O’ZBEKISTON RESPUBLIKASI ALOQA, AXBOROTLASHTIRISH VA TELEKOMMUNIKATSIYA TEXNOLOGIYALARI DAVLAT QO’MITASI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI SAMARQAND FILIALI

Qo’l yozma huquqida

UDK **004.047**

AXMEDOVA AZIZA ERKINOVNA

Predmetli orientirl axborot tizimida ma’lumotlarni intellektual taxlil qilish modellarini yaratish va tahlil qilish

5A330201 – Kompyuter tizimlari va ularning dasturiy ta’minoti (tarmoqlar bo’yicha)

Magistr
akademik darajasini olish uchun yozilgan
dissertatsiya

Ilmiy rahbar:
Dots. Ya.Yorbekov

SAMARQAND – 2014

M U N D A R I J A

KIRISH

O’zbekiston Respublikasi prezidentining “Kompyuterizatsiyalash va axborot kommunikatsion texnologiyalarni tadbiq etishni rivojlantirish” to’g’risidagi 2002 yil 30 maydagi farmonida [1], hamda vazirlar maxkamasi qarorlarida [2-3] axborotlarni ishlashni takomillashtirishga jiddiy axamiyat berish ko’zda tutilgan. Bitiruv malakaviy ishimiz ham asosan shu muammoga bog’ishlangan. Prezident Islom Karimov aytganidek “Kommunikatsiya tizimlarini rivojlantirish haqida gapirganda, yuqori texnologik telekommunikatsiya tarmog’ini rivojlantirish biz uchun muhim strategik ahamiyatha ega ekanini alohida qayd etish darkor, bugungi kunda hayotimizni kompyuter texnikasi, axborot texnologiyalari, Internet, mobil telefon aloqasisiz tasavvur qilib bo’lmaydi”[4].

“Bugun biz tez sur’atlar bilan o’zgarib boraytgan, insoniyat hozirga qadar boshdan kechirgan davrlardan tubdan farq qiladigan o’ta shiddatli va murakkab bir zamonda yashamoqdamiz. Davlat va siyosat arboblari, faylasuflar va jamiyatshunos olimlar, sharhlovchi va jurnalistlar bu davrni turlicha ta’riflab, har xil nomlar bilan atamoqda. Kimdir uni yuksak texnologiyalar zamoni desa, kimdir tafakkur asri, yana birov yalpi axborotlashuv davri sifatida izohlamoqda.

Bugungi kunda zamonaviy axborot maydonidagi harakatlar shu qadar tig’iz, shu qadar tezkorki, endi ilgaridek, ha, bu voqea bizdan juda olisda yuz beribdi, uning bizga aloqasi yo’q, deb beparvo qarab bo’lmaydi. Ana shunday kayfiyatga berilgan xalq yoki millat taraqqiyotdan yuz yillar orqada qolib ketishi hech gap emas”[5].

Prezident Islom Karimovning 2008 yilda boshlangan jahon moliyaviy inqirozi haqida bildirgan fikrlarini eslaylik -“Bugungi kunning eng dolzarb masalasi – bu 2008 yilda boshlangan jahon moliyaviy inqirozi, uning ta’siri va salbiy oqibatlari, yuzaga kelayotgan vaziyatdan chiqish yo’llarini izlashdan iborat. Inqirozga qarshi choralar dasturining konkret bo’limlari-belgilangan kompleks chora-tadbirlar quyidagi asosiy vazifalarni hal etishga qaratilgan.

Birinchdan- korxonalarni modernizatsiya qilish, texnik va texnologik qayta jihozlashni yanada jadallashtirish, zamonaviy, moslashuvchan texnologiyalarni kehg joriy etish;

Ikkinchidan- joriy kon'yuktura keskin yomonlashib borayotgan hozirgi sharoitda eksportga mahsulot chiqaradigan korxonalarning tashqi bozorlarda raqobatdosh bo'lishini qo'llab-quvvatlash bo'yicha konkret chora-tadbirlarni amalgam oshirish va eksportni rag'batlantirish uchun qo'shimcha omillar yaratish;

Uhinchdan- qat'iy tejamkorlik tizimini joriy etish, ishlab chiqarish xarajatlari va mahsulot tannarxini kamaytirishni rag'batlantirish hisobidan korxonalarning raqobatdoshligini oshirish;

To'rtinchidan- elektroenergetika tizimini modernizatsiya qilish, energiya iste'molini kamaytirish va energiya tejashning samarali tizimini joriy etish choralarini amalgam oshirish;

Beshinchidan- jahon bozorida talab pasayib borayotgan bir sharoitda, ichki bozorda talabni rag'batlantirish orqali mahalliy ishlab chiqaruvchilarni qo'llab-quvvatlash iqtisodiy o'sishning yuqori sur'atlarini saqlab qolishda g'oyat muhim ahamiyatga ega" [6].

Ob'ektga mo'jallangan tushincha hozirgi vaqtda juda keng ishlatilishiga qaramasdan hozirgacha uning universal ta'rifi mavjud emas. Odatda, kelishishga qarab, u juda ko'p xususiyatlarni qamrab oladi, ularning har biri bir biriga bog'liq bo'lmay mavjud bo'lishi mumkin. Ammo, aniq bir xususiyatni qo'shimcha qilish va olib tashlash haligacha hal qilinmagan. Real dunyoda ob'ektga mo'jallangan tizimlarning kuchi g'oyalarning unikal kombinatsiyalariga asoslanadi, ular bir birini to'ldiradi[7-10].

Tizimlar qaralayotganda ob'ektga mo'jallangan termini to'rtta bazaviy xususiyatlar asosida aniqlanadi:

- ma'lumotlar va protseduralar ob'ektlar deb ataluvchini aniqlaydi;
- habar bu ob'ektlarning bog'lanishini aniqlash uchun ishlatiladi;
- o'xshash ob'ektlar klasslarga guruhlanishadi;

- ma’lumotlar va protseduralar klasslar ierarxiyasi tomonidan meroslanishlari mumkin.

Yuqorida keltirilgan to’rtta bazaviy xususiyatlarga ega bo’lgan istalgan tizim(til, instrumental vositalar yoki metodologiya) ob’ektga mo’jallangan deb ataladi. Agar faqat birinchi ikkita xususiyatga ega bo’lsa, tizim faqat ob’ektli deb ataladi. Ob’ektli tizimlar operatsion tizimlarni loyixalashda muhim ro’l o’ynaydi.

Ob’ektga mo’jallangan g’oya shuningdek ma’lumotlar bazasini loyixalashda va DV ni yaratish metodologiyasini yaratishda ham ishlatiladi. Ananaviy protsedurali dasturlash tillarida dastur matni spetsifik protseduralar, qism dasturlar yoki funktsiyalarga mos keluvchi modullarda yoziladi va saqlanadi. Bu modullar bloklarni tashkil qiladi va ulardan to’liq dastur shakkillantiriladi. Dastur tarkibi qism programmalarni, protseduralarni, yoki funktsiyalarni chaqiruvchi operatorlar ketma ketligidan iborat bo’ladi[11,12].

Real dunyo ob’ektlardan tashkil topadi, ob’ektlar o’z ustida amal qilish mumkin bo’lgan instruktsiyalar haqida kombinatsiyalashgan axborotlar bilan xarakterlanadi.

Real dunyo va uni modellashtiruvchi dastur orasida kontseptual portlash vujudga keladi. Ob’ektlar axborotlarni yashirishning tabiiy mexanizmiga ega. Ob’ektlardan foydalanuvchi, undagi usullarning nomlari bo’lishlari mumkin. Ob’ektning ma’lumotlari va funktsiyalar matni faqat muallifga ma’lum bo’ladi. Ma’lumotlarni bunday ximoyalash inkapsulyatsiyalash deyiladi. Ob’ektni markazida ma’lumotlarni saqlovchi kapsula deb atash mumkin. Ma’lumotlarga dostup usullar orqali amalga oshiriladi. Kapsulaning sirti nomlar va usllarni aniqlashning umumiy ma’lumotlarini saqlaydi.

Real dunyo ob’ektlari a’lohida mavjud bo’lmaydi – ular bir birlari bilan umumiy xususiyatlari bilan bog’lanishda bo’lishadi va ular bir biri bilan umumiy bo’lmagan xususiyatlari bilan ajralishadi.

Ob’ekt, dasturni yaratishning asosiy bloki sifatida kiritiladi, shu sababli o’xshash ob’ektlar klasslarga birlashtiriladi. Klass – bu ob’ekt bo’lib, uning azolariga qo’llash mumkin bo’lgan hamma ma’lumotlarni va usullarni saqlaydi.

Ma’lumotlarning bir qismi klassning o’ziga ham tegishli bo’ladi: bu klasslarning o’zgaruvchilari va usullari. Ma’lumotlarning bir qismi ba’zi bir klass ekzempilyarlariga tegishli ob’ektlarga tegishli bo’ladi: ular yaratilishida maxsus ma’lumotlar bilan initsalizatsiya qilinadi.

Predmetli –orientirli yondoshuwning bazaviy til sintaksisini tanlash amaliy masalalarni yechishda muhim ahamiyatga ega. Odatda baza sifatida mavjud dasturlash tillari sintaksisi ishlatiladi. Bu albatta ob’ektga mo’jallangan tizimlar yaratishni ancha osonlashtiradi. Ob’ektga mo’ljllangan tildagi dasturni tushinchalar daraxti va uning parametrlari shaklida tasavvur qilish mumkin, bazaviy sintaksis esa matnda shu daraxt tugunlarini ajratishda ishlatiladi[13,14].

Predmet sohasi mohiyatini realizatsiya qilish uchun, leksik, sintaksistik taxlil qilish va so’ngra kodni generatsiya qilish lozim bo’ladi. Dasturiy kodni generatsiya qilish qoidasi mavjud dasturlash tillari kompilyatsiya qoidalari asosida amalga oshirilishi mumkin. Kodni generatsiya qilish jarayonida ishlash tezligini optimizatsiya qilishning har xil usullarini o’zgartirish mumkin

Ushbu dissertatsiya ishida biz predmetli orientirli yondoshuvning afzalligini, predmetli orientirli tillardan foydalanish modelida namoyish etdik. Misol sifatida dinamik geometriyaning figuralarini qurish masalalari olindi. Bu masala ishda keltirilgan ob’ektga mo’ljallangan texnoogiyani bosqichma bosqich realizatsiya qilish metodologiyasida namoyish etildi. Bu misolda predmetli orientirli yondoshuvning juda effektiv ekanligi isbotlandi.

Mavzuning dolzarbligi. Bugungi kunda ob’ektga mo’ljallangan dasturlash texnologiyasi asosiy daturlash texnologiyasiga aylanib bormoqda, bu texnologiyalar bilan yechiladigan masalalar soni yanada oshib bormoqda. Dunyoda hozirning o’zidayoq bunday masalalarni yechish dasturlari ko’plab yaratilgan. Bu masalalar intellektualligi munosobati bilan va bu tizimlarning o’rgatilishligi imkoniyati mavjud bo’lganligi uchun bu tizimlarni intellektual deb atay boshlashdi. Bu masalalarni yechish dasturlari esa intellectual dasturlar deb yuritilaboshlandi. Bu dasturlarga ekspert tizimlar, qaror qabul qilishni sonli

asoslash tizimlari(decision support systems), hamda timsollarni bilib olish tizimlari kiradi. Dissertatsiya ishini aynan shunday muammolarga bog'ishlangan.

Mazkur ishimizning maqsadi. Dissertatsiya ishining asosiy maqsadi-

Predmetga mo'ljallangan tizim yaratish va bu tizimni yaratish metodologiyasi asosida Predmetga mo'ljallangan dastur yaraishdan iborat.Yaratilgan dastur asosida qurilish konstruksiyalari, jumladan ferma masalalarini konstrksiyalash va qurish masalalarini yechish va natijalarni taxlil qilish. Predmetga mo'ljallangan tizimli dasturning afzalligi va bu tizimning yangi imkoniyatlari taxlil qilish.

Ilmiyligi. Dasturiy mahsulotni yaratishning har xil bosqichlarida ishlatiluvchi zamonaviy instrumentlari, tizim loyixasi va uning realizatsiyasi orasidagi uzilishni qisqartirish uchun hizmat qiladi. Tizim loyixasi – bu tizim qanday ishlashini aniqlovchi, foydalanuvchining har xil xodisalariga reaktsiya qanday bo'lishi lozimligini aniqlavchidir. Dissertatsiya ishida sodda bazaviy klasslar yaratildi va ular yordamida konstruksiyalarni chizaoladigan klasslar yaratildi, ferma konstruksiyalarini chizish uchun predmetga yo'naltirilgan dasturiy tizimi yaratildi. Dasturiy tizim to'rtburchakli, uchburchakli, ko'pburchakli va ularning kombitatsiyasidan tashkil topgan ferma konstruksiyalarini chizaoladi.

Dasturiy mahsulotlarni yaratishning klassik usullaridan biri tizimga quyiladigan talablarni shakillantiruvchi, loyixaning barqaror holati o'rnatilgach tuziladigan hujjatni yaratishdir. Bu erda hosil bo'ladigan muammolardan biri- bu tizimga qo'yilgan talablar o'zgarmasdan qolishi mumkin emas, ular rivojlanib, borgan sari aniqlanib boradi. Talablarning o'zgarishi loyixaning mos qismlarining o'zgarishiga olib keladi, ular esa dastlabki kodning jiddiy o'zgarishiga olib keladi.Bu yondoshuv predmetga mo'ljallangan dasturlash texnologiyasining asosini tashkil etadi. Bunday masalalarni universal dasturlash tillari orqali realizatsiya qilish mumkin emas. UML va XML universal modellashtirish tillari ham yaramay qoladi. XML-konstruktsiyalar juda ko'payib ketadi. Natijada murakkab dasturiy tizimlarni realizatstya qilish uchun yangi tillar yaratish va shu sahoga tegishli bo'lgan biznes talablarni yozish talab qilinadi.

Bundan tashqari har bir loyixalanuvchi tizim uchun maxsus tillar terilmasi tanlash lozim bo'ladi.

Intellektual tizimlarni yaratishdagi muammolardan biri bilimlarni tasvirlash modelini tanlash hisoblanadi[3, 5, 8, 14]. Aynan bilimlarni tasvirlash modeli IT ning arxitekturasini, tizimning imkoniyatlari va xususiyatlarini aniqlaydi. Aynan shu parameter tizimning qanchalik intellektualligini aniqlaydi.

Bilimlarni tasvirlashning mohiyati uni formallashtirishdadir, ya'ni ularni simvolli tasvirlashga o'tkazishdir. Hozirgi vaqtda bir nechta bilimlarni tasvirlash bazaviy modellari mavjud. Masalan, dalillar va qoidalar asosida, neyron tarmoqlari, cemantik tarmoqlar va boshqalar yordamida modellashtirish mumkin. Ularning istalganidan foydalanib zamonaviy information tizimlarni yaratish mumkin. Bunday tizimlar endi ancha samarador va tushunarli hamda takomillashtirishga oson tizimlar bo'ladi.

Tarmoq intellektual tizimlari katta dasturiy mahsulotlarga kirgani uchun ularga katta material va vaqt sarflanadi, shu sababli ular asoasan korxonalar byurtmalari shaklida realizatsiya qilinadi. Bunday ekspert tizimlari birinchidan ma'lum bir soha predmeti masalalarini echishni baholash uchun ishlatiladi; ikkinchidan bunday ekspert tizimlari juda qimmat turishadi ; uchunchidan , bunday ekspert tizimlari apparat ta'minotiga yuqori talablarni qo'yadi; va nahoyat to'rtinchidan , ular tizimda ishlovchilarga cheklanishlar qo'yadi.

Bu kamchiliklar hozirgi fan va texnoogiyalarning gurkirab o'sishi munosabati bilan sekin asta bartaraf qilinmoqda.

Magistirlik dissertatsiyasi beshta bobdan**,** xulosa, foydalanilgan adabiyotlar va ilovalardan tashkil topgan.

Birinchi bobda ob'ektga mo'ljallangan dasturlashning asosiy tushunchalari, ob'ektga mo'ljallangan tizimlar, OMT ning muammolari va ularni yechish yo'llari ularning tarkibi taxlil qilindi.

Ikkinchi bobda ob'ektga mo'jallangan tizimlar bozori, shu jumladan maxsulotlar bozori va dasturlar bozori ham taxlil qilindi.

Uchunchi bobda zamonaviy intellektual tizimlar tahlil qilindi, ko’proq ma’lumotlarni va bilimlarni tasvirlashga axamiyat berildi.
Bu bobda bobda ob’ektga mo’jallangan dasturlash texnologiyasining asosiy prinsiplari, mexanizmlari taxlil qilindi.

To’rtinchi bobda predmetli –orientirli tillarning modeli va realizatsiya qilish usullari o’rganildi va taxlil qilindi. Predmetlii –orientirli yondoshuvning qo’llanilish metodikasi ishlab chiqildi. Geometrik qurishlar maslasini yechishni avtomatik tekshirish tizimi algoritmi yaratildi Tabiiy tilda qo’yilgan masaladan predmetli orientirli elementlarni ajratish jarayoni tavsiflandi, geometrik qurishlar maslasini yechishning Predmetga mo’ljallangan dasturi yaratildi. Geometrik qurishlar maslasini yechishning Predmetga mo’ljallangan dasturidan foydalanib masalalarni yechish natijalari olindi.

Beshinchi bobda qurilish konstruksiyalarini yaratish uchun bazaviy klasslar yaratish va qator qurilish konstruksiyalari, jumladan ferma nmasalalari yechildi va konstrksiyalandi, ferma chizmalari yaratildi. Ferma konstruksiyalarini yaratish predmetli orientirli tizimli dasturi yaratildi.

I-BOB. OB'EKTGA MO'LJALLANGAN AXBOROT TIZIMLARINING HOZIRGI HOLATINI TAHLIL QILISH

1.1. Asosiy tushunchalar

Ba'zi dasturlar ishlab chiqish va tarqatish bilan shug'ullanuvchi firmalar uchun "ob'ektga mo'jallangan" – termini juda muhim bo'lib u maxsulotlarni reklama qilishda ishlatiladi. Ba'zi dasturiy maxsulot ishlab chiquvchilar uchun esa bu termin ishlab chiqarish masalasini yechishdagi samaradorlikni oshirish deb tushiniladi[15,16,17].

Ob'ektga mo'ljallangan texnologiyaning aosini dasturiy vositani tahlash va yaratish usuli tashkil etadi. Bu texnologiya kompyuter texnlogiyalarida markaziy ro'l o'ynaydi. Shuni qayd qilish lozimki ob'ektlar ekspert tizimlarida, mikroprosessorlarda va ko'plab boshqa saholarda ishlatiladi.

Bu texnologiya nafaqat professional dasturchilarni va ilmiy tadqiqotchilarni sirli qiziqtiradi, balki butun dunyoda bu texnologiyaga qiziqish juda katta. Personal kompyuterlar dunyosida «tashabbuskorlik» juda ko'p narsani bildiradi. Bill Gates bu texnologiyani juda yuqori baholagan edi. Uning texnologiyani baholash uchun so'zlarni ishlovchi protsessorlarni misol sifatida keltirdi, bu holda foydalanuvchiga xujjatlarga "ob'ektlar" turini tovush shaklida, rasm yoki jadval shaklida kiritishni ta'minlaydi. Tovush tanlanishida dastur mos qurilmalar orqali ob'ektni eshitishi mumkin.

Ob'ektga mo'jallangan tizimlar bilan Microsoft firmasi shug'ullandi va firma ichida undan foydalandi. Noxoyat 1989 yili fevral oyida firma bu loyixalarda o'zining qatnashishini e'lon qildi: bu loyixa Glockenspiel (Dublin) kompaniyasi bilan C++ versiyasini ishlab chiqishga shartnoma tuzildi.

Boshqa kompaniyalar ham bu texnologiyaga o'z hissalarini qo'shishdi. Steve Jobs 1988 yili NeXT ni ob'ektga mo'jallangan interfeys va instrumentlar bilan ta'minladi. Lotus Mitch Kapor firmasi esa ob'ektga mo'jallangan yondoshuvni o'zining ON Technology kompaniyasida ishlatdi.

1.2. Ob'ektga mo'jallangan tizimlar

Ob’ektga mo’jallangan texnologiya – bu DV yaratish usuli bo’lib, u real dunyo ob’ektlarini ifodalashning usullarini ananaviy ifodalashdagiga qaraganda jiddiy farq bilan amalga oshiradi. DT ni yaratishda ob’ekt tushinchasi oddiy ob’ektlar va turlari hamda ularning o’zaro bog’lanishini modellashtirishda ishlatadi. Umuman olganda ob’ekt bu nomlangan istalgan narsadir. Har bir ob’ekt uni tavsiflovchi xususiyatlarga ega: bu uning holatini aniqlaydi. Ob’ekt unga biron bir harakat qilinganda o’zini biron bir holda tutadi: ya’ni u xulq atworga ega. Dasturiy ob’ekt bu real dunyo ob’ektining biron bir nusxasidir. U ham holati va xulq atworiga ega, bular ma’lumotlar va protseduralar tarzida ifodalanadi[18,19].

Kundalik ob’ektlar bir birlari bilan signallar jo’natishi va qabul qilishligi bilan munosobatda bo’lishadi. Dasturiy ob’ektlar esa bir biriga xabar jo’natish orqali bog’lanib turishadi. Xabar ob’ektlar tomonidan olinganda ular ob’ekt protseduralari bilan solishtiriladi, shundan keyin biron bir hatti harakat qilinadi. Harakat boshqa ob’ektga xabar jo’natish orqali ham bildirilishi mumkin.

Inson kundalik ob’ektlarni bir hil xususiyatlar asosida klasslarga birlashtiradi. Masalan, agar biz ob’ektni konvert desak, u holda bu ob’ekt to’g’risida biz juda ko’p narsani bilib olishimiz mumkin. Bir xil xususiyatlarga ega bo’lgan dasturiy ob’ektlar ham klasslarga birlashtiriladi.

Va nixoyat kundalik ob’ektlarning klassining o’zlari ham ierarxiyani tashkil etishi mumkin. Ular ierarxiya bo’yicha ancha katta klasslarning umumiy xususiyatlarini meroslaydi, ularni super klasslar deb ham atashadi. Konvertlar – qism klass konvertlarini saqlovchi superklassdir. Dasturiy klasslar ham ma’lumotlarni va protseduralarni meroslashi mumkin.

Ob’ektga mo’jallangan texnologiya foydalanuvchilarni DV ning samarali ishi bilan qiziqtiradi. Ob’ektga mo’ljallangan texnologiya quyidagi xususitlari bilan ajralib turadi:

- ob’ektlardan asosiy modellar sifatida foydalanish foydalanuvchiga real dunyoning murakkab tizimini modellashtirishni ta’minlaydi.
- ob’ektga mo’jallangan matnlarning ishlatilishi foydalanuvchi o’zgartirishi talablariga tezroq reaktsiya qilinishini ta’minlaydi;

- standart komponentalardan qayta foydalanish kodlarning ixcham yozilishini ta’minlaydi va DV yaratish vaqtini qisqartiradi;

Shu oshkora yangi imkoniyatlardan tashqari, ob’ektga mo’jallangan tillar va dasturlash muxitlari DV ni qadamba qadam yaratishni ta’minlaydi. Ob’ektga mo’jallangan tizimlar murakkablik bilan kurashishni ta’minlaydi.

Ob’ektga mo’jallangan tizimlarning birinchi afzalligi uning real dunyo bilan tabiiy bog’lanishidadir. DV yaratuvchi fizik sistemani dasturga o’tkazishni loyixalashni, avvalo hamma muhim fizk ob’ektlarni va uning mos xususiyatlarini aniqlab amalga oshirishi mumkin. O’zaro bog’langan fizik ob’ektlarning guruhi klasslarda ifodalanadi.

Ob’ektga mo’jallangan o’zgartirishni amalga oshirish uchun mo’jallangan.

Ob’ektga mo’jallangan texnologiyaning ikkinchi afzalligi – bu ob’ektlarning habar orqali bog’lanishidir. Yuqorida keltirilgan misolda, «o’lchashni boshlang» turidagi umumiy habarni tizimning hamma datchiklariga yuborish mumkin, ularning har biri ma’lum bir spetsifik tarzda javob qaytaradi. Agar biron bir fizik datchik eskirsa, u darxol almashtirilishi lozim. Bu holda bir vaqtda tizimning mos klassi ham almashtirilishi lozim, u yangi datchikni xarakterlovchi protseduralarni qamrab oladi. Yangi klass esa super klassdan o’ziga kerakli protseduralarni meroslaydi.

Ob’ektga mo’uljallangan tizimlarning juda moslashuvchanligi juda tez o’zgaruvchan muxitlarda foydalanuvchiga betakror katta imkoniyatlar yaratadi, masalan, dasturlash texnologiyasida. Masalan, Computer Science Corporation kompaniyasi Design Generator maxsulotini yaratishda ob’ektga mo’jallangan til Smalltalkdan foydalandi. Bu kompaniyaning takidlashicha ob’ektga mo’jallangan texnologiyadan foydalanishdan kelgan imkoniyat natijasida tezda bozorning yangi oqimlariga juda tez reaktsiya qilish imkoniyatiga ega bo’lishgan.

1.3. Ob’ektlarning takroriy ishlatilishi

Ob’ektga mo’jallangan tizimlarning uchunchi afzalligi shundan iboratki, endi klasslar boshqa klasslarning protseduralarini ham meroslashlari mumkin. Kompaniya eng ko’p ishltiluvchi va o’zlarida kelgusida yangi amaliy masalalarni yechish protseduralarini saqlovchi klasslar kutubxonasini yaratishi mumkin. Masalan, dastur yaratish bilan shug’ullanuvchi kompaniya, grafik primitivlar uchun klasslar yaratishi, masalan, silindir, konus yoki sfera uchun. Ular qism klasslar kononik kesmalar yoki kesimlar uchun asos bo’lib hizmat qiladi. Dastlabki matnlardan qayta foydalanish dastur yaratish vaqtini sezilarli kamaytiradi va loyixalovchilarga har xil soxa masalalarni ishonchli yechaolish imkoniyatini yaratadi.

Ilgari qism dasturlar kutubxonasidan dasturiy ta’minot yaratuvchilar standart masalalarni yechishda masalan, matematik hisoblashlarda foydalanishgan. Ob’ektga mo’jallangan tizimlar dastur matnlaridan takroriy foydalanish imkoniyatini yaratadi. Birinchi foydalanuvchilardan biri, Cadre Technologies, ob’ektga mo’jallangan tizimlar ishlatilganda dastur matni 5:1 ga qisqarishini aniqladi.

Ob’ektlar kutbxonasini ham mustaqil etkazib beruvchilardan olish ham mumkin. Hozirgi vaqtda bunday klasslar kutubxonalari asosan dasturlarning foydalanuvchi interfeyslarini yaratishda ko’p ishlatiladi. Chunki bunday interfeyslarni noldan boshlash – bu ancha murakkab masaladir. Quyidagi Apple va Whitewater Group turidagi kompaniyalar bunday interfeyslarni yaratishning instrumentariyalarini yaratishadi va foydalanuwchilarga yetkazib beradi. Bu instrumentariyalar bir nechta foydalanuvchi klasslaridan foydalanib, masalan, Window, Menu, ScrollBar va Icon turidagi bazoviy klasslardan foydalanib tezda interfeyslarni yaratish imkoniyatini beradi. Foydalanuwchilar bu klasslardan va ularning qism klasslaridan interfeys yaratishda foydalanishlari mumkin, masalan interfeysda piktogrammalar qushimcha qilishda ishlatishadi.

1.4. OMT larning oson qo’llanilishi

Ob’ektga mo’jallangan tizimlarning to’rtinchi afzalligi shundan iboratki-ob’ektga mo’jallangan tizimda oson ob’ektli-orientirli dasturlar modullari komplektatsiya qilinadi. Ananaviy DT ma’lumotlar va ularga dostup beruvchi va bu ia’lumotlarni o’zgartirishni amalga oshiruvchi protseduralardan iborat bo’ladi. Ma’lumotlar va protseduralar a’lohida-a’lohida komplektatsiya qilinadi, shu sababli berilmalar tarkibini o’zgartirish har xil kishilar tomonidan yozilgan har xil modullarga ta’sir qiladi. Ob’ektga mo’jallangan tizimlarda ma’lumotlar va protseduralar yaxlit bitta paket qismlari sifatida qaraladi. Berilmalarni o’zgartirishda hama ishlatilayotgan protseduralar osongina identifikatsiya qilinadi va o’zgartiriladi. O’zgartirilish tizimning bir qismida amalga oshirilishi munosobati bilan, uning ta’siri butun tizimga unchalik katta bo’lmaydi.

Biz bilamizki, dasturni yetkazib berish harajatlari dasturiy maxsulot tizimli dastur hayotiy siklining 80% gacha qiymatini tashkil etadi. Katta murakkab tizimlarni yaratuvchilar bu tizimlarni tez tez takomillashtirish muammolariga duch kelishadi, bunday hollarda esa OMT (ob’ektga mo’jallangan tizimlar) ana shu xarajatlarni kamaytirishga olib keladi. Masalan, Wild Leitz (Toronto, Kanada) kompaniyasi geografik axborot tezimini yaratishda ob’ektga mo’jallangan til Objective-C dan foydalandi.

1.5. OMT ning muammolari va ularni yechish yo’llari

Takomillashmaganligi

Zamonaviy OMT dagi cheklanishlar ularning takomillashmaganligidan kelib chiqmoqda. Bu cheklanishlarni bartaraf qilish esa DT sotuvchilar orasida yangi konkurensiya tashkil qilishga olib keldi. Bu bo’limda biz asosan ob’ektga mo’jallangan tizimlarni takomillashtirish muammolari ustida va bu tizimlarni takomillashtirish usullari ustida mulohozalar yuritamiz.

Hozirgi vaqtda ob’ektga mo’jallangan tizimlarining rivojlanishiga to’sqinlik qilayotgan narsa- bu texnik va boshqaruv personallar qarshiligi hisoblanadi. Ob’ektga mo’jallangan tizimlarining takomillashmaganligi hisoblanadi. Bu

takomillashmovlik hamma yangi texnologiyalarga hos xususiyatlardan iborat bo'ladi:

- qator standart platformalarga qator dostuplarning qo'yilishi;
- mavjud tizimlar va ma'lumotlar bazalari bilan integratsiyalanish lozimligi;
- keng qamrovli tizimlarni dasturlashdagi DT yetishmasligi.

Ob'ektga mo'jallangan tizimlarining rivojlanish texnologiyalarining muvaffaqiyatli rivojlanishi, uning asosan kompyuter texnologyalari industriyasiga tayangani hisoblanadi. Bu muvaffaqiyatli rivojlanish albatta yuqoridagi muammolarning hal bo'lishiga bog'liq bo'ladi. Bu esa hozircha juda qimmatli va ancha ko'p vaqt talab qiladi.

Dostupliligi

Foydalanuvchilarning dasturning olib yuruchanligi, tizim ochiqligiga, standartligiga qo'yilgan joriy talablari allaqachon ob'ektga mo'jallangan tizimlar yaratuvchilar tomonidan e'tiborga olingan. Asosiy ob'ektga mo'jallangan tillar foydalanuvchilarga dostupli bo'lishi uchun ma'lum bir standarlarga kelishi kerak, masalan, C++ va Smalltalk-80 muhitlari shu talablarga javob berishi mumkin. Bundan tashqari dasturiy mahsulot hamma apparat platformali har xil turdagi interfeyslar uchun ishlayolishi lozim, shu jumladan MS Windows, Presentation Manager, Open Look va OSF Motif da ham.

Integratsiyalashuvi

Hamma yangi tizimlar qatori ob'ektga mo'jallangan tizimlar mavjud dasturlash muhitlariga va realiatsion ma'lumotlar bazalariga mos kelishi kerak. Ular boshqa algoritmik tillarida yozilgan dasturlarga dostupga ega bo'lishi lozim. Chunki juda ko'p amaliy masalalarni yechishga Si, Paskal, Fortran va Kobol tillarida dasturlar yaratilgan va ulardan foydalanish lozim. Ob'ektga mo'jallangan tillarda yechilayotgan yangi masalalar mavjud shu vaqtgacha yozilgan dasturlar bilan moslashishi lozim. Ammo umuman olganda, integratsiyalashuv jarayoni- bu vaqt va resurslar jarayoni bo'lib u xech qachon to'liq hal qilinmaydi.

***Keng q**amrowli DT yaratish*

Smalltalk kabi ob'ektga mo'jallangan muhitlar ba'zi dasturchilar tomonidan muvaffaqiyatli ishlatilmoqda, ammo bari bir ob'ektga mo'jallangan tizimlar DT soxasidagi keng qamrovli loyixalarni yaxshi qo'llay olishmaydi. Masalan, mavjud (CASE) dasurlash texnologiyalari vositalari ob'ektga mo'jallangan tushinchalarni qo'llay olmaydi. Shu sababli CASE texnologiyalarini sotuvchilar bu kamchilikni bartaraf qilish imkoniyatiga egadirlar.

Servis roli

Hozirgi vaqtda ob'ektga mo'jallangan tizimlarni amaliyotga tadbiq qilishdagi asosiy to'sqinlik – bu konservativlik hisoblanadi. Bu kamchilikni yengib ob'ektga mo'jallangan tizimlarni keng tarqalishini ta'minlashimiz mumkin bo'ladi. Ob'ektga mo'jallangan tushincha kelajak kompyuter indisturiyasining fundamental tushinchasiga aylanadi, ammo hozircha ko'pchilik uchun bu tushincha hali ham yangilik hisoblanadi. Potensial haridorlarga ob'ektga mo'jallangan tizimlarning naqadar qimmatliligini tushintirish uchun, ko'plab ob'ektga mo'jallangan tizimlar yordamida olingan natijalar bilan tanishtirishga to'gri keladi. Bundan tashqari ularni ob'ektga mo'jallangan tizimlarda ishlash uchun o'rgatish ham kerak bo'ladi. Bundan tashqari faydalanuvchilarni ob'ektga mo'jallangan dasturlashga ham o'rgatish kerak bo'ladi. Ko'pchilik Si++ va Fortran muhitlarida dastur yozuvchilar dasturlashdagi yangi yondoshuvni qiyinchilik bilan bartaraf qilishmoqda. Aslida esa ob'ektga mo'jallangan model intuiktiv ravishda tanish modellashtiriluvchi masalalar yaratuvchilar va muxandislar uchun tushinarli bo'lsada, protsedura terminlarida fikr yurituwchi dasturchilar uchun ancha muammoga aylanib qolmoqda. Ba'zi ob'ektga mo'jallangan tizim yaratuvchilar va ularni sotuwchilar bu muammolarni bartaraf qilish uchun o'z hizmatlarini taklif qilishmoqda. Masalan, Arthur Andersen and Coopers & Lybrand kompaniyasi shunday ishlarni bajarishga kirishdi.

Ob'ektlarning doimiyligi

Yuqorida keltirilgan takomillashuvlilik yo'qligi muammosidan tashqari ob'ektga mo'jallangan texnologiyani ishlatish uchun yana ikkita texnik masalani hal qilishga to'gri keladi. Ob'ektga mo'jallangan tizimlarning birinchi va eng muhim

cheklanishlaridan biri – bu ma’lumotlarning ob’ektlarda saqlanishi bo’lib bu holda ularni foydalanuvchilar orasida bo’lib chiqish qiyinlashadi. Ob’ektga mo’jallangan tizimning alohida olingan foydalanuvchisiga bu muammo ta’sir qilmaydi. Ammo bu texnologiyadan kommertsiya sferasida foydalanishda yuqorida keltirilgan muammo absalyut cheklanishga aylanadi. Bu muammoni hal qilish birinchi navbatda ma’lumotlar bazalarini yaratuvchilarga katta yo’l ochib beradi.

Amaliy masalalarni yechish kommertsiya DT lari bitta umumiy ma’lumotlarga dostupga ega bo’lishi lozim. Bu g’oyani amalga oshirish uchun shunday usul yaratish lozimki, bu usuldan foydalanilganda ma’lumotlar va protseduralarni saqlovchi ob’ektlar dastur o’z ishini tugatganda ham mavjud bo’laversin. Bunday ob’ektlar "doimiy" ob’ektlar deb ataladi va ular ob’ektga mo’jallangan ma’lumotlar bazasida saqlanadi. Bunday ma’lumotlar bazasini rivojlantirishga hozirgi vaqtda katta e’tibor berilmoqda. Bu mahsulotlarning ko’pchiligi kommertsiya dostupiga egadirlar: masalan, Graphael, Servio Logic va Ontologic komponiyalari mahsulotlaridirlar, ammo ular ham takomillashtirilishi lozim bo’ladi.

Ob’ektga mo’jallangan ma’lumotlar bazalarini yaratishdan tashqari, amaliy dasturlar mavjud ma’lumotlar bazalariga dostupga ega bo’lishlari lozim. Ba’zi yaratuvchilar bu masalalr bilan shug’ullanishmoqda. Masalan, Intellicorp kompaniyasi KeeConnection mahsulotini yetkazib bermoqdaki ular instrumentlarning har xilini tanlash imkoniyatini yaratadi - AI, Kee, lar s Oracle yoki Ingres. ParcPlace Systems, postawщik Smalltalk-80 lar bilan erkin ishlay olishni ta’minlaydi.

Samaradorligi

Ob’ektga mo’jallangan tillarning realizatsiya qilinishi operatsion tizimlarga har xil yangi talablarni qo’yadi. Hozirgi vaqtda ishlab chiqarish samaradorligini oshirish katta nakladnoy harajat tizimlari yaratishni talab qiladi:

- ob’ektga mo’ljallangan aloqa yoki habar bog’lanishi va mos protseduralar dinamik tashkil qilinishi lozim ;

- agar keraksiz ob’ektlar uchun ajratilgan xotiralar avtomatik tarzda bo’shatilishi talab qilinsa ;
- agar tizim juda ko’p uncha katta bo’lmagan ob’ektlarga ega bo’lsa va ular asosiy xotiradan qo’shimcha xotiraga so’rovlar yordamida o’tkazilayotgan bo’lsa.

1990 yilning o’rtasiga kelib yuqorida keltirilgan muammolar yechilgach, ob’ektga mo’jallangan tizimlar kompyuterr texnikasining asosiy elementlaridan biriga aylanadi. Shunday bo’lganda birinchi rejaga samaradorlik muammosi chiqadi.

II. OB’EKTGA MO’JALLANGAN TIZIMLAR BOZORI

2.1. Bozor tasnifi

1988 yilga kelib ob’ektga mo’jallangan tizimlar daromadning 96% ikki gurux mahsulotlar hisobiga to’gri kelar edi. Ulardan birinchi guruhi – sun’iy intellekt tizimining gibrid instrumentlariga, ikkinchisi esa – tillar va dasturlash vositalariga. Qo’shimcha daromadlar ob’ektga mo’jallangan ma’lumotlar bazalari hisobiga olinar edi. Bu mahsulotlar uchta turdagi kompaniyalar tomonidan etkazilib berilar edi: sun’iy intellekt tizimlarini sotuvchilar tomonidan, apparat vositalarini ishlab chikaruvchilar tomonidan va yosh kompaniyalar tomonidan[20,21,23].

Bu mahsulotlar hir xil kompaniyalar tomonidan ilmiy va amaliy sohalarda ishlatilmoqda. Bu mahsulotlarni yetkazib beruvchilar, shu jumladan apparatura ishlab chiqaruvchilar va DT yaratuvchilar ham foydalanuvchilar hisoblanadi. Faol foydalanuvchilarga quyidagilar kiradi: telekommunikatsion kompaniyalari, harbiy va aerokosmik firmalari, ishlab chiqarish va texnik kompaniyalar. Hozirgi vaqtda ob’ektga mo’jallangan maxsulotlarning kommertsiyali ishlatilishi katta korporatsiyalarni boshqarishda ayniqsa qo’l kelmoqda.

Ob’ektga mo’jallangan maxsulotlarning qo’llanish soxalari hozirgi vatda quyidagilarni o’z ichiga oladi: foydalanuwchi interfeyslarini ishlab chiqish, dasturlash texnologiyasi, CAD/CAM, ekspert tizimlari va multimedia informatsion tizimlari.

1995 yilga kelib ob’ektga mo’jallangan texnologiya ko’plab yangi mahsulotlari bilan bozorga chiqib keldi. Mavjud mahsulotlar bilan birgalikda endi ular quyidagilardan tashkil topadi:

- redaktorlar, otladchiklar, ko’rib chiqish oynalari va boshqa dasturiy mahsulotlar. Ular ko’p oynali integrallashgan muhit qismlarini tashkil etadi;
- klasslar kutubxonasi va amaliy masalalar generatori. Hozir ulardan ko’pchiligi foydalanuvchi interfeyslarining asosini tashkil etadi;
- ob’ektga mo’jallangan tizimlarni tahlil qilish va yaratish uchun dasturlash texnologisi vositalari (CASE) :
- ob’ektga mo’jallangan ma’lumotlar bazalarini boshqarish tizimlari;

- foydalanuvchilar uchun amaliy instrumentlar.

Mahsulotlar diapazoninig kengayshi, xususan mawjud texnologiyalar bilan to'liq mos keluvchi ob'ektga mo'jallangan ma'lumotlar bazalarining paydo bo'lishi, kommertsiya amaliy masalalari bozorini ochdi.

2.2. Maxsulotlar

Ob'ektga mo'jallangan tillar

Ushbu tahlilda har xil foydalanuvchilar guruhlarini qiziqtiruvchi uchta asosiy tillar bazasi qaraladi:

C tilida dastur yaratuvchi dasturchilar juda tez C++ ga o'tib ketishmoqda. C++ da AT&T kompaniyasi tomonidan Si tili ob'ektga mo'jallangan kengaytirilishga ega bo'ldi. 90 yillarga kelib C++ tili yangi loyixa yaratuvchilarning asosiy dasturlash tiliga aylandi.

Sun'iy intellekt tizimlarini yaratuvchilar esa Lisp asosida yaratilgan dasturlash tillaridan foydalanishdi. Ular Common Lisp Object System (CLOS) dan foydalanishmoqda.

Smalltalk hamjamiyati. Smalltalk – bu ob'ektga mo'jallangan til va dasturlash muhitidir, u Xerox PARC firmasida yaratildi. Bu ob'ektga mo'jallangan tili va muhitlarning juda muvaffakiyatli varianti hisoblanadi.

Bundan tashqari kommertsiya sohasida boshqa ob'ektga mo'jallangan tillar ishlatiladi:

- Simula – skandinaviya firmalari tomonidan etkazilayotgan birinchi ob'ektga mo'jallangan til hisoblanadi;
- Objective-C- Stepstone Corporation firmasi tomonidan patentlangan til hisoblanadi, u C va Smalltalk ga asoslangan;
- Eiffel - Interactive Software Engineering firmasi tomonidan patentlangan til, Simula va Ada tillariga asoslangan;
- Object Pascal –Paskalning ob'ektga mo'jallangan kengaytmasiga ega bo'lgan til, u Apple firmasi tomonidan ishlatilmoqda;

- Actor – Whitewater Group ishchi guruhi tomonidan ishlab chiqilgan,Paskal asosida ob’ektga mo’jallangan kengaytmasiga ega

Bu tillar ob’ektga mo’jallanganlikning ma’lum bir qirralarini amalga oshiradi. Masalan, Ada ob’ekt va klasslar asosiy tushinchalarini qullaydi, ammo unda meroslash mexanizmi yo’q.

Hozrgi vaqtda ko’proq ochiq tizimlarga talab ko’paymoqda. Quyidagi uchta asosiy ob’ektga mo’jallangan tillarga ahamiyat berilmoqda: C++, Smalltalk va CLOS.

Gibrid instrumental vositalar

Bu xil vositalar ekspert tizimlarini qo’llash va rivojlantirishga mo’jallangan. Ular ob’ektga mo’jallangan texnologiyalar imkoniyatlaridan foydalanishadi, ammo faqat ko’plab vositalar to’plamining biri sifatida. Bozorda asosan Intellicorp firmasi tomonidan yaratilgan Kee mahsuloti liderlik qiladi.

Ko’pchilik foydalanuvchilar ob’ektga mo’jallangan dasturlash bilan shu nuqtai nazrda tanishdirlar. Ba’zi hollarda ob’ektga mo’jallangan tilini a’lohida olish mumkin: masalan, Xerox firmasi ob’ektga mo’jallangan tilni Loops muhitidan ajratib qo’ydi va Lisp tilining CommonLoops - ob’ektga mo’jallangan kengaytmasini yaratdi[24,25,26]

2.3. Bozorning yangi imkoniyatlari

Sinflar kutubxonasi va amaliy masalalar generatori

Ob’ektga mo’ljallangan texnologiya yangi bozorlarni tashkil qildi. Bu asosan ob’ektga mo’jallangan ikki asosiy xususiyatlari hisobiga: o’hshash ob’ektlarni guruhlashtirish va ma’lumotlarni meroslash va protseduralar hisobiga mumkin bo’ldi.

Dasturiy ob’ektlarning klasslari masalan, stek, massiw, oyna yoki piktogrammalar kutubxona ob’ektlari shaklida tashkil qilinadi. Shunday qilinganda ulardan har xil loyixalarda ishlatish imkoniyati yaratiladi. Klasslar kutubxonasini yetkazib

beruvchi yozadi, testlaydi va berilgan matn bilan ishaydi. Klasslar kutubxonasini sotib oluvchilari amaliy foydalanuvchi hisoblanadi. Klass bilan ishlayotgan foydalanuvchi, faqat uning aniqlanishini bilishi kerak bo'ladi. Foydalanuvchi habar aynan berilgan ob'ektga jo'natilayotganiga ishonchi bo'lishi kerak.

Masalan, dasturda Sequence (ketmaketlik) , Set yoki Array (massiw) turidagi elementlar terilmasi bilan ishlaydigan bazaviy klass kerak bo'lib qolishi mumkin. Klasslarni yaratuvchi bu protseduralar qanday realizatsiya qilinganini biladi. Foydalanuvchi esa protseduralardan qaysi biri shu klasslar bilan ishlatish mumkinligini bilishi kerak: masalan, massivga element qo'shimcha qilishi uchun yoki ketmaketlikda navbatdagi elementni topish uchun qaysi protseduralardan foydalanishni biladi. Bugungi kunda bunday klasslar kutubxonalari takroriy ishlatiluvchi dasturiy komponentalardan tashkil topadi.

Ammo, klasslar kutubxonasidan foydalanish uchun, foydalanuwchi ob'ektga mo'jallangan dasturlash tilini bilishi kerak, maslan, Object Pascal, C yoki C++. Amaliy masalalarni yechishda klasslar kutubxonasidan keng miqiyosda foydalanish esa yangi amaliy masalalar generatoridan foydalanishni taqozo qiladi.

Ob'ektga mo'jallangan ma'lumotlar bazasi

Ob'ektga mo'jallangan ma'lumotlar bazalarida ikkita harakat kuchlari mavjud: doimiy ob'ektlarni saqlash uchun xotira joyini ajratish va mavjud ma'lumotlar bazasi hajmini ob'ektga mo'jallangan ob'ektlar hisobiga kengaytirish va uni xotirada saqlash.

Ob'ektga mo'jallangan tizimlarning C++ kabi tillarda rivojlantirilishi CASE texnologiyasidan foydallanishni taqoza qiladi. Ob'ektga mo'jallangan loyixalashni quvvatlash uchun ma'lum bir metodologiyani ishlab chiqish lozim. Ob'ektga mo'jallangan tillar rivojlanishidagi muvaffaqiyatlar CASE bozorini ham rivojlanishiga olib keldi.

III. O’BEKTGA MO’LJALLANGAN TIZIMLARNING ASOSIY TUSHINCHALARI

3.1. Ta’riflar

Ob’ektga mo’jallangan tushincha hozirgi vaqtda juda keng ishlatilishiga qaramasdan hozirgacha uning universal ta’rifi mavjud emas. Odatda, kelishishga qarab, u juda ko’p xususiyatlarni qamrab oladi, ularning har biri bir biriga bog’liq bo’lmay mavjud bo’lishi mumkin. Ammo, aniq bir xususiyatni qo’shimcha qilish va olib tashlash haligacha hal qilinmagan. Real dunyoda ob’ektga mo’jallangan tizimlarning kuchi g’oyalarning unikal kombinatsiyalariga asoslanadi , ular bir birini to’ldiradi[27,28,29].

Tizimlar qaralayotganda ob’ektga mo’jallangan termini to’rtta bazaviy xususiyatlar asosida aniqlanadi:

- ma’lumotlar va protseduralar ob’ektlar deb ataluvchini aniqlaydi;
- habar bu ob’ektlarning bog’lanishini aniqlash uchun ishlatiladi;
- o’xshash ob’ektlar klasslarga guruhlanishadi;
- ma’lumotlar va protseduralar klasslar ierarxiyasi tomonidan meroslanishlari mumkin.

Yuqorida keltirilgan to’rtta bazaviy xususiyatlarga ega bo’lgan istalgan tizim(til, instrumental vositalar yoki metodologiya) ob’ektga mo’jallangan deb ataladi. Agar faqat birinchi ikkita xususiyatga ega bo’lsa, tizim faqat ob’ektli deb ataladi. Ob’ektli tizimlar operatsion tizimlarni loyixalashda muhim rol o’ynaydi.

ob’ektga mo’jallangan g’oya shuningdek ma’lumotlar bazasini loyixalashda va DV ni yaratish metodologiyasini yaratishda ham ishlatiladi.

3.2. Ob’ektlar va klasslar

Ob’ektga mo’jallangan tushincha hozirgi vaqtda juda keng ishlatilishiga qaramasdan hozirgacha uning universal ta’rifi mavjud emas. Odatda, kelishishga qarab, u juda ko’p xususiyatlarni qamrab oladi, ularning har biri bir biriga bog’liq bo’lmay mavjud bo’lishi mumkin. Ammo, aniq bir xususiyatni qo’shimcha qilish va olib tashlash haligacha hal qilinmagan. Real dunyoda ob’ektga mo’jallangan

tizimlarning kuchi g’oyalarning unikal kombinatsiyalariga asoslanadi , ular bir birini to’ldiradi.

Tizimlar qaralayotganda ob’ektga mo’jallangan termini to’rtta bazaviy xususiyatlar asosida aniqlanadi:

- ma’lumotlar va protseduralar ob’ektlar deb ataluvchini aniqlaydi;
- habar bu ob’ektlarning bog’lanishini aniqlash uchun ishlatiladi;
- o’xshash ob’ektlar klasslarga guruhlanishadi;
- ma’lumotlar va protseduralar klasslar ierarxiyasi tomonidan meroslanishlari mumkin.

Yuqorida keltirilgan to’rtta bazaviy xususiyatlarga ega bo’lgan istalgan tizim(til, instrumental vositalar yoki metodologiya) ob’ektga mo’jallangan deb ataladi. Agar faqat birinchi ikkita xususiyatga ega bo’lsa, tizim faqat ob’ektli deb ataladi. Ob’ektli tizimlar operatsion tizimlarni loyixalashda muhim rol o’ynaydi.

ob’ektga mo’jallangan g’oya shuningdek ma’lumotlar bazasini loyixalashda va DV ni yaratish metodologiyasini yaratishda ham ishlatiladi.

Ananaviy protsedurali dasturlash tillarida dastur matni spetsifik protseduralar, qism dasturlar yoki funktsiyalarga mos keluvchi modullarda yoziladi va saqlanadi. Bu modullar bloklarni tashkil qiladi va ulardan to’liq dastur shakkillantiriladi. Dastur tarkibi podprogrammalarni, protseduralarni, yoki funktsiyalarni chaqiruvchi operatorlar ketma ketligidan iborat bo’ladi.

Real dunyo ob’ektlardan tashkil topadi, ob’ektlar o’z ustida amal qilish mumkin bo’lgan instruktsiyalar haqida kombinatsiyalashgan axborotlar bilan xarakterlanadi.

Real dunyo va uni modellashtiruvchi dastur orasida kontseptual portlash vujudga keladi. Ob’ektlar axborot larni yashirishning tabiy mexanizmiga ega. Ob’ektlardan foydalanuvchi, undagi usullarning nomlari bo’lishlari mumkin. Ob’ektning ma’lumotlari va funktsiyalar matni faqat muallifga ma’lum bo’ladi. Ma’lumotlarni bunday himoyalash inkapsulyatsiyalash deyiladi. Ob’ektni markazida ma’lumotlarni saqlovchi kapsula deb atash mumkin. Ma’lumotlarga

dostup usullar orqali amalga oshiriladi. Kapsulaning sirti nomlar va usllarni aniqlashning umumiy ma’lumotlarini saqlaydi.

Real dunyo ob’ektlari a’lohida mavjud bo’lmaydi – ular bir birlari bilan umumiy xususiyatlari bilan bog’lanishda bo’lishadi va ular bir biri bilan umumiy bo’lmagan xususiyatlari bilan ajralishadi.

Ob’ekt dasturni yaratishning asosiy bloki sifatida kiritiladi, shu sababli o’xshash ob’ektlar klasslarga birlashtiriladi. Klass – bu ob’ekt bo’lib, uning azolariga qo’llash mumkin bo’lgan hamma ma’lumotlarni va usullarni saqlaydi. Ma’lumotlarning bir qismi klassning o’ziga ham tegishli bo’ladi: bu klasslarning o’zgaruvchilari va usullari. Ma’lumotlarning bir qismi ba’zi bir klass ekzempilyarlariga tegishli ob’ektlarga tegishli bo’ladi: ular yaratilishida maxsus ma’lumotlar bilan initsalizatsiya qilinadi.

Ananawiy protsedurali dasturlash tillarida dastur matni spetsifik protseduralar, qism dasturlar yoki funktsiyalarga mos keluwchi modullarda yoziladi va saqlanadi. Bu modullar bloklarni tashkil qiladi va ulardan to’liq dastur shakkillantiriladi. Dastur tarkibi podprogrammalarni, protseduralarni, yoki funktsiyalarni chaqiruvchi operatorlar ketma ketligidan iborat bo’ladi.

Real dunyo ob’ektlardan tashkil topadi, ob’ektlar o’z ustida amal qilish mumkin bo’lgan instruktsiyalar xaqida kombinatsiyalashgan axborot lar bilan xarakterlanadi. Real dunyo va uni modellashtiruwchi dastur orasida kontseptual portlash wujudga keladi. Ob’ektlar axborot larni yashirishning tabiy mexanizmiga ega. Ob’ektlardan foydalanuwchi, undagi usullarning nomlari larni bilishlari mumkin. Ob’ektning ma’lumotlari va funktsiyalar matni faqat muallifga ma’lum bo’ladi. Ma’lumotlarni bunday ximoyalash inkapsulyatsiyalash deyiladi. Ob’ektni markazida ma’lumotlarni saqlowchi kapsula deb atash mumkin. Ma’lumotlarga dostup usullar orkali amalga oshiriladi. Kapsulaning sirti nomlar va usllarni aniqlashning umumiy ma’lumotlarini saqlaydi.

Real dunyo ob’ektlari a’loxida mawjud bo’lmaydi – ular bir birlari bilan umumiy xususiyatlari bilan bog’lanishda bo’lishadi va ular bir biri bilan umumiy bo’lmagan xususiyatlari bilan ajralishadi.

Ob’ekt dasturni yaratishning asosiy bloki sifatida kiritiladi, shu sababli o’xshash ob’ektlar klasslarga birlashtiriladi. Klass – bu ob’ekt bo’lib, uning azolariga qo’llash mumkin bo’lgan xamma ma’lumotlarni va usullarni saqlaydi. Ma’lumotlarning bir qismi klassning o’ziga xam tegishli bo’ladi: bu klasslarning o’zgaruwchilari va usullari. Ma’lumotlarning bir qismi ba’zi bir klass ekzempilyarlariga tegishli ob’ektlarga tegishli bo’ladi: ular yaratilishida maxsus ma’lumotlar bilan initsalizatsiya qilinadi.

Masalan, yangi ma’lumot turi shape aniqlangan bo’lsin va u grafik redaktorlarda ishlatilayotgan bo’lsin. C++ da bu tur class operatori bilan quyidagicha aniqlanishi mumkin:

```
class shape { point centre; colour col; // ... kommentariy
public:
point where (){return centre; }
void move (point to)
{
   centre = to; draw ();
}
virtual void draw ();
virtual void rotate (int) ; // ...kommentariy
};
```

IV –BOB. PREDMETGA MO'LJALLANGAN TILLARNING MODELI VA REALIZATSIYA QILISH USULLARI

Dasturiy mahsulotni yaratishning har xil bosqichlarida ishlatiluvchi zamonaviy instrumentlari, tizim loyixasi va uning realizatsiyasi orasidagi uzilishni qisqartirish uchun hizmat qiladi. Tizim loyixasi – bu tizim qanday ishlashini aniqlovchi, foydalanuvchining har xil xodisalariga reaktsiya qanday bo'lishi lozimligini aniqlavchidir. Dasturiy mahsulotlarni yaratishning klassik usullaridan biri tizimga qo'yiladigan talablarni shakillantiruvchi, loyixaning barqaror holati o'rnatilgach tuziladigan hujjatni yaratishdir[29].

Bu erda hosil bo'ladigan muammolardan biri- bu tizimga qo'yilgan talablar o'zgarmasdan qolishi mumkin emas, ular rivojlanib, borgan sari aniqlanib boradi. Talablarning o'zgarishi loyixaning mos qismlarining o'zgarishiga olib keladi, ular esa dastlabki kodning jiddiy o'zgarishiga olib keladi. Bunday masalalarni universal dasturlash tillari orqali realizatsiya qilish mumkin emas. UML va XML universal modellashtirish tillari ham yaramay qoladi. XML-konstruktsiyalar juda ko'payib ketadi. Natijada murakkab dasturiy tizimlarni realizatstya qilish uchun yangi tillar yaratish va shu sahoga tegishli bo'lgan biznes talablarni yozish talab qilinadi. Bundan tashqari har bir loyixalanuvchi tizim uchun maxsus tillar terilmasi tanlash lozim bo'ladi.

4.1. Predmetga mo'ljallangan yondoshuvning qo'llanilish metodikasi

Bo'lg'usi tizimga talablarni yig'ish: byurtmachilar fikrini va talablarini bilish, soha predmeti ekspertlari fikrini bilish, kelgusi foydalanuvchilarni fikrini va talablarini bilish, jarayonlarni kuzatish va avtomatlashtirilishi lozim bo'lgan bayonnomalar yaratish. Predmet sohasi bo'yicha kerakli ma'lumotnoma yig'ish va tahlil qilish (masalalarni shakillantirishda ekspertlar tomonidan o'z o'zidan keraklisi ko'rinib turgan ma'lumotlar).

Loyixaning **glossariyasini** yaratish. Glossariyaga aslida ekspertlar va dasturchilar tomonidan har bir tushiniluvchi predmet sohasi tushinchalarining hammasi

kiritiladi. Glossariylar tizimga qisqa va aniq talablarni shakillantirish uchun hizmat qiladi.

Loyixalash

Predmetli –orientirli yondoshuwning qo'llanilishi modeli quyidagicha. Dasturiy ta'minot o'zining hayotiy siklining har bir bosqichida ob'ektiv formada biron bir tilda ifodalanadi. Shuningdek avvalo u predmet sohasida biron bir talablar shaklida tabiiy tilda ifodlanadi. Oxirida esa dasturiy ta'minot aniq bir dasturiy kodda ifodlanadi (bu formal til). Bu modelda loyixalash jarayoni – bu bir tildan ikkinchisiga ketma ket o'tkazish (har bir bosqichda ancha abstraktlikdan ancha konkretlikga o'tkazilish), shuningdek boshlang'ich stadiyalarda bu o'tkazilishlarni insonning o'zi bajaradi (tabiiy tildan yuqori pog'onali formal tilga o'tkazish), keyingi bosqichlarida esa ma'lum bir avtomatik o'tkazilishlar ishlatiladi (Predvetli –orientirli yondoshuv tili translyatori). Ob'ektga mo'jallangan tizimlar yaratishda agar tilning elementlari soni kam bo'lsa, u holda uni tavsiflavchi konstruktsiyalar juda noqulay kontruktsiyalar bilan tavsiflanadi. Ayniqsa formal tavsflanishlar juda murakkab ko'rinishga ega bo'ladi:

1) Avtomatik isbotlanuvchi teoremalar.

2). Masalaning qo'yilishidan tabiiy tilda predmetli –orientirli yondoshuvning ob'ektga mo'jallangan elementlarni ajratish ya'ni predmet sohasi tushunchalarini ajratish va ularning parametrlarini ajratish. Bunda elementlar bilan amallar va operatsiyalarning o'zlari ham predmeti sohasi tushunchasi bo'ladi. Tushuncha faqat bitta so'zga mos keladi.

3) Predmetli –orientirli yondoshuwning bazaviy til sintaksisini tanlash. Odatda baza sifatida mavjud dasturlash tillari sintaksisi ishlatiladi. Bu albatta ob'ektga mo'jallangan tizimlar yaratishni ancha osonlashtiradi. Ob'ektga mo'ljllangan tildagi dasturni tushinchalar daraxti va uning parametrlari shaklida tasavvur qilish mumkin, bazaviy sintaksis esa matnda shu daraxt tugunlarini ajratishda ishlatiladi. Bundan tashqari, bazaviy sintaksis, quyidagilarni aniqlaydi:

- Predmet sohasi tushunchalarini ajratishni (masalan, bitta aniq identifikator har xil tushunchalarni ifodalashi mumkinmi).
- Predmetli –orientirli dasturning grammatik to’g’ri yozilganligini tekshirish. Noma’lum identifikatorlarni aniqlash .
- Tabiiy tildagi masalaning qo’yilishini ob’ektga mo’jallangan tilga o’tkazish. Bazaviy sintaksis tanlangach, matnda tanlangan tilda ifodalanuvchi funktsiyalar, klasslar, ob’ektlar, o’zgaruvchilardan foydalanish mumkin. Predmetli –orientirli tilga masalaning qo’yilishini to’lig’icha o’tkazish talab qilinadi. Agarda masalaning qo’yilishining ma’lum bir qismi ob’ektga mo’jallangan tilga sintaksisning cheklanganligi hisobiga o’tkazilmay qolsa, u holda oldingi qadamga qaytiladi va bazaviy sintaksis o’zgartiriladi.

Predmet sohasi mohiyatini realizatsiya qilish uchun, leksik, sintaksistik taxlil qilish va so’ngra kodni generatsiya qilish lozim bo’ladi. Dasturiy kodni generatsiya qilish qoidasi mavjud dasturlash tillari kompilyatsiya qoidalari asosida amalga oshirilishi mumkin. Kodni generatsiya qilish jarayonida ishlash tezligini optimizatsiya qilishning har xil usullarini o’zgartirish mumkin.

Realizatsiya jarayonida xosil qilinayotgan dasturiy maxsulotni testlash imkoniyati tug’iladi. Bu xolda quyidagi muammo paydo bo’ladi: biz dasturni testalashimiz mumkin, agarda u to’g’ri kompilyatsiya qilinsa. Bunday yondashishni realizatsiya qilishda 3 ta qadam takrorlanadi:

- Modulli matnning dasturiy kodi yaratiladi, unda dasturiy kodni generatsiya qilishga kirish qismiga dastlabki matn kiritiladi.
- Hamma moduli matnlar bajarilishga qo’yiladi.
- Hamma matnlar terilmasi muvaffaqiyatli bajariladi. Agar shunday bo’lmasa, 3 qadamga qaytariladi.
- Endi talab qilingan funktsionallikga erishilgach tarkiblarning takomillashtirishga kirishiladi.

4.2. Geometrik qurishlar maslasini yechishni avtomatik tekshirish tizimini yaratish

Misol. Biz dinamik geometriya tushunchasini kiritamiz. Dinamik geometriya deb shunday dasturlash muhitiga ataymizki unda kompyuter yordamida shunday geometrik qurishlar yaratiladiki unda dastlabki ob'ekt harakat qilganda geometrik qurilishlar ham o'zgara boradi. Bu jarayon dinamik geometriya muhiti deb ataladi. Shunday tizimga asos qilib o'qituvchig qurilishi lozim bo'lgan geometrik masalalarni shakillantirishni va o'quvchiga –bu vfsalani yechisn va o'qituvchi tommonidan va avtomatik tarzda nazorat qilinishi imkoniyatini yaratiladi. Bu tizimni tavsiflash uchun predikatlar terilmasi qurish lozim bo'ladi.

Yaratilayotgan tizimga talablarni shakillantirish

Geometrik masalalarni tilda shakillantirish uchun geometrik primitivlar va ular orasidagi munosobatlar kerak bo'ladi. Analitik geometriya nuqtai nazaridan hamma geometrik ob'ektlar bitta yoki bir nechta tenglamalar bilan beriladi, Konkret ob'ekt uning turi bilan aniqlanadi (masalan, nuqta, to'g'ri chiziq, kesma, nur) va mos turlarga tegishli koeffitsientlar, mos ob'ektlar tenglamalari bilan aniqlanadi.

Masalani echishni awtomatik tekshirish uchun masala sharti ob'ektga mo'jallangan tilda va tabiy tilda (o'quwchilar ob'ektga mo'jallangan tilni bilmasliklari mumkin)yozilishi lozim.

Geometrik qurish masalalari:

- burchak bessektrissasini sirkul va lineyka yordamida qurish talab qilinadi;
- uchburchak berilgan, unga ichki chizilgan aylanani qurish talab qilinadi;
- uchta bir biriga urinuvchi aylanalar berilgan, 3 ta berilmaga urinuvchi ichki chizilgan aylanani chizish talab qilinadi.

Loyixa glossariyasini yaratish

Yasash uchun masalalar – masala quyidagi atributlarga ega:

o’quvchi uchun shrtlar – o’quvchiga ko’rsatilishi lozim bo’lgan shart matni. Yarim avtomatik tarzda masalaning formal sharti asosida generatsiya qilinadi; **berilgan** – boshlangich yasashli chizma, unda hamma ob’ektlar nomlanadi; **yasash** – masala sharti bo’yicha o’quvchi yasashi lozim bo’lgan ob’ektlar ro’yxati keltiriladi;

bu ob’ektlarning xususiyatlari:

foydalanish – o’quwchi yasashda foydalanadigan instrumentlar ro’yxati.

Tekislikdagi geometriyaga bog’liq bo’lgan ob’ektlar **(planimetriya):**

- **Nuq**ta –ikkita koordinata bilan beriladi x_P va y_P;
- **To’g’ri chizi**q – Ax+By+S=0 tenglamasi bilan beriladi, konkret to’g’ri chiziq esa quyidagi uchta koeffitsient bilan aniqlanadi (A,B,S).
- **Kesma –** ikkita uchining koordinatalari bilan beriladi.
- **Nur –** boshlanish nuqtasi koordinatalari va nurda yotuvchi bitta nuqta koordinatalari bilan beriladi.
- **Aylana**– $(X-Xc)^2 +(Y-Yc)^2= R^2$ tenglamasi bilan beriladi, konkret aylana esa quyidagi uchta koeffitsient bilan aniqlanadi (Xc, Yc,R). Koeffitsientlari bir xil bo’lgan aylanalar bir xil deyiladi.
- **Doira –** quyidagi tengsizlik bilan beriladi $(X-Xc)^2 +(Y-Yc)^2 \leq R^2$ aylana shaklida aniqlanadi va taqqoslanadi.
- **To’rtburchak** – 4 uchining koordinatalari bilan beriladi, uchlarini almashtirishgacha taqqoslanadi.
- **Uchburchak** –3 uchlarining koordinatalari bilan beriladi.
- **Ko’pburchak**– n ta uchlari koordinatalari bilan beriladi.

Predikatlar :

- «**Tenglik**» – ba’zi ob’ektlar uchun (masalan, nuqtalar uchun) koeffitsientlarni taqqoslash orqali tekshiriladi, boshqa ob’ektlar uchun esa

(masalan, to’g’ri chiziq uchun), taqqoslashning maxsus protsedurasi ishlatiladi.

- «**Perpendikulyarlik**» – to’g’ri chiziq yoki kesmalar uchun mos vektorlarining skalyar ko’paytmasining nulga tengligi bilan aniqlanuvchi munosobat.
- «**Parallellik**» – bu faqat to’g’ri chiziqlar uchun ishlatiladi. Ikkita
- $A_1x+B_1y+ S_1=0$ va $A_2x+ B_2y+ S_2=0$ to’g’ri chiziqlar uchun bu shart quyidagicha yoziladi $A_1B_2 - A_2 B_1=0$
- «**Kesishish**» – ikki geometrik ob’ektlar kamida bitta umumiy nuqtaga ega bo’lishadi.
- «**Urinma**» – ikki geometrik ob’ektlar faqat bitta umumiy nuqtaga ega bo’lishadi.
- «**Ichki chizilgan**» (aylana) – mumkin bo’lgan maksimal radiusga ega bo’lgan aylana, figura ichiga joylashtirilgan va urinuvchi aylanalar.
- «**Tashqi chizilgan**» (aylana) – bu mumkin bo’lgan minimal radiusli aylana, figura bu holda to’lig’icha aylana ichiga joylashadi
- «**To’gri burchakli**» (uchburchak) –bu uchburchakning bitta burchagi to’g’ri burchak .

Instrumentlari

- **Sirkul –** aylanalarni yasashda ishlatilishi mumkin, hamda, nuqtalar orasidagi masofalarni o’lchash uchun ishlatiladi.
- **Bo’linmalariga ega bo’lmagan lineyka –** berilgan ikki nuqtadan o’tuvchi to’g’ri chiziqlarni yasashda ishlatiladi,.
- **Bo’linmalariga ega bo’lgan lineyka –** masofalarni o’lchash va istalgan masofalarni o’lchab olish uchun ishlatiladi.
- **Burchak –** to’g’ri chizikga perpendikulyar yasashda ishlatilishi mumkin

- **Transportir –** berilgan burchak ostida chiquvchi to’g’ri chiziqlarni yasashda ishlatiladi.

4.3. Tabiiy tilda qo’yilgan masaladan predmetli orientirli elementlarni ajratish

Avvalo, predmetli orientirli tilda asosan glossariyada aniqlangan tushinchalar ishlatiladi, tilning asosiy tushnchasi –bu masala hisoblanadi. U o’zida boshqa narsalarni ham agregiratsiya qiladi.

O’quvchi uchun shart: Uchburchak berilgan, unga ichki chizilgan aylanani yasash talab qilinadi, bunda faqat sirkul va lineykadan foydalaniladi. Berilgan: A nuqtasi, B nuqtasi, S nuqktasi, AB kesma, AS kesma, BS kesma. Yasash : R aylanani kurish.

Bunda : A nuqtasi «yotadi» P va B «yotadia» P va C «yotadi» P da. Instrumentlar: sirkul bo’linmalarga ega bo’lmagan lineyka. Shuni aytish lozimki «Berilgan» ni «Uchburchak ABC» deb yozish ham mumkin edi.

Predmetli orientirli tilning bazaviy sintaksisni aniqlash

Predmetli orientirli tilga matn shunday kiritilishi mumkinki, bu holda leksik va sintaksistik bosqich umuman kerak bo’lmay qolishi mumkin. Masalan kiritilayotgan vaqtda tizim darhol sintaksistik daraxtni yaratishga kirishadi. Taxrirlashda predmetli orientirli dastur huddi ichkima ichki joylashgan to’rtburchakli taxrirlanuvchi yacheykalar terilmasi sifatida qaraladi. Hamma qolgan yacheykalarni o’z ichida saqlovchi yacheyka masalaga mos keladi. Yacheykalar ichiga ma’lum bir tartibda aniqlangan yacheykalar va yacheykalar daraxti qo’uyilishi mumkin.

Yacheykalar ichida **Masala** yacheykalar ichkima ichki joylashgan:

- **O’quvchi uchun shartlar**, bu holda ikkita yacheyka ichkma ichki joylashgan: konstantali qator bilan «O’quvchi uchun shart», masala shartini tahrirlash qatori bilan beriladi.

- **Berilgan** dastlabki yasashlar bilan ichkima ichki yacheykalar ro’yxati. Bu yacheyka grafik muharrirlar yordamida to’ldiriladi, unda dastlabki geometrik yasashlar qilish mumkin, yacheykada matnlar avtomatik tarzda hosil bo’ladi. Chizmada birdaniga ob’ektlarga nomlar berish mumkin.
- **Yasash kerak**, u glossariyadan ob’ektlarni qo’shimcha qilishni ta’minlaydi (Nuqta, To’g’ri chiziq va xakozo) va ularga nomlar chiqaradi.
- **Shundaylarki**, unda glossariyadan ob’ektlar tanlashni ta’minlaydi (yoki chizmada, yoki ochilgan ro’yxatdan ob’ekt nomini olish) ular uchun bu predikat ishlatilishi mumkin.
- **Instrumentlar**, unga yangi elementlarni qo’shimcha qilish yoki mavjudlarini glossariyadan o’chirish mumkin.

Masala tavsifi xotirada ma’lumotlarning daraxtsimon tarkiblarida saqlanadi va u XML-formatga o’zgartiriladi.

Predmetli orientirli tilni ishlovchi translyatorni yaratish

Hamma klasslar va usullar uchun bu bosqichda bo’sh e’lonlar yaratiladi. So’ngra tahrirlashning umumiy mexanizmini realizatsiya qilish amalga oshiriladi, bu jarayonda masalani kiritish oynasi paydo bo’ladi. So’ngra chizmani yasash mexanizmi bilan bog’lanuvchi mexanizm realizatsiya qilinadi.

V. PREDMETLI ORIENTIRLI DASTUR YARATISH

5.1. **Abstrakt klasslar**

Biz bilamizki ko’pchilik klasslarda virtual funksiyalar aniqlanadi . Ammo shape klassida virtual funksiyalarni aniqlash juda qiyindir:

```
class shape {
  // ...
public:
  virtual void rotate(int) { error("shape::rotate"); }
  virtual void draw()  { error("shape::draw"): }
  // abstakt  figuralarni chizish va burash mumkin emas
  // ...
};
```

Shu sababli yaxshisi shape ning sof virtual funksiyasini virtual aniqlash lozim:

```
class shape {
 // ...
public:
 virtual void rotate(int) = 0; // sof  virtual funksiya
 virtual void draw() = 0;      // sof  virtual funksiya
};
```

O’zida virtual funksiya aniqlagan klass фиыекфле klasslar deb ataladi. Bunday klasslarning ob’ektlarini yaratish mumkin emas. Abstrakt klasslardan faqat boshqa klasslar uchun bazaviy klass shaklida foydalanish mumkin:

```
class circle : public shape {
 int radius;
public:
 void rotate(int) { } // shunday aniqlash mumkin:
// qayta aniqlash  shape::rotate
 void draw();         shunday aniqlash mumkin:
// qayta aniqlash  shape::draw
```

```
  circle(point p, int r);
};
```

Agar sof virtual funksiya hosila klassda aniqlanmasa, u holda hosila klass ham abstrakt klass bo'lib qolaveradi. Bunday vazziyatlarda klasslarni boshqichma bosqich realizatsiya qilish lozim bo'ladi:

```
class X {
public:
  virtual void f() = 0;
  virtual void g() = 0;
};
X b;   // xato: X abstrakt klassini tavsiflash nato'g'ri bajarilgan
class Y : public X {
  void f();  // qayta aniqlash  X::f
};
Y b;   // xato: Y abstrakt klassini tavsiflash nato'g'ri bajarilgan
class Z : public Y {
  void g();  // qayta aniqlash  X::g
};
Z c;   // shunday aniqlash mumkin
```

Misol. Biz geometrik figuralarni ekranga chizishning obektga mo'ljallangan dasturini yozamiz. Ekranda chizish uchta qismdan iborat bo'ladi:

- ekran monittori: ekran bilan ishlash uchun kerak bo'lgan funksiyalar va axboratlar tarkiblari, bu yerda faqat nuqta , chiziqlar bilan operatsiya qilinadi;
- figuralar kutubxonasi: umumiy ko'rinishdagi figuralarni aniqlash to'plamlari (masalan, to'rtburchak, aylana…) va ular bilan ishlovchi standart funksiyalar;
- Amaliy dastur: masalaga tegishli figuralarni konkret aniqlash hamda ular bilan ishlash funksiyalarini aniqlash.

Odatda, bu qistur qismlari har xil daturchilar tomonidan va har xil vaqtlarda , har xil tashkilotlarda bajariladi.

5.2. Ekran monitori

Monitor ekrani ikki o'lchamli simvollar massivi sifatida qaraladi va u put_point() va put_line() funksiyalari bilan boshqariladi. Bu yerda ekran bilan bog'lanishda **point** strukturasi ishlatiladi:

```
//  screen.h fayli
const int XMAX=40;
const int YMAX=24;
struct point {
   int x, y;
   point() { }
   point(int a,int b) { x=a; y=b; }
};
extern void put_point(int a, int b);
inline void put_point(point p) { put_point(p.x,p.y); }
extern void put_line(int, int, int, int);
extern void put_line(point a, point b)
  { put_line(a.x,a.y,b.x,b.y); }
extern void screen_init();
extern void screen_destroy();
extern void screen_refresh();
extern void screen_clear();
#include <iostream.h>
```

Ekranga chiqaruvchi funksiyalarni chaqirish uchun (put_...), ekranni inisializatsiya qiluvchi funksiyaga murojaat qilish lozim (screen_init()). Quyida biz ekranni boshqarish va axborotlarni tarkiblash funksiyalarini keltiramiz:

```
#include "screen.h"
```

```
#include <stream.h>
enum color { black='*', white=' ' };
char screen[XMAX] [YMAX];
void screen_init()
{  for (int y=0; y<YMAX; y++)
     for (int x=0; x<XMAX; x++)
 screen[x] [y] = white;
}
```

Nuqtalar faqat ular ekranga tushishganda tavsiflanadi:

```
       inline int on_screen(int a, int b)  // ekranga tushishini tekshirish
{
  return 0<=a && a <XMAX && 0<=b && b<YMAX;
}
void put_point(int a, int b)
{
   if (on_screen(a,b)) screen[a] [b] = black;
}
```

To’g’ri chiziqlarni chizish uchun put_line() funksiyasidan foydalaniladi:

```
void put_line(int x0, int y0, int x1, int y1)
/*
   (x0,y0) - (x1,y1)-  Kesmani chizish.
   To’g’ri chiziq tenglamasi b(x-x0) + a(y-y0) = 0.
    abs(eps) –kattaligi minimizatsiya qilinadi
 eps = 2*(b(x-x0)) + a(y-y0).
   Newman, Sproull
   ``Principles of interactive Computer Graphics"
 */
 {
   register int dx = 1;
   int a = x1 - x0;
```

```
  if (a < 0) dx = -1, a = -a;
  register int dy = 1;
  int b = y1 - y0;
  if (b < 0) dy = -1, b = -b;
  int two_a = 2*a;
  int two_b = 2*b;
  int xcrit = -b + two_a;
  register int eps = 0;
  for (;;) {
    put_point(x0,y0);
    if (x0==x1 && y0==y1) break;
    if (eps <= xcrit) x0 +=dx, eps +=two_b;
    if (eps>=a || a<b) y0 +=dy, eps -=two_a;
  }
 }
```

Ekranni tozalash va yangilash funksiyasi quyidagicha aniqlanadi:

```
void screen_clear() { screen_init(); }
void screen_refresh()
{
 for (int y=YMAX-1; 0<=y; y--) { // yuqorgi qatordan quyi qatorgacha
    for (int x=0; x<XMAX; x++)   //chap ustundan to o’ng ustunga qadar
 cout << screen[x] [y];
    cout << '\n';
 }
      }
```

Bu aniqlanishlar biror bir kutubxonada saqlanadi.

53. Geometrik figuralar kutubxonasini yaratish

Kutubxonani yaratish uchun figura to’g’risidagi tushunchani aniqlab olamiz. Bu aniqlanishda (shape bazaviy klassi sifatida) har xil figuralarni aniqlovchi(aylana,

kvadrat, .. va boshqalar) klasslarda bazaviy klass sifatida aniqlanishi lozim. Bu tushuncha shuningdek shape klassi bilan aniqlanuvchi figuralar bilan ishlayoluvchi interfeys yordamida boshqarilishi lozim:

```
struct shape {
  static shape* list;
  shape* next;
  shape() { next = list; list = this; }
  virtual point north() const = 0;
  virtual point south() const = 0;
  virtual point east() const = 0;
  virtual point west() const = 0;
  virtual point neast() const = 0;
  virtual point seast() const = 0;
  virtual point nwest() const = 0;
  virtual point swest() const = 0;
  virtual void draw() = 0;
  virtual void move(int, int) = 0;
 };
```

Figuralar ekranga draw() funksiyasi yordamida joylashtiriladi. Figuralarni bir birlariga nisbatan joylashtirish ham mumkin, masalan kontakt nuqtalar yordamida. Kontakt nuqtalarni belgilashda kompasning ko'rsatgichlari nomlari ishlatiladi: north - shimol, ... , neast – shimoliy-sharq, ... , swest - janub-sharq. Har bir konkret figuralarning klassi bu nuqtalarning mazmunini aniqlaydi va uni qanday chizishni ham aniqlaydi. Biz yaratgan shape::shape() konstruktori figuralar ro'yxati shape::listga yangisini kiritadi. Bu ro'yxatni qurish uchun next -a'zo dan foydalaniladi.

To'g'ri chiziqli kesmani aniqlash uchun ikkita nuqta aniqlanishi lozim :

```
class line : public shape {
/*
```

```
  To’g’ri kesma ["w", "e" ]
  north()  -nuqtani aniqlaydi
*/
  point w, e;
        public:
  point north() const
    { return point((w.x+e.x)/2,e.y<w.y?w.y:e:y); }
  point south() const
    { return point((w.x+e.x)/2,e.y<w.y?e.y:w.y); }
  point east() const;
  point west() const;
  point neast() const;
  point seast() const;
  point nwest() const;
  point swest() const;
  void move(int a, int b)
     { w.x +=a; w.y +=b; e.x +=a; e.y +=b; }
  void draw() { put_line(w,e); }
  line(point a, point b) { w = a; e = b; }
  line(point a, int l) { w = point(a.x+l-1,a.y); e = a; }
        };
```

Huddi shu usulga o’xshash to’g’ri to’rtburchak aniqlanadi:

```
        class rectangle : public shape {
        /*   nw ------ n ----- ne
    |       |
    |       |
    w        c        e
    |       |
    |       |
    sw ------ s ----- se
```

```
        */
 point sw, ne;
        public:
  point north() const { return point((sw.x+ne.x)/2,ne.y); }
  point south() const { return point((sw.x+ne.x)/2,sw.y); }
  point east() const;
  point west() const;
  point neast() const { return ne; }
  point seast() const;
  point nwest() const;
  point swest() const { return sw; }
  void move(int a, int b)
  { sw.x+=a; sw.y+=b; ne.x+=a; ne.y+=b; }
  void draw();
  rectangle(point,point);
};
```

To’rtburchak ikkita nuqtasi bilan quriladi. Konstruktor esa ularning joylashishini aniqlaydi:

```
rectangle::rectangle(point a, point b)
{
  if (a.x <= b.x) {
    if (a.y <= b.y) {
      sw = a;
      ne = b;
    }
    else {
      sw = point(a.x,b.y);
      ne = point(b.x,a.y);
    }
  }
```

```
  else {
   if (a.y <= b.y) {
      sw = point(b.x,a.y);
      ne = point(a.x,b.y);
   }
   else {
      sw = b;
      ne = a;
   }
  }
}
```

To’rburchakni qurish uchun to’rtta kesma chizish kerak bo’ladi:

```
void rectangle::draw()
{
  point nw(sw.x,ne.y);
  point se(ne.x,sw.y);
  put_line(nw,ne);
  put_line(ne,se);
  put_line(se,sw);
  put_line(sw,nw);
}
```

Figuralar kutubhonasida figuralar va ular bilan ishlash funksiyalari aniqlangan:

void shape_refresh(); // hamma figuralarni chizish

void stack(shape* p, const shape* q); // p ni q ga joylashtirish

Endi ko’pburchakli elementlarni aniqlaymiz. Qavariq n=burchakli elementni chizsh C++ dasturini keltiramiz.

- #include <stdlib.h>
- #include< iosteream.h>
- #include<graphics.h>//grafika funsiyalari bilan ishlash uchun
- #include<math.h> // trigonometrik funksiyalar bilan ishlash uchun

```
void n_burchak(int R, int n)
{ int a=0;
pointtype*p=new pointtype[n];// Nuqtalarni saqlash massivi
int x=getmaxx()/2; Ko'pburchak markazini x o'qi bo'yicha ekran markaziga joylashtiramiz
int y=getmaxy()/2; Ko'pburchak markazini y o'qi bo'yicha ekran markaziga joylashtiramiz
 double pi=3.14;
//nuqtalarni izlsah
for(int i;i<n*2+2;i++)
 { p[i].x=x+R*cos(a*pi/180);
 p[i].y=y-R*sin(a*pi/180);  }
 a=a+180/n;}
 p[n*2+1].x=p[1].x; p[n*2+1].y=p[1].y;
 moveto(p[1].x,p[1].y);
 for(i=1;i<n*2+2;i+=2)
{ lineto(p[i].x,p[i].y);}
delete []p; //xotirani tozalaymiz}
 //parametrlarni kiritish funksiyasi
void input()
{ int R,n;// R- tashqi chizilgan aylana radiusi, hamda ko'pburchak tomanlari soni
cout <<"Radius="; cin >>R;
cout <<"Tomonlar soni="; cin >>n;
n_burchak(R,n); } // n-tomonli kopburchakni yasash
 void main()
{system("CLS");
int gdriver=DETECT, gmoge,errorcode;
initgraph(&GDRIVER,&GMODE,"");
input(); // parametrlarni kiritamiz va ko'pburchakni  yasaymiz
```

- system(“PAUSE”);
- }

Bu bobda biz predmetli orientirli yondoshuwning afzalligini , predmetli orientirli tillardan foydalanish modelida namoyish etdik. Yuqorida keltirilgan misollar predmetli orientirli yondoshuvning juda effektiv ekanligini isbotlaydi.

5.4. Dasturni yaratish texnologiyasi

Dasturlash muhitini tanlash va dasurni yaratish. **Biz dasturni yaratish uchun Visual Studio .Net muhitini tanlaymiz. Visual Studio .Net – ochiq dastur yaratish muhiti**

Visual Studio .Net dastur yaratish muhiti hozirgi vaqtda juda ishonchli dasturlash muhiti sifatida e’tirof etilmoqda. Hozirgi vaqtda Microsoft kompaniyasining hamma yangi dasturiy maxsulotlari .Net bilan nomlanib bormoqda. Bu dasturga quidagi juda muhim ikkita g’oya amalgam oshirilgan:

- Dasturlash tillari uchun ochiqlik**;**
- **Framework .Net**. muhiti uchun karkasni qurishdagi prinsipial yangi yondoshuv.

Ochiqlik. *Muhit endi ochiq tillar muhiti hisoblanadi. Bu holda Microsoft firmasi muhitga kiritgan - Visual C++ .Net , Visual C# .Net, J# .Net, Visual Basic .Net, - dasturlaridan tashqari istalgan algoritmik til qo’shimcha qilinishi mumkin.*

Framework .Net – da yaratish muhiti uchun bir xil karkas ishlattiladi.

Framework .Net karkasida quyidagi ikkita asosiy komponentani keltirish mumkin:

- statistik - ***FCL (Framework Class Library)*** – karkaslar klasslari kutubxonasi;
- dinamik - ***CLR (Common Language Runtime)*** – umumtil bajarish muhiti.

Karkaslar yahlitligi. Karkas hamma dasturlash tillari uchun bir xil bo'lib qoldi. Shu sababli qaysi tilda dastur yozilishidan qat'iy nazar ular bitta klasslar kutubxonasidan foydalanishadi.

Ichki primitiv turlar. FCL kutubxonasining muhim qismini klasslar tashkil etadi, ular primitiv turlarni aniqlaydi. Karkas turlari dasturlash tillaridagi hamma

a turlarni qamrab oladi.

Tarkibiy turlar. Kutubxonaning bir qismini bu turlar egallashdi, shuningdek tarkibiy turlar ham ularga qoshilishdi.

Ilova arxitekturasi. Ilovalar yaratish arxitekturalari soni ko'paydi. Odatdagi ananaviy Windows ilovalardan tashqari endi web ilovalar ham yaratish imkoniyati tug'ildi. Qayta foydalaniladigan komponentalar yaratishga katta e'tibor berilgan. Endi klasslar, boshqarish elementlari kutubxonasi, hamda web elementlarni boshqarish kutubxonasi ham yaratilmoqda.

C# dasturlash muhitining yaratilishi

C# tili dasurlash tillari sohasidagi eng katta qiziqish uyg'atgan tildir. Bu til Microsoft korporatsiyasi hodimi Andreas Xeylsberg tomonidan yaratilgan. Andreas Xeylsberg ning fikricha, C# komponentali dasturlash tili sifatida yaratilgan, va aynan shu imkoniyati tilning bosh yutug'i hisoblanadi va u yaratilgan komponentalardan qayta foydalanish imkoniyatini yaratadi.

Bundan tashqari C# quyidagi imkoniyatlarga ham ega:

- C# dasturlash muhiti Framework.Net karkasi bilan parallel ravishda yaratildi va uning hamma imkoniyatini hisobga oldi, masalan FCL, va CLR imkoniyatlarini ham;
- C# to'lig'icha ob'ektga mo'ljallangan dasturlash tilidir
- C# da meroslash va universalizatsiya imkoniyatlari mavjud;
- C# muhiti C/C++ tillarining eng yaxshi tomonlarini meroslagan.

Loyixa turlari va ularni yaratish

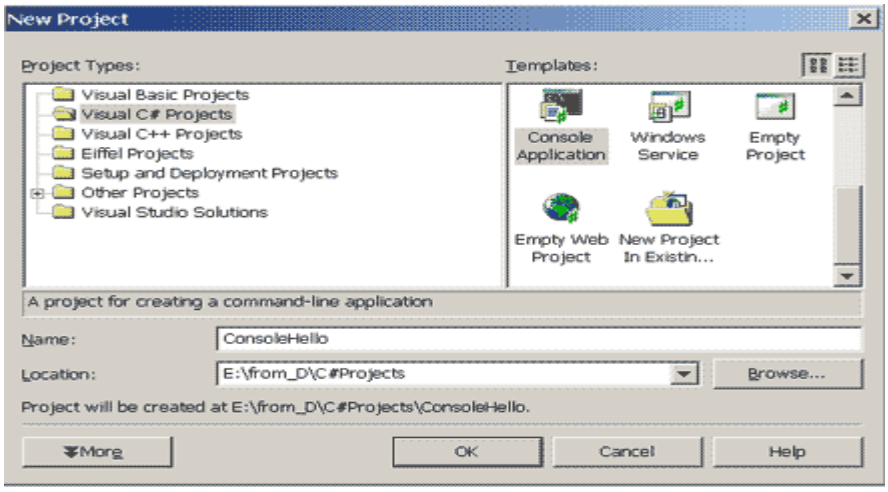
Visual Studio .Net muhiti C# tili uchun 12 ta loyixa turini havola qiladi. Ularning ichida bosh loyixa, Web xizmatiga mo'ljallangan loyixalar ham bor.

Loyixanni yaratishda quyidag tushunchalar ishlatiladi: Echish *(solution)*, loyixa *(project)*, nomlar to'plami *(namespace)*, yig'ish *(assembly)*. Endi loyixa bilan ishlayotgan Visual Studio kompillyatorining dasturchi nuqtai nazaridan ishlash natijasini ko'raylik hamda PE fayllarini protsessor kodiga kompilliyatsiya qilishini ko'raylik.

Dasturchi nuqtai nazaridan qaralganda, kompillyator yechimni yaratadi, CLR nuqtai nazaridan esa - u PE fayllarni yig'adi. Dasturchi esa yechimlar bilan ishlaydi, CLR – esa yig'ishni amalgam oshiradi.

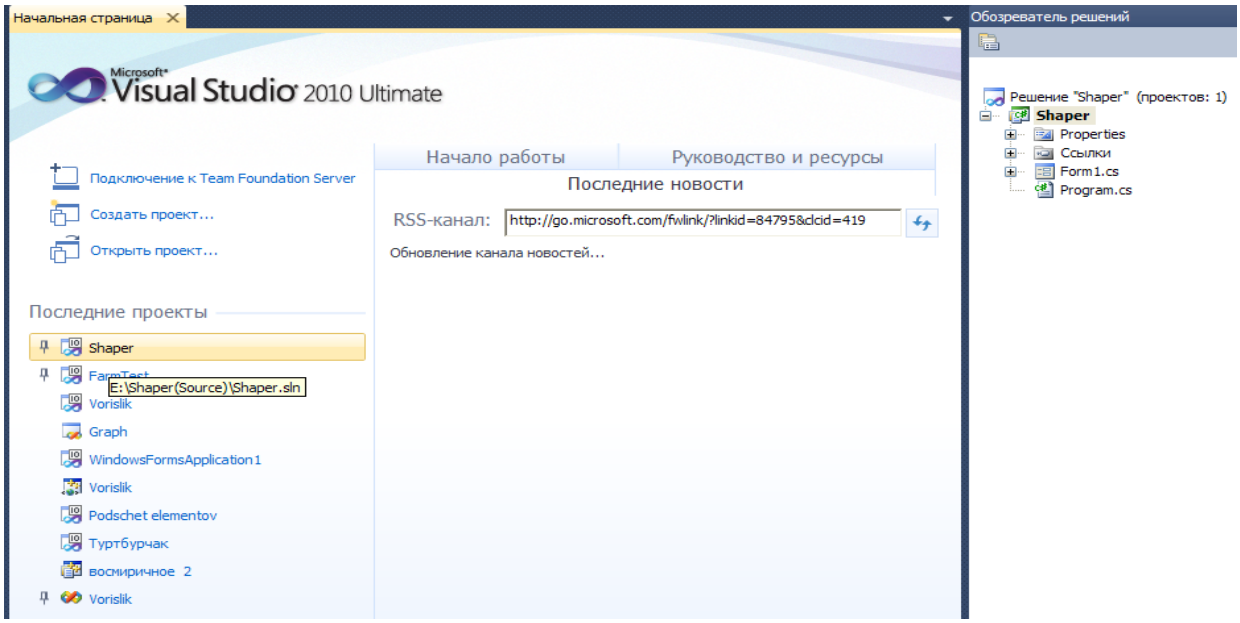
Loyixa bir nechta fazoviy nomlarga yig'ilgan klasslardan iborat. Nomlar fazosi juda ko'p klasslarga ega bo'lgan loyixalarni tarkiblashni amalga oshiradi. Agar loyixa ustida bir nechta bajaruvchilar ishlayotgan bo'lsa ularning har biri o'z nomlari fazosini tashkil etadi.

Konsol loyixa. Visual Studio .Net 2003 ochamiz va C# da yangi loyixa yaratishga kirishamiz , biz konsol ilova yaratamiz va unga - Shaper deb nom beramiz, endi bizning loyixamiz shu nom bilan saqlanadi(1-rasm).

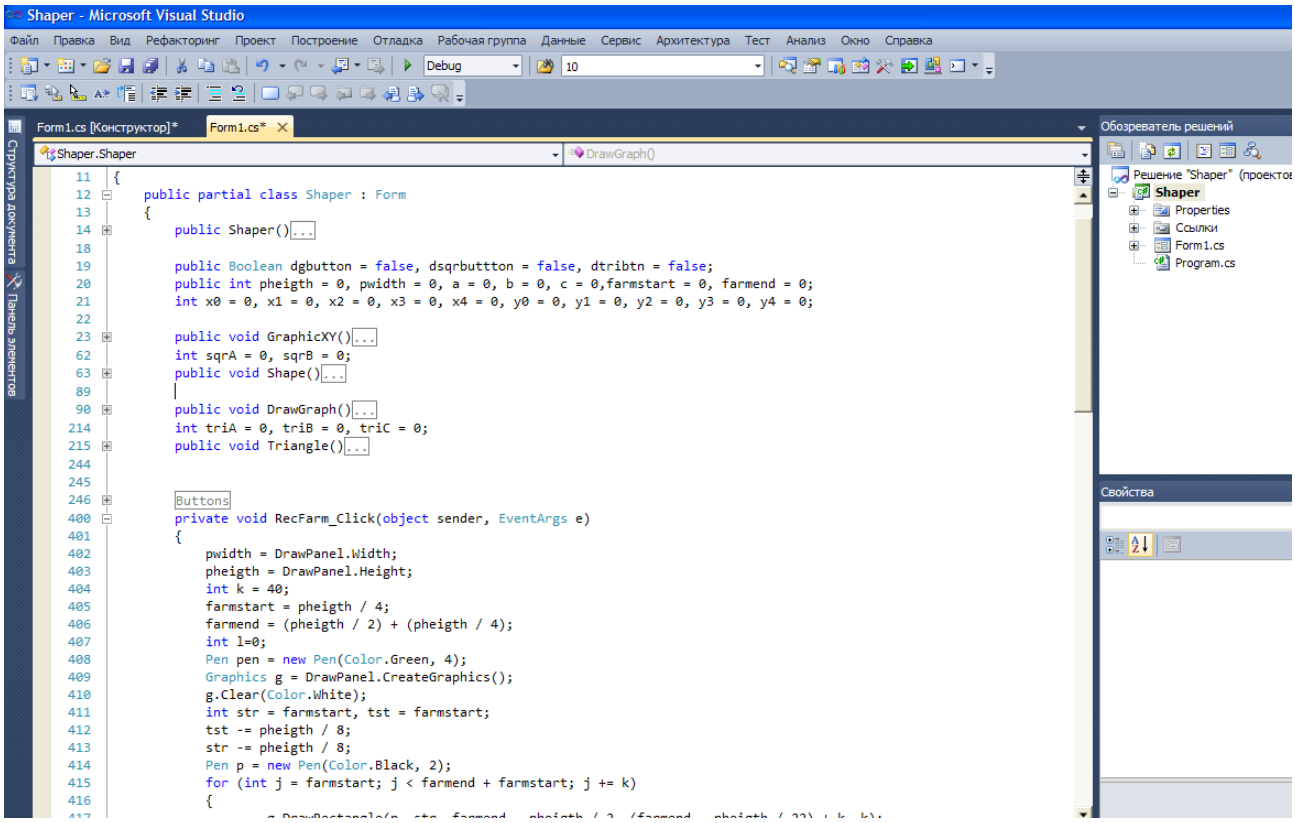


1-rasm. Yangi loyixa yaratish oynasi

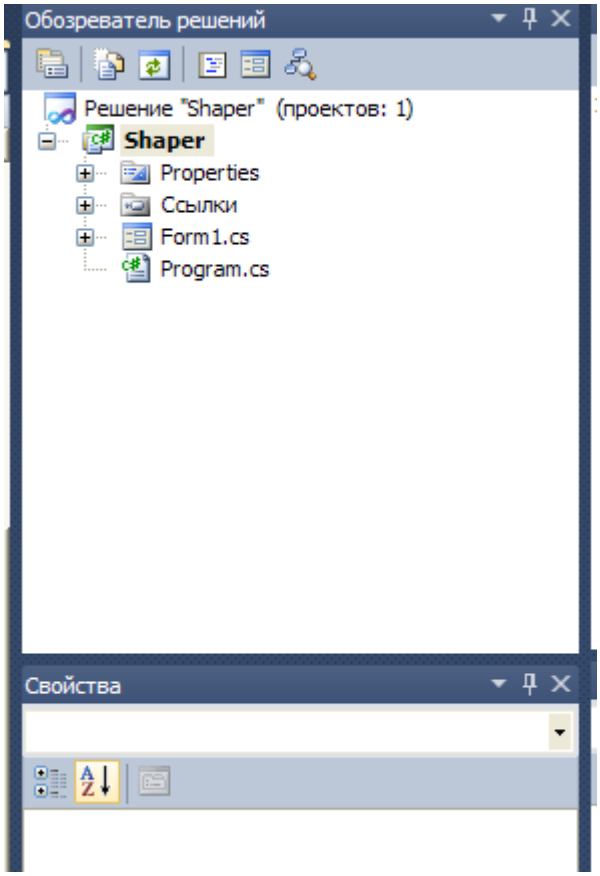
Agar shu o’rnatilishlarni qabul qilsak, holda kompilliyator yechini yaratadi, uning nomi loyxa nomi bilan bir xil bo’ladi. Bu yechimningg ko’rinishi 2-rasmda keltirilgan. IDE (Integrated Development Envirionment) Visual Studio inegrallashgan muhiti ko’p oynali va katta imkoniyatlar terilmasiga ega. Ularning ba’zilarini ko’raylik. Solution Exolorer oynasida tanlanilgan yechim elementi xususiyatini ko’rish mumkin.Hujjatlar oynasida tanlanilgan hujjat ifodalaniladi. Bizning holda esa Shaper.Class1 ifodalaniladi.



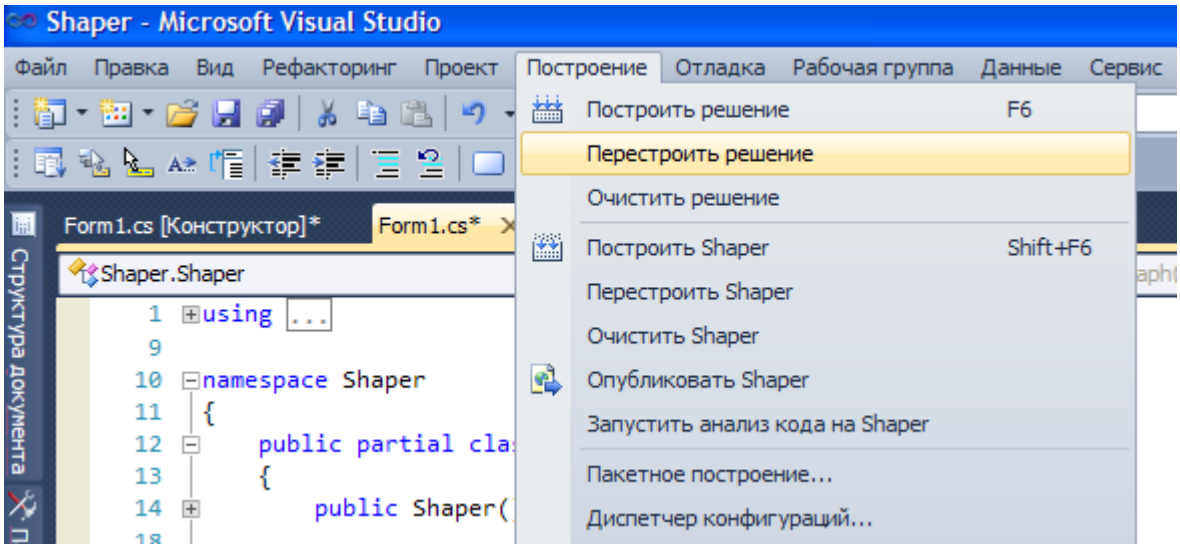
2-rasm . Loyixani yaratish



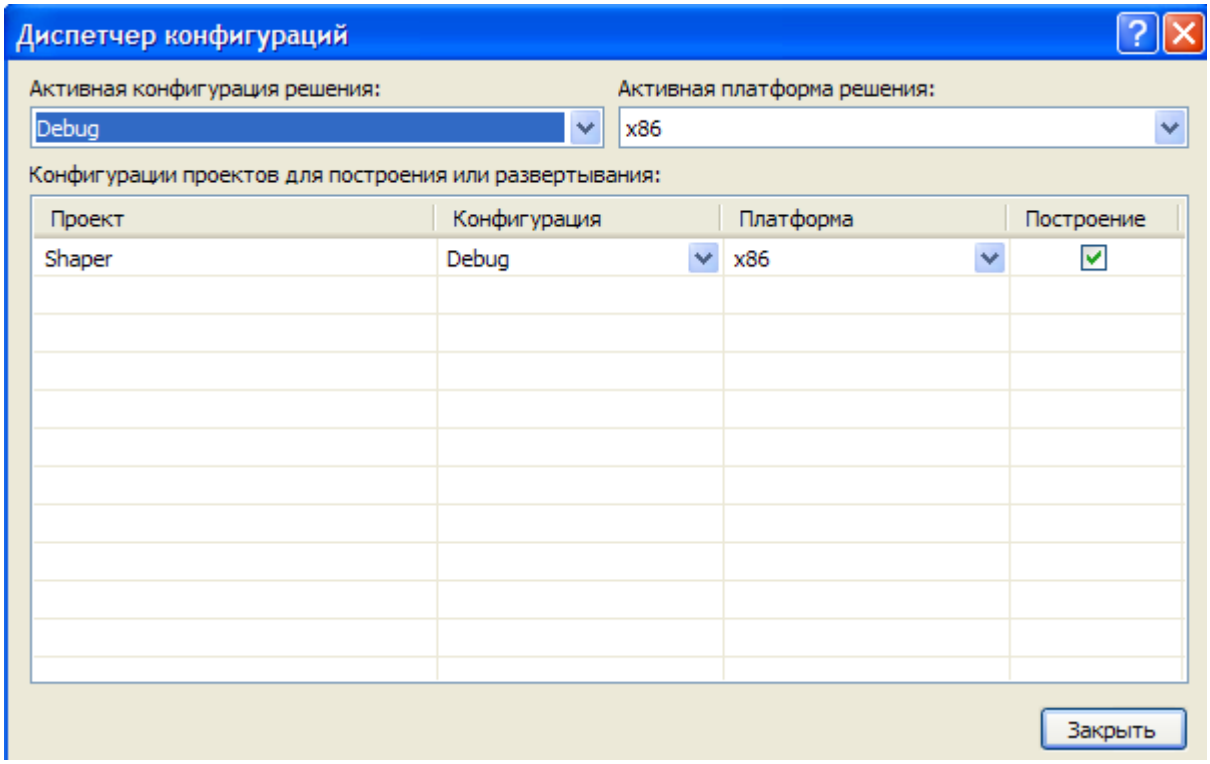
3-rasm. Loyixa kodini yaratish.



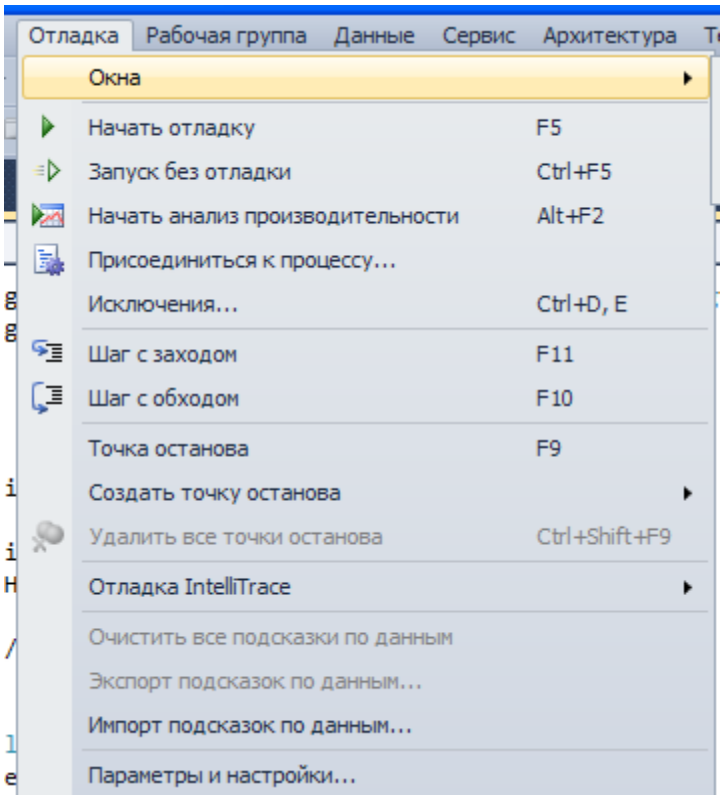
4-rasm. Ychimlar iyerarxiyasi



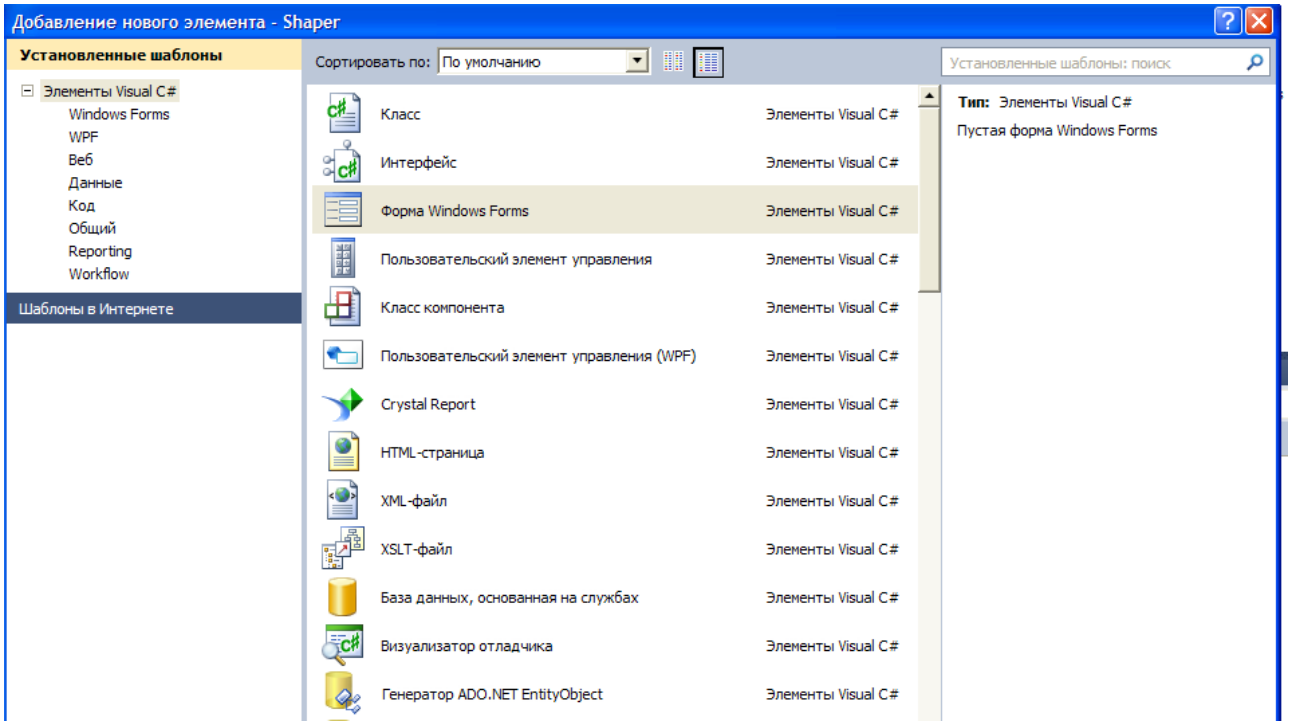
5-rasm. Yechimni qurish menusi.



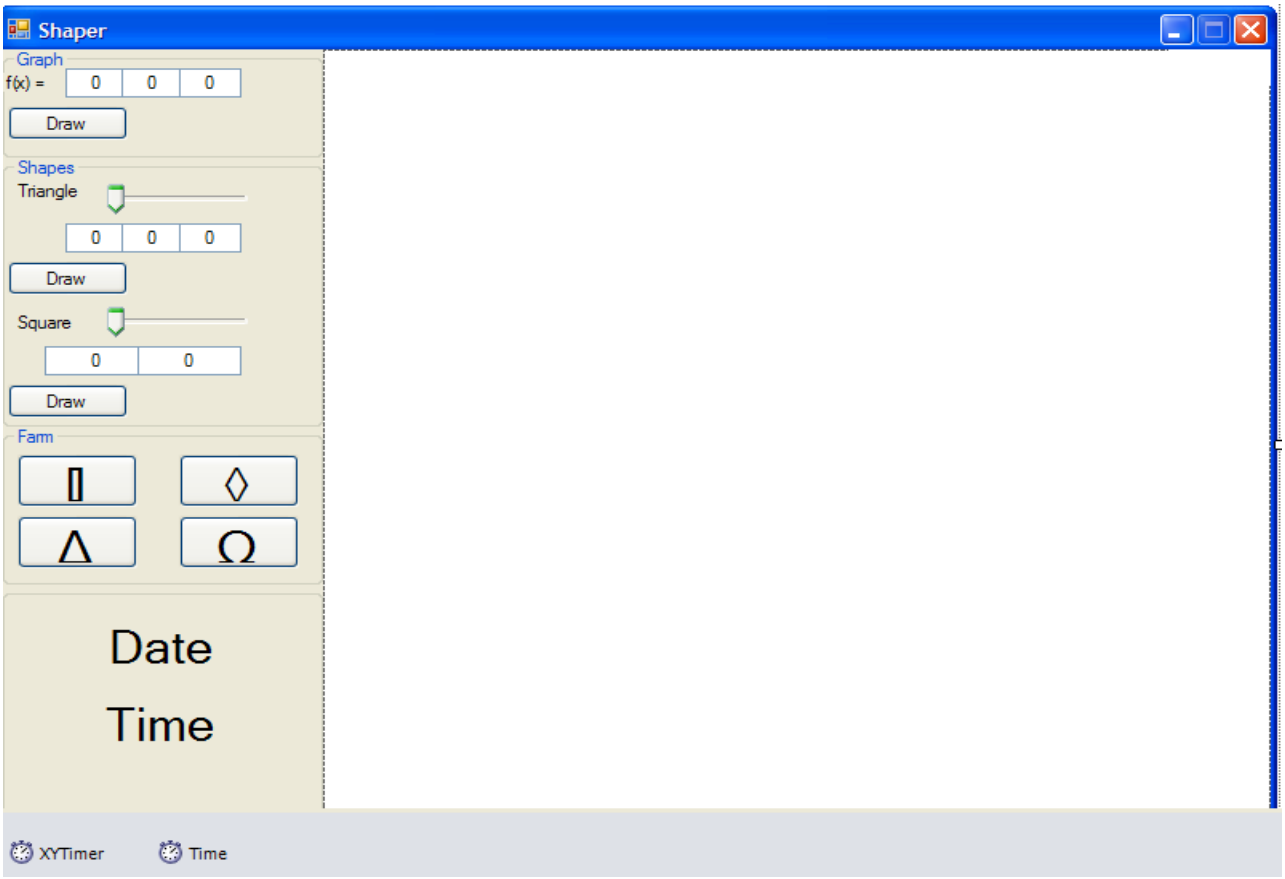
6-rasm . Kongiguratsiya dispetcheri ounasi.



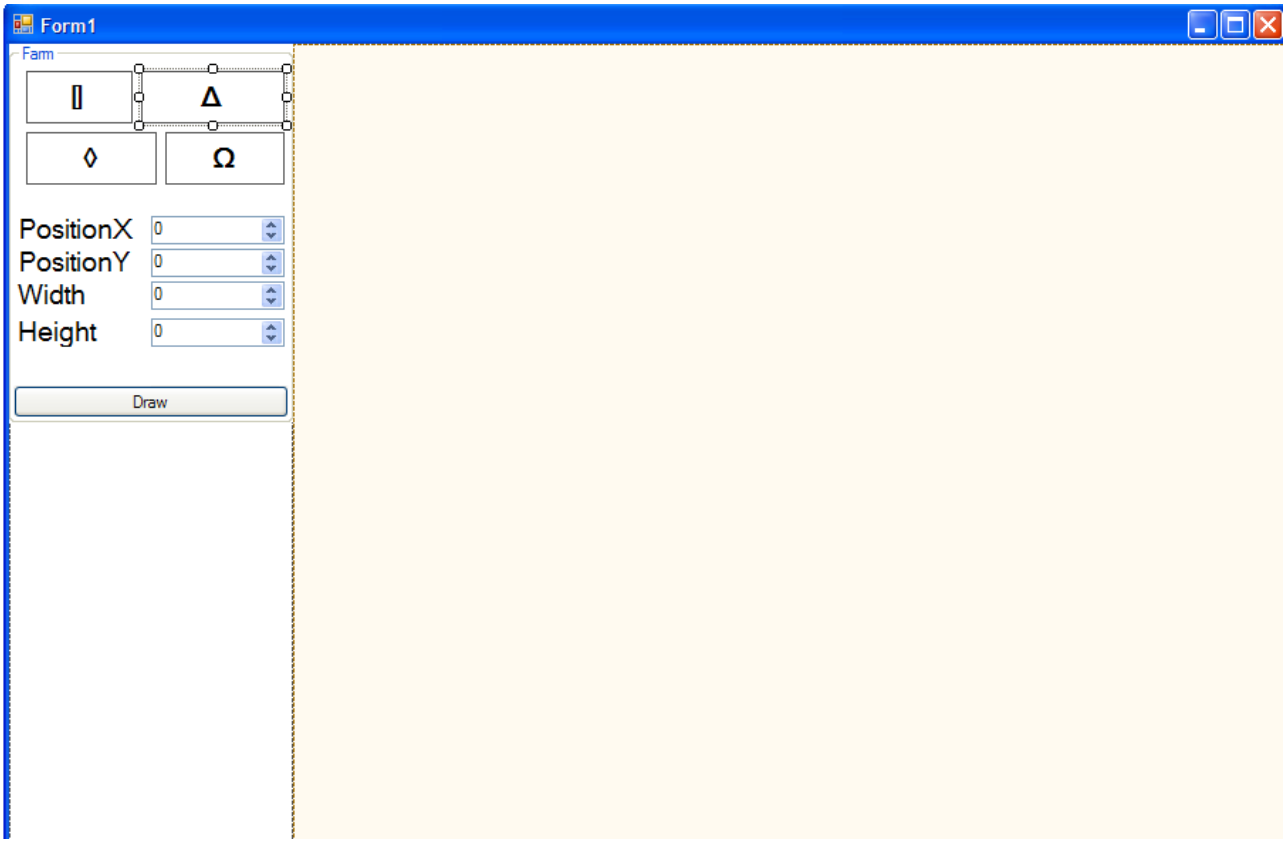
7-rasm. Dastur kodini to’g’rilash menusi.



8-rasm. Shaperga yangi element qo’shish oynasi.



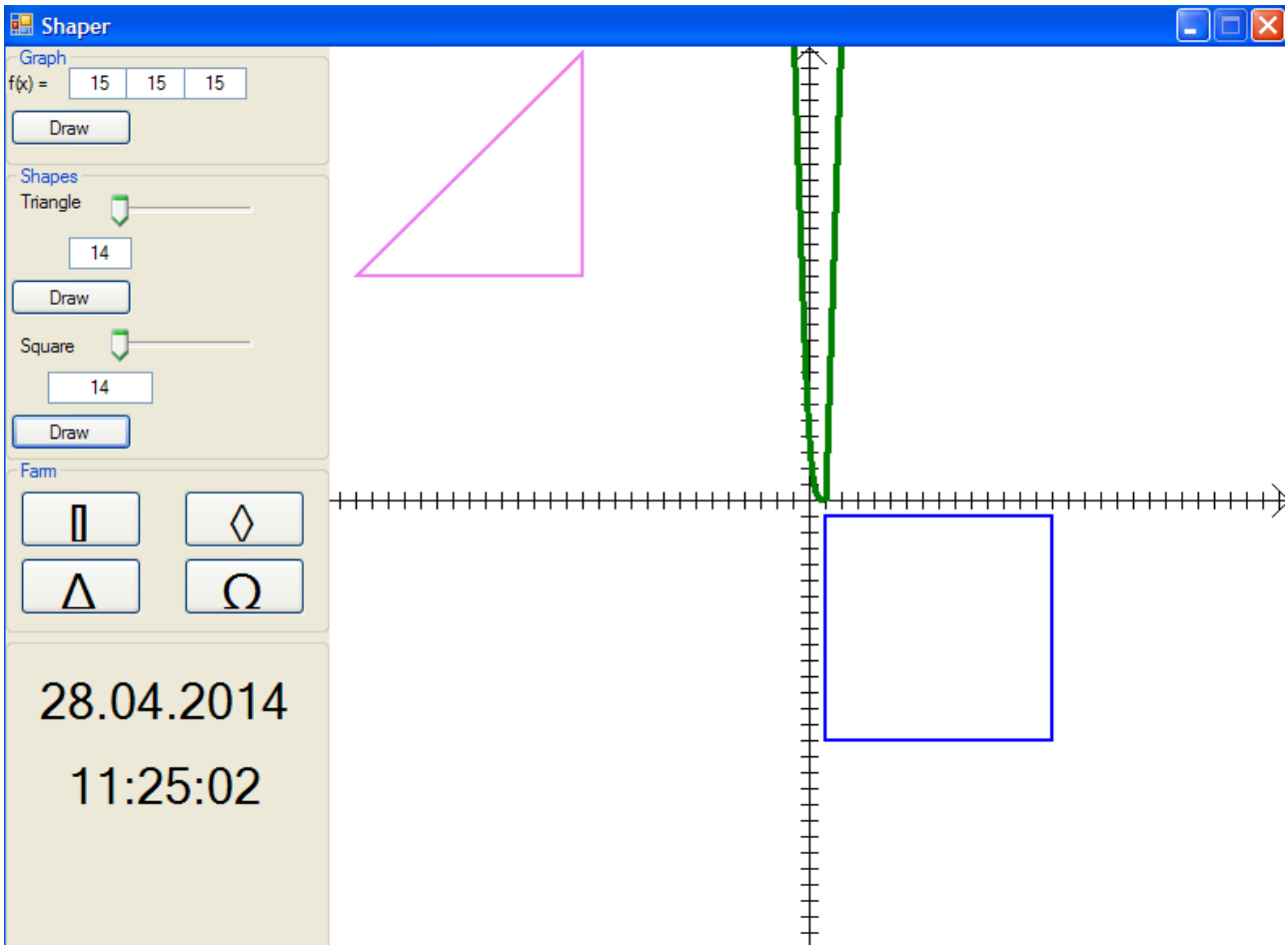
9a-rasm. Loyixaning maket formasi.



9b-rasm. Loyixaning maket formasi.

Aynan maket formada dastur interfeysi shakillaniriladi. Visual Studio inegrallashgan muhitida formalarni yaratish va unga boshqarish elementlarini o'tkazish juda osonlashtirilgan, shu bilan birgalikda bboshqarish elelemntlari soni va xususiyatlar imkoniyatlari kengaytirlgan. Shunuu qayd qilish kerakki C++Builderda ham bu imkoniyatlarning bir qismi bor edi, ammo hech qachon Visual Studio inegrallashgan muhitidagicha bo'lgan emas edi.

Quyidagi rasmda(10-rasm) bazaviy klasslardan foydalanib ba'zi chizmalar hosil qilish va natijalarni formaga chiqarish amalga oshirilgan(10-rasm).

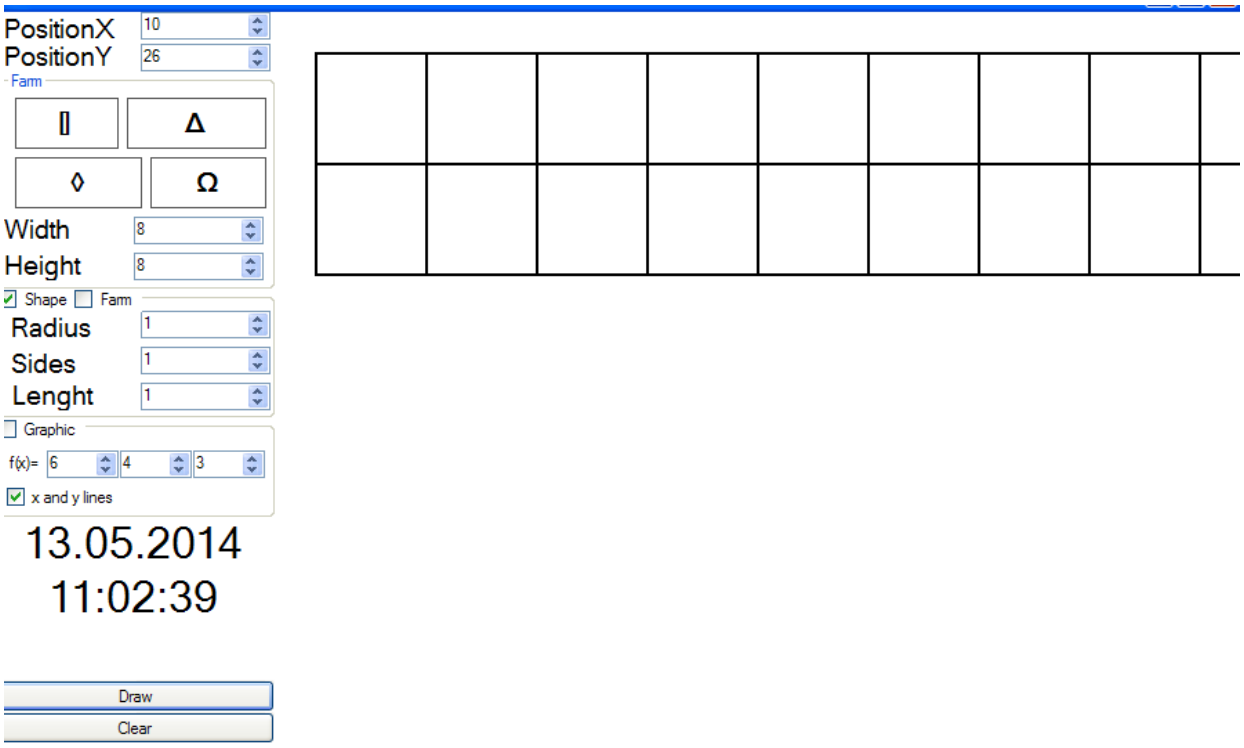


10-rasm. Bazaviy klasslardan foydalanib chizmalar hosil qilish.

Dissertatsiy ishining maqsadi bu bazaviy klasslardan foydalanib murakkab chizmalarni hosil qilishdan iborat. Shu maqsadda biz qurilish konstruksiyalarini loyixalash va chizishni amalga oshiruvchi masalalarni yechishga kirishamiz. Biz dastlab qurilish konstruksiyalari, xususan fermalarni loyixalash va chizishni

to’rtburchakli elementlardan tashkil topgan fermalarni chizishdan boshlaymiz. Quyida

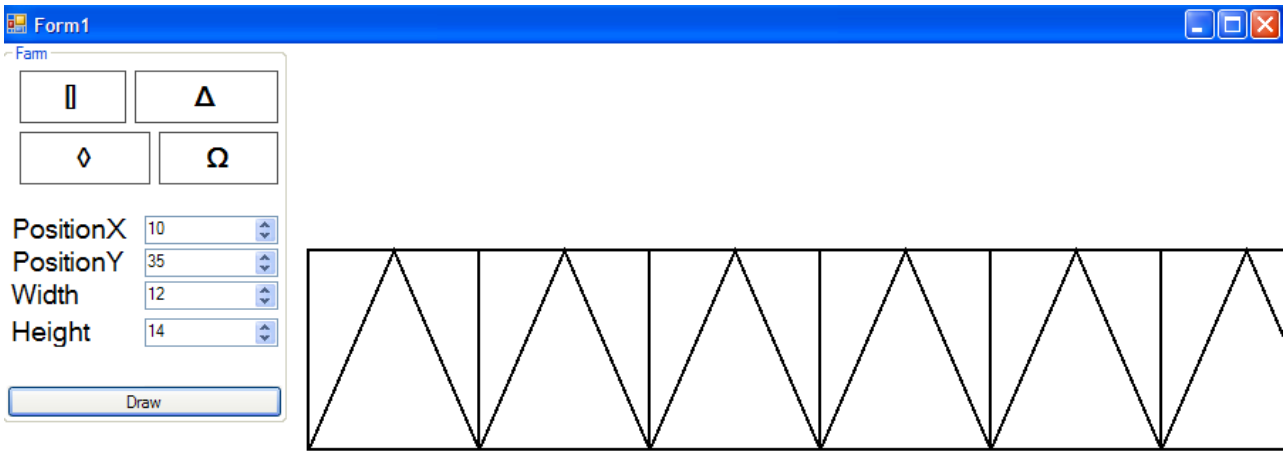
Aynan to’rtburchakli konstruksiyali ferma chizmasi keltirilgan(11-rasm)



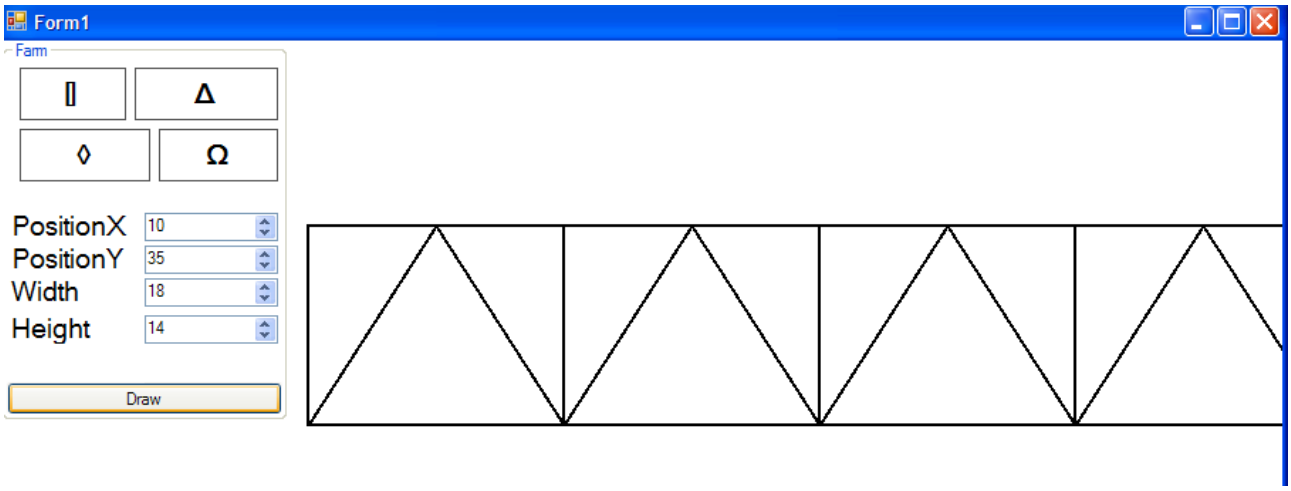
11a-rasm. To’rtburchakli konstruksiyali ferma chizmasi .

Shuni aytish lozimki biz konstruksiya chizmasini formaning istalgan joyiga joylashtirishimiz mumkin. Bu imkoniyat konstruksiyalarni birlashtirib murakkab konstruksiyalar yaratish maqsadida yaratilgandir.Shu maqsadda formaga PozitionX va PozitionY parametrlari kiritilgan. Ikkinchi parametrlar guruhi Width va Height lar esa bevosita konstruksiya masshtabini o’zgartirish uchun xizmat qiladi. Masalan quyidagicha o’zgartirish mumkin:

Endi biz qurilish konstruksiyalari, xususan fermalarni loyixalash va chizishni uchburchakli i elementlardan tashkil topgan fermalarni chizishni ko’ramiz, u quyidagi (12-rasm) keltirilgan.

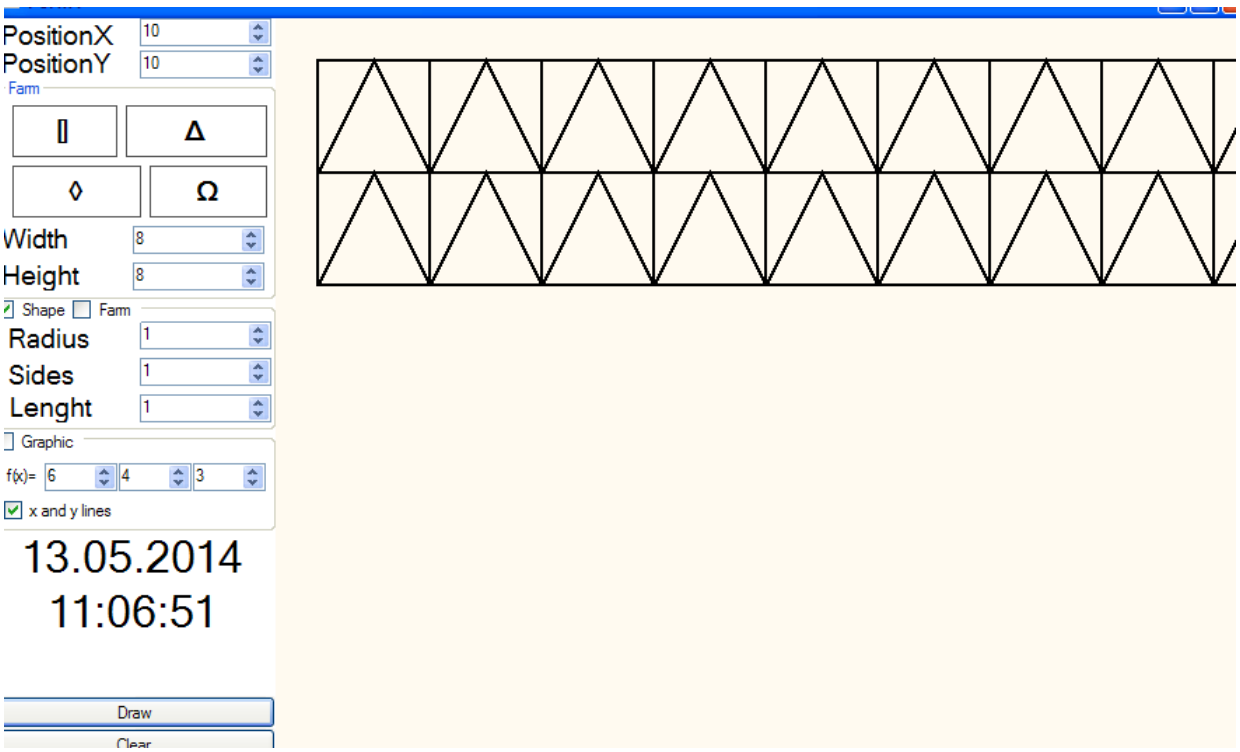


12a-rasm. Uchburchakli ellementlardan konstruksiyalangan ferma chizmasi .

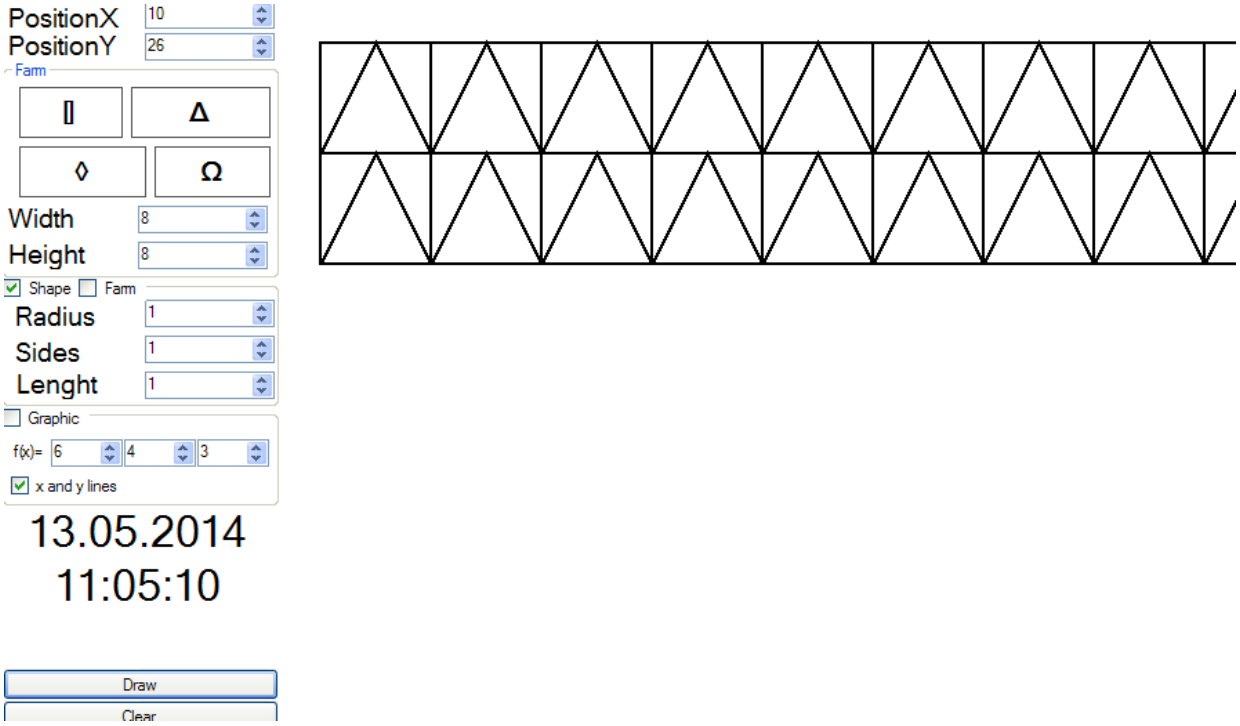


12b-rasm. Uchburchakli ellementlardan konstruksiyalangan ferma chizmasi .

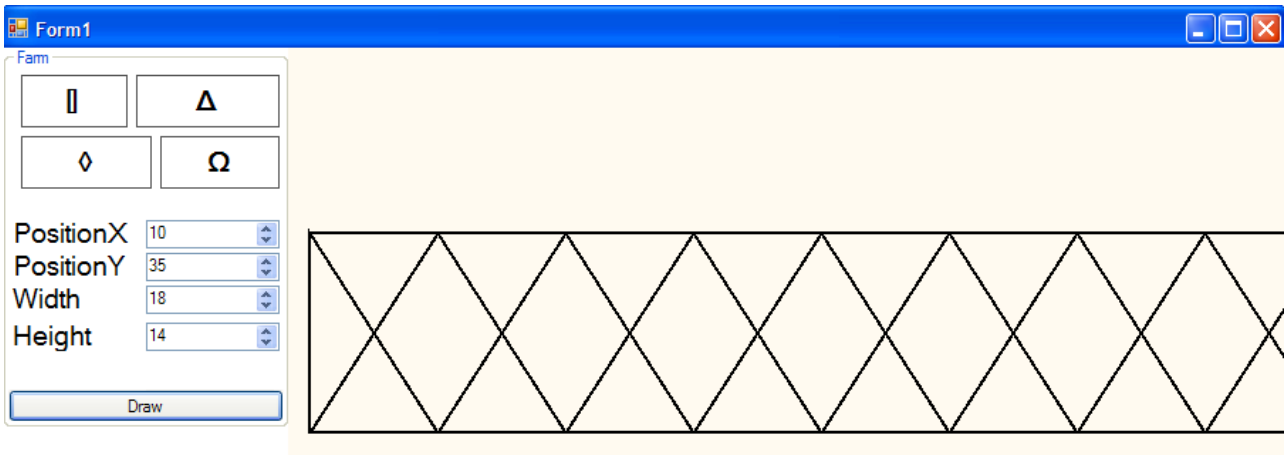
Endi biz qurilish konstruksiyalarini murakkablashtirib boramiz, xususan fermalarni loyixalashda ikkita uchburchaklarni asoslari bo'yicha birlashtirish natijasida fermalarni konstruksiyalash masalaini yechishni ko'ramiz va chizishni dasur asosida amalga oshiramiz. (13-rasm).Bu holda biz uchburchakli elementlarni birlashtirib yanada murakkab konstruksiyalar hosil qilishimiz mumkin. Dasurda bunday masalalarni yechish ham amalga oshirilishi mumkin.



13.rasm. Uchburchakli ferma konstruksiyasi

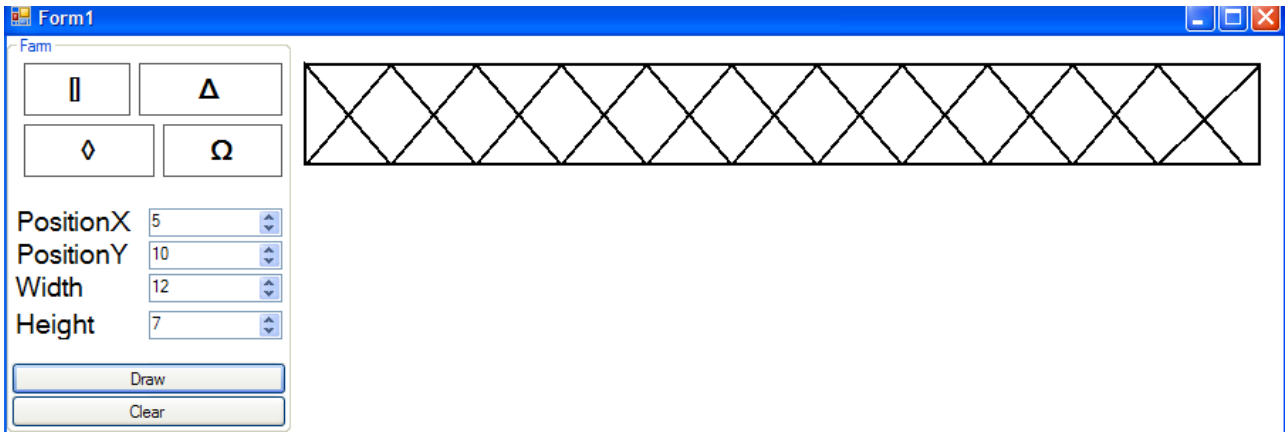


14-rasm . Uchburchakli ferma konstruksiyasini murakkablashtirish

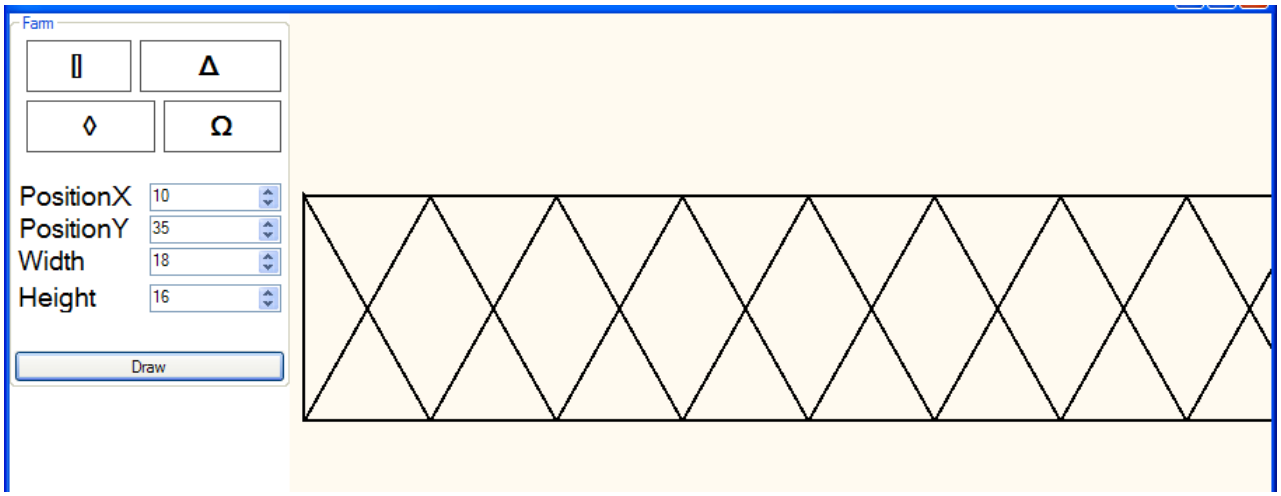


15a-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan ferma chizmasi

.

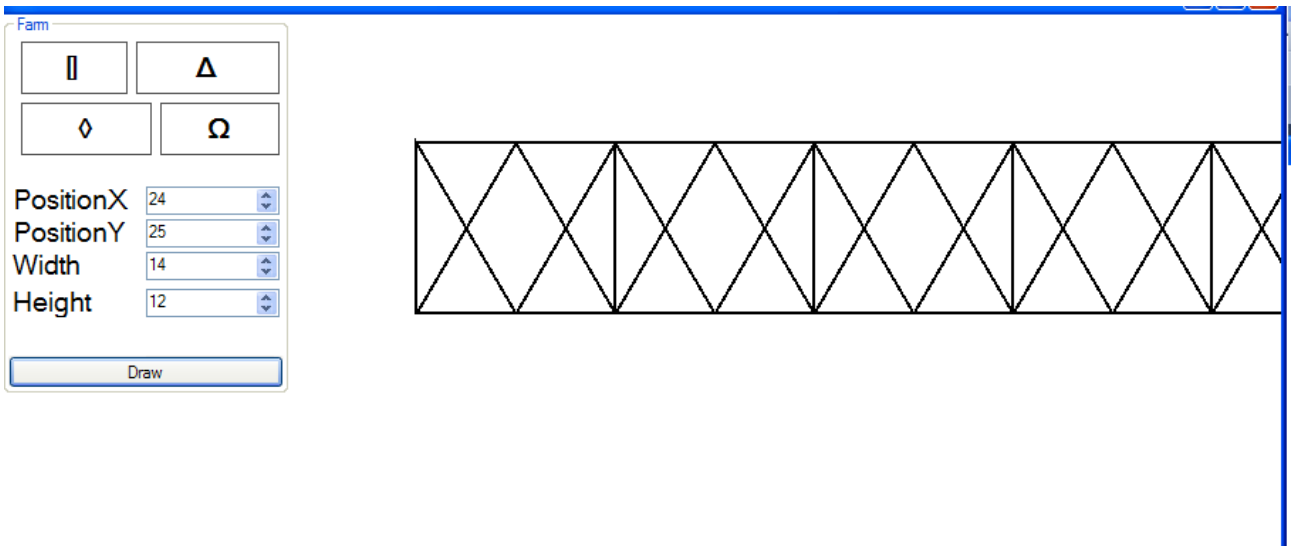


15b-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan ferma chizmasi



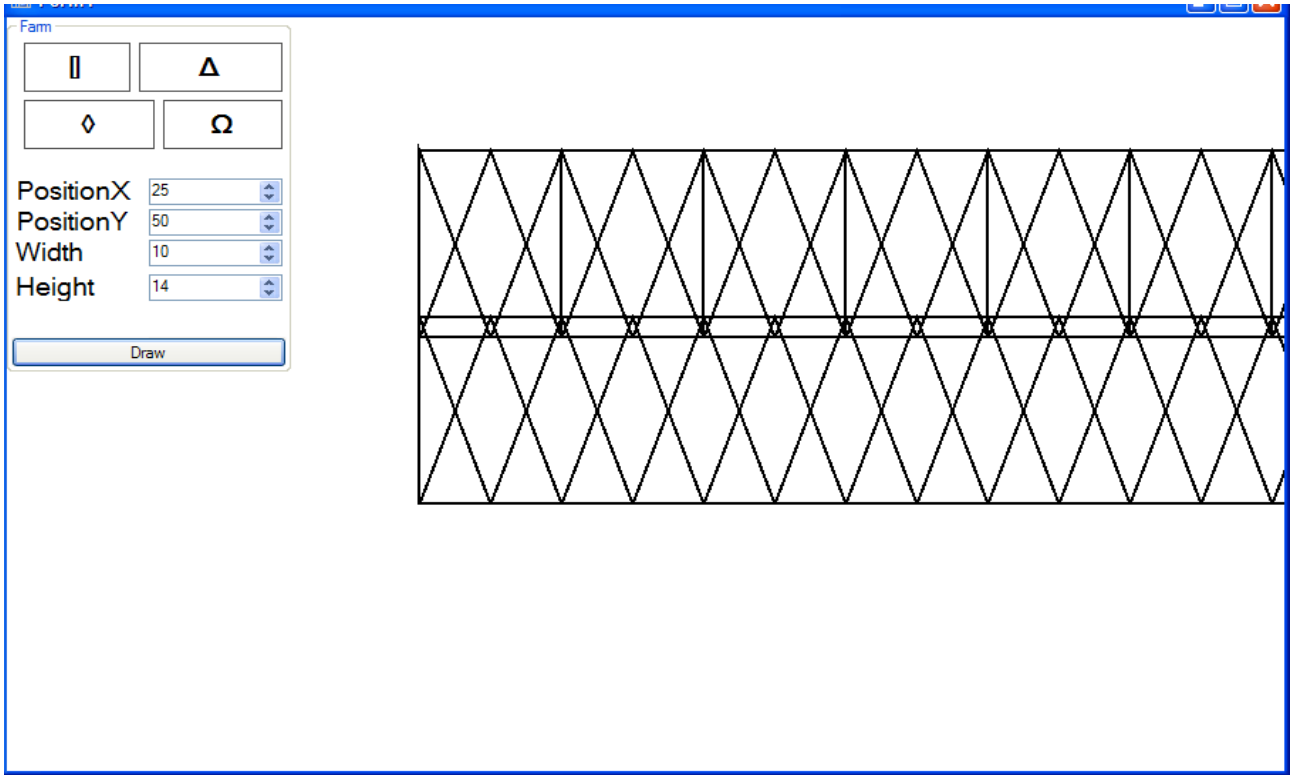
13v-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan ferma chizmasi

.

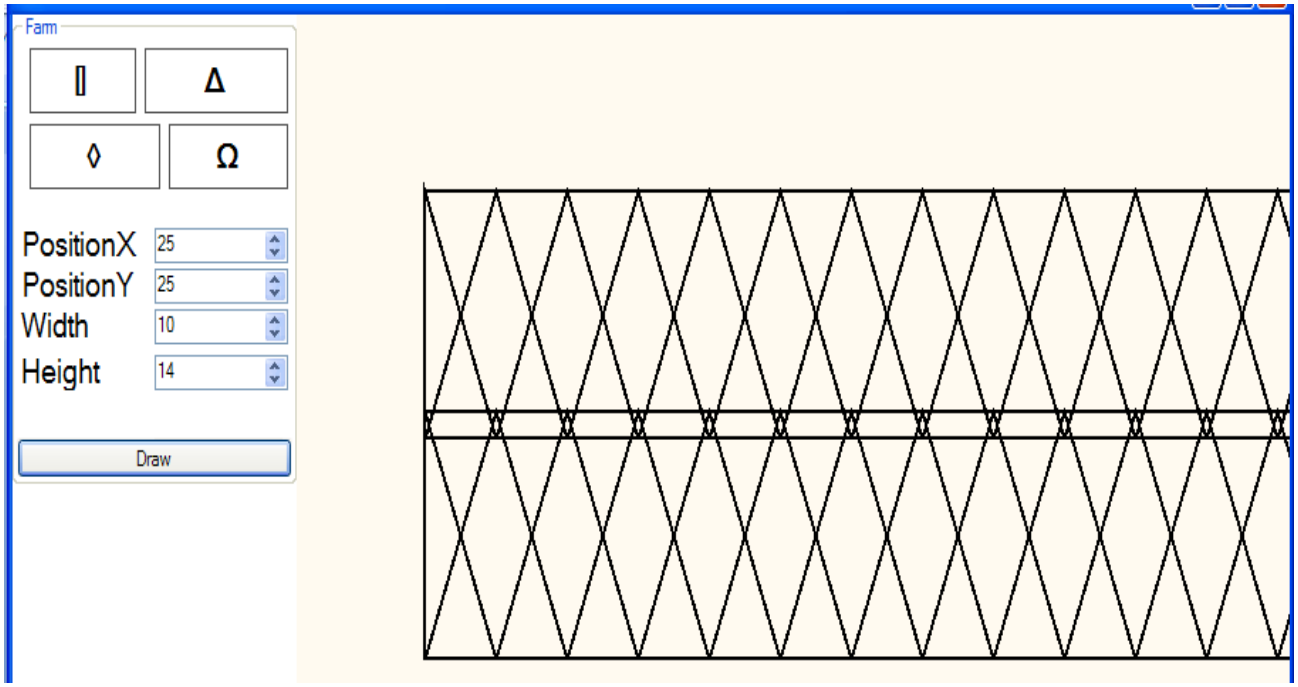


15g-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan ferma chizmasi

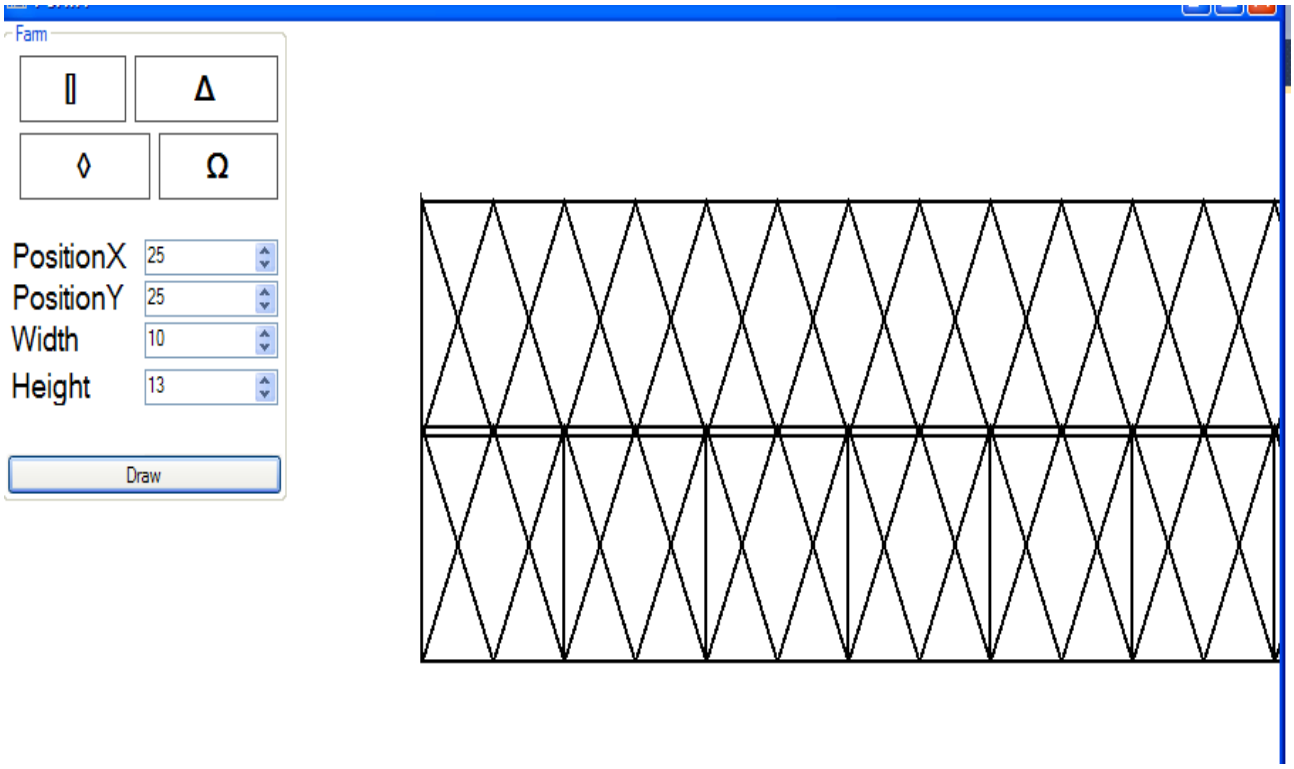
Endi biz qurilish konstruksiyalarini yanada murakkablashtirib boramiz, xususan fermalarni loyixalashda ikkita uchburchaklarni asoslari bo'yicha birlashtirish natijasida fermalarni konstruksiyalash masalasini birlashtirirish masalasini ko'ramiz(13-rasm).Bu holda biz uchburchakli elementlarni birlashtirib yanada murakkab konstruksiyalar hosil qilishimiz mumkin. Dasurda bunday masalalarni yechish ham amalga oshirilishi mumkin



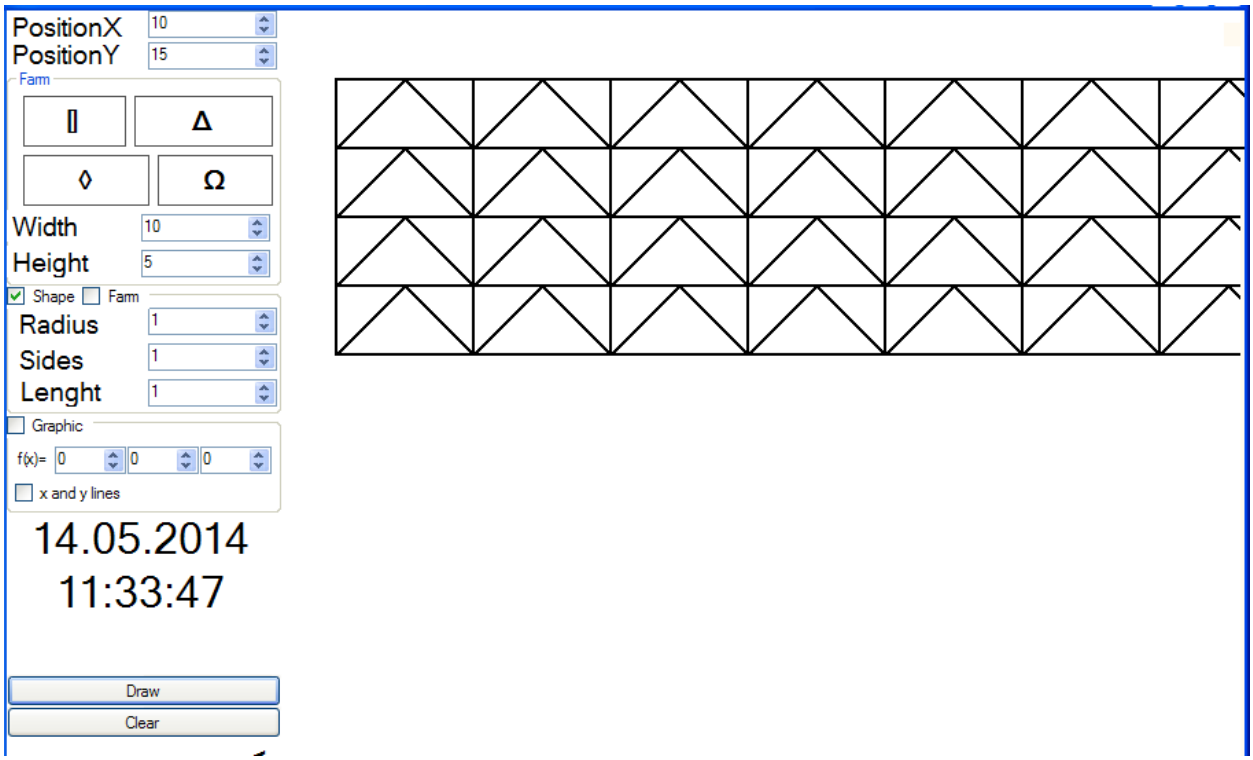
16a-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan fermalarni birlashtirish chizmasi (To'rtburchakli payvandlash)

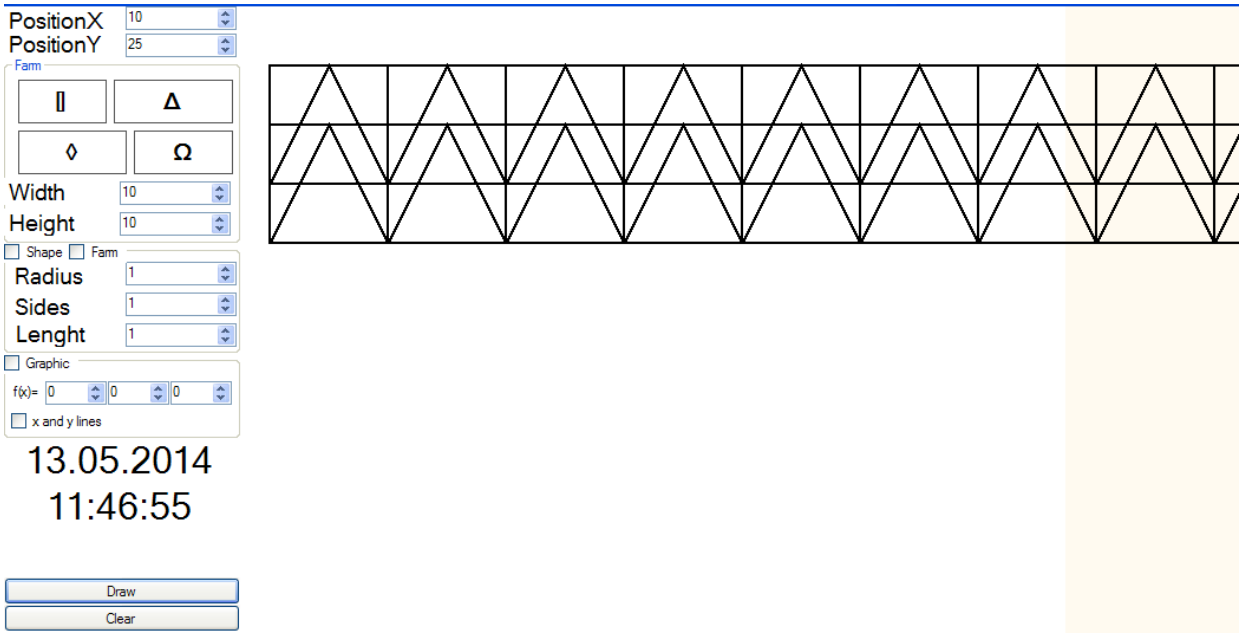


16b-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan fermalarni birlashtirish chizmasi (To’rtburchakli payvandlash)

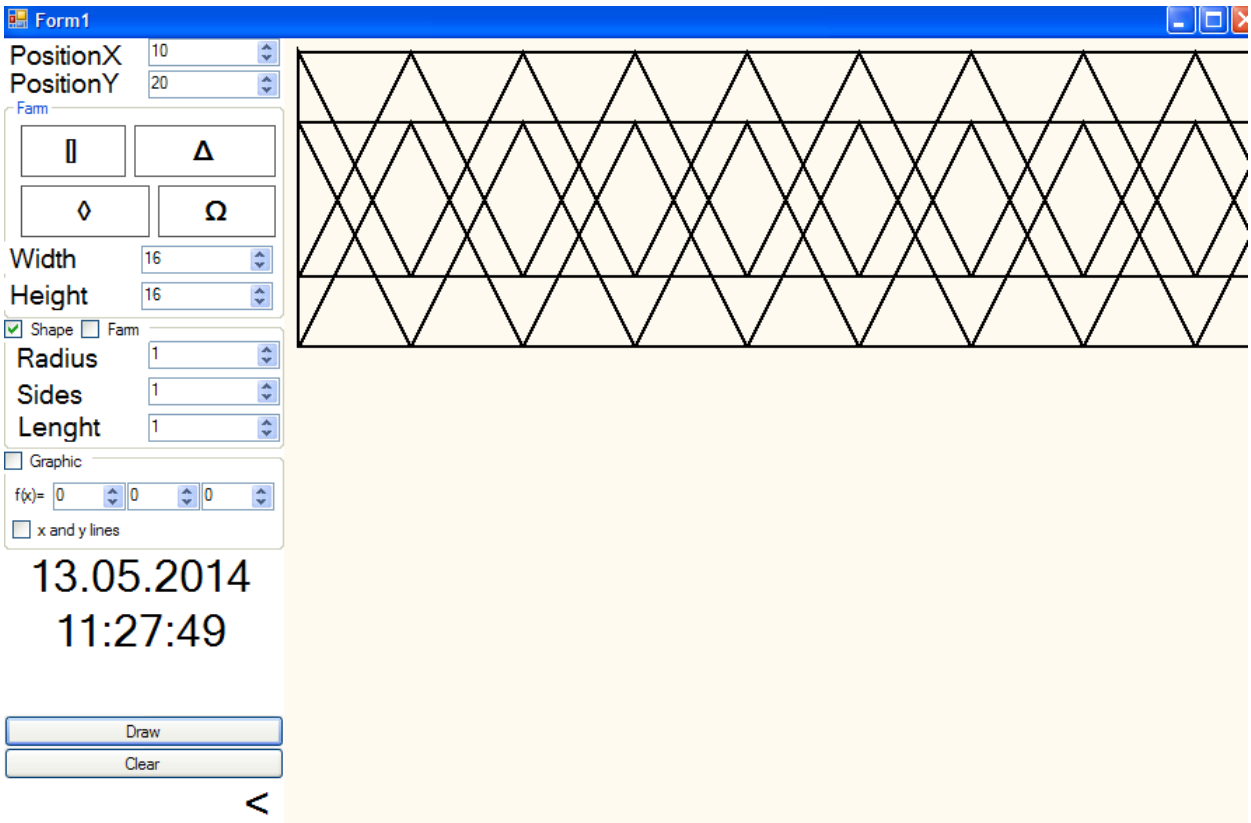


16v-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan fermalarni birlashtirish chizmasi (Kontakt payvandlash)

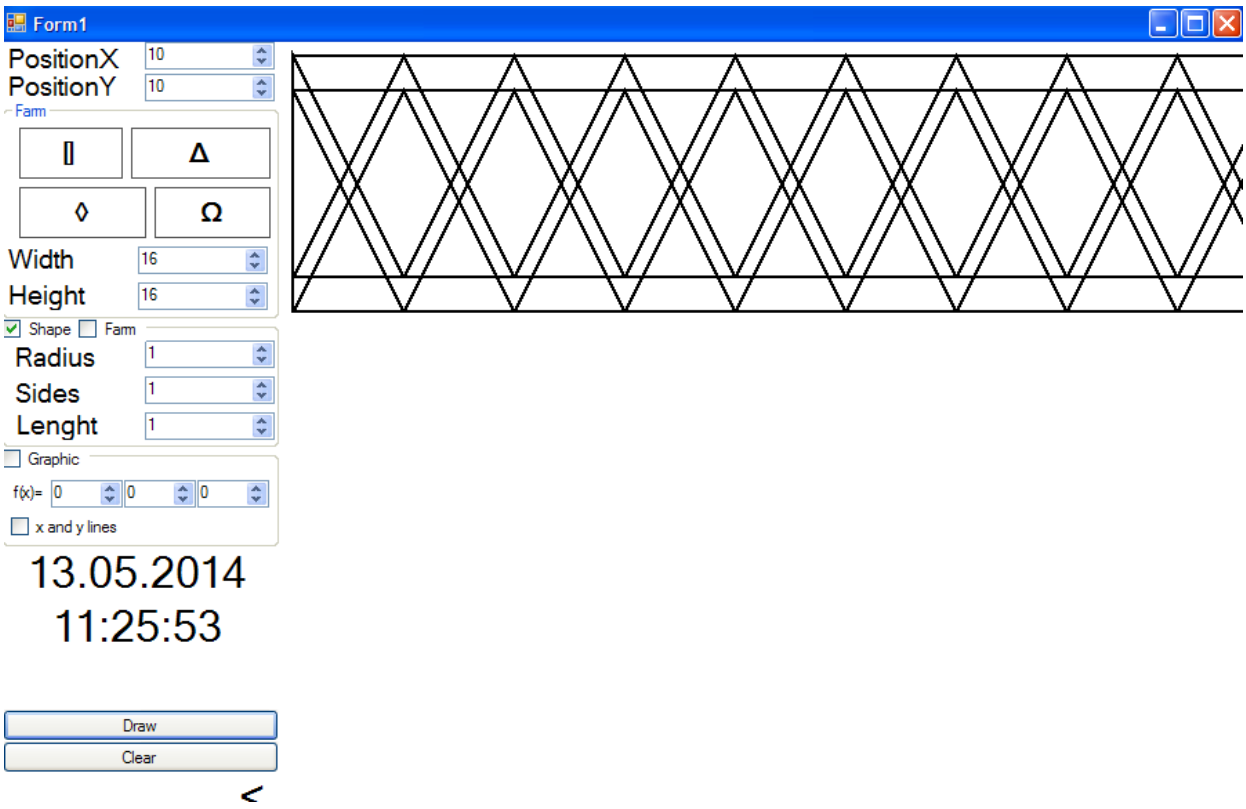




17a-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan fermalarni birlashtirish chizmasi

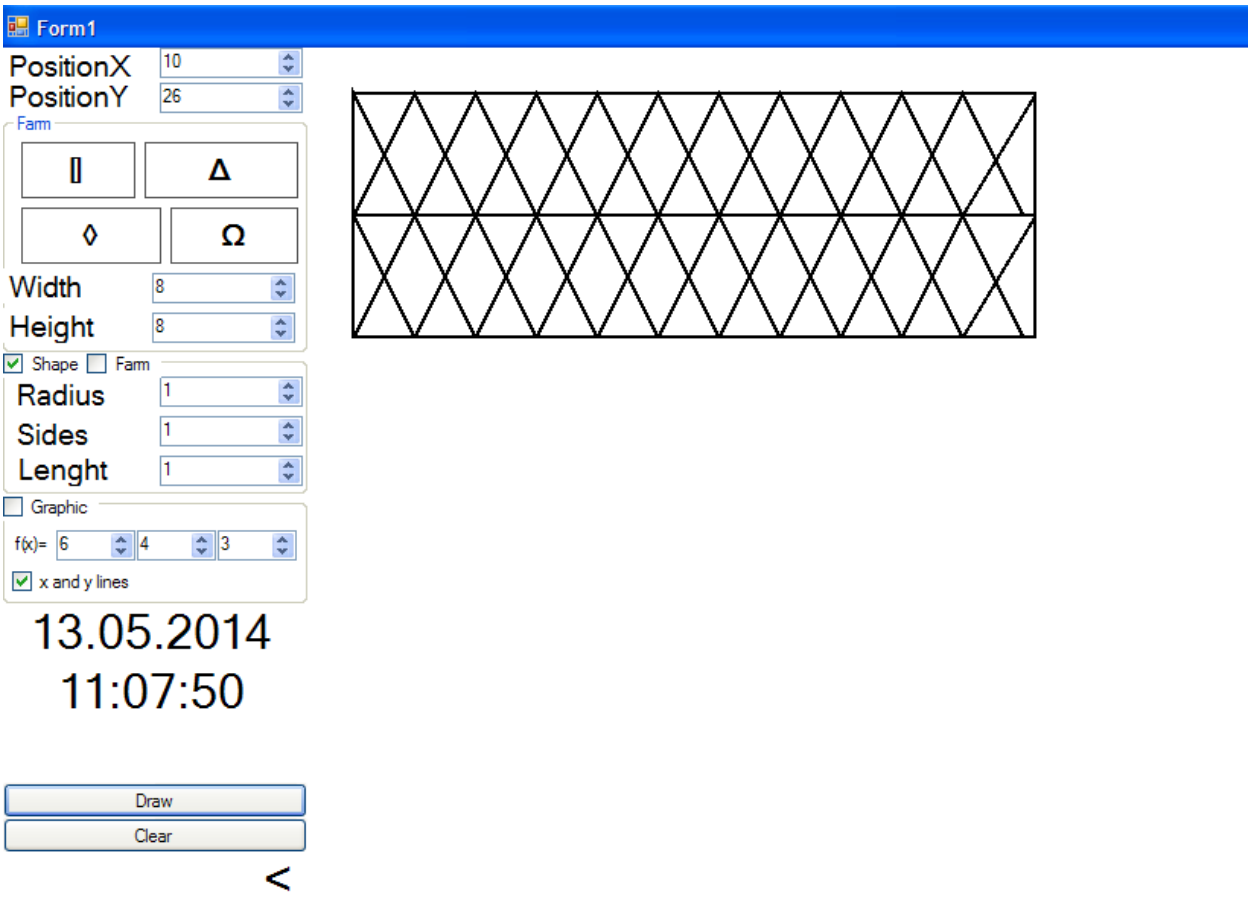


17b-rasm. Ikkita uchburchakli ellementlardan шллшдфтпфт konstruksiyalangan fermalarni birlashtirish chizmasi

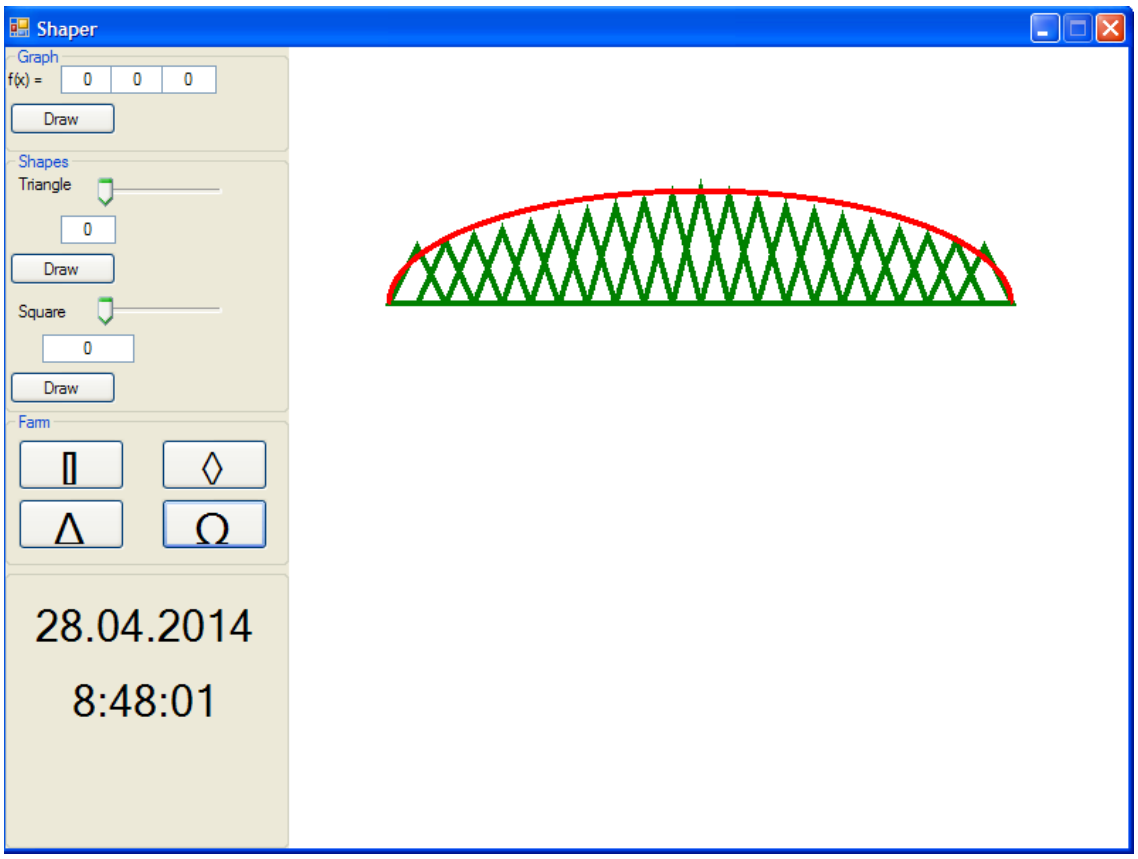


17v-rasm. Ikkita uchburchakli ellementlardan konstruksiyalangan fermalarni birlashtirish chizmasi

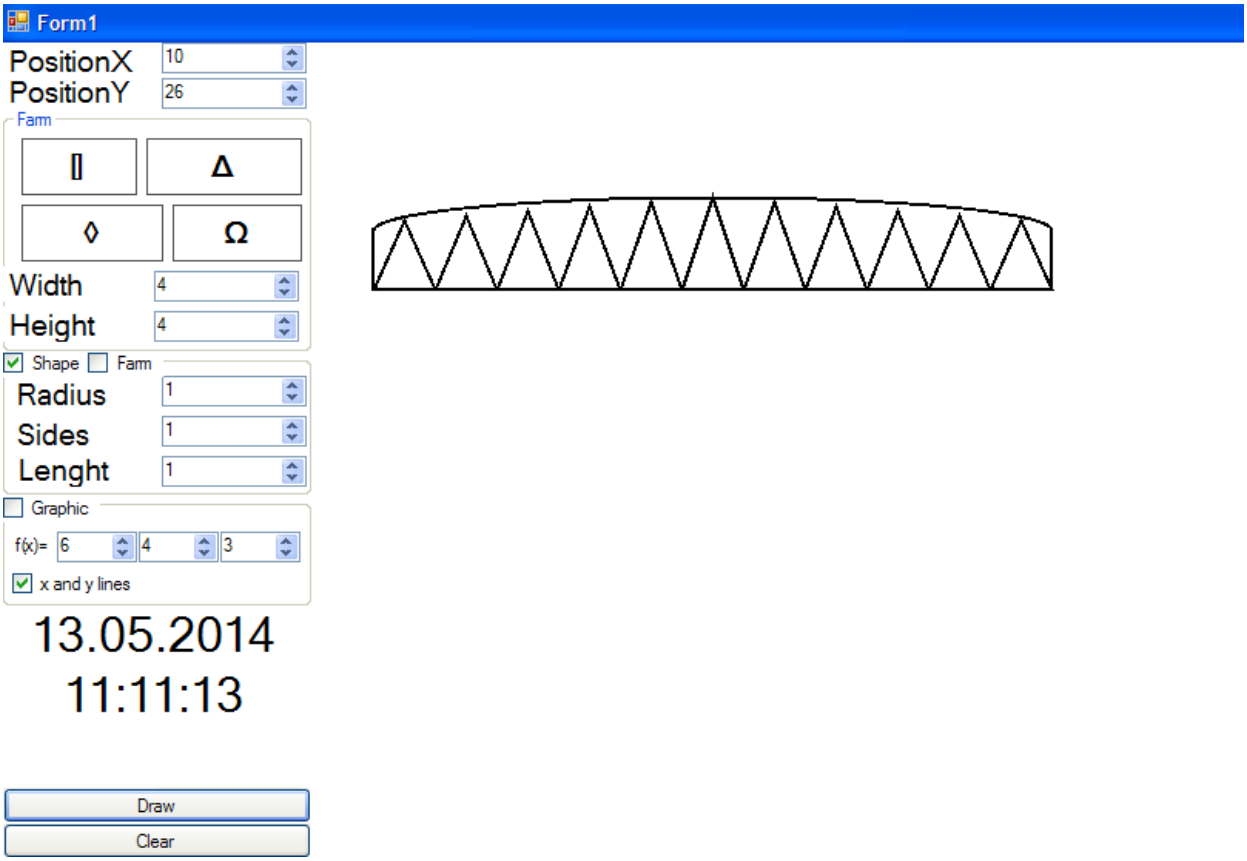
Yuqorida keltirilgan masalalarda ko’proq to’g’ri chiziq , uchburchak va to’rtburchakli elementlar ishlatildi va shunga yarash ba’zaviy klasslar ishlatildi. Ammo qurilish konstruksiyalarida ko’pincha muurakkab , hatto nogeometrik konstruksiyalar ham ishlatiladi. Quyida biz yuqori qismi yoysimon qoplamali ferma konstuksiyasi masalasini yechamiz va chixamiz. Konstruksiyaning yoysimon qismi a’lohida algaritm asosida malga oshirildi. Uchburchakli qismlarrida esa oldingi ishlatilgan konstruksiyalar elementlaridan foydalanildi. Bu masalalardagi algoritmlarda analitik geometriya usullari ham ishlatildi(14-rasm).



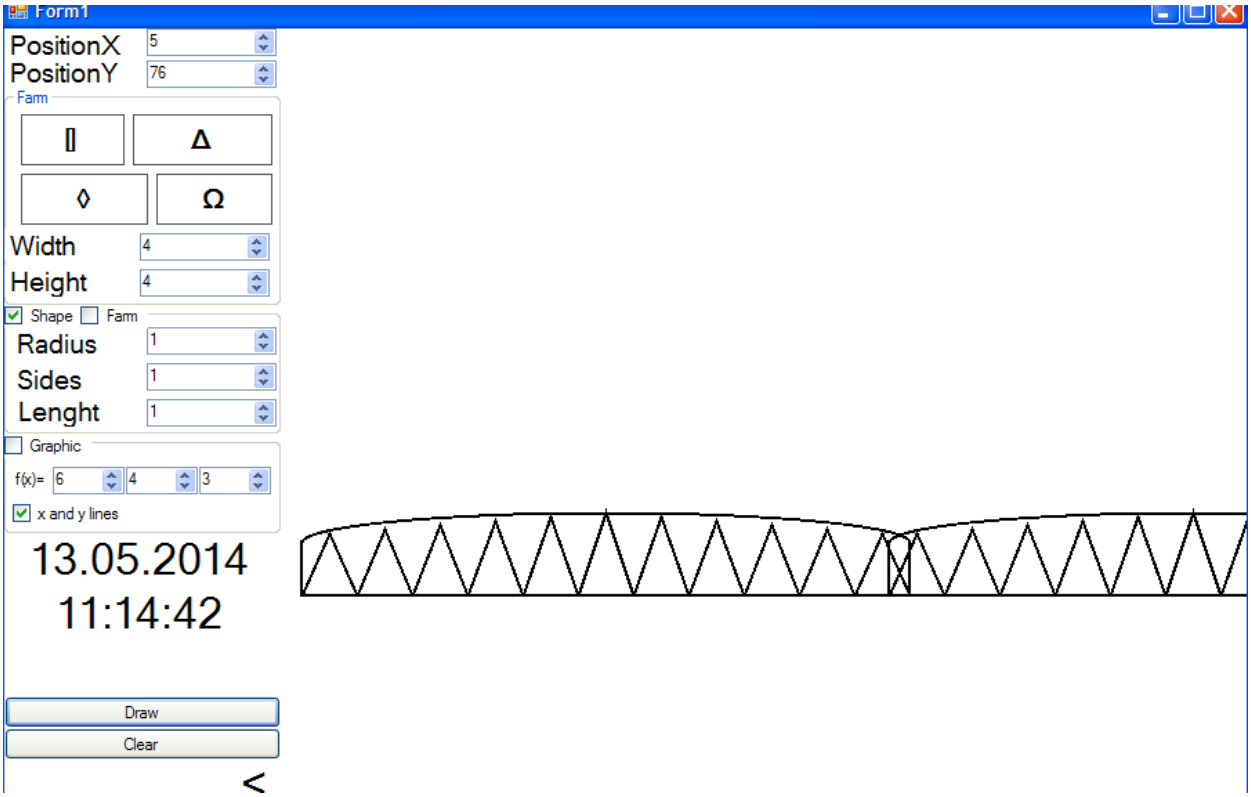
18-rasm. Ikkilangan uchburchakli ferma konstruksiyasi



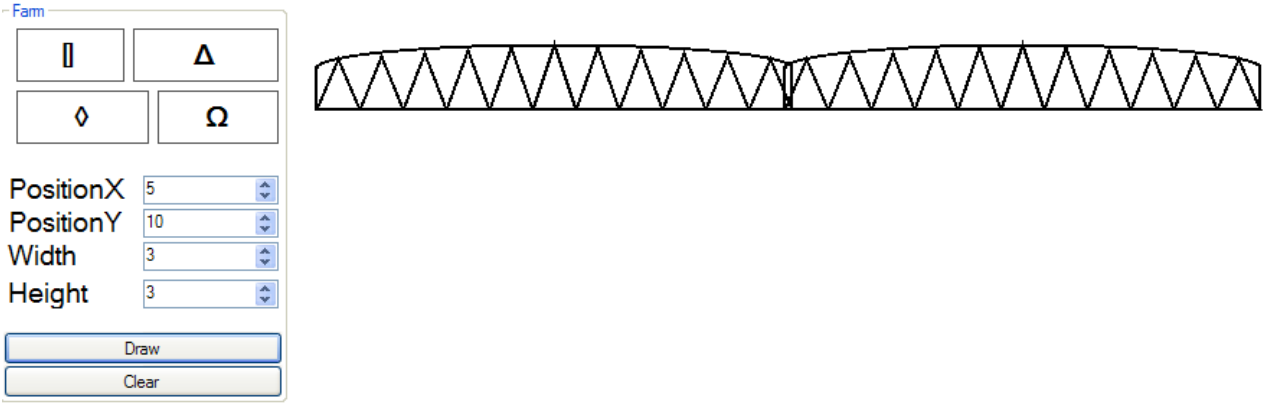
19a-rasm. N\Murakkab fermalar yaratish



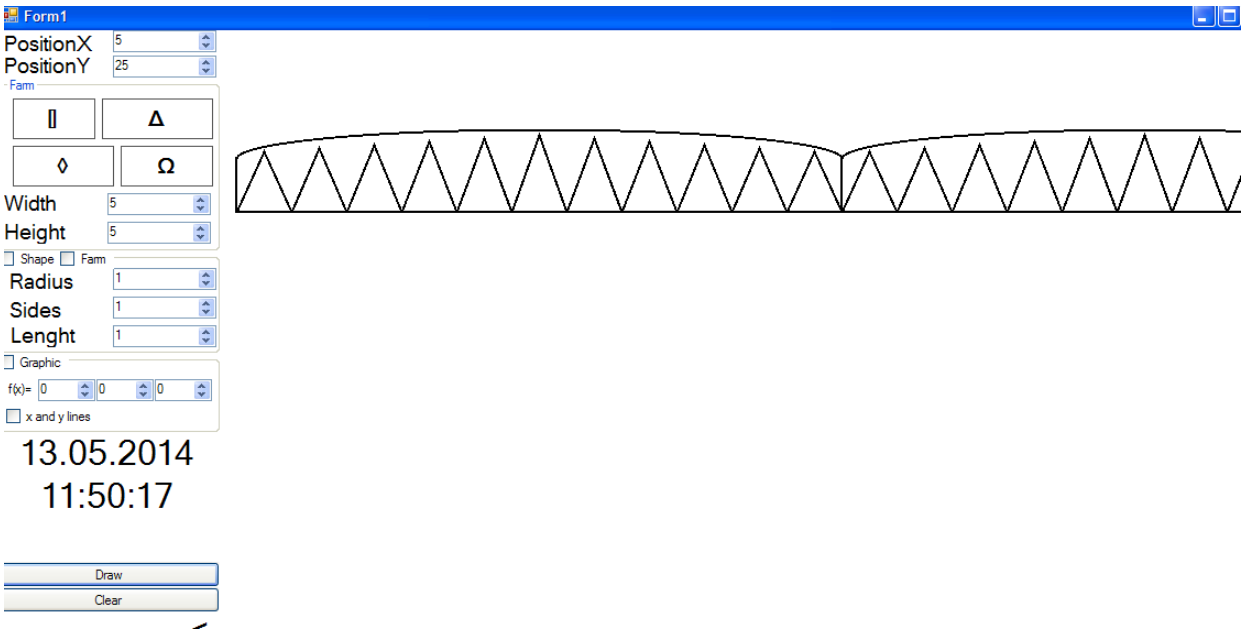
19b-rasm. Yuqori qismi yoysimon elementlar bilan konstruksiyalangan fermalarni yasash va chizish.



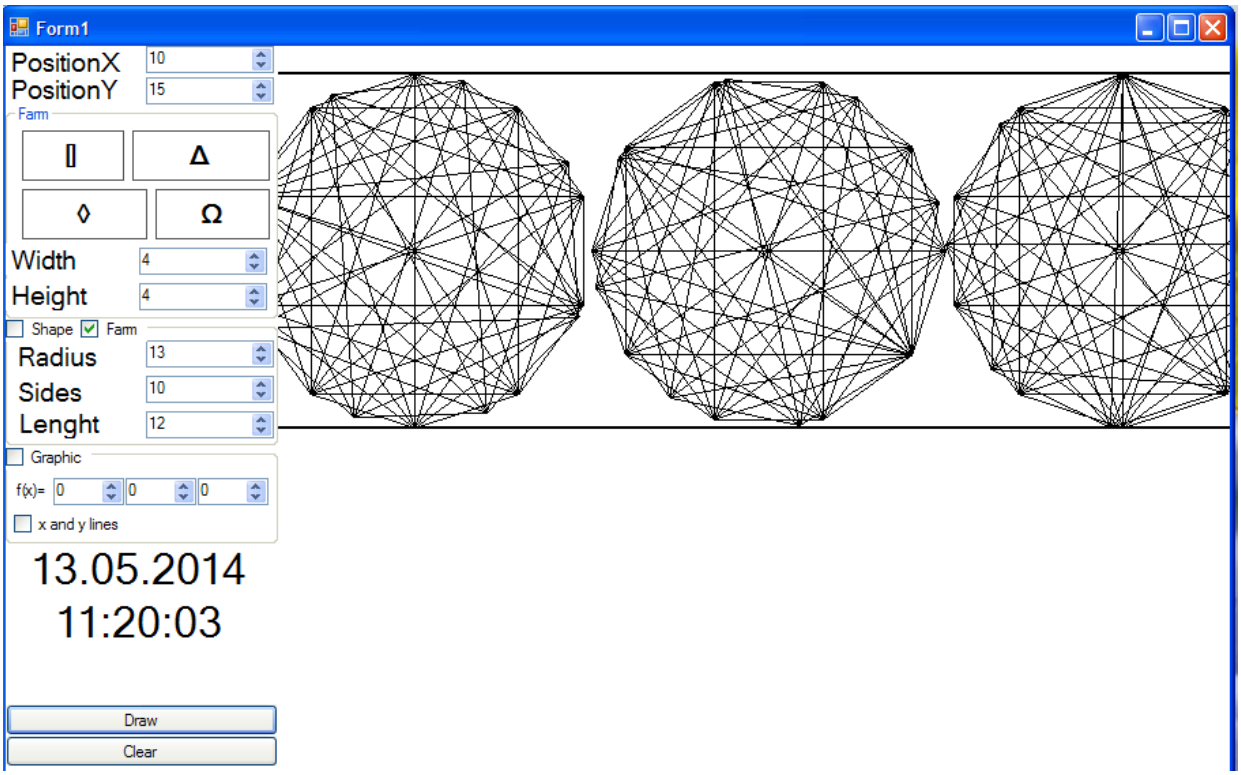
19v-rasm. Yuqori qismi yoysimon elementlar bilan konstruksiyalangan fermalarni yasash va chizish.



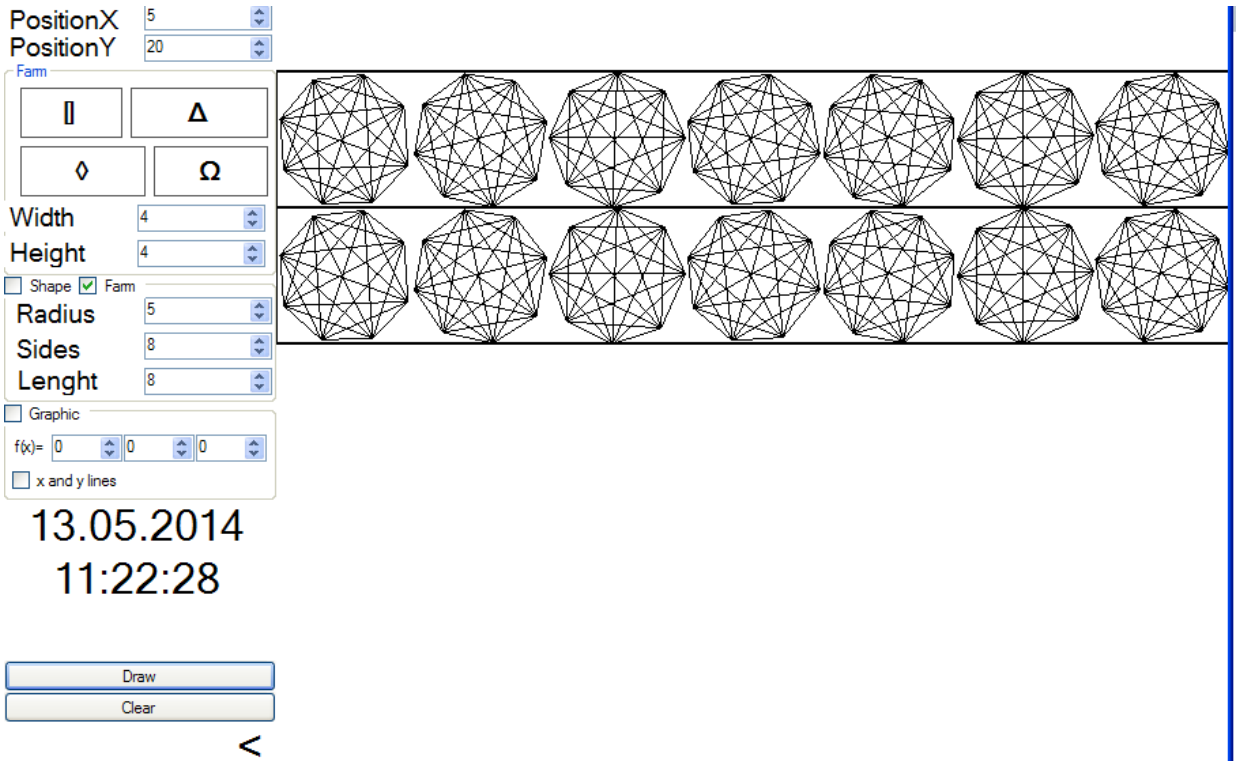
19g-rasm. Yuqori qismi yoysimon elementlar bilan konstruksiyalangan fermalarni yasash va chizish.



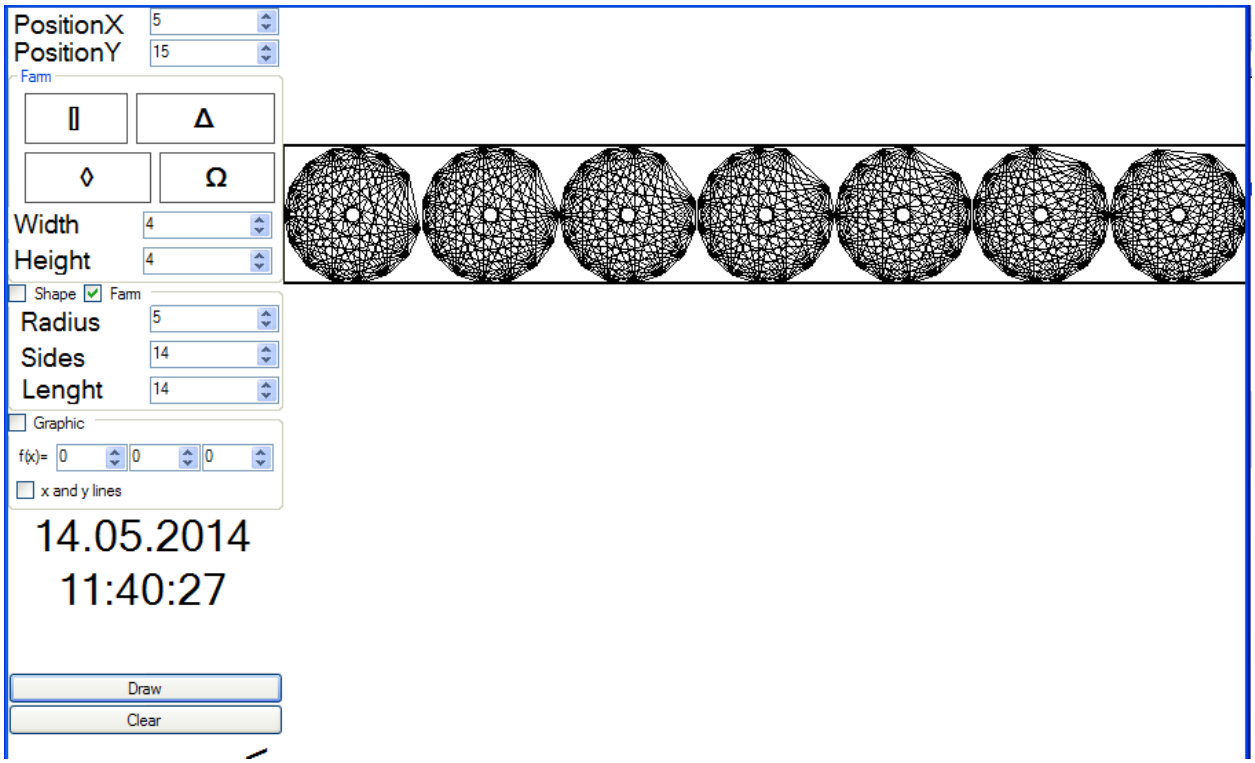
19d-rasm. Yuqori qismi yoysimon elementlar bilan konstruksiyalangan fermalarni yasash va chizish.



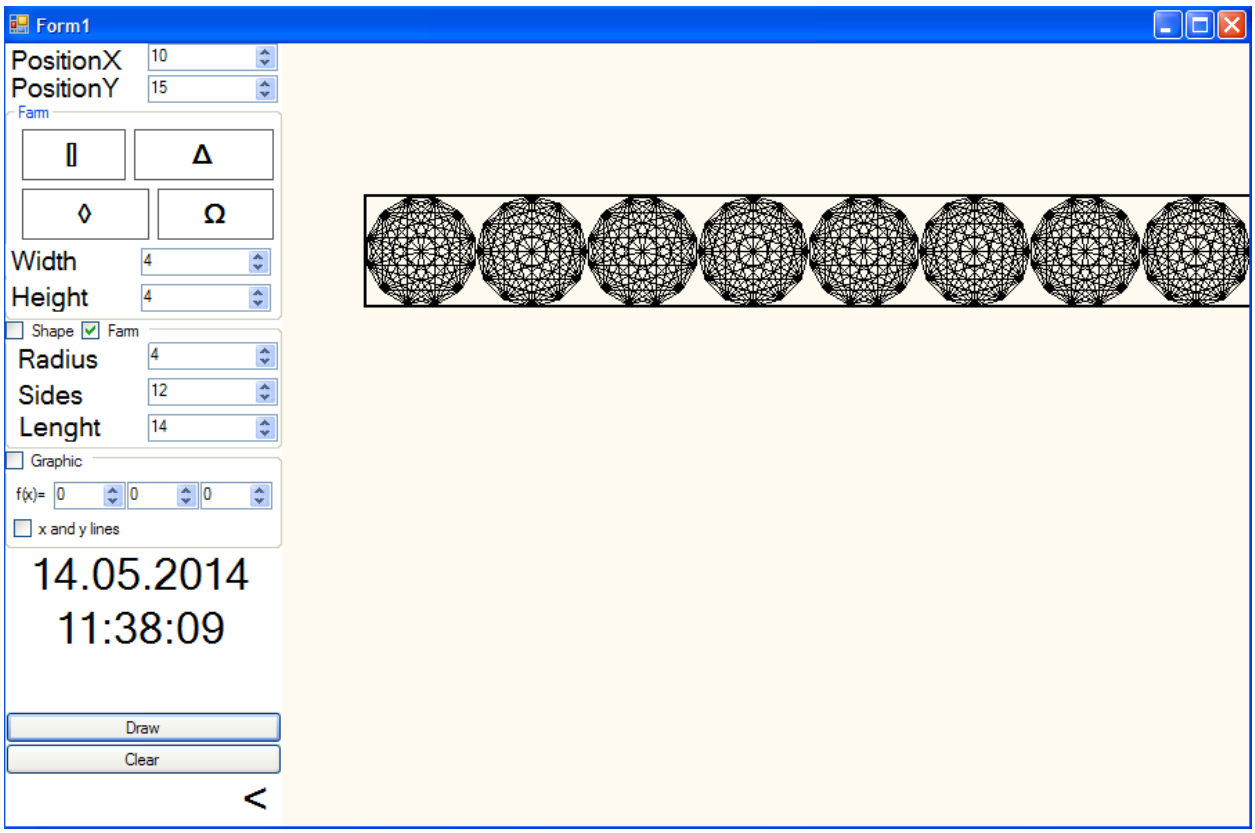
20a-rasm. Ko’pburchakli elementlardan foydalanib fermalarni konstruksiyalash



16a -rasm. Qavariq ko’pburchakli elementlardan konstruksiyalar yaratish



20b-rasm. Ko’pburchakli elementlardan foydalanib fermalarni konstruksiyalash


Form1
PositionX
10
PositionY
15
Farm
Width
4
Height
4
Shape
Farm
Radius
4
Sides
12
Lenght
14
Graphic
f(x)=
0
0
0
x and y lines
14.05.2014
11:38:09
Draw
Clear

XULOSA

Hozirgi vaqtga kelib ob'ektga mo'jallangan tizimlar daromadning asosiy qismini ikki gurux mahsulotlar hisobiga to'gri kelmoqda. Ulardan birinchi guruhi – sun'iy intellekt tizimining gibrid instrumentlariga, ikkinchisi esa – tillar va dasturlash vositalariga. Qo'shimcha daromadlar ob'ektga mo'jallangan ma'lumotlar bazalari hisobiga olinar edi. Bu mahsulotlar uchta turdagi kompaniyalar tomonidan yetkazilib berilar edi: sun'iy intellekt tizimlarini sotuvchilar tomonidan, apparat vositalarini ishlab chikaruvchilar tomonidan va yosh kompaniyalar tomonidan.

Bu mahsulotlar hir xil kompaniyalar tomonidan ilmiy va amaliy sohalarda ishlatilmoqda. Bu mahsulotlarni yetkazib beruvchilar, shu jumladan apparatura ishlab chiqaruvchilar va DT yaratuvchilar ham foydalanuvchilar hisoblanadi. Faol foydalanuvchilarga quyidagilar kiradi: telekommunikatsion kompaniyalari, harbiy va aerokosmik firmalari, ishlab chiqarish va texnik kompaniyalar. Hozirgi vaqtda ob'ektga mo'jallangan maxsulotlarning kommertsiyali ishlatilishi katta korporatsiyalarni boshqarishda ayniqsa qo'l kelmoqda.

Ob'ektga mo'jallangan maxsulotlarning qo'llanish sohalari hozirgi vatda quyidagilarni o'z ichiga oladi: foydalanuwchi interfeyslarini ishlab chiqish, dasturlash texnologiyasi, CAD/CAM, ekspert tizimlari va multimedia informatsion tizimlari.

1995 yilga kelib ob'ektga mo'jallangan texnologiya ko'plab yangi mahsulotlari bilan bozorga chiqib keldi. Mavjud mahsulotlar bilan birgalikda endi ular quyidagilardan tashkil topadi:

- redaktorlar, otladchiklar, ko'rib chiqish oynalari va boshqa dasturiy mahsulotlar. Ular ko'p oynali integrallashgan muhit qismlarini tashkil etadi;
- klasslar kutubxonasi va amaliy masalalar generatori. Hozir ulardan ko'pchiligi foydalanuvchi interfeyslarining asosini tashkil etadi;
- ob'ektga mo'jallangan tizimlarni tahlil qilish va yaratish uchun dasturlash texnologisi vositalari (CASE) :
- ob'ektga mo'jallangan ma'lumotlar bazalarini boshqarish tizimlari;

- foydalanuvchilar uchun amaliy instrumentlar.

Mahsulotlar diapazoninig kengayshi, xususan mawjud texnologiyalar bilan to'liq mos keluvchi ob'ektga mo'jallangan ma'lumotlar bazalarining paydo bo'lishi, kommertsiya amaliy masalalari bozorini ochdi.

Magistrlik dissertatsiyasida quyidagi ishlar qilindi va natijalar olindi:

- ob'ektga mo'ljallangan axborot tizimlarining hozirgi holati tahlil qilindi;
- ob'ektga mo'ljallangan axborot tizimlari va ularning asosiy tushunchalari
- o'rganildi;
- OMT larning muammolari va ularni yechish yo'llari o'rganildi va tahlil qilindi;
- ob'ektga mo'jallangan tizimlar bozori tahlil qilindi;
- ob'ektga mo'jallangan tizimlar va maxsulotlar taxlil qilindi;
- predmetli –orientirli tillarning modeli va realizatsiya qilish usullari taxlil qlindi.
- predmetlii –orientirli yondoshuwning qo'llanilish metodikasi ishlab chiqildi;
- geometrik qurishlar maslasini yechishni avtomatik tekshirish tizimi yaratildi va dasurlandi;
- geometrik qurishlar maslasini yechish dasurida qator natijalar olindi.

Biz yuqorida keltrgan predmetli orientirli dasturalsh texnologiyasi yordamida qurilish konstruksiyalarini yaratish tizim dasurini yaratishga harakat qildik:

- Predmetli –orientirli yondoshuwning qo'llanilishi modeli quyidagicha. Dasturiy ta'minot o'zining hayotiy siklining har bir bosqichida ob'ektiv formada biron bir tilda ifodalanadi. Shuningdek avvalo u predmet sohasida biron bir talablar shaklida tabiiy tilda ifodlanadi. Oxirida esa dasturiy ta'minot aniq bir dasturiy kodda ifodlanadi (bu formal til). Bu modelda loyixalash jarayoni – bu bir tildan ikkinchisiga ketma ket o'tkazish (har bir bosqichda ancha abstraktlikdan ancha konkretlikga o'tkazilish), shuningdek

boshlang’ich stadiyalarda bu o’tkazilishlarni insonning o’zi bajaradi (tabiiy tildan yuqori pog’onali formal tilga o’tkazish), keyingi bosqichlarida esa ma’lum bir avtomatik o’tkazilishlar ishlatiladi (Predvetli –orientirli yondoshuv tili translyatori). Ob’ektga mo’jallangan tizimlar yaratishda agar tilning elementlari soni kam bo’lsa, u holda uni tavsiflavchi konstruktsiyalar juda noqulay kontruktsiyalar bilan tavsiflanadi.

Biz geometrik figuralarni ekranga chizishning obektga mo’ljallangan dasturini yozamiz. Ekranda chizish uchta qismdan iborat bo’ladi:

- ekran monittori: ekran bilan ishlash uchun kerak bo’lgan funksiyalar va axboratlar tarkiblari, bu yerda faqat nuqta , chiziqlar bilan operatsiya qilinadi;
- figuralar kutubxonasi: umumiy ko’rinishdagi figuralarni aniqlash to’plamlari (masalan, to’rtburchak, aylana…) va ular bilan ishlovchi standart funksiyalar;
- Amaliy dastur: masalaga tegishli figuralarni konkret aniqlash hamda ular bilan ishlash funksiyalarini aniqlash.

Birinchi navbattda tizimli dasturga asos bo’luvchi bazaviy klasslar yaratildi.
To’g’ri chiziq chizish, egri chiq chizish, uchbirchak , to’rtburchak va ko’p burchaklarni chizish klasslarini yaratish amalga oshirildi. Bu bazaviy klasslar nogeometrik figiralarni chsh universal klasslarini ham yaratishga olib keldi. Bu esa yechimalarni yaratishdagi intellektual yechish usullaridan foydalanishga olib keldi. Jumladan tarmoq baholash kriteriyali intellektual algoritmlardan foydalanishni talab qildi.

Geometrik masalalarni tilda shakillantirish uchun geometrik primitivlar va ular orasidagi munosobatlar kerak bo’ladi. Analitik geometriya nuqtai nazaridan hamma geometrik ob’ektlar bitta yoki bir nechta tenglamalar bilan beriladi. Konkret ob’ekt uning turi bilan aniqlanadi (masalan, nuqta, to’g’ri chiziq, kesma, nur) va mos turlarga tegishli koeffitsientlar, mos ob’ektlar tenglamalari bilan aniqlanadi.

Masalani yechishni avtomatik tekshirish uchun masala sharti ob’ektga mo’jallangan tilda va tabiy tilda (o’quvchilar ob’ektga mo’jallangan tilni bilmasliklari mumkin)yozilishi lozim.

Bazaviy klasslardan foydalanib qurilish konstruksiyalarining bir turi ferma konstruksiyalarini yaratis intellektual algoritmi va dasturi yaratildi. Dissertatsiya ishida yechilgan har bir ferma o’z navbatida element sifatida qaralib murakkab konstruksiyalar yaratilishi mumkin. Bunday masalalarga bir nechta misollar keltiridi va ularning konstruksiyalari chizildi.

Biz aslida nazariy mexanika masalalari konstruksiyalash va qurish bilan shug’ullangan bo’lsakda aslida chizma qirralariga yuklar qo’yib ularning egilish va bukilish maslalarini ham yechishimiz mumkin. Dissertatsiya ishining kelejakdagi ishlari ham ana shu ishlarni tashkil etadi.

Foydalanilgan adabiyotlar ro’yxati

1. Ўзбекистон Республикаси президентининг «Информацион – коммуникацион технологияларини компьютерлаштириш ва амалиётга тадбиқ қилишни янада ривожлантириш» хақидаги 30 май, 2002 йилдаги фармони.
2. Ўзбекистон Республикаси вазирлар махкамасининг Информацион – коммуникацион технологияларини компьютерлаштириш ва амалиётга тадбиқ қилишни янада ривожлантириш» хақидаги 6 июнь, 2002 йилдаги қарори.
3. «Информацион –коммуникацион технологияларини компьютерлаштириш ва амалиётга тадбиқ қилишни 2002-2010 йилларда ривожлантирилиши» хакидаги Ўзбекистоннинг Давлат дастури.
4. Президент Ислом Каримовнинг 2009 йилнинг асосий якунлари ва 2010 йилда Ўзбекистонни ижтимоий-иктисодий ривожлантиришнинг энг мухим устивор йўналишларига багишланган Вазирлар Махкамасидаги маърузаси.
5. Каримов И.А. Юксак маънавият-енгилмас куч. -Т.: „Маънавият”, 2008. 110-112 – betlar.
6. Каримов И.А. Жахон молиявий –иктисодий инкирози, Ўзбекистон шароитида уни бартараф этишнинг йўллари ва чоралари. Т.: “Ўзбекистон”, 2009.
7. Назиров Ш.А. «Программирование С++» Методическое пособие для ИКТ. Ташкентский профессиональнқй коллеж информационных технологий, Т.: 2006.
8. Дэвис, Стефан, Р. Д94 С++ для "чайников", 4-е издание.: Пер. с англ. : — М. : Издательский дом"Вильяме", 2003. — 336 с.

9. Сабуров С.В. Языки программирования С и С++. _ М.: Бук_пресс, 2006. _ 647 с. (Справочное руководство пользователя персонального компьютера).
10. Хаймен, Майкл, Арнсон, Боб. X15 Visual C++.NET для "чайников". : Пер. с англ. — М. : Издательский дом "Вильяме". 2002. — 288 с.
11. Оберг, Роберт, Дж., Торстейнсон, Питер. OI3 Архитектура .NET и программирование с помощью Visual C++.: Пер. с англ. — М.: Издательский дом "Вильяме", 2002. — 656 с.:
12. Назиров Ш.А идругие . Языки программирования С++, ТУИТ, 2006 г.
13. Биллиг В.А.Основы программирования на C# .Интернет-университет информационных технологий – Intuit.ru, 2011
14. Фридман А.Л. Язык программирования Си++ . Интернет-университет информационных технологий - Intuit.ru, 2011
15. Костюкова Н.И., Калинина Н.А.Язык Си и особенности работы с ним Интернет-университет информационных технологий - Intuit.ru, 2011
16. Mike Wilson and al. Using objects to design and build radar ESM systems. OOPSLA 87 Conference Proceedings. ACM SIGPLAN Notices, volume 22, number 12, December 1987.
17. John K. Bennett The desigh and implementation of Distributer Smalltalk. OOPSLA 87 Conference Proceedings. ACM SIGPLAN Noties, volume 22, number 12, December 1987
18. Alan Borning The programmihg language aspects of ThingLab, a constraint-oriented simulation laboratory. ACM Transactions on Programmihg Languages and Systems, Vol. 3, number 4, October 1981
19. Alan Borning and Tim O'Shea Deltatalk: an empirically and aesthetically motivated simplification of the Smalltalk-80 language. ECOOP'87 Proceedings. Lecture Notes in Computer Science, number 276, Springer-Verlad 1987
20. Frederik P. Brooks, Jr. No silver bullet: essence and accidentes of software engineerihg. Computer, April 1987

21. Michael Caplinger An information system based on distributed objects. OOPSLA 87 Conference Proceedings. ACM SIGPLAN Notices, volume 22, number 12, December 1987
22. D. N. Card and W. W. Agresti Measurihg software design complexity. Journal of Systems and Software, June 1988
23. Luca Cardelli, James Donahue, Lucille Glassman, Mick Jordan, Bill Kalsow and Greg Nelson Modula-3 report. Digital Equipment Corp, SRC report 31, 1988
24. Brad J. Cox Object-oriented programming: an evolutionary approach. Addison-Wesley, 1986
25. Linda G. DeMichiel and Richard P. Gabriel The Common Lips Object System: an overview. ECOOP'87 Proceedings. Lecture Notes in Computer Science, number 276, Springer-Verlad 1987
26. Smalltalk/V tutorial and programming hanbook. Digitalk, 1986
27. http://www.ProgC++.com/
28. Объектно-ориентированное программное обеспечение. www.nnz-ipc.ru/
29. Объектно-ориентированная разработка программ ds.com.ua/win/rus/program/

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace FarmTest
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      #region переменные

      int positioX = 0, positioY, fwidth = 0, fheight = 0;
      public int tempx=0, tempy=0;
      public int pheight = 0, pwidth = 0;
      public double[] px = new double[8000],py = new double[8000];
      Boolean rectfarm, triafarm, diamfarm, circfarm;
      #endregion
      #region class
      class DrawFarm : Form1
      {
         Pen p = new Pen(Color.Black, 2);
```

```
        #region Farms
        public void rectangleFarm(Graphics g, int positX, int positY, int width1, int
height1)
        {
            pwidth = Convert.ToInt16(DrawBar.Width);
            int l = 0;
            int centerx;
            int centery;
            centery = positX * 5;
            centerx = positY * 5;
            int startx = (centerx - (width1 / 2) * 5),
                starty = (centery - (height1 / 2) * 5),
                endx = (centerx + (width1 / 2) * 5),
                endy = (centery + (height1 / 2) * 5);
            for (int j = startx; j <= endx; j += width1/2)
            {
                g.DrawRectangle(p, starty + (l * 10), startx, width1 * 10, height1 *
10);
                l += width1;
                g.DrawRectangle(p, starty, startx, width1 * 10, height1 * 10);
            }
            tempx = positX;
            tempy = positY;
        }
        public void triangleFarm(Graphics g, int positX, int positY, int width1, int
height1)
        {
            int l = 0;
            pheight = Convert.ToInt16(DrawBar.Height);
            pwidth = Convert.ToInt16(DrawBar.Width);
```

```
            int centerx;
            int centery;
            centery = positX * 5;
            centerx = positY * 5;
            int x = 0;
            int y = 0;
            int startx = (centerx - (width1 / 2) * 5),
               starty = (centery - (height1 / 2) * 5),
               endx = (centerx + (width1 / 2) * 5),
               endy = (centery + (height1 / 2) * 5);
            x = startx;
            y = starty;
            for (int j = startx; j <= endx; j += width1/2)
            {
               Point[] point = { new Point(starty + (l * 10), startx + (height1 * 10)),
new Point(starty+(width1*5)+l*10, startx), new Point((starty + (l *
10))+(width1*10), startx + (height1 * 10)) };
               g.DrawPolygon(p, point);
               g.DrawRectangle(p, starty + (l * 10), startx, width1 * 10, height1 *
10);
               g.DrawRectangle(p, starty, startx, width1 * 10, height1 * 10);
               l += width1;
               tempx = positX;
               tempy = positY;
            }
         }
         public void diamondFarm(Graphics g, int positX, int positY, int width1, int
height1)
         {
            int l = 0;
```

```
            pheight = Convert.ToInt16(DrawBar.Height);
            pwidth = Convert.ToInt16(DrawBar.Width);
            int centerx = 0;
            int centery = 0;
            int temp = 0;
            centery = positX * 5;
            centerx = positY * 5;
           int startx = (centerx - (width1 / 2) * 5),
               starty = (centery - (height1 / 2) * 5),
               endx = (centerx + (width1 / 2) * 5),
               endy = (centery + (height1 / 2) * 5);
            Point[] point1 = { new Point(starty + (l * 10), startx + (height1 * 10)),
new Point(starty + (l * 5), startx), new Point((starty + (l * 10)) + (width1 * 5),
startx + (height1 *10)) };
            g.DrawPolygon(p, point1);
            if (width1 % 4 != 0)
               temp = width1 - width1 % 4;
            else temp = width1;
            for (int j = startx; j < endx; j += temp / 2)
            {
               Point[] point = { new Point(starty + (l * 10), startx + (height1 * 10)),
new Point(starty + (width1 * 5) + l * 10, startx), new Point((starty + (l * 10)) +
(width1 * 10), startx + (height1 * 10)) };
               g.DrawPolygon(p, point);
               l += width1 / 2;
            }
            Point[] point2 = { new Point(starty + (l * 10), startx + (height1 * 10)),
new Point((starty + (l * 10) + (width1 * 5)+width1), startx), new Point((starty + (l
* 10) + (width1 * 5)+width1), startx + (height1 * 10)) };
            g.DrawPolygon(p, point2);
```

```
            g.DrawLine(p,starty,startx,starty+(l*10)+(width1*5)+width1,startx);
            tempx = positX;
            tempy = positY;
        }
        public void circleFarm(Graphics g, int positX, int positY, int width1, int
height1)
        {
            int l = 0;
            pheight = Convert.ToInt16(DrawBar.Height);
            pwidth = Convert.ToInt16(DrawBar.Width);
            int centerx;
            int centery;
            int temp = 5;
            centery = positX * 5;
            centerx = positY * 5;
            int start = 0, end = 0;
            int startx = (centerx - (width1 / 2) * 5),
               starty = (centery - (height1 / 2) * 5),
               endx = (centerx + (width1 / 2) * 5),
               endy = (centery + (height1 / 2) * 5);
            end = 11 * (width1 * 10);
            for (int j = start; j <= end/2 - (width1*10); j += width1*10)
            {
               {
                  Point[] point = { new Point(starty + (l * 10), startx + (height1 *
10)), new Point(starty + (width1 * 5) + l * 10, startx-temp), new Point((starty + (l *
10)) + (width1 * 10), startx + (height1 * 10)) };
                  g.DrawPolygon(p, point);
                  l += width1;
                  temp += (int)(height1*0.75);
```

```
            }
        }
        for (int j = end/2; j <= end; j += width1*10)
        {
            {
                Point[] point = { new Point(starty + (l * 10), startx + (height1 *
10)), new Point(starty + (width1 * 5) + l * 10, startx-temp), new Point((starty + (l *
10)) + (width1 * 10), startx + (height1 * 10)) };
                g.DrawPolygon(p, point);
                l += width1;
                temp -= (int)(height1*0.75);
            }
        }
        g.DrawArc(p, starty, startx-(height1*5), 11*(width1*10), (height1 *
10)+(int)(height1*0.75), 180, 180);
        g.DrawLine(p, starty, startx, starty, startx +
Math.Max(width1*10,height1*10));
        g.DrawLine(p, starty+((width1*110)), startx, starty+(width1*110), startx
+ Math.Max(width1 * 10, height1 * 10));
        tempx = positX;
        tempy = positY;
    }
    #endregion
    #region shapes
    public void shape(Graphics g, int positx , int posity , int R, int n,int array)
    {
        Pen bp = new Pen(Color.Black, 1);
        double a = 0;
        double l =positx;
        double x = positx;
```

```
            double y = posity;
            for (int k = 0; k < array; k++)
            {
                for (int i = 1; i < n * 2 + 2; i++)
                {
                    if (i % 2 == 0)
                    {
                        px[i] = x + R * Math.Cos(a * Math.PI / 180);
                        py[i] = y + R * Math.Sin(a * Math.PI / 180);
                    }
                    a += 180 / n;
                }
                for (int i = 2; i < n * 2 + 2; i += 2)
                {
                    for (int j = i; j < n * 2 + 2; j += 2)
                    {
                        g.DrawLine(bp, (float)px[i], (float)py[i], (float)px[j],
(float)py[j]);
                        // MessageBox.Show((px[i]*2).ToString() + "  " +
(py[i]*2).ToString(), (px[j]*2).ToString() + "  " + (py[j]*2).ToString());
                    }
                }
                px[n * 2 + 1] = px[1];
                py[n * 2 + 1] = py[1];
                x += positx + (R*0.75);
            }
            l += (R * 0.70) * array*(array-1);
            g.DrawLine(p,positx-(positx/2),posity,(int)l,posity);
        }
        #endregion
```

```
 }
#endregion
 #region Buttons
 private void RectFarm_Click(object sender, EventArgs e)
 {
    rectfarm = true;
    circfarm = false;
    triafarm = false;
    diamfarm = false;
 }
 private void TriaFarm_Click(object sender, EventArgs e)
 {
    triafarm = true;
    rectfarm = false;
    circfarm = false;
    diamfarm = false;
 }
 private void DiamForm_Click(object sender, EventArgs e)
 {
    diamfarm = true;
    triafarm = false;
    rectfarm = false;
    circfarm = false;
 }
 private void CircFarm_Click(object sender, EventArgs e)
 {
    circfarm = true;
    triafarm = false;
    rectfarm = false;
    diamfarm = false;
```

```
}
private void DrawButton_Click(object sender, EventArgs e)
{
    positioX = Convert.ToInt16(PositionX.Value);
    positioY = Convert.ToInt16(PositionY.Value);
    fwidth = Convert.ToInt16(Widths.Value);
    fheight = Convert.ToInt16(Heights.Value);
    Graphics graphic = DrawBar.CreateGraphics();
    DrawFarm draw = new DrawFarm();
    if (rectfarm == true)
    {
        draw.rectangleFarm(graphic,positioX,positioY,fwidth,fheight);
    }
    else
        if (triafarm == true)
        {
            draw.triangleFarm(graphic,positioX,positioY,fwidth,fheight);
        }
        else
            if (diamfarm == true)
            {
                draw.diamondFarm(graphic, positioX, positioY, fwidth, fheight);
            }
            else
                if (circfarm == true)
                {
                    draw.circleFarm(graphic, positioX, positioY, fwidth, fheight);
                }
}
private void DrwShape_Click(object sender, EventArgs e)
```

```
        {
            int R = 0, n = 0;
            int arr = 0;
            int posX = 0, posY = 0;
            posX = Convert.ToInt16(ShPosX.Value)*10;
            posY = Convert.ToInt16(ShPosY.Value)*10;
            R = Convert.ToInt16(ShRad.Value);
            n = Convert.ToInt16(ShSid.Value);
            arr = Convert.ToInt16(ShArr.Value);
            DrawFarm drw = new DrawFarm();
            Graphics grp = DrawBar.CreateGraphics();
            drw.shape(grp, posX, posY, R * 10, n, arr);
        }
        private void Clear_Click(object sender, EventArgs e)
        {
            Graphics g = DrawBar.CreateGraphics();
            g.Clear(Color.White);
        }
        #endregion
    }
}
```

2-Ilova

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace Shaper
{
   public partial class Shaper : Form
   {
      public Shaper()
      {
         InitializeComponent();
      }
     public Boolean dgbutton = false, dsqrbuttton = false, dtribtn = false;
      public int pheigth = 0, pwidth = 0, a = 0, b = 0, c = 0,farmstart = 0, farmend =
0;
      int x0 = 0, x1 = 0, x2 = 0, x3 = 0, x4 = 0, y0 = 0, y1 = 0, y2 = 0, y3 = 0, y4 =
0;
      public void GraphicXY()
      {
         int i;
         pwidth = DrawPanel.Width;
         pheigth = DrawPanel.Height;
         Pen pen = new Pen(Color.Black, 1);
         Graphics g = DrawPanel.CreateGraphics();
         g.Clear(Color.White);
        g.DrawLine(pen, new Point(pwidth / 2, 0), new Point(pwidth / 2, pheigth));
         g.DrawLine(pen, new Point(pwidth / 2, 0), new Point(pwidth / 2 - 10, 10));
         g.DrawLine(pen, new Point(pwidth / 2, 0), new Point(pwidth / 2 + 10, 10));
         g.DrawLine(pen, new Point(pwidth, pheigth / 2), new Point(0, pheigth /
2));
         g.DrawLine(pen, new Point(pwidth, pheigth / 2), new Point(pwidth - 10,
pheigth / 2 + 10));
         g.DrawLine(pen, new Point(pwidth, pheigth / 2), new Point(pwidth - 10,
pheigth / 2 - 10));
```

```
            for (i = 10; i < pwidth; i += 10)
            {
                g.DrawLine(pen, new Point(pwidth / 2 - 5, pheigth / 2 + i), new
Point((pwidth / 2) + 5, pheigth / 2 + i));
                g.DrawLine(pen, new Point(pwidth / 2 - 5, pheigth / 2 - i), new
Point((pwidth / 2) + 5, pheigth / 2 - i));
            }
            for (i = 10; i < pheigth; i += 10)
            {
                g.DrawLine(pen, new Point(pwidth / 2 + i, pheigth / 2 - 5), new
Point(pwidth / 2 + i, pheigth / 2 + 5));
                g.DrawLine(pen, new Point(pwidth / 2 - i, pheigth / 2 - 5), new
Point(pwidth / 2 - i, pheigth / 2 + 5));
            }
            if (dgbutton)
            {
                DrawGraph();
            }
            if (dsqrbuttton)
            {
                Shape();
            }
            if (dtribtn)
            {
                Triangle();
            }
        }
        int sqrA = 0, sqrB = 0;
        public void Shape()
        {
```

```
        try
          {
            if (trackBar2.Value == 1)
            {
              sqrA = Convert.ToInt32(SqrA.Text);
              sqrB = sqrA;
            }
            if (trackBar2.Value == 2)
            {
              sqrA = Convert.ToInt32(SqrA.Text);
              sqrB = Convert.ToInt32(SqrB.Text);
            }
            Pen pen = new Pen(Color.Blue, 2);
            Graphics g = DrawPanel.CreateGraphics();
            Point[] point = { new Point(pwidth / 2, (pheigth / 2) - 20), new
Point(pwidth / 2, pheigth / 2), new Point((pwidth / 2) + 20, pheigth / 2) };
            g.DrawRectangle(pen, new Rectangle((pwidth / 2) + 10, (pheigth / 2) +
10, sqrB*10, sqrA*10));
        }
          catch (Exception) { }
      }
    public void DrawGraph()
      {
        pwidth = DrawPanel.Width;
        pheigth = DrawPanel.Height;
        int k = 40;
        farmstart = pheigth / 4;
        farmend = (pheigth / 2) + (pheigth / 4);
        try
        {
```

```
            #region Graph
            if (GraphAtxt.Text != null)
            {
                a = Convert.ToInt32(GraphAtxt.Text);
            }
            if (GraphBtxt.Text != null)
            {
                b = Convert.ToInt32(GraphBtxt.Text);
            }
            if (GraphCtxt.Text != null)
            {
                b = Convert.ToInt32(GraphCtxt.Text);
            }
           x0 = 0;
            y0 = (int)(a * Math.Pow(x0, 2)) + (b * x0) + c;
            x1 = 1;
            y1 = (int)(a * Math.Pow(x1, 2)) + (b * x1) + c;
            x2 = 2;
            y2 = (int)(a * Math.Pow(x2, 2)) + (b * x2) + c;
            x3 = -1;
            y3 = (int)(a * Math.Pow(x3, 2)) + (b * x3) + c;
            x4 = -2;
            y4 = (int)(a * Math.Pow(x4, 2)) + (b * x4) + c;
            Pen pen = new Pen(Color.Green, 4);
            Graphics g = DrawPanel.CreateGraphics();
            g.DrawLine(pen, new Point(pwidth / 2 - x2 * 10, pheigth / 2 - y2 * 10),
new Point(pwidth / 2 - x1 * 10, pheigth / 2 - y1 * 10));
            g.DrawLine(pen, new Point(pwidth / 2 - x4 * 10, pheigth / 2 - y4 * 10),
new Point(pwidth / 2 - x3 * 10, pheigth / 2 - y3 * 10));
```

```
            g.DrawBezier(pen, new Point(pwidth / 2 - x3 * 10, pheigth / 2 - y3 * 10),
new Point(pwidth / 2 - x0 * 10, pheigth / 2 - y0 * 10), new Point(pwidth / 2 - x0 *
10, pheigth / 2 - y0 * 10), new Point(pwidth / 2 - x1 * 10, pheigth / 2 - y1 * 10));
            #endregion
            #region Drawing
            #region draw cfg
           //Pen pen = new Pen(Color.Green, 4);
           // Graphics g = DrawPanel.CreateGraphics();
            int str = farmstart, tst = farmstart;
            tst -= pheigth / 8;
            str -= pheigth / 8;
            Pen p = new Pen(Color.Black, 2);
            #endregion
            for (int j = farmstart; j < farmend + farmstart; j += k )
            {
               #region Triangle
               //g.DrawRectangle(p, str, farmend - pheigth / 2, (farmend - pheigth /
22) + k, k);
               // Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new
Point(tst + (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth /
2) + k) };
               // g.DrawPolygon(pen, point);
               //tst += k;
               #endregion
               #region Square
               /*
                *
                * g.DrawRectangle(p, str, farmend - pheigth / 2, (farmend - pheigth /
22) + k, k);
               g.DrawRectangle(p, str +l,farmend - pheigth / 2, k , k);
```

```
            l += k;
            */
            #endregion
        }
        #endregion
        #region circle
        /*
            //int l = 0;
            tst = farmstart/2;
            l = farmend;
            while(l>=pwidth/2)
            {
                Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new
Point(tst + (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth /
2) + k) };
                g.DrawPolygon(pen, point);
                tst += 20;
                l -=14;
                h += 4;
            }
            while (l < pwidth-(pwidth/4) -14)
            {
                Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new
Point(tst + (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth /
2) + k) };
                g.DrawPolygon(pen, point);
                tst += 20;
                l += 14;
                h -= 4;
```

```
                }
                g.DrawLine(p, farmstart - (DrawPanel.Width / 8) + 5, pheigth / 4 + k,
DrawPanel.Width - (DrawPanel.Width / 4) + k + k / 2 + k / 8, pheigth / 4 + k);
                Pen red = new Pen(Color.Red, 2);
                g.DrawArc(red, farmstart - (DrawPanel.Width / 8) + 3, pheigth / 4 - k,
DrawPanel.Width - (DrawPanel.Width / 4) - 5, k * 4, 180, 180);
                  */
            #endregion
            }
        catch (Exception) { }
    }
    int triA = 0, triB = 0, triC = 0;
    public void Triangle()
    {
        try
        {
            if (trackBar1.Value == 1)
            {
                triA = Convert.ToInt32(TriangleAtxt.Text) * 10;
                triB = triA;
                triC = triB;
            }
            else if (trackBar1.Value == 2)
            {
                triA = Convert.ToInt32(TriangleAtxt.Text) * 10;
                triC = Convert.ToInt32(TriangleBtxt.Text) * 10;
                triB = triA;
            }
            else if (trackBar1.Value == 3)
```

```
            {
                triA = Convert.ToInt32(TriangleAtxt.Text) * 10;
                triB = Convert.ToInt32(TriangleBtxt.Text) * 10;
                triC = Convert.ToInt32(TriangleCtxt.Text) * 10;
            }
            Pen pen = new Pen(Color.Violet, 2);
            Graphics g = DrawPanel.CreateGraphics();
            Point[] point = { new Point((pwidth / 2) - triB, (pheigth / 2 - triB) - triA),
new Point((pwidth / 2) - triB, (pheigth / 2) - triB), new Point(((pwidth / 2) - triB) -
triC, (pheigth / 2) - triB) };
            g.DrawPolygon(pen, point);
        }
        catch (Exception) { }
    }
    #region Buttons
    private void HidePanel_Click(object sender, EventArgs e)
    {
        WorkPanel.Visible = false;
        ShowPanel.Visible = true;
    }
    private void ShowPanel_Click(object sender, EventArgs e)
    {
        WorkPanel.Visible = true;
        ShowPanel.Visible = false;
    }
    private void XYTimer_Tick(object sender, EventArgs e)
    {
        GraphicXY();
    }
    string Date1, times = null, Date2 = null;
```

```
        private void Time_Tick(object sender, EventArgs e)
        {
            times = null;
            Date2 = null;
            DateTime dt = DateTime.Now;
            Date1 = dt.ToString();
            for (int i = 0; i < 10; i++)
            {
                Date2 += Date1[i];
            }
            for (int i = 11; i < Date1.Length; i++)
            {
                times += Date1[i];
            }
            this.TimeTXT.Text = times;
            this.DateTXT.Text = Date2;
        }
        private void About_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Creator is Murodjon Rakhmonov", "Shaper v1.0.0.0",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void About_MouseHover(object sender, EventArgs e)
        {
            About.ForeColor = Color.Black;
        }
        private void About_MouseLeave(object sender, EventArgs e)
        {
            About.ForeColor = Color.White;
        }
```

```
private void textBox1_Click(object sender, EventArgs e)
{
    GraphAtxt.Clear();
    dgbutton = false;
}
private void GraphBtxt_Click(object sender, EventArgs e)
{
    GraphBtxt.Clear();
    dgbutton = false;
}
private void GraphCtxt_Click(object sender, EventArgs e)
{
    GraphCtxt.Clear();
    dgbutton = false;
}
private void trackBar1_Scroll(object sender, EventArgs e)
{
    if (trackBar1.Value == 1)
    {
        TriangleAtxt.Visible = true;
        TriangleBtxt.Visible = false;
        TriangleCtxt.Visible = false;
    }
    else if (trackBar1.Value == 2)
    {
        TriangleAtxt.Visible = true;
        TriangleBtxt.Visible = true;
        TriangleCtxt.Visible = false;
    }
    else if (trackBar1.Value == 3)
```

```
    {
        TriangleAtxt.Visible = true;
        TriangleBtxt.Visible = true;
        TriangleCtxt.Visible = true;
    }
}
private void trackBar2_Scroll(object sender, EventArgs e)
{
    if (trackBar2.Value == 1)
    {
        SqrA.Visible = true;
        SqrB.Visible = false;
    }
    else if (trackBar2.Value == 2)
    {
        SqrA.Visible = true;
        SqrB.Visible = true;
    }
}
private void GraphDraw_Click(object sender, EventArgs e)
{
    dgbutton = true;
    XYTimer.Enabled = true;
}
private void SquareDraw_Click(object sender, EventArgs e)
{
    dsqrbuttton = true;
    XYTimer.Enabled = true;
}
private void SqrA_Click(object sender, EventArgs e)
```

```
{
    SqrA.Clear();
    dsqrbuttton = false;
}
private void SqrB_Click(object sender, EventArgs e)
{
    SqrB.Clear();
    dsqrbuttton = false;
}
private void TriangleDraw_Click(object sender, EventArgs e)
{
    dtribtn = true;
    XYTimer.Enabled = true;
}
private void TriangleAtxt_Click(object sender, EventArgs e)
{
    TriangleAtxt.Clear();
    dtribtn = false;
}
private void TriangleBtxt_Click(object sender, EventArgs e)
{
    TriangleBtxt.Clear();
    dtribtn = false;
}
private void TriangleCtxt_Click(object sender, EventArgs e)
{
    TriangleCtxt.Clear();
    dtribtn = false;
}
#endregion
```

```
        private void RecFarm_Click(object sender, EventArgs e)
        {
            XYTimer.Enabled = false;
            pwidth = DrawPanel.Width;
            pheigth = DrawPanel.Height;
            int k = 40;
            farmstart = pheigth / 4;
            farmend = (pheigth / 2) + (pheigth / 4);
            int l=0;
            Pen pen = new Pen(Color.Green, 4);
            Graphics g = DrawPanel.CreateGraphics();
            g.Clear(Color.White);
            int str = farmstart, tst = farmstart;
            tst -= pheigth / 8;
            str -= pheigth / 8;
            Pen p = new Pen(Color.Black, 2);
            for (int j = farmstart; j < farmend + farmstart; j += k)
            {
                g.DrawRectangle(p, str, farmend - pheigth / 2, (farmend - pheigth / 22)
+ k, k);
                g.DrawRectangle(p, str + l, farmend - pheigth / 2, k, k);
                l += k;
            }
        }
        private void DiamFarm_Click(object sender, EventArgs e)
        {
            XYTimer.Enabled = false;
            pwidth = DrawPanel.Width;
            pheigth = DrawPanel.Height;
            int k = 40;
```

```
            farmstart = pheigth / 4;
            farmend = (pheigth / 2) + (pheigth / 4);
           // int l = 0;
            Pen pen = new Pen(Color.Green, 4);
            Graphics g = DrawPanel.CreateGraphics();
            g.Clear(Color.White);
            int str = farmstart, tst = farmstart, h = 0;
            tst -= pheigth / 8;
            str -= pheigth / 8;
            Pen p = new Pen(Color.Black, 2);
            for (int j = farmstart; j < farmend + farmstart-k/2; j +=k/2)
            {
              Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new Point(tst
+ (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth / 2) + k)
};
                g.DrawPolygon(pen, point);
                g.DrawRectangle(p, str, farmend - pheigth / 2, (farmend - pheigth / 22) +
k, k);
                tst += k/2;
            }
        }
        private void TrianFarm_Click(object sender, EventArgs e)
        {
            XYTimer.Enabled = false;
            pwidth = DrawPanel.Width;
            pheigth = DrawPanel.Height;
            int k = 40;
            farmstart = pheigth / 4;
            farmend = (pheigth / 2) + (pheigth / 4);
            // int l = 0;
```

```
            Pen pen = new Pen(Color.Green, 4);
            Graphics g = DrawPanel.CreateGraphics();
            g.Clear(Color.White);
            int str = farmstart, tst = farmstart, h = 0;
            tst -= pheigth / 8;
            str -= pheigth / 8;
            Pen p = new Pen(Color.Black, 2);
            for (int j = farmstart; j < farmend + farmstart; j += k)
            {
                Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new
Point(tst + (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth /
2) + k) };
                g.DrawPolygon(pen, point);
                g.DrawRectangle(p, str, farmend - pheigth / 2, (farmend - pheigth / 22) +
k, k);
                tst += k;
            }
        }
        private void CircFarm_Click(object sender, EventArgs e)
        {
            XYTimer.Enabled = false;
            pwidth = DrawPanel.Width;
            pheigth = DrawPanel.Height;
            int k = 40;
            farmstart = pheigth / 4;
            farmend = (pheigth / 2) + (pheigth / 4);
            int l = 0;
            Pen p = new Pen(Color.Green, 4);
            Graphics g = DrawPanel.CreateGraphics();
            g.Clear(Color.White);
```

```
int str = farmstart, tst = farmstart, h = 0;
str -= pheigth / 8;
tst = farmstart / 2;
l = farmend;
try
{
   while (l >= pwidth / 2)
   {
      Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new
Point(tst + (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth /
2) + k) };
      g.DrawPolygon(p, point);
      tst += 20;
      l -= 14;
      h += 4;
   }
   while (l < pwidth - (pwidth / 4) - 14)
   {
      Point[] point = { new Point(tst, (farmend - pheigth / 2) + k), new
Point(tst + (k / 2), farmend - pheigth / 2 - h), new Point(tst + k, (farmend - pheigth /
2) + k) };
      g.DrawPolygon(p, point);
      tst += 20;
      l += 14;
      h -= 4;
   }
   Pen red = new Pen(Color.Red, 4);
   g.DrawArc(red, farmstart - (DrawPanel.Width / 8) + 3, pheigth / 4 - k,
DrawPanel.Width - (DrawPanel.Width / 4) - 5, k * 4, 180, 180);
```

```
                g.DrawLine(p, farmstart - (DrawPanel.Width / 8) + 5, pheigth / 4 + k,
DrawPanel.Width - (DrawPanel.Width / 4) + k + k / 2 + k / 8, pheigth / 4 + k);
            }
            catch (Exception) { }
        }
    }
}
```