاله پروسېس تقليد وکولای شي. که څه هم کمپيوټر بنايي يوازې يو پروسېس کوونکی ولري داسې چې په يوه وخت کې د کوډ د بلاک يوازې يوه کاپي اجرا کېدای شي، خو سيمولا ځانگړي متحولونه وضع کړل تر څو تقليدي مهال تعقيب کړي. کنټرول به د يوه بلاک له استازي څخه بل استازي ته «ټوپ» کړل، گواکې 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 الگول ټولې ځانگړنې په ځان کې لرلې، خو د هم مهاله پېښو د ماډل کولو لپاره يې ډېره بڼه استفاده ترې 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 بنودنې لپاره لا ډېره عمومي گټه 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 توگه رامنځته کړ چې د هماغه ډول استازو د جوړولو لپاره کارېدای شي. همدا رنگه 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 اېدا دیزاین کوونکو، سیمولا ته د هغې د جوړښتواله نظریاتو له مخې سترګې وړ اړولې، په همدې وخت کې نومیالی سي پلس پلس ژبه د «سي له کلاسونو سره» د جوړولو په هڅه را پیل شوه چې په جوړښت کې یې د سیمولا کرښې 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/jsk11/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/jsk11/talk.html>. Accessed August  
 21, 2007.

\*\*\*

( ۲۷ )

## سمالټاک (Smalltalk) يا وړې خبرې

په ۱۹۷۰ کلونو کې په زیراکس پالو آلتو څېړنيز لابراتوار (PARC) کې د کار کولو په جريان کې کمپیوټرپوه اېلن کای گن داسې نظريات او وسايل وپنځول چې د نن ورځ شخصي کمپیوټرونو ته يې لار کړې ده. د يوه وړانديز شوي لپټاپ د ډيزاين کولو پر مهال چې Dynabook نومېده، کای پرېکړه وکړه چې د نوموړي د عامل سيستم د جوړولو لپاره له نوې تکلارې څخه گټه پورته کړي.

د سمالټاک په جوړولو کې کای دوه مهمې مفکورې په ذهن کې نيولې 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 دوديز تاليف د ستومانوونکې پروسې باندې ځان سرگردانه کړي چې پروگرامونه په تعاملې شکل جوړ، امتحان او تصحيح کوي. سېمور پېپرټ دمخه لا لوگو جوړه کړې وه چې يوه تعاملې او له گرافیکي اړخه غني ژبه وه او ماشومانو ته د کمپیوټر د ډېرو پېچلو مفاهيمو د تدريس لپاره يوه 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 سمالتاک کې وکارول شو، هغه استازواله پروگراملیکنه وه چې د لومړي ځل لپاره په سیمولا-۶۷ ژبه کې رامنځته شو. د ژبو شته ځانگړنو ته یوازې د ساده کلاسونو او استازو د ورزیاتولو پر ځای، کای سمالتاک 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 سمالتاک کې استازي ترې جوړ شوو. کټه اخیستونکو نوي کلاسونه هم جوړولای شوی چې د «ځاني» کلاسونو چلند به ورسره کېده. دې خبرې ته د انډېبنې کولو هېڅ اړتیا نه وه چې وړاندې له دې چې یو خاص ډول متحولونه وکارول شي، هغه لومړی باید بیان شي؛ په سمالتاک کې متحولونه په هر ډول استازي پورې تړل کېدای شي.

د دې لپاره چې یو پروگرام 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 بېسیک ژبې ټاکنیز<sup>1</sup> عبارت ته وګورئ:

Total = Total + 1

---

<sup>1</sup> Assignment

د بېسيک په څېر په يوه دوديزه ژبه کې له پورتنې عبارت څخه داسې مفهوم اخیستل کېږي چې « ۱ له هغه وېليو سره جمع کړه چې د Total تر عنوان لاندې موقعيت کې ذخيره ده او حاصل يې بېرته په هماغه موقعيت کې ذخيره کړه. » خو په استازواله سمالتاک ژبه کې د همدې معادل عبارت په لاندې ډول دی:

$$\text{Total} \leftarrow \text{Total} + 1$$

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 ورباندې پوهېږي. له همدې کبله، هغه استازی په خپله وېليو کې ۱ جمع کوي او بیا هغه وېليو د نوي استازي په توګه واپس کوي، چې اوس د Total متحول په وسیله ورته مراجعه کېږي.

په سمالتاک 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 پورې تړلي متحولونه، یوه «کاري ساحه» جوړوي، چې د ډیسک پر مخ په دوراني شکل ذخیره کېږي.

د سمالتاک د پروګرام لیکونکي لپاره د سمالتاک 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 سمالتاک سیستم دننه پراته وي. کای، سمالتاک په خپل ذهن کې د یوه داسې بشپړ کاري



چاپېريال په توگه مجسم کړې وه چې د هغو پروگرام ليکونکو له خوا هم پراختيا وموندلی شي چې د پروگرامليکني تجربه نه لري، او نوموړي خپل مخکښ گرافيکي اړيکځای د داسې روش په توگه ډيزاين کړ چې د کټه اخيستونکو لپاره دا کار اسانه کړي خو له سيستم سره نېغ په نېغه کار وکولای شي.

سمالتاک يو «مجازي ماشين» لري چې هدايات يې د هر لوی کمپيوټري سيستم د ډول لپاره په ځانگړي کوډ کې تطبيق کېدای شي. د سمالتاک د ثابت جوړښت او له استازو څخه د هر شي د جوړولو د وړتيا له امله، د سمالتاک شاوخوا ټول سيستم پخپله سمالتاک کې ليکل کېږي، چې دا کار اسانه کوي چې کله يو ځل ماشين-ارونده تفصيلات چمتو شي، هغه بل کمپيوټر ته (له تغير پرته) ولېږدول شي.

د نوموړې ژبې د سبک، ثبات او په شخصي کمپيوټرونو کې د شتون له امله، سمالتاک له ۱۹۸۰ کلونو څخه وړاندې د پام وړ توجه را جلب کړه. ياده ژبه د اصلي تگلوري په کاريالانو کې په پراخه توگه نه ده کارول شوې، په دې چې د کلاسونو او توارث د ميتودونو د تعقيبولو لپاره د اړتيا وړ ميکانيزمونه په هغومره اغېزمن ډول پکې پلي کول ستونزمن وو، څومره ساده ميکانيزمونه چې دوديزو ژبو کارول. د اغېزمنتيا او د زده کړې په برخه کې د اسانتيا له کبله په شته ژبه باندې د استازواله ځانگړنو د ور زياتولو تگلارې (څنگه چې سي پلس پلس له سي څخه جوړه شوه) د ډېرو کسانو زړونه خواږه کړل.

د سمالتاک معنوي ځواک دا ژبه، د مصنوعي ځيرکتيا او پېچلي تقلید په ډگر کې د ځانگړو پروژو لپاره يوه زړه راښکونکې ژبه گرځولې، او دا د هغو کسانو لپاره يو لوی انعام دی چې پر استازواله روش بنا يوه کره ژبه غواړي چېرته يو کاربال په واضح ډول د رښتيني نړۍ حالت انځورولای شي. سمالتاک ماشومانو (او نورو) ته د پروگرامليکني د تدريس لپاره لاهم يو غوره انتخاب دی. د سمالتاک يوه نسخه چې squeak بلل کېږي، د گرافیک او نورو فنکشنونو لپاره يو غني کاري چاپېريال برابروي. Squeak او د سمالتاک يو شمېر نور تطبيقات د مختلفو کمپیوټري سیستمونو لپاره په وړيا 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 آوسټرھاوټ له خوا جوړه شوې، ټي سي اېل ژبه د متنوالی، پروټوټایپ، اړیکځایونو د کتنې او مختلطو کار یالانو د جوړولو لپاره استفاده کېږي.

ټي سي اېل یو ډېر غیر معمولي ساده او ثابت نحوي ترکیب لري. د ژبې متن د کمانډونو او د هغو د آرگومنټونو (پارامترونو) یوه ساده لړۍ وي. (چې یا ځاني وي او یا د ګټه اخیستونکي له خوا تعریف شوي وي). پخپله یو کمانډ کېدای شي چې یو آرگومنټ وي او یا یو بل کمانډ، چې په نورو ژبو کې د فنکشنې 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 ځغوي. 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 } {
```

```
GetData
DisplayData
}
```

په عمل کې، ډېری کمانډونه د عامل سیستم چې معمولا یونیکس یا لینکس وي، مرستندویه پوستغالي دي چې استعمالېږي. تي سي اېل یو شمېر کټور ډېټا جوړښتونه هم لري لکه اشتراکي کټارونه<sup>1</sup> چې د ډېټا توکونو جوړې پکې راځي، لکه:

```
set abbr (California)
```

## زیاتونې او کټې

تي سي اېل یو شمېر زیاتونې هم لري چې د مثال په ډول، د MySQL په څېر د ډېټابېس مشهورو فارمتونو 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 روپي لپاره د ګټه اخیستونکي د اړیکځایونو جوړولو لپاره یوه لایبرېري وړاندې کوي.

تي سي اېل د شته کاريالونو د نښلونکو لپاره د «سریس» په توګه پېژندل کېږي. نوموړې ژبه نسبتاً اسانه او یو متن په تعاملي شکل کوري (چې ډېر پر کمانډ کېنه ترسره کېږي) او بیا دا متن د یوه کاريال کوډ ته ور درج کوي. کله چې

---

<sup>1</sup> associative array

کاریال ځغلي، د تي سي اېل ترجمان هغه متن ځغلوي، چې وروسته يې راکړيز  
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 کمپني جوړه کړې او بنسټ يې په خپل مشهور ويژوال بېسيک پروگرامي ژبې ايسنودل شوی. نوموړې ژبه د هغه تحول يوه برخه ده چې مايکروسافټ «فعال 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 ويندوز له خوا چمتو کېږي (لکه د ويندوز منني کوربه<sup>1</sup> او يا د مايکروسافټ انټرنيټي اېکسپلورر دننه اجرا کېږي).

د کلاينټ-لوري پروسيس لپاره، وي بي سکرپټ د هغو متنونو د ليکلو لپاره استعمالېدای شي چې په html پاڼو کې مختلط وي، چې له معياري ډاکومنټ آېجکټ ماډل سره د نورو ويبې متنواله ژبو په څېر اړيکه نيسي (په ځانگړي ډول، وگورئ: 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 ډوبلېران بايد له انطباق پذيره جاواسکرپټ څخه گټه پورته کړي. وي

---

<sup>1</sup> Windows Script Host

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

د وينډوز د نسخو له کبله چې په وينډوز-۹۸ را پيل شوې، ټولې نسخې يې د وينډوز متني کوربه لري، خو وي بي سکرپټونه په مستقيم ډول تر وينډوز لاندې هم ليکل کېدای شي. د وي بي سکرپټ يوه خواشينوونکې پايله د بي پامه ګټه اخيستونکو لپاره هغه متنونه دي چې چينجي<sup>1</sup> به يې لرل (لکه د «زه درسره مينه لرم»<sup>2</sup> چينجي) او يا نور په ويروس ککړ پوستغالي (malware) يا په برېښنالیک کې راغلي ورسونه.

## مثالونه

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

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

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

<sup>1</sup> worms

<sup>2</sup> I love you



په متنچوکات کې له ګټه 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 لاپېرې او اړیکځایونه لري چې د دې جوګه يې ګرځوي چې لا ډېرې پېچلې کړنې سرته ورسوي لکه د ډېټابېسونو برابرول او د وينډوز د مدیریتي آلاتو<sup>1</sup> (WMI) او فعاله خدماتي اړیکځای (ADSI)<sup>2</sup> په مرسته د وينډوز سیستمونو د نورو برخو تنظیمات ترسره کول.

که څه هم نوموړې ژبه (او کوډ به يې) 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 يې د يوه نوي پروګرامي چارچوکات په لور ګام پورته

---

1 Windows Management Instrumentation

2 Active Directory Services Interface

کری او ٲول ٲام یې د ویشوال بېسیک ډاٲ نېٲ (visual basic.net) ٲه 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://cwwashington.netreach.net/depo/default.asp?topic=repository&scripttype=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.

\*\*\*

ٲای

## ماخذونہ:

- 1- Hend-erson, Harry, (1951–), Encyclopedia of Computer Science and Technology – Rev. ed. Facts on File – 132 West 31st Street New York NY 10001, 2009.
- 2- Sinha, P.K., Computer Fundamentals, PBP Publications, B- 14, Connaught Place, New Delhi-110001, 2003.

**Get more e-books from [www.ketabton.com](http://www.ketabton.com)  
Ketabton.com: The Digital Library**