M. ASHUROV
N. XAYTULLAYEVA
SH. SATTOROVA

DASTURLASH TILLARI

O'quv qo'llanma

MUNDARIJA

1-bob. OB'EKTGA YO'NALTIRILGAN DASTURLASH TILLARI

1.1. "DASTURLASH TILLARI" FANIGA KIRISH.

Reja:

1. Dasturlash tillari va ularning klassifikasiyasi.
2. Mashinaga mo'ljallangan va proseduraga mo'ljallangan dasturlash tillari.
3. Yuqori darjali dasturlash tillari.
4. Interpretatorlar va kompilyatorlar.
5. Dasturlarni translyasiyalash.
6. Muyyan dasturlash tilining alifbosi, buruqlar tizimi va operatorlari.

Tayanch iboralar: Dasturlash; dasturchi; Dasturlash tillari turlari; assembler tili; fortran tili; beysik tili; quyi darajadagi DT; o'rta darajadagi DT; yuqori darajadagi DT;Kompilyatsiya va interpretatsiya qilinuvchi tillar; Komp'yuterda masala echish bosqichlari.

Tezkor elektron xisoblash mashinalarining paydo bo'lishi ***dasturlash tili*** deb ataluvchi turli-tuman belgilar sistemalarining paydo bo'lishiga olib keldi. SHunday qilib, hisoblash mashinalarida bajarilishi kerak bo'lgan jarayonlarni tavsiflash uchun qo'llaniladigan belgilar (simvollar) sistemasini ***dasturlash tili*** deb yuritamiz.

EHMlar uchun dastur tuzish va uni ishlatish juda murakkab va ko'p mehnat talab etadigan jarayon bo'lib, uning uchun ko'p aqliy mexnat va vaqt talab qilinadi. Shuning uchun yangi algoritmik tillarni yaratuvchilar dasturlashni sodda va xayotimizning turli sohalarida mexnat qiluvchilar uchun tushunarli bo'lishiga xarakat qilishadi. Hozirgi kunda dasturlash tillarining bir necha turi mavjud:

Bundan ko'rinadiki dasturlash tillarining soni bir necha yuzdan ortiq ekanligi.

Beysik dasturlash tili 1964 y. AQSh ning Dortmunt kollejining ilmiy xodimlari Jon Kemeni va Tomes Kurts tomonidan yaratilgan. Bu til turli hisoblashlarga doir masalalarni kompyuter bilan muloqat holda xal qilish uchun yaratildi.

Basic so'zi Beginners Allpurpose Simbolic Instruction Code –degan kengaytmasidan iborat bo'lib, ya'ni boshlovchilar uchun mo'ljallangan ko'p maqsadli, belgili ko'rsatmalar tili degan ma'noni bildiradi. Ushbu til kompyuter xotirasiga

qo’yiladigan talablarning juda kamligi sababli, ShK larda ishlatiladigan til bo’lib qoldi. Bu tilning bir necha ko’rinishi mavjud bo’lib, maqsad foydalanuvchilarning muloqotida osonlik tug’dirish.

Assembler tili inglizchasiga Assembler- bu EHM uchun mashina kodlarida dastur yozish ishini yengillashtirish maqsadida 40 yillar ohiri 50 yillarning boshida yaratilgan.

Fortran tili –inglizcha Formula Translator – formulani translyatsiya qilish so’zlaridan qisqartirib olingan bo’lib, formula tarjimoni degan ma’noni bildiradi va bu til muhandislik, ilmiy-texnik masalalarni yechishga mo’ljallangan til hisoblanadi. Ushbu tilning qator variantlari yaratilgan, ulardan eng mashhurlari Fortran II va Fortran IV. Fortran tilining asosiy g’oyalari keyingi Algol-60, 68, PL-1 va boshqa tillarda rivojlantirilgan. Bu til 1954 yil yaratilgan.

Shunday maqsadni 1970 yili Shvesiyalik Oliy texnika o’quv yurtining professori Niklaus Virt o’z oldiga qo’ydi va u tomonidan tavsiya etilgan algoritmik tilni yaratdi va ulug’ fransuz olimi B. Paskal (1623-1662) nomi bilan atadi.

Dasturlash tillarining sintaktik jihatdan turlari

Dasturlash tillarining sintaktik jihatdan turlari 3 turga bo‘linadi:

DTning sintaktik jihatdan turlari		
Quyi	O'rta	Yuqori

Quyi darajadagi dasturlash tili “Mashina tili” deb ham ataladi. Ushbu tilda dasturlar to‘g‘ridan-to‘g‘ri Operativ Xotira(OX) katakchalari va protsessor reyestrlari bilan ishlab tuziladi. Ushbu tildagi buyruqlar Markaziy Protsessor(MP)ning operatsiyalariga to‘g‘ri keladi. Buyruqlar ikkilik kodda yozilgan:

```
1: 10110000 01100001
2: (AL reyestriga 16-lik 61 sonini joylash)
```

Bir paytlar perfokartalar yordamida aynan mashina tilida dasturlar yozilgan.

O‘rta daraja dasturlash tillarida protsessor buyruqlarini mnemonik kodlarga(buyruqqa mos qisqartirilgan so‘zlar) almashtirilgan. Assembler tili bunga misoldir. O‘rta darajadagi dasturlash tillarida ham bir protsessor operatsiya deyarli bir

buyruqqa mos keladi. Masalan, yuqoridagi mashina kodi Assemblerda quyidagicha yoziladi:

```
1: MOV AL, 61h
2: (Yuqoridagidan ko‘ra tushunarliroq)
```

Ko‘rib turganingizdek, Assembler tili mashina kodidan bir pog‘ona yuqorida turadi xolos.

ESDA TUTING! Keyingi ikki tilda tuzilgan dasturlar ham har doim birinchi ko`rinishga (obyekt kodi) keltiriladi.

Yuqori darajadagi dasturlash tillari esa, asosan, dasturlash jarayonini tezlashtirish uchun yaratilgan. Shuni eslatib o‘tish lozimki, har qanday dastur bajarilishidan oldin mashina kodiga o‘tkaziladi. Ushbu darajadagi dasturlash tillarida yozilgan dastur ma’lum ma’noli so‘zlardan(odatda ingliz tilidagi) tashkil topadi.

Masalan:

```
1: if (var == 6) {
2: print "Salom";
3: }
```

Ko‘rib turibsizki, dastur qismi ingliz tilidagi ma’noli so‘zlardan tashkil topgan. Hozirgi zamonaviy tillarning barchasi yuqori darajaga mansub.

Kompyuter dunyosida ko‘plab dasturlash tillari mavjud bo‘lib, dasturlash va unga qiziquvchilar soni ortib bormoqda. Bir xil turdagi ishni bajaradigan dasturlarni **Basic**, **Pascal**, **Ci** va boshqa tillarda yozish mumkin. **Pascal, Fortran** va **Kobol** tillari universal tillar hisoblanadi, **Ci** va **Assembler** tillari mashina tiliga ancha yaqin tillar bo‘lib, quyi yoki o'rta darajali tillardir. Algoritmik til inson tillariga qanchalik yaqin bo‘lsa, u tilga yuqori darajali til deyiladi. Mashina tili esa eng pastki darajali tildir. Mashina tili bu sonlardan iboratdir, Masalan: 010110100010101

Quyi darajali dasturlash tili ancha murakkab bo'lib ular juda maxsus sohalarda ishlatiladi va ularning mutaxassislari ham juda kam. Chunki quyi dasturlash tillari (masalan: assembler) ko'pincha miktoprotsessorlar bilan ishlashda kerak bo'lishi mumkin.

Odatda turli dasturlash ishlari uchun yuqori darajali dasturlash tilidan keng foydalaniladi.

EHM (Elektron Hisoblash Mashinasi) endi yuzaga kelgan paytda dastur tuzishda, faqat mashina tillarida, ya'ni sonlar yordamida EHM bajarishi kerak bo'lgan amallarning kodlarida kiritilgan. Bu holda mashina uchun tushinarli sanoq, sistemasi sifatida 2 lik, 6 lik, 8 lik sanoq sistemalari bo'lgan. Dastur mazkur sanoq sistemasidagi sonlar vositasida kiritilgan. Yuqori darajali dasturlashda, mashina tillariga qaraganda mashinaga moslashgan (yo'naltirilgan) belgili kodlardagi tillar hisoblanadi. Belgilar kodlashtirilgan tillarning asosiy tamoyillari shundaki, unda mashina kodlari ularga mos belgilar bilan belgilanadi, hamda xotirani avtomatik taqsimlash va xatolarni tashhis qilish kiritilgan. Bunday mashina moslashgan til - **ASSEMBLER** tili nomini oldi.

Odatda dasturlash yuqori saviyali dasturlash tillari (**Delphi, Java, C++, Python**) vositasida amalga oshiriladi. Bu dasturlash tillarining semantikasi odam tiliga yaqinligi tufayli dastur tuzish jarayoni ancha oson kechadi. Ko'p ishlatiladigan dasturlash tillari. Biz hozir biladigan va ishlatadigan tillarning barchasi shu guruhga mansub. Ular insonga "tushunarli" tilda yoziladi. Ingliz tilini yaxshi biluvchilar dastur kodini qiynalmasdan tushunishlari mumkin. Bu guruhga **Fortran, Algol, C, Pascal, Cobol** va h. k. tillar kiradi(ko`pchiligi hozirda deyarli qo`llanilmaydi). Eng birinchi paydo bo`lgan tillardan to hozirgi zamonaviy tillargacha ishlatish mumkin. Lekin, hozirgi web texnologiya orqali ishlaydigan tillarda(**PHP, ASP. NET, JSP**) bunday dasturlar tuzilmaydi. Chunki bunday dasturlarning ishlashi uchun yana bir amaliy dastur ishlab turishi kerak. Hozirda, amaliy dasturlar, asosan, **Visual C++, C#, Borland Delphi, Borland C++, Java, Phyhon** kabi tillarda tuziladi. O`zbekistonda ko`pchilik Delphi dan foydalanadi. Buning asosiy sababi: soddaligi, komponentlarning ko`pligi, interfeysining tushunarliligi va h. k. Delphida birinchi ishlagan odam ham qanaqadir dastur tuzishi oson kechadi. Lekin, Windows da dasturning asosiy ishlash mohiyatini ancha keyin biladi(komponentlarning ko`pligi va API funksiyalari dasturda ko`rsatilmasligi uchun). Yana bir tarafi, Delphi(Pascal) operativ xotirani tejashga kelganda ancha oqsaydi. Unda o`zgaruvchilarni oldindan e'lon qilib qo`yish evaziga ishlatilmaydigan o`zgaruvchilar va massivlar ham joy olib turadi. Eng keng tarqalgan

dasturlash tili(Windows OS ida) **Microsoft Visual C++** tilidir. Ko`pchilik dasturlar hozirda shu tilda tuziladi. Umuman olganda, C ga o`xshash(C-подобный) tillar hozirda dasturlashda yetakchi. Deyarli hamma zamonaviy tillarning asosida C yotadi. Bundan tashqari, Turli komputer o'yinlari tuzishda yoki kichik hajmdagi dasturlar tayyorlashda **LUA script** yoki **javascript** tillari ham keng ishlatilmoqda.

Biz sizga xozirgi kunda keng tarqalgan **desktop** dasturlashda ishlatiladigan dasturlash tillaridan bazilari haqida aytib o'tamiz:

Delphi (talaff. délfi) — dasturlash tillaridan biri. Borland firmasi tomonidan ishlab chiqarilgan. Delphi dasturlash tili ishlatiladi va avvaldan Borland Delphi paketi tarkibiga kiritilgan. Shu bilan bir qatorda 2003-yildan hozirgacha qo‘llanilayotgan shu nomga ega bo'lgan. Object Pascal — Pascal tilidan bir qancha kengaytirishlar va to‘ldirishlar orqali kelib chiqqan bo‘lib, u ob’yektga yo‘naltirilgan dasturlash tili hisoblanadi. Avvaldan ushbu dasturlash muhiti faqatgina Microsoft Windows amaliyot tizimi uchun dasturlar yaratishga mo'ljallangan, keyinchalik esa **GNU/Linux** hamda **Kylix** tizimlari uchun moslashtirildi, lekin 2002-yilgi **Kylix 3** sonidan so'ng ishlab chiqarish to'xtatildi, ko'p o'tmay esa MicrosoftNET tizimini qo'llab quvvatlashi to'g'risida e'lon qilindi. Lazarus proekti amaliyotidagi (Free Pascal) dasturlash tili Delphi dasturlash muhitida **GNU/Linux, Mac OS X** va **Windows CE** platformalari uchun dasturlar yaratishga imkoniyat beradi.

Visual Basic (talaffuzi: "Vijual Beysik") – Microsoft korporatsiyasidan dasturlash tili va uning uchun dasturlash muhitdir. U BASICdan ko`p tushunchalar oldi va tez rasmli interfeys bilan dasturlar taraqqiyot ta`minlaydi. Oxirgi versiya 6. 0 1998 yilda reliz kelishdi. Microsoftdan voris Visual BasicNET 2002 yilda paydo bo`ldi.

Java dasturlash tili - eng yaxshi dasturlash tillaridan biri bo'lib unda korporativ darajadagi mahsulotlarni(dasturlarni) yaratish mumkin. Bu dasturlash tili **Oak** dasturlash tili asosida paydo bo'ldi. Oak dasturlash tili 90-yillarning boshida Sun Microsystems tomonidan platformaga(Operatsion tizimga) bog'liq bo'lmagan holda ishlovchi yangi avlod aqlli qurilmalarini yaratishni maqsad qilib harakat boshlagan edi. Bunga erishish uchun Sun hodimlari **C++** ni ishlatishni rejalashtirdilar, lekin ba'zi sabablarga ko'ra bu fikridan voz kechishdi. Oak muvofaqiyatsiz chiqdi va

1995-yilda Sun uning nomini **Java** ga almashtirdi, va uni WWW rivojlanishiga hizmat qilishi uchun ma'lum o'zgarishlar qilishdi. Java Obyektga Yo'naltirilgan Dasturlash(**OOP-object oriented programming**) tili va u C++ ga ancha o'xshash. Eng ko'p yo'l qo'yiladigan xatolarga sabab bo'luvchi qismlari olib tashlanib, Java dasturlash tili ancha soddalashtirildi. Java kod yozilgan fayllar(*. java bilan nihoyalanuvchi) kompilatsiyadan keyin bayt kod(bytecode) ga o'tadi va bu bayt kod interpretator tomonidan o'qib yurgizdiriladi.

C++ (talaffuzi: si plyus plyus) — turli maqsadlar uchun mo‘ljallangan dasturlash tili. 1979-yili Bell Labsda Biyarne Stroustrup tomonidan C dasturlash tilining imkoniyatlarini kengaytirish va OOP(object Oriented Programming) xususiyatini kiritish maqsadida ishlab chiqarilgan. Boshida „**C with Classes**“ deb atalgan, 1983-yili hozirgi nom bilan ya’ni C++ deb o‘zgartirilgan. C++ C da yozilgan dasturlarni kompilyatsiya qila oladi, ammo C kompilyatori bu xususiyatga ega emas. C++ tili operatsiyon tizimlarga aloqador qisimlarni, klient-server dasturlarni, EHM o‘yinlarini, kundalik ehtiyojda qo‘llaniladigan dasturlarni va shu kabi turli maqsadlarda ishlatiladigan dasturlarni ishlab chiqarishda qo‘llaniladi.

Quyidagi jadvalda dasturlash tillari haqida ma’lumotlar keltirilgan.

Til	Yaratilgan yili	Mualliflar	Tashkilot, firma
Ada	1979-80	Jean Ichbian	Cii-Honeywell (Fransiya)
Algol	1960		International Commitee
ARL	1961-1962	Kenneth Iverson, Adin Falkoff	IBM
DELPHI	1995	Borland	VASIS,
Beysik	1964-1965	JohnKemeny, Thomas Kurtz	Dartmouth Colleje
C	1972-1973	Dennis Ritchie	Bell Laboratories
C++	1980	Bjarne Strostrup	Bell Laboratories
Kobol	1959-1961	Grace Murray	Hopper
Fort	1971	Charles H. Moore	
FORTRAN	1950-1958	John Backus	IBM
HTML	1989	im Berners-Li CERN,	Jeneva LISP,
LISP	1956-1960	John MC	Carthy
LOGO	1968-70	Seymour Papert	Massachusetts Institute of Techn.

Pascal	1967-1971	Niklaus Wirth	Federal Institute of Technology (SHveysariya)
PL1	1964-1966		
PROLOG	1978	Alan Kalmeroe	
SIMULA	1967	Ole-Yoxan Dal, Kristen Nigaard	Norvegiya XM
Java	1995	Djeyms Gosling Sun	Microsystems

Dasturlash tillari komputerda bajarilishiga qarab kompilyatsiya qilinuvchi va interpretatsiya qilinuvchi tillarga bo‘linadi.

Kompilyatsiya qilinuvchi dasturlash tillarida dastur kodi kompilyator tomonidan mashina kodiga o‘tkaziladi. Operatsion tizim(OT) esa, shu kodni to‘g‘ridan-to‘g‘ri ishlataveradi. Kompilyatsiya jarayoni komputer protsessori va OT talablariga mos ravishda amalga oshiriladi. Shuning uchun, bir OT uchun kompilyatsiya qilingan dasturning mashina kodi ikkinchi OT da ishlamaydi. Ushbu turdagi tillarga quyidagilarni misol qilib keltirishimiz mumkin: C, C++, Pascal va h. k.

Microsoft Windows OTlarida kompilyatsiya qilingan dastur nomi *. exe ko‘rinishidagi fayl bo‘ladi. Linux, Unix(va shularning davomchilari) kabi OT larda esa fayl kengaytmasining ahamiyati yo‘q.

Kompilyatsiya qilinuvchi dasturlash tillarining asosiy yutuqlaridan biri — u OT dan boshqa biror dastur yoki kutubxona(Library, mas. DLL) o‘rnatishni talab qilmaydi. Bundan tashqari, interpretatsiya qilinuvchi tillarga nisbatan ancha tez ishlaydi.

Interpretatsiya qilinuvchi dasturlash tillarida tuzilgan dastur kodi kompilyatsiya qilinmaydi. Ushbu turdagi dasturni ishlatishdan oldin dastur kodi interpretatsiya qilinadi. Interpretatsiya qilinuvchi dasturlash tillarida tuzilgan dastur mos interpretator o‘rnatilgan komputerlardagina ishlaydi. Ushbu turdagi dasturlash tillariga PHP, Python, Ruby kabi tillar kiradi.

Interpretatsiya qilinuvchi dasturlash tillari kompilyatsiya qilinuvchilaridan, asosan, yozilgan dasturning deyarli hamma platformalarda ishlashi bilan ajralib turadi. Dastur biror turdagi OT yoki protsessor uchun yozilmaydi — faqat interpretatorgina turli platformalar uchun yoziladi.

Interpretatsiya qilinuvchi dastur kodi bajarilishidan oldin interpretator tomonidan oraliq kodga “kompilyatsiya” qilinadi. Shu oraliq kod interpretator tomonidan bajariladi. Python kabi tillar oraliq kodni saqlab qo‘yadi, dastur kodi o‘zgarmaguncha shu oraliq kodni ishlatadi.

Dastur biror masalani echishda elektron hisoblash mashinalari bajarishi lozim bo’lgan amallarning izchil tartibidan iborat. EHM uchun dastur tuzish jarayoni dasturlash deyiladi. Dasturlash echilishi kerak bo’lgan masala algoritmini EHM tiliga, ya’ni «mashina tili»ga o’tkazishdir. EHM uchun dastur tuzish – masalani echish usulini mashina buyruqlarining shunday majmui (dasturi)ga, keltirish demakki, bu buyruqlar xotiraga joylashib, tartib bilan amalga oshadi va tegishli hisoblashlarni bajaradi.

DASTUR biror masalani echishda elektron hisoblash mashinalari bajarishi lozim bo’lgan amallarning izchil tartibidan iborat.

EHM uchun dastur tuzish jarayoni **DASTURLASH** deyiladi.

KOMPYUTERDA MASALA YECHISH BOSQICHLARI (KMEB).

1. Masalaning qo’yilishi- bunda berilgan masalaning to’g’ri qo’yilganligi va uni yechish uchun qanday ma’lumotlar kerakligi va qanday natija olinishini bilish kerak.
2. Masalaning matematik modelini tuzish-bu bosqichda matematik formula tanlanadi
   matematik modeli tuziladi.
3. Masalani yechishni sonli usuli-bu bosqichda Hosil qilingan mat-k masalaning yechish usuli Tanlanadi va buning uchun sonli usuldan foydalaniladi.
4. Algoritmini tuzish-bunda mashina tushunadigan biror bir algoritmik tilda dG’tuzish.
5. Dasturni kiritish va chiqarish. Mavjud muhitdan foydalangan xolda.
6. Kiritilgan dastur natijaga erishishi.
7. Natijani tahlil qilish.

Mavzuga oid savollar

1. Dastur nima?
2. Dasturlash tillarining qanday turlarini bilasiz?
3. Dasturlash tillari va ularning klassifikasiyasi.
4. Mashinaga mo‘ljallangan va proseduraga mo‘ljallangan dasturlash tillari haqida ma’lumot bering.
5. Yuqori darjali dasturlash tillariga misol keltiring
6. Interpretatorlar va kompilyatorlar nima?
7. Dasturlarni translyasiyalash deganda nimani tushunasiz?

1.2. OB’EKTGA YO‘NALTIRILGAN DASTURLASH TILLARI.

Reja:

1. Ob’ektga yo‘naltirilgan dasturlash tillari tarixi.
2. Proseduraviy, strukturaviy va obyektlarga mo’ljallangan dasturlash.

Tayanch iboralar: Ob’ektga yo‘naltirilgan dasturlash tillari. Dasturlashning ob’ektga yo‘naltirilgan paradigmasi.

Ob’ektga yo‘naltirilgan dasturlash tillari tarixi

Obyektga mo’ljallangan yondoshuv (OMYo) bir kunda o’ylab topilgan emas. Uning paydo bo’lishi dasturiy ta’minotning tabiiy rivojidagi navbatdagi pog’ona xolos. Vaqt o’tishi bilan qaysi uslublar ishlash uchun qulay-u, kaysinisi noqulay ekanini aniqlash oson bo’lib bordi. OMYo eng muvaffaqiyatli, vaqt sinovidan o’tgan uslublarni o’zida samarali mujassam etadi.

Dastlab dasturlash anchayin boshqotirma ixtiro bo’lib, u dasturchilarga dasturlarni kommutasiya bloki orqali kompyuterning asosiy xotirasiga to’g’ridan-to’g’ri kiritish imkonini berdi. Dasturlar mashina tillarida ikkilik kurinishda yozilar edi. Dasturlarni mashina tilida yozishda tez-tez xatolarga yo’l qo’yilar edi, eng ustiga ularni tuzilmalashtirish imkoni bo’lmagani tufayli, kodni kuzatib borish amalda deyarli

mumkin bo’lmagan hol edi. Bundan tashqari, mashina kodlaridagi dastur tushunish uchun g’oyat murakkab edi.

Vaqt o’tishi bilan kompyuterlar tobora kengroq qo’llana boshladi hamda yuqoriroq darajadagi prosedura tillari paydo bo’ldi. Bularning dastlabkisi FORTRAN tili edi. Biroq obyektga mo’ljallangan yondoshuv rivojiga asosiy ta’sirni keyinroq paydo bo’lgan, masalan, ALGOL kabi prosedura tillari ko’rsatdi. Prosedura tillari dasturchiga axborotga ishlov berish dasturini pastroq darajadagi bir nechta proseduraga bo’lib tashlash imkonini beradi. Pastroq darajadagi bunday proseduralar dasturning umumiy tuzilmasini belgilab beradi. Ushbu proseduralarga izchil murojaatlar proseduralardan tashkil topgan dasturlarning bajarilishini boshqaradi.

Dasturlashning bu yangi paradigmasi mashina tilida dasturlash paradigmasiga nisbatan ancha ilg’or bo’lib, unga tuzilmalashtirishning asosiy vositasi bo’lgan proseduralar qo’shilgan edi, Maydaroq funksiyalarni nafaqat tushunish, balki sozlash ham osonroq kechadi. Biroq, boshqa tomondan, prosedurali dasturlash koddan takroran foydalanish imkonini cheklab qo’yyadi. Buning ustiga dasturchilar tez-tez «makaron» dasturlar ham yozib turishganki, bu dasturlarni bajarish likopdagi spagetti uyumini ajratishga o’xshab ketar edi. Va, nihoyat, shu narsa aniq bo’ldiki, prosedurali dasturlash usullari bilan dasturlarni ishlab chiqishda diqqatni ma’lumotlarga qaratishning o’zi muammolarni keltirib chiqarar ekan. Chunki ma’lumotlar va prosedura ajralgan, ma’lumotlar inkapsullanmagan. Bu nimaga olib keladi? Shunga olib keladiki, har bir prosedura ma’lumotlarni nima qilish kerakligini va ular qayerda joylashganini bilmog’i lozim bo’ladi. Agar prosedura o’zini yomon tusa-yu, ma’lumotlar ustidan noto’g’ri amallarni bajarsa, u ma’lumotlarni buzib qo’yishi mumkin. Har bir prosedura ma’lumotlarga kirish usullarini dasturlashi lozim bo’lganligi tufayli, ma’lumotlar taqdimotining o’zgarishi dasturning ushbu kirish amalga oshirilayotgan barcha o’rinlarining o’zgarishiga olib kelar edi. Shunday qilib, xatto eng kichik to’g’rilash ham butun dasturda qator o’zgarishlar sodir bo’lishiga olib kelar edi.

Modulli dasturlashda, masalan, Modula2 kabi tilda prosedurali dasturlashda topilgan ayrim kamchiliklarni bartaraf etishga urinib ko’rildi. Modulli dasturlash

dasturni bir necha tarkibiy bo’laklarga, yoki, boshqacha qilib aytganda, modullarga bo’lib tashdlaydi. Agar prosedurali dasturlash ma’lumotlar va prosedsralarni bo’lib tashlasa, modulli dasturlash, undan farqli o’laroq, ularni birlashtiradi. Modul ma’lumotlarning o’zidan hamda ma’lumotlarga ishlov beradigan proseduralardan iborat. Dasturning boshqa qismlariga moduldan foydalanish kerak bo’lib qolsa, ular modul interfeysiga murojaat etib qo’yaqoladi. Modullar barcha ichki axborotni dasturning boshqa qismlarida yashiradi.

Biroq modulli dasturlash ham kamchiliklardan holi emas. Modullar kengaymas bo’ladi, bu degani kodga bevosita kirishsiz hamda uni to’g’ridan-to’g’ri o’zgartirmay turib modulni qadamma-qadam uzgartirish mumkin emas. Bundan tashqari, bitta modulni ishlab chiqishda, uning funksiyalarini boshqasiga o’tkazmay (delegat qilmay) turib boshqasidan foydalanib bo’lmaydi. Yana garchi modulda turni belgilab bo’lsa-da, bir modul boshqasida belgilangan turdan foydalana olmaydi.

Modulli va prosedurali dasturlash tillarida tuzilmalashtirilgan va tuzilmalashtirilmagan ma’lumotlar o’z «tur»iga ega. Biroq turni kengaytirish usuli, agar «agregatlash» deb ataluvchi usul yordamida boshqa turlarni yaratishni hisobga olmaganda, mavjud emas.

Va, nihoyat, modulli dasturlash – bu yana proseduraga mo’ljallangan gibridli sxema bo’lib, unga amal qilishda dastur bir necha proseduralarga bo’linadi. Biroq endilikda proseduralar ishlov berilmagan ma’lumotlar ustida amallarni bajarmaydi, balki modullarni boshqaradi.

Obyektga mo’ljallangan dasturlash (OMD) modulli dasturlashdan keyingi mantiqiy pog’onani egallaydi, u modulga nasldan-naslga o’tishni va polimorfizmni qo’shadi. OMD dan foydalanr ekan, dasturchi dasturni bir qator oliy darajali obyektlarga bo’lish yo’li bilan tizimlashtiradi. Har bir obyekt hal qilinayotgan muammoning ma’lum bir tomonini modellashtiradi. OMD endilikda dasturni bajarish jarayonini boshqarish uchun dasturchi diqqatini proseduralarni ketma-ketlikda chaqirib olish ro’yxatini tuzib o’tirishga qaratmaydi. Buning o’rniga obyektlar o’zaro aloqada bo’ladi. OMYo yordamida ishlab chiqilgan dastur hal qilinayotgan muammoning amaldagi modeli bo’lib xizmat qiladi.

Dasturga obyektlar atamalari bilan ta’rif berish dasturiy ta’minotni ishlab chiqishning eng tushunarli usulidir. Obyektlar hamma narsani obyekt nima qilayotgani nuqtai nazaridan idrok etishga, ya’ni uning hatti-xarakatlarini hayolan modellashtirishga majbur qiladi. Shu tufayli obyektga yondoshishda u dasturning bajarilishi jarayonida qanday ishlatiladi degan nuqtai nazardan biroz e’tiborni chalg’itish mumkin. Shunday qilib, dasturni yozish jarayonida haqiqiy dunyoning tabiiy atamalaridan foydalanish mumkin. Dasturni alohida proseduralar va ma’lumotlar shaklida (kompyuter dunyosi atamalarida) qurish o’rniga, obyektlardan iborat dastur qurish mumkin. Obyektlar otlar, fe’llar va sifatlar yorlamida haqiqiy dunyoni dasturda modellashtirishga imkon beradi. Joriy qilish (realizasiya) hatti-xarakatlar qanday bajarilayotganini belgilaydi. Dasturlash atamalarida joriy qilish – bu dasturiy kod.

Yechilayotgan masala atamalari bilan fikrlab, joriy qilishning mayda-chuyda detallarida o’ralashib qolish havfidan qochish mumkin. Albatta, ayrim oliy darajadagi obyektlar kompyuter bilan aloqa qilishda past darajadagi, mashinaga mo’ljallangan usullardan foydalanishi lozim. Biroq obyekt bu aloqani tizimning boshqa qismlaridan izolyasiya qiladi.

Obyekt dastur konsturksiyasi bo’lib, unda holat va hatti-xarakat inkapsulalangan bo’ladi. Obyekt holati bu ichki obyekt o’zgaruvchanlari qiymatlarining yig’indisidir.

Ichki o’zgaruvchan deb obyekt ichida saqlanadigan qiymatga aytiladi.

Mohiyat e’tibori bilan, obyekt bu sinfning ekzemplyari (nushalaridan biri)dir.

OMD, haqiqiy dunyo kabi, obyektlardan tashkil topadi. Obyektga mo’ljallangan sof dasturlash tilida, eng dastlabki, bazaviy, butun, mantiqiy turlardan tortib, to sinflarning murakkabroq nushalarigacha, barchasi obyekt hisoblanadi. Biroq obyektga mo’ljallangan tillarning hammasi ham bu darajada chuqurlashib ketmagan. Ayrim tillarda (masalan, Java kabi) int va float ga o’xshash oddiy primitivlar obyekt sifatida olib qaralmaydi.

OMD obyektlari, haqiqiy olam obyektlari kabi, o’z xususiyatlari va xatti-harakatlari bo’yicha tasniflanadi.