1-TOM, 12-SON

DASTURIY MODULNI ISHLAB CHIQISH TARTIBIGA DOIR AYRIM MULOHAZALAR

Buriyev Bobur Erkinovich

ALFRAGANUS UNIVERSITY Nodavlat oliy ta'lim tashkiloti Raqamli texnologiyalari kafedrasi o'qituvchisi

Annotatsiya: Ushbu maqolada dasturiy modulni ishlab chiqish tartibiga doir ayrim mulohaza va fikrlar yuritilgan. Unda dasturlash tilini tanlash, berilgan tuzilmani aniqlash, modulni dasturlash, modul matnini silliqlash, tekshirish va kompilyatsiyalash borasida tahlillar asosida ma'lumotlar berilgan.

Kalit so'zlar: modul dastur, spetsifikatsiya, algoritm, kompilyatsiyalash, assembler tili, funksional tuzilma

Abstract: This article contains some comments and thoughts on the procedure for developing a software module. It provides analysis-based information on choosing a programming language, determining a given structure, programming a module, smoothing the text of a module, checking and compiling.

Key words: modular program, specification, algorithm, compilation, assembly language, functional structure

Kirish

Dasturiy modulni ishlab chiqishda quyidagi tartibga rioya qilish maqsadga muvofiq:

- modul spetsifikatsiyasini o`rganish va tekshirish, dasturlash tilini tanlash;
- berilganlarning algoritmini va tuzilmasini tanlash;
- modulni dasturlash (kodlash);
- modul matnini silliqlash;
- modulni tekshirish;
- modulni kompilyatsiyalash.

Dasturiy modulni ishlab chiqishning birinchi qadami ko`proq dastur tuzilmasini pastdan aralash nazorat qilishni ifodalaydi: ishlab chiquvchi modulning o`ziga xos xususiyatlarini o`rgana turib, bu xususiyatlar unga tushunarli va u shu modulni ishlab chiqish uchun yetarli ekanligiga ishonch hosil qilishi kerak. Bu qadam oxirida dasturlash tili tanlanadi: u butun DV uchun oldindan aniqlangan bo`lsa ham, ba`zi bir hollarda, mazkur modulni amalga oshirish uchun ko`proq to`g'ri keladigan boshqa til (agar dasturlash tizimi bunga yo`l qo`ysa), masalan, assembler tili, tanlanishi mumkin.

Asosiy qism

Dasturiy modulni ishlab chiqishning ikkinchi qadamida qo`yilgan masalani yoki unga yaqin bo`lgan masalani echishning qandaydir bir algoritmlarining mavjud-mavjudmasligini aniqlash zarur. Agar to`g'ri keladigan shunday algoritm topilsa, undan foydalanish maqsadga muvofiq bo`ladi. Modulning o`z funktsiyalarini bajarishda, ishlatiladigan ma`lumotlarning to`g'ri keluvchi tuzilmasini tanlash, ishlab chiqilayotgan modulning mantig'ini va sifat ko`rsatkichlarini ma`lum darajada oldindan aniqlaydi, shuning uchun unga juda ham mas`uliyatli qaror deb qarash kerak.

Uchinchi qadamda tanlangan dasturlash tilida modul matnini tuzish amalga oshiriladi. Modul spetsifikatsiyasida ko`rsatilgan funktsiyalarni amalga oshirishda hisobga olinishi kerak bo`lgan detallarning ko`pligi juda ham chalkashib ketgan, bir talay xatoliklar va noaniqliklarga ega bo`lgan matnni yaratishga osonlik bilan olib kelishi mumkin. Bunday modulda xatolarni izlash va unga talab qilingan o`zgarishlarni kiritish juda ham ko`p mehnat talab qiladigan masaladir. SHuning uchun modul matnini tuzishda texnologik asoslangan va amalda tekshirilgan dasturlash fanidan foydalanish juda muhimdir. Birinchi marta bu narsaga Deykstr, tuzilmali dasturlashning asosiy tamoyillarini tuzib va ularni asoslab, e`tiborni qaratgan.

Modulni ishlab chiqishning navbatdagi qadami modul matnini, DV sifat spetsifikatsiyasiga mos tarzda, tugallangan ko`rinishga olib kelish bilan bog'liq. Ishlab chiquvchi modulni dasturlashda, izohlarni oxirigacha ishlab chiqmasdan va dastur uslubiga qo`yilgan ba`zi bir talablarning buzilishiga yo`l qo`ygan holda, asosiy e`tiborni modul funktsiyasini to`g'ri amalga oshirilishiga qaratadi. Modul matnini silliqlash paytida, u matnda mavjud bo`lgan izohlarni tahrirlashi va unga, talab qilingan sifatni ta`minlash maqsadida, qo`shimcha izohlarni kiritishi mumkin. Shu maqsadda stilistik talablarni bajarish uchun dastur matni tahrirlanadi.

Modulni tekshirish qadami, modulning ichki mantig'ini, uning yaxshilanishi (uning kompyuterda bajarilishidan foydalanuvchi) boshlanguncha, qo`lda tekshirishni ifodalaydi, dasturlashning, DV ishlab chiqishning har bir bosqichida qabul qilinadigan qarorlarini nazorat qilishning zarurligi haqidagi texnologiyasini muhokama qilish uchun tuzilgan umumiy tamoyilni amalga oshiradi.

Modulni ishlab chiqishning oxirgi qadami modulni tekshirish tugaganligini (kompilyator yordamida) va modulni yaxshilash jarayoniga o`tilganligini bildiradi.

Modulni dasturlashda, dastur nafaqat kompilyator uchun, balki odam uchun ham tushunarli bo`lishi kerakligini e`tiborga olish kerak: modulni ishlab chiquvchi ham, modulni tekshiruvchi shaxslar ham, modulni yaxshilash uchun testlar tayyorlayotgan test tuzuvchilar ham, modulning talab qilingan o`zgarishlarini amalga oshirayotgan

dasturiy vosita yo`ldoshchilari ham modul ishi mantig'ini juda ko`p martalab ko`rib chiqishga majburdirlar. Zamonaviy dasturlash tillarida bu mantiqni chalkashtirib yuborish va oqibatda modulni inson uchun tushunmaydigan, buning natijasida esa, uni ishonchsiz yoki qiyin yuritiladigan qilish uchun vositalar yetarli. Shu sababli to`g'ri keluvchi til vositalarini tanlash uchun zaruriy choralar ko`rish, va dasturlashning ma`lum faniga rioya qilish zarur. Shunga bog'liq ravishda Deykstr dasturni, dasturning ishlash mantig'ini tushunishni yaxshilashga imkon beruvchi, boshqaruvchi konstruktsiyalarning bir necha turlarining kompozitsiyasi sifatida qurishni taklif qildi. Faqat shunday konstruktsiyalardan foydalangan dasturlash ***tuzilmali*** deyiladi.

Tuzilmali (strukturaviy) dasturlashning asosiy konstruktsiyalariga: *izma-iz borish, tarmoqlanish va takrorlanish* kiradi. Bu konstruktsiyalarning tashkil etuvchilariga umumlashtirilgan operatorlar (qayta ishlash tugunlari) S, S1, S2 va shart (predikat) kiradi. Umumlashtirilgan operator sifatida dasturlash tilining ishlatilayotgan ixtiyoriy oddiy operatori (o`zlashtirish, kiritish, chiqarish operatorlari, protseduraga murojaat), yoki tuzilmali dasturlashning asosiy boshqaruvchi konstruktsiyalarining kompozitsiyasi bo`lgan dasturning bir qismi ish ko`rishi mumkin. Bu kompozitsiyalarning har biri boshqarishi bo`yicha bitta kirish va bitta chiqishga ega. Shunga bog'liq ravishda umumlashtirilgan operator ham faqat bitta kirish va bitta chiqishga ega bo`ladi.

Tuzilmali dasturlash ba`zan "*GO TO siz dasturlash*" deb ham aytiladi. Ammo bu yerda gap GO TO operatorida emas, balki uning tartibsiz ishlatilishidadir. Ko`pincha tuzilmali dasturlashni qo`llashda, ba`zi bir dasturlash tillarida (masalan, FORTRAN da) o`tish (GO TO) operatori tuzilmali konstruktsiyalarni amalga oshirish uchun ishlatiladi, bu tuzilmali dasturlash tamoyillarini hech ham buzmaydi. Dasturni "tuzilmasiz" o`tish operatorlari, ayniqsa modul matnida bajarilayotgan o`tish operatoridan yuqorida (oldinda) joylashgan operatorga o`tish chalkashtirib yuboradi. Shunga qaramay, o`tish operatoridan foydalanishdan qochish, ba`zi bir oddiy hollarda juda ulkan tuzilmali dasturlarni tuzishga olib keladi, bu ularni tushunishga aniqlik kiritmaydi, aksincha, modul matnida qo`shimcha xatolarning paydo bo`lish xavfiga olib keladi. Shuning uchun o`tish operatoridan mumkin bo`lgan hamma joylarda ishlatishdan qochish kerak, ammo bu dasturga aniqlik kiritish hisobiga bo`lmasligi kerak.

Tuzilmali dasturlash modul matni qanday bo`lishi haqida tavsiyalar beradi. Dasturchi shunday matnni tuzishi uchun qanday yo`l tutishi kerak, degan savol tug'iladi. Ko`pincha modulni dasturlash, uning ishlash mantig'ining umumlashgan chizig'ini bayon etuvchi blok-sxemasini tuzish bilan boshlanadi. Ammo zamonaviy

dasturlash texnologiyasi, bu ishga to`g'ri keladigan kompyuter quvvatlovisiz, buni qilishni tavsiya etmaydi. Blok-sxemalar modulning ishlash mantig'ini ko`rgazmali qilib ifodalashga imkon bersa ham, ularni dasturlash tilida qo`lda kodlashda xatolarning o`ziga xos manbasi paydo bo`ladi: ikki o`lchamli tuzilmalarni (modulni ifodalovchi chiziqli matnga tuzilgan blok-sxema) ifodalashda modul ishi mantig'ini buzish xavfi mavjud, uning takroriy qarab chiqilishida, yuqori darajadagi e`tiborni psixologik saqlash juda qiyin bo`ladi. Bundan quyidagi hol mustasno: blok-sxemani tuzishda grafik redaktordan foydalanilgan va u bo`yicha matn dasturlash tilida avtomatik ravishda generirlanadigan (xuddi, masalan, R-texnologiyada qilingani kabi) darajada shakllantirib tuzilgan.

Dasturlashning zamonaviy texnologiyasi modulning matnini tuzishning asosiy metodi sifatida *qadamba-qadam bo`laklashni* tavsiya etadi. Modul matnini ishlab chiqish jarayonini bir qator qadamlarga bo`lish mazkur metodning mohiyatini tashkil qiladi.

Birinchi qadamda modul ishining umumiy sxemasi chiziqli matn shaklida (ya`ni yirik tushunchalardan foydalanib) bayon etiladi, bunda bu bayon to`liq shakllantirilmagan va insonning qabul qilishiga yo`naltirilmagan bo`ladi. Har bir keyingi qadamda, oldingi qadamlardan birida ishlab chiqilgan qandaydir bir bayonda tushunchalardan birining aniqlanishi va bo`laklanishi (uni biz aniqlanadigan, deb ataymiz) amalga oshiriladi. Bu qadam natijasida tanlangan aniqlangan tushunchaning bayoni yoki bazaviy dasturlash tilining iboralarida (ya`ni, modulni ifodalash uchun tanlangan), yoki birinchi qadamdagi kabi yangi aniqlanadigan tushunchalardan foydalanib, yaratiladi. Hamma aniqlanayotgan tushunchalar aniqlanib bo`lingach (ya`ni, oxirida dasturlashning bazaviy tilida ifodalangan), bu jarayon to`xtaydi. Dasturlashning bazaviy tilida modulning matnini, hamma aniqlanadigan tushunchalar ularning berilgan bayonlari bilan va tuzilmali dasturlash konstruktsiyalarining ifodalarini shu dasturlash tili vositalari bilan almashtirish yo`li bilan olish oxirgi qadam bo`ladi.

Qadamba-qadam bo`laklash ko`rsatilgan bayonlarni ifodalash uchun, ***psevdokod*** deb nomlangan, qisman shakllangan tildan foydalanishga bog'liq. Bu til tuzilmali dasturlashning hamma konstruktsiyalaridan foydalanishga imkon beradi, ular rasmiy bo`lmagan bo`laklar bilan birga umumlashtirilgan operatorlar va shartlarni tabiiy tilda ifodalash uchun rasmiylashtiriladi. Umumlashtirilgan operatorlar va shartlar sifatida dasturlashning bazaviy tilida muvofiq bo`laklar ham berilishi mumkin.

Xulosa

Modulning bazaviy dasturlash tilida tashqi rasmiylashtirilishini psevdokoddagi bosh bayon deb hisoblash mumkin, u quyidagilarga ega bo`lishi kerak:

- bazaviy tilda yozilgan modul boshiga, ya`ni bu modulning birinchi gapi yoki sarlavhasi (spetsifikatsiyasi);
- bazaviy tilda yozilgan bayonlar bo`limi (to`plami), bunda protsedura va funktsiyalarning bayoni o`rniga ularning faqat tashqi rasmiylashtirilgan ko`rinishi keltiriladi;
- modul tanasidagi operatorlar ketma-ketligining bitta umumlashtirilgan operator sifatida norasmiy ifodalanishi (pastga qaralsin), shuningdek protsedura va funktsiyaning har bir bayoni tanasining bitta umumlashtirilgan operator sifatida norasmiy ifodalanishi;
- modul so`ngi gapining (oxirining) bazaviy tilda yozilishi.

Umumlashtirilgan operatorning psevdokoddagi norasmiy ifodalanishi tabiiy tilda, uning mazmunini umumiy holda ochib beruvchi ixtiyoriy gap bilan amalga oshiriladi. Bunday ifodalashni rasmiylashtirish uchun yagona rasmiy talab quyidagidan iborat: bu gap bitta yoki bir nechta grafik (chop etilgan) satrlarni egallashi va nuqta bilan tugashi kerak (yoki shuning uchun ajratilgan boshqa biror belgi bilan).

Foydalanilgan adabiyotlar:

1. Закревский А. Д. Параллельные алгоритмы логического управления. М.: Эдиториал УРСС, 2003. 200 с.

2. Wagner F., Schmuki R., Wagner T., Wolstenholme P. Modeling software with finite state machines: a practical approach // Auerbach Publications. 2006. 390 p.

3. Harel D. Statecharts: A visual formalism for complex systems // Science of computer programming. 1987. Vol. 8.3. P. 231–274.

4. Зюбин В. Е. Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. 2005. № 11. С. 31–35.

5. Лях Т. В., Зюбин В. Е., Сизов М. М. Опыт применения языка Reflex при автоматизации Большого солнечного вакуумного телескопа // Промышленные АСУ и контроллеры. 2016. № 7. С. 37–43.

6. Зюбин В. Е. К пятилетию стандарта IEC 1131-3. Итоги и прогнозы // Приборы и системы.

7. Гаранина Н. О., Зюбин В. Е., Лях Т. В. Онтологический подход к организации шаблонов требований в рамках системы поддержки формальной верификации программных систем. // Системная информатика. 2017. № 9. С. 111–132.

8. Shilov N. V., Garanina N. O. Combined Logics of Knowledge, Time, and Actions for Reasoning about Multi-agent Systems. Knowledge Processing and Data Analysis // Lecture Notes in Computer Science. 2011. Iss. 6581. P. 48–58.

9. Шелехов В. И. Верификация и синтез программ сложения на базе правил корректности операторов // Моделирование и анализ информационных систем. 2010. Т. 17, № 4. С. 101–110.

10. Clarke E. M., Gao S. Model checking hybrid systems // International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Lecture Notes in Computer Science. 2014. Iss. 8803. P. 385–386.

11. Clarke E. M., Grumberg O., Peled D. Model checking. MIT Press, 1999. 314 p.

12. Clarke E. M., Emerson A., Sistla P. Automatic verification of finite-state concurrent systems using temporal logic specifications // ACM Transactions on Programming Languages and Systems. Vol. 8. Iss. 2. 1986. P. 244–263.