

کمپیوٹری ژپ

Ketabton.com
کمپیوٹرستان (۱)

کمپیوٲری ژبې

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

کمپیوٲرستان (۱)

ژباړه: رحمت شاه فراز

د کتاب پېژندنه

کمپیوټري ژبې

د کتاب پېژندنه

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

## لیکړ

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

|          |                         |
|----------|-------------------------|
| 94.....  | سي شارپ (C#)            |
| 98.....  | سي پلس پلس (C++)        |
| 107..... | کوبال (COBOL)           |
| 114..... | آیفل (Eiffel)           |
| 119..... | فورت (Forth)            |
| 123..... | فورٲران (Fortran)       |
| 128..... | جاوا (Java)             |
| 137..... | جاوا سکرپٲ (JavaScript) |
| 143..... | لیسپ (LISP)             |
| 150..... | لوگو (Logo)             |
| 156..... | لووا (Lua)              |
| 159..... | پاسکال (Pascal)         |
| 166..... | پرل (Perl)              |
| 171..... | پی اېچ پی (PHP)         |
| 175..... | پی اېل - ۱ (PL/1)       |

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

## سريزه

کمپیوټرپوهنه که په افغانستان کې نوی علم نه وي، نو ډېر زوړ هم نه دی، او لویه نښه یې دا ده چې په دې برخه کې تر اوسه پورې په پښتو ژبه یو باوري او معتبر اثر هم نه لرو. په دې برخه کې تحقیق، څېړنه، د لویو پوستغالو پنځول او د 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 کمپیوټرپوهنې ته تر ډېره پاملرنه کړې؛ او همدا رنگه دا هم له پامه نه شو غورځولای چې د کمپیوټرپوهنې په ځینو برخو لکه د ډېسکټاپ او موبایل کارپالونو 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 خوشالۍ 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 وګړو د دې برخې اړتیا د یوه هېواد د پرمختګ او ودې لپاره درک کړې او په پراښته 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 ښايي دا لومړنی کتاب وي، چې د دې څانګې مینه والو ته په پښتو ژبه وړاندې کېږي. د «کمپیوټري ژبې» کتاب له دوو کتابونو څخه را ژباړل شوې مقالې دي: لومړی، «د کمپیوټرپوهنې او ټکنالوژۍ پوهنغونډ» چې د کمپیوټرپوهنې د نړۍ تکره او د ډېرو اثارو څښتن لیکوال «هېري هېنډرسن» لیکلی او که څوک غواړي چې د کمپیوټرپوهنې له هر کونج څخه ځان خبر کړي او په دې ناپایه کاینات کې قدمونه ووهي، نو زما تر سترګو له دې نه زیات علمي، مستدل او 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 ګټور اثر ګڼلی شو.

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

---

<sup>1</sup> Programming Language



د کتاب د ژباړې په هکله به هم په دې ځای کې ځينې ټکو ته اشاره بڼه وي. دا چې په دې برخه کې تر دې دمخه هېڅ کتاب نه وو ژباړل شوی (يا زما ترسترگو نه دی تېر شوی)؛ او د کمپیوټرپوهنې د اصطلاحاتو لپاره پښتو بدیلونه نه لرو؛ او که يې لرو، نو هغه هم یوازې د معارف په کتابونو کې څه نا څه لیدل کېږي او بیا د کمپیوټري ژبو موضوع په هکله خو دا برخه له نشت سره مخ ده. له همدې امله، د کتاب د ژباړې په اوږدو کې تل دوه وېرې راسره وې: یوه وېره دا وه چې که چېرته د کمپیوټرپوهنې اصطلاحات را ونه ژباړم، نو عادي لوستونکي به خندی کوي چې دا نو څنگه ژباړه ده؟ یوازې انگلیسي متن په پښتو تورو لیکل شوی. او بل خوا که د ژباړې هڅه يې 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 يې تر انگلیسي 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 يې 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 معیارونه هم پکې مراعت شوي وي، هغه به وژباړم؛ او کوم چې پاتې شوو، په هغه برخه کې به د دې علم له لوستونکو، زده کوونکو، زده کړیالانو، ښوونکو او ماهرینو نه هیله وکړم چې وړ او معیاري بدیلونه ورته ومومي او د یاد علم په بډاینه کې چې د یوه تن د وس کار نه دی، ونډه واخلي. د کتاب په دریم مخ کې مې خپله اړیکه او برېښنالیک همدې ته په کتو زیات کړي.

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

ستاسو

رحمت شاه فراز

# د «کمپیوټر اساسات»

له کتاب څخه

لیکوال: پي.کي. سینا

ژباړه: رحمت شاه فراز

## کمپیوټري ژبې

### له طبیعي ژبو سره یې ورته والی

ژبه د پوهېدلو او پوهولو (افهام او تفهیم) یو سیستم دی. په خپلو طبیعي ژبو لکه په پښتو کې نور وګړي په خپلو احساساتو او نظرياتو 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. موږ په عامه توګه د هغې ځانګړې ژبې د سمبولونو ټولګې ته وېبپانګه وایو. د مثال په ډول، هغه کلمات چې موږ یې په انګلیسي کې کاروو، د انګلیسي ژبې سمبولونه دي چې د دې ژبې وېبپانګه جوړوي. هره کلمه ځانګړې معنا لري چې په سیند (قاموس) کې موندل کېدای شي. په ورته ډول، د کمپیوټر ټولې ژبې خپله ځانګړې وېبپانګه لري. د دې وېبپانګې 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>، ریاضيکي عملیات، د مرکزي پروسېس کوونکي ترمنځ د معلوماتو د حرکت عملیات، او منطقي یا مقایسوي عملیات <sup>4</sup>.

هره طبیعي ژبه د خپلو سمبولونو د کارونې لپاره یو سیسټماتیک میتود لري. په انګلیسي کې، دغه میتود د ګرامري اصولو په بڼه وړاندې کېږي. په اصل کې همدا اصول موږ ته وایي چې کوم کلمات وکاروو او په څه ډول یې وکاروو. په ورته ډول، د یوې ځانګړې کمپیوټري ژبې سمبولونه هم باید د خپل ټاکلو اصولو له مخې چې د نحوي ترکیب <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 چیرته 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 لیکنېې 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 باینري 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 طبیعت او تحول وڅېړو.

#### ماشینواله 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، ماشینواله ژبه یا د کمپیوټر ماشینواله کوډ بلل کېږي. ماشینواله کوډ د یوه کمپیوټر بنیادي ژبه ده او عموماً د باینري متکړینو (صفر او یو) په شکل لیکل کېږي. د کمپیوټر سرکټ په داسې ډول اوږدل شوی چې سمدلاسه ماشینواله 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 ماشینواله ژبه کې لیکل شوی وي، دوه برخیز فارمټ<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 بښي 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 کوونکي 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 ګډون لري. په عادي عملیو کې لوستل<sup>3</sup>، جمع کول، تفریق کول، لیکل<sup>4</sup> او داسې نور راځي.

| OPCODE        | OPERAND    |
|---------------|------------|
| (عملیاتي کوډ) | (ادرس/ځای) |

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

- 
- 1 Manipulation
  - 2 Data Fields
  - 3 read
  - 4 Write



101100111111101001101100

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

0100000000000110011101

0110001100000010000000

0110000000011100101110

1010001111101110010111

0000000000000000000000

ښکاره ده چې دا ژبه دومره اسانه نه ده چې زده شي، یو څه له دې امله چې  
لوستل او پرې پوهېدل ستونزمن دي او یو څه سبب یې دا دی چې دا د اعدادو  
په هغه سیستم کې لیکل شوې چې موږ 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 ورکړیز واحد<sup>1</sup> ته وسپارل چې بایزري حالت ته یې واړوي. په حقیقت کې، له ډېرو هڅو پرته، یو کمپیوټر داسې اوږدل 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 پروګرام لاندینی بڼه خپلوي:

10001471

14002041

30003456

50773456

00000000

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

---

1 Input Unit

2 Instruction code

## د ماشینواله ژبې کټي 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 2 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 اوږدوالي<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. د 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 پېچلتيا:** د ماشينواله ژبې پروگرامونو اصلاح<sup>1</sup> او يا تغير پکې راوستل سخت وي. د تېروتنو د موندلو په موخه د ماشينواله لارښوونو کتنه هماغومره ستونزمنه وي خومره چې يې په پيل کې ليکل سخت وي. په همدې ډول، يو څه وخت وروسته د ماشينواله ژبې په پروگرام کې بدلون راوستل هم ستونزمن وي چې په دې ځای کې به ډېری پروگرام ليکونکي دا ښه وېلي چې په زاړه پروگرام کې د اړينو بدلونونو پر ځای، له پيل څخه نوی منطق کوډ کړي. په لنډو ټکو کې ويلای شو چې په ماشينواله ژبه کې د يوه پروگرام ليکل 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 حروفي سمبولونه – نیمونیکس – رامنځته شوو. یو نیمونیک (یا د زېرمې مرسته) هر هغه ذهني چل او طریقي ته ویل کېږي چې له موږ 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 کټور 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 «تقریق» 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 ويني، 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 پروګرام د ژباړلو لپاره روزلی شو چې د کمپیوټر یو پروګرام په ماشینواله ژبه کې د عددونو پر ځای د سمبولونو په کارولو ولیکل شي. بیا موږ کولای شو چې د کمپیوټر لپاره د عددونو پر ځای د سمبولونو په کارولو یو پروګرام ولیکو، او کمپیوټر دې ته چمتو کړو چې پخپله یې ژباړه وکړای شي. دا د پروګرام لیکونکي لپاره اسانتیا رامنځته کوي، ځکه نوموړی کولای شي چې د خپل پروګرام د لیکلو لپاره د عددونو پر ځای له نیمونیکونو، سمبولونو او تورو څخه ګټه پورته کړي. د مثال په ډول، پورتنی پروګرام چې په ماشینواله ژبه کې لیکل شوی وو داسې چې دوه عددونه جمع کوي او بیا یې د جمعي حاصل پرېنت کوي، کولای شو په لاندې ډول ولیکو:

CLA A

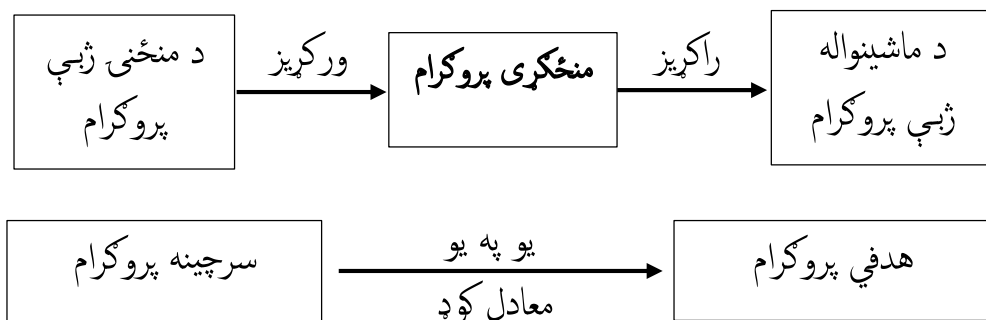
ADD B

STA C

TYP C

HLT

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



په دې ځای کې باید ځینې نور اصطلاحات هم زده کړو. هغه ژبه چې د ماشینواله ژبې په 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 راجسټر له موادو سره جمع کېږي.             |
| SUB     | 15          | د یوه رجسټر له موادو څخه تفریق کوي.         |
| STA     | 30          | یو رجسټر ذخیره کوي.                         |

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

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

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

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

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

14، STA په 30 او HLT په 00 ترجمه<sup>1</sup> کړي، او په دې ډول به د ماشینواله ژبې پروګرام 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، موږ کولای شو چې په کمپیوټر باندې تر دې هم د کار برخه ور زیاته کړو.

---

<sup>1</sup> Interpret

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

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

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

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

FRST = 1000

SCND = 1001

ANSR = 1002

له دې وروسته به یوازې په خپلو نومونه ور غږ کوو. منځګړی پروګرام به په جدول کې هغه نوم ګوري او دقیق ادرس به راته چمتو کوي او په دې ډول به مو له ډېر کار څخه ځان ساتلی وي. دا په دې معنا چې 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 منځګړی پروګرام د هر عدد لپاره ادرس لټوي، هغه نیمونیک عملیاتي کوډ، عددي عملیاتي کوډ ته ترجمه کوي او په پای کې د ماشینواله ژبې یو پروګرام راته چمتو کوي. د مثال په ډول، په لومړنۍ لارښوونه کې CLA FRST، منځګړی پروګرام CLA د 10 عملیاتي کوډ ته ژباړي او بیا په جدول کې د FRST ادرس لټوي، منځګړی دا ادرس د 1000 په ډول مومي، او په دې ډول د ماشینواله ژبې لومړنۍ 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 ماشینواله ژبې پروګرام چمتو کوي.

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

START PROGRAM AT 0000 AND START DATA AT 1000

SET ASIDE AN ADDRESS FOR SCND

SET ASIDE AND ADDRESS FOR ANSR

CLA FRST

ADD SCND

STA ANSR

HLT



## د پروگرام پښتو ژباړه:

پروگرام په 0000 پیل کړه او دېتا 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. موږ 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 موږ په څه ډول غواړو چې زموږ پروگرام اجرا شي.

---

<sup>1</sup> Pseudo Instructions

د دې لپاره چې وگورو منځګړی دا سمبولیک پروګرام څنګه ماشینواله ژبې ته ژباړي، راځئ چې پړاو په پړاو یې نندارې ته کښو.

لومړنی پړاو منځګړي ته وایي چې پروګرام 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 پړاو منځګړي ته وایي چې د FRST لپاره یو ادرس پیل کړي. په دې ډول منځګړی پروګرام 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 موږ هغه ادرس ته د FRST په سمبول اشاره کوو، نو موږ په دې اړه هېڅ اندېښنه نه کوو چې هغه ادرس په دقیق ډول څه شی دی، هغه که 1000 وي یا 1001 یا 4253 وي، هېڅ توپیر نه کوي، ځکه چې منځګړي په خپله حافظه کې زېرمه کړی دی.

بل پړاو منځګړي ته وایي 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 منځګړی هر هغه ځل چې د SCND سمبول کاروو، د 1001 ادرس برابروي.

په همدې ډول، راتلونکي پړاو منځګړي ته وايي چې د ANSR لپاره یو ادرس بېل کړي، چې دا ځل 1002 دی. دا وروستی جعلي لارښوونه ده، او راتلونکي لارښوونه په خپله اصلي پروګرام پیلوي.

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

## پر ماشینواله ژبه د منځنۍ ژبې ګټې

منځنۍ ژبه پر ماشینواله ژبه لاندینۍ ګټې لري:

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

---

<sup>1</sup> Comments

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

**۲. د تېروتنو موندلو او سمولو اسانتیا:** کله چې په منځنۍ ژبه کې پروگرام لیکل کېږي، د تېروتنه شمېر کم وي، او کومې تېروتنې چې شوې وي د هغو موندل او سمول د نیمکونیکونه او ساحو لپاره د سمبولیکو نومونو له کبله اسانه وي. سربېره پر دې، منځګړي پروگرامونه داسې ډیزاین شوي وي چې په خپلکاره<sup>1</sup> ډول تېروتنې په ګوته کوي. که چېرته موږ داسې نیمونیک یا نوم وکاروو چې هېڅکله نه وي تعریف 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 یوه لارښوونه ADD AREA لوستل کېږي، او له موږ نه دا هېر شوي وي چې AREA تعریف کړو، نو منځګړی پروگرام به په خپل جدول کې په AREA پسې سترګې وغړوي؛ نو که یې ونه مومي، تېروتنه به را برسېره کړي.

**۳. د بدلون راوستو اسانتیا:** د پروگرام لیکونکو لپاره د ماشینواله 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 ایستلو لپاره د دې اړتیا 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 ګڼو څخه دا

هم ده چې د ډېټا او لارښوونو په ادرسونو پورې تړلې اندېښنې له منځه وړي. دا تر هغه ډېر مهم دی څومره چې په لومړني نظر ښکاري. فرض کړئ چې موږ یو اوږد ماشینواله 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 موږ هغه لارښوونه درج کوو، نو باید ټولې لارښوونې مو په یادو زده وي، او ټول پروگرام ټکي په ټکي 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 موږ همدا پروگرام په سمبولیکه ژبه ولیکو، نو موږ 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 موږ ناڅاپه پوهېږو چې موږ 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 پروګرام يو ځل بيا ځغلوو. د دې لارښوونو معادل ماشينواله پروګرام به خپل پروګرام د 1000 په ځای له 3000 څخه پيل کړي، او په دې ډول به له بل پروګرام سره هېڅ ټکر نه خوري. په بل عبارت، سمبولیکه ژبه په ډېرې اسانۍ پروګرامونه د حافظې له يوې برخې څخه بلې هغې ته لېږدولی شي؛ په دې اساس موږ ويلای شو چې د منځنۍ ژبې پروګرامونه په اسانۍ د انتقال وړ دي ځکه چې د دوی موقعيت يوازې د لومړنۍ لارښوونې په بدلون، په اسانه بدليږي. دا کار په ماشينواله پروګرامليکنه کې دومره په اسانه سرته نه رسېږي.

**٦. د ماشينواله ژبې اغېزمنتيا:** د هغو ګټو ترڅنګ چې پورته يادې شوې، د منځنۍ ژبې پروګرام د خپل معادل ماشيني کوډ له اغېزمنتيا او ګټو 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 کوونکي 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 ماشین-اړونده دي، نو پروگرام ليکونکی اړ دی چې د پروگرام د ليکلو پر مهال، د يوه ځانگړي ماشین له ځانگړنو او شرايطو څخه خبر وي. په منځنۍ ژبه کې پروگرام ليکونکی بايد پوه وي چې د هغه ماشین 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 ماشینواله کچې کوډ<sup>1</sup> ليکل:** په منځنۍ ژبه کې هم لارښوونې د ماشینواله کچې په کوډ ليکل کېږي – په دې معنا چې د ماشینواله کوډ يوه لارښوونه خپل ځای د منځنۍ ژبې يوې لارښوونې ته پرېږدي.

ماشینواله او منځنۍ ژبې چې ماشین-اړونده ژبې دي، د ټيټو 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 پرېنت 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 پرېنت 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 پرېنت 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 لیکونکي 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

موږ 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 ليکونکي 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 رجسترونه اغېزمنوي. که څه هم، موږ پورته ولوستل چې د دې لپاره چې يو ښه کمپيوټري پروگرام وليکل شي، نو پروگرام ليکونکي 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 ليکونکي د دې جوگه گرځوي چې د انګليسي کلماتو او د عادي رياضيکي سمبولونو په کارولو خپلې لارښوونې وليکي. په دې ډول د نوموړي لپاره دا اسانه کېږي چې د پروگرامليکني 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 نیممونیکو 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 کټورې پرېوځي. په دې اساس هره ژبه د ستونزو د يوې ځانگړې ټولگې د حل کولو لپاره کټوره 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

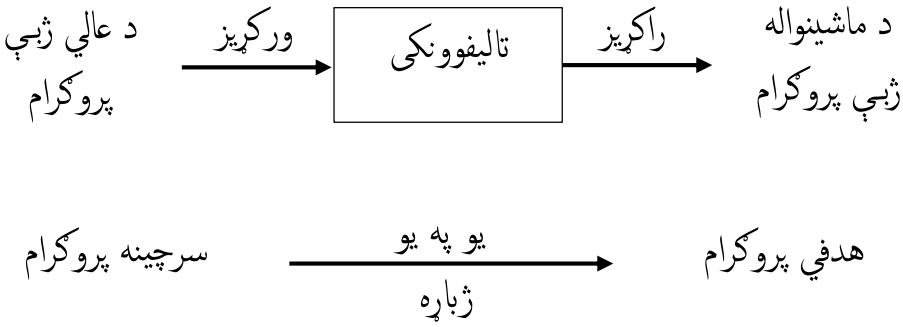
## تالیفونکي<sup>1</sup>

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. موږ ولیدل چې منځنۍ ژبې د اوږون 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 هدفی 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 هدفی پروګرام دی چې د ماشینواله ژبو له لارښوونو څخه جوړ وي. پام مو وي چې سرچینه پروګرام او هدفی پروګرام دواړه یو دي، مګر د تحول په بېلابېلو پړاوونو کې قرار لري.

---

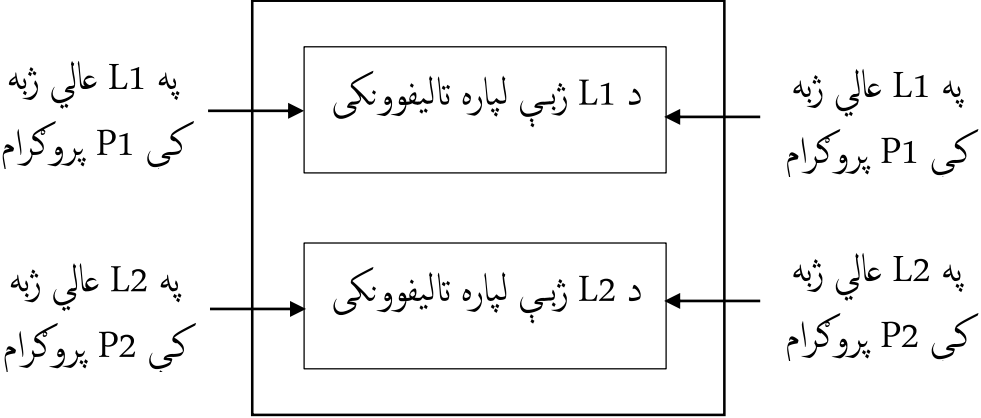
<sup>1</sup> Compilers

<sup>2</sup> Conversion process

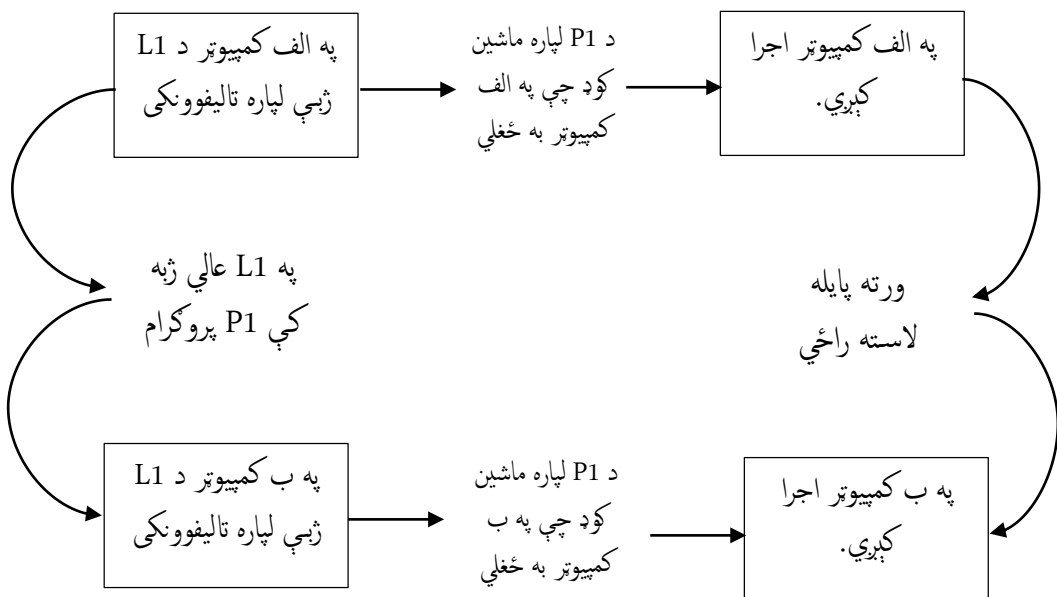


پورتنی انځور د تالیفونکي د ژباړې پروسه تشریح کوي.

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 هدفی پروګرام له هغه هدفی 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 تر لاسه 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 پروسېس 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 برسېږه کوي. دغه تېروتنو ته تشخیصي تېروتنې (diagnostic erros) 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 پروګرام کې د لارښوونو نامناسب اوډون (ترتیب)<sup>1</sup>

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

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

---

<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 کمپيوټرونو (کوچنيو کمپيوټرونو) سره کېد رايي. پر تاليفوونکي باندې د ترجمان کټه په سرچينه پروگرام کې د بدلونونو پر وړاندې چټک غبرگون ښودل دي. ترجمان هغه اړتيا له منځه وړي چې هر ځل چې د پروگرام د ځانگړنو زياتولو يا تېروتنو سمولو لپاره تغيرات راوستل کېږي، نو لومړی بايد تاليف شي. د دې ترڅنگ، تاليفوونکی د يوه ترجمان په پرتله يو پېچلی پروگرام دی. ترجمان پروگرامونه په اسانه ليکل کېږي او په کمپيوټر کې د لويې حافظې شتوالي ته هم اړتيا نه لري. خو بل خوا، ترجمان د ژباړې يو وخت نيوونکی ميتود دی، په دې چې هر عبارت، هر ځل چې له سرچينه پروگرام څخه اجرا کېږي، بايد وژباړل شي. په دې توگه، تاليف شوی ماشينواله پروگرام د ترجمه شوي پروگرام په پرتله څو چنده چټک ځغلي.

منځکړي، تاليفوونکي او ژباړونکي هغه سيستمي پوستغالي دي چې د کټه اخيستونکي له لوري ليکل شوی سرچينه پروگرام، يوه داسې هدفې پروگرام ته ژباړي چې د کمپيوټر د سختغالي لپاره معنا ولري. دې ژباړونکو ته ژبني پروسېس کوونکي<sup>2</sup> هم ويل کېږي، ځکه چې دوی يوه ځانگړې ژبه پروسېس کوي.

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

د پورتنیو خبرو پر بنسټ، موږ دې نتیجې ته رسېږو چې په عموم کې، یوه پروګرامي ژبه باید له لاندې ځانګړنو څخه برخمنه وي 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 ورځنۍ 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 کمپنۍ چې خپل کمپیوټرونه تغیروي – یا حتی یوه پېل تولیدوونکي ته انتقال کېږي، نو هېڅ اړتیا نشته چې ټول هغه پروگرامونه بیا حلې ولیکل شي چې همدا اوس تر کار لاندې دي. حتی هغه پروگرامونه چې په عالي ژبو کې لیکل شوي وي، یوازې ځینو 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 لیکل شوی وي په څو مختلفو کمپیوټرونو کې په ډېر لږ زحمت او یا په عملي کچه له هېڅ ډول زحمت پرته هم خغلول کېدای شي.

۲. **اسانه زده کړه او کارونه:** دغه ژبې له هغو ژبو سره ډېر زیات نږدېوالی لري چې موږ یې په خپل ورځني ژوند کې کاروو. په دې اساس، د دې ژبو زده کړه او کارونه اسانه ده. پروگرام لیکونکي اړتیا نه لري چې د خپل کمپیوټر په هکله په هر څه پوه شي. دی اړتیا نه لري چې په دې اړه اندېښنه وکړي چې عددونه په څه ډول په کمپیوټر کې ذخیره کړي، چېرته یې ذخیره کړي، څه ورسره وکړي، او داسې نور. په دې معنا چې پروگرام لیکونکي 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 یو څه مطلوبه وي ځکه چې پروگرام لیکونکي ته اجازه ورکوي چې له سیستم څخه په ډېر اغېزمن ډول گټه پورته کړي.

**۳. کمې ټېروتنې:** 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 ټېروتنو کولو امکان پکې ډېر کم دی. کمپیوټر په خپله وړو مسایلو ته اوږه ورکوي، او تر هغو په خپل سر یوه ټېروتنه نه زیږوي 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 ټېروتنې په نښه کولای او نیولای شي. په دې اساس، که پروگرام یوه نیمه تشخیصي ټېروتنه ولري، نو پروگرام لیکونکی یې په اسانه ډول تشخیص کولای او سمولای شي.

**۴. ټیټ لګښت:** په عالي ژبو کې د پروگرامونو لیکل لږ وخت او کم زحمت غواړي، چې په طبیعي ډول د پروگراملیکني ټیټ لګښت ته لاره برابروي. عموماً، د پروگرام د چمتو کولو د ټولو فازونو (کوډ کول، ډي بګ کول، ازمايښت کول، او داسې نورو) لګښت 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 لارښودليکني لپاره، ډېر کم او يا په عملي ډگر کې له نشت سره برابر، نظر عبارتونه<sup>1</sup> پکار وي.

**۶. اسانه بدلون:** په عالي ژبه کې ليکل شوي پروگرامونه د منځنۍ او ماشينواله 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 ترتيب د تېروتنو موندل او سمول او په لارښوونو کې بدلونونه راوستل هم اسانه وي. له يوه پروگرام څخه د ځينو لارښوونو درج يا ايستنه له کوم تاليف پرته شونتيا مومي. په دې اساس، لوی بدلونونه په ډېر کم زحمت د پلي کېدو وړ وي.

## د عالي ژبو محدوديتونه

د عالي ژبو دوه زيانونه په لاندې ډول دي:

**۱. داغېزمنتيا کمښت:** عموماً، هغه پروگرام چې په منځنۍ يا ماشينواله ژبه کې ليکل 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 اړتيا لري.

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

---

<sup>1</sup> comment statements



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

---

<sup>1</sup> Mathematic

# «د کمپیوټر پوهنې او ټکنالوژۍ له پوهنځونې»

## 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 پېرېوته. 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 ګوډامونو 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>۱</sup> incompatible<sup>۲</sup> general-purpose<sup>۳</sup> scanners

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

## د ژبې ځانګړنې

د اېډا په ډیزاین کولو کې، د ژبې جوړوونکو د دودیزو اصولو پر اساس د ژبې هغه بنيادي 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.

اېډا (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 رانغاړي چې هغو کلاسونو ته ورته دي چې وروسته په سي پلس پلس او نورو ژبو کې وکارول شوو. څنګه چې په دې ساده مثال کې لیدل کېږي، د اېډا 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 وییکی<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 پروګرام لومړنی کرښه دا مشخصوي چې په پروګرام کې به کوم پېکجونه کارول کېږي. پېکجونه هغه جوړښتونه دي چې له ډېټا ډولونو او اړونده فنکشنونو څخه جوړ وي؛ د مثال په ډول هغه فنکشنونه چې د متن د 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 ځانګړي پېکجونه هم مشتق کېدای شي.

په اصلي پروګرام کې **Begin** د ډېټا اصلي پروسېس پیلوي چې په دې ځای کې د Put فنکشن په مرسته Ada.Text.IO فنکشن یو پیغام څرګندوي او د Get\_Line د فنکشن په وسیله د ګټه 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

اېډا د لويو او پېچلو سافټوېري پروژو لپاره يوه مناسبه ژبه ده، ځکه چې د پېکچونو کارول په يوه ډېټا ډول باندې د کار کولو او پلي کولو تفصيلات له نورو ګټه اخيستونکو څخه پټوي او ساتنه يې کوي. هغه پروګرام ليکونکي چې په پروګرام کې يې يو داسې پېکج کارول شوی وي چې دا مشخصه کړي چې کوم پارامترونه<sup>1</sup> بايد له هر فنکشن سره وکارول شي، نو په دې توګه نوموړی پروګرام ليکونکي له ښکارېدونې اړيکې<sup>2</sup> څخه ګټه نه شي پورته کولای. د اېډا تاليفوونکي په ډېر غوره ډول قانون بندي شوي<sup>3</sup> تاليفوونکي دي خو د ډېټا د بېلا بېلو ډولونو د پروسېس لپاره له ټاکلو ځانګړنو سره سمون خوري. همدا رنگه، ياده ژبه پياوړې ليکنې<sup>4</sup> ژبه ده، په دې معنا چې هر ډېټا ډول بايد په مشخص ډول تعريف شي، د سي ژبې پر خلاف، چې کله ډېټا ډولونه په خپلواک ډول له يو ډول څخه بل ډول ته اوړي تر څو ترمنځ يې همغږي را پيدا شي، نو کوچنۍ او ګونګې تېروتنې ترې پيدا کېږي.

له دې امله چې دا ژبه په مختلطو<sup>5</sup> سيستمونو او د رښتين-مهاله<sup>6</sup> فعاليتونو لپاره کارېږي، په دې اساس اېډا کې ګڼ شمېر داسې ځانګړنې ډيزاين شوې خو ماشينواله کوډ په اغېزمن ډول ورباندې وليکل 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



ترڅنګ په منځنيو او نورو عالي ژبو کې د روتینونو<sup>1</sup> د لیکلو لپاره هم اسانتیاوې وړاندې کوي. د دې ژبې تازه بڼه، اېډا-۹۵، د موازي پروګرام جوړونې<sup>2</sup> په ملاتړ هم ټینګار کوي. که څه هم د اېډا راتلونکي ناڅرګند دي، ځکه چې د دفاع وزارت نور دې ته اړتیا نه لري چې یاده ژبه په دولتي تړونونو کې وکاروي. خو په ځانګړې ډول د پراخو استازواله ژبو<sup>3</sup> د ځانګړنو په ګډون چې د ګنشمېر توارث<sup>4</sup> د اړیکځایونو لپاره ملاتړ چمتو کوي، د متنګرښو<sup>5</sup> د غوره سمبالښت او نورو ډېټا ډولونو او فایلونو په برخه کې د اېډا پرمختګ تر اوسه پورې دوام لري؛ او دغه راز د رښتین-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 ټولګه چې د یوه ځانګړي کار د ترسره کولو لپاره ډیزاین شوې 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، مخکېنۍ پروګرام لیکونکو د یوې داسې پروګرامي ژبې په جوړولو لاس پورې کړ چې یوازې د یوه ځانګړي کار او یا هارډویري دریځ لپاره نه وي ډیزاین شوې. له ۱۹۵۷ز نه مخکې، د جرمني <sup>1</sup>GAMM او د امریکې <sup>2</sup>ACM یا د کمپیوټري ماشینۍ 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 یا الګول-۵۸ په بڼه مخې ته راغله او الګول-۶۰ د دې ژبې هغه نسخه شوه چې تر ټولو لومړی په لویه کچه استعمال او تطبیق<sup>۱</sup> کړای شوه.

## د ژبې ځانګړنې

الګول یوه بلاک-جوړښته<sup>۲</sup> او پروسوي<sup>۳</sup> ژبه ده. په نوموړې ژبه کې هر متحول<sup>۴</sup> د ډېټا د یو کوچني شمېر ډولونو پورې د اړوند کېدو لپاره تعریفېږي د بېلګې په ډول کامل اعداد<sup>۵</sup>، حقیقي اعداد او یا له دواړو څخه د یو ډول اعدادو لړۍ. په داسې حال کې چې د ډېټا ډولونو شمېر پکې محدود دی او دا چې د نوو ډولونو د تعریفولو لپاره اصلاً اسانتیا پکې نشته، نو د تالیفونکي ډول کتنه<sup>۶</sup> (دا باوري کول چې د ډېټا توک<sup>۷</sup> د تعریف شوي متحول له ډول سره سمون خوري او که نه) تر یوې کچې داسې خونديتوب چمتو کاوه چې په پخوانیو ژبو کې نه تر سترګو کېده.

د الګول یو پروګرام په لوی شمېر بېلا بېلې پروسې<sup>۸</sup> لرلای شي او یا هم په خارج کې له تعریف 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 وېولي او يا يې کال<sup>1</sup> کړي. نوموړې ژبه له معياري کنټرول جوړښتونو څخه هم برخمنه ده.

لاندینی 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 پایله پر صفحه څرګندوي:

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

---

<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>۱</sup> او سم/ناسم وېلیوګانو<sup>۲</sup> په ګډون) د ډېټا ډولونو شمېر ډېر کړای شوی او د ګټه اخیستونکي له خوا تعریف شوي ډولونه او سټرکټونه<sup>۳</sup> هم ور زیات شوي. سټرکټونه هغه ریکارډونه دي چې د ډېټا د مختلفو ډولونو لپاره خپلې ساحې لري. نښه کوونکي<sup>۴</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>۵</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>۶</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 ښکاره او منظم ډول څرگندې کړای شي. د نوموړې ژبې په غونډ<sup>3</sup> شکل د رياضيکي عمليو د وړاندې کولو ځواک د ډېری 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 بياتر تیتول 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 توګه، د اې پي اېل د کوډ لاندې کرښه:

$$X [\Delta X]$$

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

بل خوا، اې پي اېل د زياتره پروگرام ليکونکو لپاره ګڼې نيمګړتياوې او عيونه 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»، «کرنیکان» 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 ګڼ بدیلونه لیکل شوي دي.

د نورو متنواله ژبو په څېر، د آک یو پروګرام د هغو کمانډونو په یوه لیکلې 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 پروګرام 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 څو 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 نسبي 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)

## بېسیک (BASIC)

بېسیک یا «د مبتدي ټول-هدفه سېمولیک لارښود کوډ»<sup>1</sup> ژبه په ډارټماؤت کالج کې په ۱۹۶۴ کې د «جې. کېمینی» او «ټي. کرتز» له خوا جوړه شوه. هغه مهال د کالج ټول کېمپس له ترمینلونو سره تړل شوو په مهال-شریکه<sup>2</sup> کمپیوټرونو باندې سمبال وو چې دا کار په داسې وخت کې یو لوی نوښت وو چې ډېری کمپیوټرونه به یوازې له یوه ځای څخه د پنچکارتونو په وسیله پروگرام کېدل. جان. جې. کېمینی او تامس کرتز غوښتل چې یوه داسې ژبه جوړه کړي چې په اسانۍ سره زده او وکولای شي چې د کیبورډ له لارې لیکل شوي اوامر سمډلاسه تالیف او ځواب ورته ووايي. او په دې ډول د خپلو سیستمونو له تعاملې<sup>3</sup> ځانګړنې څخه ګټه پورته کړي. د دوی دا فکر د هغه مهال له ژبو لکه کوبال، الګول او فورټران سره له یوې مخې په تضاد کې وو، چې په دې ژبو کې به پروګرامونه لومړی په بشپړ ډول لیکل کېدل او بیا به یې 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-ویوې 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 بایت څخه تر ۶۴ کیلو بایت 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 پرمختللې نسخې (د تالیفونکو په ګډون) په ټېپ<sup>2</sup> کې زېرمه کېدای 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 ځانګړي دریځ<sup>2</sup> لپاره د کارول کېدو وړ وګرځول.

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

## تنقید او لرلید

د بېسک زیاتره نسخو له کرښه-ګڼو<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 پروګرام کنټرول یوې ځانګړې کرښې ته ټوپ کړي، د goto عبارت هم پکې کارول کېدای شوی. که څه هم په نوموړې ژبه کې ساده فرعي پروګرامونه ول (چې د gosub عبارت په مټ ورته لاسرسی کېده)، خو بیا هم د پاسکال او سي ژبو په څېر یوه فنکشن ته د متحول د ور تېرولو له وړتیا څخه بې برخې

---

<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 وېلیوکانې تغیر 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 پروګراملیکني د ښوونې او سي ژبې ته یې د سیستمي پروګراملیکني 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 کمپنۍ د سیستمي پروګراملیکني 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 پېکج ته د لاسرسي له اسانتیاوو څخه ترکیب وو. دغو پېکجونو مېنوگانې<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 پروګراملیکني تعاملې تګلاره<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 Petrussha. *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>۱</sup> او بي.<sup>۲</sup> ژبو ايښودل شوی وو. سي د لومړي ځل لپاره په DEC PDP-11 کمپيوټرونو کې وکارول شوه چې په همدې وروستيو وختونو کې جوړ شوی يونيکس عامل سيستم باندې ځغلېده، په دې ځای کې سي ژبې د 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 بچونۍ سي پلس پلس، تر ټولو زياتې کارېدونکې پروګرامي ژبې ګرځېدلې دي.

## د ژبې ځانګړنې

د لومړنۍ الګول او تر يو څه بريده پاسکال په څېر، سي يوه پروسوي ژبه<sup>1</sup> ده چې د هغه پروګرامليکنې فلسفه انعکاس کوي چې د ۱۹۷۰ کلونو په اوږدو کې يې ورو په ورو بڼه وموندله. په عمومي ډول، د سي ژبې تګلاره داسې بيانېدای شي چې د کمپيوټر د رېسټيني فطرت لپاره اړينې ځانګړتياوې 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 `double` او `float` 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 متنګرښه 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 پېښه شي، خو په اصل کې دا ځانګړنه د سي ژبې هغه ټينګار 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. آرګومېنټونه، فنکشنونو ته د وېلیوګانو په وسیله ور تېرېږي، او د نښه کوونکي په کارولو کولای شو چې د ریفرنس له لارې یې هم ور تېر کړو.

## د پروګرام بېلګه

لاندې د یوه لنډ پروګرام بېلګه وړاندې شوې:

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

<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 پروګرام 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 پروګرام 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 سټانډر لایبرېري کې پراته وي. (د دې ترتیب موخه دا ده چې پخپله ژبه ساده او سپکه شي، او که چېرته نوموړی پروګرام یوه بل دریځ 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 پرنت کوي) او `scanf()` (چې په پروگرام دېټا 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 دېټا 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 دېټا د دې پر ځای چې له بلونکي عبارت څخه یې واخلي، د ګټه 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 پرنت کړي چې د هغه فنکشن له خوا واپس شوی وي. (د نورو عبارتونو دننه یوه فنکشن ته بلنه ورکول د سي د غښتلي نحوي ترکیب یوه بېلگه ده.)

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

یو پروګرام لیکونکی کولای شي چې د یوه 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 پروګرام 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 اجرايي پروګرام فایل a.out ترې چمتو کېږي. له هغه وروسته که همدا نوم په کمانډ پرېمپټ کې ولیکل شي نو پروګرام ځغلي، چې لومړی اعداد اخلي او ورپسې یې اوسط راوباسي.

```
% gcc test.c
```

```
% a.out
```

```
5
```

```
7
```

```
9
```

```
.
```

```
The average is: 7.000000
```

**بریا او بدلون**

له رامنځته کېدو درې 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 وځغولي، سي ژبه د مايکروسافټ ډاس، وينډوز او مېکنتاش پروگرامونو د ليکلو لپاره د خوښې وړ انتخاب وگرځېد. د بېلګې په ډول، د وينډوز لپاره د کاريال پروگرامي اړيکځای يا  $API^1$  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 ليکني<sup>3</sup> لپاره يو څوک هڅولای شي، د سي اصلي بڼه (چې کرنېګن او ريچي په The C Programming Language کتاب کې تعريف کړې) پارامترونه نه ګوري 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 پروګرام کې د لوی شمېر داسې تېروتنو لامل ګرځي چې موندل يې ستونزمن وي. خو د امريکې د ملي معيارونو ادارې له خوا معياري سي ژبه چې په پېچلو شرطونو ولاړه ده، په تاليفوونکي کې د ډول-کتنې د ځانګړنې 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 توګه يوه استازواله ژبه وګرځېده. په داسې حال کې چې سي ژبې خپل ځای د نوو پروګرامونو د جوړولو لپاره سي پلس پلس او جاوا ته پرې ايښی، خو د سي پروګرام ليکونکو لپاره د نوو ژبو زده کړه نسبتاً اسانه وي او همدا رنگه د سي ژبې د ګوډ 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 مايکروسافټ 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 ټايپونه 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 اوړېدنو<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 ګنشمېر 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 ګڼشمېر اړیکځایونو<sup>1</sup> په تعریفولو او یا یوه کلاس ته د لاسرسي د بېلابېلو طریقو په مټ کولای شو چې دا کار ترسره کړو.)

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

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

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

---

<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. لومړنۍ 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. پورتنۍ پروګرام یوازې یو کلاس (Hello) لري، چې په هغه کې یو مېن فنکشن دی چې هر پروګرام باید یو دانه ولري او یوازې یو. دا فنکشن د System نومي کلاس، Console غړی رابولي، او له هغه وروسته د متن د ښودلو لپاره له WriteLine میتود څخه ګټه اخلي.

### سي پلس پلس او د مایکروسافټ پرمختګ

سي شارپ د سي پلس پلس، 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



مایکروسافټ 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 ابتدایي ډول یاده ژبه د مایکروسافټ په سافټوېرونو 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 بېخبنسټ<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 هیل، نیو جرسي کې په AT&T لابراتوار کې د بیارنې سټراؤسټرپ<sup>1</sup> له خوا ډیزاین شوه چې په ۱۹۷۹ز کې یې کار ورباندې پیل کړی وو. تر دې مهاله سي ژبه د سیسټمي پروګراملیکنې لپاره د یوې ځواکمنې وسیلې په توګه کارول کېده. که څه هم سټراؤسټرپ (او نور) په دې باور وو چې د سي محدود ډېټا جوړښتونه او د فنکشنو میکانیزم په لویو سافټوېري پروژو کې چې د استازو<sup>2</sup> ترمنځ یې اړیکې پیچلې وي، د خپلمنځي اړیکو د څرګندوای لپاره د سي ژبه ناکافي وه.

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

---

<sup>1</sup> Bjarne Stroustrup

<sup>2</sup> Objects

<sup>3</sup> 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 ګډه 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، بیاکننه 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-تړلی لست<sup>1</sup>).

په داسې حال کې چې د استازواله تگلارې گڼو خپل لوري ته ور مایل کړی وو، سټراوسټرپ غوښتل چې په څرگند ډول د ماشین د چلند کنټرولولو ځانگړنه چې سي ژبې لرله، هغه په خپل ځای پاتې شي چې په سیستمواله پروگرام لیکنه کې پکارېږي. په دې اساس، نوموړي غوښتل چې یوه داسې ژبه جوړه کړي چې نحوي ترکیب یې سي ته ورته وي او د استازوالو ژبو ځانگړنې هم ورسره وي. سټراوسټرپ د انگلستان په کمبریج پوهنتون کې د دوکتورا د تیزس په توګه د سي پلس پلس لومړنۍ نسخه ولیکله 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 نوعیت استازي جوړ کړي. هر استازی د ډېټا یوه ټولګه 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 یو کتار تعریفوي

<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 کړو:

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

```

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

```

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

```

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

```

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

چې د غړو فنکشنونو لپاره د اړتیا وړ پارامترونه نه وي بدل شوي، هغه پروګرامونه چې د ډېری له استازو څخه ګټه اخلي، اړتیا به نه وي چې بدل شي.

د کلاسونو او توارث ترڅنګ، سي پلس پلس ځینې نورې مهمې ځانګړتیاوې هم لري. د فنکشن د پارامترونو لپاره ډېټا ډولونه په بشپړ ډول تعریفېدای شي، او د ډېټا ډولونه په خپلواک ډول کتل کېږي (که څه هم پروګرام لیکونکی کولای شي چې دا ډول کتنه ودروي که چېرته په 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 اشتقاقی 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. ټېمپلېټ هغه انتزاعي خصوصیت ده چې د هغو ډېټا ډولونو لپاره د کلاس تعریفونو د جوړولو لپاره کارېدای شي چې دې کلاس ته ور تېرېږي. په دې اساس، يوه لیست ټېمپلېټ ته يو وکتور او دوه-بعدي کتار ور تېرېدای شي او دا ټېمپلېټ به د هر ډېټا ډول لپاره د لیست کلاس تعریفونه جوړ کړي. ټېمپلېټونه 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 پروګراملیکني څخه نیولې بیا تر کاروباري کاریالونو او ګېمونو پورې د مختلف ډوله سافټوېرونو 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 نوښت 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. (آخر پوځ هم یو کاروبار وو، چې د کودامونو کنټرول او محاسبوي اړتیاوې یې له لویو شرکتونو څخه هم زیاتې او لوړې وې. هغه مهال، د مېنفرېم کمپیوټرونو لپاره اساسي کاروبارواله ژبه فلوېمټیک<sup>2</sup> وه، نوموړې ژبه د گریس 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 پروگرام 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» پورتنۍ 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 چاپېریال برخه<sup>2</sup> د چاپېریال (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 کارېدونکي ماشین مشخص کړي.

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

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

DATA DIVISION.

FILE SECTION.

FD INFILE

LABEL RECORDS ARE OMITTED.

01 STUDENT-DATA.

02 STUDENT-ID PIC 999999.

02 STUDENT-NAME.

03 LAST-NAME PIC X(15).

03 INITIAL PIC X.

03 FIRST-NAME PIC X(10).

02 GPA PIC 9.99

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

د پروسی<sup>1</sup> برخه هغه عبارتونه چمتو کوي چې پر دېټا اصلي عملیې ترسره کوي. پروسه کولای شو چې په فرعي پروگرامونو ترتیب کړو (چې تر یو څه بریده د نورو ژبو له پروسو او فنکشنونو سره ورته والی لري). د پروسی 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
```

ریاضیکي معادلات د compute عبارت په کارولو محاسبه کېدای شي:

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

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

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

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

## اغېز او لرلید

له ۱۹۶۰ څخه تر ۱۹۸۰ کلونو پورې، کوبال په مېنفرېم او منځنیو کمپیوټرونو کې د کاروباري پوستغالو د جوړولو لپاره یوه اساسي ژبه وګرځېده 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 ورته‌ره شوه.

د کوبال ځينو اوسنیو نسخو د جوړښت‌واله پروګراملیکنې وروستي پرمختګونه هم په کې ځان کې راټول کړي (لکه modularization) او حتی استازواله ډیزاین. کوبال د مایکروسافټ ډاټ نېټ او ورسره د XML ډېټا د پروسېس کولو په ګډون د جوړونې د اوسنیو چارچوکاټونو د خپلولو لپاره هم د پام وړ تغیرات له ځانه ښودلي. خو بیا هم، د کوبال استعمال ورځ تر بله د ځورتیا پر لور درومي، په دې چې پروګرام لیکونکو ورځ په ورځ خپل مخ د سي پلس پلس په څېر ژبو، متنواله ژبو، او د ډېټابېس جوړونې سیستمونو 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 مپیر او د هغه د کمپنۍ 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 پروګراملیکني 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 پروګرام پوهېدل اسانه کوي، باندې ټینګار کوي او هڅه کوي چې له ټیټې کچې او مبهم کوډ لکه د تالیفونکي آپټیمایزېشن څخه مخنیوی وکړي.

## د پروګرام جوړښت

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

**class**

*COUNTER*

**feature**—access counter value

*total*: **INTEGER**

**feature**—manipulate counter value

*increment is*—increase counter by one

**do**

*total* :- *total* + 1

**end**

*decrement is*—decrease counter by one

**do**

*total* := *total* - 1

**end**

*reset is*—reset counter to zero

**do**

*total* := 0

**end**

**end**

په دې لیستواله ژبه<sup>1</sup> کې، اړین-ویوې 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 تالیفونکی 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 تالیفونکو له وړاندې څخه د شتوالي کټه پورته کوي.

د آیفل یوه خاصه ځانګړنه دا ده چې قراردادونه منظم کولای شي چې دا مشخصوي چې کلاسونه به په څه ډول له یو بل سره اړیکه ټینګوي. (همدا کار د پارامترونو او ډېټا ډولونو د تعریفولو په برخه کې هم په ښه ډول پلي کېږي.) د مثال په ډول، د Counter کلاس سره یو ثابت هم تعریفولای شو لکه  $\text{total} \geq 0$ . 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 پروګرام لارښودلیکنې<sup>1</sup> ته هم وده ورکولای شي.

## تطبيق او استعمال

د آیفل ملاتړ کوونکي وایي چې نوموړې ژبه له یوې ژبې څخه ډېره ده: نوموړې ژبه د دې لپاره ډیزاین شوې تر څو د سافټوېر جوړونې په ټول جریان کې د پروګرام بېلا بېلې برخې بیاځلې وکتل او وکارول شي. د آیفل ژبې اوسنی 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 ځانګړنه 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 لارښودلیکني لپاره ګټوره ثابتوي.

آیفل له سي پلس پلس ژبې سره په ورته مهال کې رامنځته شوه. که څه هم، آیفل له شک پرته د سي پلس پلس په پرتله په ډیزاین کې غوره او ستره ژبه ده. خو سي پلس پلس ته دوو ځانګړنو پر آیفل غوره والی ور پرځه کړی: د ارزانه او وړیا تالیفونکو شتون او د هغو زرګونو پروګرام 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 پروګرامي چاپېریال په مرسته د محدود شمېر سوداګریزو کاریالونو جوړولو لپاره وکارول شوه.

آیفل د استازواله سافټوېر ډیزاین د جوړولو په ډګر کې د پراخې برخې اخیستنې په سبب پرېمانه شهرت لري او په دې وروستیو کې د کمپیوټري ماشینۍ 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 بنودنه<sup>1</sup>، وروستایزه بنودنه<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 نښتي 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 څو 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 کليدي ځانگړنه د دې ژبې د پراختيا موندنې خصوصيت دی. کله مو چې يوه کلمه تعريف کړه، نو نوې کلمه هم کټ مټ په هماغه ډول کارول کېدای شي لکه له وړاندې څخه تعريف شوې کلمه چې کارېږي. د کلمو د تعريفولو لپاره 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 اړيکې او په څرگند ډول پروگرام سره د کار کولو وړتيا ته په ښه سترگه گوري. نوموړې ژبه په بشپړ ډول تعاملې ژبه ده، ځکه هره کلمه په کيبورډ ليکل کېدای شي، تر څو اجرا شي او پايله يې په صفحه ښکاره کړي.



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

فورت هېڅکله د پروګرام 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 ښه کاوه. په داسې حال کې چې د فورت پروګرامونه په څرګند ډول ترتیب کېدای شي، خو هغه پروګرامونه چې په ناوړه ډول لیکل شوي وي، د هغو لوستل شاوخوا ناممکنه وي. که څه هم، ځینې وختونه فورت په ځینو برخو کې لوړ دریځ خپل کړی (لکه، پسمتنواله<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 څرګندې کړای شي چې د انسان لپاره د لوستو وړ وي. په یوه عالي ژبه کې، پروګرام لیکونکي متحولونه تعریفوي، او پر هغو باندې د عملیو د ترسره کولو لپاره عبارتونه او فنکشنونه لیکي. په دې ډول پروګرام لیکونکي نور په دې اړه اندېښنه نه کوي تر څو په مفصل ډول د کمپیوټر زېرمه ورته مشخصه کړي او له کمپیوټر څخه بایزري 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 ښودنه 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 پرمختللې نسخه یې (چې تېروتنې یې 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 لیټل<sup>۱</sup> عددي وېلیوګانو ور پورته کول یې اسانه کړل. د فورټروان دې پرمختللې نسخې ته د زیاتره هغو ساینسپوهانو او انجینرانو له لوري هرکلی وشو، چې د هغو محاسبو لپاره یې 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، همالته 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 لیکل شوو پروگرامونو سره د هوسا تعامل کولو د وړتیا په کېون، یو شمېر نوي فیچرونه هم لري. فورټران ۲۰۰۸ تر یو څه بریده د لا پرمختللي نسخې لپاره لا هم یو تجربی نوم دی.

## د پروگرام بېلگه

لاندې ساده بېلگه د دودیز فورټران پروگرام ځینې ځانګړنې واضح کوي:

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

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

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

د خپلې همعصره، کوبال ژبې، په څېر فورټران هم د زیاتره اوسنیو پروګرام لیکونکو له لوري د یوې ګډې ودې او مهالتهره<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 پروګرام کولای شي چې د لایبرېري روتینونو 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 کټورتیا په ځان کې لري چې ډېره اندېښنه یې د پروګراملیکونې 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 کارپالونو د جوړولو لپاره کارېږي چې په انټرنېټ لکه وېب سرورونو باندې ځلي. د جاوا د پروگرام يو ځانگړی ډول applet بلل کېږي چې له وېبپاڼو<sup>1</sup> سره وصل کېدای او د گټه اخيستونکي په وېبرانيستونکي<sup>2</sup> باندې ځلول 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 اړيکځای د استازو<sup>3</sup> جوړول لکه د عامل سيستم کړکۍ<sup>4</sup> او بټنونه 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

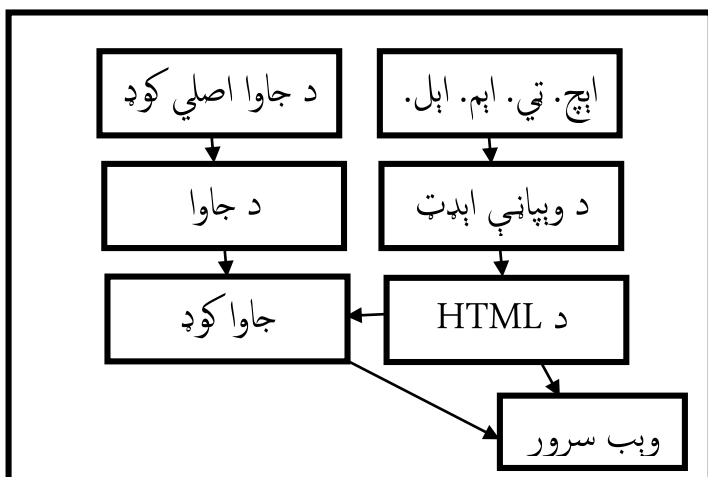


## د پروګرام جوړښت

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

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

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



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

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

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

د دې پروګرام د جوړولو لپاره، پروګرام لیکونکی نوموړی کوډ په جاوا کمپایلر کې تالیف کوي. ورپسې پروګرام لیکونکی د html پاڼه جوړوي چې یو 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 مایکروسیسټم کمپنۍ کې د یوې کورنۍ پروژې په توګه پیل شوه خو یوه داسې ژبه

ډیزاین کړای شي چې د «ځیرکو» مصرفي وسایلو د پروگرام کولو لپاره کار وکړي، لکه ګرېډونکي 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 تر څو 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 هماغسې 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 ډوبلېرانو 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

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

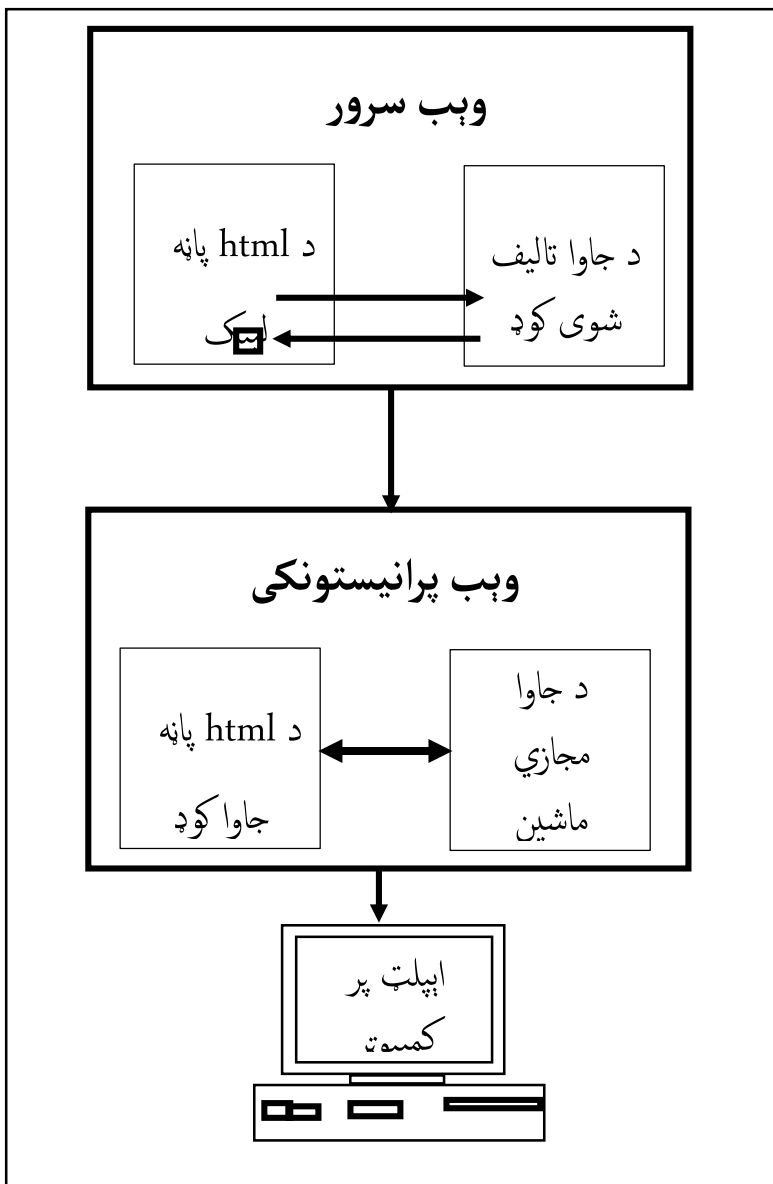
د جاوا یوه بله ستره ځانګړنه دا ده چې جاوا خپلواک-دریځه ژبه ده. پخپله ژبه د ګڼو عامل سیستمونو له دریځونو څخه بېله او پردی ده. د هر دریځ لپاره، د جاوا مجازي ماشین (Java Virtual Machine) جوړېږي چې د جاوا د تالیفونکو لپاره تولید شوی کوډ کمپایل یا ژباړي خو په یاد دریځ وځغېدای شي.

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، وېپیرانیستونکي هم د جاوا اېپلټونو د مخنیوي لپاره غیرفعاله کېدای شي.

## بالغه ټکنالوژي

د سن کمپنۍ جاوا په دوو ذایقو کې راځي: د جاوا معیاري اېډیشن (J2SE) د مایکروسافټ 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 پیچلو چاپېریالونو کې ورته اړتیا وي. مایکروسافټ د وینډوز لپاره د جاوا خپله لهجه هم جوړه کړې، خو د سن د قانوني اقدام په پایله کې یې لاس ترې واخیست.

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



د دې لپاره چې جاوا ایپلټ وځغلي ، کټه 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 ځانګړي 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 بېل مجازي ماشین (JVM) شتون لري.

جاوا، د سافټوېر بیا کارېدونکو اجزاوو ته ځانګړې 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 پروګرامي اړیکځایونه هم شتون لري. په داسې حال کې چې د کلاينټ لوری، جاوا ایپلټونه پر وېب پرائیستونکي وځغوي، خو د جاوا سرور پاڼې (JSP) همدا کوډ د HTML په یوه پاڼه کې ور ځایوي او نوموړی کوډ د سرور لوري کارپال «سرولېټ»<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 تګلوري<sup>1</sup> استاوازله پروګراملیکنه په لوی شمېر دریځونو باندې د پلي کېدو وړ وګرځوي. یاده ژبه نن ورځ د سي یا سي پلس پلس پر ځای د لومړنۍ ژبې په توګه تدریس کېږي. خو، نن ورځ د سافټوېر جوړونې د نړۍ په چټک پرمختګ کې، د نوموړې ژبې په هکله نور دا نظر د اطلاق وړ نه برېښي چې همدا یوازینی مشهوره او غوره ژبه ده.

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

---

<sup>1</sup> Streamline

## اضافي 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." *Info-World*, 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)

## جاواسکریپټ (JavaScript)

جاواسکریپټ له هغو نوموتو ژبو څخه ده چې وېبپاڼې د دې وړ گرځوي چې له ګټه 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 پرانیستونکي کې د لایوسکریپټ<sup>۲</sup> تر نوم لاندې راڅرګنده شوه. له تخنیکي پلوه، جاواسکریپټ 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 جاواسکریپټ نومونه سره نږدې دي، خو جاواسکریپټ د جاوا له پروګرامي ژبې سره په مستقیم ډول کومه اړیکه نه لري.

په لومړنیو کلونو کې جاواسکریپټ 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 بعدی بټنونو او پاپ-اپ کړکیو، د ور زیاتولو لپاره اسانه لاره چمتو کوله. خو څنګه چې پخوانیو نسلونو د ګېشمېر فوټونو له کارولو سره لېوالتیا لرله، نو په دې اساس د جاواسکریپټ د لومړنیو پروګرام لیکونکو وېبپاڼې به له سرګردانونکو او بې ګټو 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 جاواسکرېټ له هغو توپيرونو څخه هم کړېده چې یاده ژبه به په لویو پرایسټونکو کې په څه ډول پلي کېږي. په پایله کې به د نېټسکېپ د ګټه اخیستونکو کار کله ناکله د هغه جاواسکرېټ له امله ټکنی کېده چې د مایکروسافټ په انټرنېټي پرایسټونکي کې به لیکل شوي وو او یا به په جاواسکرېټ کې د مایکروسافټ لپاره لیکل شوي وو. په پای کې د پرایسټونکو نیمګړتیاوو کله نا کله جاواسکرېټ ته اجازه ورکوله چې په وایرس ککړ شوو د پرایسټونکو 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 پرایسټونکو کې د جاواسکرېټ اجرا، غیرفعاله کړي.

## د جاواسکرېټ کارول

د جاواسکرېټ 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 ورکیز/راکیز) باندې د عملیو د ترسره کولو هېڅ وړتیا نه لري. د دې پرځای، جاواسکرېټ یوه ماشین<sup>2</sup> ته بلنه ورکوي، چې د هر کوربه چاپېریال (عموما

<sup>1</sup> browser helpers

<sup>2</sup> engine

یو وېب پرائیستونکي) لپاره لیکل شوی وي. یاد ماشین هغه ځانگړنې تطبیق کوي چې د دې لپاره ډیزاین شوې وي خو دا کنټرول کړي چې یوه وېبپاڼه له کټه اخیستونکي سره په څه ډول تعامل کوي، لکه د کړکيو او کنټرولونو لکه مېنوگانو، بټنونو او یا اېزارېټو<sup>1</sup> څرگندول. جاواسکرېپټ تر دې دمخه چې سرور ته تسلیم شي، په پرائیستونکي کې د یوه وېب چوکاټ<sup>2</sup> د قانون بندۍ لپاره هم کارېږي. عموماً، د پرائیستونکي لوري<sup>3</sup> جاواسکرېپټ پروسېس پر وېب سرورونو باندې بوج کوي او په ورته وخت کې پاڼه چټک غبرگون ته چمتو کوي، د بېلگې په ډول، کله چې کټه اخیستونکي د پاڼې په یوه برخه د ماؤس نښه کوونکی ودروي، نو ورسره گرافیک هم بدل شي.

د جاواسکرېپټ او html د پاڼو ترمنځ اساسي اړیکځای، ډاکومنټ آبجکټ ماډل دی. د نړیوال جال ائتلاف<sup>4</sup> د DOM معیاري فنکشنونه تعریف کړي او ډېری پرائیستونکي په ثابت ډول د دې معیارونو لومړني او دوهمي پوړونو څخه کټه اخلي. که څه هم داسې کټه اخیستونکي شته چې د جاواسکرېپټ معیارونه نه شي ځغلولی، لکه په سترگو رانده کټه اخیستونکي، هغه کټه اخیستونکي چې د موبایلونو پرائیستونکي کاروي (لکه ډیجیټل مرستیال (PDA) یا ځیرک ټلیفونونه)، او یا هغه کټه اخیستونکي چې د امنیتي ستونزو له امله یې جاواسکرېپټ غیرفعاله

---

<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 جاواسکرپټ د وېبپاڼې د اساسي فنکشنونو لپاره کارول کېږي (لکه د فورم پروسېس)، نو ډوېلېر 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 جاواسکرپټ کوډ په مستقیم ډول د ټېګونو په کارولو د html پاڼې سره یو ځای کېږي، لکه په لاندې مثال کې:

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

کله چې هغه پرانیستونکي 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 کارپالونو کې هم اغېز کېدای شي: د مثال په ډول، اډوب اکروباټ او لوستونکي یې او حتی د عامل-سیسټم 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 يې نورو ژبو لکه PHP ته اړولی وي، خو جاوااسکریپټ 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 میکارتی د یوه بېل ډول کمپیوټري ژبې په هکله خپل فکر وړاندې کړ. نوموړی په دې باور وو چې دا ډول یوه ژبه باید د دې وړ وي چې په خپلو اصطلاحاتو کې له ریاضیکي فنکشنونو سره تعامل وکړي — او پر سمبولونو عملیې ترسره کړي، نه دا چې یوازې اعداد محاسبه کړي.

له ماروین مینسکي سره یو ځای، میکارتی د یوې ژبې پلي کول پیل کړل چې LISP یا لیست پروسېسر<sup>۲</sup> نومېدله. څنګه چې یې له نومه ښکاري، 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 ګڼ 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 توګه زېرمه کېږي چې د ډېټا وېلیوګانو 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 ګارېج 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 پروګرامونه په لومړي نظر تړلي ښکاري، ځکه نوموړي پروګرامونه ګڼ 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 ژباړونکي (چې ایول یا eval بلل کېږي ځکه دی راکړیز ارزوي<sup>3</sup>) د ډېټا مسیر ته ګوري او لومړی دا پوښتنه کوي چې آیا راتلونکی توک ثابت دی (لکه عدد، یا په لینډیو کې لیکلی سمبول، متنګرښه، 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 کوري چې آيا نوموړی توک له وړاندې څخه تعريف شوی متحول دی، او که همداسې وي، نو د هغه وېليو واپس کوي. په پايله کې، ژباړونکی کوري چې آيا کوم ليست شته. که چېرته هو، نو له ياد ليست سره د يوې فنکشنې بلنې<sup>1</sup> په څېر چلند کېږي چې ترڅنګ يې ارګومنټونه راغلي وي. په دې اساس فنکشن ته بلنه ورکول کېږي، دېټا ورکول کېږي، او پايله يې واپس کېږي.

د ليسپ 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 ليست  | (۵ ۱ ۴ ۱ ۳) | (۶ ۱ ۴ ۱ ۳)   |
| بولين              | 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 ټینګار کوي. خو بل خوا، لیسپ بیا لومړنۍ فنکشنې<sup>1</sup> ژبه وه. د لیسپ د یوه پروګرام غوښتنه برخه هغه فنکشنونه جوړوي چې له خپلو ارګومنټونو سره یوځای ارزول کېږي. په لیسپ ژبه کې ځاني، او یا اولیه فنکشنونه شتون لري. د ریاضي د عادي عملیو سربېره، د بنسټیز لیست پروسېس لپاره اولیه فنکشنونه لري. د مثال په ډول، لیست فنکشن له خپلو ارګومنټونو څخه یو لیست جوړوي: (لیست ۱ ۲ ۳) په خپله همدا لیست (۱ ۲ ۳) واپس کوي، په داسې حال کې چې cons فنکشن د لیست په سر کې یو اټوم ځای په ځای کوي، او append فنکشن یې بېرته پېخ ته تېل وهي. پروګرام لیکونکي د خپلو فنکشنونو د تعریفولو لپاره د defun له اړین-ویوګي 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. پايله يې دا چې فنکشن بيا بيا خپل ځان ته بلنه ورکوي، لنډمهاله وېليوگانې 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 بنيادي د لیسپ تر ټولو خوندوره ځانگړنه دا وي چې نوموړې ژبه د پروگرامونو (فنکشنونو) او ډېټا ترمنځ هېڅ توپیر 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 عملیو کې بدلون راولي.

## 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 لیکلو لپاره يوه گړندی ژبه وگڼل شوه. په داسې حال کې چې مخکښو کمپیوټرپوهانو ډېر کله د الگوریتمونو د بیانولو 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 کاریلانو 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 وېشي. 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 دریځونو 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 ځانګړنې په ځان کې راتلوي، او د ډېټا له غني ډولونو څخه برخمنه ده، او همدا رنگه د تاکیدي، او ترتیواله پروګراملیکني تګلارو کارولو ته هم پراخه اجازه ورکوي لکه سي ته ورته ژبو کې. په دې ډول نوموړې ژبه د پروګراملیکني لپاره ګڼ سبکونه وړاندې کوي. لیسپ نن ورځ د سوداګریزو او اشتراکي پوستغالو (shareware) په نسخو کې په پراخه توګه شتون لري.

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

### اضافي لوست:

- 1- Dybvig, R. Kent. *The SCHEME Programming Language*. 3rd ed. Cambridge, Mass.: MIT Press, 2003.
- 2- “Lisp Information and Resources.” Available online. URL: <http://www.lispmachine.net/>. Accessed August 13, 2007.
- 3- Queinnec, Christian. *Lisp in Small Pieces*. New York: Cambridge University Press, 1996.
- 4- Seibel, Peter. *Practical Common Lisp*. Berkeley, Calif.: Apress, 2005.

\*\*\*

---

<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>۱</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 اپېل لوگو په رسامي 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 پیپرټ 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 پیپرټ زړه ته، د ودې او پرمختګ له ارواپوه جان پیاجټ سره له کار کولو وروسته د یوې ښوونیزې کمپیوټري ژبې د جوړولو لېوالتیا ور ولوېده. پیپرټ، د پیاجټ پر رغښتواله<sup>۲</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 مفاهیمو د ځایولو له لارې زده کړه کوي. پېپرټ په دې باوري شو چې د یوې انتزاعي کمپیوټري ژبې لکه فورټران او یا حتی بېسک زده کړه د ماشومانو لپاره سخته ده، ځکه چې د دې ژبو الجبري فورمولونو او نحوي ترکیب د هغوی له ورځنیو فعالیتونو لکه قدم وهل، لوبو کولو، رسامي کولو، او یا د شیانو له جوړولو سره ډېر کم سمون درلود.

د مثال په ډول، ډېری کمپیوټري ژبې د هغو عبارتونو په کارولو ګرافیک پلي کوي چې د کارټیزن محورونو (X,Y) په مرسته د پردې ټکي مشخصوي. د مثال په ډول، یوه 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 سیستم سره بلدتیا په پای کې یو څوک د دې جوګه کوي چې دا عملیه په ذهن کې ترسیم کړي، او دا له درک څخه ډېر لږې خبره ده.

پېپرټ په خپله لوګو ژبه کې یو کیشپ راوړي. کیشپ په اصل کې یو رېښتینی روبات وي چې شاوخوا ته د حرکت کولو لپاره پروګرام کېدای شي؛ نن ورځ په ډېری سیستمونو کې نوموړی کیشپ د یوه کرسر په بڼه وړاندې کېږي. څنګه

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 پیاوړتیاو ته په پام سره، زده کړه زموږ له فزیکي نړۍ او ورځینو فعالیتونو سره اوږه په اوږه 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 پرمهانه 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 غږونه، 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 پروگراملیکني 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 چاپېریال 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 کیشیونه 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 تعاملې فطرت زلمیو زده کوونکو ته هم د کمپیوټرپوهنې د ښوونې 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. جانیرو» 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 (تاکیدی) پروګراملیکني او فنکشنی پروګراملیکني 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 جوړښت لري:

```
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 سرپرېره، 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 متنکړنه وي. د مثال په ډول، لاندې جدول درې وېلیوگانې لري چې د متنکړنو 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 ډګر او کارونه

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

په ژبه کې د غلمهاله<sup>1</sup> پېکجونو له امله، لووا زیاتره وختونه کارپالي پوستغالو ته ور اضافه کېږي تر څو د پوستغالي د پراختیا او سمون لپاره اړیکهای 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 لومړیو وختونو کې د کمپیوټر پوهانو لویه اندېښنه د داسې لارو موندل وو چې خپل پروګرامونه په غوره ډول منظم او ورغوي. په اصل کې تر یو څه بریده داسې یوه ژبه (وګورئ: الګول) جوړه شوې هم وه چې د پروګراملیکني روغ اصول یې چمتو او پلي کوي، چې د کنټرول جوړښتونو کارونه هم پکې یادولی شو. خو الګول د ډېټا ډولونو او نورو هغو ځانګړنو له کمښت سره مخ وه چې د عملي پروګراملیکني لپاره ورته اړتیا وه، په ورته وخت کې له شک پرته د یوې ښې ښوونیزې ژبې لپاره خورا پېچلې او بې ثباته ژبه هم وه.

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

د پاسکال اصلي تکلوری پر الګول ورغلی وو او ورسره یې دا ژبه پراخه کړې هم وه. د تورو، بولین، او د ډېټا ډولونو لپاره د ملاتړ په چمتو کولو، پاسکال ګڼه 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> (اوږد عدد)، او معاش (یو اعشاري عدد) سره یو ځای او ترکیب کړي. په دقیق ډول د ډېټا ډولونو کارونه د پروسو له رابللو او تعریفولو تر روش پورې هم ادامه کولای شي.

پاسکال د غوره تعریف شوې ژبې په وړاندې کولو چې الګوریتمونه پکې په څرګند ډول تشریح کېدای شوی، په لویه کچه د کمپیوټر پوهانو او ښوونکو پام ځان ته را وکاږه. د پاسکال په مقبولیت کې د نوموړې ژبې د تالیفونکو نوښتګر ډیزاین هم لوی لاس درلود. د هغه وخت د ماشین-اړونده<sup>2</sup> تالیفونکو پر خلاف، د پاسکال تالیفونکي په مستقیم ډول ماشین کوډ نه رامنځته کاوه. بلکې، د نوموړي راکړیز «P-code» وو، چې یو ډول انتزاعي کمپیوټري ژبه وه. یو غځلمهاله سیستم چې د هر کمپیوټر لپاره بېل 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

<sup>2</sup> machine-specific



ترجمان په لیکلو، هغه بل ماشین ته ورل کېدای شوی. همدا ستراتیژي دوه لسیزې وروسته د وېب کار یا لانونو لپاره د نوميالی ژبې د جوړوونکو له خوا وکارول شوه. (وګورئ: جاوا)

## د پاسکال د پروګرام جوړښت

لاندینی ساده پروګرام په پاسکال ژبه کې د یوه پروګرام اساسي جوړښت تشریح کوي. (په پرېرمخ لیکل شوي کلمات، اړین-ویکي دي چې د پروګرام د جوړښت لپاره کارېږي). پروګرام په Type برخه پیل کېږي چې د ګټه اخیستونکي ډېټا ډولونه وضع کوي. په دې برخه کې کټارونه، ټولګي، ریکارډونه (مرکب ډولونه چې د ډېټا بېلا بېل اساسي ډولونه لرلای شي) لري. په دې ځای کې تر لسو پورې د (کاملو اعدادو) یو کټار تعریفېږي چې IntList بلل کېږي.

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

لکه یوه متحول ته د اړتیا په ځای کې د یوه بل ناتعریف شوي متحول شتون. نن ورځ 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 وېلیوګانو د ټاکلو لپاره استعمالېږي. باندینی if عبارت د دې لپاره استعمال شوی چې ګټه اخیستونکی د توکونو ناسم شمېر درج نه کړي. ورپسې د for څرخ ورکړیزه وېلیو لولي، په کتار کې یې په خپل ځای، ځای پر ځای کوي، او ټولټال ذخیره کوي. هماغه ټولټال حاصل، بیا د اوسط د محاسبه کولو لپاره کارېږي، چې په پای کې د writeln عبارت په مرسته راکړیز راوباسي.

```

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

```

---

<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 اپېل-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 تریو 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 چاپېریال درلود چې د سافټوېر جوړول

يې د پخوانيو درنو او گرانبيه تاليفوونکو لکه د مايکروسافټ 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 ترلاسه کړ چې د پروگرامليکني د بنوونې او الگوريتمونو د څرگندولو لپاره د لومړنۍ 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> نحوي ترکيب څخه گټه پورته کوله او د ډېټا ډولونو په اړه د پاسکال له ميکانيزم څخه هم تشه وه. سيستمي پگرام ليکونکو سي ژبې ته ځانگړي لومړيتوب ورکاوه چې کولای يې شوای 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 ډوبلېران يې د يونيکس د مهمو ډوبلېرانو له ډلې څخه وو، چې هغه مهال د کمپيوټري چاپريالونو په کمپس کې يو ډېر نوموتی عامل سيستم 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 مایډولا-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 ابتدا په جوړولو کې هم پرېمانه اغېز درلود، هغه ژبه چې د امریکې د فدرالي دولت له لوري هم ومنل شوه او جوړښتواله پروګراملیکنه یې د استازواله پروګراملکني له ځانګړنو سره ترکیب کړې وه، او د کوچنیو پروګرامونو د پراخه پېکجونو 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>۱</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>۲</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 سمبالېده. 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 پروتوکولونو 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> funcnality

```
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 پروګراملیکنې ته لاره هواروي. خو د پرل ډېری مینوال دې ژبې ته د یوې هر اړخیزې ژبې او د وېب پروګراملیکنې د ننګونکې نړۍ لپاره د یوې اساسي ابزارټولګې په سترګه کوري. خومره چې د ۱۹۹۰ کلونو په وروستیو کې ژبې وده کوله، په هماغه ډول د استازواله پروګراملیکنې ځانګړنې هم ور زیاتېدې.

## د پرل د پروګرام بېلګه

لاندې ډېر ساده کوډ د پرل داسې پروګرام تشرېح کوي چې له یوه فایل څخه د ډېټا یو څو کرښې ور اخلي او بیا یې پرېنټ 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 پرېنټ 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 یو څه کم رنګه شوې، خو پرل پوه بالغه ټکنالوژي ده چې اوس هم په لویه کچه کارول کېږي، په ځانګړي ډول د ډېټا د استخراج، اړولو، او لاسوهنې لپاره. د پرل د ټولیز ارشیف شبکه (CPAN)<sup>1</sup> له ۱۲۰۰۰ څخه زیات ماډولونه لري چې په وړیا توګه د پروګرام لیکونکو لپاره د لاسرسي وړ دي.

### اضافي لوست:

- 1- Comprehensive Perl Archive Network (CPAN). Available online.  
URL: <http://www.cpan.org/>. Accessed August 17, 2007.
- 2- Conway, Damian. *Perl Best Practices*. Sebastapol, Calif.: O'Reilly, 2005.
- 3- Lee, James. *Beginning Perl*. 2nd ed. Berkeley, Calif.: Apress, 2004.
- 4- Schwartz, Randall L., Tom Phoenix, and Brian D. Foy. *Learning Perl*. 4th ed. Sebastapol, Calif.: O'Reilly, 2005.
- 5- Wall, Larry, Tom Christiansen, and Jon Orwant. *Programming Perl*. 3rd ed. Sebastapol, Calif.: O'Reilly, 2000.

\*\*\*

## (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 وایست. په ۱۹۹۷ کې د ژبې تجزيه کوونکي<sup>2</sup>، دوو اسرائيلي ډولپرانو 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 کوونکي. 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 کاريالونو 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 محدودوونکو <? او ?> دننه تجزيه کوي. د دې محدودوونکو معادل بيا په لاندې ډول دی:

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

سربېره پر دې چې کوډ د HTML په پاڼو کې مختلط وي، پي اېچ پي په کمانډکرنښه (command line) کې په تعاملې ډول هم کارول کېدای شي، چېرته چې دې ژبې د آگ، پرل او شېل په څېر متنواله زړو ژبو ځای نیولی دی. پي اېچ پي د گټه اخيستونکي د اړیکځای له لايبرېريانو 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 فايلونو، د ډېټا ريکارډونه او داسې نور بېرته راگرځوي. د دې ترڅنګ، پي اېچ پي-5 د شخصي

او خوندي، غړو متحولونو، کنسټرکټرونو او ډیکنسټرکټرونو او د نورو هغو ځانگړنو په ګډون 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 ګډ 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 سمبالښت لپاره ژبنی زیاتونې چمتو کوي لکه د اډوب فلش خوځښتونه او داسې نور. پروګرام 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 ځانګړنو او په اسانه د متنوالۍ تعامل پي ایچ پي ژبه د ډېری هغو وېب ډویلرانو لپاره د خوښې وړ ژبه ګرځولې چې د ټکنالوژیو د ډلې چې LAMP نومېږي، د یوه غړي په توګه ګټه 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 ډولپېرانو 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 ونيوله. 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» يا  $NPL^1$  په نوم ياده شوه. دا چې دا لنډيز له وړاندې څخه لا د برتانيې د ملي فزيکي لابراتوار (National Physical Laboratory) لپاره استفاده کېده، نو د ژبې نوم په  $PL/1$  يا «لومړنۍ پروګرامي ژبه» بدل شو.

## د ژبې ځانګړنې

پي اېل/۱ د «هرکاره» ژبې توګه پېژندل کېږي، ځکه چې د پروګرام ليکونکو لپاره ډېر زيات داسې مشخصات چمتو کوي چې له بېلا بېلو سرچينو څخه اخيستل شوي وو. اساسي بلاک جوړښتونه او کنټرول جوړښتونه له الکول څخه اخيستل شوي، چې يوه نښتۍ کوچنۍ ژبه وه چې د کمپيوټر پوهانو له خوا د جوړښتواله پروګرامليکنې د يوه ماډل په توګه وړاندیز شوې وه، او پاسکال ژبې ته ورته وه (وګورئ: پاسکال). بلاکونه په يو بل کې اغړل کېدای، او هغه متحولونه چې د يوه بلاک دننه وضع شوي وي، يوازې د هماغه بلاک دننه او د نورو تړلو بلاکونو دننه د لاسرسي وړ وي، که نه بله لاره يې دا ده چې په نړيوال ډول وضع شي.

پي اېل/۱ د ډېټا ډولونو پرېمانه شمېر په ځان کې رانغاړي، تر دې چې د عددي ډېټا لپاره د يوه عدد د رقمونو شمېر هم مشخصولای شي. د PICTURE عبارت کوټال ژبې ته ورته د يوې ټاکلې اډانې د تعينولو 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 اتوماتیک (د بلاکونو له درج او ایستلو سره په مشترک ډول، یا ځای نیسي او یا ځای پرېږدي) او Controlled پورې (چېرته چې حافظه باید په ښکاره ډول 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 سمالښت لپاره پرېمانه اسانتیاوې لري چې دا ډول تېروتنې د هارډویر له شرایطو، ریاضیکي معادلو، د فایلونو له سمالښت او یا نورو شرایطو څخه را ټوکېدلای شي.

## د پروګرام بېلګه

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

```
COUNTM: 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 COUNTM;
```

## د ژبې اغېز

د خپلو عملي ځانګړنو او د آی بي اېم شرکت د نومباليو 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 دریځونو او عامل سیستمونو ته هم انتقال شوه. کله چې شخصي کمپیوټرونه منځته راغلل، پي اېل ۱/ د آی بي 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 پروګراملیکنې د سرلاري اېډسګر 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 موږ به د خپل ذهني چوکاټ دننه د خپلو پروګرامونو وده په خپل ځای وساتلای شو، کله چې زموږ پروګرامي ژبه له دومره زیاتو ځانګړنو سره – ګوزاره مې وکړه! دمخه لا زموږ له ذهني کنټرول څخه په 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.

## اضافی 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) خو 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 تعریفوي چې پر دې حقایقو پلي کېږي. د مثال په ډول:

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

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

کله چې یو پروگرام ځغلي، نو همدا ډول ګروپړنې<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 ګروپړنو پلي کېږي او هڅه کوي چې یو داسې منطقي زنځیر جوړ کړي چې د هغې ګروپړنې په اثبات رسولو ته لاره هواره کړي. که چېرې زنځیر مات شي (په دې چې هېڅ سمون لرونکی حقیقت یا اصول ونه موندل شي)، سیستم بېرته په شا راګرځي او په یوه بل مطابقت کوونکي حقیقت یا اصول پسې ګوري او له دې ځایه د یو بل زنځیر لپاره هڅه کوي.

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

---

<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>۱</sup> د جوړولو په لټه کې وو او پرولوګ یې د خپلې خونې وړ ژبه وټاکله. که څه هم ځینې ماشینونه جوړ شول، خو د دې نظریې هېڅکله رښتینې بری په برخه نه شو. یورلېنډ انټرنېشنل (د بریالي تریو پاسکال جوړوونکي) تریو پرولوګ را وایستله چې یاده ژبه یې د هغو زده کوونکو لپاره د لاسرسي وړ وګرځوله چې له شخصي کمپیوټرونو څخه ګټه پورته کوي، که څه هم په دې ژبه کې ځینې غیرمعیاري ژبنی زیاتونې استفاده کېدې.

که څه هم په سوداګریز ډګر کې یې بریا محدوده وه، خو پرولوګ د مصنوعي ځیرکتیا د څېړنو په لوی شمېر ډګرونو کې کارول شوې. د نوموړې پر اصولو ولاړ جوړښت د ځیرکو سیستمونو، پوهنښتونو<sup>۲</sup>، او طبیعي ژبې پروسېس لپاره مناسبه ژبه وه. یاده ژبه د هغو نمونوي (پروټایپي) سیستمونو د ډیزاین لپاره هم کارول کېدای شي چې وروسته د چټکتیا او اغېزمنتیا لپاره په دودیزو ژبو کې بیاځلي 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/~jrffisher/www/prolog\\_tutorial/contents.html](http://www.csupomona.edu/~jrffisher/www/prolog_tutorial/contents.html). Accessed August 17, 2007.
- 4- Sterling, Leon, and Ehud Shapiro. *The Art of Prolog: Advanced Programming Techniques*. 2nd ed. Cambridge, Mass.: MIT Press, 1994.
- 5- SWI-Prolog [free Prolog for Windows, Linux, and Mac OS]. Available online. URL: <http://www.swi-prolog.org/>. Accessed August 17, 2007.

\*\*\*



(۲۳)

## پایتان (Python)

د ګایډو وان روسم له خوا جوړه شوې او په لومړي ځل په ۱۹۹۰ کې وړاندې شوې ژبه، پایتان، یو څه اسانه خو ځواکمنه متنواله ژبه ده. پایتان نوم د بریتانیې د کومیدي له نومیالی ډلې «مونتي پایتان»<sup>۱</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 کورنۍ 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 سیمیکالو 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 ځغلبډای شي. د دې د بدیل په توګه، 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 ګټور پېرېوخي. بل خوا پايتان، ډېره عمومي او په څرګند جوړښت رغېدلي ژبه ده چې د لوی شمېر کاريالونو لپاره يوه وړ ژبه او د لويو او پېچلو پوستغالو لپاره په اساني سره د پراختيا موندنې له لويو وړتياوو څخه برخمنه ده.

## اضافي لوست:

- 1- Lutz, Mark. *Learning Python*. 3rd. ed. Sebastapol, Calif.: O'Reilly, 2007.
- 2- ———. *Programming Python*. 3rd ed. Sebastapol, Calif.: O'Reilly, 2006.
- 3- Python Programming Language—Official Web site. Available online. URL: <http://www.python.org/>. Accessed August 17, 2007.
- 4- Zelle, John M. *Python Programming: An Introduction to Computer Science*. Wilsonville, Ore.: Franklin, Beedle & Associates, 2004.

\*\*\*

(۲۴)

## آرپي جي (RPG)

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

ډېري کوبال پروگرامونه ډېټا لولي، ازموینې او محاسبې ترسره کوي او پایلې یې پرنټ کوي. «آر پی جی» چې د نوو مېن فرېم کمپیوټرونو، سیسټم/۳۶۰ او کوچنیو کمپیوټرونو، سیسټم/۳۶۰ لپاره جوړه شوه، د لومړي ځل لپاره په ۱۹۶۴ کې د استعمال لپاره وړاندې شوه. نوموړې ژبه د کوبال پورته یاده پروسه ساده کوي او د پروگرام د کوډ ډېري عبارات له منځه وړي.

د آر پی جی یو کلاسیک پروگرام د آر پی جی په ډایروي جریان کې جوړېږي، چې درې پړاوونه لري. په ورکړیز پړاو کې، ورکړیز واحد(ونه)، فایل 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 پروګرامي ژبه ده چې په وروستيو کلونو کې يې ښه نوم پيدا کړی، په ځانګړي ډول د وېب جوړونې په برخه کې. نوموړې ژبه د يوکيرو ماخوموتو له خوا ډيزاين او په ۱۹۹۵ کې د استعمال لپاره راووته؛ روبي، غونډه نځوي ترکيب لري چې د پرل او نورو متنواله ژبو ګټه 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 ټينګار کړی چې د ژبې ډيزاين په داسې شکل وکړي چې د پروګرام ليکونکي لپاره طبيعي او خوندوره اوسي، نه دا چې د پروګرام ليکونکي د ماشين پر اړتياوو خپله انرژي راټوله کړي.

## جوړښت

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

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

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

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

---

<sup>1</sup> Hashes

متکرېنه Hello world درج کوي. ورپسې ارزوونکي داسې راپور چمتو کوي چې: puts میتود تشه وېلیو (nil) راگرځولې 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 اورګاډي (Ruby on Rails)

د روبي د پروګراملیکني لپاره ډېر نومیالی کاري چاپیریال روبي پر اورګاډي دی، چې یو پرانیستي سرچینه کاریالي چارچوکات دی چې خاص په دې موخه جوړ شوی چې داسې پروګرامونه ورباندې ولیکل شي چې وبسایتونه له ډېتابېسونو سره وصل کړي. نوموړی چارچوکات د ماډل-لید کنټرولونکي<sup>1</sup> تګلارې باندې بنا دی چې ډېټا 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 ډېټا په اساس حرکت وکړي. ډوبلېران کولای شي چې د روبي په مرسته نښتي کارپالونه<sup>2</sup> او شته پېکجونه هم پراخه کړي.

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

<sup>2</sup> plug-ins



## اضافي لوست:

- 1- Baird, Kevin. *Ruby by Example: Concepts and Code*. San Francisco: No Starch Press, 2007.
- 2- Burd, Barry. *Ruby on Rails for Dummies*. Hoboken, N.J.: Wiley, 2007.
- 3- Cooper, Peter. *Beginning Ruby: From Novice to Professional*. Berkeley, Calif.: Apress, 2007.
- 4- “Ruby: A Programmer’s Best Friend.” Available online.  
URL: [http:// www.ruby-lang.org/en/](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/](http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html)  
[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 ډېرو په زړه پورې کټو 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، ناروېژي 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 ځانګړو خدماتي استازو (د کړکۍ له صرافانو) څخه د خدمت تقاضا کوي. دوی په یوه لیکه کې حرکت کوي او د دوی حرکت د وخت په بېلا بېلو نقطو کې ثبتېږي.

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

شي دا کار سخت کړی وو چې داسې یو 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 ښکاري 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 ګډه 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 ټینګار لیدلوری یې را غبرګاوه (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 ګڼه ګوښې 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 توګه باید اډېټ شي.

الګول ۶۰ تر مخکې لا د ګوډ 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 پرانيستلې شي. 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 بنيادي 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، کواکې 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 الکول ټولې ځانگړنې په ځان کې لرلې، خو د هم مهاله پېښو د ماډل کولو لپاره يې ډېره بڼه استفاده ترې وکړه.

سيمولا-۱ خورا بريالۍ 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 نومیالۍ سي پلس پلس ژبه د «سي له کلاسونو سره» د جوړولو په هڅه را پیل شوه چې په جوړښت کې یې د سیمولا کړنې هم ګرځېدې.

### اضافي لوست:

- 1- Holmevik, Jan-Rune. "Compiling Simula: a Study in the History of Computing and the Construction of the SIMULA Programming Languages." *STS Report* (Trondheim). Available online. URL: <http://staff.um.edu.mt/jskl1/simula.html>. Accessed August 21, 2007.
- 2- Nygaard, K., and O.-J. Dahl. "The Development of the Simula Languages" in *The History of Programming Languages*, R. L. Wexelblat, ed. New York: Academic Press, 1981.
- 3- Pooley, R. *Introduction to Programming with Simula*. Oxfordshire, U.K.: Alfred Waller, 1987.
- 4- Sebesta, Robert W. *Concepts of Programming Languages*. 7th ed. Boston: Addison-Wesley, 2006.

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

\*\*\*

( ۲۷ )

## سمالټاک (Smalltalk) یا وړې خبرې

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

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

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

بل اساسي فکر چې په سمالتاک کې وکارول شو، هغه استازواله پروگراملیکنه وه چې د لومړي ځل لپاره په سیمولا-۶۷ ژبه کې رامنځته شو. د ژبو شته ځانګړنو ته یوازې د ساده کلاسونو او استازو د ور زیاتولو پر ځای، کای 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. ګټه اخیستونکو نوي کلاسونه هم جوړولای شوی چې د «ځاني» کلاسونو چلند به ورسره کېده. دې خبرې ته د اندېښنې کولو هېڅ اړتیا نه وه چې وړاندې له دې چې یو خاص ډول متحولونه وکارول شي، هغه لومړی باید بیان شي؛ په سمالتاک کې متحولونه په هر ډول استازي پورې تړل کېدای شي.

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

Total = Total + 1

---

<sup>1</sup> Assignment

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

$Total \leftarrow Total + 1$

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 وېلیو کې ۱ جمع کوي او بیا هغه وېلیو د نوي استازي په توګه واپس کوي، چې اوس د Total متحول په وسیله ورته مراجعه کېږي.

په 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 پوري 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 توګه مجسم کړې وه چې د هغو پروګرام لیکونکو له خوا هم پراختیا وموندلې شي چې د پروګرامليکني تجربه نه لري، او نوموړي خپل مخکښ ګرافیکي اپیکهای د داسې روش په توګه ډیزاین کړ چې د ګټه اخیستونکو لپاره دا کار اسانه کړي خو له سیستم سره نېغ په نېغه کار وکولای شي.

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

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

د سمالتاک معنوي ځواک دا ژبه، د مصنوعي ځیرکتیا او پېچلي تقلید په ډگر کې د ځانگړو پروژو لپاره یوه زړه رابښکونکې ژبه گرځولې، او دا د هغو کسانو لپاره یو لوی انعام دی چې پر استازواله روش بنا یوه کره ژبه غواړي چېرته یو کارپال په واضح ډول د رښتیني نړۍ حالت انځورولای شي. سمالتاک ماشومانو (او نورو) ته د پروگراملیکونې د تدریس لپاره لا هم یو غوره انتخاب دی. د سمالتاک یوه نسخه چې squeak بلل کېږي، د گرافیک او نورو فنکشنونو لپاره یو غني کاري چاپیریال برابروي. Squeak او د سمالتاک یو شمېر نور تطبیقات د مختلفو کمپیوټري سیستمونو لپاره په وړیا ډول د ډاونلوډ کېدو وړ دي.

### اضافي لوست:

- 1- Ducasse, Stephane. *Squeak: Learn Programming with Robots*. Berkeley, Calif.: Apress, 2005.
- 2- Klimas, Edward J., Suzanne Skublics, and David A. Thomas. *Smalltalk with Style*. Englewood Cliffs, N.J.: Prentice Hall, 1996.
- 3- Lewis, Simon. *The Art and Science of Smalltalk*. Englewood Cliffs, N.J.: Prentice Hall, 1995.
- 4- Smalltalk. Available online. URL: <http://smalltalk.org>. Accessed August 21, 2007.
- 5- Squeak. Available online. URL: <http://www.squeak.org/>. Accessed August 21, 2007.

\*\*\*

(۲۸)

## ټي سي اېل 1 (TCL)

په ۱۹۸۸ کې د جان 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. ټي سي اېل یو شمېر ګټور ډېټا جوړښتونه هم لري لکه اشتراکي کټارونه<sup>1</sup> چې د ډېټا توکونو جوړې پکې راځي، لکه:

```
set abbr (California)
```

## زیاتونې او ګټې

ټي 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 روبي لپاره د ګټه اخیستونکي د اړیکځایونو جوړولو لپاره یوه لایبرېري وړاندې کوي.

ټي سي اېل د شته کارپالونو د نښلونکو لپاره د «سریښ» په توګه پېژندل کېږي. نوموړې ژبه نسبتاً اسانه او یو متن په تعاملې شکل کوري (چې ډېر پر کمانډ کرښه ترسره کېږي) او بیا دا متن د یوه کارپال ګوډ ته ور درج کوي. کله چې

---

<sup>1</sup> associative array

کاريال ځعلي، د تي سي اېل ترجمان هغه متن ځغلوې، چې وروسته يې راګړيز  
د اصلي کاريال 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 سکریپټ (VBScript)

وي بي سکریپټ، هغه ژبه چې د زوکړې مهال يې د ۱۹۹۰ کلونو منځینو وختونو ته رسېږي، یوه متنواله ژبه ده چې مایکروسافټ کمپنۍ جوړه کړې او بنسټ یې په خپل مشهور وېژوال بېسیک پروګرامي ژبې ایښودل شوی. نوموړې ژبه د هغه تحول یوه برخه ده چې مایکروسافټ «فعال متنواله» باله، او په هغو اجزاوو ولاړه وه چې د پوستغالي وړتیاوو ته له بهر څخه د لاسرسي اجازه ورکوي. هغه کوربه چاپېریال چې متنونه پکې ځغلي د ویندوز له خوا چمتو کېږي (لکه د ویندوز متني کوربه<sup>1</sup> او یا د مایکروسافټ انټرنټي اېکسپلورر دننه اجرا کېږي).

د کلاينټ-لوري پروسیس لپاره، وي بي سکریپټ د هغو متنونو د لیکلو لپاره استعمالېدای شي چې په html پاڼو کې مختلط وي، چې له معیاري ډاکومنټ آېجکټ ماډل سره د نورو وېبې متنواله ژبو په څېر اړیکه نیسي (په ځانګړي ډول، وګورئ: جاواسکریپټ). که څه هم، د مایکروسافټ له پرائیستونکو پرته په نورو پرائیستونکو لکه فایرفاکس یا اوپېرا کې د وي بي سکریپټ ملاتړ نه کېږي، نو د دې پر ځای ډوبلېران باید له انطباق پذیره جاواسکریپټ څخه ګټه پورته کړي. وي

---

<sup>1</sup> Windows Script Host

بي سکرپټ په وېب سرور باندې د پروسېس 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 متنچوکاټ 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 یې ګرځوي چې لا ډېرې پیچلې کړنې سرته ورسوي لکه د ډېټابېسونو برابرول او د وینډوز د مدیریتي آلاتو <sup>1</sup>(WMI) او فعاله خدماتي اړیکځای <sup>2</sup>(ADSI) په مرسته د وینډوز سیستمونو د نورو برخو تنظیمات ترسره کول.

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

---

1 Windows Management Instrumentaion

2 Active Directory Services Interface

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

### اضافي لوست:

- 1- Jones, Don. *VBScript, WMI, and ADSI Unleashed*. Indianapolis: Sams, 2007.
- 2- VBScript Sample Scripts. Available online. URL: <http://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.

\*\*\*

پای

## 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**