O‘ZBEKISTON RESPUBLIKASI
OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

OLIY TA’LIM TIZIMI PEDAGOG VA RAHBAR KADRLARINI QAYTA TAYYORLASH VA ULARNING MALAKASINI OSHIRISHNI TASHKIL ETISH BOSH ILMIY - METODIK MARKAZI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI HUZURIDAGI PEDAGOG KADRLARNI QAYTA TAYYORLASH VA ULARNING MALAKASINI OSHIRISH TARMOQ MARKAZI

“INFORMATIKA VA AXBOROT TEXNOLOGIYALARI”

yo‘nalishi

“DASTURLASH TAMOYILLARI”

MODULI BO‘YICHA

O‘ Q U V –U S L U B I Y M A J M U A

TOSHKENT - 2016

O’ZBEKISTON RESPUBLIKASI
OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI

OLIY TA’LIM TIZIMI PEDAGOG VA RAHBAR KADRLARINI QAYTA TAYYORLASH VA ULARNING MALAKASINI OSHIRISHNI TASHKIL ETISH BOSH ILMIY-METODIK MARKAZI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI HUZURIDAGI PEDAGOG KADRLARNI QAYTA TAYYORLASH VA ULARNING MALAKASINI OSHIRISH TARMOQ MARKAZI

“DASTURLASH TAMOYILLARI” moduli bo'yicha

O’QUV-USLUBIY MAJMUA

TOШKEHT - 2016

Mazkur o`quv-uslubiy majmua Oliy va o`rta maxsus ta`lim vazirligining 2016 yil 6 apreldagi 137-sonli buyrug`i bilan tasdiqlangan o`quv reja va dastur asosida tayyorlandi.

Tuzuvchi:	TATU, «Informatika asoslari» kafedrasi assistenti	A.Z.Maxmudov
	TATU, «Informatika asoslari» kafedrasi katta o‘qituvchisi	Sh.T.Qosimova
	TATU, «Informatika asoslari» kafedrasi assistenti	Sh.B.Abidova
Taqrizchi:	TATU, AKT bo`yicha maslahatchi prorektori, Janubiy Koreyalik mutaxassis	Li Chul Su

O`quv-uslubiy majmua Toshkent axborot texnologiyalari universiteti Kengashining qarori bilan nashrga tavsiya qilingan (2016 yil 29 avgustdagi 1(661) -sonli bayonnoma)

PEER REVIEW

TO THE EDUCATION PROGRAM FOR THE COURSE OF RETRAINING PEDAGOGUE CADRES OF HIGHER EDUCATION ORGANIZATIONS IN THE DIRECTION OF "INFORMATICS AND INFORMATION TECHNOLOGIES"

The following program has been concluded according to the Presidential Decree №4732 adopted on 12th of June, 2015, "on measures for further enhancement of the system of retraining and improving qualification of executives and pedagogic cadres of higher educational institutions", and is aimed to retrain ~~cadres~~ with modern requirements, to enhance the content of the process of retraining as well as to continually rise professional competence of pedagogic cadres in higher educational institutions.

The content of the program includes basics of regulatory-legal framework and legal norms, advanced learning technologies and pedagogical skills, the application of information and communication technologies in educational process, practical foreign languages, basics of structural analysis and decision-making, scientific and applied research on the basis of specific subjects, technological progress and the latest developments in modern methods of organizing the educational process, teachers' professional competence and creativity, global Internet network, acquiring new ability and skills on mastering ways of distance teaching and formation of e-learning environment.

The topics mentioned in the program are formed on retraining pedagogues and improving qualification in the field of education, general requirements for the quality and preparation as well as the syllabus. With this, teachers of higher education institutions will be provided with competencies of professional skills, improving continual scientific activity, organizing educational-behavioral process and structural analysis of managing, as well as optimal decision-making in pedagogic situations by having them acquire modern education and innovation technologies on specialty, effective usage of advanced foreign experience, implementation of information communication technologies to teaching process, level of intensive learning of foreign languages.

In the content of the program prospects for the development of computer science and information technologies as well as reforms and foreign experience in the field of information and communication technologies in the Republic of Uzbekistan have been taken into consideration and actual topics have been added.

As the education program is useful for learners in the direction of "Informatics and information technologies" it totally matches with the requirements of the course of teacher training and retraining and it is expedient to be used in education process.

Vice rector of ICT, TUIT Chul Soo LEE

“ИНФОРМАТИКА ВА АХБОРОТ ТЕХНОЛОГИЯЛАРИ” ЙЎНАЛИШИ БЎЙИЧА ОЛИЙ ТАЪЛИМ МУАССАСАЛАРИ ПЕДАГОГ КАДРЛАРИНИ ҚАЙТА ТАЙЁРЛАШ ВА МАЛАКАСИНИ ОШИРИШ КУРСИ УЧУН ТАЙЁРЛАНГАН ЎҚУВ ДАСТУРИГА

Т А Қ Р И З

Ушбу дастур Ўзбекистон Республикаси Президентининг 2015 йил 12 июндаги “Олий таълим муассасаларининг раҳбар ва педагог кадрларини қайта тайёрлаш ва малакасини ошириш тизимини янада такомиллаштириш чора-тадбирлари тўғрисида” ги ПФ-4732-сон Фармонидаги устувор йўналишлар мазмунидан келиб чиққан ҳолда тузилган бўлиб, у замонавий талаблар асосида қайта тайёрлаш ва малака ошириш жараёнларининг мазмунини такомиллаштириш ҳамда олий таълим муассасалари педагог кадрларининг касбий компетентлигини мунтазам ошириб боришни мақсад қилади.

Дастур мазмуни олий таълимнинг норматив-ҳуқуқий асослари ва қонунчилик нормалари, илғор таълим технологиялари ва педагогик маҳорат, таълим жараёнларида ахборот-коммуникация технологияларини қўллаш, амалий хорижий тил, тизимли таҳлил ва қарор қабул қилиш асослари, махсус фанлар негизида илмий ва амалий тадқиқотлар, технологик тараққиёт ва ўқув жараёнини ташкил этишнинг замонавий услублари бўйича сўнгги ютуқлар, педагогнинг касбий компетентлиги ва креативлиги, глобал Интернет тармоғи, масофадан ўқитиш усуллари ва Олий таълим муассасаларида электрон-таълим муҳитини шакллантиришни ўзлаштириш бўйича янги билим, кўникма ва малакаларини эгаллашни назарда тутади.

Дастур доирасида берилаётган мавзулар таълим соҳаси бўйича педагог кадрларни қайта тайёрлаш ва малакасини ошириш мазмуни, сифати ва уларнинг тайёргарлигига қўйиладиган умумий малака талаблари ва ўқув режалари асосида шакллантирилган бўлиб, бу орқали олий таълим муассасалари педагог кадрларининг соҳага оид замонавий таълим ва инновация технологиялари, илғор хорижий тажрибалардан самарали фойдаланиш, ахборот-коммуникация технологияларини ўқув жараёнига кенг татбиқ этиш, чет тилларини интенсив ўзлаштириш даражасини ошириш ҳисобига уларнинг касбий маҳоратини, илмий фаолиятини мунтазам юксалтириш, олий таълим муассасаларида ўқув-тарбия жараёнларини ташкил этиш ва бошқаришни тизимли таҳлил қилиш, шунингдек, педагогик вазиятларда оптимал қарорлар қабул қилиш билан боғлиқ компетенцияларга эга бўлишлари таъминланади.

Дастур мазмунида информатика ва ахборот технологияларининг ривожланиш истиқболлари ҳамда Ўзбекистон Республикасида ахборот коммуникация технологиялари соҳасидаги ислоҳотлар ва хориж тажрибаси эътиборга олинган ва ҳозирги кунда мазкур йўналишда энг долзарб ҳисобланган мавзулар қамраб олинган.

Ўқув дастури “Информатика ва ахборот технологиялари” йўналишидаги тингловчиларга фойдали бўлиб, педагог кадрларни қайта тайёрлаш ва малакасини ошириш курси талабларига тўлиқ жавоб беради ва уни ўқув жараёнида қўллаш мақсадга мувофиқдир.

ТАТУ АКТСКТ факультети
декани, п.ф.д., профессор: **Ф.М.Закирова**

MUNDARIJA

I. БЎЛИМ

ИШЧИ ДАСТУР

I. ISHCHI DASTUR

Kirish

Dastur O‘zbekiston Respublikasi Prezidentining 2015 yil 12 iyundagi “Oliy ta’lim muassasalarining rahbar va pedagog kadrlarini qayta tayyorlash va malakasini oshirish tizimini yanada takomillashtirish chora-tadbirlari to‘g‘risida” gi PF-4732-son Farmonidagi ustuvor yo‘nalishlar mazmunidan kelib chiqqan holda tuzilgan bo‘lib, u zamonaviy talablar asosida qayta tayyorlash va malaka oshirish jarayonlarining mazmunini takomillashtirish hamda oliy ta’lim muassasalari pedagog kadrlarining kasbiy kompetentligini muntazam oshirib borishni maqsad qiladi. Dastur mazmuni oliy ta’limning normativ-huquqiy asoslari va qonunchilik normalari, ilg‘or ta’lim texnologiyalari va pedagogik mahorat, ta’lim jarayonlarida axborot-kommunikasiya texnologiyalarini qo‘llash, amaliy xorijiy til, tizimli tahlil va qaror qabul qilish asoslari, maxsus fanlar negizida ilmiy va amaliy tadqiqotlar, texnologik taraqqiyot va o‘quv jarayonini tashkil etishning zamonaviy uslublari bo‘yicha so‘nggi yutuqlar, pedagogning kasbiy kompetentligi va kreativligi, global Internet tarmog‘i, multimedia tizimlari va masofadan o‘qitish usullarini o‘zlashtirish bo‘yicha yangi bilim, ko‘nikma va malakalarini shakllantirishni nazarda tutadi.

Dasturda kompyuterlar, insonlar va dasturlash o‘rtasidagi o‘zaro bog‘liqlik, dasturiy ta’minot, dasturlash asoslari, obyektlar, qiymat va toifalar, hisoblash, xatoliklar, kiritish va chiqarish, kiritish - chiqarish oqimi modeli, fayllar, grafika, foydalanuvchining grafik interfeyslari, vektor va bo‘sh xotira, vektorlar va massivlar ustida amallar va ular bilan ishlashda yuzaga keladigan muammolar va ularning yechimlari bayon etilgan.

Qayta tayyorlash va malaka oshirish yo‘nalishining o‘ziga xos xususiyatlari hamda dolzarb masalalaridan kelib chiqqan holda dasturda tinglovchilarning maxsus fanlar doirasidagi bilim, ko‘nikma, malaka hamda kompetensiyalariga qo‘yiladigan talablar takomillashtirilishi mumkin.

Modulning maqsadi va vazifalari

Dasturlash tamoyillari modulining maqsad va vazifalari: Dasturlashda ma’lumotlarning turli tarkibiy tuzilmalarini tadbiq etish. Tilning sodda va murakkab tuzilmalarini qo‘llash. Algoritmlarni baholash. Qo‘yilgan masalani yechish algoritmini tanlash, tanlovni asoslash, algoritmni tadbiq etish. S/C++ dasturlash tilining asosiy konstruksiyalari, ma’lumotlar toifalari va tuzilmalarini qo‘llash. S/C++ dasturlash tilida aniq algoritmlarini tadbiq etish. Tilning statik va dinamik imkoniyatlarni tadbiq etish. Obyektga yo‘naltirilgan dasturlash asoslarini qo‘llash. Dasturiy ishlanmalarni ishlab chiqarish jarayonini ta’minlash. Ma’lumotlarning dinamik informasion tuzilmasini tadbiq etish. Aniq loyihalarni ishlab chiqish va tadbiq etish. Dasturiy ishlanmalarni testlash va sozlash. Murakkab dasturiy tizimlarni tashkil etish. Natijalarni tahlil qilish.

Modul bo‘yicha tinglovchilarning bilimi, ko‘nikmasi, malakasi va kompetensiyalariga qo‘yiladigan talablar

“Dasturlash tamoyillari” modulini o‘zlashtirish jarayonida amalga oshiriladigan masalalar doirasida**:**

Tinglovchi:

- dasturlash asoslari qismida dasturlash tilining tuzilmasi (C++ tili asosi), funksiyalari va asosiy parametrlari;
- dasturiy ta’minot konfigurasiyasini boshqarish;
- dasturiy ta’minotni testlash va sifatini ta’minlash usullari haqida **bilimlarga ega bo‘lishi**;

Tinglovchi:

- qo‘yilgan masalaga mos algoritmlarni tanlash;
- dastur strukturasini ishlab chiqish;
- dasturda xatoliklarni bartaraf etish va boshqarish;
- grafik foydalanuvchi interfeysini shakllantirish va boshqarish **ko‘nikma va malakalarini egallashi;**

Tinglovchi:

- C++ dasturlash tilining asoslarini va dasturiy muhitlarni tadbiq qilish;
- tilning sodda va murakkab tuzilmalarini qo‘llash;
- algoritmlarni baholash, qo‘yilgan masalani yechish algoritmini tanlash, tanlovni asoslash va algoritmni tadbiq etish;
- obyektga mo‘ljallangan dasturlash texnologiyalaridan foydalanish **kompetensiyalarni egallashi lozim.**

Modulni tashkil etish va o‘tkazish bo‘yicha tavsiyalar

“Dasturlash tamoyillari” moduli ma’ruza va amaliy mashg‘ulotlar shaklida olib boriladi.

Modulni o‘qitish jarayonida ta’limning zamonaviy metodlari, pedagogik texnologiyalar va axborot-kommunikasiya texnologiyalari qo‘llanilishi nazarda tutilgan:

- ma’ruza darslarida zamonaviy kompyuter texnologiyalari yordamida prezentasion va elektron-didaktik texnologiyalardan foydalanish;
- o‘tkaziladigan amaliy mashg‘ulotlarda texnik vositalardan, ekspress-so‘rovlar, test so‘rovlari, aqliy hujum, guruhli fikrlash, kichik guruhlar bilan ishlash, kollokvium o‘tkazish, va boshqa interaktiv ta’lim usullarini qo‘llash nazarda tutiladi.

Modulning o‘quv rejadagi boshqa modullar bilan bog‘liqligi va uzviyligi

“Dasturlash tamoyillari” moduli mazmuni o‘quv rejadagi “Ma’lumotlar bazasini boshqarish tizimlari” va “Operasion tizimlar” o‘quv modullari bilan uzviy bog‘langan holda pedagoglarning dasturlash tamoyillari bo‘yicha kasbiy pedagogik tayyorgarlik darajasini oshirishga xizmat qiladi.

Modulning oliy ta’limdagi o‘rni

Modulni o‘zlashtirish orqali tinglovchilar dasturlarni yaratishda mos algoritmni tanlash, dasturni bajarish, xatoliklarni aniqlash va ularni bartaraf etish usullarini o‘rganish, amalda qo‘llash va baholashga doir kasbiy kompetentlikka ega bo‘ladilar.

Modul bo‘yicha soatlar taqsimoti

№	Modul mavzulari	Tinglovchining o‘quv yuklamasi, soat					
		Hammasi	Auditoriya o‘quv yuklamasi				Mustaqil ta’lim
			Jami	jumladan			
				Nazariy	Amaliy	Ko‘chma mashg’ulot	
1.	Kompyuterlar, insonlar va dasturlash. Dasturlash asoslari. Dasturlashda ma’lumotlarning turli tarkibiy tuzilmalarini tadbiq etish. Tilning sodda va murakkab tuzilmalarini qo‘llash. Algoritmlarni baholash.	4	4	2	2	2	
2.	Qo‘yilgan masalani yechish algoritmini tanlash, tanlovni asoslash, algoritmni tadbiq etish. C/C++ dasturlash tilining asosiy konstruksiyalari, ma’lumotlar toifalari va tuzilmalarini qo‘llash.	4	4	2	2		
3.	C/C++ dasturlash tilida aniq algoritmlarini tadbiq etish. C++ tilining grafik imkoniyatlari. Foydalanuvchi grafik interfeysini tashkil etish. Ro‘yxat, navbat va stek.	6	6	2	2		
4.	C++ tilida sinf va ob’ektlarni tashkil etish va ulardan foydalanish. Tilning statik va dinamik imkoniyatlarni tadbiq etish.	6	4	2	2		2
5.	Ob’ektga yo‘naltirilgan dasturlash asoslarini qo‘llash. Dasturiy ishlanmalarni ishlab chiqarish jarayonini ta’minlash. Ma’lumotlarning dinamik informatsion tuzilmasini tadbiq etish. Aniq loyihalarni ishlab chiqish va tadbiq etish.	6	6	2	4		
6.	Dasturiy ishlanmalarni testlash va sozlash. Murakkab dasturiy tizimlarni tashkil etish. Natijalarni tahlil qilish.	4	4	2	2		
	Jami:	**30**	**28**	**12**	**14**	**2**	**2**

Nazariy mashg‘ulotlar mazmuni

1 – ma’ruza. Kompyuterlar, odamlar va dasturlash.

Ushbu mavzu, biz dasturlashda muhim va qiziqarli deb hisoblaydigan, bizning qarashlarimizni ifodalaydi. Mavzu sizga bizning asosiy maqsadlarimizni va dasturchilar qanday bo‘lishi haqidagi tushunchani beradi. Ushbu mavzuda ayrim savollarga javob topishga xarakat qilamiz. Nima uchun dasturlash diqqatga sazovor mashg‘ulot hisoblanadi? Bizning sivilizasiyamizda dasturlash qanday ahamiyatga ega? Dasturchilar kelgusida mag‘rurlanishi mumkin bo‘lgan sohalar? Dasturiy ta’minotlarni ishlab chiqish, qo‘llash sohasida qanday masalalar ustida bosh qotirilmagan? Dasturlash axborot texnolgiyalari, dasturiy ta’minotni ishlab chiqishda, kompyuter ilmlari sohasida qanday o‘ringa ega? Dasturchilar nimalar bilan shug‘ullanadi? Ular qanday mahoratga ega bo‘lmoqlari lozim?

2 – ma’ruza. Dasturlash asoslari.

Ushbu mavzuda asosan dasturni yratishda obyekt, sinf, qiymat va toifalarni o‘zlashtirish. Birinchi dasturni shakllantirish. Yaratilgan dasturni ishga tushurish va unda yuz beradigan xatolliklar turlari va ularni bartaraf etish usullari. Hisoblash kabi mavzularga to‘xtalib o‘tilgan. Shuningdek ayrim standart kutubxona fayllari va ularni qo‘llash bo‘yicha ko‘rsatmalar berilgan.

3 - mavzu: Kiritish va chiqarish

Mavzuda asosan dasturlashtirish jarayonida kiritish va chiqarish oqimlarini boshqarish, Kiritish - chiqarish oqimi modeli, fayllar va fayllar yordamida kiritish chiqarishni amalga oshirish usullari, kiritish va chiqarishni sozlash kabi masalalar atroflicha o‘rganib chiqilgan.

4 - mavzu: Foydalanuvchi grafik interfeyslari

Bu mavzuda C++ tilining grafik imkoniyatlari va foydalanuvchi grafik interfeysi hususida so‘z yuritilgan. Shuningdek grafik tasvirlarni ekranga chiqarish,

ularni boshqarish, koordinatalar sistemasi, oynalarni boshqarish, tasvirlarni hosil qiluvchi sinflar va obyektlar haqida ham so‘z yuritilgan.

5 - mavzu: Vektorlar va bo‘sh xotira

Ushbu ma’ruzada vektorlar bo‘yicha asosiy tushunchalar, xotira, adres va ko‘rsatgich tushunchalari, bo‘sh xotira va ko‘rsatkichlar, bo‘sh xotiraga joylashtirish, destruktorlar va elementlarga kirish kabi mavzular yoritib berilgan.

6 - mavzu: Vektorlar,shablonlar va istisno

Ushbu ma’ruzada asosan vektorlar va massivlar, ularni tashkil etish, bir massivdan ikkinchi bir massivga nusxa ko‘chirish, ular ustida turli amallarni bajarish, ularning elementlariga murojaat qilish usullari va ularga misollar keltirib o‘tilgan.

AMALIY MASHG‘ULOTLAR MAZMUNI

1-amaliy mashg‘ulot. C++ tilining obyektlari, qiymat va toifalarini o‘zlashtirish. Birinchi oddiy dasturni tashkil etish.

C++ dasturlash tili uchun kompilyatorni o‘rnatish va sozlash. C++ tilining standart kutubxonasi va alifbosi bilan tanishish. Birinchi oddiy dasturni inisializatsiya qilish. Dasturiy yechim to‘g‘riligini avtomatik testlovchi tizim http://acm.tuit.uz saytidan ro‘yhatdan o‘tish va murakkab bo‘lmagan dasturlarni tizimga yuborishni o‘rganish.

2-amaliy mashg‘ulot. C++ dasturlash tilida kiritish va chiqarish

C++ dasturlash tilida kiritish va chiqarish operatorlari, fayl yordamida kiritish chiqarish usullarini o‘rganish. Masalaning algoritmi va xususiyatidan kelib chiqib kiritish va chiqarish operatorlarini tanlash va ulardan foydalanish amaliy ko‘nikmalarini xosil qilish.

3-amaliy mashg‘ulot. C++ tilining grafik imkoniyatlari. Foydalanuvchi grafik interfeysini tashkil etish

C++ tilining grafik imkoniyatlari bilan tanishish, kompilyatorni grafik muhit uchun sozlash, grafik kutubxona fayllarini o‘rnatish va sozlash bo‘icha amaliy ko‘nikmalarga ega bo‘lish. Berilgan masalalarni C++ ning grafik operatorlari, sinflari va obyektlaridan foydalanib yechish.

4-amaliy mashg‘ulot. C++ tilida *vector* yordamida massivlar ustida amallar bajarish.

C++ dasturlash tilida ko‘rsatkichlar va ularni tashkil etish, murojaat etish va foydalanish bo‘yicha amaliy ko‘nikma hosil qilish. Vector tushunchasini o‘rganish, vector ustida amallar bajarishga mo‘ljallangan operatorlar, funksiyalar va amallar bilan tanishib chiqish. http://acm.tuit.uz saytidan amaliy mashg‘ulot mavzusiga mos masalalarni dasturini tuzish va tizimga yuborish.

5, 6 – amaliy mashg‘ulot. Ro‘yxat, navbat va stek

C++ dasturlash tilining dinamik information tuzulmalari bilan ishlashni o‘rganish. Ro‘yxatni tashkil etish, unga element qo‘shish, o‘chirish va ro‘yxatdagi tugun bo‘sh yoki bo‘sh emasligini tekshirish bo‘yicha amliy ko‘nikma hosil qilish. Navbat va stekga doir masalalarni yechish. Ularni dasturini tuzish va testdan o‘tkazish.

7 – amaliy mashg‘ulot. C++ tilida sinf va obyektlarni tashkil etish va ulardan foydalanish

C++ dasturlash tilida sinf va obyektlarni tashkil etish, ularga murojaat etish va dasturda qo‘llash bo‘yicha amaliy ko‘nikmalarni shakllantirish. Berilgan variantlar asosida sinflarni tashkil etish dasturini tuzish. Dasturni testdan o‘tkazish va yuzaga kelgan xatoliklarni bartaraf etish usullari bilan atroflicha tanishish. Yaratilgan sinfga ma’lumotlarni qo‘shish va o‘chirish.

KO‘CHMA MASHG‘ULOT MAZMUNI

Mavzu: Toshkent axborot texnologiyalari universiteti ilmiy-o‘quv laboratoriyalari bilan tanishish va ularning ish jarayonini o‘rganish.

Mashg‘ulot davomida tinglovchilar universitetning “Algoritmlash va dasturlash” ilmiy-o‘quv laboratoriyasi, “Multimedia” o‘quv laboratoriyasi, “Robototexnika” ilmiy tadqiqot markazlarida bo‘lib, nazariy egallagan bilimlarini amaliyotga tadbiq etish ko‘nikmalariga ega bo‘lishini ta’minlash.

BAHOLASH MEZONI

№	Baholash turlari	Maksimal ball	Ballar
1	Keys topshiriqlari	2.5	1.2 ball
2	Mustaqil ish topshiriqlari		0.5 ball
3	Amaliy topshiriqlar		0.8 ball

II. BO`LIM

MODULNI O`QITISHDA FOYDALANILADIGAN INTERFAOL TA`LIM METODLARI

II. MODULNI O‘QITISHDA FOYDALANILADIGAN INTERFAOL TA’LIM METODLARI

“SWOT-tahlil” metodi

Metodning maqsadi: mavjud nazariy bilimlar va amaliy tajribalarni tahlil qilish, taqqoslash orqali muammoni hal etish yo‘llarini topishga, bilimlarni mustahkamlash, takrorlash, baholashga, mustaqil, tanqidiy fikrlashni, nostandart tafakkurni shakllantirishga xizmat qiladi.

Namuna: C++ dasturlash tilining SWOT tahlilini ushbu jadvalga tushiring.

S	C++ dasturlash tilidan foydalanishning kuchli tomonlari	Ushbu dasturlash tili tizimli dasturlash tili bo‘lib, boshqa dasturlash tillarida ham tizimli masalalarni yechishda C++ ga murojaat etiladi.
W	C++ dasturlash tilidan foydalanishning kuchsiz tomonlari	Dasturlash tilining strukturasi murakkabligi.
O	C++ dasturlash tilidan foydalanishning imkoniyatlari (ichki)	C++ dasturlash tilini mukkammal o‘zlashtirgan dasturchilar muammoli masalalarni yechishda va yuzaga keladigan xatoliklarni bartaraf etishda mukammal muhit hisoblanadi.
T	To‘siqlar (tashqi)	Ma’lumotlar xavfsizligining to‘laqonli ta’minlanmaganligi

Xulosalash» (Rezyume, Veyer) metodi

Metodning maqsadi: Bu metod murakkab, ko‘ptarmoqli, mumkin qadar, muammoli xarakteridagi mavzularni o‘rganishga qaratilgan. Metodning mohiyati shundan iboratki, bunda mavzuning turli tarmoqlari bo‘yicha bir xil axborot beriladi va ayni paytda, ularning har biri alohida aspektlarda muhokama etiladi. Masalan, muammo ijobiy va salbiy tomonlari, afzallik, fazilat va kamchiliklari, foyda va zararlari bo‘yicha o‘rganiladi. Bu interfaol metod tanqidiy, tahliliy, aniq mantiqiy fikrlashni muvaffaqiyatli rivojlantirishga hamda o‘quvchilarning mustaqil g‘oyalari, fikrlarini yozma va og‘zaki shaklda tizimli bayon etish, himoya qilishga imkoniyat yaratadi. “Xulosalash” metodidan ma’ruza mashg‘ulotlarida individual va juftliklardagi ish shaklida, amaliy va seminar mashg‘ulotlarida kichik guruhlardagi ish shaklida mavzu yuzasidan bilimlarni mustahkamlash, tahlili qilish va taqqoslash maqsadida foydalanish mumkin.

Metodni amalga oshirish tartibi:

trener tinglovchilarni 5-6 kishidan iborat kichik guruhlarga ajratadi;

trening maqsadi, shartlari va tartibi bilan ishtirokchilarni tanishtirgach, har bir guruhga umumiy muammoni tahlil qilinishi zarur bo‘lgan qismlari tushirilgan tarqatma materiallarni tarqatadi;

har bir guruh o‘ziga berilgan muammoni atroflicha tahlil qilib, o‘z mulohazalarini tavsiya etilayotgan sxema bo‘yicha tarqatmaga yozma bayon qiladi;

navbatdagi bosqichda barcha guruhlar o‘z taqdimotlarini o‘tkazadilar. Shundan so‘ng, trener tomonidan tahlillar umumlashtiriladi, zaruriy axborotlr bilan to‘ldiriladi va mavzu yakunlanadi

Namuna:

Dasturlash tillari					
C		C++		Visual C++	
afzalligi	kamchiligi	afzalligi	kamchiligi	afzalligi	kamchiligi
Xulosa:					

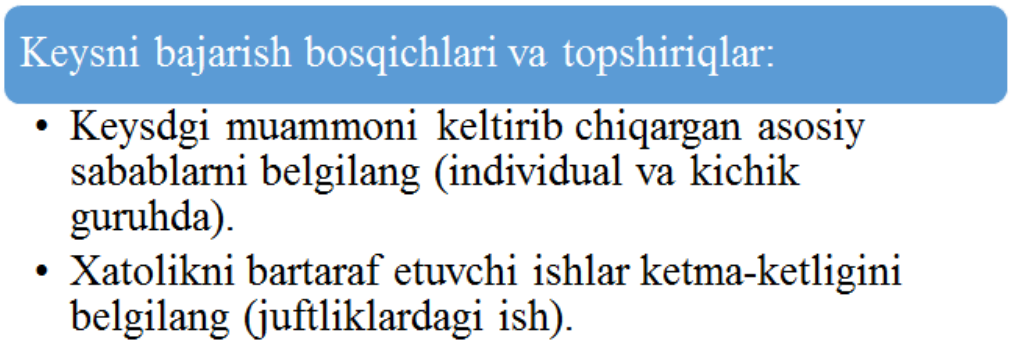
“Keys-stadi” metodi

«Keys-stadi» - inglizcha so‘z bo‘lib, («case» – aniq vaziyat, hodisa, «stadi» – o‘rganmoq, tahlil qilmoq) aniq vaziyatlarni o‘rganish, tahlil qilish asosida o‘qitishni amalga oshirishga qaratilgan metod hisoblanadi. Mazkur metod dastlab 1921 yil Garvard universitetida amaliy vaziyatlardan iqtisodiy boshqaruv fanlarini o‘rganishda foydalanish tartibida qo‘llanilgan. Keysda ochiq axborotlardan yoki aniq voqyea-hodisadan vaziyat sifatida tahlil uchun foydalanish mumkin. Keys harakatlari o‘z ichiga quyidagilarni qamrab oladi: Kim (Who), Qachon (When), Qayerda (Where), Nima uchun (Why), Qanday/ Qanaqa (How), Nima-natija (What).

“Keys metodi” ni amalga oshirish bosqichlari

Ish bosqichlari	Faoliyat shakli va mazmuni
1-bosqich: Keys va uning axborot ta’minoti bilan tanishtirish	✓ yakka tartibdagi audio-vizual ish; ✓ keys bilan tanishish(matnli, audio yoki media shaklda); ✓ axborotni umumlashtirish; ✓ axborot tahlili; ✓ muammolarni aniqlash
2-bosqich: Keysni aniqlashtirish va o‘quv topshirig‘ni belgilash	✓ individual va guruhda ishlash; ✓ muammolarni dolzarblik iyerarxiyasini aniqlash; ✓ asosiy muammoli vaziyatni belgilash
3-bosqich: Keysdagi asosiy muammoni tahlil etish orqali o‘quv topshirig‘ining yechimini izlash, hal etish yo‘llarini ishlab chiqish	✓ individual va guruhda ishlash; ✓ muqobil yechim yo‘llarini ishlab chiqish; ✓ har bir yechimning imkoniyatlari va to‘siqlarni tahlil qilish; ✓ muqobil yechimlarni tanlash
4-bosqich: Keys yechimini yechimini shakllantirish va asoslash, taqdimot.	✓ yakka va guruhda ishlash; ✓ muqobil variantlarni amalda qo‘llash imkoniyatlarini asoslash; ✓ ijodiy-loyiha taqdimotini tayyorlash; ✓ yakuniy xulosa va vaziyat yechimining amaliy aspektlarini yoritish

Keys. Berilgan topshiriq asosida dastur algoritmi tuzilib C++ dasturlash tilida dastur matni yozildi. Dasturni acm.tuit.uz saytiga yuborilganda “kompilyatsiyada hatolik” habari chiqdi. Ya’ni Sistema yechimni qabul qilmadi.



«FSMU» metodi

Texnologiyaning maqsadi: Mazkur texnologiya ishtirokchilardagi umumiy fikrlardan xususiy xulosalar chiqarish, taqqoslash, qiyoslash orqali axborotni o‘zlashtirish, xulosalash, shuningdek, mustaqil ijodiy fikrlash ko‘nikmalarini shakllantirishga xizmat qiladi. Mazkur texnologiyadan ma’ruza mashg‘ulotlarida, mustahkamlashda, o‘tilgan mavzuni so‘rashda, uyga vazifa berishda hamda amaliy mashg‘ulot natijalarini tahlil etishda foydalanish tavsiya etiladi.

Texnologiyani amalga oshirish tartibi:

- qatnashchilarga mavzuga oid bo‘lgan yakuniy xulosa yoki g‘oya taklif etiladi;
- har bir ishtirokchiga FSMU texnologiyasining bosqichlari yozilgan qog‘ozlarni tarqatiladi:

- ishtirokchilarning munosabatlari individual yoki guruhiy tartibda taqdimot qilinadi.

FSMU tahlili qatnashchilarda kasbiy-nazariy bilimlarni amaliy mashqlar va mavjud tajribalar asosida tezroq va muvaffaqiyatli o‘zlashtirilishiga asos bo‘ladi.

Namuna.

Fikr: “Polimarfizim obyektga yo‘naltirilgan dasturlashning asosiy tamoyillaridan biridir”.

Topshiriq: Mazkur fikrga nisbatan munosabatingizni FSMU orqali tahlil qiling.

“Assesment” metodi

Metodning maqsadi: mazkur metod ta’lim oluvchilarning bilim darajasini baholash, nazorat qilish, o‘zlashtirish ko‘rsatkichi va amaliy ko‘nikmalarini tekshirishga yo‘naltirilgan. Mazkur texnika orqali ta’lim oluvchilarning bilish faoliyati turli yo‘nalishlar (test, amaliy ko‘nikmalar, muammoli vaziyatlar mashqi, qiyosiy tahlil, simptomlarni aniqlash) bo‘yicha tashhis qilinadi va baholanadi.

Metodni amalga oshirish tartibi:

“Assesment” lardan ma’ruza mashg‘ulotlarida talabalarning yoki qatnashchilarning mavjud bilim darajasini o‘rganishda, yangi ma’lumotlarni bayon qilishda, seminar, amaliy mashg‘ulotlarda esa mavzu yoki ma’lumotlarni o‘zlashtirish darajasini baholash, shuningdek, o‘z-o‘zini baholash maqsadida individual shaklda foydalanish tavsiya etiladi. Shuningdek, o‘qituvchining ijodiy yondashuvi hamda o‘quv maqsadlaridan kelib chiqib, assesmentga qo‘shimcha topshiriqlarni kiritish mumkin.

Namuna. Har bir katakdagi to‘g‘ri javob 5 ball yoki 1-5 balgacha baholanishi mumkin.

Test

- 1.10^{-8} aniqlikda natijani chop etish qanday bajariladi?
- A. printf(“%.8f”,x)
- B. cout<<x
- C. cout<<(“%.8f”,x)

Qiyosiy tahlil

- Dasturiy mahsulotlardan foydalanish ko‘rsatkichlarini tahlil qiling?

Tushuncha tahlili

- STD qisqartmasini izohlang.

Amaliy ko‘nikma

- Grafik muhitida ishlash uchun qo‘shimcha kutubxona fayllarini o‘rnating va sozlang.

“Insert” metodi

Metodning maqsadi: Mazkur metod o‘quvchilarda yangi axborotlar tizimini qabul qilish va bilmlarni o‘zlashtirilishini yengillashtirish maqsadida qo‘llaniladi, shuningdek, bu metod o‘quvchilar uchun xotira mashqi vazifasini ham o‘taydi.

Metodni amalga oshirish tartibi:

- o‘qituvchi mashg‘ulotga qadar mavzuning asosiy tushunchalari mazmuni yoritilgan input-matnni tarqatma yoki taqdimot ko‘rinishida tayyorlaydi;
- yangi mavzu mohiyatini yorituvchi matn ta’lim oluvchilarga tarqatiladi yoki taqdimot ko‘rinishida namoyish etiladi;
- ta’lim oluvchilar individual tarzda matn bilan tanishib chiqib, o‘z shaxsiy qarashlarini maxsus belgilar orqali ifodalaydilar. Matn bilan ishlashda talabalar yoki qatnashchilarga quyidagi maxsus belgilardan foydalanish tavsiya etiladi:

Belgilar	1-matn	2-matn	3-matn
“V” – tanish ma’lumot.			
“?” – mazkur ma’lumotni tushunmadim, izoh kerak.			
“+” bu ma’lumot men uchun yangilik.			
“– ” bu fikr yoki mazkur ma’lumotga qarshiman?			

Belgilangan vaqt yakunlangach, ta’lim oluvchilar uchun notanish va tushunarsiz bo‘lgan ma’lumotlar o‘qituvchi tomonidan tahlil qilinib, izohlanadi, ularning mohiyati to‘liq yoritiladi. Savollarga javob beriladi va mashg‘ulot yakunlanadi.

“Tushunchalar tahlili” metodi

Metodning maqsadi: mazkur metod talabalar yoki qatnashchilarni mavzu buyicha tayanch tushunchalarni o‘zlashtirish darajasini aniqlash, o‘z bilimlarini mustaqil ravishda tekshirish, baholash, shuningdek, yangi mavzu buyicha dastlabki bilimlar darajasini tashhis qilish maqsadida qo‘llaniladi.

Metodni amalga oshirish tartibi:

- ishtirokchilar mashg‘ulot qoidalari bilan tanishtiriladi;
- o‘quvchilarga mavzuga yoki bobga tegishli bo‘lgan so‘zlar, tushunchalar nomi tushirilgan tarqatmalar beriladi (individual yoki guruhli tartibda);
- o‘quvchilar mazkur tushunchalar qanday ma’no anglatishi, qachon, qanday holatlarda qo‘llanilishi haqida yozma ma’lumot beradilar;
- belgilangan vaqt yakuniga yetgach o‘qituvchi berilgan tushunchalarning tugri va tuliq izohini uqib eshittiradi yoki slayd orqali namoyish etadi;
- har bir ishtirokchi berilgan tugri javoblar bilan uzining shaxsiy munosabatini taqqoslaydi, farqlarini aniqlaydi va o‘z bilim darajasini tekshirib, baholaydi.

Namuna: “Moduldagi tayanch tushunchalar tahlili”

Tushunchalar	Sizningcha bu tushuncha qanday ma’noni anglatadi?	Qo‘shimcha ma’lumot
#define	Makro deriktivalarni belgilash	a directive that defines a macro.
#include	Bir source fayl ichida boshqa bir faylag murojatni amalga oshirish mexanizmi	a mechanism for textual inclusion of one source file into another.

+=	add-and-assign operatori; masalan a+=b vazifazi jihatdan a=a+b bilan bir xil	add-and-assign operator; a+=b is roughly equivalent to a=a+b.
.c file	Dastur jodini o‘zida jamlovchi fayl	file containing definitions.
.cpp file	Dastur jodini o‘zida jamlovchi fayl	file containing definitions.
.h file	Sarlavha fayli	header file
address	Hotira manzili	a memory location

Izoh: Ikkinchi ustunchaga qatnashchilar tomonidan fikr bildiriladi. Mazkur tushunchalar haqida qo‘shimcha ma’lumot glossariyda keltirilgan.

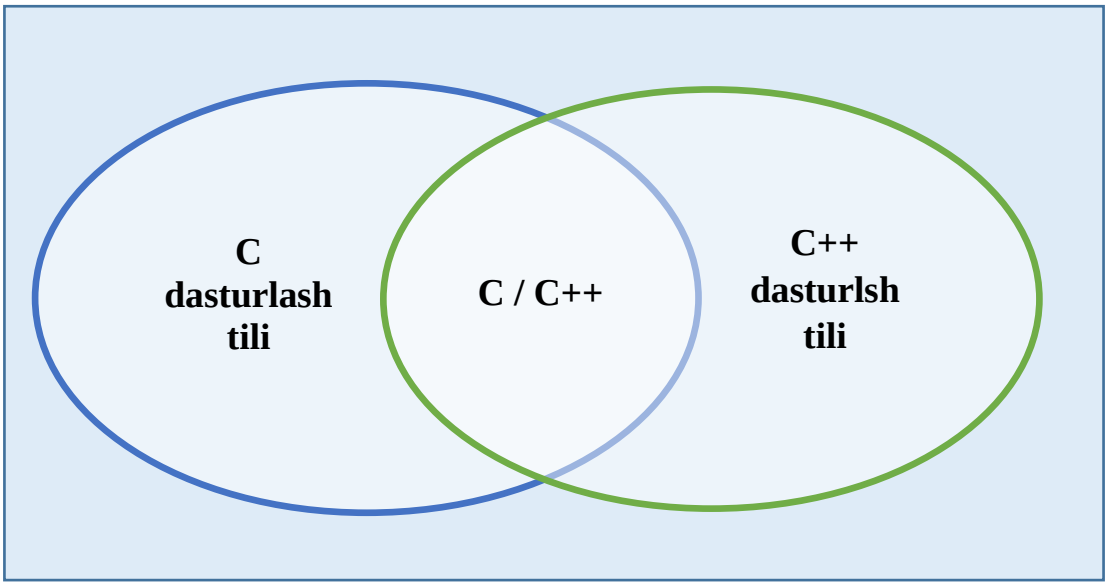
Venn Diagrammasi metodi

Metodning maqsadi: Bu metod grafik tasvir orqali o‘qitishni tashkil etish shakli bo‘lib, u ikkita o‘zaro kesishgan aylana tasviri orqali ifodalanadi. Mazkur metod turli tushunchalar, asoslar, tasavurlarning analiz va sintezini ikki aspekt orqali ko‘rib chiqish, ularning umumiy va farqlovchi jihatlarini aniqlash, taqqoslash imkonini beradi.

Metodni amalga oshirish tartibi:

- ishtirokchilar ikki kishidan iborat juftliklarga birlashtiriladilar va ularga ko‘rib chiqilayotgan tushuncha yoki asosning o‘ziga xos, farqli jihatlarini (yoki aksi) doiralar ichiga yozib chiqish taklif etiladi;
- navbatdagi bosqichda ishtirokchilar to‘rt kishidan iborat kichik guruhlarga birlashtiriladi va har bir juftlik o‘z tahlili bilan guruh a’zolarini tanishtiradilar;
- juftliklarning tahlili eshitilgach, ular birgalashib, ko‘rib chiqilayotgan muammo yohud tushunchalarning umumiy jihatlarini (yoki farqli) izlab topadilar, umumlashtiradilar va doirachalarning kesishgan qismiga yozadilar.

Namuna: Mobil ilova ma’lumotlarini saqlash turlari bo‘yicha



“Blis-o‘yin” metodi

Metodning maqsadi: o‘quvchilarda tezlik, axborotlar tizmini tahlil qilish, rejalashtirish, prognozlash ko‘nikmalarini shakllantirishdan iborat. Mazkur metodni baholash va mustahkamlash maksadida qo‘llash samarali natijalarni beradi.

Metodni amalga oshirish bosqichlari:

1. Dastlab ishtirokchilarga belgilangan mavzu yuzasidan tayyorlangan topshiriq, ya’ni tarqatma materiallarni alohida-alohida beriladi va ulardan materialni sinchiklab o‘rganish talab etiladi. Shundan so‘ng, ishtirokchilarga to‘g‘ri javoblar tarqatmadagi «yakka baho» kolonkasiga belgilash kerakligi tushuntiriladi. Bu bosqichda vazifa yakka tartibda bajariladi.

2. Navbatdagi bosqichda trener-o‘qituvchi ishtirokchilarga uch kishidan iborat kichik guruhlarga birlashtiradi va guruh a’zolarini o‘z fikrlari bilan guruhdoshlarini tanishtirib, bahslashib, bir-biriga ta’sir o‘tkazib, o‘z fikrlariga ishontirish, kelishgan holda bir to‘xtamga kelib, javoblarini «guruh bahosi» bo‘limiga raqamlar bilan belgilab chiqishni topshiradi. Bu vazifa uchun 15 daqiqa vaqt beriladi.

3. Barcha kichik guruhlar o‘z ishlarini tugatgach, to‘g‘ri harakatlar ketma-ketligi trener-o‘qituvchi tomonidan o‘qib eshittiriladi, va o‘quvchilardan bu javoblarni «to‘g‘ri javob» bo‘limiga yozish so‘raladi.

4. «To‘g‘ri javob» bo‘limida berilgan raqamlardan «yakka baho» bo‘limida berilgan raqamlar taqqoslanib, farq bulsa «0», mos kelsa «1» ball quyish so‘raladi. Shundan so‘ng «yakka xato» bo‘limidagi farqlar yuqoridan pastga qarab qo‘shib chiqilib, umumiy yig‘indi hisoblanadi.

5. Xuddi shu tartibda «to‘g‘ri javob» va «guruh bahosi» o‘rtasidagi farq chiqariladi va ballar «guruh xatosi» bo‘limiga yozib, yuqoridan pastga qarab qo‘shiladi va umumiy yig‘indi keltirib chiqariladi.

6. Trener-o‘qituvchi yakka va guruh xatolarini to‘plangan umumiy yig‘indi bo‘yicha alohida-alohida sharhlab beradi.

7. Ishtirokchilarga olgan baholariga qarab, ularning mavzu bo‘yicha o‘zlashtirish darajalari aniqlanadi.

«Dasturiy vositalarni o‘rnatish va sozlash» ketma-ketligini joylashtiring. O‘zingizni tekshirib ko‘ring!

Harakatlar mazmuni	Yakka baho	Yakka xato	To‘g‘ri javob	Guruh bahosi	Guruh xatosi
#include <iostream>			1		
using namespace std;			2		
int main() {			3		
Cout<< “Hello, World!” <<endl;			4		
return 0;			5		
}			6		

“Brifing” metodi

“Brifing”- (ing. briefing-qisqa) biror-bir masala yoki savolning muhokamasiga bag‘ishlangan qisqa press-konferensiya.

O‘tkazish bosqichlari:

Taqdimot qismi.

Muhokama jarayoni (savol-javoblar asosida).

Brifinglardan trening yakunlarini tahlil qilishda foydalanish mumkin. Shuningdek, amaliy o‘yinlarning bir shakli sifatida qatnashchilar bilan birga dolzarb mavzu yoki muammo muhokamasiga bag‘ishlangan brifinglar tashkil etish

mumkin bo‘ladi. Tinglovchilar yoki tinglovchilar tomonidan yaratilgan mobil ilovalarning taqdimotini o‘tkazishda ham foydalanish mumkin.

“Portfolio” metodi

“Portfolio” – (ital. portfolio-portfel, ingl.hujjatlar uchun papka) ta’limiy va kasbiy faoliyat natijalarini autentik baholashga xizmat qiluvchi zamonaviy ta’lim texnologiyalaridan hisoblanadi. Portfolio mutaxassisning saralangan o‘quv-metodik ishlari, kasbiy yutuqlari yig‘indisi sifatida aks etadi. Jumladan, talaba yoki tinglovchilarning modul yuzasidan o‘zlashtirish natijasini elektron portfoliolar orqali tekshirish mumkin bo‘ladi. Oliy ta’lim muassasalarida portfolioning quyidagi turlari mavjud:

Faoliyat turi	Ish shakli	
	Individual	Guruhiy
Ta’limiy faoliyat	Tinglovchilar portfoliosi, bitiruvchi, doktorant, tinglovchi portfoliosi va boshq.	Tinglovchilar guruhi, tinglovchilar guruhi portfoliosi va boshq.
Pedagogik faoliyat	O‘qituvchi portfoliosi, rahbar xodim portfoliosi	Kafedra, fakultet, markaz, OTM portfoliosi va boshq.

III. BO`LIM

NAZARIY MATERIALLAR

III. NAZARIY MATERIALLAR

1-ma’ruza. Kompyuterlar, insonlar va dasturlash

Reja:

1. Kirish.
2. Dasturiy ta’minot.
3. Insonlar.
4. Kompyuterlashtirilgan fanlar.
5. Kompyuterlar olamda.
6. Dasturchilarning yuksak orzulari.

Kalit so‘zlar: *kommunikatsiya, dasturiy ta’minot, tashxis, teskari aloqa, loyihalash, foydalanuvchi interfeysi, foydalanuvchi, aniqlik, Stereotip, buyurtmachi, dasturchi, samaradorlik.*

1.1. Kirish

Har bir o‘qitish jarayoni kabi, dasturlashni o‘qitish tovuq va tuxum dilemmasiga borib taqaladi. Biz ishni tezroq boshlashni istaymiz, ammo bir vaqtning o‘zida nima uchun aynan shu mavzularni tanlaganimizni tushuntirib berishni hoxlaymiz. Biz vaqt yo‘qotishni istamaymiz, shu bilan birga sizni shoshiltirishni ham hohlamaymiz.

Ushbu mavzu, biz dasturlashda muhim va qiziqarli deb hisoblaydigan, bizning qarashlarimizni ifodalaydi. Unda biz bir necha o‘n yillar davomida shug‘ullanayotgan ishimizning sabablari keltirilgan. Mavzu sizga bizning asosiy maqsadlarimizni va dasturchilar qanday bo‘lishi haqidagi tushunchani beradi. Ushbu mavzuda ayrim savollarga javob topishga xarakat qilamiz. Nima uchun dasturlash diqqatga sazovor mashg‘ulot hisoblanadi? Bizning sivilizasiyamizda dasturlash qanday ahamiyatga ega? Dasturchilar kelgusida mag‘rurlanishi mumkin bo‘lgan sohalar? Dasturiy ta’minotlarni ishlab chiqish, qo‘llash sohasida qanday

masalalar ustida bosh qotirilmagan? Dasturlash axborot texnolgiyalari, dasturiy ta’minotni ishlab chiqishda, kompyuter ilmlari sohasida qanday o‘ringa ega? Dasturchilar nimalar bilan shug‘ullanadi? Ular qanday mahoratga ega bo‘lmoqlari lozim?

Tinglovchilar uchun biror-bir g‘oya, uslub yoki darslikning ba’zi boblarini o‘rganishining asosiy sabablaridan biri imtihondan yaxshi baho olish bo‘lishi mumkin, ammo buning uchun yuqori maqsad ham bo‘lishi mumkin. Dasturiy ta’minot ishlab chiqarish sohasida ishlayotgan shaxslar uchun biror-bir g‘oya, uslub yoki darslikning ba’zi boblarini o‘rganishining asosiy sabablaridan biri sifatida oylik oshirishi, lavozimni ko‘tarish imkoniyatiga ega bo‘lgan boshliqning quvvatlashi bo‘lishi mumkin, ammo buning uchun bundan ham yuqoriroq maqsadlar ham bo‘lishi mumkin Agar bizning ishimiz olamni yaxshiroq qilsa, boshqa insonlarga yordam bersa, biz yaxshiroq ishlaymiz. Biz yillar davomida yechayotgan masalalarimiz uchun (professional martabasi shulardan tashkil topadi) g‘oyalar va oliy maqsadlar hayotiy muhim hisoblanadi.

Sivilizasiyamiz hayotiy faoliyati dasturiy ta’minot bilan bog‘liqdir. Dasturiy ta’minotni yaxshilash va uning tadbiqi uchun yangi sohalarni axtarib topish ko‘p insonlarning hayotining yaxshilanishiga olib keladi. Bunda dasturlash muhim ahamiyatga ega.

1.2. Dasturiy ta’minot

Yaxshi dasturiy ta’minot ko‘zga ko‘rinmaydi. Siz uni ushlab ko‘rishingiz, og‘irligini o‘lchashingiz yoki urishingiz mumkin emas. Dasturiy ta’minot — ma’lum kompyuterda bajariladigan dasturlar to‘plamidir. Ayrim hollarda biz ushbu kompyuterni ushlab ko‘rishimiz mumkin, ammo ko‘proq biz ushbu kompyuter tarkibida bo‘lgan qurilmalarni ko‘rishimiz mumkin, masalan, telefon, fotoapparat, avtomobilni. Biz faqat dasturiy ta’minot ishining natijasini qabul qilishimiz mumkin. Dunyoda nechta kompyuter bor? Balki, milliarddir. Ularning soni insonlar sonidan ko‘p. Xalqaro telekommunikasiya uyushmasi (International Telecommunication Union - ITU) ma’lumotiga ko‘ra 2004 yilda dunyoda 772

million shaxsiy kompyuterlar mavjud bo‘lgan, ammo ko‘pgina kompyuterlar ushbu kategoriyaga kiritilmagan.

Siz har kuni nechta kompyuterdan foydalanasiz? Masalan, mening avtomobilimda o‘ttizdan ortiq kompyuter o‘rnatilgan, mobil telefonda — ikkita, MP3-pleyerda — bitta va yana biri videokamerada. Bundan tashqari, menda yana noutbuk va stasionar kompyuter bor. Kontrollerning kondisioneri ham oddiy kompyuter vazifasini o‘taydi. Agar siz zamonaviy televizordan foydalansangiz, uning ichida hyech bo‘lmaganda bitta kompyuter topasiz. Bitta veb sahifadan boshqasiga o‘tish jarayonida siz telekommunikasiya tizimlari orqali, bir necha ming kompyuterlardan (telefon kommutatori, marshrutizator) tashkil topgan, yuzlab serverga ulanishingiz mumkin.

Kun davomida siz nechta kompyuterga bog‘liqsiz? Agar siz katta shaharda yashasangiz, taomga ega bo‘lishingiz uchun, kimdir rejalashtirish, trasportirovka, mahsulotlarni saqlash kabi ishlarni bajarishi lozim. Albatta, mahsulot tarqatish tarmog‘ini boshqarish, ixtiyoriy kommunikasiya tizimlari kabi, kompyuterlashtirilgan. Agar sizga uzoqdagi kompyuter bilan aloqa kerak bo‘lsa, unda trafik ham kompyuter orqali boshqariladi. Siz poyezdda sayohat qilishni ma’qul ko‘rasizmi? Ushbu poyezd ham kompyuterlashtirilgan bo‘ladi; ayrim poyezdlar xattoki mashinistsiz yuradi, bunda poyezddagi ayrim tizimlar ham ko‘plab kompyuterlardan tashkil topadi.

Yuqorida qayd etilgan har bir kompyuterda dasturiy ta’minot o‘rnatilgan. Har bir dasturiy ta’minot insonlar tomonidan yaratiladi. Kompyuter tomonidan bajarilgan dasturning har bir satri ma’lum ma’noga ega. Bunda biz yuzlab dasturlash tillarida yozilgan dasturlarning milliardlab satrlarini nazarda tutyapmiz. Bularning barchasi to‘g‘ri ishlashi uchun, insonlar ko‘p bilimlarini qo‘lladilar.

Biz takomillashtirishni istamagan biror bir qurilma yo‘q. Har bir qurilmani tezroq va sifatliroq ishlaydigan ko‘rinishga olib kelish, kegroq imkoniyatlar va quvvatlar bilan ta’minlamoq, uni chiroyliroq va arzonroq qilish mumkin. Ehtimol, buning uchun dasturlashdan foydalanishga to‘g‘ri kelar.

1.3. Insonlar

Kompyuterlar insonlar uchun va ular tomonidan yaratilgan. Kompyuter universal qurilma bo‘lib, undan keng ko‘lamdagi masalalarni yechish uchun foydalanish mumkin. Aynan shuning uchun ham dasturlar foyda keltiradi.

Nomdor kompyuter ilovalari haqida o‘ylab ko‘ring. Kompyuter qurilmalari, tizimlari dasturiy ta’minot fragmentini ishlab chiqishni o‘nlab, yuzlab, xattoki minglab insonlar ishtirokisiz tasavvur qilib bo‘lmaydi. Bunda dasturchilar, loyihachilar, testlovchilar, animatorlar, psixologlar, ma’murlar, loyiha menedjerlari, foydalanuvchi interfeysini yaratuvchilar, uskuna ta’minoti interfeysini yaratuvchilar, sifat bo‘yicha injenerlar, texnik topshiriqlarni ishlab chiquvchi mutaxassislar, dasturiy ta’minot ishlab chiqish bo‘yicha menedjerlar, dasturlar kutubxonasini tashkil qiluvchi mutaxassislar, xavfsizlik xizmati xodimlari kabilar qatnashadi.

Bunda yagona murakkablik shundan tashkil topgan-ki, yaxshi dasturiy ta’minot ishlab chiqarilishiga ta’sir etuchi barcha insonlar turli ta’limga egaligi, ularning turli qiziqishlari va odatlari mavjudligini e’tiborga olish lozim. Ushbu insonlarga bizning hayotimiz sifati, ayrim hollarda xatto hayotimiz ham bog‘liq. Hyech bir inson yuqorida keltirilgan barcha vazifalarni bajara olmaydi.

Biz hamma vaqt dasturchilar va dasturlash haqida so‘z yurityapmiz, ammo dasturlash umumiy tasvirning bir qismidir. Kema yoki mobil telefonni ishlab chiqaruvchi shaxslar o‘zlarini dasturchi deb hisoblanmaydi. Bundan tashqari, dasturlash dasturiy ta’minot ishlab chiqishning muhim qismi bo‘lsa ham, dasturiy ta’minot ishlab chiqish—bu faqtgina dasturlash emas..

Shunday qilib, dasturlash biz uchun nega kerak? Balki, mohir dasturchi bo‘lmagan holda, siz uni o‘zingizning tadqiqotlaringizda asosiy uskuna sifatida qo‘llarsiz. Balki, siz dasturlashni kasb qilib olgan holda, o‘z ishingizning bir qismiga aylantirishingiz mumkin bo‘lar. Xattoki, siz dasturlashni kasb qilib olgan holingizda ham, dasturlashdan boshqa ilmga ega bo‘lasiz.

Dasturlash — o‘z g‘oyalarini dastur ko‘rinishida ifodalash usulidir. Bu masalalar yechishga yordam beradi. Agar sizning g‘oyalaringiz bo‘lmasa,

yechilishi lozim bo‘lgan masalalar bo‘lmasa, dasturlash vaqtni bekor o‘tkazish demakdir.

1.4. Kompyuterlashtirilgan fanlar

Dasturlash, xattoki keng miqiyosda, kattorq ilmiy fanning qismi xisoblanadi. Biz uni dasturiy ta’minot bilan bog‘liq axborot texnologiyalari, kompyuter texnikasi, kompyuterlashtirilgan fanlar yoki boshqa ilmiy fanning bir qismi sifatida ko‘rishimiz mumkin. Dasturlash informatikada, texnikada, fizikada, biologiyada, medisinada, tarixda, adabiyotda va boshqa akademik sohalarda qo‘llanadigan qo‘shimcha texnologiyadir.

Dasturlash — amalda qo‘llash, tajriba asosida tahrirlash, tekshirish imkoniyatini yaratuvchi amaliy va fundamental masalalar yechilishini ifodalovchi asosiy uskunadir. Dasturlash — g‘oyalar va nazariya voqyelik bilan to‘qnashuvchi fandir; unda kompyuterlashtirilgan fanlar nazariy bo‘lmasdan eksperimental fanga aylanadi va olamga ta’sir etishni boshlaydi. Bunda qayd etish lozimki, dasturlash — amaliy va nazariy uslublarni amalga oshiruvchi vositalardir.

1.5. Kompyuterlar olamda

Har birimiz qachondir yoki qayerdadir kompyuterlar yoki dasturlar haqida eshitganmiz. Ko‘p odamlar kompyuter sifatida ekran va klaviaturali qutini tasavvur qilishadi. Bunday kompyuterlar, odatda, yozuv stolining tagida turadi va musiqani tinglash, elektron pochta, ma’lumot ayirboshlash, o‘yin o‘ynash uchun qo‘llanadi. Boshqa kompyuterlar, noutbuklar samoletlarda ma’lumotlar bazasini ko‘zdan kechiruvchi, kompyuter o‘yinlarini o‘ynovchi, videofilmlarni ko‘ruvchi biznesmenlar tomonidan ishlatiladi. Kompyuterlarning ko‘p soni umuman ko‘zimizga ko‘rinmaydi, ammo bizning sivilizasiyamizning mavjudligi aynan shularning ishi bilan bog‘liq. Ayrim kompyuterlar butun xonalarni egallaydi, boshqalarining o‘lchami tangadan ham kichik. Ko‘p kompyuterlar odamlar bilan

umuman, klaviatura, sichqoncha yoki boshqa qurilmalar yordamida amalga oshiriluvchi muloqotda bo‘lmaydi.

1.5.1. Ekran bilan yoki ekransiz

Kompyuterning ekran va klaviaturali quti sifatidagi tasavvur keng tarqalgan va turg‘un. Ammo, quyidagi ikki kompyuterni tasavvur qilaylik.

Ikkala kompyuterni bevosita ko‘rish mumkin. Bundan tashqari, ular turli kiritish chiqarish tizimli, bir modeldagi kompyuterlardir. Chapdagi qurilmada vaqt kichik ekranga chiqariladi, o‘ngdagisida esa, kichik elektr motor orqali boshqariluvchi, ananaviy siferblatga chiqariladi. Ularning kiritish-chiqarish tizimi, yuqori aniqlikdagi atom soat bilan sinxronizasiyani ta’minlovchi radiopriyemnik va to‘rtta tugmadan tashkil topgan. Ushbu ikki kompyuterni boshqaruvchi ko‘p dasturlar ular uchun umumiy.

1.5.2. Kemasozlik

Ushbu ikki suratda ulkan dizelli kema dvigateli va shunday dvigatel o‘rnatilishi mumkin bo‘lgan ulkan kema tasvirlangan.


Ularning qaysi qismida kompyuter va dasturiy ta’minot hal qiluvchi rol o‘ynashini ko‘rib chiqamiz.

Loyihalashtirish. O‘z-o‘zidan ma’lumki, kema ham, uning dvigateli ham kompyuterlar yordamida loyihalashtiriladi. Ularning qo‘llanish ro‘yhati deyarli cheksiz bo‘lib, unga arxitektura va injenerlik chizmalarini tayyorlash, umumiy hisobotlar, imorat va konstruksiyalarni ko‘z bilan chamalashlar kiritilgan.

Qurilish. Zamonaviy kompyuterlar kompyuterlashtirilgan. Kemani yig‘ish kompyuter yordamida rejalashtiriladi, yig‘ish jarayoni ham kompyuter boshqaruvida amalga oshiriladi. Payvandlash robotlar yordamida bajariladi. Xususan, zamonaviy ikki korpusli tankerlarni, korpuslar orasiga o‘ta oladigan kichik payvandchi-robotlarsiz qurilishini tasavvur qilish qiyin. Korpuslar oralig‘ida inson sig‘ishi mumkin bo‘lgan joyning o‘zi yo‘q. Kemalar uchun po‘lat plitalarni qirqish birinchilar qatorida kompyuterli loyihalashtirish va ishlab chiqish tizimi CAD/CAM (computer-aided design and computer-aided manufacture)ning qo‘llanishi orqali bajarildi.

1.5.3. Telekommunikasiyalar

Ushbu ikki suratda telefon kommutatori va (fotoapparat, MP3-pleyer, FM-radiopriyemnik i veb-brauzerli) telefon ifodalangan.

Ularning qaysi qismida kompyuter va dasturiy ta’minot hal qiluvchi rol o‘ynashini ko‘rib chiqamiz.

Tasavvur qiling, siz qo‘lingizga telefon olib, chaqiriq qildingiz, siz qo‘ng‘iroq qilayotgan shaxs javob beradi va sizlar suhbatni boshlaysizlar. Balki siz avtomatlashtirilgan ma’lumot tizimi bilan bog‘lanmoqchi bo‘ldingiz, yoki o‘rnatilgan fotoapparat yordamida olingan suratni jo‘natish, yoki matnli ma’lumotni jo‘natishni istaysiz. Shubhasiz, bunday telefon kompyuterdir. Agar telefon ekranga ega bo‘lsa va oddiy telefonlarga nisbbatan ko‘proq, masalan, veb-brauzer funksiyalariga ega bo‘lsa, bu tushunarliroq bo‘ladi. Aslida, bunday telefonlar bir necha kompyuterga ega: bittasi ekranni boshqaradi, boshqasi telefon stansiyasi bilan aloqani o‘rnatadi, uchinchisi yana nimadir qiladi.

Chap tarafdagi suratda tasvirlangan imorat Nyu-York Uoll-stritidagi amerika fond birjasining sotuv maydonini ko‘rsatgan holda, o‘ng tomondagi xarita Internetning bir qismini namoyish qiladi.

Ko‘rganingizdek, biz raqamli suratlarni yaxshi ko‘ramiz va axborotni tasavvur qilish imkonini beruvchi kompyuterlarni maxsus xaritalarni tasvirlash uchun qo‘llaymiz.

1.5.4. Tibbiyot

Quyidagi ikki suratda kompyuterli aksial tomografiyaning skaneri CAT va kompyuter xirurgiyasi uchun operasiya xonasi (robotoxirurgiya deb ham ataladi) namoyish qilingan.

Ularning qaysi qismida kompyuter va dasturiy ta’minot hal qiluvchi rol o‘ynashini ko‘rib chiqamiz. Skanerlar— bu asosan kompyuterlardir, kompyuterlar ulardan tarqalgan impulslarni boshqaradi, ammo olingan axborot, murakkab algoritmlar tomonidan ishlov berilib, tananing mos qismini tushunarli uch o‘lchamli tasvirga aylantirmagunga qadar, tartibsizliklar kelib chiqadi. Kompyuter yordamida jarroxlik operasiyasini o‘tkazish uchun biz yanada olg‘a siljishimiz lozim. Shifokorlar o‘rtasida qaysi yangi asbob foydaliroq ekanligi yuzasidan baxslashuvlar bo‘lmoqda. Kompyuterli aksial tomografiyaning skanerimi? Magnitorezonans tomografiyali skanermi? Qonni avtomatik tahlil qiluvchi apparatmi? Yuqori sig‘imli ultratovush qurilmalarmi? Shaxsiy axborot qurilmalarimi? Ushbu “musobaqa”da bemor holati haqidagi yozuvlarga uzluksiz murojaatni ta’minlovchi uskunalar “g‘olib” bo‘ldi.

Bemor kasallik tarixini (u boshidan o‘tkazgan kasalliklar, tibbiy yordam turlari, allergik reaksiyalar, nasliy muammolar, sog‘liqning umumiy holati, joriy davolash va x.k.) bilish diagnostikani osonlashtiradi, xatolar ehtimolligini kamaytiradi.

1.5.5. Axborot

Quyidagi ikki suratda oddiy shaxsiy kompyuterlar va serverlar guruxi tasvirlangan.

Biz bejiz diqqatimizni apparat qurilmalariga qaratmadik: dasturiy ta’minotni ko‘rish, ushlash, eshitish mumkin emas.Dastur suratini ko‘rsatish imkoni bo‘lmagani uchun, biz ushbu dasturni bajaradigan uskunani namoyish qilyapmiz. Lekin, dasturiy ta’minotning ko‘p turi axborot bilan ishlaydi. Shunday qilib, oddiy dasturiy ta’minotni bajaruvchi oddiy kompyuterlarning oddiy qo‘llanishini ko‘rib chiqamiz.

Kompyuterlarning bunday qo‘llanishi axborotga ishlov berish deb ataladi. Ushbu katta xajmdagi berilganlarga qaratilgan bo‘lib, axborotni tashkil etish va uzatish, hamda berilganlarning ulkan massivlarini taqdim etish jarayonlarida ma’lum muammolarni keltirib chiqaradi: foydalanuvchi interfeysi ma’lumotlarga ishlov berishda muhim aspekt hisoblanadi

1.5.6. Yuqoridan ko‘rinish

Aytishadiki, paleontologlar kichik suyakni o‘rganish natijasida, dinazavr ko‘rinishini tiklashlari, uning hayot tarzini va yashash muhitini tavsiflashlari

mumkin ekan. Balki, bu bo‘rttirma bo‘lib tuyulishi mumkindir, ammo ayrim hollarda oddiy artefaktni o‘rganib, u qanday oqbatga olib kelishi ustida fikrlash foydali bo‘ladi. NASA marsyuruvchisi tomonidan olingan mars landshafti suratiga e’tiboringizni qarating.

Agar kosmmonavtika bilan shug‘ullanishni istasangiz, unda yaxshi dasturchi bo‘lishingiz lozim. Ko‘psonli kosmik dasturlarda, fizikani, matematikani, elektrotexnikani, mexanikani, medisina texnikasini, ya’ni Kosmos tadqiqoti uchun zarur bo‘lgan barcha ilmiy fanlarni biluvchi ko‘psonli dasturchilar qatnashadi. To‘rt yil davomida (uch oyga mo‘ljallangan) — bizning sivilizasiyamiz katta yutuqlaridan biridir.

Bundan tashqari, ko‘psonli dasturchilar tasvirlarga ishlov berish (marsoxodddan kelayotgan suratlarga ishlov berish, shular jumlasidan), suratlarni tahrirlash va xarakatga keltirish (veb tarmoqda marsda yashovchilar tasvirlangan marsdan yuborilgan suratlarning variantlari tarqalgan) bilan ham shug‘ullanadilar [1].

1.6. Dasturchilarning yuksak orzulari

Biz dasturlarimizdan nimalarni kutmoqdamiz? Xususan, biror bir dasturdan niima istaymiz? Dasturimiz to‘g‘ri va ishonchli ishlashini istaymiz Agar dastur undan talab qilinayotgan xarakatlarni bajarmasa, yoki ishonchli bo‘lmasa, unda yaxshi holatda bu jiddiy nyuans, yomonida esa — xatarlilik. Shu bilan birga, biz

[1] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

dasturimiz yaxshi loyihalashtirlgan bo‘lishini, ya’ni bizning talablarimizni qanoatlantirishini istaymiz; ammo xaqiqatada, dastur to‘g‘ri ishlayaptimi, noto‘g‘rimi, ahamiyatga molik emas; ushbuni amalga oshirilayotgan uslub tashvishga soladi.Bundan tashqari biz dasturimiz tejamkor bo‘lishini istaymiz, ehtimol men rolls-roysda yurishni, korporativ samoletda uchishni istarman, ammo men hali millioner emasman, ushbu xuzur-halovatnining narxini e’tiborga olishim lozim.

Dasturiy ta’minotning aynan shu aspektlari (uskunalar, tizimlar) dasturchi bo‘lmagan shaxslar tomonidan munosib baholanishi mumkin.

Dastur ishlab chiqish jarayoni to‘rtta bosqichga jaratilishi mumkin:

- *Tahlil*. Masalaning mohiyati nimadan iborat? Foydalanuvchi nimani istaydi? Foydalanuvchi uchun nimalar zarur? Bizga Qanday ishonarlilik bizga zarur?
- *Loyihalashtirish*. Maslani qanday yechish lozim? Tizim qanday tuzilmaga ega bo‘lishi kerak? U qanday qismlardan tashkil topishi lozim? Ushbu qismlar qanday muloqotda bo‘ladilar? Tizim va foydalanuvchi o‘rtasidagi muloqot qanday amalaga oshiriladi?
- *Dasturlash*. Masala (loyiha) yechimini dastur ko‘rinida tasvirlaymiz. Barcha cheklanishlarni(vaqt, xajm, moliya, ishonchlilik) e’tiborga olib, dasturni yozamiz. Dastur to‘g‘ri ishlayotganiga, qulayligiga ishonch tug‘diramiz.
- *Testlash*. Hamma ko‘zda tutilgan hollarda ishlayotganiga dastur to‘g‘ri ishlatganiga ishonch bildiramiz.

Ko‘p hollarda dasturlashni va testlashni tatbiq etish deb ataladi. Albatta, ushbu to‘rt qismga ajratish shartlidir. Ushbu mavzular bo‘yicha ko‘p kitoblar yozilgan, bu mavzularning o‘zaro bog‘liqligi to‘0risida esa yanada ko‘prog‘i yozilgan. Odatda, biz tahlildan boshlaymiz, ammo testlash jarayonining natijasi dasturlashga ta’sir qiladi; dasturlash jarayonidagi muammolar, loyihalash jaryonidagi muammolar oqibatidir; o‘z navbatida v loyihalash jarayoni tahlil

jarayonidagi xatolarni aniqlashi mumkin. Haqiqatdan, tizimning ishlashi, odatda, darrov tahlilning zaif tomonini aniqlaydi.

Nazorat savollari

1. Dasturiy ta’minot nima?
2. Dasturiy ta’minotning muhimligi nimada?
3. Dasturiy ta’minotning zaruriyligi nimada ko‘rinadi?
4. Dasturiy ta’minot noto‘g‘ri ishlasa, qanday xodisa ro‘y beradi? Misollar keltiring.
5. Dasturiy ta’minot qanday sohalarda muhim ahamiyatga ega? Misollar keltiring.
6. Qanday faoliyat turlari dasturiy ta’minot ishlab chiqish bilan bog‘liq? Misollar keltiring.
7. Kompyuter fanlari va dasturlash o‘rtasida qanday farq bor?
8. Kemalarni loyihalash, konstruktorlash, qo‘llash jarayonlarining qay birida dasturiy ta’minot qo‘llanadi?
9. Serverlar guruxi nima?
10. Tarmoqda qanday so‘rovlarni jo‘natasiz? Misollar keltiring.
11. Dasturiy ta’minot ilmiy izlanishlarda qanday qo‘llanadi? Misollar keltiring.
12. Dasturiy ta’minot tibbiyotda qanday qo‘llanadi? Misollar keltiring.
13. Dasturiy ta’minot ko‘ngil ochar industriyada qanday qo‘llanadi? Misollar keltiring.
14. Yaxshi dasturiy ta’minot qanday xususiyatlarga ega bo‘lmog‘i lozim?
15. Dasturiy ta’minot ishlab chiqaruvchi shaxs qanday ko‘rinishga ega?
16. Dasturiy ta’minot ishlab chiqish bosqichlarini aytib bering.

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

2-ma’ruza. Dasturlash asoslari

Reja:

2.1 Hello, World!

2.2 Obyektlar, qiymat va toifalar

2.3 Hisoblash

2.4 Xatoliklar

Kalit so‘zlar: *toifalar, xatolik, sintaktik xatolik, dastur ishlashi davomidagi xatolik, testlash, kompilyatsiya paytidagi xatolik, talab, mantiqiy xato, turlar xatosi*.

2.1 Hello, World!

Dasturlar. Kompyuterni biron bir amalni bajprishga majburlash uchun, siz (yoki boshqalar) unga nima xoxlayotganingizni aniq, batafsil aytishingiz kerak.

Bundan tashqari, biz o‘zimiz bajarishimiz kerak bo‘lgan vazifa tavsifini olamiz, masalan, "yaqin oradagi kinoteatrga qanday borish mumkin" yoki "to‘lqinli pechda go‘shtni qanday qovurish mumkin". bunday tavsiflar va dasturlar orasidagi farq aniqlik darajasida aniqlanadi: insonlar sog‘lom aql bilan qo‘llanmani noaniqligini aniqlashga harakat qiladilar, kompyuter bunday qila olmaydi. Masalan, "yo‘lak bo‘ylab o‘nga, zinadan yuqoriga, so‘ngra chapga" - yuqori qavatdagi yuvinish xonasini topish imkonini beruvchi aniq qo‘llanma. Biroq, agar siz bunday sodda qo‘llanmaga qarasangiz, u holda ular grammatik noaniqligi va to‘liq emasligini ko‘rish mumkin. Masalan, siz stol atrofida o‘tiribsiz va yuvinish xonasiga qanday o‘tishni so‘radingiz. Sizga javob beruvchi, o‘rningizdan turishingizni, uni aylanib o‘tishingizni va boshqalarni aytishi shart emas. Yana sizga hyech kim sanchqini stolga qo‘yishingiz, zanadan ko‘tarilayotganda chiroqni yoqishingiz kerakligini, yuvinish xonasiga kirish uchun eshikni ochish kerakligini maslahat bermaydi [2].

[2] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

Qarama-qarshi holatda bunga kompyuterning aqli yetmaydi. Unga barchasini aniq va batafsil tavsiflash kerak. Kompyuterga qo‘llanmani batafsil tavsiflash uchun, o‘ziga xos grammatikaga ega bo‘lgan aniq belgilangan til hamda biz bajarishni xoxlayotgan faoliyatlarni barcha ko‘rinishlari uchun yaxshi aniqlikdagi lug‘at kerak bo‘ladi. Bunday til dasturlash tili va ko‘p qamrovli masalalarni yechish uchun ishlab chiqilgan - C++ dasturlash tili deb nomlanadi.

Kompyuterlar, dasturlar va dasturlash bo‘yicha falsafiy qarashlar 1-maruzada kengroq yoritib berilgan. Bu yerda biz juda oddiy dasturdan boshlanadigan kodni hamda uning bajarilishi uchun kerak bo‘ladigan bir qancha usullar va qurilmalarni ko‘rib chiqamiz.

Birinchi klassik dastur. Birinchi klassik dasturlarni variantlarini keltiramiz. U ekranga Hello, World! xabarini chiqaradi.

```
// Bu dastur ekranga "Hello, World! xabarini chiqaradi"
#include "std_lib_facilities.h"
intmain() // C++ da dasturlar main funksiyasidan boshlanadi
{
cout<< "Hello, World!\n"; // "Hello, World!" ni chiqarish
return 0;
}
```

Kompyuter bajarishi lozim bo‘lgan bu buyruqlar to‘plami, pazandalik resepti yoki yangi o‘yinchoqni yig‘ish bo‘yicha qo‘llanmani eslatadi. Eng boshidan boshlab, dasturning har bir satrini ko‘rib chiqamiz:

```
cout<<"Hello, World!\n";// "Hello, World!" chiqarish
```

Aynan mana shu satr xabarni ekranga chiqaradi. U yangi satrga o‘tuvchi belgi bilan Hello, World! belgilarini chop etadi; aks holda, Hello, World! belgisini chiqargandan so‘ng kursor yangi satrning boshiga joylashtiriladi. Kursor - bu keyingi belgi qayerda chiqishini ko‘rsatib turuvchi katta bo‘lmagan yonib o‘chib turuvchi belgi yoki satr.

C++ tilida satrli literallar qo‘shtirnoq (") bilan belgilanadi; ya’ni "Hello, Word!\n" — bu belgilar satri. \n belgi - yangi satrga o‘tishni bildiruvchi maxsus belgi. cout standart chiqarish oqimiga tegishli. "cout oqimida chiquvchilar" << chiqarish operatori yordamida ekranda aks etadi. cout “see-out” kabi talaffuz qilinadi, lekin “character putstream” ("belgilarni chiqarish oqimi") qisqartmasi hisoblanadi. Dasturlashda qisqartmalar yetarlicha keng tarqalgan. Albatta, qisqartmalar birinchi marta eslab qolish uchun noqulay ko‘rinishi mumkin, lekin o‘rganib qolgach ulardan voz kecha olmaysiz, ya’ni ular qisqa va boshqarib bo‘ladigan dasturlarni yaratishga imkon beradi.

Satr oxiri

```
// "Hello, World!" chiqarish
```

izoh hisoblanadi. // (ya’ni, ikkita egri chiziqdan keyin (/)) belgidan keyin yozilgan barchasi izoh hisoblanadi. U kompilyator tomonidan inkor qilinadi va dasturni o‘quvchi dasturchilar uchun mo‘ljallangan. Bu holatda biz satrning birinchi qismi nimani anglatishini sizga xabar qilish uchun izohdan foydalandik.

Izohlar insonlar uchun dastur matnida ifodalashning iloji bo‘lmagan foydali ma’lumotlarni qamrab oladi va dastur manzilini tasvirlaydi. Umuman olganda, izoh o‘zingiz uchun ham foydali bo‘lishi mumkin, ya’ni yaratgan dasturingizga xafta yoki yillar o‘tib murojat qilganingizda nima uchun yaratilganini tezda anglab olishga yordam beradi. O‘zingizni dasturlaringizni yaxshi xujjatlashtirishga harakat qiling.

Dastur ikkita auditoriya uchun yoziladi. Albatta, biz ularni bajaruvchi kompyuterlar uchun dastur yozamiz. Biroq biz kodni o‘qish va modifikasiyalashga uzoq yillar sarflaymiz. Shunday qilib, dastur uchun ikkinchi auditoriya bo‘lib

boshqa dasturchilar hisoblanadi. Shuning uchun dasturning yaratilishini insonlar o‘rtasidagi muloqot shakli deb sanash mumkin. Albatta, insonlarni o‘z dasturlarini birinchi o‘quvchilari deb hisoblash maqsadga muvofiq: agar ular qiyinchilik bilan o‘z yozganlarini tushunishsa, u holda dastur qachonlardir to‘g‘ri bo‘lishi dargumon. Shu sababli, kod o‘qish uchun mo‘ljallanganligini esdan chiqarmaslik kerak - dastur onson o‘qilishi uchun barcha imkoniyatlarni qilish kerak. Istalgan holatda izohlar faqat insonlar uchun kerak, kompyuter ularni inkor qiladi.

Dasturning birinchi satri - bu o‘quvchilarga dastur nima qilishi kerakligi haqida xabar beradigan toifaviy izoh.

```
// Bu dastur ekranga "Hello, World!" xabarini chiqaradi.
```

Bu izohlarning foydali tomoni shundaki, bunda dastur kodi bo‘yicha dastur nima qilayotganini tushunish mumkin, lekin biz aynan nima hohlayotganimizni aniqlash mumkin emas. Bundan tashqari, kodning o‘zidan tashqari, biz izohlarda dasturning maqsadini qisqacha tushuntirishimiz mumkin. Ko‘pincha bunday izohlar dasturning birinchi satrida joylashgan bo‘ladi. Boshqa narsalar orasida, ular biz nima qilmoqchi ekanligimizni eslatib o‘tadi.

Satr

```
#include "std_lib_facilities.h"
```

o‘zi bilan #include direktivasini akslantiradi. U std_lib_facilities.h faylida tasvirlanganlarni imkoniyatlarini "faollashtirish" uchun kompyuterni majburlaydi. Bu fayl C++ (C++ tilining standart kutubxonasi) tilining barcha amalga oshirishlarida ko‘zda tutilgan imkoniyatlaridan foydalanishni osonlashtiradi.

std_lib_facilities.h faylning imkoniyatlari berilgan dastur uchun shunda hisoblanadiki, uning yordami bilan biz C++ tilining standart kiritish-chiqarish vositalaridan foydalanish imkoniyatiga ega bo‘lamiz. Bu yerda biz faqatgina cout standart chiqarish potoki va << chiqarish operatoridan foydalanamiz. #include direktivasi yordamida dasturga kiruvchi fayl odatda .h kengaytmasiga ega bo‘ladi

va sarlavha (header) yoki sarlavha fayli (header file) deb nomlanadi. Sarlavha biz dasturimizda foydalanadigan cout kabi atamalarni aniqlashni o‘z ichiga oladi.

Dastur bajarilishi boshlanadigan nuqtani dastur qanday aniqlaydi? U main nomli funksiyani ko‘rib chiqadi va uni qo‘llanmalarini bajarishni boshlaydi. Bizning “Hello, World!” dasturimizda main funksiyasi quyidagicha ko‘rinadi:

```
int main() // C++ da dasturlash main funksiyasi yordamida amalga oshiriladi
{
cout << "Hello, World!\n"; // "Hello, World!" chiqarish
return 0;
}
```

Bajarishning boshlang‘ich nuqtasini aniqlash uchun, C++ tilidagi har bir dastur main nomli funksiyadan tashkil topgan bo‘lishi zarur. Bu funksiya to‘rtta qismdan iborat.

1. Qiymat qaytaruvchi toifa, bu funksiyada - int toifa (ya’ni, butun son), funksiya chaqirish nuqtasida qanday natija qaytarishini aniqlaydi (agar u qandaydir qiymat qaytarsa). int so‘zi C++ tilida zahiralangan hisoblanadi (kalit so‘z), shuning uchun uni boshqa bir narsaning nomi sifatida foydalanish mumkin emas.

2. Nom, main berilgan holatda.

3. Parametrlar ro‘yxati, aylana qavsda berilgan; berilgan holatda parametrlar ro‘yxati bo‘sh.

4. Funksiya tanasi, figurali qavsda berilgan va funksiya bajarishi kerak bo‘lgan faoliyatlar ro‘yxati.

Bundan kelib chiqadiki, C++ tilidagi kichik dastur quyidagicha ko‘rinadi:

```
int main() { }
```

Bu dastur hech qanday amal bajarmaganligi uchun undan foyda kam. “Hello, World!” dasturining main funksiyasining tanasi ikkita qo‘llanmadan iborat:

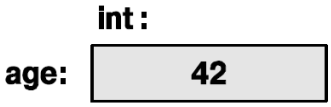
```
    cout << "Hello, World!\n"; // "Hello, World!"
chiqish

    return 0;
```

Birinchidan, u ekranga Hello, World! satrini chiqaradi, so‘ngra chaqirish nuqtasiga 0 (nol) qiymat qaytaradi. Qachonki main() funksiyasi tizim sifatida chaqirilsa, biz qiymat qaytarmasdan foydalanamiz. Lekin ayrim tizimlarda (Unix/Linux) bu qiymatdan dasturni muvaffaqiyatli bajarilishini tekshirish uchun foydalanish mumkin. main() funksiyasidagi qaytariluvchi Nol (0), dastur muvaffaqiyatli bajarilganini bildiradi [3].

2.2 Obyektlar, qiymat va toifalar

“Hello world” dasturi faqatgina ekranga yozuv chiqaradi holos. U hechnima o‘qimaydi, ya’ni, foydalanuvchi hechnima kiritmaydi. Bu juda zerikarli. Haqiqiy dasturlar odatda biz unga kiritgan ma’lumotlarga qarab qandaydir hisob kitoblar o‘tkazadi.

Ma’lumotlarni o‘qib olish uchun bu ma’lumotlarni saqlashga joy bo‘lishi kerak, boshqacha qilib aytganda, o‘qib olingan ma’lumotni yozish uchun bizga kompyuter hotirasidan joy kerak. Bu joyni biz Obyekt deb ataymiz. Obyekt – bu ma’lum bir tur (bu joyda saqlash mumkin bo‘lgan ma’lumot turi) ga ega bo‘lgan hotiradagi joy. Nom berilgan obyekt esa o‘zgaruvchi deyiladi. Masalan, satrlar *string* turiga ega bo‘lgan o‘zgaruvchilarga saqlanadi, butun sonlar esa – *int* turiga ega bo‘lgan o‘zgaruvchilarga saqlanadi. Obyektni uning ichiga ma’lum bir turdagi ma’lumotni saqlash mumkin bo‘lgan “quti” deb tasavvur qilishimiz mumkin [4].



[3] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

[4] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

Masalan, rasmda *int* turiga mansub, nomga ega bo‘lgan va 42 butun soni o‘zida saqlagan obyekt tasvirlangan. Satrni turli o‘zgaruchilarni kiritish qurilmasidan o‘qib olib quyidagicha ekranga chop etishimiz mumkin:

```
// ismni o‘qish va yozish

#include "std_lib_facilities.h"

int main()

{

cout << "Iltimos, ismingizni kiriting (keyin 'enter' tugmasini bosing):\n";

string first_name; // first_name — bu string turli o‘zgaruvchi

cin >> first_name; // first_name o‘zgaruvchiga belgilarni o‘qib olamiz

cout << "Hello, " << first_name << "!\n";

}
```

#include direktivasi bizning barcha dasturlarimizda ishlatilgani uchun, chalkashishlardan qochish maqsadida biz buni o‘rganishni ortga suramiz. Shu kabi ba’zida biz *main()* yoki boshqa bir funksiya ichiga yozilganda ishlaydigan kodlarni keltirib o‘tamiz.

```
cout << "Iltimos, ismingizni kiriting (keyin 'enter' tugmasini bosing):\n";
```

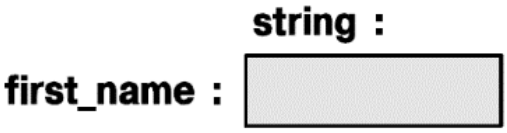
Biz sizni bu kodni testlash uchun to‘liq dasturga qanday qo‘shishni bilasiz deb hisoblaymiz.

main() funksiyasining birinchi qatori foydalanuvchiga ismini kiritishni taklif qiluvchi habar chiqaradi. Keyingi qatorlarda string turli *first_name* o‘zgaruvchi e’lon qilindi, ekrandan shu o‘zgaruvchiga ma’lumot o‘qib olindi va ekranga *Hello*

so‘zidan so‘ng *first_name* o‘zgaruvchining qiymati chiqarildi. Shu qatorlarni birin ketin ko‘rib chiqamiz.

```
string first_name; // first_name — bu string turga ega bo‘lgan o‘zgaruvchi
```

Bu qatorda kompyuter hotirasidan belgilarni saqlash uchun joy ajratilmoqda va unga *first_name* nomi berilmoqda.

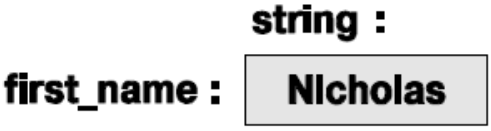


Xotiraan joy ajratish va unga nom berish jarayoni *e’lon qilish* deyiladi.

Keyingi qatorda kiritish qurilmasidan (klaviaturadan) o‘zgaruvchiga ma’lumot o‘qib olinmoqda:

```
cin >> first_name; // first_name o‘zgaruvchisiga belgilarni o‘qib olamiz
```

cin (“si-in” singari o‘qiladi, ingilizcha character input so‘zlari qisqartmasidir) nomi standart kutibhonada e’lon qilingan standart oqim kirituviga tegishli. Keyinigi >> (kirituv) operatorining operandi kirituv natijasi saqlanadigan joyni ko‘rsatadi. Demak, agar biz Nicholas ismini kiritib yangi qatorga o‘tganimizda (ya’ni enter tugmasini bosganimizda) *“Nicholas”* satri *first_name* o‘zgaruvchisining qiymatiga aylanadi.



Yangi qatorga o‘tish kompyuterning e’tiborini jalb yetish uchun muhim. Yangi qatorga o‘tilmagunga qadar (enter tugmasi bosilmagunga qadar) kompyuter

shunchaki belgilarni to‘plab boradi. Bu kutish bizga ba’zi belgilarni o‘chirish yoki boshqa belgilar bilan almashtirish imkonini beradi.

first_name o‘zgaruvchisiga qiymatni kiritganimizdan so‘ng uni kelgusida ishlatishimiz mumkin bo‘ladi.

```
cout << "Hello, " << first_name << "!\n";
```

Bu qator Hello, so‘zi va undan so‘ng Nicholas ismini (*first_name* o‘zgaruvchisining qiymati), ohirida undov belgisi (!) va yangi qatorga o‘tish belgisi (‘*n*’)ni chiqaradi

```
Hello, Nicholas!
```

Agar biz takrorlanish va ko‘p matn yozishni yoqtirganimizda yuqoridagi kodni quyidagicha yozishimiz mumkin:

```
cout << "Hello, ";

cout << first_name;

cout << "!\n";
```

E’tibor bering, Hello, so‘zini bir qo‘shtirnoqda chiqardik *first_name* o‘zgaruvchisini bo‘lsa qo‘shtirnoqlarsiz yozdik. Qo‘shtirnoqlar literal satrlar bilan ishlash uchun zarur, agar satr qo‘shtirnoqsiz yozilgan bo‘lsa demak u biror bir nomga (o‘zgaruvchiga) murojaat qilgan bo‘lamiz.

```
cout << "Ism" << " – " << first_name;
```

Bu yerda “Ism” uchta belgidan iborat satrni tashkil qiladi, *first_name* bo‘lsa ekranga *first_name* o‘zgaruvchisining qiymatini (bizning holatda Nicholas) chiqaradi. Demak natija quyidagicha ko‘rinishga ega:

```
Ism – Nicolas
```

O‘zgaruvchilar. Yuqorida keltirilgan misolda ko‘rsatilganidek kompyuter hotirasida ma’lumot saqlash imkinoyitisiz kompyuterda hech qanday qiziqarli ish qilib bo‘lmaydi. Ma’lumotlar saqlanadigan joyini obyekt deb ataymiz. Obyektga murojaat qilish uchun obyekt nomini bilish kerak. Nomlangan obyekt o‘zgaruvchi

deyiladi va unda qanday ma’lumotni saqlash mumkinligini aniqlovchi (masalan *int* turidagi obyektda butun sonlarni saqlash mumkin *string* turidagi obyektga ‘*Hello world*’ singari satrlarni saqlash mumkin) aniq bir turga ega bo‘ladi. (masalan, *int* yoki *string*), bundan tashqari ular ustida amallar bajarish mumkin (masalan *int* turidagi obyekt ustida * operatori orqali ko‘paytirish amalini bajarish mumkin, *string* turidagi obyektlarni esa <= operatori orqali taqqoslash mumkin). O‘zgaruvchilarga yozilgan ma’lumotlar qiymat deyiladi. O‘zgaruvchini aniqlash instruktsiyasi e’lon qilish deyiladi va o‘zgaruvchi e’lon qilinayotgan paytda unga boshlang‘ich qiymat berib ketsa bo‘ladi. Quyidagi misolni ko‘rib chiqamiz:

```
string name = "Annemarie";
int number_of_steps = 39;
```

Bu o‘zgaruvchilarni quyidagicha tasavvur qilish mumkin:

Biz o‘zgaruvchiga to‘g‘ri kelmaydigan turdagi ma’lumotni yoza olmaymiz.

```
string name2 = 39; // xato: 39 — satr emas
int number_of_steps = "Annemarie"; // xato:
"Annemarie" —
                                   // butun son emas
```

Kompilyator har bir o‘zgaruvchining turini saqlab qoladi va o‘zgaruvchini uning turiga mos ravishda ishlatishingizga yo‘l qo‘yadi.

C++ tilida juda ko‘plab turlar tanlovi mavjud. Lekin ularning atiga 5 tasidan foydalangan holda foydali dasturlar tuzish mumkin.

```
int number_of_steps = 39; // int — butun sonlar
uchun
```

```
    double flying_time = 3.5; // double – haqiqiy sonlar uchun
    char decimal_point = '.'; // char – belgilar uchun
    string name = "Annemarie"; // string – satrlar uchun
    bool tap_on = true;  // bool – mantiqiy o‘zgaruvchilar uchun
```

E’tibor bering, barcha o‘zgaruvchilar o‘zining yozilish uslubiga ega:

```
    39  //  int: butun son
    3.5 //  double: xaqiqiy son
    '.' //  char: bittalik tirnoqcha ichida yagona belgi
    "Annemarie"  //  string: qo‘shtirnoq ichida yozilgan belgilar to‘plami
    true//  bool: yoki rost (true), yoki yolg‘on (false) qiymatga ega
```

Boshqacha qilib aytganda, raqamlar ketma ketligi (masalan 1234, 2 yoki 973) butun sonni bildiradi, bittalik tirnoqcha ichidagi belgi (masalan, ‘1’, ‘@’ yoki ‘x’) belgini bildiradi, xaqiqiy sonlar esa (masalan, 123.123, 0.12, .98) haqiqiy sonlarni bildiradi, qo‘shtirnoq ichidagi belgilar ketma ketligi esa (masalan, “123141”, “Howdy!” yoki “Annemarie”), satrni bildiradi. Batafsil ma’lumot ilova qilingan adabiyotlarda berilgan.

Kiritish va tur. Kiritish operatori ma’lumotlar turiga juda ta’sirchan, ya'ni u kirituv amalga oshayotgan o‘zgaruvhi turiga mos ravishda ma'lumotlarni o‘qiydi. Quyidagi misolga e’tibor bering:

```
// ism va yoshni kiritish
```

```
int main()
{
cout << "Iltimos ismingiz va yoshingizni kiriting\n";
string first_name; // string turdagi o‘zgaruvchi
int age; // integer turidagi o‘zgaruvchi
cin >> first_name; // string turdagi ma’lumotni o‘qib olamiz
cin >> age;  // integer turidagi ma’lumotni o‘qib olamiz
cout << "Hello, " << first_name << " (age " << age << ")\n";
}
```

Demak siz klaviaturada Carlos 22 ni tersangiz kiritish operatori >> first_name o‘zgaruvchisiga Carlos ni 22 sonini esa age o‘zgaruvchisiga o‘qib oladi va quyidagi natijani ekranga chop etadi:

```
Hello, Carlos (age 22)
```

Nega Carlos 22 satri butunlayicha *first_name* o‘zgaruvchisiga yozilmaganining sababi satrlarni o‘qish ajratish belgisi (whitespace) ya’ni probel yoki tabulyatsiya belgisi uchrashi bilan yakunlanadi. Bunday holatda ajratish belgisi kiritish operatori >> tomonidan tashlab ketiladi va sonni o‘qishga o‘tiladi.

Agar siz klaviaturada 22 Carlos ni terib ko‘rsangiz kutilmagan natijaga guvoh bo‘lasiz. 22 soni first_name o‘zgaruvchisiga yoziladi, chunki 22 ham belgilar ketma ketligi hisoblanadi. Boshqa tomondan esa Carlos butun son emas va u o‘qilmasdan tashlab ketiladi. Natijada esa ekranga 22 soni va daviomda “(age” literali va ihtiyoriy son masalan -9842 yoki 0 chop etiladi. Nega? Chunki siz age o‘zgaruchisining boshlang‘ich qiymatini kiritmadingiz va hech nima kiritmadingiz,

natijada unda musor qiymat qolib ketdi. Hozir esa shunchaki age o‘zgaruchisiga boshlang‘ich qiymat berib qo‘yamiz.

```
// ism va yoshni kiritish (2- usul)
int main()
{
cout << "Iltimos ismingiz va yoshingizni kiriting\n";
string first_name = "???";// string turidagi o‘zgaruvchi
// ("???" ism kiritilmaganligini bildiradi")
int age = -1;    // int turidagi o‘zgaruvchi (-1 “yosh aniqlanmaganligini bildiradi”)
cin >> first_name >> age; // satr undan so‘ng butun sonni o‘qiymiz
cout << "Hello, " << first_name << " (age " << age << ")\n";
}
```

Endi 22 Carlos satrini kiritish quyidagi natijaga olib keladi:

```
Hello, 22 (age -1)
```

E’tibor bering, biz kiritish operatori orqali bir nechta qiymatlarni kiritishimiz mumkin, bitta chiqarish operatori bilan ularni chop etishimiz mumkin. Bundan tashqari chiqarish operatori << ham kiritish operatori >> singari turlarga sezuvchandir, shuning uchun string turidagi o‘zgaruvchi va bir qator satrlar bilan birgalikda butun son (int) turdagi o‘zgaruvchini chop etishimiz mumkin.

string turidagi obyektni kiritish operatori >> orqali kiritish ajratish belgisi uchraganda to‘xtatiladi boshqacha qilib aytganda kiritish operatori alohida

so‘zlarni o‘qiydi. Ba’zida bizga bir nechta so‘zlarni o‘qish kerak bo‘ladi. Buning ko‘plab usuli bor, masalan ikkita so‘zdan iborat ismni o‘qib olish mumkin:

```
int main()
{
cout << "Iltimos ism, familiyangizni kiriting\n";
string first;
string second;
cin >> first >> second; // ikkita satr o‘qib olamiz
cout << "Hello, " << first << ' ' << second << '\n';
}
```

Bu yerda biz kiritish opratorini >> ikki marta ishlatdik. Agar bu so‘zlarni ekranga chiqarish kerak bo‘lsa ular orasidan probel qo‘yish zarur.

Amallar va operatorlar. O‘zgaruvchilarning turlari ularda qanday ma’lumot saqlanishidan tashqari ular bilan qanday amallar bajarish mumkinligini ham ko‘rsatadi:

```
int count;
cin >> count; // kiritish operatori >> butun sonni count obyektiga yozadi
string name;
cin >> name; // kiritish operatori kiritilgan satrni name o‘zgaruvchisiga yozadi
int c2 = count+2; // + operatori ikki sonni qo‘shadi
```

```
string s2 = name + " Jr. "; // + operator belgilar bilan to‘ldiradi

int c3 = count-2;     // - ikki sonni ayiradi

string s3 = name - "Jr. ";    // xato: - operatori satrlar uchun aniqlanmagan.
```

Xato o‘rnida biz kompiltatorning dasturni kompilyatsiya qilmasligini nazarda tutyapmiz. Kompilyator har bir o‘zgaruvchiga qanday amallarni bajarish mumkinligini biladi va xato qilsihga yo‘l qo‘ymaydi. Lekin kompilyator qaysi o‘zgaruvchilarga qanday amallarni bajarish mumkinligi haqida bilmaydi va quyidagidek be’mani xatolarga yo‘l qo‘yib beradi:

```
int age = -100;
```

Ko‘rinib turibdiki, inson manfiy yoshga ega bo‘la oilmaydi, lekin hech kim bu haqida kompilyatorga aytmadi, shuning uchun bunday hollarda kompilyator hech qanday xatolik haqida habar bermaydi. Quyida eng ko‘p tarqalgan turlar uchun amallar ro‘yxati keltirilgan:

	bool	**char**	**int**	**double**	**string**
Tenglash (qiymat	=	=	=	=	=
Qo‘shish			+	+	
Ulash					+
Ayirish			–	–	
Ko‘paytirish			*	*	
Bo‘lish			/	/	
Qoldiq			%		
Birga inkrement			++	++	
1ga			– –	– –	
n ga inkrement			+=n	+=n	
Ohiriga qo‘shish					+=
n ga qo‘shish			-=n	-=n	
Ko‘paytirib tenglash			*=	*=	
Bo‘lib tenglash			/=	/=	
Qoldiq olish va			%=		
s fayldan x ga o‘qish	s>>x	s>>x	s>>x	s>>x	s>>x
x ni s faylga yozish	s<<x	s<<x	s<<x	s<<x	s<<x
Teng	==	==	==	==	==

Teng emas	!=	!=	!=	!=	!=
Katta	>	>	>	>	>
Katta yoki teng	>=	>=	>=	>=	>=
Kichig	<	<	<	<	<
Kichig yoki teng	<=	<=	<=	<=	<=

Bo‘sh kataklar amal bu turlarga to‘g‘ridan to‘g‘ri qo‘llanib bo‘lmasligini ko‘rsatadi.Vaqti bilan barcha opreatorlarni tushuntirib o‘tamiz.

Xaqiqiy sonlarga doir quidagi na’munani ko‘rib chiqamiz:

```
// operatorlar ishini ko‘rsatuvchi sodda dastur
int main()
{
cout << "Iltimos haqiqiy son kiriting: ";
double n;
cin >> n;
cout << "n == " << n
<< "\nn+1 == " << n+1
<< "\ntri raza po n == " << 3*n
<< "\ndva raza po n == " << n+n
<< "\nn kvadrati == " << n*n
<< "\nyarim n == " << n/2
<< "\nn ning kvadrat ildizi == " << sqrt(n)
<< endl; // yangi qatorga o‘tishning sinonimi ("end of line")
}
```

Ko‘rinib turibdiki, oddiy arifmetik amallar odatiy ko‘rinishga ega va ularning vazifasi bizga maktab dasturidan ma’lum. Albatta haqiqiy sonlar ustidagi

barcha amallar ham operatorlar ko‘rinishida tasvirlanmagan, masalan, kavadrat ildiz funksiya ko‘rinishida tasvirlangan. Ko‘plab amallar funksiyalar sifatida tasvirlangan. Bu holda esa n ning kvadrat ildizini topish uchun standart kutubhonalardagi sqrt(n) funksiyasi ishlatilmoqda [5].

string turi uchun kam operatorlar ko‘zda tutilgan lekin bu tur ustida ko‘plab amallar funksiyalar sifatida ifodalangan. Bundan tashqari ularga operatorlarni ham qo‘llash mumkin:

```
// ism familyani kiritish
int main()
{
cout << "Iltimos, ism va familiyangizni kiriting\n";
string first;
string second;
cin >> first >> second; // ikki satrni o‘qib olamiz
string name = first + ' ' + second; // satrlarni birlashtiramiz
cout << "Hello, " << name << '\n';
}
```

Satrlar uchun + operatori birlashtirishni bildiradi; boshqacha qilib aytganda, agar s1 va s2 o‘zgaruvchilari string turiga ega bo‘lsa, unda s1 + s2 ham s1 satrdan keyin s2 satr belgilari kelgan satr hisoblanadi. Masalan s1 “Hello” qiymatiga, s2 bo‘lsa “World” qiymatiga ega bo‘lsa, unda s1 + s2 “HelloWorld” qiymatiga ega bo‘ladi. Satrlar ustidagi amallarning asosiylaridan biri bu taqqoslashdir.

```
// ismni kiritish va taqqoslash
```

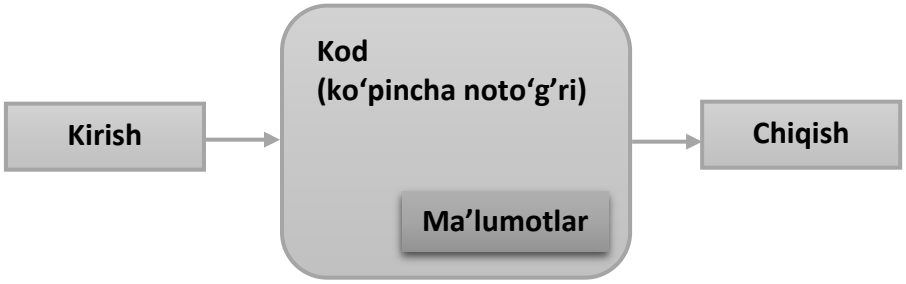
[5] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

```
int main()
{
cout << "Iltimos ikkita ism kiriting\n";
string first;
string second;
cin >> first >> second; // ikkita satr o‘qib olamiz
if (first == second) cout << "ismlar bir hil\n";
if (first < second)
cout << first << " alifbo bo‘yicha birinchi keladi
" << second <<'\n';
if (first > second)
cout << first << " alifbo bo‘yicha keyin keladi "
<< second <<'\n';
}
```

Bu yerda harakatlarni tanlash uchun if operatori ishlatilgan. Bu operator haqida ilovada keltirilgan adabiyotlardan o‘qishingiz mumkin.

2.3. Hisoblash

Barcha dasturlar nimanidir hisoblaydi; boshqacha aytganda ular kirishda qandaydir ma’lumotlarni oladi va qandaydir natijalarni chiqaradi. Bundan tashqari, dasturlar bajariladigan qurilmaning o‘zi kompyuter deb nomlanadi (ingliz tilidan tarjima qilganda computer - hisoblagich). Bu nuqtai nazar biz kiritish va chiqarish muomalasiga keng rioya qilishimizda to‘g‘ri va oqilona hisoblanadi.



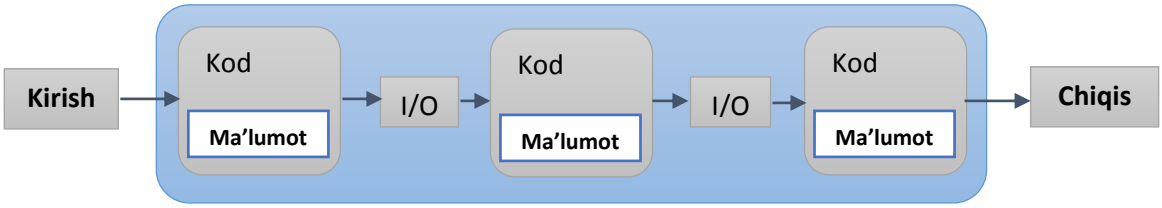
Kiruvchi ma’lumotlar klaviaturadan, sichqonchadan, sensor ekrandan, fayllardan, kiritishning boshqa qurilmalaridan va dasturning boshqa qismlaridan kiritilishi mumkin. "Kiritishning boshqa qurilmalari" kategoriyasi qiziqarli ma’lumotlar manbaini o‘z ichiga oladi: musiqiy klavishli pultlar, videoyozuv qurilmalari, harorat hisoblagichlari, raqamli videokameralar sensori va shu kabilar. Bu qurilmalarning turlari chegarasi yo‘q.

Dasturga kiruvchi ma’lumotlarni qayta ishlash uchun odatda ma’lumotlar tuzilmasi (data structures) yoki ularning tarkibi (states) deb nomlanuvchi maxsus ma’lumotalrdan foydalaniladi. Masalan, kalendarni aks ettiruvchi dasturda turli mamlakatlardagi bayram kunlari ro‘yxati va ish yuzasidan uchrashuvlaringiz keltirilgan bo‘lishi mumkin. Ma’umotlarning ayrimlari avval boshidan dastur qismi hisoblanadi, boshqalari esa, qachonki dastur ma’lumotlarni o‘qiganda va ularda foydali ma’lumotalrni uchratganda sodir bo‘ladi. Masalan, kalendarni aks ettiruvchi dastur, ish bo‘yicha uchrashuvlaringizni kiritishni boshlaganingizda ro‘yhatni yaratishi mumkin. Bu holatda, asosiy kiruvchi ma’lumot uchrashuv oy va kuniga so‘rov yuborishda va ish bo‘yicha uchrashuvlar ma’lumotlarini kiritishda hisoblanadi.

Kiruvchi ma’lumot turli xil manbalardan kirishi mumkin. Shunga o‘xshash, natijalar ham turli xil qurilmalarda chop etilishi mumkin: boshqa dasturlar yoki dastur qismlari ekranida. Chiquvchi qurilmaga tarmoq interfeyslari, musiqiy sintezatorlar, elektr motorlar, quvvat generatorlari, isitgichlar va boshqalar kiradi.

Dasturchi nuqtai nazaridan kiritish-chiqarishda eng muxim va qiziqarli kategoriyalar “boshqa dasturga” va “dasturning boshqa qismlariga” hisoblanadi. Qanday qilib dasturni o‘zaro ta’sir etuvchi qismlar ko‘rinishida tasavvur qilish va

ma’lumotlarga o‘zaro kirish va ma’lumotlar almashishni ta’minlash mumkinligi. Bu dasturlashning kalit savollari. Ularni grafik ko‘rinishda tasvirlaymiz.



I/O qisqartmasi kiritish-chiqarishni bildiradi. Bunday holatda dasturning bir qismidan chiqish keyingi qismga kirish hisoblanadi. Dasturning bu qismlari doimiy xotira qurilmasida ma’lumotlarni saqlash yoki tarmoq aloqalari orqali uzatiladigan asosiy xotirada saqlanadigan malumotlarga kirishga ega bo‘ladi.

Dastur qismida kiruvchi ma’lumotlar, odatda argument, dastur qismidan chiquvchi ma’lumotlar esa natija deb ataladi.

Hisoblash deb, aniq kiruvchi ma’lumotlar asosida va aniq natijalarni hosil qiluvchi qandaydir ta’sirlarga aytiladi. Eslatib o‘tamizki, 1950 yilgacha AQSh da kompyuter deb, hisoblashlarni amalga oshiruvchi insonlar atalgan, masalan, hisobchilar, navigator, fizik. Hozirgi kunda oddiygina ko‘plab hisoblashlarni kompyuyeterga topshirdik, bularning orasida kalkulyator eng soddasi hisoblanadi.

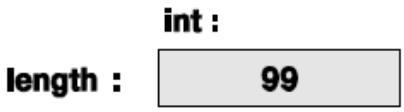
Ifoda. Dasturning asosiy konstruktor qurilmasi ifoda hisoblanadi. Ifoda aniqlangan operandlar soni asosida qandaydir qiymatlarni hisoblaydi. Sodda ifodalar o‘zida oddiy literal konstantalarni ifodalaydi, masalan, 'a', 3.14 yoki "Norah".

O‘zgaruvchilar nomi ham ifoda hisoblanadi. O‘zgaruvchi – bu nomga ega bo‘lgan obyekt. Misol ko‘ramiz.

```
// maydonni hisoblash:
int length = 20; // literal butun qiymat
// (o‘zgaruvchini inisializasiyalash uchun
foydalaniladi)
int width = 40;
```

```
int area = length*width; // ko‘paytirish
```

Bu yerda length so‘zi, o‘zlashtiruvchi operator chap operandalarini belgilovchi “length nomli obyektni” anglatadi, shuning uchun bu ifoda quyidagicha o‘qiladi: “length nomli obyektga 99 raqamini yozish”. O‘zlashtiruvchi yoki inisializasiyalovchi operatorning chap qismida (u “length o‘zgaruvchining lvalue” deb nomlanadi) va bu operatorlarning o‘ng qismida (bu holatda u “length o‘zgaruvchining rvalue”, “length nomli obyektning qiymati” yoki oddiy “length qiymati”) joylashgan length o‘zgaruvchini ajratib olish kerak. Bu kontekstda nomlangan quti ko‘rinishida o‘zgaruvchilarni tasvirlash foydali.



Boshqacha aytganda, length – bu 99 tagacha qiymatni o‘z ichiga oluvchi int toifali obyekt nomi. Ba’zida length nomi (lvalue sifatida) qutiga, ba’zida esa (rvalue sifatida) – mana shu qutida saqlanayotgan qiymatning o‘ziga tegishli bo‘ladi.

+ va * operatorlari yordamida ifodalarni birlashtirib, quyida ko‘rsatilganidek murakkabroq ifodalarni yaratishimiz mumkin. Ifodalarni guruhlash kerak bo‘lganda qavslardan foydalanish mumkin.

```
int perimeter = (length+width)*2; // qo‘shish va
ko‘paytirish
```

Qavssiz bu ifodani quyidagi ko‘rinishda yozish mumkin:

```
int perimeter = length*2+width*2;
```

juda qo‘pol va xatoliklarni keltirib chiqaradi.

```
int perimeter = length+width*2; // length bilan
width*2 qo‘shish
```

Oxirgi xatolik mantiqiy hisoblanadi va kompilyator uni topa olmaydi. Kompilyator aniq ifodada inisializasiyalangan perimeter nomili o‘zgaruvchini

ko‘radi. Agar ifoda natijasi ma’noga ega bo‘lmasa, bu sizning muammoyingiz. Siz perimetrning matematik aniqliklarini bilasiz, kompilyator esa yo‘q.

Dasturda operatorlar bajarilish tartibini aniq belgilaydigan, oddiy matematik qoidalardan foydalaniladi, shuning uchun `length+width*2`, `length+(width*2)` ni anglatadi. Shunga o‘xshash `a*b+c/d`, `a*(b+c)/d` emas `(a*b)+(c/d)` ni anglatadi.

Qavsdan foydalanishning birinchi qoidasi shundaki: “Agar ikkilanayotgan bo‘lsang qavsdan foydalan”. Umuman olganda dasturchi `a*b+c/d` formula qiymatida ikkilanmaslik uchun ifodalarni to‘g‘ri shakllantirishni o‘rganishi kerak. Operatorlardan keng foydalanish, masalan `(a*b)+(c/d)` dastur o‘qilishini susaytiradi.

Nima uchun biz o‘qilishiga to‘xtalib o‘tdik? Chunki sizning kodingizni faqatgina siz emas, balki boshqa dasturchilar ham o‘qishi mumkin, chalkashtirilgan kod o‘qishni sekinlashtiradi va uning tahliliga to‘sqinlik qiladi. Qo‘pol kodni faqatgina o‘qish emas, balki uni to‘g‘irlash ham qiyin bo‘ladi. Yomon yozilgan kod odatda mantiqiy xatoliklarni berkitib turadi. Uning o‘qilishida qancha ko‘p kuch ketgan bo‘lsa, o‘zingizni va boshqalarni uning to‘g‘riligiga ishontirish shuncha qiyin bo‘ladi. Juda ham qiyin bo‘lgan ifodalarni yozmang va har doim ma’noli nomlarni tanlashga harakat qiling.

```
a*b+c/d*(e-f/g)/h+7 // juda qiyin
```

2.4 Xatoliklar

Dastur ishlab chiqishda xatolardan qochib bo‘lmaydi, lekin dasturning ohirgi nusxasi iloji boricha xatolarsiz bo‘lishi zarur [6].

Xatolarning ko‘plab turi bor:

- *Kompilyatsiya paytidagi xato*. Kompilyator tomonidan aniqlangan xatolar.

[6] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

Bularni dasturlash tilining qaysi qoidalarnini buzishiga qarab bir nechta turlarga bo‘lishimiz mumkin.

- Sintaktik xatolar;
- Tiplar bilan yo‘lga qo‘yilgan xatolar.

- *Bog‘lanishlar paytidagi xato.* Bu bog‘lanishlar tahrirlovchisi tomonidan obyekt fayllarni bajarilish moduliga qo‘shish paytida topilgan xatolar.
- *Bajarilish paytidagi xatolar*. Bu dastur bajarilish davomida vujudga kelgan xatolar. Bularni quyidagi turlarga bo‘lish mumkin:
  - Kompyuter tomonida aniqlangan xatolar (uskunaviy taminot va/yoki operatsion tizim tomonidan aniqlangan xatolar);
  - Kutubhonalar tomonidan aniqlangan xatolar (masalan standart kutubhonalar tomonidan);
  - Foydalanuvchining dasturi tomonidan aniqlangan xatolar.
- *Mantiqiy xatolar*. Dasturchi tomonidan noto‘g‘ri vaziyatlar yuzaga kelayotganda topilgan xatolar.

Oddiy qilib aytganda dasturchining vazifasi – barcha xatolarni bartaraf etish. Lekin ko‘p xollarda bu vazifa amallab bo‘lmas bo‘ladi. Aslida haqiqiy dasturlar uchun “barcha xatolar” deganda nimani tushunish juda qiyin. Masalan, dastur ishlab turgan paytda biz kompyuterni elektor manbaidan uzib qo‘ysak buni xato sifatida qarab unga qarshi chora ko‘rish kerakmi? Ko‘p hollarda rad javobini beramiz, lekin meditisina monitoringi dasturida yoki qo‘ng‘iroqlarni yo‘naltirish dasturlarida bunday emas. Bunday xollarda foydalanuvchi sizning dasturingizdan ma’lum bir hisoblashlarni davom ettirishini talab qiladi. Asosiy savol shundan iborat: Dasturingiz o‘zi xatolikni aniqlashi kerakmi yoki yo‘qligida.

Agar bu aniq ko‘rsatilmagan bo‘lsa, biz sizning dasturingiz quyidagi shartlarni bajarishi kerak:

1. Istalgan kiruvchi ma’lumotlarda dasturdan talab qilingan hisoblash ishlarini bajarishi kerak.
2. Barcha to‘g‘irlab bo‘lmas hollarda kerakli xabarni chiqarishi kerak.

3. Uskunaviy ta’minotning barcha xatoliklari xaqida ma’lumot berishi shart emas.
4. Dasturiy ta’minotning barcha xatosini chiqarishi shart ema.
5. Xatolik topilganda dastur ishini yakunlashi kerak.

3-5 ko‘rsatilgan shartlarga javob bermaydigan dasturlarni ko‘rib chiqmaymiz. Shu bilan birga 1 va 2 professional talablardan biri hisoblanadi, professionallik esa bizning maqsadimiz. 100% mukammalikka erisha olmasakda u qisman mavjud bo‘lishi zarur.

Dastur tuzishda xatolar odatiy hol va ulardan qochib bo‘lmaydi. Bizning fikrimizcha, jiddiy dasturlar yaratishda, xatolarni chtlab o‘tish, topish va to‘g‘irlash 90% ko‘proq vaqtni oladi. Xavfsizligi muhim bo‘lgan dasturlarda esa bu vaqt undan xam ko‘p bo‘ladi. Kichik dasturlarda xatolardan osongina qochish mumkin, lekin katta dasturlarda bu xatoga yo‘l qo‘yish ehtimoli juda yuqori.

Biz dastur tuzishda quyidagi uch uslubni taklif qilamiz:

- Dasturiy ta’minotni xatoliklarini iloji boricha kamaytirib ishlab chiqish.
- Ko‘plab xatolarni dstur ishlab chiqish va testlash jarayonida to‘g‘irlab ketish.
- Qolgan xatolarni jiddiy emasligiga ishonch hosil qilish.

Bu uslublarning birortasi ham o‘z o‘zidan xatolarni to‘g‘ri bo‘lishini ta’minlamaydi. Shuning uchun ham biz barcha uch uslubni ham ishlatamiz.

Ishonchli dasturlarni ishlab chiqishda malaka katta ahamiyatga ega [7].

Xatolarning bir nechta manbalarini keltirib o‘tamiz.

Yomon tasvirlash. Agar biz dasturning vazifasini tasavvur qila olmasak, uning “qora burchak” larini tekshirib chiqishimiz juda qiyin bo‘ladi.

[7] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

Chala dasturlar. Dasturni yaratish davomida biz e'tiborga olmay ketgan xatoliklar albatta yuzaga chiqadi. Bizning vazifamiz barcha holatlar to‘g‘ri ishlab chiqilganligiga ishonch hosil qilish.

Ko‘zda tutilmagan argumentlar. Funksiyalar argumentlar qabul qiladi. Agar funksiya ko‘zda tutilmagan argument qabul qilsa, muammo hosil bo‘ladi, masalan standart kutubhonada mavjud sqrt() funksiyasiga manfiy ishoralik son berilsa. sqrt() funksiyasi faqatgina musbat xaqiqiy son qabul qilgani uchun u to‘g‘ri natija qaytara olmaydi. Bunday muammolar 5.5.3 bo‘limda ko‘rib o‘tiladi.

Ko‘zda tutilmagan kiruvchi ma'lumotlar. Odatda dasturlar malumotlarni o‘qib olishadi (masalan, klaviaturadan, fayldan, lokal va global tarmoqlardan). Odatda dasturlar kiruvchi ma'lumotlar uchun ko‘plab shart qo‘yadi, masalan foydalanuvchi son kiritishi kerakligi. Agar foydalanuvchi kutilayotgan son o‘rniga “Bas qil!” degan satrni kiritsachi? Muammoning bunday turi 5.6.3 va 10.6 bo‘limlarida ko‘rib chiqiladi.

Kutilmagan holat. Ko‘plab dasturlar ko‘plab ma'lumotlarni (holatlarni) saqlashga mo‘ljallangan, masalan tizimnig qismlari. Ular safiga manzillar ro‘yxati, telefon katalogi va havo harorati haqidagi ma'lumotlar vector turidagi obyektga yozilgan bo‘lsin. Bu ma'lumotlar to‘liq bo‘lmasa yoki noto‘g‘ri bo‘lsa nima bo‘ladi? Bunday holatda dasturning turli qismlari boshqaruvni saqlab qolishi zarur.

Mantiqiy xatolar. Bunday xatolar dasturdan talab etilgan vazifani bajarmaslikka olib keladi. Bunday xatolarni topib bartaraf etishimiz zarur.

Sintaktik xatolar.

area() funksiyasini quyidagicha chaqirsak nima sodir bo‘ladi:

```
int s1 = area(7; // xato: qavs tushirib qoldirilgan
)

int s2 = area(7) // xato: nuqta vergul tushirib
qoldirilgan ;

Int s3 = area(7);     // xato: Int — tur emas
```

```
    int s4 = area('7); // xato: tirnoqcha tushirib
qoldirilgan '
```

Xar bir qator sintaktik xatoga ega, boshqacha qilib aytganda ular C++ grammatikasiga to‘g‘ri kelmaydi. Afsuski barcha hollarda ham xatolarni dasturchi tushinishiga oson qilib ifodalash qiyin. Natijada eng oddiy sintaktik xatolar xam tushunarsiz ifodalanadi, bundan tashqari xatolikka ko‘rsatayotgan qator ham bir oz uzoqroqda joylashgan bo‘ladi. Shuning uchun kompilyator ko‘rsatayotgan qatorda hech qanday xatolik ko‘rmayotgan bo‘lsangiz biroz yuqoriroq qatorlarni tekshirib chiqing.

Turlar bilan bog‘liq xatoliklar.

Sintaktik xatolarni to‘g‘irlaganingizdan so‘ng kompilyator turlarni e’lon qilishdagi xatoliklar xaqida ma’lumot bera boshlaydi. Masalan e’lon qilinmagan o‘zgaruvchi, yoki unga berilayotgan qiymatning to‘g‘ri kelmasligi to‘g‘risidagi xatoliklar:

```
    int x0 = arena(7);    // xato: e’lon qilinmagan
funksiya
```

```
    int x1 = area(7);     // xato: argumentlar soni
noto‘g‘ri
```

```
    int x2 = area("seven",2); // xato: birinchi
argument noto‘g‘ri tipga ega
```

Xato emas.

Kompilyator bilan ishlash davomida qaysidir payt u sizning xatolaringizni oldindan bilishini xoxlaysiz va ularni xato sifatida qaramasligini hohlaysiz. Lekin malakangiz oshgani sari siz kompilyatorni iloji boricha ko‘roq xatolar topishini hohlab qolasiz. Quyidagi misolni ko‘rib chiqamiz:

```
    int x4 = area(10,-7); // OK: lekin tomoni -7 ga
teng bo‘lgan to‘g‘ri to‘rtburchakni qanday tasavvur
qilish kerak?
```

```
    int x5 = area(10.7,9.3); // OK: lekin aslida
area(10,9) chaqirilyapti

    char x6 = area(100, 9999); // OK: lekin natija
kesib tashlanadi
```

Kompilyator x4 o‘zgaruvchisi xaqida xech qanday xabar chiqarmaydi. Uning qarashi bo‘yicha area(10, -7) to‘g‘ri hisoblanadi: area() funksiyasi ikkita butun son talab qiladi, siz esa ikki butun sonni unga yuboryapsiz; hech kim bu sonlar musbat bo‘lishi xaqida gapirmagan.

Nazorat savollari

1. To‘rtta asosiy xatoliklar turini ayting va ularni qisqacha ifodalab bering.
2. Tinglovchilar dasturidagi qaysi xatolarni tashlab ketishimiz mumkin?
3. Xar qanday yakunlangan proyekt nimani kafotlashi kerak?
4. Xatoliklarni topish va bartaraf etishning uchta asosiy uslubini sanab o‘ting.
5. Nega biz xato qidirishni yoqtirmaymiz?
6. Sintaktik xato nima? Beshta misol keltiring.
7. Turlar xatosi nima? Beshta misol keltiring.
8. Mantiqiy xato nima? Uchta mmisol keltiring.

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

3-ma’ruza. Kiritish va chiqarish.

Reja:

3.1 Kiritish va chiqarish oqimlari

3.2 Kiritish - chiqarish oqimi modeli

3.3 Fayllar

3.4 Kiritish va chiqarishni sozlash

Kalit so‘zlar: *bad(), iostream, kiritish amali, buffer, istream, chiqarish amali, clear(), ofstream, kiritish tugallanganligi belgisi, close(), open(), eof() oqimining holati, ostream, kiritish qurilmasi, fail(), unget(), chiqarish qurilmasi, good(), qurilma drayveri, fayl, ifstream.*

3.1 Kiritish va chiqarish oqimlari

Ma’lumotlarsiz hisob – kitob ma’nosizdir. Bizni qiziqtirayotgan hisoblashni amalga oshirish uchun dasturga ma’lumotlarni kiritishimiz va natijalarni olishimiz kerak.

Agar biz ehtiyotkorona bo‘lmasak, bizning tuzgan dasturimiz faqat aniq belgilangan manbadan ma’lumot olib, natijani aniq belgilangan chiqarish qurilmasiga chiqaradi. Ba’zi bir dasturlar, masalan, raqamli fotoapparatlar yoki yonilg‘i injektori sensorida bu narsa qo‘l kelishi mumkin (ba’zi hollarda esa bu zarur ham), lekin umumiy miqyosdagi masalalarni yechishda biz haqiqiy kiritish va chiqarish qurilmalaridan ma’lumot o‘qish va chiqarishga yordam beruvchi turli imkoniyatlarni bo‘lib qo‘llashimiz lozim bo‘ladi. Agarda biz dasturimizni bevosita ma’lum qurilmalar bilan ishlaydigan qilib yaratsak, unda har safar yangi qurilmalar yaratilganda dasturimizni o‘zgartirishimiz yoki foydalanuvchilarni faqat bizning dasturimiz ishlaydigan qurilmalarni qo‘llash bilan cheklashimiz kerak bo‘lib qoladi. Albatta, bu noto‘g‘ri.

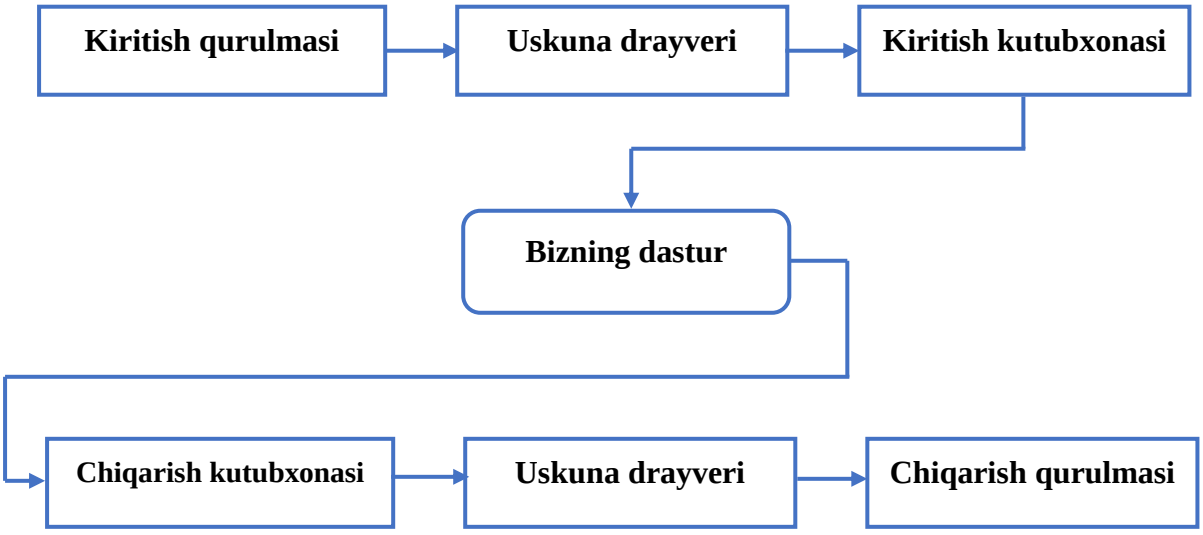
Ko‘pchilik zamonaviy operatsion tizimlar kiritish - chiqarish qurilmalari bilan ishlashni tegishli drayverlarga topshirishadi, so‘ngra dasturlar kiritish –

chiqarish kutubxonasidan foydalanib shu drayverlarga murojaat qiladi. Umuman olganda, qurilmalarning drayverlari operatsion tizimga chuqur biriktirilgan va ko‘pchilik foydalanuvchilar uchun cheklangan. Kutubxona ma’lumotlari esa kiritish – chiqarishning mavhumligini ta’minlaydi. Shunday ekan, dasturchi qurilmalar va ularning drayverlari haqida bosh qotirmasligi kerak.

Bunday model ishlatilayotganda, barcha kiritish va chiqarish ma’lumotlari kiritish – chiqarish kutubxonasi tomonidan ishlanadigan bayt(simvol)lar oqimi sifatida qaralishi mumkin. Bizning dasturchi sifatida ishimiz quyidagicha.

1. Kiritish – chiqarish oqimlarini tegishli ma’lumot manbalari va manzillariga sozlash.
2. Ularning oqimlarini o‘qish va yozib olish.

Ma’lumotlar manbai:



Qurilmadan qurilmaga simvollarni o‘tkazish amaliy detallari kiritish – chiqarish kutubxonasi va qurilmalar drayverlari zimmasida bo‘ladi. Bu va keyingi bo‘limlarda C++ tilining standart kutubxonasi yordamida formatlangan

ma’lumotlar oqimidan tashkil topgan kiritish – chiqarish tizimini qanday yaratishni ko‘rib chiqamiz [8].

Dasturchi sifatida qaralsa ko‘plab kiritish – chiqarish turlari mavjud..

- (Fayllar, tarmoqdagi ulanishlar, yozib oluvchi qurilmalar yoki displeylar bilan bog‘liq bo‘lgan) Ma’lumotlarning (ko‘plab) birliklari oqimi.
- Klaviatura orqali foydalanuvchi bilan o‘zaro aloqa.
- Grafik interfeys (obyektlarni chiqarish, sichqoncha tugmasini bosish va h.k.) orqali foydalanuvchi bilan o‘zaro aloqa.

Bu klassifikatsiya yagona emas, kiritish – chiqarishning 3 ta turi orasidagi farq esa unchalik aniq emas. Masalan, agar simvollarni chiqarish oqimi o‘zi bilan brauzerga yo‘naltirilan HTTP – hujjatni namoyish qilsa, natijada foydalanuvchi bilan aloqaga o‘xshagan va grafik elementlarni o‘zida jamlay oladigan narsa hosil bo‘ladi. Biz birinchi bo‘limdan boshlab iostream kutubxonasini ishlatib keldik va bu hamda keying bo‘limlarda ham uni ishlatamiz. Foydalanuvchi bilan grafik interfeys orqali grafik aloqani turli xildagi kutubxonalar ta’minlaydi. Kiritish – chiqarishning bu ko‘rinishini 12-16 bo‘limlarda ko‘rib chiqamiz.

3.2 Kiritish - chiqarish oqimi modeli.

C++ tili standart kutubxonasi kiritish oqimlari uchun istream va chiqarish oqimlari uchun ostream kabi tuzilishga ega. Biz dasturlarimizda istream standart oqimini cin nomi bilan, ostream standart oqimini esa cout nomi bilan ishlatdik. Shuning uchun ham standart kutubxonaning (ko‘pincha iostream deb ataluvchi) bu qismi biz uchun notanish emas.

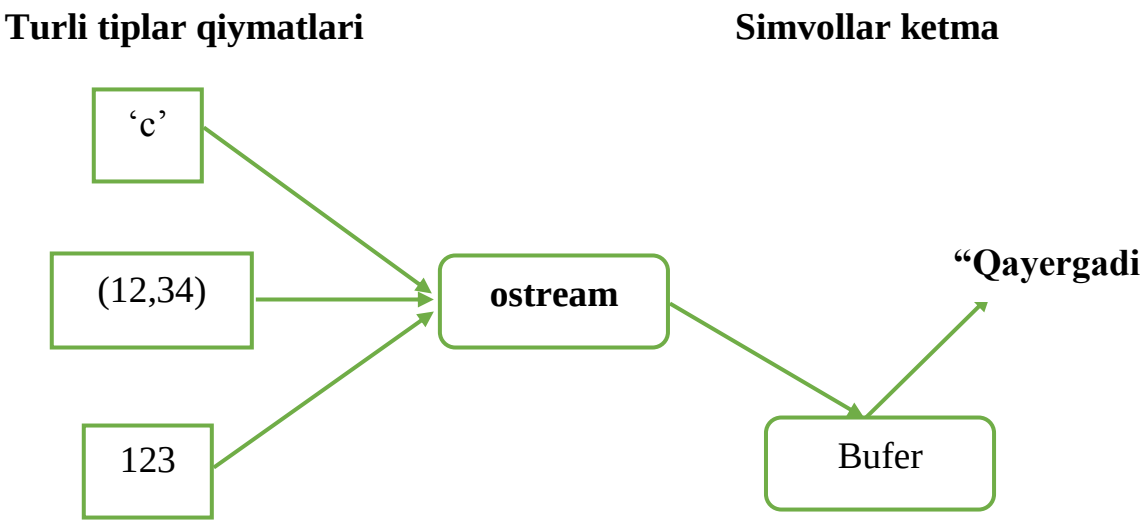
Ostream oqimi quyidagilarni bajaradi.

- Turli tipdagi qiymatlarni belgilar ketma – ketligiga aylantiradi.

[8] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

- Bu belgilarni “qayergadir”(masalan, konsolga, faylga, asosiy xotiraga yoki boshqa kompyuterga) jo‘natadi.

ostream oqimini quyidagicha tasavvur qilish mumkin.



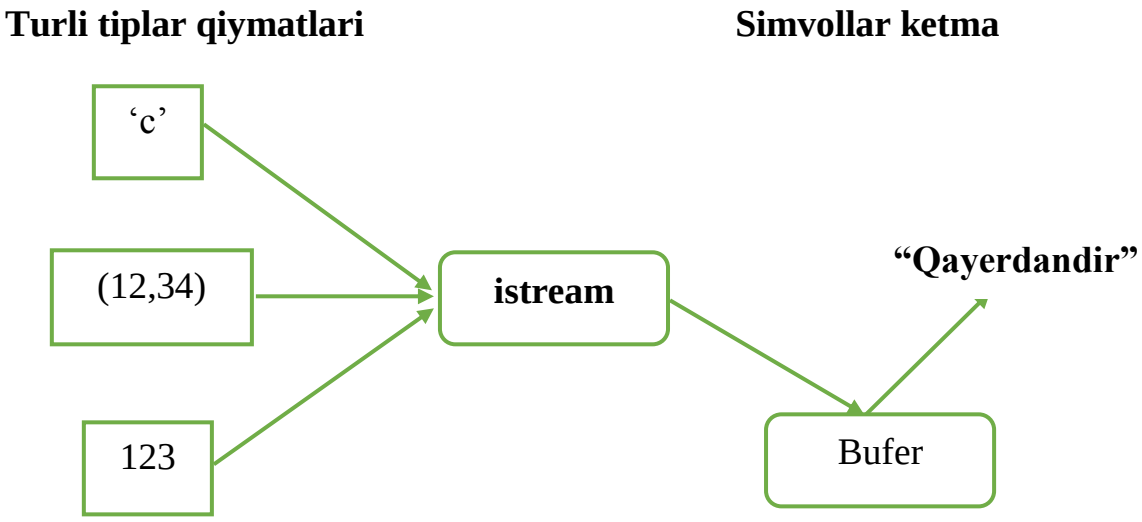
Bufer – bu operatsion tizim bilan aloqa qilish jarayonida sizdan olingan ma’lumotlarni saqlash uchun ostream oqimi qo‘llaydigan ma’lumotlar tuzilmasidir. Ma’lumotlarni ostream oqimiga yozib olish va belgilarning yetib borgan manzilida ko‘rinishi orasidagi qisqa vaqt (ushlanib qolish) bu belgilarning buferda joylashuvi bilan izohlanadi.

istream oqimi quyidagilarni bajaradi.

- Belgilar ketma – ketligini turli tipdagi qiymatlarga aylantiradi.
- Bu belgilarni “qayerdandir” (masalan, konsolda, fayldan, asosiy xotiradan yoki boshqa kompyuterdan) oladi.

istream oqimini quyidagicha tasavvur qilish mumkin.

ostream oqimi kabi operatsion tizim bilan aloqa o‘rnatish uchun istream oqimi buferni qo‘llaydi. Bunda buferlash jarayoni foydalanuvchiga sezilishi mumkin. Agarda siz istream oqimini klaviatura orqali ishlatayotgan bo‘lsangiz, unda barcha kiritilayotgan ma’lumotlar to siz <Enter> tugmasini bosmaguningizcha buferda qoladi. Mabodo kiritishda xatoga yo‘l qo‘ysangiz yoki fikringizdan qaytib qolsangiz, ma’lumotni o‘chirish uchun <Backspace> tugmasidan foydalanasiz (<Enter> tugmasini bosmaguningizcha).



Chiqarishning eng ko‘p qo‘llanishlaridan biri – insonlar o‘qishi mumkin bo‘lgan ma’lumotlarni tashkillashtirishdir. Bunga misol qilib elektron pochta xabarlari, akademik maqolalar, veb – sahifalar, hisoblar, hisobotlar, qurilmalar ko‘rsatkichlari va h.k.larni olish mumkin [9].

3.3 Fayllar

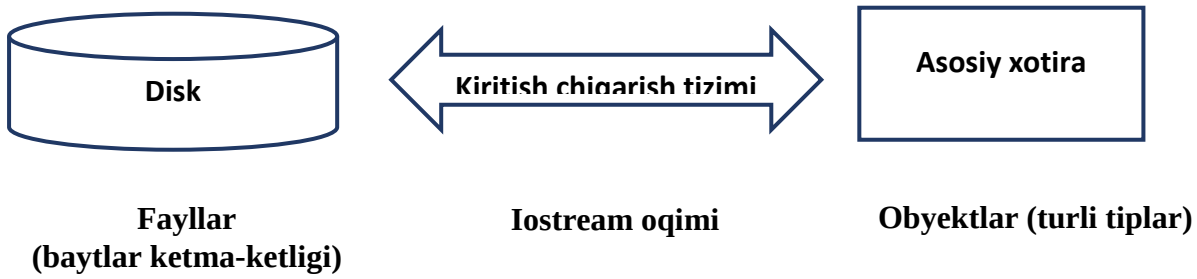
Ko‘pincha bizga tegishli ma’lumotlar kompyuterimiz asosiy xotirasiga sig‘maydi. Shuning uchun ham biz ma’lumotlarimizning bir qismini turli xildagi kata hajmli ma’lumot saqlovchi qurilmalar, disklarda saqlaymiz. Bu qurilmalar shuningdek, kompyuter o‘chganda ma’lumotlar yo‘qolishini oldini oladi. Fayl eng quyi darajada noldan boshlab raqamlangan baytlar ketma – ketligidan iboratdir.

Fayl o‘zining formatiga ega; boshqacha qilib aytganda, baytlar ma’nosini anglatuvchi qoidalar to‘plami. Masalan, agar fayl matnli bo‘lsa, unda daslabki 4 ta bayt birinchi 4 ta belgini ifodalaydi. Boshqa tomondan, agar fayl butun sonlarning

[9] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

binar ko‘rinishini saqlayotgan bo‘lsa, unda dastlabki 4 ta bayt birinchi butun sonning binar ko‘rinishi uchun ishlatiladi (11.3.2 bo‘lim). Fayllar uchun format xuddi asosiy xotiradagi obyektlar uchun tip kabi rol o‘ynaydi.

Fayllar bilan ishlaganda ostream oqimi asosiy xotirada saqlanuvchi obyektlarni baytlar oqimiga aylantiradi va ularni diskka yozadi. istream oqimi esa aksincha ish tutadi; boshqacha qilib aytganda, u diskdagi baytlar oqimini hisoblab, ulardan obyekt yasaydi.



Ko‘pincha biz diskdagi baytlarni oddiy belgilar to‘plamidan olingan belgilar deb hisoblaymiz. Bu har doim ham to‘g‘ri emas. Bundan tashqari barcha fayllar diskda saqlanadi deb hisoblaymiz. Bu ham har doim to‘g‘ri emas (masalan, flesh-xotira), lekin dasturlashning bu bosqichida saqlash qurilmasi ahamiyatga ega emas. Bu fayl mavhumligi va oqimning asosiy ustunliklaridan biridir.

Faylni o‘qish uchun biz

- uning nomini bilishimiz;
- uni ochishimiz (o‘qish uchun);
- belgilarni sanashimiz;
- faylni yopishimiz kerak.

Faylga yozish uchun biz

- unga nom berishimiz;
- faylni ochishimiz yoki shu nom bilan yangi fayl yaratishimiz;
- obyektlarimizni yozishimiz;
- faylni yopishimiz kerak.

Hozir fayllarni qanday ochishni ko‘rib chiqamiz.

Faylni ochish. Agar fayldan ma’lumot o‘qish yoki faylga ma’lumot yozishni xohlasangiz, unda aynan shu fayl uchun maxsus oqim ochishingiz kerak. fstream oqimi — bu iostream oqimi bo‘lib, uni o‘qish uchun ham, yozish uchun ham ishlatish mumkin. Fayl oqimini ishlatishdan oldin uni fayl bilan bog‘lash kerak. Buni misolda ko‘ramiz.

```
cout << "Fayl nomini kiriting: ";
string name;
cin >> name;
ifstream ist(name.c_str()); // ist — bu name satri bilan
                            // berilgan faylning kiritish
                            // oqimi
if (!ist) error("Bu faylni kiritish uchun ochish mumkin emas: ",name);
```

ifstream oqimi name satri bilan berilgan faylni o‘qish uchun ochadi. c_str() funksiyasi – bu string tipi a’zosi bo‘lib, shu tipdagi obyektdan C tili uslubida quyi bosqich satrini yaratadi. !ist ko‘rinishida tekshiruv esa fayl to‘g‘ri ochilgan yoki yo‘qligini aniqlaydi. Shundan so‘ng fayldan ma’lumotlarni xuddi istream oqimi kabi o‘qib olish mumkin. Masalan, >> kiritish operatori Point tipi uchun aniqlangan deb faraz qilamiz. Unda biz quyidagi qism dasturni yozishimiz mumkin bo‘ladi:

```
vector<Point> points; Point p;
while (ist>>p) points.push_back(p);
```

Xuddi shu usulda fayllarga ma’lumot chiqarishni ofstream oqimi orqali bajarishimiz mumkin. Buni misolda ko‘ramiz.

```
cout << "Chiqarish uchun fayl nomini kiriting: ";
```

```
    string oname;
    cin >> oname;
    ofstream ost(oname.c_str()); // ost — bu name satri
bilan
                                 // berilgan faylning
                                chiqarish
                                 //oqimi
    if (!ost) error("Bu faylni chiqarish uchun ochish
mumkin emas: ",oname);
```

Bu yerda << chiqarish operatori Point tipi uchun aniqlangan deb faraz qilamiz:

```
    for (int i=0; i<points.size(); ++i)
    ost << '(' << points[i].x << ',' << points[i].y <<
")\n";
```

Agar fayl oqimi nazar doirasidan chiqib ketsa, unga bog‘liq fayl yopiladi. Agarda fayl yopilsa, unga bog‘liq buffer “tozalanadi”(flush); boshqacha qilib aytganda buferdagi belgilar faylga yoziladi.

Faylning ochilishi ostream va istream oqimlari yaratilish jarayonining qismi hisoblanadi.

Misolda ko‘rib chiqamiz.

```
    void fill_from_file(vector<Point>& points, string&
name){
    ifstream ist(name.c_str()); // o‘qish uchun faylni
ochamiz
    if (!ist) error("Bu faylni kiritish uchun ochish
mumkin emas: ",name);
    // . . . ist oqimini ishlatamiz . . .
```

```
// funksiyadan chiqqanimizda fayl o‘zi yopiladi
}
```

Undan tashqari, open() va close() (B.7.1 bo‘lim) amallarini bajarish ham mumkin. Misolda ko‘rib chiqamiz.

```
ifstream ifs;
// . . .
ifs >> foo; // bajarilmadi: its oqimi uchun birorta
ham fayl
          //ochilmagan
// . . .
ifs.open(name,ios_base::in); // name satri bilan
berilgan
                        // nomdagi faylni ochamiz
// . . .
ifs.close(); // faylni yopamiz
// . . .
ifs >> bar; // bajarilmadi: ifs oqimi bilan bog‘liq
fayl
          // yopiq
// . . .
```

Haqiqiy dasturda paydo bo‘ladigan muammolar ancha murakkabdir. Lekin biz fayl oqimini oldindan yopmay turib uni qaytadan ocha olmaymiz. Misolda ko‘rib chiqamiz.

```
fstream fs;
fs.open("foo", ios_base::in) ; // kiritish faylini
ochamiz
```

```
    // close()funksiyasi tashlab ketildi
    fs.open("foo", ios_base::out); // bajarilmadi: ifs
oqimi
                                   // allaqachon ochilgan
    if (!fs) error("mumkin emas");
```

Oqim ochilgandan so‘ng uni tekshirishni unutmang.

3.4 Kiritish va chiqarishni sozlash

Doimiylik va nodoimiylik. Kiritish – chiqarish kutubxonasi C++ tili standart kutubxonasining bir qismi hisoblanadi. U matnni kiritish va chiqarish uchun yagona va kengayuvchi bazani ta’minlaydi. Bu yerda “matn” so‘zi orqali qandaydir belgilar ketma – ketligini tasavvur qilishi mumkin. Shunday qilib, agar biz kiritish va chiqarish haqida gapirsak, unda 1234 butun soni ham matn sifatida qaraladi.

Hozirgacha biz kiruvchi ma’lumotlar manbalarini farqlamadik.

Ba’zida bu narsa yetishmaydi. Masalan, fayllar boshqa ma’lumot manbalaridan farqlanadi. Ushbu bo‘limda kiritish va chiqarishni o‘zimizning zaruriyatlarimiz uchun sozlashga yordam beruvchi yo‘llar ko‘rib chiqiladi.

Dasturchi sifatida biz doimiylikni ma’qul ko‘ramiz. Shunday qilib, dasturchi sifatida biz dasturning qiyinligi va foydalanuvchilarning shaxsiy talablariga moslanishi o‘rtasidagi muvozanatni saqlashimiz lozim.

Chiqarishni formatlash. Insonlar ko‘pincha diqqatini o‘qimoqchi bo‘lgan ma’lumotlar bilan bog‘liq bo‘lgan mayda detallarga qaratadi. Dasturchi sifatida biz chiqaruvchi ma’lumotlarimizni iloji boricha aniq va dasturimiz foydalanuvchilari ehtiyojlarini qondiruvchi qilishga intilamiz. Chiqarish oqimi (ostream) tegishli tipdagi ma’lumotlarni chiqarishning juda ko‘p imkoniyatlarini o‘zida mujassam qiladi. Foydalanuvchi tomonidan belgilangan tiplar uchun dasturchining o‘zi tegishli << amallarini aniqlashi kerak.

Chiqarishdagi detallar, imkoniyatlar soni cheksizdek tuyuladi, kiritishda esa bor yo‘g‘i bir necha variant bor. Masalan, o‘nli kasr belgisini turli belgilar (qoida bo‘yicha, nuqta yoki bergul) bilan ishlatish mumkin, har xil valyutadagi pul qiymatlari turlicha chiqariladi, rost mantiqiy qiymatini esa true so‘zi (yoki vrai or sandt) orqali ham 1 soni orqali ham ifodalash mumkin. Undan tashqari, satrga yoziladigan belgilarni cheklovchi turli xil usullar mavjud. Bu ma’lumotlar ishlatmaguningizcha siz uchun qiziq emas. Shuning uchun ham o‘quvchilarni ma’lumotnoma va Langer Standart C++ IOStreams and Locales; 21 – bo‘lim va Straustrupning The C++ Programming Language kitobidagi D dasturi kabi maxsus kitoblarga yo‘naltiramiz.

Butun sonlarni chiqarish. Butun sonlarni 8 lik, 10 lik va 16 lik sanoq tizimlari ko‘rinishida chiqarishimiz mumkin. Agarda bu sanoq tizimlari haqida ma’lumotga ega bo‘lmasangiz, A.2.1.1 bo‘limini o‘qib chiqing. Ko‘pchilik hollarda chiqarish uchun 10 lik sanoq tizimidan foylalaniladi. 16 lik sanoq tizimi esa apparat ta’minotiga oid ma’lumotlarni chiqarishda keng qo‘laniladi.

C tili yaratilgan paytda (1970 - yillarda) 8 lik sanoq tizimi ommaviy bo‘lgan, lekin hozirda bu tizim juda kam qo‘llaniladi.

Biz (10 lik sanoq tizimidagi) 1234 sonini 10 lik, 16 lik va 8 lik sanoq tizimlarida chiqaramiz.

```
cout << 1234 << "\t(decimal)\n"
<< hex << 1234 << "\t(hexadecimal)\n"
<< oct << 1234 << "\t(octal)\n";
```

'\t' belgisi “tabulyatsiya belgisi”ni ifodalaydi. U chiqarish ma’lumotlarini quyidagicha ko‘rinishini ta’minlaydi:

```
1234 (decimal)
4d2 (hexadecimal)
2322 (octal)
```

Sonlarni 10 lik sanoq tizimidan boshqa tizimlarda tasavvur qilish o‘quvchini adashtirib qo‘yishi mumkin. Masalan, agar son qaysi sanoq tizimda ekanligi oldindan e’lon qilinmasa, unda 11 soni 9 sonining 8 lik sanoq tizimidagi ko‘rinishini emas, 11 sonining 10 lik sanoq tizimidagi ko‘rinishini bildirishi mumkin. Bunday muammoga uchramaslik uchun, oqimdan butun son qaysi bazada ekanligini so‘rash mumkin. Misolda ko‘ramiz.

```
    cout << 1234 << '\t' << hex << 1234 << '\t' << oct
<< 1234 << '\n';

    cout << showbase << dec; // bazani ko‘rsatish

    cout << 1234 << '\t' << hex << 1234 << '\t' << oct
<< 1234 << '\n';
```

Natija:

```
1234 4d2 2322

1234 0x4d2 02322
```

Xo‘sh, 10 lik sanoq tizimida prefiks mavjud emas, 8 lik sanoq tizimi 0 prefiks, 16 lik sanoq tizimi esa 0x prefiksga ega. Misolda ko‘ramiz.

```
    cout << 1234 << '\t' << 0x4d2 << '\t' << 02322 <<
'\n';
```

10 lik sanoq tizimida bu sonlar quyidagicha ko‘rinishda bo‘ladi:

```
1234 1234 1234
```

Butun sonlarni kiritish. O‘z holida >> operatori sonlarni 10 lik sanoq tizimida deb hisoblaydi. Lekin uni 16 lik yoki 8 lik sanoq tizimida butun sonlar kiritishga majburlash mumkin [10].

```
    int a; int b; int c; int d;

    cin >> a >> hex >> b >> oct >> c >> d;
```

[10] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

```
cout<<a<<'\t'<<b << '\t' << c << '\t' << d << '\n';
```

Agar klaviaturada ushbu sonlarni yozsak

```
1234 4d2 2322 2322
```

Dastur ularni quyidagicha chiqaradi:

```
1234 1234 1234 1234
```

0 va 0x prefikslarini qabul qilish va to‘g‘ri interpretatsiya qilish uchun >> amalidan foydalanish mumkin. Buning uchun o‘z holidagi sozlashlarni bekor qilish lozim. Misolda ko‘ramiz.

```
cin.unsetf(ios::dec);//  10 lik deb hisoblamaslik
cin.unsetf(ios::oct);//  8 lik deb hisoblamaslik
cin.unsetf(ios::hex);//  16 lik deb hisoblamaslik
```

unsetf() oqimining funksiya-a’zosi argument sifatida ko‘rsatilgan bayroqlarni tashlab yuboradi. Bunda, siz quyidagicha yozsangiz

```
cin >>a >> b >> c >> d;
```

va quyidagilarni kiritsangiz

```
1234 0x4d2 02322 02322
```

unda quyidagi ma’lumotlarni olasiz

```
1234 1234 1234 1234
```

Nazorat savollari

1. Zamonaviy kompyuterlarning kiritish va chiqarish vositalari qanchalik turlicha bo‘lishi mumkin?
2. istream oqimi nima qiladi?
3. ostream oqimi nima qiladi?
4. Fayl nima?
5. Fayl formati nima?

6. Dasturga ma’lumotlar kiritish va chiqarish uchun ishlatiladigan 4 xil tipdagi qurilmalarni ayting..

7. Faylni o‘qishning 4 bosqichini aytib bering.

8. Faylga yozishning 4 bosqichini aytib bering.

9. Oqimlarning 4 xil holatini ayting va aniqlang.

10. Ushbu kiritishga oid masalalarni yechish usullari haqida gapiring.

10.1. Foydalanuvchi mumkin bo‘lgan diapazondan oshib ketuvchi qiymat kiritdi.

10.2. Ma’lumotlar tugadi (fayl oxiri).

10.3. Foydalanuvchi noto‘g‘ri tipdagi qiymat kiritdi.

11. Kiritishning chiqarishga nisbatan qiyinligi nimada?

12. Chiqarishning kiritishga nisbatan qiyinligi nimada?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

4-ma’ruza. Grafika. Foydalanuvchining grafik interfeyslari.

Reja:

Kalit so‘zlar: *HTML, grafika, koordinatalar, JPEG, foydalanuvchi grafik interfeysi (GUI), oyna, XML, display, sloy, dasturiy vosita, FLTK kutubxonasi, rang bilan to‘ldirish, chiziq ko‘rinishi, foydalanuvchi grafik interfeysi kutubxonasi.*

4.1. Ekranga chiqarish

4.1.1. Nima uchun grafika?

Dasturlash bilan bog‘liq, biz muhokama qilmaydigan va faqatgina grafikaga tegishli savollarni qisqacha tahlilini qilishimiz mumkin bo‘lgan bir qancha qiziqarli mavzular mavjud. Demak, nima uchun grafika? Asosiysi grafika - bu dasturiy ta’minotni loyihalashga, dasturlashga hamda dasturiy vositalarga tegishli, muxim savollarni tadqiq qilishga imkon beradi.

Grafik foydali. Dasturlash grafikaga nisbatan keng mavzu, foydalanuvchi grafik interfeysi yordamida muammolarni manipulyasiya kodiga nisbatan dasturiy ta’minotlar keng ma’noli. Ammo ko‘p muhitlarda yaxshi grafika muhim rol o‘ynaydi [11].

Grafika chiroyli. Bu kod fragmentini bajarilish natijasi aniq bo‘lganda (xatolarni bartaraf etganda) hisoblash bilan bog‘liq faoliyatning kamyob yo‘nalishlaridan biridir. Grafika bilan ishlash u aniq natija keltirmasa ham yoqimli!

[11] Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013. 488 b.

Grafik dasturlar juda qiziqarli. Dasturlashni o‘rganish ko‘plab dasturlarni o‘qishni o‘z ichiga oladi .

Grafika - loyihalash bilan bog‘liq misollarga to‘la manbaa. Yaxshi grafikani va foydalanuvchi grafik interfeysini ishlab chiqarish va amalga oshirish qiyin. Grafika - loyixalash usullari va loyihalash yechimlari uchun aniq va amaliy misollarga juda boy manbaa.

Grafika obyektga – yo‘naltirilgan dasturlashga kirish va uning til vositalarini qo‘llab-quvvatlash uchun qulay.

Grafika bilan bog‘liq ayrim tushunchalar eskirgan emas. Shuning uchun ular astoydil bayon qilinishi lozim.

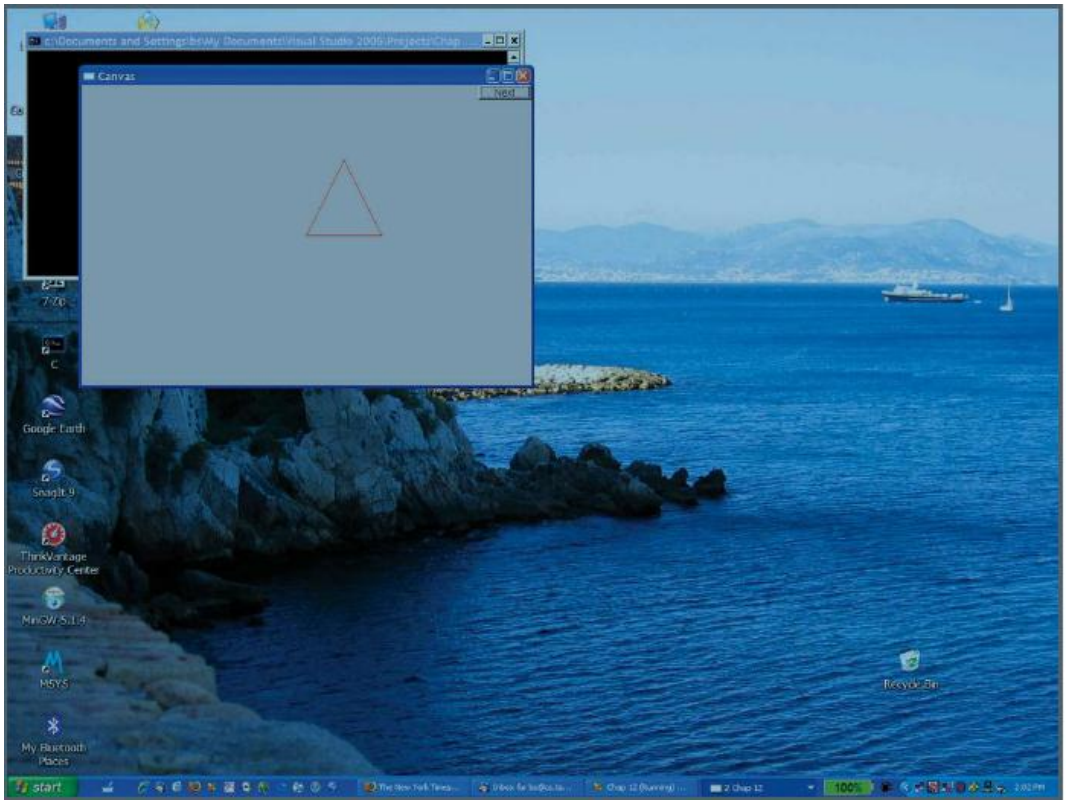
4.1.2. Birinchi misol

Bizning vazifamiz – ekranga chiqarish uchun obyektlarni yaratish mumkin bo‘lgan, sinflarni aniqlash. Masalan, siniq chiziq ko‘rinishida grafik chizishimiz mumkin. Quyida bu vazifani bajarish uchun katta bo‘lmagan dastur keltirilgan:

```
    #include "Simple_window.h" // oyna kutubxonasiga
kirishni ochadi
    #include "Graph.h" // garfik kutubxonaga kirishni
ochadi
    int main()
    {
    using namespace Graph_lib; // bizning grafik
vositalarimiz
    // bo‘shliqda joylashgan
    // Graph_lib nomi
    Point tl(100,100);   // ekranning yuqori chap
burchagini beramiz
    Simple_window win(tl,600,400,"Canvas"); //sodda
oynani yaratamiz
    Polygon poly;    // shaklni yaratamiz(ko‘pburchak)
poly.add(Point(300,200));// nuqtani qo‘shamiz
```

```
    poly.add(Point(350,100));// boshqa nuqtani
qo‘shamiz
    poly.add(Point(400,200));// uchinchi nuqtani
qo‘shamiz
    poly.set_color(Color::red); // poly obyekt
vositasini aniqlaymiz
    win.attach (poly);   // poly obyektini oyna bilan
bog‘laymiz
    win.wait_for_button();   // boshqaruvni ekran
drayverga uzatamiz
    }
```

Bu dasturni ishga tushirib, biz taxminan quyidagi tasvirni ko‘ramiz.



Dastur satrlari bo‘ylab, uni qanday ishlashini ko‘rib chiqamiz. Birinchi navbatda dasturga grafik interfeysimizdagi kutubxonaning boshlang‘ich fayllarini keltiramiz.

```
    #include "Simple_window.h" // oyna kutubxonasiga
kirishni ochadi
```

```
#include "Graph.h" // garfik kutubxonaga kirishni ochadi
```

Keyin main() funksiyasida biz kompyuterga, grafik kutubxonamiz vositalari Graph_lib nomli makonda joylashganini xabar qilamiz.

```
using namespace Graph_lib; // grafik vositalari Graph_lib nomli makonda joylashgan
```

Keyin oynamizning yuqori chap burchagi koordinatalari deb hisoblaydigan nuqtani belgilaymiz.

```
Point tl(100,100); // ekrannnig yuqori chap burchagi koordinatalarini beramiz
```

Keyin ekranda oyna yaratamiz.

```
Simple_window win(tl,600,400,"Canvas"); // sodda oyna yaratamiz
```

Buning uchun Graph_lib kutubxonasida oynani tasvirlovchi, Simple_window sinfidan foydalanamiz. Simple_window sinfining aniq obyekti win nomi bilan nomlanadi; boshqacha aytganda win – bu Simple_window sinfining o‘zgaruvchisi.

Canvas satri oynani belgilash uchun foydalaniladi. Agar yaxshilab qaralsa, unda oyna ramkasining yuqori chap burchagida Canvas so‘zini ko‘rish mumkin.

Oynaga obyektni joylashtiramiz.

```
Polygon poly;    // shaklni yaratamiz (ko‘pburchak)
poly.add(Point(300,200)); // nuqta qo‘shamiz
poly.add(Point(350,100)); // boshqa nuqta qo‘shamiz
poly.add(Point(400,200)); // uchinchi nuqtani qo‘shamiz
```

poly ko‘pburchakni belgilaymiz, so‘zngra unga nuqtani qo‘shamiz. Bizning grafik kutubxonamizda Polygon sinf obyektlari bo‘sh yaratiladi, biz unga xoxlagancha nuqtalar sonini qo‘shishimiz mumkin. Biz uchta nuqta qo‘shdik va uchburchak hosil qildik. Nuqtalar o‘zida oynada berilgan gorizontal va vertikal x va u koordinatalarni aks ettiradi.

Bunday imkoniyatni namoyish etish uchun, ko‘pburchak tomonlarini qizil rang bilan berdik.

```
poly.set_color(Color::red); // poly obyekti vositalarini aniqlaymiz
```

poly obyektini **win** oynasi bilan bog‘laymiz.

```
win.attach(poly); // poly obyektini oyna bilan bog‘laymiz
```

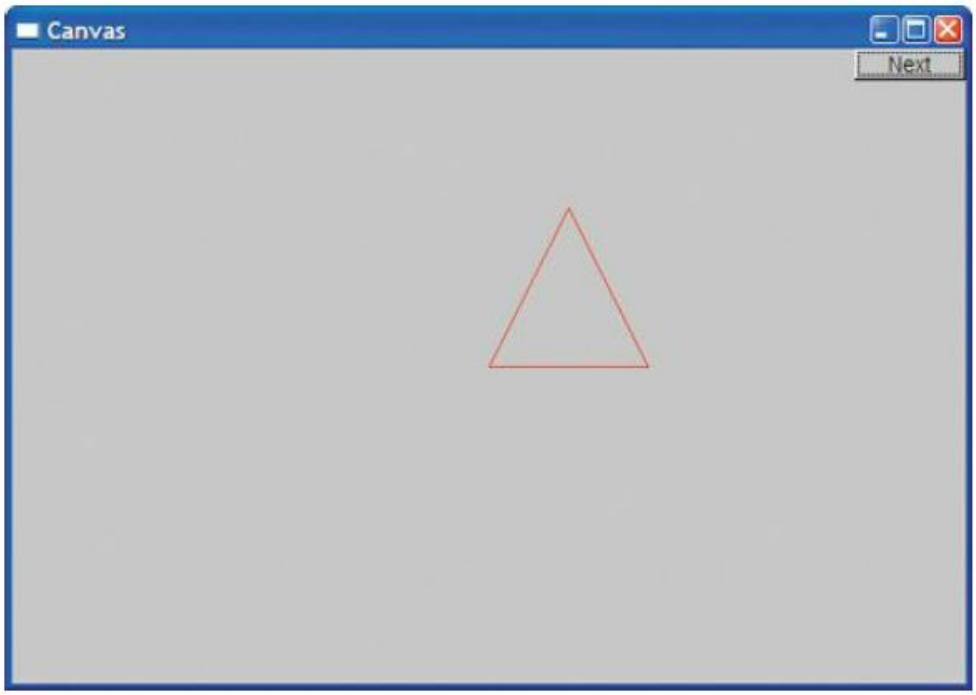
Ekranda hozircha hyech qanday amal bajarilmayotganligini anglash qiyin emas. Biz oyna (aniqrog‘i, Simple_window sinfi obyekti) va ko‘pburchak (poly nomli) yaratdik, ko‘pburchakni qizil rangga bo‘yadik (Color::red) va uni win oynasi bilan bog‘ladik, lekin biz bu oynani ekranga chiqarishga buyruq bermadik. Buni dasturning oxirgi satri bajaradi.

```
win.wait_for_button(); // ekran drayveriga boshqaruvni uzatamiz
```

Grafik foydalanuvchi interfeysi tizimi ekranda obyektlarni aks ettirishi uchun, biz boshqaruvni tizimga uzatdik. Bu vazifani Simple_window oynasida Next tugmachasini bosmaguningizcha tizimni kutishga majbur qiluvchi wait_for_button() funksiyasi bajaradi.

Bu bizga dastur o‘z ishini tugatishidan avval oynani ko‘rishga imkon beradi va oyna yopiladi. Qachonki siz tugmachani bossangiz dastur oynani yopib ishini tugatadi.

Bizning oynamiz quyigi ko‘rinishda bo‘ladi.



Bu grafikada Next tugmachasi ko‘rinmaydi, chunki biz uni Simple_window sinfida qurdik.

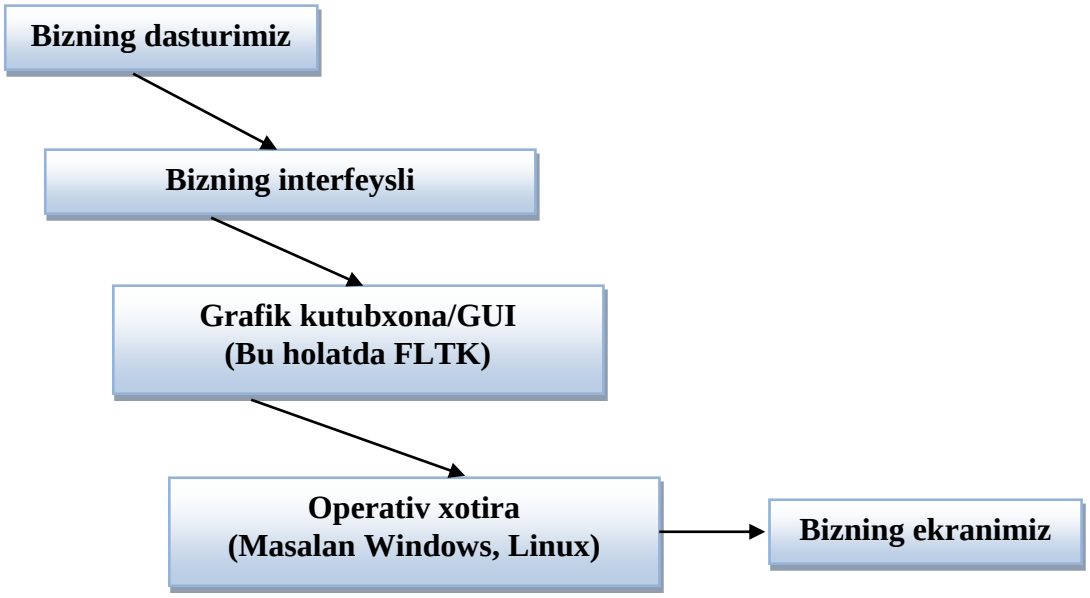
4.1.3. Grafik foydalanuvchi interfeys kutubxonasidan foydalanish

Matnlarni kiritish-chiqarish kabi, dasturimizni soddalashtirish uchun, kiritish-chiqarish qurilmalari, operasion tizimlar va shu kabilar o‘rtasidagi farqlarni bartaraf etuvchi kutubxonadan foydalanamiz. Afsuski, C++ tilida, kiritish-chiqarish standart oqimi kutubxonasiga o‘xshash grafik foydalanuvchi interfeysi standart kutubxonasiga ega emas, shuning uchun mavjud kutubxonalardan biridan foydalanamiz.

Bizning interfeys sinflar taqdim etayotgan dasturlash modellari, odatiy instrumental vositalar to‘plami taklif etayotganga nisbatan soddaroq. Masalan, bizning grafik vositalar va grafik foydalanuvchi interfeysi to‘liq kutubxonasi C++ tilida 600 ga yaqin kod satridan tashkil topgan, shu vaqtning o‘zida FLTK kutubxonasi 370 satrdan tashkil topgan bo‘ladi [12].

Bizning “grafik olam” qismimizni quyidagi ko‘rinishda tasvirlash mumkin.

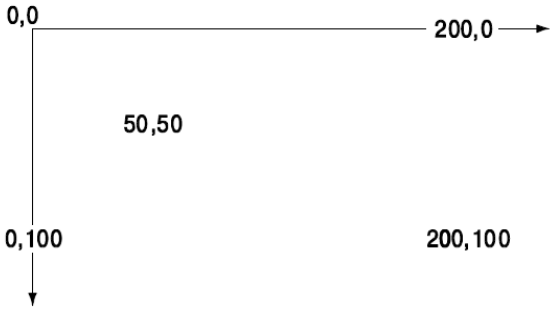
[12] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.



Bizning interfeys sinflar, rang uchun qo‘llab-quvvatlovchi ikki o‘lchamli shakllarni yaratish uchun oddiy va kengaytirilgan asoslarni tashkil qiladi. Bu sinflarni boshqarish uchun, ekranda joylashgan qayta chaqirish funksiyalari, ishga tushiruvchi tugmachalar va boshqa boshqaruvchi elementlarni grafik foydalanuvchi interfeysining oddiy mexanizmidan foydalanishni taklif etamiz.

Kordinatalar. Kompyuter ekrani – bu piksellardan tashkil topgan to‘g‘ri burchakli muhit.

Piksel – bu kichik rangli piksel. Ko‘pincha dasturda ekran to‘g‘ri burchakli piksellar kabi modellashtiriladi. Har bir piksel gorizontal *x* koordinata va vertikal *u* koordinataga ega bo‘ladi. Boshlang‘ich *x* koordinata nolga teng va eng chekka chap piksellarga mos keladi. *x* o‘qi eng o‘ng pikselga yo‘naltirilgan. Boshlang‘ich *u* koordinata nolga teng va yuqori pikselga mos keladi. *u* o‘qi pastga quyi pikselga yo‘naltirilgan.



E’tiboringizni *u* koordinata pastgi yo‘nalishga qarab o‘sib borishiga qarating. Matematiklarga bu qiziq ko‘rinishi mumkin, lekin ekranlar har xil o‘lchamga (ekranga paydo bo‘lgan oynalar ham), va ularda yagona umumiy bo‘lgan yuqori chap burchaklarga ega bo‘lishi mumkin.

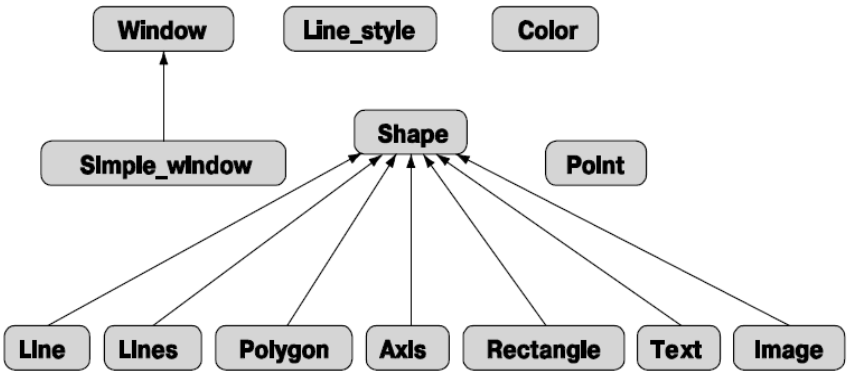
Piksellar soni ekranga bog‘liq: eng keng tarqalganlari 1024×768, 1280×1024, 1450×1050 va 1600×1200 hisoblanadi.

Oyna ekran kabi aniq belgilanadi. Prinsipga ko‘ra oynani kichik ekran kabi izohlash mumkin. Masalan, agar dastur quyidagi qo‘o‘lanmadan tashkil topgan bo‘lsa

```
Simple_window win(tl,600,400,"Canvas");
```

u holda, uni yo‘llash uchun chapdan o‘nga 0 dan 599 va pastdan yuqoriga 0 dan 399 gacha kengligi 600 ga, balandligi esa 400 ga teng bo‘lgan to‘g‘ri burchakli maydon yaratishimizni bildiradi. Ekranda tasvirlash mumkin bo‘lgan oyna maydoni *kanvoy*(canvas) deb ataladi. 600×400 maydon oynaning ichki maydoni hisoblanadi, ya’ni tizimli kadrda joylashgan maydon; u satr va sarlavhani, chiqish tugmalarini o‘z ichiga olmaydi.

Shape sinfi. Bizning ekranda chizish uchun asosiy to‘plamlarimiz o‘nikkita sinfdan tashkil topgan.



Strelka, u chiqadigan sinf u ko‘rsatadigan joyning qayerida sinf talab etilsa shu yerda foydalanilishi mumkinligini anglatadi. Masalan, Polygon sinfi, Shape sinfi qayerda talab etilsa shu yerda foydalanilishi mumkin; boshqacha aytganda, Polygon sinfi Shape sinfining bir turi hisoblanadi.

Avval keyingi sinflardan foydalanishni tavsiflaymiz:

- `Simple_window, Window`
- `Shape, Text, Polygon, Line, Lines, Rectangle, Function va boshqalar.`
- `Color, Line_style, Point`
- `Axis`

Grafik primitivlardan foydalanish. Bu bo‘limda grafik kutubxonaning ayrim elementar primitivlarini ko‘rib chiqamiz: Simple_window, Window, Shape, Text, Polygon, Line, Lines, Rectangle, Color, Line_style, Point, Axis.

Starma – satr uni tushuntiradigan va ekranda uning ishini namoyish etadigan sodda dasturdan boshlaymiz. Bu dasturni ishga tushirganingizda, yangi tasvirlar qo‘shilganda o‘zgarishni va oynada joylashgan mavjud shakllarni modifikasiyalanishini ko‘rishingiz mumkin. Prinsipga ko‘ra bu tahlil animasiyani eslatadi.

main grafik sarlavhalar fayli va funksiyalari.

Birinchi navbatda grafik sinflar va grafik foydalanuvchi interfeys sinflari aniqlangan sarlavha fayllarini kiritamiz.

```
#include"Window.h"    //odatiy oyna
#include"Graph.h"
```

yoki

```
#include "Simple_window.h" // agar bizga Next tugmachasi kerak bo‘lsa
#include "Graph.h"
```

Fahmlaganingizdek Window.h fayli oyna bilan bog‘liq vositalardan, Graph.h fayli esa, oynada shakllarni chizish bilan bog‘liq uskunalardan tashkil topgan. Bu vositalar Graph_lib nomli maydonda tavsiflanadi. Belgilarni soddalashtirish uchun using namespace direktivasidan foydalanamiz.

```
using namespace Graph_lib;
```

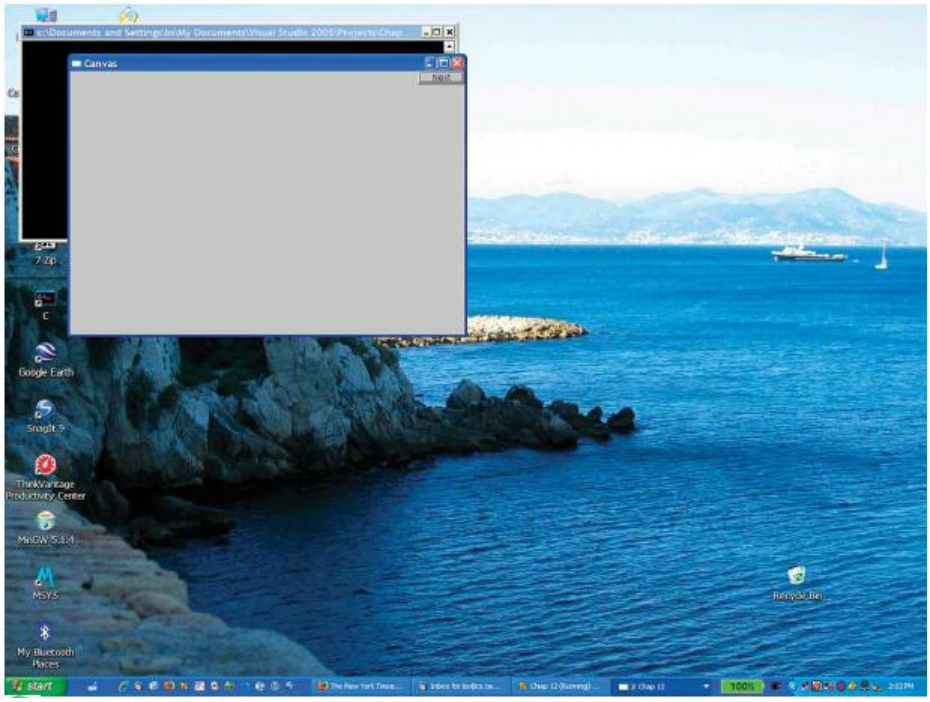
Odatda main() funksiyasi biz bajarishni xoxlaydigan kodni hamda kamdan-kam vaziyatlarni qayta ishlashni o‘z ichiga oladi.

```
int main ()
try
{
// . . . bu yerda bizning kod joylashadi . . .
}
catch(exception& e) {
// xatolar haqida xabar
return 1;
}
catch(...) {
// xatolar haqida boshqa xabar
return 2;
}
```

Deyarli bo‘sh oyna. Bu yerda biz xatolarni qayta ishlashni muhokama qilamiz, hamda main() funksiyasida grafikani tasvirlashga o‘tamiz:

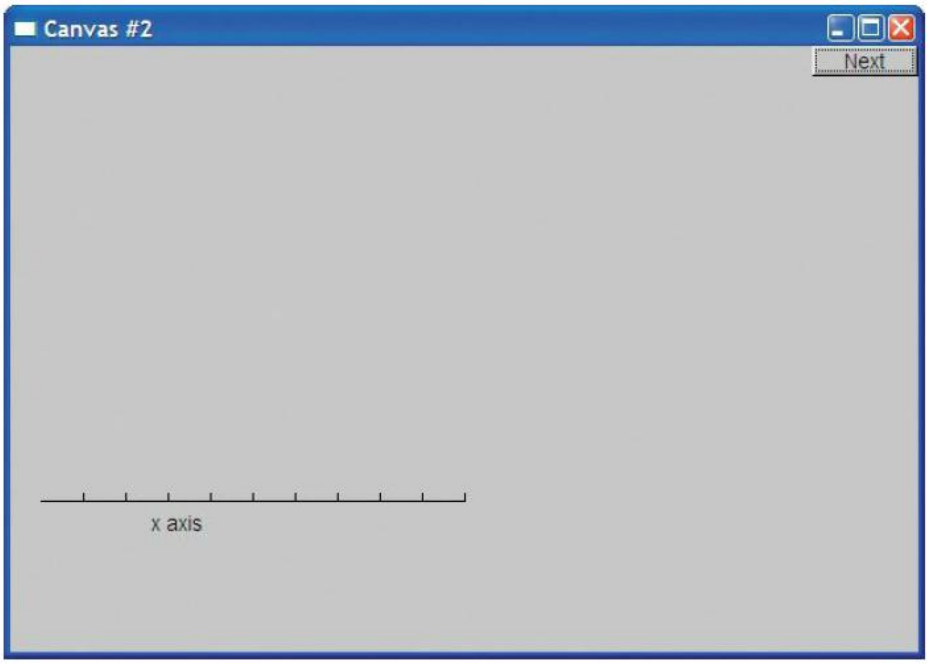
```
Point tl(100,100); // oynamizning yuqori chap burchagi
Simple_window win(tl,600,400,"Canvas");
// tl oyna koordinatalari chap yuqori burchakka joy beradi
// 600*400 oyna o‘lchami
// sarlavha: Canvas
win.wait_for_button(); // tasvirlash!
```

Ekranda oyna chiqishi uchun, biz tizim boshqaruvini grafik foydalanuvchi interfeysiga beramiz. Buning uchun win.wait_for_button() funksiyasidan foydalanamiz. Natija quyidagi rasmda keltirilgan.



Koordinata o‘qlari. Koordinata o‘qlarisiz berilganlarni tasvirlab berishni iloji yo‘q. Shuning uchun bizga koordinata o‘qlari kerak bo‘ladi.

```
    Axis xa(Axis::x, Point(20,300), 280, 10, "x axis");
// Axis obyektni yaratamiz
    // Axis sinf — Shape sinfining turli ko‘rinishi
    // Axis::x gorizontal o‘qni anglatadi
    // o‘q boshi —(20,300) nuqtada
    // o‘q uzunligi — 280 piksel 10 bo‘linuvchi
    // "x o‘qi" — o‘q belgisi
```




```
    win.attach(xa);  // xa obyektni oyna bilan bog‘laymiz
    win win.set_label("Canvas #2"); // oyna belgisini
o‘zgartiramiz
    win.wait_for_button();    // tasvirlash!
```

Ish ketma-ketligi shundaki: Axis sinfida obyekt yaratamiz, uni oynaga qo‘shamiz va ekranga chiqaramiz.

Natijalarni identifikasiyalash uchun Window sinfining set_label() funksiyasi yordamida "Canvas #2" satrida ekran belgisini o‘zgartirdik.

Endi *u* o‘qini qo‘shamiz.

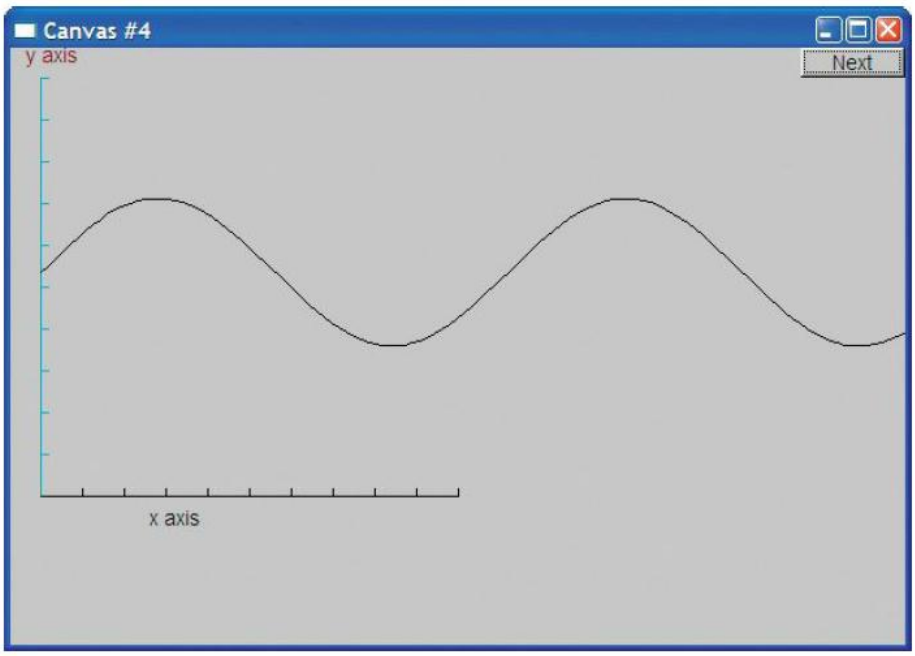
```
    Axisya(Axis::y, Point(20,300), 280, 10, "yaxis");
    a.set_color(Color::cyan);// rangni tanlaymiz
    ya.label.set_color(Color::dark_red);  // matn
rangini tanlaymiz
    win.attach(ya);
    win.set_label("Canvas #3");
    win.wait_for_button();    // tasvirlash!
```

Ayrim imkoniyatlarni namoyish etish uchun, biz *u* o‘qini xavo ranga (cyan), belgini esa to‘q qizilga bo‘yadik.

Funksiya grafigi. Sinus grafigini tasvirlovchi shakl yaratamiz va uni oyna bilan bog‘laymiz.

```
    Function sine(sin,0,100,Point(20,150),1000,50,50);
// sinus grafigi
    // 1000 nuqtadan foydalanib(0,0) dan (20,150) gacha
[0:100) oraliqda sin() chizamiz v diapazone [0:100) ot
(0,0) do (20,150),
    Masshtablash uchun koordinatalar 50 ga
ko‘paytiriladi
    win.attach(sine);
    win.set_label("Canvas #4");
    win.wait_for_button();
```

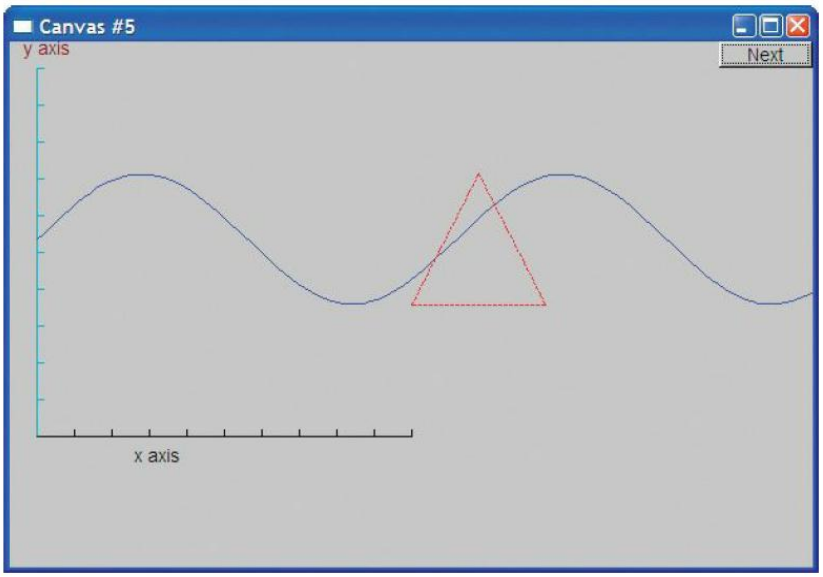
Bu yerda sine nomli Function sinf obyekti sin() funksiyasi standart kutubxonasidan foydalangan holda sinus grafikasini chizadi.



Ko‘pburchak. Polygon sinfi obyekti chiziqlar bilan bog‘langan nuqtalar ketma-ketligida beriladi. Birinchi chiziq birinchi nuqtani ikkinchisi bilan bog‘laydi, ikkinchi chiziq ikkinchi nuqtani uchinchisi bilan bog‘laydi, oxirgi chiziq esa oxirgi nuqtani birinchisi bilan bog‘laydi.

```
    sine.set_color(Color::blue);//sinus grafigi rangini
o‘zgartirdik
    Polygonpoly; //Polygon sinfi – bu Shape sinfining
turli ko‘rinishi
    poly.add(Point(300,200)); //uchta nuqta uchburchakni
tasvirlaydi
    poly.add(Point(350,100));
    poly.add(Point(400,200));
    poly.set_color(Color::red);
    poly.set_style(Line_style::dash);
    win.attach(poly);
    win.set_label("Canvas#5");
    win.wait_for_button();
```

Quyidagi natijani olamiz.



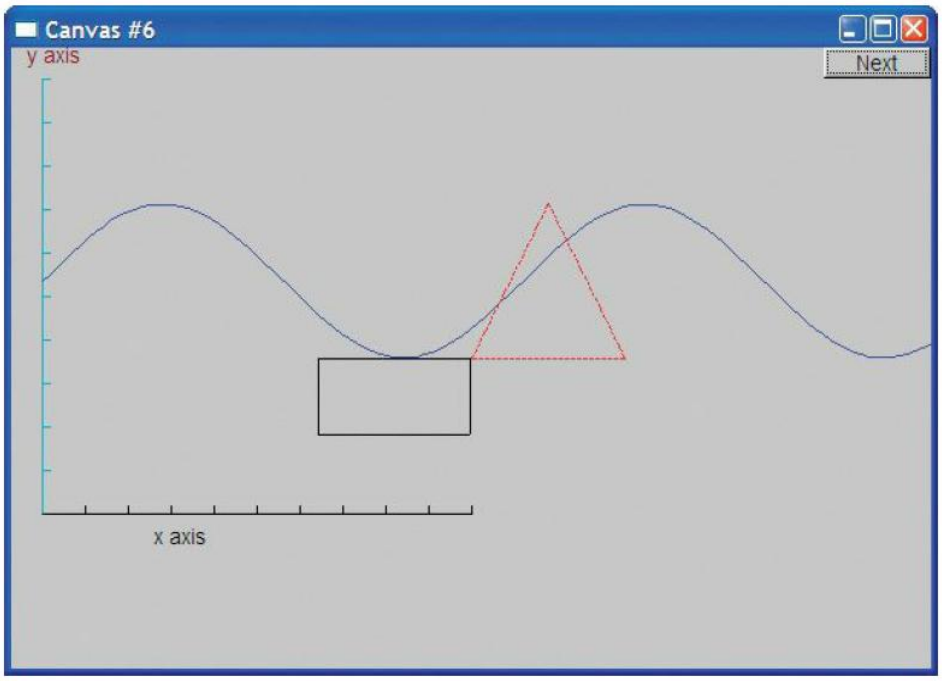
To‘g‘ri burchak. Ekran – bu to‘g‘ri burchak, oyna – bu qog‘oz varag‘i to‘g‘ri burchagi. Shakllarning katta hajmi to‘g‘ri burchak hisoblanadi.

Rectangle sinfi bo‘yi va hajmi bo‘yicha chap yuqori burchak koordinatalarida xarakterlanadi.

```
    Rectangle r(Point(200,200), 100, 50); // yuqori
chap burchak,
    // hajm, bo‘y
```

```
win.attach(r);
win.set_label("Canvas #6");
win.wait_for_button();
```

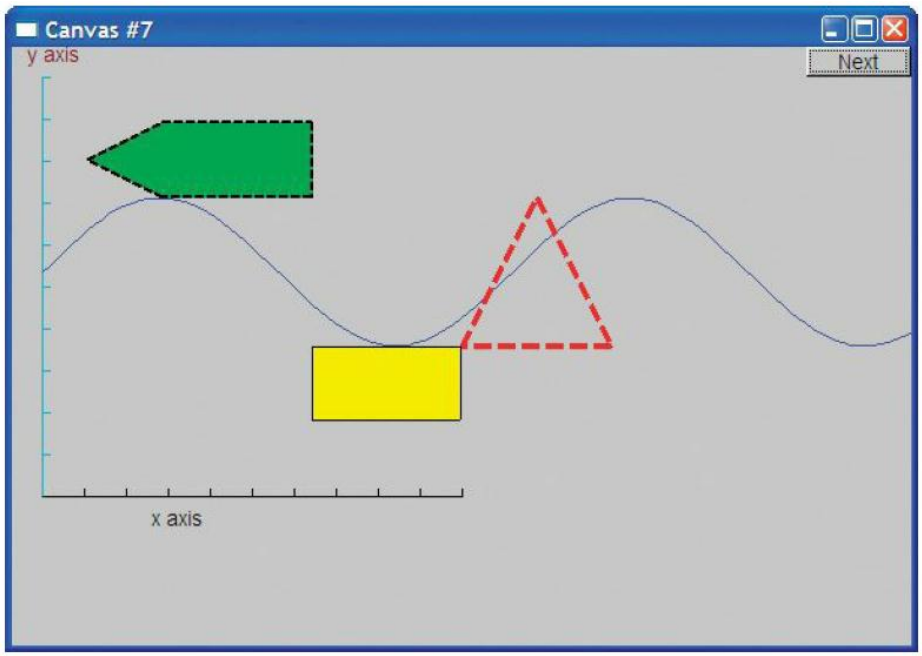
Bu fragment ekranda quyidagi oynani ochadi.



To‘ldirish. Hozirgi vaqtgacha bizning shakllarimiz rangsiz chizilgan edi. Ularni rang bilan to‘ldirish mumkin [13].

```
r.set_fill_color(Color::yellow);//to‘g‘ri burchak
ichidagi rang
poly.set_style(Line_style(Line_style::dash,4));
poly_rect.set_style(Line_style(Line_style::dash,2))
;
poly_rect.set_fill_color(Color::green);
win.set_label("Canvas#7");
win.wait_for_button();
```

[13] Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013. 488 b.



4.2. Grafik sinflar

Asosiy interfeys sinflari quyidagi jadvalda keltirilgan [14].

Interfeysli grafik sinflar

Color	Chiziq, matn yaratish va shaklni to‘ldirish uchun foydalaniladi
Line_style	Chiziq chizish uchun foydalaniladi
Point	Window sinfi obyekti ichida va ekranida joy borligini tekshiruvchi vazifalar uchun foydalaniladi
Line	Point sinfi ikkita obyekti ekranida ko‘ringan chiziqlarni kesish
Open_polyline	Point sinfi obyekti ketma-ketligida aniqlangan kesilgan chiziqlarni bir biri bilan bog‘lash ketma-ketligi
Closed_polyline	Open_polyline sinfiga o‘xshash, lekin farqi shundaki chiziqlar kesimi Point sinfi oxirgi obyektini birinchisi bilan bog‘laydi
Polygon	Closed_polyline sinfi, bu yerda kesmalar hyech qachon

[14] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

	kesishmaydi
Text	Belgilar satri
Lines	Point sinfi aniqlagan kesmalar chizig‘i to‘plami
Rectangle	Tez va qulay tasvirlash uchun optimallashgan shakl
Circle	Radius va markazi aniqlangan aylana
Ellipse	Markazi va ikkita o‘qlari aniqlangan ellips
Function	Aniqlangan kesmada berilgan bitta o‘zgaruvchi funksiyasi
Axis	Belgilangan koordinata o‘qi
Mark	Belgi bilan belgilangan nuqta (masalan, x yoki 0)
Marks	Belgilar bilan belgilangan nuqtalar ketma-ketligi (masalan, x yoki 0)
Marked_polyline	Belgilar bilan belgilangan Open_polyline sinfi
Image	Rasmli fayllar tarkibi

Grafik foydalanuvchi interfeysi sinfi

Window	Grafik obyektlar aks etadigan ekran maydoni
Simple_window	Next tugmachali oyna
Button	Tugmachani bosib biron bir funksiyani chaqirish mumkin bo‘lgan oynadagi to‘g‘ri burchak
In_box	Foydalanuvchi satrni kiritishi mumkin bo‘lgan oyna maydoni
Out_box	Satrni chiqarish mumkin bo‘lgan oyna maydoni
Menu	Button sinfi obyektlari vektori

Boshlang‘ich kod quyidagi fayllardan tashkil topgan.

Boshlang‘ich interfeysli grafik fayllar

Point.h	Point sinflari

Graph.h	Barcha qolgan grafik interfeysli sinflar
Window.h	Window sinfi
Simple_window.h	Simple_window sinfi
GUI.hButton	Grafik foydalanuvchi interfeysi barcha qolgan sinflari
Graph.cpp	Graph.h faylidan funksiyalarni aniqlash
Window.ccp	Window.h faylidan funksiyalarni aniqlash
GUI.cpp	GUI.h faylidan funksiyalarni aniqlash

Nazorat savollari

1. Grafika nima uchun kerak?
2. Nima uchun grafikasiz munosabat qilib bo‘lmaydi?
3. Nima uchun grafika dastruchilar uchun qiziqarli?
4. Oyna nima?
5. Bizning grafik interfeys (bizning grafik kutubxonamiz) sinfimiz aynan qanday muhitda joylashgan?
6. Garfik vositalardan foydalanish uchun bizning kutubxonamizda qanday fayl yetishmaydi?
7. Odiiy oyna bizga nimani taqdim etadi?
8. Kichik (Minimal) oyna bizga nimani taqdim etadi?
9. Oyna nishoni nima?
10. Oyna nishonini qanday qo‘yiladi?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

5-ma’ruza. Vektor va bo‘sh xotira.

Reja:

5.1 Asosiy tushunchalar

5.2 Xotira, adres va ko‘rsatgich

5.3 Bo‘sh xotira va ko‘rsatgich

5.4 Bo‘sh xotiraga joylashtirish

5.5 Destruktorlar

5.6 Elementlarga ruxsat

Kalit so‘zlar: *delete, masofa keltirish, delete[], new, indeks, this, indeksirlash, [] bo‘sh xotira, void*, konteyner, ro‘yxat, manzil, nolinchi ko‘rchsatkich, tugun, adres olish &, bo‘shatish, ko‘rsatkich, virtual destruktor, xotira, xotira chiqishi, destruktor, toifani o‘zlashtirish, resurslar chiqishi, a’zo destruktori.*

5.1 Asosiy tushunchalar

Juda oddiy namuna yordamida vector sinfining izchil rivojlanishini tashkil etamiz [15].

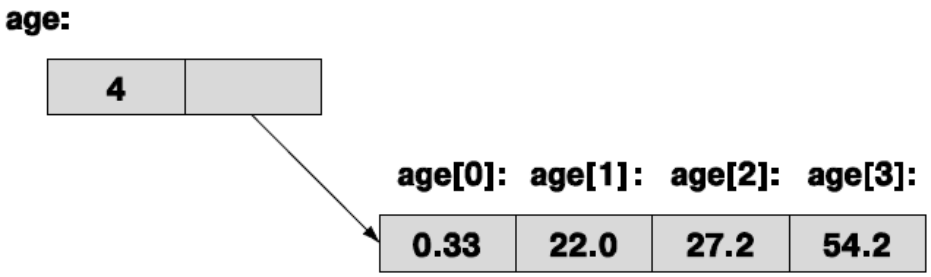
```
vector<double> age(4); // double tipiga mansub 4 ta elementdan iborat vektor

age[0]=0.33;

age[1]=22.0;

age[2]=27.2;

age[3]=54.2;
```

Guvohi bo‘lganingizdek ushbu kod double tipli 4 ta elementdan iborat vector sinf obyektini yaratadi va 0.33, 22.0, 27.2 va 54.2 qiymatlarni

[15] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

o‘zlashtirmoqda. Bu to‘rt element 0, 1, 2 va 3 tartibi bilan raqamlanadi. C++ standart konteynerlarida elementlarni raqamlash doimo 0 (nol) dan boshlanadi. 0 dan boshlab raqamlash C++ tilida dastur tuzuvchilar o‘rtasida universal kelishuv bo‘lib, doimo ishlatidi [16].

vector obyekt sinfidagi elementlar soni uning o‘lchami deb ataladi. Shunday qilib, age vektorining o‘lchami 4 ga teng. Vektor elemetlari 0 dan size-1 gacha raqamlanadi (indekslanadi). Masalan age vektori elementlari 0 dan age.size()–1 gacha raqamlanadi. age vektorini quyidagicha tasvirlashimiz mumkin:



Ushbu sxemani kompyuter xotirasida qanday amalga oshirish mumkin? Qiymatlarni qanday saqlaymiz va ularga ruxsatni qanday amalga oshiramiz? Shubhasiz, biz sinfni tanlashimiz va uni vector deb nomlashimiz lozim. Shundan so‘ng, vektor o‘lchamini va elementlarini saqlash uchun bizga alohida-alohida sinf a’zosi kerak bo‘ladi. Elementlar majmuini qanday taqdim qiladi, ularning sonini o‘zgartirish mumkinmi? Buni vector standart sinfinidan foydalanib amalga oshirish mumkin, lekin, bu doirada u firibgarlik bo‘ladi: biz aynan shu sinfni ishlab chiьiaramiz.

Shunday qilib, rasmdagi tasvirga o‘qni qanday qaratamiz? O‘zimizni u emas deb ishontiramiz. Biz shunda ma’lumotlar tuzulmasining fiksirlangan o‘lchamini belgilashimiz mumkin.

```
class vector {
int size, age0, age1, age2, age3;
// . . .
```

[16] Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013. 488 b.

```
};
```

Quyidagi tasvir kabi belgilarga bog‘liq ayrim detallarni hisobga olmasdan xech narsaga erisha olmaymiz

	age:			
size:	age[0]:	age[1]:	age[2]:	age[3]:
4	0.33	22.0	27.2	54.2

Bu oddiy va chiroyli, lekin, faqatgina qiyin vaziyatda push_back() funksiyasi yordamida element qo‘shishga urinishni amalga oshirishimiz mumkin: Biz element qo‘sha olmaymiz, elementlar soni belgilangan va u to‘rtga teng. Bizga fiksirlangan elementlar sonini saqlaydigan ma’lumotlar tuzilmasiga nisbatan ko‘proq narsalar kerak. vector sinfi obyektida elementlar sonini o‘zgartiruvchi push_back() amallarini, agar vector sinfida elementlar soni fiksirlangan bo‘lsa, amalga oshirish mumkin emas.

Bizga birinchi elementning manzili kerak. C++ tili ma’lumotlar toifasining manzilini saqlash qobiliyati ko‘rsatkich deb ataladi (pointer). Sintaksik jihatdan u * suffiksi bilan belgilanadi, ya’ni double* double tipli obyektga ko‘rsatkichni bildiradi. Endi vector sinfining birinchi variantini aniqlashimiz mumkin.

```
// juda soddalashtirilgan double tipiga mansub vektor elementlari (vector<double> kabi)
class vector {
int sz;  // o‘lchami
double* elem;    // birinchi elementga ko‘rsatgich (double tipli)
public:
vector(int s); // konstruktor: s songa xotira ajratadi
```

```
// double tipi,
// ularga elem ko‘rsatgichini o‘rnatadi,
// s sonini sz da saqlaydi
int size() const { return sz; } // joriy o‘lcham
};
```

vector sinfini loyihalashni davom ettirishdan avval, “ko‘rsatkich” tushunchasini kengroq o‘rganamiz. “ko‘rsatkich” tushunchasi – “massiv” tushunchasi bilan uzviy bog‘langan – bu C++ tilida “xotira” tushunchasiga kalit.

5.2 Xotira, adres va ko‘rsatgich

Kompyuter xotirasi – bu baytlar ketma-ketligidir. Ushbu baytlar nol dan boshlab oxirgisigacha raqamlanadi. Adres (address) identifikasiyalangan xotira yacheykasidagi son deb ataladi. Adres turli ko‘rinishdagi butun sonlar bilan hisoblanadi. Xotiraning birinchi bayti 0 adresga ega, ikkinchisi 1 va x.z. Xotira megabaytlarini quyidagi ko‘rinishda ifodalashimiz mumkin:

Xotirada joylashgan barcha narsa manzilga ega. Misol ko‘ramiz [17].

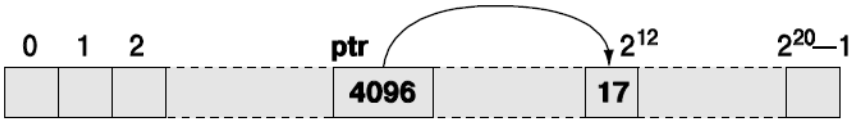
```
int var = 17;
```

Bu qo‘llanma var o‘zgaruvchini saqlash uchun o‘lchami int toifali o‘lcham bilan aniqlanadigan xotira maydonini zahiralaydi va u yerga 17 sonini yozib qo‘yadi. Bundan tashqari, manzillarni saqlash mumkin va ularga amallarni qo‘llash mumkin. Manzilni saqlovchi obyekt ko‘rsatkich deb ataladi.

```
int* ptr = &var; // ptr ko‘rsatgichi var
o‘zgaruvchi adresini o‘zida saqlaydi
```

[17] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

Obyekt adresini belgilash uchun unar & adres olish operatori ishlatiladi. Demak, agar var o‘zgaruvchisi xotira xududida saqlangan bo‘lsa, u xolda birinchi yacheyka adresi 4096 (yoki 2^{12}) ga teng bo‘lib ptr ko‘rsatgichi 4096 sonini o‘zida saqlashi mumkin.



Mavjudligiga qarab, kompyuter xotirasini 0 dan size-1 gacha raqamlangan baytlar ketma-ketligi kabi qarash mumkin. Ayrim mashinalar uchun bunday tasdiqlash juda ham sodda xarakterga ega bo‘ladi, lekin bizning model uchun shuning o‘zi yetarli.

Har qanday toifa mos ko‘rsatkich toifasiga mos bo‘ladi. Misol ko‘ramiz.

```
char ch = 'c';
char* pc = &ch; // char ga ko‘rsatgich
int ii = 17;
int* pi = &ii;   // int ga ko‘rsatgich
```

Agar biz yo‘naltirilayotgan obyekt qiymatini ko‘rmoqchi bo‘lsak, u holda ko‘rsatkichga * unar operatorni qo‘llashimiz mumkin. Misol ko‘ramiz.

```
cout << "pc==" << pc << "; pc qiymati==" << *pc << "\n";
cout << "pi==" << pi << "; pi qiymati==" << *pi << "\n";
```

*pc qiymati c belgisi hisoblanadi, *pi ning qiymati 14 butun soniga teng. pc va pi o‘zgaruvchilar qiymatlari, kompilyator ch va ii o‘zgaruvchilarni xotiraga joylashtirishiga bog‘liq.

```
*pc = 'x';     //      OK: pc ko‘rsatkich yo‘naltirilgan char o‘zgaruvchi,
               //      'x' belgini o‘zlashtirish mumkin
```

pi = 27; // OK: int ko‘rsatkich int ga yo‘naltiriladi, shuning uchun *pi — bu int

*pi = *pc; // OK: (*pc) belgini int (*pi) toifasi o‘zgaruvchisiga o‘zlashtirish mumkin.

E’tiboringizni qarating: ko‘rsatkich qiymati butun son hisoblanishiga qaramasdan, ko‘rsatkichning o‘zi butun son hisoblanmaydi. “int nimaga yo‘naltiriladi?” – noto‘g‘ri savol. Butun sonlar emas, ko‘rsatkichlar yo‘naltiriladi. Ko‘rsatkich toifasi manzillar ustida amallarni bajarish imkonini beradi, shu vaqtning o‘zida int toifasi butun sonlar ustida amallar bajarishga imkon beradi (arifmetik va mantiqiy). Demak, ko‘rsatkich va butun sonlarni aralashtirish mumkin emas.

```
int i = pi; // xato: int* toifasi obyektini int toifasi obyekti bilan o‘zlashtirish mumkin emas
```

```
pi = 7; // xato: int toifasi obyektini int* toifasi obyektiga o‘zlashtirish mumkin emas
```

char (ya’ni char*) ga ko‘rsatkich – bu int (ya’ni int*) ga ko‘rsatkich emas.

Misol ko‘ramiz.

```
pc = pi; // oshibka: nelzya prisvoit obyekt tipa int*
 // obyektu tipa char*
pi = pc; // oshibka: nelzya prisvoit obyekt tipa char*
     // obyektu tipa int*
```

Nima uchun pc o‘zgaruvchini pi o‘zgaruvchiga o‘zlashtirish mumkin emas? Bunga javob sifatida shuni aytish mumkinki — char belgi int toifasidan kichikroq.

```
char ch1 = 'a';
char ch2 = 'b';
```

```
    char ch3 = 'c';

    char ch4 = 'd';

    int* pi = &ch3; // char toifali o‘lchamga ega
bo‘lgan o‘zgaruvchiga yo‘naltiriladi

            // xato: char* obyektini int* toifali
            obyektga o‘zlashtirish mumkin emas

            // biroq buni bajarish mumkinligini
            o‘zimizda tasavvur qilib ko‘ramiz

            *pi = 12345; // char toifali o‘lchamga ega
        xotira maydoniga yozishga xarakat qilib ko‘rish

    *pi = 67890;
```

Kompiyaltor xotirada o‘zgaruvchilarni joylashtirishi uni amalga oshirilishiga bog‘liq va bu quyidagi shaklda bo‘ladi.

Agar kompilyator bunday kodni o‘tkazib yuborsa, u holda biz &ch3 manzilidan boshlanadigan xotira yacheykasida 12345 sonlarni yozishimiz mumkin bo‘ladi. Bu xotira atrofida mavjud bo‘lganlarni o‘zgartirishi mumkin, ya’ni ch2 va ch4 o‘zgaruvchilar qiymatini. Yomon xolatda aynan pi o‘zgaruvchi qismini qayta yozgan bo‘lardik! Bu holatda *pi=67890 keyingi o‘zlashtirishlari umuman boshqa xotiraga 67890 sonlarni joylashtirishga olib kelardi. Bunday o‘zlashtirish taqiqlanganligi juda yaxshi, lekin bunday mexanizmlar xavfsizligi dasturlashning quyi bosqichlarida juda kam.

Demak, apparat ta’minotga juda yaqinmiz. Dasturchi uchun bu juda noqulay. Bizning ixtiyorimizda bir nechta oddiy amallar bor va qo‘llab-quvvatlovchi kutubxonalar yo‘q. Biroq biz bilishimiz kerak, vector sinfi kabi yuqori darajali vositalar qanday amalga oshirilishini. Modomiki barcha kodlar ham

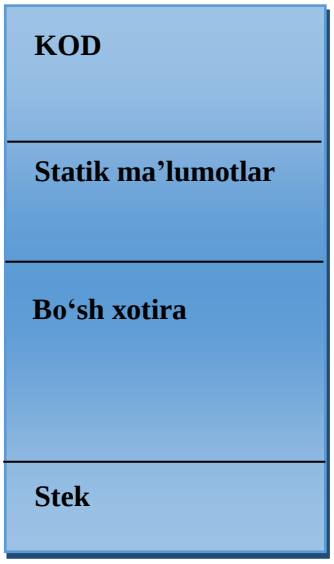
yuqori darajali bo‘la olmaydi, shuning uchun biz quyi darajada kod yozishni bilishimiz kerak. Bundan tashqari yuqori darajali dasturlashning nisbatan ishonchligi va qulayliklarini baholash uchun, quyida darajadagi dasturlashning murakkabliklarini his qilib ko‘rish kerak. Bizning maqsad – ifodalangan chegaralar va qo‘yilgan masala yo‘l qo‘yadigan abstraksiyaning yuqori darajalari bilan har doim ishlash.

5.3 Bo‘sh xotira va ko‘rsatkichlar

5.1. bo‘lim oxirida keltirilgan vector sinfini amalga oshirilishiga qaraymiz. vector sinfi o‘zining elementlarini saqlash uchun qayerdan joy topadi? elem ko‘rsatkichini u ularga yo‘naltirilishi uchun qanday o‘rnatish? Qachonki C++ tilida yozilgan dasturlar bajarilishi boshlansa, kompilyator uning kodi uchun xotira (ayrim hollarda bu xotirani kod deb atashadi (code storage) yoki matnli (text storage)) va global o‘zgaruvchilarni (bu xotirani statik deb atashadi (static storage)) zaxiralaydi. Bundan tashqari, u ularning argumentlari va lokal o‘zgaruvchilarini (bu xotira stekli (stack storage) deb nomlanadi yoki avtomatli (automatic storage)) saqlash uchun funksiyalarni chaqirishda foydalaniladigan xotirani belgilaydi. Kompyuterning qolgan xotirasi boshqa maqsadlar uchun foydalanilishi mumkin; u bo‘sh deb ataladi (free). Bu xotira bo‘linishigi quyidagi shaklda ko‘rish mumkin [18].

[18] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

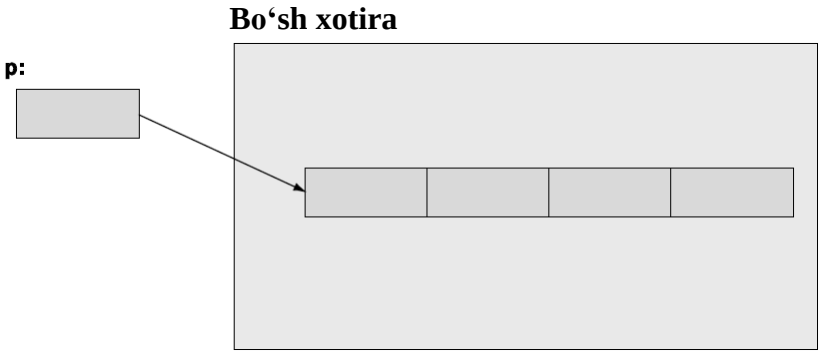
Xotira sxemasi:



C++ tili bu bo‘sh xotirani new operatori yordamida kirish mumkin bo‘lgan holatga olib keladi. Misol ko‘ramiz [19].

```
double* p = new double[4]; // bo‘sh xotiraga double 4 sonini joylashtiramiz
```

Yuqorida keltirilgan qo‘llanmada bo‘sh xotiraga double toifasining to‘rtta sonini joylashtiruvchi va ko‘rsatkichni ularning birinchisiga qaytarilishini dastur bajarishini tizimdan so‘raydi. Bu ko‘rsatkich r o‘zgaruvchini inisializasiya qilish uchun foydalaniladi. Bu quyidagi ko‘rinishda bo‘ladi.



new opretori o‘zi yaratgan obyektga ko‘rsatkichni qaytaradi. Agar new operatori bir nechta obyekt yaratgan bo‘lsa (massiv), u holda u bu massivlardan birinchisiga ko‘rsatkich qaytaradi. Agar bu obyekt X toifaga ega bo‘lsa, unda new operatori qaytaradigan kshrsatkich X* toifasiga ega bo‘ladi. Misol ko‘ramiz.

[19] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

```
char* q = new double[4]; // xato: double* ko'rsatkichi char* ga o'zlashtiriladi
```

Berilgan new operatori double toifali o‘zgaruvchiga ko‘rsatkich qaytaradi, leikn double toifasi char toifasidan farq qiladi, shuning uchun biz char toifali o‘zgaruvchi ko‘rsatkichiga double toifali o‘zgaruvchi ko‘rsatkichini o‘zlashtirishimiz mumkin emas.

5.4 Bo‘sh xotiraga joylashtirish

new operatori bo‘sh xotirani (free store) belgilashni (allocation) bajaradi.

- new operator belgilangan xotiraga ko‘rsatkichni qaytaradi.
- Ko‘rsatkich qiymati belgilangan xotiraning birinchi bayt manzili hisoblanadi.
- Ko‘rsatkich ko‘rsatilgan toifa obyektiga yo‘naltiriladi.
- Ko‘rsatkich qancha elementlar soniga yo‘naltirilayotganligini bilmaydi.

new operatori alohida elementlar kabi, elementlar ketma-ketligi uchun ham xotirani belgilashi mumkin. Misol ko‘ramiz.

```
int* pi =new int;// bitta int o'zgaruvchi uchun xotirani belgilaymiz
```

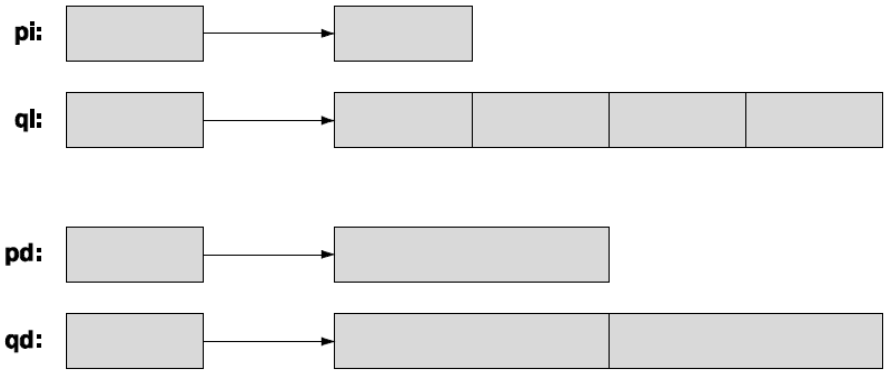
```
int* qi = new int[4]; // to'rtta int o'zgaruvchisi uchun xotirani belgilaymiz
                      // (massiv)
```

```
double* pd = new double; // bitta double o'zgaruvchi uchun xotirani belgilaymiz
```

```
double* qd = new double[n]; // double ning n o'zgaruvchilari uchun xotirani belgilaymiz
```

E’tiboringizni shunga qaratinki, obyektlar soni o‘zgaruvchi sifatida berilishi mumkin. Bu muhim, dasturni bajarishga kirishda qancha massivni

joylashtirishimiz mumkinligiga imkon beredi. Agar n 2 ga teng bo‘lsa, bu quyidagicha bo‘ladi.



Turli toifa obyektlari ko‘rsatkichlari turli toifaga ega bo‘ladi. Misol ko‘ramiz.

```
pi = pd; // xato: int* ko‘rsatkichiga double*
ko‘rsatkichini o‘zlashtirish mumkin emas
```

```
pd = pi; // xato: double* ko‘rsatkichiga int*
ko‘rsatkichini o‘zlashtirib bo‘lmaydi
```

Nima uchun yo‘q? Oxir oqibat biz int o‘zgaruvchini double va aksincha o‘zgaruvchiga o‘zlashtira olamiz. Sababi [] operatorga bog‘lanadi. Elemtentni topish uchun u uning toifasi o‘lchami haqidagi ma’lumotdan foydalanadi. Masalan, qi[2] element qi[0] elementdan ikkita int toifali o‘lchamga teng masofada joylashadi, qd[2] element esa qd[0] elementdan ikkita double toifasi o‘zlchamiga tang masofada joylashadi. Agar int toifasi o‘lchami double toifasi o‘lchamidan farq qilsa, ko‘pgina kompyuterlar singari, qd ko‘rsatkich manzili uchun belgilangan xotiraga qi ko‘rsatkichni ruxsat bergan holda yo‘naltiriladi, bunda tushunarsiz natijalarni olishimiz mumkin.

Bu amaliy nuqtai nazardan tashuntirish. Nazariy nuqtai nazardan tushuntirish esa quyidagicha: turli xil toifadagi ko‘rsatkichlarni bir biriga o‘zlashtirish toifa xatoliklarini (type errors) keltirib chiqarishi mumkin.

Ko‘rsatkichlar yordamida kirish. Ko‘rsatkichni * operatoridan tashqari ko‘rsatkichga indeksirlash[] operatorini ham qo‘llash mumkin. Misol ko‘ramiz.

```
    double* p = new double[4]; // bo‘sh xotiradan
double toifali to‘rtta o‘zgaruvchi uchun xotirani
belgilaymiz

    double x = *p;   //  p yo‘naltirilgan obyektni
o‘qiymiz

    double y = p[2]; //  r yo‘naltirilgan uchinchi
obyektni o‘qiymiz
```

vector sinfi kabi indeksirlash operatori hisoblashni noldan boshlaydi. Bu shuni anglatadiki, ifoda p[2] uchinchi elementga yo‘naltiriladi; p[0] — bu birinchi element, shuning uchun *p nimani anglatsa, p[0] ham shuni anglatadi. [] va * operatorlarni yozish uchun ham foydalanish mumkin.

```
    *p = 7.7;    // r yo‘naltirilgan obyektga sonni
yozamiz

    p[2] = 9.9; // r yo‘naltirilgan uchinchi obyektga
sonni yozamiz
```

Ko‘rsatkich xotirada joylashgan obyektga yo‘naltiriladi.

Ukazatel ssыlayetsya na obyekt, raspolojennыy v pamyati. «*» operatori r ko‘rsatkich yo‘naltirilgan obyektga yozish va o‘qishga imkon beradi.

```
    double x = *p; // r ko‘rsatkichga yo‘naltirilgan
obyektni o‘qiymiz

    *p = 8.9;    // r ko‘rsatkichga yo‘naltirilgan
obyektni yozamiz
```

[] operator qachonki, r ko‘rsatkichda qo‘llanilsa, u r ko‘rsatkich birinchi yo‘naltiriladigan xotirani obyekt ketma-ketligi kabi izohlaydi,

```
    double x = p[3]; // r ga yo‘naltirilgan to‘rtinchi
obyektni o‘qiymiz
```

```
    p[3] = 4.4;  // r ga yo‘naltirilgan to‘rtinchi
obyektni yozamiz

    double y = p[0]; // p[0] – huddi *p kabi
```

Ana bo‘ldi. Bu yerda hyech qanday tekshiruv, hyech qanday amalga oshirish yo‘q — xotiraga kirishning onson yo‘li.

p[0]:	p[1]:	p[2]:	p[3]:
8.9		9.9	4.4

Xotiraga kirish uchun aynan shunday onson va optimal samarali mexanizm biz uchun vector sinfini amalga oshirishda kerak.

5.5 Destruktorlar

Endi biz vektorda elementlarni saqlashni bilamiz. Biz faqatgina yetarlicha bo‘sh xotira sonini belgilaymiz va unga ko‘rsatkich yordamida murojaat qilamiz.

```
    // double toifasidan tashkil topgan judda sodda
vektor

    class vector {

    int sz;  // o‘lcham

    double* elem; // elemetnlarga ko‘rsatkich

    public:

    vector(int s)    // konstruktor

    :sz(s),  // sz a’zolari inisializasiyasi

    elem(new double[s])  // elem a’zolari
inisializasiyasi

    {

    for (int i=0; i<s; ++i) elem[i]=0; // elemetnlar
inisializasiyasi

    }
```

```
int size() const { return sz; } // joriy o‘lcham
// . . .
};
```

Demak, sz a’zolari elementlar sonini saqlaydi. Biz uni konstruktorda inisializasiya qilamiz, vector sinfi foydalanuvchisi esa size() funksiyasini chaqirish orqali elemetnlar sonini aniqlashi mumkin. Elementlar uchun xotira new operatori yordami bilan konstruktorda belgilanadi, new operatori orqali qaytariladigan ko‘rsatkich, elem a’zosida saqlanadi.

E’tiboringizni qarating, biz elemetnlarni ularning (0.0) yashirin qiymatlari bo‘yicha inisializasiya qilamiz. Standart kutubxonadagi vector sinfi aynan shunday qiladi, shuning uchun biz boshidan shunday qilishga qaror qildik.

Afsuski, bizning sodda vector sinfimiz xotiradan chiqib ketishga yo‘l qo‘yadi. Konstruktorda u elementlar uchun xotirani new operator yordamida belgilaydi.

Misol ko‘ramiz.

```
void f(int n)
{
vector v(n); // double toifali n uchun xotirani belgilaymiz
// . . .
}
```

f() funksiyasidan chiqgandan so‘ng, bo‘sh xotirada yaratilgan u vektor elemetnlari o‘chirilmaydi. Biz clean_up() funksiyani aniqlashimiz mumkin – vector sinfi a’zosi va quyidagi ko‘rinishda chaqiriladi:

```
void f2(int n)
{
```

```
    vector v(n); // int toifali boshqa n o‘zgaruvchilar
uchun xotirani belgilovchi
    // . . . v vektordan foydalanamiz. . .
    v.clean_up(); // clean_up() funksiyasi elem
a’zosini o‘chiradi
    }
```

Bu yondoshuv ishlashi kerak edi. Biroq bo‘sh xotira bilan bog‘liq keng tarqalgan muammolardan biri, odamlar delete operatorini esdan chiqarib qo‘yishlaridadir. Ekvivalent muammo clean_up() funksiyasi bilan ham yuzaga kelishi mumkin; odamlar uni chaqirishni unutib qo‘yishadi. Biz samaraliroq yechimni taklif etishimiz mumkin. Asosiy g‘oya shundan tashkil topadiki, kompilyator faqatgina kompilyator haqida emas balki, konstruktorga qarama-qarshi rol o‘ynaydigan funksiya haqida ham bilishi kerak. Bunday funksiyani mantiqiy jihatdan destruktor (destructor) deb ataladi. Konstruktor sinf obyekti yaratilganda oshkora chaqirilmaydi, destruktor esa qachonki obyekt ko‘rish maydoni chegarasida chiqsa, oshkora chaqirilmaydi. Konstruktor obyekt to‘g‘ri yaratilishi va inisializasiyalanishi kafolatlaydi. Destruktor aksincha, obyekt yo‘qotilishidan avval to‘g‘ri tozalanishini kafolatlaydi. Misol ko‘ramiz.

```
    // double toifasidan tashkil topgan juda sodda
vektor
    class vector {
    int sz;  // o‘lcham
    double* elem; // elementlarga ko‘rsatkich
    public:
    vector(int s)    // konstruktor
    :sz(s), elem(new double[s]) // xotirani belgilash
    {
```

```
    for (int i=0; i<s; ++i) elem[i]=0; // elementlarni
inisializasiyalash
    }
    ~vector()    // destruktor
    { delete[] elem; } // xotirani bo‘shatish
    // . . .
    };
```

Demak, keyingi kodni yozish mumkin:

```
    void f3(int n)
    {
    double* p = new double[n];    // double toifasidagi
n son uchun xotirani belgilaymiz
    vector v(n); // vektorni aniqlaymiz (double
toifasidagi n ta boshqa son uchun xotirani belgilaymiz)
    // . . . p va v foydalanamiz. . .
    delete[ ] p; // double toifali sonlar massivi bilan
band xotirani bo‘shatamiz
    } // vector sinfi v obyekt bilan band xotirani
avtomat tarzda bo‘shatadi
```

5.6 Elementlarga kirish

vector sinfi bilan ishlash qulay bo‘lishi uchun, elementlarni o‘qish va yozish kerak. Avvalo sodda funksiya a’zolarini ko‘rib chiqamiz get() va set().

```
    // double toifali sonlar uchun juda sodda vektor
    class vector {
```

```
    int sz;  // o‘lcham
    double* elem; // elemetnlarga ko‘rsatkich
    public:
    vector(int s):
    sz(s), elem(new double[s]) { /* */}   //
konstruktor
    ~vector() { delete[] elem; }  // destruktor
    int size() const { return sz; }   // joriy o‘lcham
    double get(int n) const { return elem[n]; } //
o‘qishga ruxsat:
    void set(int n, double v) { elem[n]=v; }   //
yozishga ruxsat:
    };
```

get() va set() funksiyalari elem ko‘rsatkichiga [ ] operatorni qo‘llab, elementlarga kirishni ta’minlaydi.

Endi biz, double toifali sonlardan tashkil topgan vetorni yaratishimiz va undan foydalanishimiz mumkin.

```
    vector v(5);
    for (int i=0; i<v.size(); ++i) {
    v.set(i,1.1*i);
    cout << "v[" << i << "]==" << v.get(i) << '\n';
    }
    Natijalar quyidagicha ko‘rinishda bo‘ladi:
    v[0]==0 v[1]==1.1 v[2]==2.2 v[3]==3.3 v[4]==4.4
```

vector sinfiga berilgan variant haddan ziyod sodda, get() va set() funksiyasi foydalanilgan kod esa, figurali qavs asosida odatiy kiruvchilar bilan solishtirganda

juda xunuk. Biroq bizning maqsadimiz, shunda belgilanadiki, katta bo‘lmagan va sodda variantlardan boshlab, so‘ngra qadamma-qadam har bir bosqichini tastdan o‘tkazib uni rivojlantirib borish. Bu doimiy testdan o‘tkazish va strategik yechim xatolar sonini va sozlash vaqtini kamaytiradi.

Nazorat savollari

1. Elementlar soni o‘zgaruvchi bo‘lgan ma’lumotlar tuzilmasi nima uchun kerak?

2. Oddiy dasturlarda foydalaniladigan to‘rtta xotira ko‘rinishini ayting.

3. Bo‘sh xotira nima? Uni yana qanday atashadi? Bo‘sh xotira bilan qanday operatorlar ishlaydi?

4. «*» operatori nima va u nima uchun kerak?

5. Manzil nima? C++ tili manzillar bilan qanday manipulyasiya qiladi?

6. Ko‘rsatkich unga yo‘naltirilayotgan obyekt haqida qanday ma’lumotlarni tashiydi?

Qanday foydali ma’lumotlarni yo‘qotadi?

7. Ko‘rsatkich nimaga yo‘naltiriladi?

8. Xotiradan chiqish nima?

9. Resurs nima?

10. Ko‘rsatkich qanday inisializasiyalanadi?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

6-ma’ruza. Vektorlar va massivlar

Reja:

6.1 Nusxa olish.

6.2 Asosiy amallar.

6.3 Vektor elementlariga murojaat.

6.4 Massivlar.

Kalit so‘zlar: *Chuqur nusxalash, Asosiy amallar, massivni initsializatsiya qilish, nusxalovchi konstruktor, palindrom, explicit konstruktori, massiv, yuzaki nusxalash, odatiy konstruktor, nusxalash.*

6.1 Nusxa olish

Vector sinfini 5- ma’ruzadagi kabi ko‘rinishda qaraymiz

```
class vector {
int sz;  //  o‘lchami
double* elem;    //  elementga ko‘rsatkich
public:
vector(int s)    // konstruktor
:sz(s), elem(new double[s]) { /* */ } // xotiradan
// joy ajratyapti
~vector()    // destruktor
{ delete[ ] elem; }  // xotiradan ajratilgan
// joyni bo‘shatyapti
// . . .
};
```

Shunday vektorlar biridan nusxa olamiz

```
    void f(int n)
    {
    vector v(3); // 3 ta elementli vektorni aniqlaymiz
    v.set(2,2.2);    // ikkinchi elementi v[2]=2.2 ga
teng
    vector v2 = v;   // bu amal nima vazifani
bajaryapti?
    // . . .
    }
```

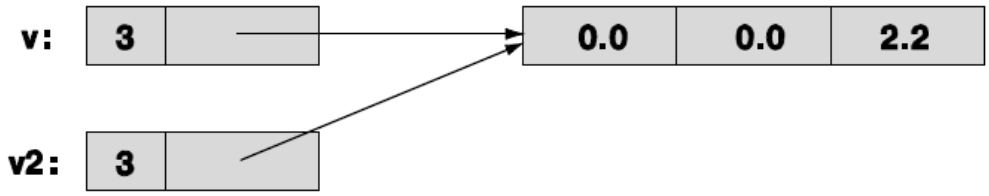
Nazariy jixatdan v2 obyekt v obyektning nusxasi bo‘lishi kerak, boshqacha aytganda barcha i lar [0:v.size()) diapazondagi v2.size()==v.size() va v2[i]==v[i] shartlarni bajarishi kerak. Dastur funksii f() dan chiqqanda barcha foydalanilgan xotira bo‘shatilgan bo‘lishi kerak. Ammo bu ishni standart kutubxonaga tegishli vector sinfi amlga oshiradi, bizning sinf vector emas.

Biznig maqsad vector sinfini mukammallashtirishdan iborat, yuqorida ko‘rsatilgan masalalarni to‘g‘ri yechishi uchun, oldin bu sinfimiz o‘zi qanday ishlashi bilan tanishib chiqamiz.Xatolik nimada nima uchun noto‘g‘ri ishlaydi? Agar buni tushuna olsak, xatolikni to‘g‘rilay olamiz.Eng muximi shundaki biz bu va shunga o‘xshagan muammolarni aniqlay bila olishimiz va shuni amalda boshqa vaziyatlarda xam tatbiq qila olishimizdir.

Odatda sinflarda nusxa olish degani sinfning barcha berilgan a’zolaridan nusxa olishni bildiradi. Masalan Point sinf obyektidan nusxa olish degani nuqtaning koordinatalaridan nusxa olishni bildiradi.Agar sinf a’zolaridan nusxa olayotganda ko‘rsatkich ega obyektlar bo‘lsa, ba’zi muammolar chiqishi mumkin [20].

[20] Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

Xususan vektorlar uchun bizning misoldagi v.sz==v2.sz va v.elem==v2.elem shartlar bajarilsa, vektorlar quyidagicha ko‘rinishda bo‘ladi:



Boshqacha aytganda v2 obyekt v obyektning elementlari nusxasini o‘zida ifoda etmagan; ikkalasi xam birgalikda v obyektdan foydalanayapti. Biz boshqa kod yozishimiz mumkin:

```
v.set(1,99);  // v[1]=99;

v2.set(0,88); // v2[0]=88;

cout << v.get(0) << ' ' << v2.get(1);
```

Bu fragment natijasida 88 99 vektorlar xosil bo‘ladi. Biz bunga intilmagan edik. Agarda v va v2 obyektlar orasida yashirin aloqa bo‘lmaganda edi natija 0 0 bo‘lgan bo‘lar edi, ammo biz bunday qiymatni v[0] yoki v2[1] yacheykaga yozmagan edik.

Siz bunga norozilik qilishingiz mumkin, bunday xolat qiziqarli va ayni paytda foydali, ammo biz buni kutmagan edik, bu aslida standart sinf vector kuzatilmaydi. Bundan tashqari, biz funksii f() dagi natijaga qaytsak, umuman noto‘g‘ri natijani kuzatimiz. Bunda v va v2 obyektlarning destruktorlari oshkormas chaqiriladi ; v obyektning destruktori instruksiya bo‘yicha xotirani bo‘shatadi:

```
delete[] elem;
```

Xuddi shu ishni v2 obyektning destruktori xam amalga oshiradi

Xar ikkala obyekt v va v2 uchun xam elem ko‘rsatkichi xotiraning ayni bitta yacheykasiga murojaat qiladi, bu xotira yacheykasi ikki marta o‘chiriladi va natijada katta muammolarga olib kelishi mumkin.

Nusxalash konstruktori. Nima qilish kerak? Bu oddiy va aniq : nusxa olish amalini qayta ko‘rib chiqish kerak, nusxa olish mumkin bo‘lsin va bitta vektorni ikkinchi vektorda ifodalash imkoni mavjud bo‘lsin. Demak bizga nusxa oluvchi konstruktor kerak ekan. Bunday konstruktor, o‘z o‘zidan ma’lumki nusxa oluvchi konstruktor deyiladi (copy constructor). Argument sifatida u obyektga ko‘rsatkichni qabul qiladi.Demak sinf vector quyidagi ko‘rinishda bo‘ladi:

```
vector(const vector&);
```

Qachonki vector sinfining bitta obyekti boshqa obyektiga inisializasiya qilinayotganda nusxalash konstruktori chaqiriladi.

Biz obyektni ko‘rsatkich orqali uzatamiz, konstruktor argumentini nusxa qilmoqchi emasmiz. Biz bu ko‘rsatkichni const spesifikatori bilan birga uzatamiz, sababi argumentni modifikasiya qilishni xoxlamaymiz. Vector sinfini aniqlaymiz

```
class vector { int sz; double* elem;
void copy(const vector& arg);
// copy elementlarini nusxalaydi
public:
vector(const vector&) ;
// nusxalash konstruktori
// . . .
};
```

copy() funksiya – a’zo vektor elementlar nusxasini oladi

```
void vector::copy(const vector& arg)
// elementlardan nusxa oladi [0:arg.sz-1]
{
for (int i = 0; i<arg.sz; ++i) elem[i] =
arg.elem[i];
```

```
}
```

copy() funksiya a’zo barcha vektor elementlariga murojaat qilish imkoniga ega.Bu ning uchun fu funksiyaning o‘zini yopiq e’lon qildik. Bu funksiyani faqat vector sinfi ichida chaqirish mumkin va bu vektorlar o‘lchamlari mos kelishi kerak.

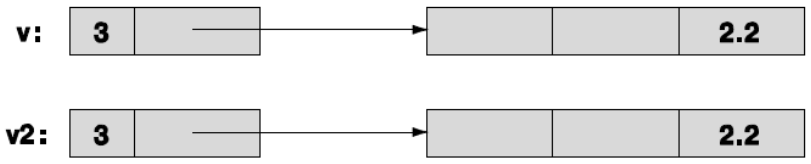
Nusxalash konstruktori elementlar miqdorini aniqlaydi va elementlar uchun xotiradan joy ajratadi.

```
vector:: vector(const vector& arg)
:sz(arg.sz), elem(new double[arg.sz])
{
copy(arg);
}
```

Nusxalash konstruktoriga ega bo‘lgandan keyin dastlabki misolga qaytishimiz mumkin bo‘ladi

```
vector v2 = v;
```

Bu ifoda vector sinfi nusxalash konstruktorini v argument bilan chaqirish xisobiga v2 obyektni inisializasiya qiladi. Agarda vector sinfi uchta elementga quyidagicha muammo paydo bo‘lishi mumkin:



Endi destruktorlar to‘g‘ri ishlaydi.Xar bir element to‘g‘ri o‘chiriladi. Ma’lumki vector sinfining ikkita obyekti endi bir biriga bog‘liq emas, ularning elementlarini xoxlagancha o‘zlashtirishimiz mumkin.

Masalan.

```
v.set(1,99); // v[1] = 99

v2.set(0,88); //v2[0] = 88

cout << v.get(0) << ' ' << v2.get(1);
```

Natijada 0 0.

Instruksiya o‘rniga

```
vector v2 = v;
```

instruksiyani yozishimiz mumkin

```
vector v2(v);
```

Agarda v obyektlari va v obyektlari bir xil turga ega bo‘lsa nusxalash to‘g‘ri amalga oshirilgan bo‘ladi, yuqoridagi ikkita instruksiya ekvivalentdir ularning xar ikkalasi xam to‘g‘ri ishlaydi tanlash foydalanuvchiga xavola.

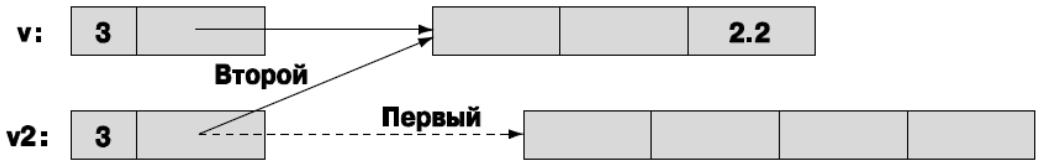
O‘zlashtirishni nusxalash. Vektorlarni nusxalash nafaqat inisializasiyalashda balki o‘zlashtirish xam sodir bo‘lishi mumkin.

Masala qaraymiz

```
void f2(int n)

{

vector v(3); // vektorni aniqlaymiz

v.set(2,2.2);

vector v2(4);

v2 = v;  // o‘zlashtirish : ?

// . . .

}
```

Biz xoxlagandik v2 vektor v vektorning nusxasi bo‘lsin, ammo bizning sinfda nusxalash ma’nosi aniqlanmagan, odatda uning o‘rniga o‘zlashtirish ishlatiladi, boshqacha aytganda o‘zlashtirish a’zoma a’zo bajariladi, v2 obyektning a’zolari sz , elem v obyektning a’zolari sz , elem ga aynan mos keladi.

Buni xolatni quyidagicha ifodalashimiz mumkin:



funksii f2() ishlaganda yana oldingi xolatlar takrorlanishi mumkin, xuddi funksii f() dagi kabi, bunda xam nusxa oluvchi konstruktordan foydalanishimiz mumkin. Bundan tashqari xotirada ba'zi ma'lumotlar o'chirilmay qolib ketishi mumkin. Biz ularni o'chirishni esdan chiqarib qoldirishimiz mumkin. Bu muammoni yechish aslida inisializasiya qilish bilan nusxa olishdan farq qilmaydi.

O'zlashtirish bilan nusxa oluvchi operator ni aniqlaymiz

```
class vector { int sz; double* elem;
void copy(const vector& arg);
public:
vector& operator=(const vector&) ;
};
vector& vector::operator=(const vector& a)
{
double* p = new double[a.sz];
for (int=0; i<asz; ++i)
p[i]=a.elem[i];  // elementlarni nusxalaymiz
delete[] elem;   // xotirani bo'shatamiz
elem = p;    // endi elementni yangilash mumkin
sz = a.sz;
return *this;    //joriy obyektga murojatga
qaytamiz
```

```
}
```

O‘zlashtirish ancha murakkab yaratish va eski elementlar bilan ishlashga qaraganda. Vector sinfidan elementlarni nusxalash asosiy maqsadimiz.

```
double* p = new double[a.sz];
for(int=0; i<asz; ++i) p[i]=a.elem[i];
```

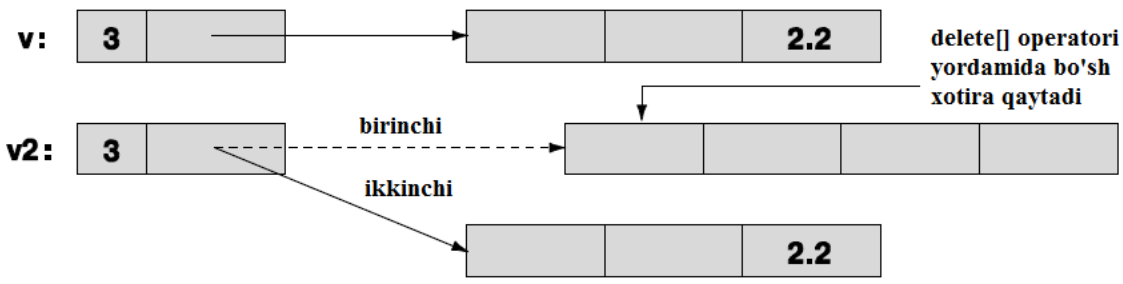
Endi butun klas obyekti vektordan eski elementlarni ozod qilamiz.

```
delete[] elem; // osvobojdayem zanyatuyu pamyat
```

Yakunda ko‘rsatkichni yangi elementga o‘rnatamiz.

```
elem = p;
sz = a.sz;
```

Natijani quyidagicha ko‘rishimiz mumkin.



Endi vector sinfida xotirani yo‘qotish muammosi xal qilindi, endi (delete[]) bir marta ishlaydi.

O‘zlashtirishni qayta ishlash ni soddalashtirish mumkin, xotiradagi eski elementlarni o‘chirish xisobiga, agar aniq ishonch xosil qilmaguncha ma’lumotlarni o‘chirishni maslaxat bermagan bo‘lardik.Bundan tashqari vector sinf obyektini o‘zini o‘ziga o‘zlashtirishda qiziq xolatlar bo‘lishi mumkin, masalan quyidagicha:

```
vector v(10);
v=v;
```

Marxamat qilib tekshirib ko‘ring va ishonch xosil qiling, bizning ishlab chiqgan kodlarimiz to‘g‘ri ishlaydimi?

6.2 Asosiy amallar

Sinf da qanday konstruktorlar bo‘lishi mumkin, albatta sinfda destruktor bo‘lishi shartmi yoki nusxa oluvchi o‘zlashtirish bo‘lishi kerakmi. Quyida 5 ta muxim amallarni qarab chiqamiz

- Bitta yoki bir nechta argumentli konstruktorlar.
- Odatga ko‘ra konstruktor.
- Nusxa oluvchi konstruktor (bitta turga tegishli obyektlar ustida)
- O‘zlashtirish bilan nusxa olish (bitta turga tegishli obyektlar ustida)
- Destruktorlar .

Odatda sinfda bitta yoki bir nechta konstruktorlar bo‘lishi mumkin.

```
string s("Triumf");

vector<double> v(10);
```

Bundan ko‘rinib turibdiki, inisializasiya to‘la ma’noda konstruktor bilan aniqlanadi.Standart string sinfi konstruktori dastlabki qiymat sifatida simvollar satri qabul qilsa, vector sinfi standart konstruktori esa parametr sifatida elementlar miqdorini qabul qiladi. Odatda konstruktordan invariantni o‘rnatish uchun foydalaniladi. Agar biz sinf uchun yaxshi invariant aniqlay olmasak, demak sinf yomon ifodalangan yoki ma’lumotlar strukturasi yaxshi emas degan ma’noni anglatadi [21].

Argumentga ega konstruktorlar sinfga kuchli bog‘liq bo‘ladi, boshqa amallar standart strukturaga bog‘liq.

Odatga ko‘ra konstruktor nima uchun kerak bo‘ladi?

Sinf obyektlarini inisializasiyalamasdan turib yaratish uchun kerak bo‘ladi.

Bunga eng ko‘p uchraydigan misol keltiramiz, vector sinf obyektlarini standart qolipga joylamoqchi bo‘lsak, quyida keltirilgan instruksiyalar faqat int, string va vector<int > lar uchun ishlaydi.

[21] Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

```
vector<double> vi(10);

vector<string> vs(10);

vector<vector< int> > vvi(10);
```

Qachon odat bo‘yicha konstruktor foydali bo‘ladi? Qachon biz sinf invariantini ma’noli qiymatni o‘rnata olsak, masalan, sonli turlar uchun 0 butun son uchun xaqiqiy son uchun 0.0 kabi, string uchun esa “ ”, vector sinfi uchun esa bo‘sh vektorni olish mumkin. Agar T tur uchun odatiy qiymat T() kabi o‘rnatiladi. Masalan, double() = 0.0, string() = "", vector<int>() kabi o‘rnatilishi kerak.

Agar sinf biror ma’lumotga , resursga ega bo‘lsa, albatta u destruktorga ega bo‘lishi kerak.Resurs deganda sinf boshqa obyektlardan olgan ma’lumotlar yoki uzatishi kerak bo‘lgan ma’lumotlar va xakozolarni nazarda tutish kerak bo‘ladi.

Oddiy misol sifatida xotirani olishimiz mumkin, new bilan joy ajratiladi delete[] bilan o‘chiriladi. Vector sinfi xam o‘z elementlarini joylashtirish uchun joy talab qiladi va kerak bo‘lganda bo‘shatib beradi, shunga ko‘ra unga destructor kerak bo‘ladi.

Boshqa resurslar boshqa murakkab dasturlarda ishlatiladi bular - fayllar xisoblanadi(agar ochgan bo‘lsangiz uni albatta bekitish kerak bo‘ladi), bloklash, oqim deskriptorlari(thread handles), ikki tomonlama kanallar(sockets).

Oshkor konstruktorlar . Konstruktor.

Misol qaraymiz.

```
class complex {

public:

complex(double);

complex(double,double);

// . . .

};

complex z1 = 3.18;
```

```
complex z2 = complex(1.2, 3.4);
```

Oshkormas turga keltirish ba’zi muammolarni keltirib chiqarishi mumkin, shuning uchun extiyot bo‘lib foydalanamiz, aks xolda, kutimagan effektlarga duch kelishimiz mumkin.

Masalan vector sinfi yuqorida aniqlangan, int argumentli konstruktorga ega, bundan kelib chiqadi, u int turini vector sinfiga akslantiradi.

Misol qaraymiz.

```
class vector {
// . . .
vector(int);
// . . .
};
vector v = 10;
v = 20;
void f(const vector&);
f(10);
```

Nazarimda biz bu misoldan kutganimizdan ko‘p ma’lumot oldik. Bunday oshkormas akslantirishni inkor qilish juda oson explicit kalit so‘zli konstruktor ishlatiladi.

Misol qaraymiz.

```
class vector {
// . . .
explicit vector(int);
// . . .
};
vector v = 10;
```

```
v = 20;
vector v0(10); // OK
void f(const vector&);
f(10);
f(vector(10)); // OK
```

Kutilmagan akslantirishlar bo‘lmasligi uchun biz til standartida mavjud instrumentdan foydalandik.Ammo barcha konstruktorlar spesifika-tor explicit ega emaslar. Agar shubxa qilsangiz tekshirib ko‘ring.

6.3 Vektor elemenlariga murojaat

Xaligacha vektor elementlariga murojaat qilish uchun set() va get(). funksiya a’zolardan foydalanamiz. Bu usul ba’zi noqulayliklar keltirib chiqarishi mumkin.Biz oddiy indeksasiyani xoxlaymiz: v[i]. Bunga ko‘ra operator[] kabi funksiya a’zoni aniqlaymiz, eng sodda variantini qaraymiz:

```
class vector {
int sz;  // o‘lcham
double* elem; // ko‘rsatkichna elementlari
public:
// . . .
double operator[](int n) { return elem[n]; }
};
```

Xammasi yaxshi va oddiy ammo juda sodda. Indekslashni operator[]()) va o‘qishni to‘g‘riladik ammo yozishni qanday to‘g‘rilash mumkin.

```
vector v(10);
int x = v[2]; // yaxshi
v[3] = x;
```

Bu yerda v[i] ifoda xuddi v.operator[](i) operatorni chaqirishni interpretasiya qiladi, xuddi vektorning I- elementini qaytaradi. Bunday xolatda vector sinfi ning v[3] qiymati suzuvchi vergulli son deb o‘zgaruvchi deb tushunilmaydi.

Keyingi versiyada biz operator[] ga ko‘rsatkichni mos elementga qaytarishni yuklaymiz:

```
class vector {
int sz;  // o‘lcham
double* elem; // elementga ko‘rsatkich
public:
// . . .
double* operator[](int n) { return &elem[n]; }
};
```

Bunday xolda biz endi elementlarni yoza olamiz

```
vector v(10);
for (int i=0; i<v.size(); ++i) {
*v[i] = i;
cout << *v[i];
}
```

Bu yerda v[i] ifoda xuddi v.operator[](i) operatorni chaqirishni interpretasiya qiladi va ko‘rsatkichni v[i] indeksli elementga qaytaradi.

Endi biz operator * ni ifodalovchi kodni yozishimiz kerak, ya’ni ko‘rsatkich nomini almashtiradigan va elementga murojaat qiladigan xolda. Bu muammoni ko‘rsatkich indekslarini operatordan qaytarish bilan amalga oshirish mumkin bo‘ladi.

```
class vector {
// . . .
double& operator[ ](int n) { return elem[n]; } };
```

Endi navbatdagi variant kodni yozishimiz mumkin

```
vector v(10);
for (int i=0; i<v.size(); ++i) {
v[i] = i;
cout << v[i];
}
```

Biz odatiy va keng tarqalgan belgilashlarni ta’minladik: bu yerda v[i] ifoda xuddi v.operator[](i) operatorni chaqirishni interpretasiya qiladi va ko‘rsatkichni v[i] indeksli elementga qaytaradi.

6.4 Massivlar

Xaligacha massiv deganda xotiradagi ketma ket obyektlar degan ma’noni anglashadi. Ammo aslida massiv ixtiyoriy joyda joylashgan bo‘lishi mumkin.Ular quyidagicha foydalanishi mumkin.Massivlardan qayerlarda foydalanish mumkin.

- Global o‘zgaruvchi sifatida (ammo juda yaxshi g‘oya emas)
- Lokal o‘zgaruvchi sifatida (ammo juda yaxshi g‘oya emas)
- Funksiya argumenti sifatida(massiv ulchami ma’lum emas)
- Sinf a’zosi sifatida (ammo bunda inisializasiya murakkab bo‘lishi mumkin)

E’tibor bergan bo‘lsangiz kerak vector sinfiga katta e’tibor bergan edik, shuning uchun xam imkon bulgan vaqtda biz std::vector dan foydalanishimiz mumkin bo‘ladi. Ammo massivlar dasturlashda std::vector dan oldin paydo bo‘lgan va ko‘pgina tillarda u massiv protitipi xolda kirib kelgan. Demak bu ikkita tushuncha ikki xil yondashuv eski va yangi dasturlash usullari bilan paydo qilingan deyish mumkin.

Demak, massiv - bu fiksirlangan miqdordagi ayrim qiymatlar-ning (massiv elementlarining) tartiblangan majmuasidir. Barcha elementlar bir xil turda bo‘lishi kerak va bu tur *element turi* yoki massiv uchun *tayanch tur* deb nomlanadi. Dasturda ishlatiladigan har bir konkret massiv o‘zining individual nomiga ega

bo‘lishi kerak. Bu nomni *to‘liq o‘zgaruvchi* deyiladi, chunki uning qiymati massivning o‘zi bo‘ladi. Massivning har bir elementi massiv nomi, hamda kvadrat qavsga olingan va *element selektori* deb nomlanuvchi indeksni ko‘rsatish orqali oshkor ravishda belgilanadi. Murojaat sintaksisi:

```
<massiv nomi >[<indeks>]
```

Bu ko‘rinishga *xususiy o‘zgaruvchi* deyiladi, chunki uning qiymati massivning alohida elementidir.Masalan,

const int max = 100;

```
int gai[max];

void f(int n)

{

char lac[20];

int lai[60];

double lad[n];

}
```

E’tibor qiling elementlar miqdori kompilyasiya etapigacha ma’lum bo‘lishi kerak, agar noma’lum bo‘lsa ko‘rsatkich bilan aniqlashimiz kerak bo‘ladi.Bu xolda vector sinfi elementlaridan massiv xosil qilinadi.Qanday qilib massiv elementlari ‘otiraga joylashishini misol orqali ko‘rsatamiz

Masala qaraymiz.

```
void f2()

{

char lac[20];

lac[7] = 'a';

*lac = 'b';

lac[-2] = 'b';   //   ??
```

```
lac[200] = 'c';  //  ??
}
```

Bu funksiya kompilyasiya qilinadi, ammo ma’lumki barcha funksiya xam to‘g‘ri ishlayvermaydi, [] operatordan foydalanildi ammo u massiv chegaralaridan chiqib ketmasligi tekshirilmadi, shuning uchun xam bu f2() funksiya ishlaydi ammo natijada lac[–2] va lac[200] yozuvlar dasturni salbiy natijalarga olib keladi. Xar doim massiv o‘lchamlari xaqida gap ketganda dasturchi o‘lchamlar masalasini o‘zi aniqlagan va tekshirgan bo‘lishi zarurligini aytib o‘tamiz.

Bu ishlarni dasturchi o‘rniga kompilyator aniqlashi mumkinmi, massiv lac faqat 20 ta element olishi mumkinligi, bunday lac[200] bo‘lishi mumkin emasligini?

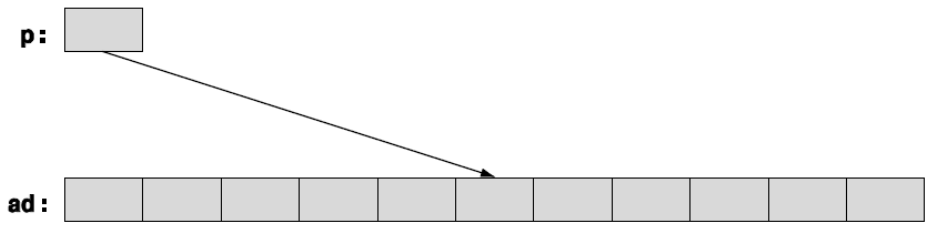
Ammo aslida bu nazariy jixatdan mumkin ish ammo xozirgacha bitta xam shunday kompilyator mavjud emas. Muammo shundaki, massiv chegaralari kompilyasiya davomida aniqlash mumkin emas, xatoni topish esa barcha muammolarni yecha olmaydi.

Massiv elementlariga ko‘rsatkich. Ko‘rsatkich massiv elementiga murojaat qilishi mumkin.

Masala qaraymiz.

```
double ad[10];
double* p = &ad[5];
```

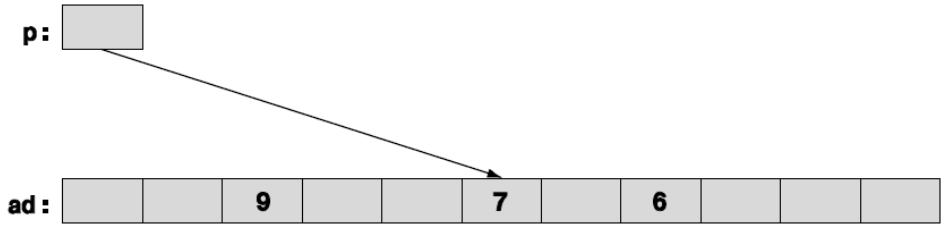
Qo‘rsatkich p o‘zgaruvchi double turiga murojaat qilyapti, rasmdan ma’lumki ad[5] elementga.



Ko‘rsatkich ni indekslash va qayta nomlash.

```
*p =7; p[2] = 6; p[-3] = 9;
```

U xolda natija quyidagicha bo‘ladi



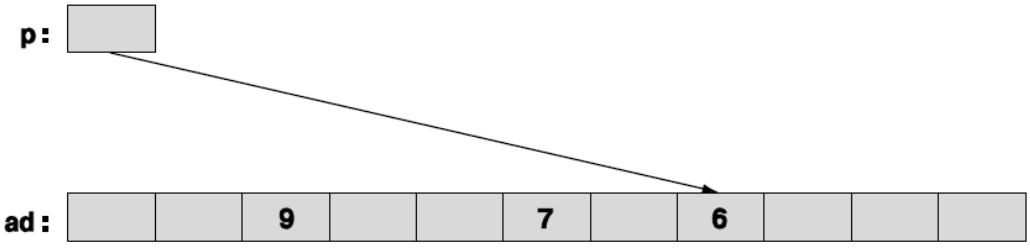
Boshqacha qilib aytsak biz ko‘rsatkichni musbat va manfiy sonlar bilan indekslashimiz mumkin. Agar ko‘rsatkich indekslari ko‘rsatilgan diapozondan chiqib ketmasa barchasi to‘g‘ri bo‘ladi.Agar diapozondan chiqib ketsa uni kompilyator aniqlay olmaydi ertami kechmi albatta dastur xato ishlay boshlaydi.

Agar ko‘rsatkich massivning ichki elementlarini ko‘rsatib turgan bo‘lsa, , uni boshqa elementlarga o‘tishini qo‘shish yoki ayirish amali orqali bajarish mumkin bo‘ladi.

Masala qaramiz.

```
p += 2;
```

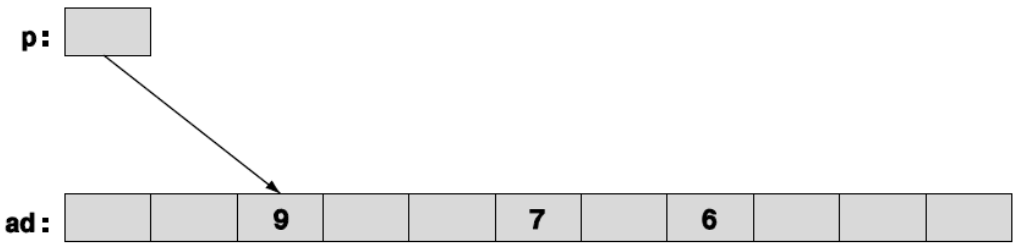
Bu xolda quyidagi xolatga kelamiz.



Xuddi shuningdek,

```
p -= 5;
```

Natijada quyidagiga kelamiz



Quyidagi amallar +, –, += va –= ko‘rsatkichni siljitish uchun xizmat qiladi, bu amallarni ba’zida ko‘rsatkichlar ustida amallar (pointer arithmetic) deb

atashadi. Ammo bunda massivlar ustida ishlayotganda massiv chegarasidan chiqib ketishiga extiyot bo‘lish kerak bo‘ladi.

```
p += 1000;

double d = *p;

*p = 12.34;
```

Afsuski, ko‘rsatkichli arifmetika sababli xatoliklarni tezda bartaraf qilish mumkin. Iloji boricha ko‘rsatkichli arifmetikadan foydalanmaslik kerak.

Eng keng tarqalgan ko‘rsatkichli arifmetik amal inkrement(++) va dekrement(--)

Masalan ad massiv elementlarini chiqazish uchun quyidagicha kod yoziladi:

```
for (double* p = &ad[0]; p<&ad[10]; ++p) cout << *p << '\n';
```

Va teskari tartibda :

```
for (double* p = &ad[9]; p>=&ad[0]; --p) cout << *p << '\n';
```

Bunday ko‘rsatkichli arifmetikadan kamdan kam xolda foydalaniladi.

Ammo bizning fikrimizcha oxirgi misolda teskari tartibda xafvsizlik xollari ta’minlanmagan.

Nima uchun &ad[9] bunday ifodalangan &ad[10] day emas?

Nima uchun >= bunday , > day emas?

Bu misollar bir xil yaxshi, agar indeksasiyalashdan foydalansak. Ular vector sinfi bilan ekvivalent diapozondan tashqariga chiqishini tekshirish oson xal qilingan.

Eslatib o‘tamiz ko‘pgina real dasturlarda ko‘rsatkich arifmetikasi funksiya argumentiga ko‘rsatkichni uzatish bilan bog‘liq bo‘ladi.

Bu xolda kompilyator bilmaydi ko‘rsatkich qancha elementni uzatayotganini buni dasturchi tekshirishi kerak bo‘ladi. Qo‘ldan kelgancha xotolik noma’lum xolatlardan qochish kerak bo‘ladi

Nima uchun C++ da ko‘rsatkich arifmetikasi ruxsat berilgan? Ammo bu lar yordamida indeksasiyalash ishlarini muammosiz amalga oshirish mumkin.

Misol qaraymiz.

```
double* p1 = &ad[0]; double* p2 = p1+7; double* p3 = &p1[7];
```

```
if (p2 != p3) cout << "impossible!\n";
```

Asosan bu til tarixi bilan bog‘liq, ko‘rsatkich arifmetikasi qoidalari S tilidan olingan, buni yo‘qqa chiqarish xozircha mumkin emas, chunki juda ko‘plab kutubxonalarni ishlab chiqish lozim bo‘ladi. Ko‘rsatkich arifmetikasi qoidalari quyi darajali dasturlash tillari va xotirani boshqarish bilan bog‘liqligi va ba’zi xollarda ustunliklarga ega bo‘lganligi uchun xam voz kechilmayapti.

Ko‘rsatkich va massivlar. Massiv nomi barcha elementlarga tegishli bo‘ladi

Misol qaraymiz.

```
char ch[100];
```

sh massiv o‘lchami sizeof(ch), 100 teng bo‘ladi.Massiv nomi ba’zi sabablarga ko‘ra ko‘rsatkichga aylanadi.

```
char* p = ch;
```

Bu yerda ko‘rsatkich p &ch[0] adres bilan inisializasiya qilinadi, va o‘lchami sizeof(p) 4 ga teng bo‘ladi(100 emas). Bu xususiyat foydali bo‘lishi mumkin.Masalan, funksiyu strlen() qaraydigan bo‘lsak, massivdagi simvollarni sanaydigan, oxiri nol bilan tugaydigan.

```
int strlen(const char* p)
{
int count = 0;
while (*p) { ++count; ++p; }
return count;
```

```
}
```

Endi uni strlen(ch) ko‘rinishda yoki strlen(&ch[0]) ko‘rinishda chaqirish mumkin

Bunday funksiya bizga ko‘plab imkoniyatlar keltirishi mumkin bo‘ladi.

Bunga bitta sabab massiv ismi ko‘rsatkichga aylanishi mumkin bo‘ladi, katta xajmdagi berilganlarni qiymati bo‘yicha uzatishdan qochish imkonini beradi.

Misol qaraymiz.

```
int strlen(const char a[])
{
int count = 0;
while (a[count]) { ++count; }
return count;
}
char lots [100000];
void f()
{
int nchar = strlen(lots);
// . . .
}
```

Bu kod bajarilganda 100 ming simvol nusxa qilingan bo‘lardi, ammo argument sifatida strlen() funksiya ishlatilsa, bunday bo‘lmaydi.

char p[ ] argument e’loni o‘rniga unga ekvivalent char* p e’londan foydalanamiz, strlen(lots) funksiya o‘rniga unga ekvivalent strlen(&lots[0]) funksiyadan foydalanamiz.Bu ko‘p mexnatli nusxalash ammo ishonchli va xavfsiz.

E’tibor qiling massiv nomidan iborat ko‘rsatkich birinchi elementga o‘rnatilgan o‘zgaruvchiga emas,

```
char ac[10];

ac = new char [20];

&ac[0] = new char [20];
```

Bu muammoni kompilyator aniqlashi kerak bo‘ladi. Oshkormas xolda massiv nomini ko‘rsatkichga berishimiz oqibatida o‘zlashtirish operatori bilan xam massivdan nusxa ola olmaymiz.

```
int x[100];

int y[100];

// . . .

x = y;   // xatolik

int z[100] = y;  // xatolik
```

Bu kodni yozish mantiqan oson emas. Agar massivni nusxalash kerak bo‘lsa, murakkab kod yozishga to‘g‘ri keladi.

Misol qaraymiz.

```
for (int i=0; i<100; ++i) x[i]=y[i];

memcpy(x,y,100*sizeof(int));

copy(y,y+100, x);
```

S tilida vektor tushunchasi yo‘q, uning o‘rniga bir o‘lchovli massiv ishlatiladi. S++ tilida yozilgan juda ko‘plab dasturlarda massivlardan foydalaniladi. S stilidagi satrlardan S++ tilida foydalaniladi ammo juda ko‘plab muammolar keltirib chiqaradi.

Agar nusxa olmoqchi bo‘lsangiz vector sinfi kabi sinflardan foydalaning aks xolda qo‘shimcha kodlar yozishga to‘g‘ri keladi.

Misol qaraymiz:

```
vector<int> x(100);

vector<int> y(100);
```

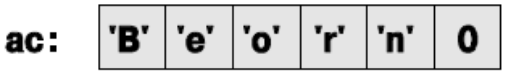
```
// . . .

x = y;
```

Massivni inisializasiya qilish. Massiv vektor va shunga o‘xshash konteynerlardan bitta ustunlikka ega bu bo‘lsa S++ tilida massivlarni inisializasiya qilish imkoniyati mavjudligidir.

Misol qaraymiz.

```
char ac[] = "Beorn";
```

Simvollarni sanang, ular 5 ta ammo tasvirda oxirida yakunlovchi nol simvolini qo‘shib qo‘ygan, chunki satrli literalar uchun kompilyator nol qo‘shib qo‘yadi.

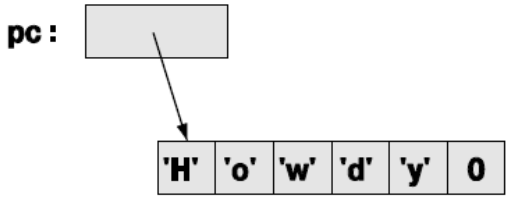


S stilidagi satr nol bilan tugashi kerak. Agar bu qoidani ko‘rsak bas demak *S* (C-style string) deb tushunishimiz kerak bo‘ladi.

Misol qaraymiz.

```
char* pc = "Howdy";
```

Grafik ko‘rinishda buni quyidagicha ifodalshimiz kerak.



Char ko‘rinishidagi o‘zgaruvchi qiymati 0 ga teng bu son, xarf, simvol emas. Bu belgining ma’nosi satr yoki massiv oxirini kompilyatorga anglatishdan iborat.

Satr va massiv oxirini bildiruvchi nol ni anglatish uchun quyidagi kodni keltiramiz:

```
int strlen(const char* p)
```

```
{
int n = 0;
while (p[n]) ++n;
return n;
}
```

Aslida bizga strlen() funksiyani aniqlashga xojat yo‘q, bu standart <string.h> kutubxonada aniqlangan test qilingan funksiyadir.E’tibor qilgan bo‘lsangiz strlen() funksiya satr simvollarini sanaydi ammo yakunlovchi nolni xisoblamaydi.

Faqat simvolli massivlar literal konstantalar bilan inisializasiya qilinishi mumkin, massiv turlari mos kelishini xisobga olish kerak bo‘ladi.

Misol qaraymiz.

```
int ai[] = { 1, 2, 3, 4, 5, 6 };
int ai2[100] = { 0,1,2,3,4,5,6,7,8,9 };
double ad[100] = { };
char chars[] = { 'a', 'b', 'c' };
```

E’tibor qiling ai massiv elementlari soni 6 ga teng 7 emas, chars massivida esa 3 ga teng 4 emas, oxiriga nolni qo‘shib qo‘yish faqat simvolli satrlarga xosdir.Agar massiv o‘lchami oshkor bo‘lmasa, u xolda inisializasiya ro‘yxatiga ko‘ra aniqlanadi. Bu ancha foydali qoida.Agar elementlar miqdori ro‘yxatda kam bo‘lsa, qolgan elementlar xuddi ai2 va ad massivdagi kabi to‘ldiriladi.

Nazorat savollari

1. “Xaridor, hushyor bo‘lishi lozim!” ifodasi nimani anglatadi?
2. Qanday sinf obyektlarini nusxalash odatdagidek ishlatiladi?
3. Nusxalash konstruktori nima?
4. Yuzaki nusxalash va chuqur nusxalash nima?
5. Sinflar bilan bog‘liq 5 ta asosiy amallarni sanang?
6. Explicit kalit so‘zidagi konstruktor nimani taqdim etadi?

7. Massiv nima?
8. Massivni qanday nusxalash mumkin?
9. Qanday qilib massivni initsializatsiya qilinadi?
10. Palindrom nima?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://www.cplusplus.com/

IV. BO`LIM

AMALIY MASHG`ULOT MATERIALLARI

IV. AMALIY MASHG‘ULOT MATERIALLARI

1-amaliy mashg‘ulot. C++ tilining obyektlari, qiymat va toifalarini o‘zlashtirish. Birinchi oddiy dasturni tashkil etish.

Ishdan maqsad: C++ tilining obyektlari, qiymat va toifalarini o‘rganish xamda dasturni shakllantirishda ulardan foydalanish. Dastur bajarilishi davomida yuzaga keladigan xatoliklarni bartaraf etishni o‘rganish. Berilgan topshiriqlarni dasturiy yechim to‘g‘riligini avtomtik testlovchi tizim (http://acm.tuit.uz) yordamida tekshirish.

Masalaning qo‘yilishi: C++ tilida “Hello, World!” matnli xabarini chiqarish dasturi tuzilsin. Variant bo‘yicha chiziqli algoritmga oid dasturlarni yaratib, ularni http://acm.tuit.uz saytidan ro‘yxatdan o‘tgan xolatda tekshirilsin.

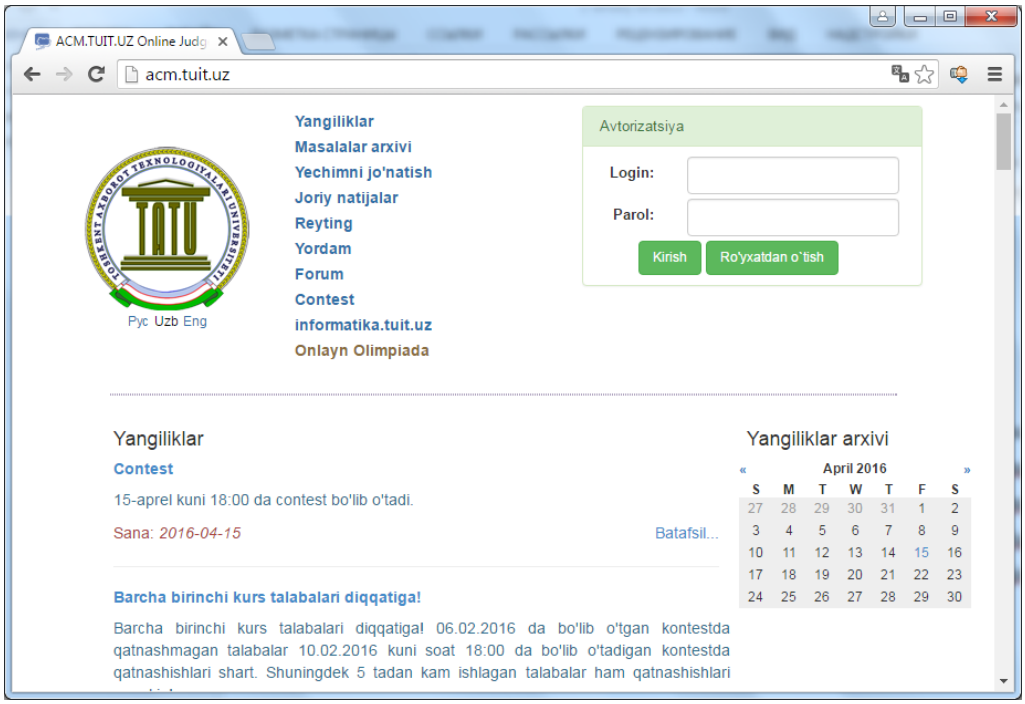
Ishni bajarish uchun namuna

Oddiy ko‘rinishdagi “Hello, World!” habarini chiqaruvchi dastur.

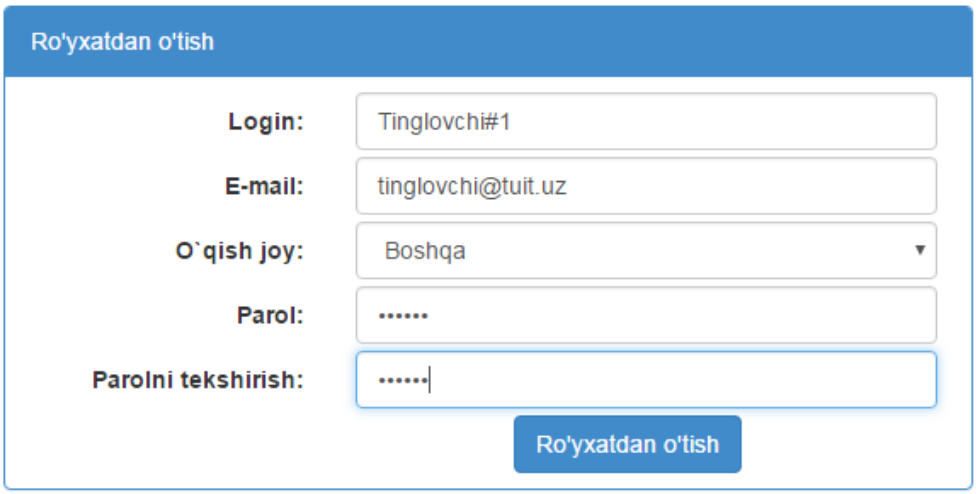
```
#include "std_lib_facilities.h"
int main() // C++ da dasturlash main funksiyasi yordamida
          //amalga oshiriladi
{
cout << "Hello, World!\n"; // "Hello, World!" chiqarish
return 0; // Null qiymat qaytarish
}
```

Dasturiy yechim to‘g‘riligini avtomatik testlovchi tizim (http://acm.tuit.uz) dan ro‘yhatdan o‘ting va murakkab bo‘lmagan masalalarni dasturini tuzib, tizimga yuboring. Tizim qabul qilmagan taqdirda yuzaga kelgan hatoliklarni bartaraf eting.

Buning uchun biz internet baruzerni ishga tushirib uning adresslar qatoriga [22] http://acm.tuit.uz manzilini kiritamiz. Shundan so‘ng dasturiy yechim to‘g‘riligini avtomatik testlovchi tizim ishga tushadi:



Tizim asosiy oynasida joylashgan “Ro‘yxatdan o‘tish” tugmasini bosamiz. Shundan so‘ng quyidagi oyna xosil bo‘ladi:



Yuqoridagi rasmda ko‘rsatilgan namuna bo‘yicha foydalanuvchi logini, electron pochta manzili, o‘qish joyi (agarda TATU o‘qituvchisi yoki talabasi bo‘lsangiz Universitetni tanlaysiz, aks holda Boshqa qolgani maqul) va parolni kiritish lozim.

[22] http://acm.tuit.uz/ - дастурий ечим тўғрилигини автоматик тестловчи тизим.

Tizim bilan ishlash bo‘yicha tavfsiyalar

Aktivlashish.

1. Sizga administrator tomonidan login va parol beriladi. Login va parolingizni yozing hamda *Kirish* tugmasini bosing.
2. Login va parol orqali tizimga kirganinggizdan keyin to‘rtta *Masalalar ro‘yxati, Masalani jo‘natish, Natijalar va Monitor* giperko‘rsatkichlari paydo bo‘ladi. (Bu giperko‘rsatkichlarni login va parol yozmasdan oldin ham ko‘rish mumkin, lekin ayrim funksiyalari ishlamaydi.)

Masala shartlarini ko‘rish

Masalalarni *Masalalar arxivi* giperko‘rsatkichini bosgan holda ko‘rishingiz mumkin. Bu masalalar turli mavzularga oid va turli qiyinchilik darajalarida berilgan.

Masalani yechimini tekshirish uchun jo‘natish

1. Masala yechimini jo‘natish uchun *Masalani jo‘natish* bo‘limiga o‘ting.
2. Ishlagan masalangizni nomini **Masala** combo box sidan tanlang.
3. Masalani ishlagan dasturiy tilini *Dasturlash tili* combo box sidan tanlang.
4. “Выберите файл” tugmasini bosing va yechim (faqatgina *.cpp. *.c, *.pas. *.dpr, *.java kengaytmali fayllarni) faylini tanlang. Aks holda sizga jarima ballari beriladi.
5. *Jo‘natish* tugmasini bosing.
6. Masala yechimi faylini jo‘natganinggizdan keyin jo‘natilganlikni tasdiqlovchi ma'lumot chiqadi va Siz avtomatik ravishda *Natijalar* sahifasiga o‘tasiz.
7. Hakamlar tominidan masala yechimi natijalarini kutiing. Kutish davomida Siz boshqa masalani yechishingiz mumkin.

Nazorat savollari

1. acm.tuit.uz saytida masala yechimini yuborganimizda qanday xatoliklar yuzaga kelishi mumkin?
2. Yechim fayli deganda nimani tushunasiz?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. http://acm.tuit.uz/forum/

2-amaliy mashg‘ulot. C++ dasturlash tilida kiritish va chiqarish.

Ishdan maqsad: C++ dasturlash tiling kiritish va chiqarish operatorlari, ulardan foydalanish usullari va masala algoritmi, xususiyatlaridan kelib chiqib kiritish chiqarish operatorlarini tanlash bo‘yicha amaliy ko‘nikma xosil qilish.

Masalaning qo‘yilishi: Sizga AM/PM formatdagi soat vaqti berilgan. Sizning vazifangiz shu vaqtni 24 soatlik formatga o‘girishdan iborat.

Eslatma: 24 soatlik formatdagi 00:00:00 vaqti AM/PM formatda 12:00:00AM ko‘rinishida beriladi, 12:00:00PM ko‘rinishida emas.

Ekranga ma’lumot chiqarish uchun cout so‘zini, undan so‘ng chiqarish operatorini(<<) kiritish lozim. C++ kompilyatori (<<) belgisini bitta operator deb qaraydi. Quyidagi dasturni tahlil qilamiz:

```
#include <iostream>
using namespace std;
int main(){
    cout << "Salom";
    return 0;
}
```

Dastur natijasi:

```
Salom
```

Bu dastur ishlatish natijasida biz ekranda “Salom” yozuvini ko‘rishimiz mumkin.

Klaviaturadan ma’lumot kiritish uchun cin so‘zini va undan so‘ng kiritish operatorini(>>) kiritish lozim. Quyidagi dasturni tahlil qilamiz:

```
#include <iostream>
using namespace std;
int main(){
    int a;
```

```
    cout << "a = ";
    cin >> a;
    cout << "a^2 = " << a*a;
    return 0;
}
```

Dastur natijasi:

a = 3

a^2 = 9

Bu dastur ishlatish natijasida biz ekranda “a = ” yozuvini ko‘rishimiz mumkin. Biz esa a ga qiymat sidatida “3” kiritsak dastur bizga a ning kvadrati 9 (“a^2 = 9”) ekanligi chiqarib beradi.

cin va cout orqali biz ixtiyor tipdagi ma’lumotlarni kiritish va chiqarish mumkin, kiritish va chiqarish esa yuqoridagi kabi amalga oshiriladi.

string tipidagi ma’lumotlarni kiritish uchun cin dan foydalanishga misol:

```
#include <iostream>
using namespace std;
int main(){
    string fio;
    cout << "Familya, ism va otangizni ismini kiriting: ";
    cin >> fio;
    cout << fio;
    return 0;
}
```

Dastur natijasi:

Familya, ism va otangizni ismini kiriting: Abdukarimov Sirojiddin Sayfiddin o‘g‘li
Abdukarimov

Bu dasturda cin birinchi kelgan probel yoki entergacha bo‘lgan ma’lumotni o‘qidi. Shuning uchun bizning “fio” o‘zgaruvchimizning qiymati “Abdukarimov”ga teng bo‘lib qoldi [23].

Biz string tipidagi o‘zgaruvchiga to‘liq satrdagi ma’lumotni kiritish uchun getline kalit so‘zidan foydalanamiz.Quyidagi dasturni ko‘rib chiqamiz:

```
#include <iostream>
using namespace std;
int main(){
    string fio;
    cout << "Familya, ism va otangizni ismini
kiriting: ";
    getline(cin, fio);
    cout << fio;
    return 0;
}
```

Dastur natijasi:

```
Familya, ism va otangizni ismini kiriting: Abdukarimov Sirojiddin Sayfiddin o‘g‘li
Abdukarimov Sirojiddin Sayfiddin o‘g‘li
```

Kiritish va chiqarishni endi scanf va printf kalit so‘zlari orqali bajarib ko‘ramiz.

scanf (”format”, & o‘zgaruvchilar);

Agar klaviatura orqali birorta o‘zgaruvchiga qiymat kiritish kerak bo‘lsa, u holda o‘zgaruvchi nomi oldiga &-ampersand belgisi qo‘yiladi.

Masalan:

scanf(”%d”, &a), bu yerda a butun toifadagi o‘zgaruvchi.

scanf(”%f”, &a), bu yerda a haqiqiy toifadagi o‘zgaruvchi.

scanf(”%I64d %f %s’’, &a, &b, &c) – bu yerda a katta butun o‘zgaruvchi, b haqiqiy o‘zgaruvchi, va s qator toifali o‘zgaruvchi.

[23] http://acm.tuit.uz/ - дастурий ечим тўғрилигини автоматик тестловчи тизим.

Chiqarish operatsiyasi. Natijani chiqarish uchun quyidagi protsedura ishlatiladi. printf(“%f %d”, x, y) printf(“x=%f y=%d”, x,y). Demak, natijani ekranga chiqarishning bir necha ko‘rinishi mavjud ekan. Agar format oxiriga \n belgisini qo‘ysak, natijani chiqargandan keyin kursor yangi satrga o‘tadi, aks holda kursor turgan yerida qoladi. \n belgisini format boshiga ham qo‘ysa bo‘ladi. Masalan: printf(‘’x=%d y=%ld\n‘’, x,y) printf(‘\n yig‘indi=%f‘’, s).

Scanf va printfda ma’lumotlarni formati wuyidagi bo‘ladi:

Bool	%d
short int	%d
Int	%d
long long	%lld
__int64	%I64d
Float	%f
Double	%lf
Char	%c
char[]	%s

Belgili tipdagi ma’lumotlarni kiritish va chiqarish uchun “getchar” va “putchar”dan foydalaniladi.

Quyida ularning ishlatilishini ko‘rib chiqamiz:

```
#include <iostream>
using namespace std;
int main(){
    char a;
    cout << "Belgini kiriting: ";
    a = getchar();
    cout << "Siz kiritgan belgi: ";
    putchar(a);
    return 0;
}
```

Dastur natijasi:

```
Belgini kiriting: s
Siz kiritgan belgi: s
```

getchar probel va enterni ham o‘qiydi.

Undan tashqari belgili massivlarni kiritish va chiqarish uchun gets va puts dan foydalanamiz. gets char tipidagi massivni kiritish uchun ishlatiladi, u to‘liq satrdagi ma’lumotlarni char massivga o‘qib beradi. Quyida ularga misol ko‘rishimiz mumkin:

```
#include <iostream>
using namespace std;
int main(){
    char a[111];
    cout << "Matn kiriting: ";
    gets(a);
    cout << "Siz kiritgan matn: ";
    puts(a);
    return 0;
}
```

Dastur natijasi:

```
Matn kiriting: XXI asr axborot texnologiyalari asri.
Siz kiritgan matn: XXI asr axborot texnologiyalari asri.
```

Ma’lumotlarni fayldan kiritish va faylga chiqarish. Fayldan ma'lumotlarni o‘qish yoki yozish uchun ochish fopen funksiyasi orqali amalga oshiriladi.

FILE * fopen (const char * filename, const char * mode);

filename - o‘zgaruvchisi char toifasidagi satr bo‘lib, faylning to‘liq nomini ko‘rsatishi lozim (filename = "D:\c++\misol.txt"). Agar faylning faqat nomi ko‘rsatilgan bo‘lsa, fayl joriy katalogdan qidiriladi (filename = "misol.txt").

mode - o‘zgaruvchisi ham char toifasidagi satr bo‘lib, faylni qaysi xolatda ochish lozimligini bildiradi. mode qiymati faylning ochilish xolati faylni yozish

uchun ochish. falename o‘zgaruvchisida ko‘rsatilgan fayl hosil qilinadi va unga ma'lumot yozish mumkin"w" bo‘ladi. Agar fayl oldindan bor bo‘lsa (ya'ni oldin hosil qilingan bo‘lsa), faylning ma'lumotlari o‘chiriladi va yangi bo‘sh fayl faqat yozish uchun ochiq holda bo‘ladi. Fayl o‘qish uchun ochiladi. Agar fayl oldindan mavjud bo‘lmasa,"r" xatolik sodir bo‘ladi. Ya'ni ochilishi lozim bo‘lgan fayl oldindan hosil qilingan bo‘lishi shart.

Faylga yangi ma'lumotlar qo‘shish - kiritish uchun ochiladi. "a"Yangi kiritilgan ma'lumotlar fayl oxiriga qo‘shiladi. Agar fayl oldindan mavjud bo‘lmasa, yangi fayl hosil qilinadi. Yozish va o‘qish uchun faylni ochish. Agar fayl oldindan bor bo‘lsa (ya'ni oldin hosil qilingan bo‘lsa), faylning ma'lumotlari "w+"o‘chiriladi va yangi bo‘sh fayl yozish va o‘qish uchun ochiqholda bo‘ladi. "r+"Oldindan mavjud bo‘lgan faylni o‘qish va yozish uchun ochish. Fayl ma'lumotlarni o‘qish va yangi ma'lumot qo‘shish uchun "a+" ochiladi. fseek, rewind faylni ochishda xatolik sodir bo‘lsa, fopen funksiyasi NULL qiymat qaytaradi.

Ochilgan faylni yopish uchun fclose funksiyasi ishlatiladi.

int fclose (FILE * stream);

Faylni yopishda xato sodir bo‘lmasa, fclose funksiyasi nol qiymat qaytaradi. Xato sodir bo‘lsa, EOF - fayl oxiri qaytariladi.

Faylga ma'lumot yozish va o‘qish size_t fread (void * ptr, size_t size, size_t n, FILE * stream); fread funksiyasi, fayldan ptr ko‘rsatkichi adresiga size xajmdagi ma'lumotdan n tani o‘qishni amalga oshiradi. Agar o‘qish muvoffaqiyatli amalga oshsa fread funksiyasi o‘qilgan bloklar soni n ni qaytaradi. Aksholda nol qaytariladi size_t fwrite (const void * ptr, size_t size, size_t n, FILE * stream); fwrite funksiyasi, faylga ptr ko‘rsatkichi adresidan boshlab size xajmdagi ma'lumotdan n tani yozishni amalga oshiradi.

fopen orqali ochilgan fayldan ma’lumot kiritish va chiqarishga misol:

```
#include <stdio.h>
using namespace std;
int main(){
    FILE *f;
```

```
    f = fopen("misol.in", "r");
    int a, b;
    fscanf(f, "%d %d", &a, &b);
    fclose(f);
    f = fopen("misol.out", "w");
    fprintf(f, "%d", a + b);
    fclose(f);
    return 0;
}
```

Yuqorida qo‘yilgan masalani bajarish uchun namuna

Sizga AM/PM formatdagi soat vaqti berilgan. Sizning vazifangiz shu vaqtni 24 soatlik formatga o‘girishdan iborat.

Eslatma: 24 soatlik formatdagi 00:00:00 vaqti AM/PM formatda 12:00:00AM ko‘rinishida beriladi, 12:00:00PM ko‘rinishida emas.

Kiruvchi ma'lumotlar: AM/PM formatdagi vaqtni hh:mm:ssAM yoki hh:mm:ssPM shaklida kiritiladi. $01 \leq hh \leq 12$

Chiquvchi ma'lumotlar: Berilgan vaqtni 24 soatlik formatda hh:mm:ss shaklida chiqaring. $00 \leq hh \leq 23$

Kiritishga misol	Chiqarishga misol
07:05:45PM	19:05:45

Bu masalani yechishda kiruvchi ma’lumotlarni kiritib olish uchun scanf va chiqarish uchun printfdan foydalangan qulayroq. Quyida masala yechimini ko‘rib chiqamiz:

```
#include <string>
#include <stdio.h>
using namespace std;
int main (){
    int a, b, c;
    char d;
    scanf("%d:%d:%d%cM", &a, &b, &c, &d);
```

```
    if (d == 'P'){
     if (a < 12) printf("%02d:%02d:%02d",a+12,b,c);
        else printf("%02d:%02d:%02d",a,b,c);
    }
    else{
    if (a == 12)printf("00:%02d:%02d",b,c);
    else printf("%02d:%02d:%02d",a,b,c);
    }
    return 0;
}
```

Nazorat savollari

1. Zamonaviy kompyuterlarning kiritish va chiqarish vositalari qanchalik turlicha bo‘lishi mumkin?
2. istream oqimi nima qiladi?
3. ostream oqimi nima qiladi?
4. Faylni o‘qishning 4 bosqichini aytib bering.
5. Faylga yozishning 4 bosqichini aytib bering.

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. http://www.stroustrup.com/4th.html
4. http://acm.tuit.uz/forum/

3-amaliy mashg‘ulot. C++ tilining grafik imkoniyatlari. Foydalanuvchi grafik interfeysini tashkil etish.

Ishdan maqsad: C++ tilining tilining grafik imkoniyatlarini o‘rganish xamda dasturni shakllantirishda ulardan foydalanish. Dastur bajarilishi davomida garfik imkoniyatlardan to‘liq foydalanish.

Masalaning qo‘yilishi: 600×400 piksel o‘lchamidagi Mening oynam nishonli Simple_window bo‘sh oyna obyektini yaratish va siniq chiziq ko‘rinishidagi grafik , uchburchak va sinus grafigini chizish dasturini tuzing, uni kompilyatsiya qiling, aloqalarni tahrirlang va bajaring.

Ishni bajarish uchun namuna

Matnlarni kiritish-chiqarish kabi, dasturimizni soddalashtirish uchun, kiritish-chiqarish qurilmalari, operasion tizimlar va shu kabilar o‘rtasidagi farqlarni bartaraf etuvchi kutubxonadan foydalanamiz. Afsuski, C++ tilida, kiritish-chiqarish standart oqimi kutubxonasiga o‘xshash grafik foydalanuvchi interfeysi standart kutubxonasiga ega emas, shuning uchun mavjud kutubxonalardan biridan foydalanamiz. Buning uchun quyida berilgan fayllarni “c:\Program Files\Dev-Cpp\MinGW64\include\” manziliga yuklashingiz kerak bo‘ladi:

- **Graph.h**
- **Gui.h**
- **GUI_2.h**
- **Point.h**
- **Simple_window.h**
- **std_lib_facilities.h**
- **Window.h**

***main* grafik sarlavhalar fayli va funksiyalari.** Birinchi navbatda grafik sinflar va grafik foydalanuvchi interfeys sinflari aniqlangan sarlavha fayllarini kiritamiz.

```
#include"Window.h"    //odatiy oyna
#include"Graph.h"
yoki
#include "Simple_window.h" // agar bizga Next
tugmachasi kerak bo‘lsa
#include "Graph.h"
```

Fahmlaganingizdek Window.h fayli oyna bilan bog‘liq vositalardan, Graph.h fayli esa, oynada shakllarni chizish bilan bog‘liq uskunalardan tashkil topgan. Bu vositalar Graph_lib nomli maydonda tavsiflanadi. Belgilarni soddalashtirish uchun using namespace direktivasidan foydalanamiz [24].

```
using namespace Graph_lib;
```

Odatda main() funksiyasi biz bajarishni xoxlaydigan kodni hamda kamdan-kam vaziyatlarni qayta ishlashni o‘z ichiga oladi.

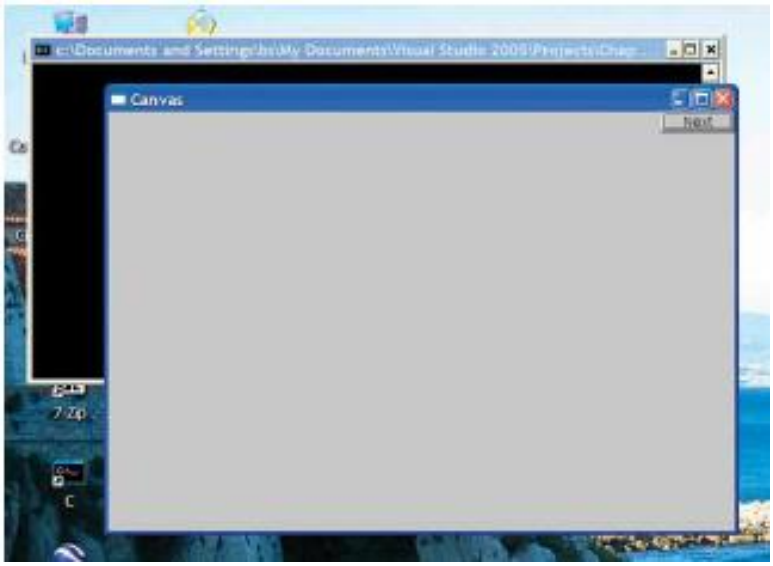
```
int main ()
try
{
// . . . bu yerda bizning kod joylashadi . . .
}
catch(exception& e) {
// xatolar haqida xabar
return 1;
}
catch(...) {
// xatolar haqida boshqa xabar
return 2;
}
```

Deyarli bo‘sh oyna. Bu yerda biz xatolarni qayta ishlashni muhokama qilamiz, hamda main() funksiyasida grafikani tasvirlashga o‘tamiz:

[24] http://www.cplusplus.com/

```
    Point tl(100,100); // oynamizning yuqori chap
burchagi
    Simple_window win(tl,600,400,"Canvas");
    // tl oyna koordinatalari chap yuqori burchakka joy
beradi
    // 600*400 oyna o‘lchami
    // sarlavha: Canvas
    win.wait_for_button(); // tasvirlash!
```

Ekranda oyna chiqishi uchun, biz tizim boshqaruvini grafik foydalanuvchi interfeysiga beramiz. Buning uchun win.wait_for_button() funksiyasidan foydalanamiz. Natija quyidagi rasmda keltirilgan.

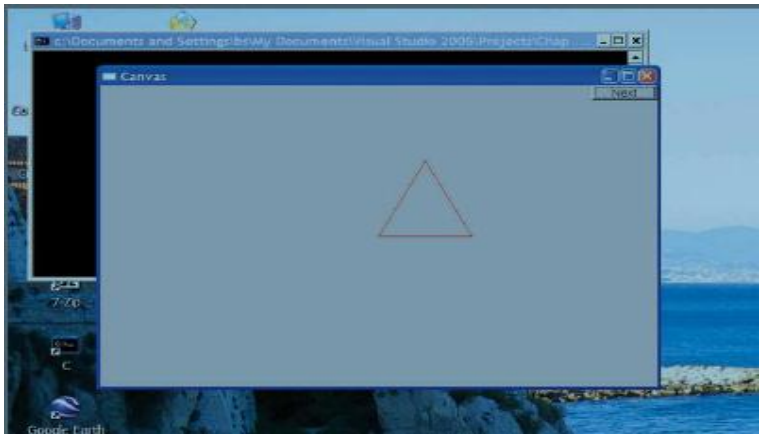


Bizning vazifamiz – ekranga chiqarish uchun obyektlarni yaratish mumkin bo‘lgan, sinflarni aniqlash. Masalan, siniq chiziq ko‘rinishida grafik chizishimiz mumkin. Quyida bu vazifani bajarish uchun katta bo‘lmagan dastur keltirilgan:

```
    #include "Simple_window.h" // oyna kutubxonasiga
kirishni ochadi
    #include "Graph.h" // garfik kutubxonaga kirishni
ochadi
    int main()
    {
```

```
    using namespace Graph_lib; // bizning grafik
vositalarimiz
    // bo‘shliqda joylashgan
    // Graph_lib nomi
    Point tl(100,100);   // ekranning yuqori chap
burchagini beramiz
    Simple_window win(tl,600,400,"Canvas"); //sodda
oynani yaratamiz
    Polygon poly;    // shaklni yaratamiz(ko‘pburchak)
poly.add(Point(300,200));// nuqtani qo‘shamiz
    poly.add(Point(350,100));// boshqa nuqtani
qo‘shamiz
    poly.add(Point(400,200));// uchinchi nuqtani
qo‘shamiz
    poly.set_color(Color::red); // poly obyekt
vositasini aniqlaymiz
    win.attach (poly);   // poly  obyektini oyna bilan
bog‘laymiz
    win.wait_for_button();   // boshqaruvni ekran
drayverga uzatamiz
    }
```

Bu dasturni ishga tushirib, biz taxminan quyidagi tasvirni ko‘ramiz.



Dastur satrlari bo‘ylab, uni qanday ishlashini ko‘rib chiqamiz. Birinchi navbatda dasturga grafik interfeysimizdagi kutubxonaning boshlang‘ich fayllarini keltiramiz.

```
#include "Simple_window.h" // oyna kutubxonasiga kirishni ochadi
#include "Graph.h" // garfik kutubxonaga kirishni ochadi
```

Keyin main() funksiyasida biz kompyuterga, grafik kutubxonamiz vositalari Graph_lib nomli makonda joylashganini xabar qilamiz.

```
using namespace Graph_lib; // grafik vositalari Graph_lib nomli makonda joylashgan
```

Keyin oynamizning yuqori chap burchagi koordinatalari deb hisoblaydigan nuqtani belgilaymiz.

```
Point tl(100,100); // ekrannnig yuqori chap burchagi koordinatalarini beramiz
```

Keyin ekranda oyna yaratamiz.

```
Simple_window win(tl,600,400,"Canvas"); // sodda oyna yaratamiz
```

Buning uchun Graph_lib kutubxonasida oynani tasvirlovchi, Simple_window sinfidan foydalanamiz. Simple_window sinfining aniq obyekti win nomi bilan nomlanadi; boshqacha aytganda win – bu Simple_window sinfining o‘zgaruvchisi.

Canvas satri oynani belgilash uchun foydalaniladi. Agar yaxshilab qaralsa, unda oyna ramkasining yuqori chap burchagida Canvas so‘zini ko‘rish mumkin.

Oynaga obyektni joylashtiramiz.

```
Polygon poly;    // shaklni yaratamiz (ko‘pburchak)
poly.add(Point(300,200)); // nuqta qo‘shamiz
poly.add(Point(350,100)); // boshqa nuqta qo‘shamiz
poly.add(Point(400,200)); // uchinchi nuqtani qo‘shamiz
```

poly ko‘pburchakni belgilaymiz, so‘zngra unga nuqtani qo‘shamiz. Bizning grafik kutubxonamizda Polygon sinf obyektlari bo‘sh yaratiladi, biz unga xoxlagancha nuqtalar sonini qo‘shishimiz mumkin. Biz uchta nuqta qo‘shdik va uchburchak hosil qildik. Nuqtalar o‘zida oynada berilgan gorizontal va vertikal x va u koordinatalarni aks ettiradi.

Bunday imkoniyatni namoyish etish uchun, ko‘pburchak tomonlarini qizil rang bilan berdik.

```
poly.set_color(Color::red); // poly  obyekti vositalarini aniqlaymiz
```

poly obyektini **win** oynasi bilan bog‘laymiz.

```
win.attach(poly); // poly  obyektini oyna bilan bog‘laymiz
```

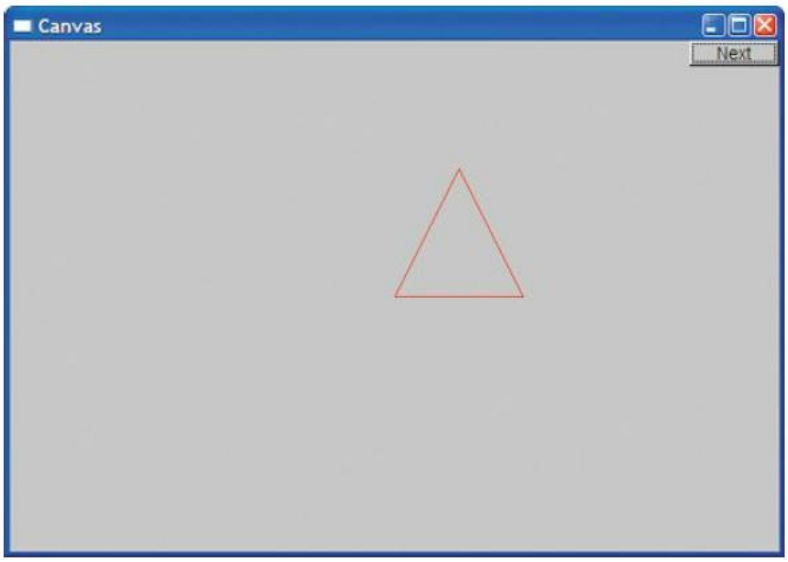
Ekranda hozircha hyech qanday amal bajarilmayotganligini anglash qiyin emas. Biz oyna (aniqrog‘i, Simple_window sinfi obyekti) va ko‘pburchak (poly nomli) yaratdik, ko‘pburchakni qizil rangga bo‘yadik (Color::red) va uni win oynasi bilan bog‘ladik, lekin biz bu oynani ekranga chiqarishga buyruq bermadik. Buni dasturning oxirgi satri bajaradi.

```
win.wait_for_button(); // ekran drayveriga boshqaruvni uzatamiz
```

Grafik foydalanuvchi interfeysi tizimi ekranda obyektlarni aks ettirishi uchun, biz boshqaruvni tizimga uzatdik. Bu vazifani Simple_window oynasida Next tugmachasini bosmaguningizcha tizimni kutishga majbur qiluvchi wait_for_button() funksiyasi bajaradi.

Bu bizga dastur o‘z ishini tugatishidan avval oynani ko‘rishga imkon beradi va oyna yopiladi. Qachonki siz tugmachani bossangiz dastur oynani yopib ishini tugatadi.

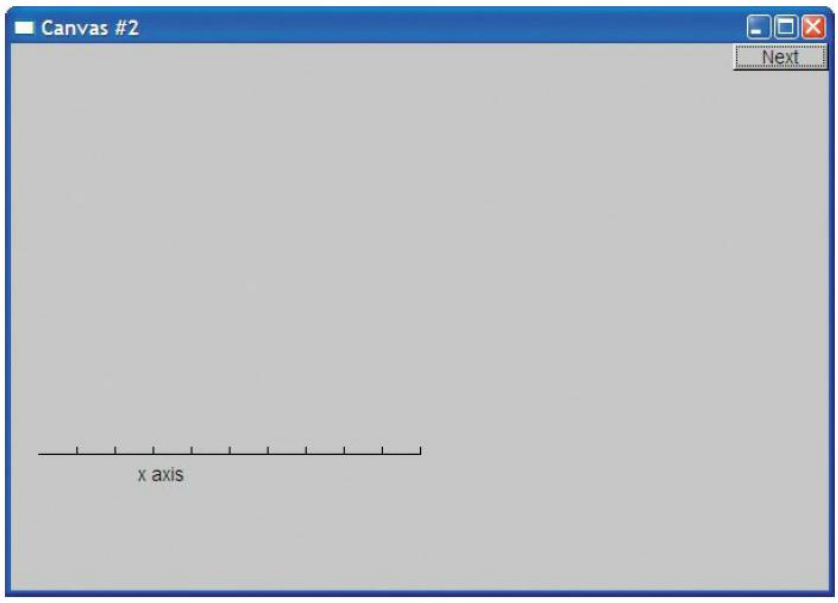
Bizning oynamiz quyigi ko‘rinishda bo‘ladi.



Bu grafikada Next tugmachasi ko‘rinmaydi, chunki biz uni Simple_window sinfida qurdik.

Koordinata o‘qlari. Koordinata o‘qlarisiz berilganlarni tasvirlab berishni iloji yo‘q. Shuning uchun bizga koordinata o‘qlari kerak bo‘ladi.

```
    Axis xa(Axis::x, Point(20,300), 280, 10, "x axis");
// Axis obyektni yaratamiz
    // Axis sinf — Shape sinfining turli ko‘rinishi
    // Axis::x gorizontal o‘qni anglatadi
    // o‘q boshi —(20,300) nuqtada
    // o‘q uzunligi — 280 piksel 10 bo‘linuvchi
    // "x o‘qi" — o‘q belgisi
```



```
    win.attach(xa);  // xa obyektni oyna bilan
bog‘laymiz
    win win.set_label("Canvas #2"); // oyna belgisini
o‘zgartiramiz
    win.wait_for_button();    // tasvirlash!
```

Ish ketma-ketligi shundaki: Axis sinfida obyekt yaratamiz, uni oynaga qo‘shamiz va ekranga chiqaramiz.

Natijalarni identifikasiyalash uchun Window sinfining set_label() funksiyasi yordamida "Canvas #2" satrida ekran belgisini o‘zgartirdik.

Endi *u* o‘qini qo‘shamiz.

```
    Axisya(Axis::y, Point(20,300), 280, 10, "yaxis");
    a.set_color(Color::cyan);// rangni tanlaymiz
    ya.label.set_color(Color::dark_red);  // matn
rangini tanlaymiz
    win.attach(ya);
    win.set_label("Canvas #3");
    win.wait_for_button();    // tasvirlash!
```

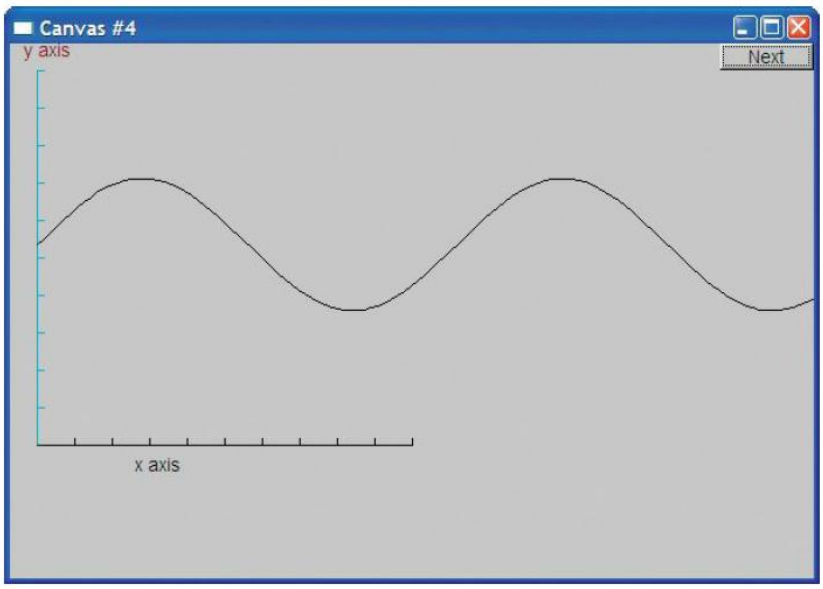
Ayrim imkoniyatlarni namoyish etish uchun, biz *u* o‘qini xavo ranga (cyan), belgini esa to‘q qizilga bo‘yadik.

Funksiya grafigi. Sinus grafigini tasvirlovchi shakl yaratamiz va uni oyna bilan bog‘laymiz.

```
    Function sine(sin,0,100,Point(20,150),1000,50,50);
// sinus grafigi
    // 1000 nuqtadan foydalanib(0,0) dan (20,150) gacha
[0:100) oraliqda sin() chizamiz v diapazone [0:100) ot
(0,0) do (20,150),
    Masshtablash uchun koordinatalar 50 ga
ko‘paytiriladi
    win.attach(sine);
    win.set_label("Canvas #4");
```

```
win.wait_for_button();
```

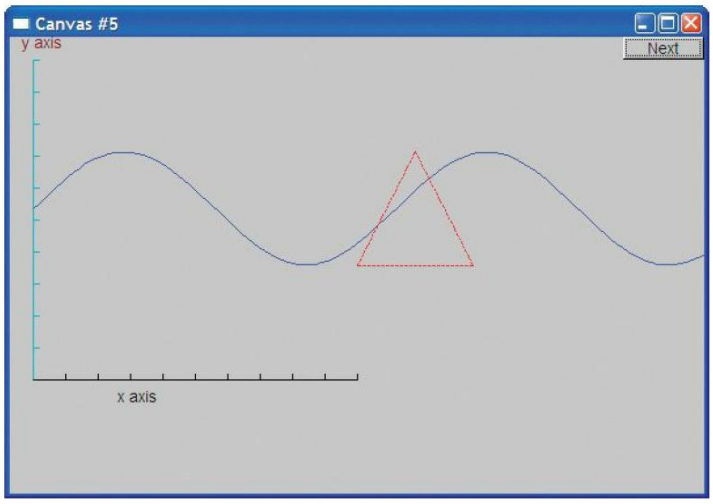
Bu yerda sine nomli Function sinf obyekti sin() funksiyasi standart kutubxonasidan foydalangan holda sinus grafikasini chizadi.



Ko‘pburchak. Polygon sinfi obyekti chiziqlar bilan bog‘langan nuqtalar ketma-ketligida beriladi. Birinchi chiziq birinchi nuqtani ikkinchisi bilan bog‘laydi, ikkinchi chiziq ikkinchi nuqtani uchinchisi bilan bog‘laydi, oxirgi chiziq esa oxirgi nuqtani birinchisi bilan bog‘laydi.

```
sine.set_color(Color::blue);//sinus grafigi rangini o‘zgartirdik
Polygonpoly; //Polygon sinfi – bu Shape sinfining turli ko‘rinishi
poly.add(Point(300,200)); //uchta nuqta uchburchakni tasvirlaydi
poly.add(Point(350,100));
poly.add(Point(400,200));
poly.set_color(Color::red);
poly.set_style(Line_style::dash);
win.attach(poly);
win.set_label("Canvas#5");
win.wait_for_button();
```

Quyidagi natijani olamiz.

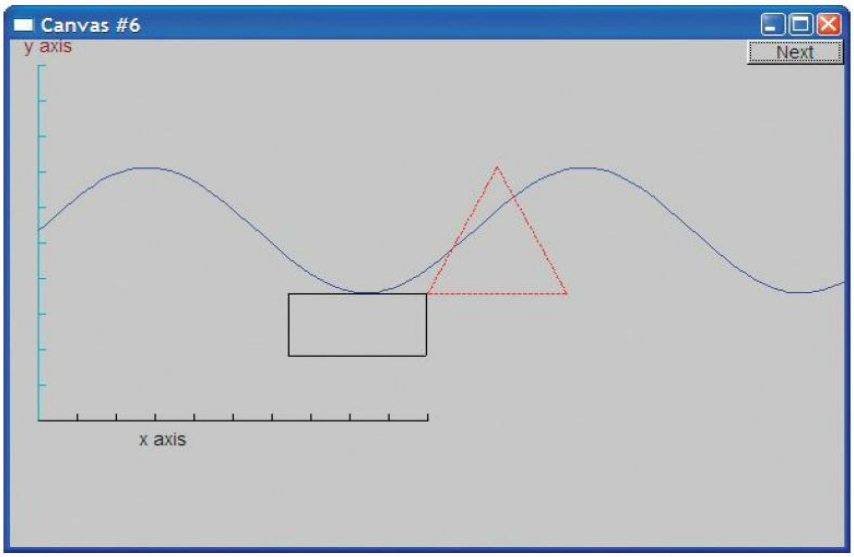


To‘g‘ri burchak. Ekran – bu to‘g‘ri burchak, oyna – bu qog‘oz varag‘i to‘g‘ri burchagi. Shakllarning katta hajmi to‘g‘ri burchak hisoblanadi.

Rectangle sinfi bo‘yi va hajmi bo‘yicha chap yuqori burchak koordinatalarida xarakterlanadi.

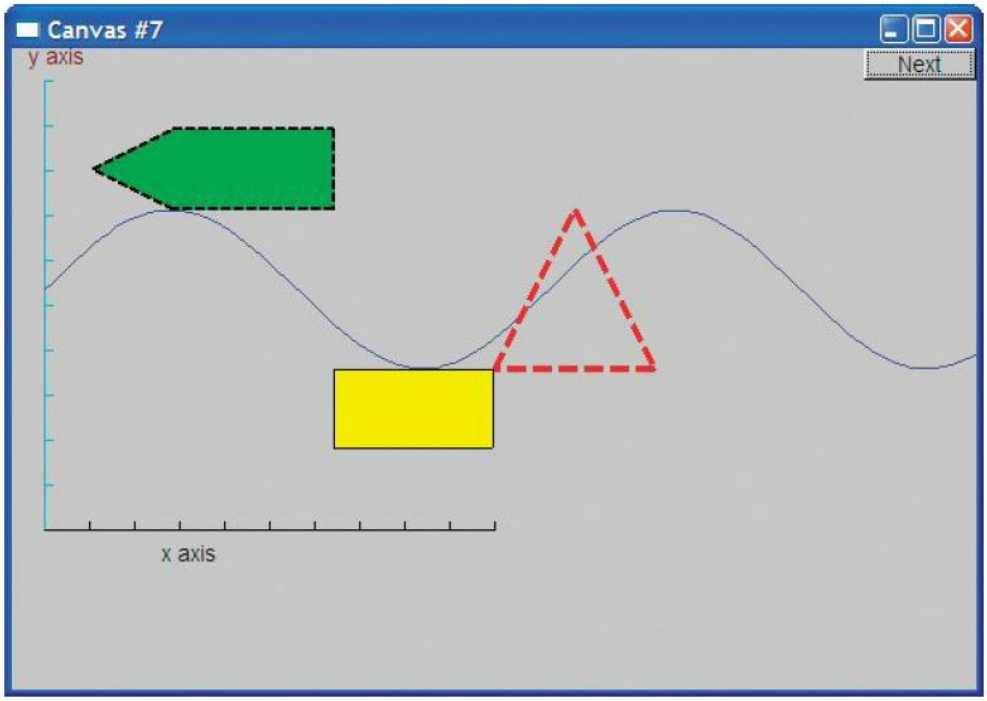
```
Rectangle r(Point(200,200), 100, 50); // yuqori
chap burchak,
// hajm, bo‘y
win.attach(r);
win.set_label("Canvas #6");
win.wait_for_button();
```

Bu fragment ekranda quyidagi oynani ochadi.



To‘ldirish. Hozirgi vaqtgacha bizning shakllarimiz rangsiz chizilgan edi. Ularni rang bilan to‘ldirish mumkin.

```
r.set_fill_color(Color::yellow);//to‘g‘ri burchak ichidagi rang
poly.set_style(Line_style(Line_style::dash,4));
poly_rect.set_style(Line_style(Line_style::dash,2));
poly_rect.set_fill_color(Color::green);
win.set_label("Canvas#7");
win.wait_for_button();
```



Interfeysli grafik sinflar

Color	Chiziq, matn yaratish va shaklni to‘ldirish uchun foydalaniladi
Line_style	Chiziq chizish uchun foydalaniladi
Point	Window sinfi obyekti ichida va ekranida joy borligini tekshiruvchi vazifalar uchun foydalaniladi
Line	Point sinfi ikkita obyekti ekranida ko‘ringan chiziqlarni kesish
Open_polyline	Point sinfi obyekti ketma-ketligida aniqlangan kesilgan chiziqlarni bir biri bilan bog‘lash ketma-ketligi
Closed_polyline	Open_polyline sinfiga o‘xshash, lekin farqi shundaki chiziqlar kesimi Point sinfi oxirgi obyektini birinchisi bilan bog‘laydi
Polygon	Closed_polyline sinfi, bu yerda kesmalar hyech qachon

	kesishmaydi
Text	Belgilar satri
Lines	Point sinfi aniqlagan kesmalar chizig‘i to‘plami
Rectangle	Tez va qulay tasvirlash uchun optimallashgan shakl
Circle	Radius va markazi aniqlangan aylana
Ellipse	Markazi va ikkita o‘qlari aniqlangan ellips
Function	Aniqlangan kesmada berilgan bitta o‘zgaruvchi funksiyasi
Axis	Belgilangan koordinata o‘qi
Mark	Belgi bilan belgilangan nuqta (masalan, x yoki 0)
Marks	Belgilar bilan belgilangan nuqtalar ketma-ketligi (masalan, x yoki 0)
Marked_polyline	Belgilar bilan belgilangan Open_polyline sinfi
Image	Rasmli fayllar tarkibi

Grafik foydalanuvchi interfeysi sinfi

Window	Grafik obyektlar aks etadigan ekran maydoni
Simple_window	Next tugmachali oyna
Button	Tugmachani bosib biron bir funksiyani chaqirish mumkin bo‘lgan oynadagi to‘g‘ri burchak
In_box	Foydalanuvchi satrni kiritishi mumkin bo‘lgan oyna maydoni
Out_box	Satrni chiqarish mumkin bo‘lgan oyna maydoni
Menu	Button sinfi obyektlari vektori

Nazorat savollari

1. Grafika nima uchun kerak?
2. Nima uchun grafikasiz munosabat qilib bo‘lmaydi?
3. Nima uchun grafika dastruchilar uchun qiziqarli?
4. Oyna nima?
5. Bizning grafik interfeys (bizning grafik kutubxonamiz) sinfimiz aynan qanday muhitda joylashgan?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

4-amaliy mashg‘ulot. C++ tilida *vector* yordamida massivlar ustida amallar bajarish.

Ishdan maqsad: C++ dasturlash tilining *vector* obyekti bilan yaqindan tanishish va u bilan ishlash bo‘yicha amaliy ko‘nikmalarni hosil qilish. Berilgan topshiriqlarni dasturiy yechim to‘g‘riligini avtomtik testlovchi tizim (http://acm.tuit.uz) yordamida tekshirish [25].

Masalaning qo‘yilishi: Butun sonlar ketma-ketligi berilgan. Ushbu ketma-ketlikni undagi mavjud takrorlanuvchi sonlarni olib tashlab qayta tuzing, Agar bitta son ketma-ketlikda ikki yoki undan ko‘p marta uchrasa uni birinchi marta uchraganini olib qolib qolganini o‘chirib tashlash lozim. Masala yechimini http://acm.tuit.uz saytidan ro‘yxatdan o‘tgan xolatda tekshirilsin.

Ishni bajarish uchun namuna

Bazi hollarda massivdan foydalanish ayrim noqulayliklarga olib keladi. Massivning boshiga element qo‘shish, massiv ichidan elementlarni o‘chirish amallarini bajarishda noqulayliklarga olib keladi. Bu amallarni bajarish uchun massiv elementlarini joylashini (o‘lchamini) o‘zgartirishga ham to‘g‘ri keladi. Bunday noqulayliklarga uchramaslik uchun *vector* dan foydalanish kerak bo‘ladi.

vector o‘lchami o‘zgaruvchi massiv bo‘lib, unga elementlarni ixtiyoriy ravishda, ixtiyoriy joyga qo‘shish, ixtiyoriy joydan o‘chirish imkoniyatlarini beradi.

Quyidagi jadvalda *vector*ning asosiy funksiyalari keltirilgan:

Funksiya	Vazifasi
A.push_back()	A vectorning oxirga elementni qo‘shadi.
A.pop_back()	A vectorning oxirdagi elementni o‘chiradi.
A.clear()	A vectorni tozalaydi.
A.size()	A vectorning o‘lchamini aniqlaydi
A.insert()	A vectorga element qo‘shish

[25] http://acm.tuit.uz/forum/

A.resize()	A vectorga yangi o‘lcham berish
A.erase()	A vectordan element o‘chirish
A.swap(B)	A vector bilan B vectorni almashtiradi
A.empty()	A vector bo‘shligini tekshirish

Yuqorida qo‘yilgan masalani vector yordamida ishlaymiz:

```
#include <vector>
#include <conio.h>
#include <iostream>
using namespace std;
vector<int> A;
int main()
{
    int n, k;
    cin >> n;
    for (int i = 0; i < n; i ++)
    {
        cin >> k;
        A.push_back(k);
    }
    for (int i = 0; i < A.size(); i ++)
    {
        int j = i + 1;
        while (j < A.size())
              if (A[j] == A[i])
                 A.erase(A.begin() + j);
              else
                  j ++;
    }
    for (int i = 0; i < A.size(); i ++)
        cout << A[i] << "\n";
}
```

Nazorat savollari

1. *vector* oddiy massivdan nimasi bilan farqlanadi?
2. *vector* ning asosiy imkoniyatlarini sanab bering?
3. *vector* yordamida ikki o‘lchamli massivlar ustida amallarni bajarish mumkinmi?
4. Dinamik massiv bilan *vector* ni taqqoslash mumkinmi?
5. A.empty() funksiyasi qanday amal bajaradi?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. http://www.cplusplus.com/reference/vector/vector/

5, 6-amaliy mashg‘ulot. Ro‘yxat, navbat va stek.

Ishning maqsadi

1. Ro‘yxat bilan tanishish.
2. Navbat bilan tanishish.
3. Stek bilan tanishish.
4. Dinamik ma’lumotlat strukturasi bilan ishlashni o‘rganish.

Topshiriq

1. Har bir tinglovchi jurnaldagi tartib raqami bo‘yicha vazifalarning dasturini tuzishi lozim.
2. Hisobot shaklida oldin vazifa, uni bajarishda foydalanilgan funksiyalar, dastur kodi va natijalarni keltirib o‘tishi lozim.
3. 6-amaliy mashg‘ulotda beriladigan barcha vazifalarni bitta hisobot shaklida topshiring.

Ro‘yxat

Ko‘p misollarda tarkibi, hajmi o‘zgaruvchan bo‘lgan murakkab konstruksiyalardan foydalanishga to‘g‘ri keladi. Bunday o‘zgaruvchan ma’lumotlar dinamik informasion strukturalar deb ataladi.

Eng asosiy dinamik informasion struktura bu **ro‘yxatdir**. Ro‘yxat uchun 3 ta oddiy amal aniqlangan.

- Navbatga yangi element joylashtirish
- Navbatdan element o‘chirish.
- Navbatni bo‘sh yoki bo‘sh emasligini aniqlash.

Ro‘yxatning asosiy xossasi shundan iboratki, ixtiyoriy elementini o‘chirish va ixtiyoriy elementidan keyin yangi element qo‘shish mumkin. Bu amallar boshqa elementlarga ta’sir o‘tkazmaydi.

Bir bog‘lamli ro‘yxatning har bir bo‘g‘inida ma’lumot va keyingi element adresi joylashgan. Agar bu ko‘rsatkich nol qiymatga ega bo‘lsa ro‘yxat oxirigacha

o‘qib bo‘lingan bo‘ladi. Ro‘yxatni ko‘rib chiqishni boshlash uchun birinchi elementining adresini bilish yetarli.

Bir bog‘lamli ro‘yxat elementlari quyidagi strukturali tur orqali ta’riflangan obyektlardan iborat bo‘ladi

```
struct strukturali_tur_nomi
{
struktura elementlari;
struct strukturali_tur_nomi*ko‘rsatkich;
};
```

Bu ro‘yxat ma’lumot saqlovchi maydonlar, hamda keyingi element adresini saqlovchi ko‘rsatkichdan iborat.

Quyida ko‘riladigan bir bog‘lamli ro‘yxat har bir tuguni butun son va keyingi element adresi saqlanadigan ko‘rsatkichdan iborat. Eng birinchi elementi boshlang‘ich marker bo‘lib ma’lumot saqlash uchun emas balki ro‘yxat boshiga o‘tish uchun xizmat qiladi. Bu element hech qachon o‘chirilmaydi.

Ro‘yxat elementi quyidagi strukturali tur obyekti

```
struct slist_node
{
int info;
struct slist_node* next;
};
```

Ro‘yxat bilan ishlash uchun asosiy funksiyalar sarlavhalari:

void delete(struct slist_node*p) – berilgan tugundan keyingi elementni o‘chirish

void insert(struct slist_node*p, int nn) - berilgan elementdan keyin element qo‘shish

int empty(struct slist_node*beg) – ro‘yxat bo‘shligini tekshirish

Bu funksiya asosida quyidagi funksiyalar kiritilgan

struct slist_node* creat_slist(int size) – berilgan sondagi tugundan iborat ro‘yxatni yaratish. Tugun axborot qismi klaviatura orqali kiritilgan sonlar bilan to‘ldiriladi.

void free_slist(struct slist_node* beg) – ro‘yxatni o‘chirish

void print_slist(struct slist_node* beg) - ro‘yxat elementlarini ekranga chiqarish

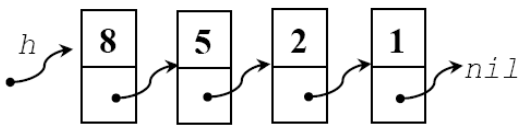
Stek

Stek deb shunday strukturaga aytiladiki, stekka kelib tushgan oxirgi elementga birinchi bo‘lib xizmat ko‘rsatiladi va stekdan chiqariladi. Mazkur ko‘rinishdagi xizmat ko‘rsatishni LIFO (Last input-First output, ya’ni oxirgi kelgan – birinchi ketadi) nomlash qabul qilingan. Stek bir tomondan ochiq bo‘ladi. Stekka bolalar piramidasi va kitoblar qatlamini misol keltirish mumkin.

6.1.-rasm. Stekka misollar

Steklar asosan arifmetik ifodali masalalarni tahlil qilishda, perebor (ajratish) li masalalarda hamda graflardagi algoritmlarda ishlatiladi.

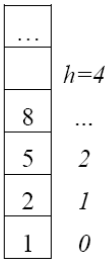
Stekni amalga oshirish uchun chiziqli ro‘yxatdan foydalaniladi, ya’ni ro‘yxatda boshida ko‘rsatkichda saqlanadi:



6.2.-rasm. Stekni chiziqli ro‘yxat orqali ifodalash

Qo‘shish va o‘chirish amallari ro‘yxat boshidan amalga oshiriladi. Bo‘sh stekda ko‘rsatkich boshi (0) ga teng.

Misol, agar stek 4 ta sondan tashkil topgan bo‘lsin. Ular o‘z navbatida 0, 1, 2 va 3 bilan raqamlangan. h = 4 ga teng, ya’ni stekda 4 ta son bor va keyingi qo‘shilayotgan sonni o‘rni stek massivida 4 bo‘ladi. Buni quyidagicha tasvirlash mumkin:

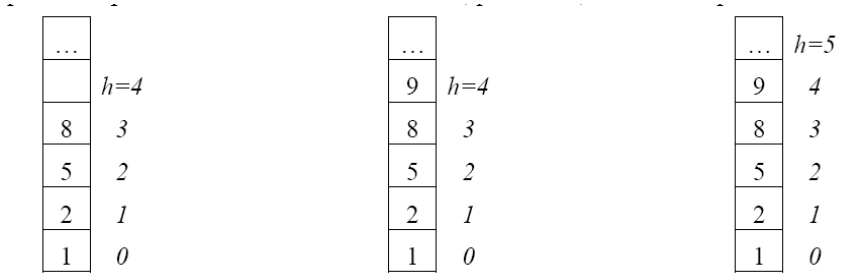


6.3-rasm. Stekni massiv orqali tasvirlash

Stek boshiga element qo‘shish uchun qiymatni yozamiz va h ko‘rsatkichni oshiramiz:

```
a[h ++] = k;
```

Stekga qiymati k=9 sonini qo‘shish jarayonini quyidagicha grafik ko‘rinishda tasvirlash mumkin:



6.4-rasm. Massivga element qo‘shish

Stek boshidan elementni chiqarish uchun teskari amaldan foydalanish lozim:

```
k = a[ -- h];
```

Bo‘sh stekning boshidagi ko‘rsatkichi h = 0 ga teng. Massivga element qo‘shish va o‘chirish davomida stek boshi massiv bo‘ylab ko‘chib turadi.

Universal stekning har bir tuguni axborot qismi void turidagi ko‘rsatkichdan iborat strukturadir:

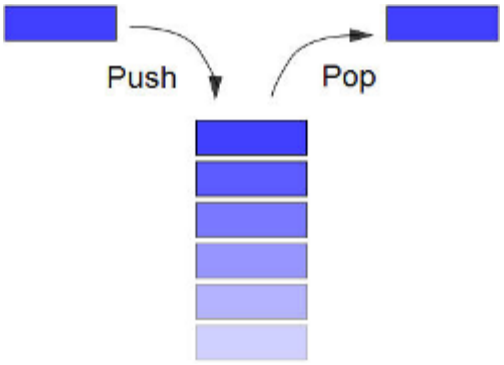
```
struct slist_node
{
 void* info;
 struct slist_node* pred;
};
```

Stek tugunining ro‘yxat tugunidan farqi shundaki, o‘zidan oldingi tugun adresini saqlovchi ko‘rsatkich ishlatilgan [26].

Stek o‘zi alohida struktura sifatida kiritilgan

```
struct stack
{
struct slist_node* end;
int size;
int width;
};
```

Bu yerda end oxirgi tugunga ko‘rsatkich, width ma’lumot hajmi, size navbatdagi elementlar soni.



Asosiy funksiyalar:

void pop(struct stack*p) – stek oxiridagi elementni o‘chirish.

void push(struct stack*p, void* val) –stek oxiriga element qo‘shish. Bu yerda val kiritilayotgan ma’lumotga ko‘rsatkich.

char* top(struct stack p) – stek oxiridagi tugun axborot qismiga ko‘rsatkich qaytarish.

[26] http://acm.tuit.uz/forum/

int empty(struct stack p) – stek bo‘shligini tekshirish.

int size (struct stack p) – stek elementlari soni.

Bundan tashqari stekni inisiallash uchun quyidagi sarlavhali funksiya kiritilgan

void ini_stack (struct stack* p,int n) - Bu yerda n kiritilayotgan ma’lumotlar hajmi.

```
#include<stdio.h>
#include<stdlib.h>
#include<mem.h>
#include<conio.h>
struct slist_node
{
void* info;
struct slist_node* pred;
};

struct stack
{
struct slist_node* end;
int size;
int width;
};

void pop(struct stack*p)
{
struct slist_node* p1 = p->end;
if (p->size>0)
{
p->end = p1->pred;
free(p1);
```

```
    p->size--;
    }
    }

    void push(struct stack*p, void* val)
    {
    struct slist_node* p1 = (struct
slist_node*)malloc(sizeof(struct slist_node));
    p1->info = malloc(p->width);
    memcpy(p1->info,val,p->width);
    if(p->size == 0)
    {
    p->end = p1;
    }
    else
    {
    p1->pred = p->end;
    p->end = p1;
    }
    p->size++;
    }

    char* top(struct stack p)
    {

    return p.end->info;
    }

    int size(const struct stack* p)
    {
```

```
return p->size;
}

int empty(struct stack p)
{
if (p.size>0) return 0;else return 1;
}
void ini_stack(struct stack* p,int n)
{
p->end = NULL;
p->size = 0;
p->width = n;
}
int main()
{
int i,m;
int*pi;
struct stack ps;
ini_stack(&ps,sizeof(int));
for(i = 0;i<5;i++) push(&ps,&i);
while(!empty(ps))
{
pi = (int*)top(ps);
printf("%d ",*pi);
pop(&ps);
}
getch();
return 0;
}
```

Namuna. Agar stekka kiruvchi yonma-yon turgan elementlari teng bo‘lsa ularni o‘chiring va “Yes”, aks holda “No” chiqaring.

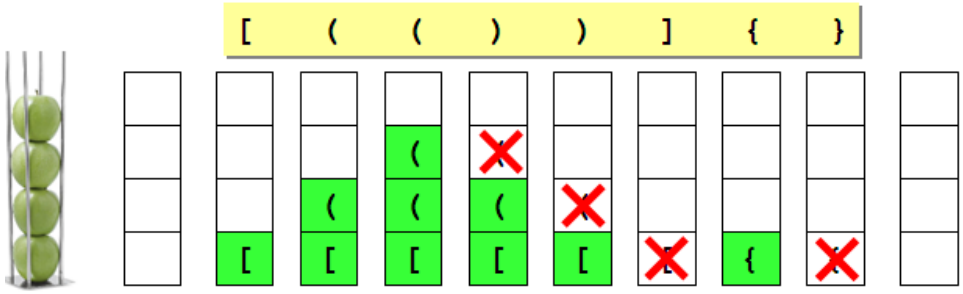
Kiruvchi	**Chiquvchi**
1122559977	Yes
1123558874	No

```
#include <iostream.h>
#include <conio.h>
char lifo[100], ch[100]; int i, top;
void push(char number){
     lifo[top] = number;
     top++;
     }
int main()
{
    cin>>ch; top = 0;
    for (i=0; i<strlen(ch); i++)
     {
       push(ch[i]);
       if (lifo[top-1] == lifo[top-2] && top>1)
        top -=2;
     }
    if (top==0) cout<<"Yes"; else cout<<"No";
  getch();
  return 0;
}
```

Namuna. Masala: [], {} va () uchta tiplardan tashkil topgan satr berilgan. Qavslar to‘g‘ri qo‘yilganligini tekshiring (boshqa simvollarga qaramagan holda). Masalan:

```
[()]{}    ][     [({})]
```

Yechim: qavslar ichma ichligini hisoblagich. Agarda hisoblagich 0 bo‘lsa, qavs to‘g‘ri qo‘yilgan bo‘ladi.



Algoritm:

1. Boshida stek bo‘sh;
2. Sikl boshida satr barcha simvollarini tartib bo‘yicha ko‘rib chiqamiz;
3. Agar navbatdagi simvol ochiladigan qavs bo‘lsa, uni stek boshiga o‘tkazamiz;
4. Agar simvol yopiladigan qavs bo‘lsa, stek boshini tekshiramiz: u yerda berilgan qavsga mos ochiluvchi qavs bo‘lishi lozim (agarda bunday bo‘lmasa, u holda xatolik);
5. Agar stek oxiri bo‘sh bo‘lmasa, u holda ifoda noto‘g‘ri.

Dastur kodi

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
struct Stack {
   char data[100]; // 100 ta simvolli stek
   int  size;          // elementlar soni
   };
int Push ( Stack &S, char x )
{
if ( S.size == 100 ) return 0;
S.data[S.size] = x;
S.size ++;
```

```
return 1;
}
char Pop ( Stack &S )
{
if ( S.size == 0 ) return char(255);
S.size --;
return S.data[S.size];
}
int isEmpty ( Stack &S )
{
if ( S.size == 0 )
     return 1;
else return 0;
}

int main()
{
char br1[3] = { '(', '[', '{' };
  char br2[3] = { ')', ']', '}' };
  char s[80], upper;
  int i, k, error = 0;
  Stack S;
  S.size = 0;
  printf("Qavsli ifodani kiriting > ");
  gets ( s );
for ( i = 0; i < strlen(s); i++ )
  {
  for ( k = 0; k < 3; k++ )
    {
```

```
        if ( s[i] == br1[k] )// agar ochilivchi qavs
bo‘lsa
          {
          Push ( S, s[i] ); // stekka kiritish
          break;
          }
        if ( s[i] == br2[k] )// agar yopilivchi qavs
bo‘lsa
          {
          upper = Pop ( S );  // yoqori elementni olish
          if ( upper != br1[k] ) error = 1;
          break;
          }
        }
      if ( error ) break;
      }
      if ( ! error && (S.size == 0) )
           printf("\nError\n");
      else printf("\nExcelent\n");
        getch();
        return 0;
    }
```

Navbat

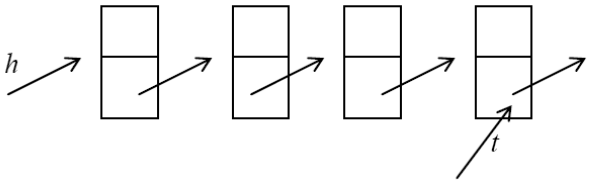
Navbat deb shunday strukturaga aytiladiki, navbatga kelib tushgan birinchi elementga birinchi bo‘lib xizmat ko‘rsatiladi va navbatdan chiqariladi. Navbatga biz kundalik hayotda magazinda, o‘qishda, ishda har doim duch kelamiz. Buni albatta dasturlashda ham ishlatish mumkin. Masalan, siz printerga bir vaqtning o‘zida uchta hujjatni chop etish bo‘yicha buyruq berdingiz. Ular o‘z navbatida navbatida hosil qiladi va eng birinchi navbat (masalan, yuzdan bir sekun oldin) olgani birinchi chop etiladi.

Navbat boshi – massivdan o‘chiriladigan elementning raqami.

Navbat oxiri – navbatga kiritiladigan massiv elementi.

Mazkur ko‘rinishdagi xizmat ko‘rsatishni navbat, ya’ni **FIFO** (*First input-First output*, ya’ni birinchi kelgan – birinchi ketadi) nomlash qabul qilingan. Navbat har ikkala tomondan ochiq bo‘ladi.

Navbatni amalga oshirish uchun ikkita ko‘rsatkichdan foydalaniladi: h boshi va t oxiri.



Qo‘shish oxiridan va o‘chirish ro‘yxat boshidan amalga oshiriladi. Agarda navbat bo‘sh bo‘lsa, h va t ning qiymati nol ga teng bo‘ladi (ya’ni navbat bo‘sh bo‘ladi). Shuning uchun element qo‘shishda ro‘yxat o‘zgarmaydi.

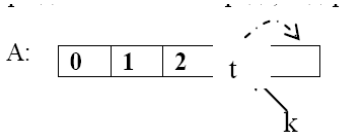
Navbatlar asosan arifmetik ifodali masalalarni tahlil qilishda, prebor (ajratish)li masalalarda hamda graflardagi algoritmlarda ishlatiladi.

Navbatni eng oddiy ko‘rinishda chegaralangan massiv A[0..N-1] ko‘rinishida tasvirlash mumkin.

Navbatni oxiridan yangi element qo‘shish uchun h va t zarur. Buning uchun qiymatni yozamiz va t ni oshiramiz:

```
A[t] = k;
t = (t + 1) mod N;
```

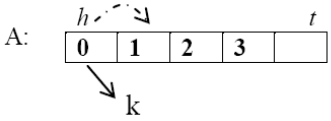
0, 1 va 2 dan tashkil topgan navbatga yangi element qo‘shganda k = 3 ga teng bo‘ladi:



Navbat boshidan o‘chirish:

```
k = A[h];
h = (h + 1) mod N;
```

Masalan, yuqoridagi navbatdan 0 elementini o‘chirgandan so‘ng quyidagini olamiz:



Universal navbat har bir tuguni axborot qismi void turidagi ko‘rsatkichdan iborat strukturadir:

```
struct slist_node
{
void* info;
struct slist_node* next;
};
```

Navbat o‘zi alohida struktura sifatida kiritilgan.

```
struct que
{
struct slist_node* beg;
struct slist_node* end;
int size;
int width;
};
```

Bu yerda beg birinchi tugunga ko‘rsatkich, end oxirgi tugunga ko‘rsatkich, width ma’lumot hajmi, size navbatdagi elementlar soni.

Asosiy funksiyalar:

void pop(struct que*p) – navbat oxiridagi elementni o‘chirish.

void push(struct que*p, void* val) –navbat boshiga element qo‘shish. Bu yerda val kiritilayotgan ma’lumotga ko‘rsatkich.

char* top(struct que p) – navbat boshidagi tugun axborot qismiga ko‘rsatkich qaytarish.

int empty(struct que p) – navbat bo‘shligini tekshirish.

int size (struct que p) – navbat elementlari soni.

```
#include<stdio.h>
#include<stdlib.h>
#include<mem.h>
#include<conio.h>
struct slist_node
{
void* info;
struct slist_node* next;
};
struct que
{struct slist_node* beg;
struct slist_node* end;
int size;
int width;
};
void pop(struct que*p)
{
struct slist_node* p1 = p->beg;
if (p->size>0)
{
p->beg = p1->next;
free(p1);
p->size--;
}
}
void push(struct que*p, void* val)
{
struct slist_node* p1 = (struct
slist_node*)malloc(sizeof(struct slist_node));
p1->info = malloc(p->width);
memcpy(p1->info,val,p->width);
if(p->size == 0)
{
p->beg = p1;
p->end = p1;
}
```

```
else
{
p->end->next = p1;
p->end = p1;
}
p->size++;
}
char* top(struct que p)
{
return p.beg->info;
}
int size(const struct que* p)
{
return p->size;
}
int empty(struct que p)
{
if (p.size>0) return 0;else return 1;
}
void ini_que(struct que* p,int n)
{
p->beg = NULL;
p->end = NULL;
p->size = 0;
p->width = n;
}
int main()
{
int i,m;
int*pi;
struct que ps;
ini_que(&ps,sizeof(int));
for(i = 0;i<5;i++) push(&ps,&i);
while(!empty(ps))
{
pi = (int*)top(ps);
printf("%d ",*pi);
pop(&ps);
}
getch();
return 0;
}
```

Stek va navbatga oid topshiriqlar

1. Uchta tipli (oddiy, kvadrat va figurali) qavslardan tashkil topgan ifoda berilgan. Shu berilgan ifodadagi qavslarni to‘g‘ri yoki noto‘g‘ri qo‘yilganligini aniqlovchi dastur tuzing. Bunda albatta ochilgan qavs yopilgan bo‘lishi lozim. Bunda ularni o‘rinlariga ham e’tibor berish lozim. Qavslar to‘g‘ri bo‘lsa “To‘g‘ri”, aks holda “Noto‘g‘ri” so‘zini chiqaring

Kiruvchi	**Chiquvchi**
({}	Noto‘g‘ri
[{}([])]	To‘g‘ri
[(])	Noto‘g‘ri
()]}	Noto‘g‘ri

2. Postfiks formada (teskari polyak yozuvi) ifoda berilgan. Oldin son keyin esa amallar belgisi berilgan. Uning qiymatini topuvchi dastur tuzing.

Kiruvchi	**Chiquvchi**	**Изох**
62+	8	6+2=8
56-	-1	5-6=1
735*4+-	-12	7-(3*5+4)=-12

3. Matrisa shaklidagi dengizni tasvirlovchi xarita berilgan (0 – dengiz, 1-quruqlik). Orol sifatida faqat birlar bilan belgilangan sohani olish lozim. Berilgan xaritadan orollar soni va eng katta orol maydonini toping.

Kiruvchi	**Chiquvchi**	**Изох**
5 7 0 1 1 1 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1	4 7	0 1 1 1 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1

4. N ta doira berilgan. Bu doiralardan ustma-ust qo‘ygan holda piramida yig‘ilmoqda. Yig‘ishda ularning markazlari bitta nuqtada turishi lozim. Demak,

n ta doira va ularning radiuslari berilgan. Agar ularni ustma-ust qo‘yganda tepadan qaraganda qaysi radiusli doiralarni ko‘rish mumkin.

Kiruvchi	**Chiquvchi**
6 6 2 1 5 3 4	6 5 4
6 1 2 3 4 5 6	6

5. Satr ketma-ket keluvchi simvollardan tashkil topgan. Bitta simvol bir nechta joyda kelishi mumkin. Bu satrdan bir xil simvollarni o‘chirib joriy satrni chiqaring.

Kiruvchi	**Chiquvchi**
abccdeef	abdf
abccrrrrbf	af

Nazorat savollari

1. Ro‘yxatlar turlarini ko‘rsating.
2. Stek qanday prinsipda ishlaydi?
3. Navbat deb qanday strukturaga aytiladi?
4. Stekka misollar keltiring.
5. Stek qayerda ishlatiladi?
6. Navbatga misollar keltiring.
7. Navbat qayerda ishlatiladi?
8. Ro‘yxatga misollar keltiring.
9. Ro‘yxat qayerda ishlatiladi?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Nazirov Sh.A., Kabulov R.V., Babajanov M.B., Raxmanov Q.S. C va C++ tili: kasb-hunar kollejlari uchun qo‘llanma / O‘zbekiston respublikasi Oliv va o‘rta maxsus ta’limi markazi. – Toshkent: “Voris-nashriyot”, 2013.-488b.
3. http://www.cplusplus.com/reference/stack/

7-amaliy mashg‘ulot. C++ tilida sinf va obyektlarni tashkil etish va ulardan foydalanish.

Ishdan maqsad:

3. C++ da sinf yaratish va foydalanish bilan tanishish.
4. Turli sinflarni yaratish (Borland C++, Dev C++, Borland Builder).
5. Sinf o‘zgaruvchilari va funksiyalarini yaratish bilan tanishish.
6. Sinflardan foydalanishni o‘rganish.

Sinflar va sinf a’zolari

Yangi tip sinfni e’lon qilish bilan tuziladi. Sinf - bu bir – biri bilan funksional orqali bog‘langan o‘zgaruvchilar va metodlar to‘plamidir. Sinflarga amaliyotdan ko‘pgina misollar keltirish mumkin. Masalan, avtomabilni g‘ildirak, eshik, o‘rindiq, oyna va boshqa qismlardan tashkil topgan kollekstiya yoki haydash tezligini oshirish, to‘xtatish, burish imkoniyatlariga ega bo‘lgan obyekt deb tasavvur qilish mumkin. Avtomobil o‘zida turli ehtiyot qismlarni va ularni funksiyalarini inkapsulyastiya qiladi. Avtomobil kabi sinfda ham inkapsulyastiya qator imkoniyatlarni beradi. Barcha ma’lumotlar bitta obyektda yig‘ilgan va ularga osongina murojaat qilish, ularni o‘zgartirish va ko‘chirish mumkin. Sizning sinfingiz bilan ishlovchi dasturiy qismlar, ya’ni mijozlar sizning obyektingizdan, uning qanday ishlashidan tashvishlanmasdan, bemalol foydalanishlari mumkin.

Sinf o‘zgaruvchilarning ixtiyoriy kombinastiyasidan, shuningdek boshqa sinflar tiplaridan iborat bo‘lishi mumkin. Sinfdagi o‘zgaruvchilar o‘zgaruvchi – a’zolar yoki xossalar deyiladi. `Sar` sinfi o‘rindiq, radiopriyomnik, shina va boshqa o‘zgaruvchi - a’zolardan iborat. O‘zgaruvchi – a’zolar faqatgina o‘zlarining sinflarida yotadilar. G`ildirak va motor avtomobilning qanday tarkibiy qismi bo‘lsa, o‘zgaruvchi – a’zolar ham sinfning shunday tarkibiy qismidir.

Sinfdagi funksiyalar odatda o‘zgaruvchi a’zolar ustida biror bir amal bajaradilar. Ular funksiya – a’zolar yoki sinf metodlari deb aytiladi. `Mashina` sinfi metodlari qatoriga `Haydash()` va `Tuxtatish()` metodlari kiradi.

`Mushuk` sinfi hayvonni yoshi va og‘irligini ifodalovchi o‘zgaruvchi – a’zolarga ega bo‘lishi mumkin. Shuningdek, bu sinfning funksional qismi `Uxlash()`, `Miyovlash()`, `SichqonTutish()` metodlaridan iborat bo‘ladi.

Funkstiya – a’zolar ham o‘zgaruvchi a’zolar singari sinfda yotadi. Ular o‘zgaruvchi a’zolar ustida amallar bajaradi va sinfni funksional imkoniyatlarini aniqlaydi.

Sinfni e’lon qilish

Sinfni e’lon qilish uchun `class` kalitli so‘zi, undan so‘ng ochiluvchi figurali qavs, so‘ng xossalar va metodlari ro‘yxati ishlatiladi. Sinfni e’lon qilish yopiluvchi figurali qavs va nuqtali vergul orqali yakunlanadi. Masalan, `Mushuk` sinfini quyidagicha e’lon qilish mumkin.

```
class Mushuk
{
unsigned int itsYosh;
unsigned int itsOgirlik;
void Miyovlash()
}
```

`Mushuk` sinfini e’lon qilishda xotira zaxiralanmaydi. E’lon qilish, kompilyatorga `Mushuk` sinfini mavjudligini, hamda unda qanday qiymatlar saqlashi mumkinligi (`itsYosh`, `itsOgirlik`) va u qanday amallarni bajarishi mumkinligi (`Miyovlash()` metodi) haqida xabar beradi. Bundan tashqari, bu e’lon qilish orqali kompilyatorga `Mushuk` sinfining o‘lchami, ya’ni har bir `Mushuk` sinfi obyekti uchun kompilyator qancha joy ajratishi lozimligi haqida ham ma’lumot beradi. Masalan, joriy misolda butun qiymat uchun to‘rt bayt talab qilinsa, `Mushuk` sinfi obyekti o‘lchovi sakkiz bayt bo‘ladi.

(`itsYosh` o‘zgaruvchisi uchun to‘rt bayt, `itsOgirlik` o‘zgaruvchisi uchun to‘rt bayt). `Miyovlash()` metodi xotiradan joy ajratishni talab qilmaydi.

Obyektni e’lon qilish

Yangi turdagi obyekt xuddi oddiy butun sonli o‘zgaruvchidek aniqlanadi. Haqiqatan ham ixtiyoriy butun sonli o‘zgaruvchi quyidagicha aniqlanadi:

```
unsigned int MyVariable
// ishorasiz butun sonni aniklaymiz
```

`Cat` sinfidagi obyekt esa quyidagicha aniqlanadi:

```
Mushuk Frisky // Mushuk obyektini aniqlaymiz.
```

Bu dasturiy kodlarda `unsigned int` tipidagi `MyVariable` nomli o‘zgaruvchi va `Mushuk` sinfining `Frisky` nomli obyekti aniqlandi.

Ko‘pgina hollarda sinf va obyekt tushunchalarini ishlatishda chalkashlikka yo‘l qo‘yiladi. Shuning uchun, `obyekt sinfning biror bir ekzemplyari (nusxasi)` ekanligini yana bir bor ta’kidlash joiz.

Sinf a’zolariga murojaat qilish imkoni

`Mushuk` sinfining real obyektini aniqlaganimizdan so‘ng, bu obyektning a’zolariga murojaat qilish zaruriyati tug‘ilishi mumkin. Buning uchun bevosita murojaat `(.)` operatori qo‘llaniladi. Masalan, `Frisky` obyektining `Weight` o‘zgaruvchi - a’zosiga `50` sonini o‘zlashtirmoqchi bo‘lsak quyidagi jumlani yozishimiz lozim.

```
Fresky.Weight=50;
```

`Meow()` metodini chaqirish uchun esa

```
Frisky.Meow();
```

jumlasini yozish lozim.

Qiymat sinfga emas obyektga o‘zlashtiriladi

C++ tilida berilganlar tipiga qiymat o‘zlashtirilmaydi. Qiymat faqatgina o‘zgaruvchilarga beriladi. Masalan, quyidagi yozuv noto‘g‘ridir:

```
Int=s // noto‘g‘ri
```

Kompilyator `int` tipiga qiymat o‘zlashtirilishi xatolik ekanligi haqida xabar beradi. Xuddi shu nuqtai – nazardan quyidagi yozuv ham noo‘rindir:

```
Cat.itsYosh= 5 // noto‘g‘ri
```

Agarda `Mushuk` obyekt bo‘lmasdan sinf bo‘lsa, yuqoridagi ifodani ham kompilyator xato deb hisoblaydi. O‘zlashtirish amalini bajarishda xatolikka yo‘l qo‘ymaslik uchun oldin `Mushuk` sinfiga tegishli `Frisky` obyektini hosil qilish va uning `ItsYosh` maydoniga `5` qiymatini berish lozim.

```
Mushuk Frisky;
Frisky.itsYosh=5;
```

Sinf a’zolariga murojaat qilish imkonini chegaralash

Sinfni e’lon qilishda bir nechta kalit so‘zlardan foydalaniladi. Ulardan eng muhimlari `publis (ochiq)` va `private` (yopiq) kalit so‘zlari bo‘lib, ular orqali obyektning a’zolariga murojaat qilish imkoniyati chegaralanadi.

Sinfning barcha metodlari va xossalari boshlang‘ich holda yopiq deb e’lon qilinadi. Yopiq a’zolarga faqatgina shu sinfning metodlari orqaligina murojaat qilish mumkin. Obyektning ochiq a’zolariga esa dasturdagi barcha funksiyalar murojaat qilishlari mumkin. Sinf a’zolariga murojaat qilish imkonini belgilash juda muhim xususiyat bo‘lib, bu masalani yechishda uncha katta tajribaga ega bo‘lmagan dasturlarchilar ko‘pincha qiyinchiliklarga duch keladilar. Bu holatni batafsilroq tushuntirish uchun mavzuning boshida keltirilgan masalamizga qaytamiz.

```
class Mushuk
 {
 unsigned int itsYosh;
 unsigned int itsOgirlik;
 void Miyovlash();
 }
```

Bu tarzda sinfni e’lon qilishda `itsYosh` va `itsOgirlik` maydonlari ham, `Miyovlash()` metodi ham yopiq a’zo sifatida aniqlanadi. Dasturda yuqoridagi tartibda `Mushuk` sinfi e’lon qilingan bo‘lsa va bu sinf ekzemplyari bo‘lgan obyektning `itsYosh` a’zosiga `main()` funkstiyasi tanasidan turib murojaat qilsak kompilyator xatolik ro‘y berganligi haqida xabar beradi.

```
Mushuk Baroq;
Baroq.itsYosh = 5 // Xatolik!
// Yopik a’zoga murojaat kilish mumkin emas.
```

`Mushuk` sinfi a’zolariga dasturning boshqa obyektlari tomonidan murojaat qilish imkonini hosil qilmoqchi bo‘lsak, uni `public` kalitli so‘zi orqali amalga oshiramiz.

```
class Mushuk
 {
 public:
 unsigned int itsYosh;
 unsigned int itsOgirlik;
 void Meow( );
 }
```

Endi `public` kalit so‘zi orqali sinfning barcha a’zolari `(itsYosh, itsOgirlik, Miyovlash())` ochiq a’zo bo‘ldi.

Konstruktorlar va destruktorlar

Butun sonli o‘zgaruvchini aniqlashning ikki xil yo‘li bor. Birinchisi, oldin o‘zgaruvchini aniqlash, keyin esa unga biror bir qiymat o‘zlashtirishdir. Masalan,

```
int a; // o‘zgaruvchini aniqlash
........ // bu erda boshqa ifodalar bor
a=7; // o‘zgaruvchiga qiymat o‘zlashtiramiz.
```

Ikkinchisi, o‘zgaruvchi aniqlanishi bilan birga unga darhol qiymat o‘zlashtiriladi. Masalan,

```
int a=7; //o‘zgaruvchini e’lon qilamiz
  //va unga qiymat o‘zlashtiramiz.
```

Qiymat berish amali o‘zgaruvchi aniqlanishi bilan unga boshlang‘ich qiymat o‘zlashtirilishini anglatadi. Keyinchalik, bu o‘zlashtirilgan qiymatni o‘zgartirishingiz ham mumkin.

Sinfning o‘zgaruvchi–a`zosiga qanday qiymat o‘zlashtirildi? Buning uchun sinfda konstruktor deb ataluvchi maxsus funksiya – a`zo ishlatiladi. Zaruriy vaqtda

konstruktor bir nechta parametrni qabul qiladi. Lekin hech qanday tipdagi qiymat qaytarmaydi. Konstruktor – bu sinf nomi bilan ustma – ust tushadigan sinf metodidir.

Sinfda konstruktorni e’lon qilinishi bilan destruktorlar ham aniqlanishi lozim. Agarda konstruktor sinf ob`ektini tuzish va uning o‘zgaruvchi – a’zolariga qiymat berish vazifasini bajarsa, destruktor mavjud ob`ektning xotiradan o‘chiradi. Destruktorlar sinf nomi oldiga tilda (~) belgisini qo‘yish orqali aniqlanadi. Destruktorlar hech qanday argument qabul qilmaydi va hech qanday qiymat qaytarmaydi [27].

Boshlang‘ich berilgan konstruktor va destruktorlar

Agarda siz konstruktor yoki destruktorni aniqlamasangiz, siz uchun bu ishni kompilyatorning o‘zi bajaradi. Standart konstruktor va destruktorlar birorta argument qabul qilmaydi va hech qanday amal bajarmaydi.

1- Namuna. Kasr sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.

```
#include<iostream.h>
#include<conio.h>
using namespace std;
class kasr {
public:
int get(){
cout<<"\n"<<surat<<" "<<mahraj;
}
void set(int a, int b){
surat=a;
mahraj=b;
}
void addkasr(int a, int b, int c, int d){
```

[27] Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013. 488 b.

```
surat= (a*d)+(c*b);
mahraj= b*d;
}
void subkasr(int a, int b, int c, int d){
surat= (a*d)-(c*b);
mahraj= b*d;
}
void multkasr(int a, int b, int c, int d){
surat= a*c;
mahraj= b*d;
}
void divkasr(int a, int b, int c, int d){
surat= a*d;
mahraj= b*c;
}
private:
int surat;
int mahraj;
};
//sinf konstruktori
kasr:: kasr (float mahraj)
{
 m= maxraj;
}
kasr::~kasr( ){ }
int main(){
kasr k;
int a; cin>>a;
int b; cin>>b;
k.set(a,b);
```

```
k.get();
k.addkasr(2,3,4,5);
k.get();
k.subkasr(3,2,6,2);
k.get();
k.multkasr(1,2,3,2);
k.get();
k.divkasr(1,2,4,2);
k.get();
getch();
return 0; }
```

Sinf va obyektga oid topshiriqlar

7.1-jadval.

№	Masala sharti
1	Tinglovchi sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
2	Avtomashina sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
3	Mijoz sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
4	Tovar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
5	Avia reys sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.

Nazorat uchun savollar

1. Ochiq (public) va yopiq (private) o‘zgaruvchi – a’zolar orasida qanday farq bor?
2. Sinfning funksiya a’zolari qachon yopiq bo‘lishi lozim?
3. Sinfning funksiya a’zolari qachon ochiq bo‘lishi lozim?
4. Agar sinf class so‘zi yordamida ta’riflangan bo‘lsa ko‘zda tutilgan bo‘yicha komponentalari qanday murojaat huquqiga ega bo‘ladi?
5. Qaysi holda sinf usullari joylashtiriluvchi funksiya hisoblanadi?

6. Agarda sinfning ikkita obyektini e’lon qilsak, ularning o‘zgaruvchi a’zolari qiymati turlicha bo‘lishi mumkinmi?
7. Konstruktorlar xossalarini ko‘rsating.
8. Sinf obyektini hosil qilishda qanday funksiya chaqiriladi?

Foydalanilgan adabiyotlar

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Nazirov Sh.A., Kabulov R.V., Babajanov M.B., Raxmanov Q.S. C va C++ tili: kasb-hunar kollejlari uchun qo‘llanma / O‘zbekiston respublikasi Oliv va o‘rta maxsus ta’limi markazi. – Toshkent: “Voris-nashriyot”, 2013.-488b.
3. http://www.cplusplus.com/reference/

V. BO`LIM

KEYSLAR BANKI

V. KEYSLAR BANKI

Eslatma: *Quyida keltiriladigan keyslarda vaqt bo‘yicha chegara: 2 sekund va xotira bo‘yicha chegara: 64 megabayt etib belgilanganini e’tiborga olish lozim.*

1-keys. Summa (acm.tuit.uz dagi masala shifri **0005)**

1 dan N gacha bol'gan sonlar yig‘indisini hisoblang.

Kiruvchi ma'lumotlar:

Qiymati 10^4 dan oshmaydigan N butun soni berilgan.

Chiquvchi ma'lumotlar:

1 dan N gacha bol'gan sonlar yig‘indisini chiqarish lozim.

Kiritishga misol	Chiqarishga misol
5	15
10	55

2-keys. 5 x 5 = 25!(acm.tuit.uz dagi masala shifri **0006)**

Farux va Murod maktabda bir sinfda o‘qishadi. Yaqinda Murod Faruxga 5 raqami bilan tugaydigan natural sonlarni kvadratga ko‘tarish usulini o‘rgatdi. Endi Farux 5 raqami bilan tugaydigan ikki xonali(ba'zi bir uch xonali) sonlarni osonlik bilan kvadratga ko‘tarishi mumkin. Bu usul quyidagicha amalga oshiriladi: 5 raqami bilan tugaydigan sonlarni kvadratga ko‘tarish uchun oxirgi 5 raqamini o‘chirib qolgan songa shu sondan 1ga ko‘p sonni ko‘paytiramiz va chiqgan son oxiriga (o‘ng tomoniga) 25 sonini yozib qo‘yamiz. Masalan, 125 sonini kvadratga ko‘paytirish uchun, shunchaki 12ni 13ga ko‘paytirib ko‘paytma oxiriga 25 sonini yozib qo‘yamiz, ya'ni 12*13=156 oxiriga 25, javob 15625. Demak 125ning kvadrati 15625ga teng. Murod o‘rganganlarini sinab ko‘rishi uchun 5 raqami bilan tugaydigan sonni kvadratga ko‘taradigan dastur yozing.

Kiruvchi ma'lumotlar: Yagona qatorda 5 raqami bilan tugaydigan, $4*10^5$dan oshmaydigan, bitta natural A son yozilgan.

Chiquvchi ma'lumotlar: A^2 natural sonni chiqaring.

Kiritishga misol	Chiqarishga misol
15	225
75	5625
4255	18105025

3-keys. Arifmetika(acm.tuit.uz dagi masala shifri **0007)**

Ikki sonni ko‘paytmasini tekshirish

Kiruvchi ma'lumotlar: Uchta A, B va C natural sonlari probel orqali berilgan. A va B $\leq 10^6$, C $\leq 10^9$

Chiquvchi ma'lumotlar: Agarda A*B=C bo‘lsa YES, aks holda NO chiqaring.

Kiritishga misol	Chiqarishga misol
8 54 432	YES
16 19 777	NO

4-keys. Svetofor(acm.tuit.uz dagi masala shifri **0010)**

Farxod chorrahaga keldi va shu vaqtda svetofor qizil yonganligini ko‘rdi. Farxod qiziqib qoldi, svetofor chiroqlari qanday aniqlik bilan yonmoqda:

qizil –sariq – yashil –sariq –qizil – sariq –yashil - …

Navbatdagi yashil chiroq yonganda Farxod yo‘lni kesib o‘tishga ahd qildi. Bu vaqtda yashil chiroq i-nchi marta yongandi. Shu vaqtgacha necha marta qizil chiroq (farxod kelgan vaqtni hisobga olgan holda) va necha marta sariq chiroq yonganligini aniqlovchi dastur tuzing.

Kiruvchi ma'lumotlar: Svetoforning yashil chirog‘i necha marta yonganligini ko‘rsatuvchi bitta i ($1 \leq i \leq 100$) soni berilgan.

Chiquvchi ma'lumotlar: Ikkita son chiqsin. Birinchisi – necha marta qilzil chiriq, ikkinchisi – necha marta sariq chiroq yonganligini bildiradi.

Kiritishga misol	Chiqarishga misol
2	2 3

5-keys. $a^2+ b^2$(acm.tuit.uz dagi masala shifri **0039)**

Ikkita a va b sonlari berilgan ($-10^9 \leq a,b \leq 10^9$). $a^2+ b^2$berilgan sonlarning yig‘indisini chiqaruvchi dastur tuzing. Yordam: Borland Delphi va Free Pascal uchun int64, Visual C/C++ uchun __int64, GNU C/C++ uchun long long yoki Java uchun long dan foydalaning

Kiruvchi ma'lumotlar: a va b sonlari berilgan

Chiquvchi ma'lumotlar: Ularni kvadratlari yig‘indisini toping

Kiritishga misol	Chiqarishga misol
2 3	13
17 -18	613

6-keys. Magazin (acm.tuit.uz dagi masala shifri **0024)**

Oybek akaning oilasi katta bo‘lib, uning uch o‘g‘li va to‘qqizta nabirasi bor. Ularni hammasini ta’minlash lozim. Shuning uchun Oybek aka haftada bir marta magazinga boradi.

Bir kuni Oybek aka magazinga keldi va “har bir k-chi tovar bepul” nomli aksiya bo‘layotganini ko‘rib qoldi. Oybek aka aksiyani shartlari bilan tanishib chiqdi. Xaridor tovarlari bilan kassaga borib chek oladi. Masalan chekda n ta Tovar bo‘lsa, u holda n/k eng arzoni unga tekinga tushadi.

Masalan, agar chekda 200, 100, 1000, 400 va 100 so‘mli beshta tovar va k = 2 bo‘lsa, u holda 100 so‘mluk ikkita Tovar ham tekinga tushadi. Jami bo‘lib xaridor 1600 so‘m to‘lashi zarur.

Oybek aka tovarlarni oldi va kassaga keldi. Tushundiki, agarda olgan tovarlarni bir nechta cheklarga bo‘linsa kamroq pul to‘lanar ekan.

Oybek akaga olgan tovarlarini bir nechta cheklarga bo‘lgan holda kammablag‘ to‘lashiga yordam bering.

Misoldagi Oybek aka olgan tovarlarni ikkita chekga ajratamiz. Birinchi chekka 1000 va 400 so‘mlisi, bunda 400 so‘mlisi tekinga tushadi. Ikkinchi chekda esa 100 so‘mli tovar tekinga tushadi. Demak jami bo‘lib 1300 to‘lanadi.

Kiruvchi ma'lumotlar: Birinchi satrda ikkita butun son berilgan n, k ($1 \le n \le 100\ 000$, $2 \le k \le 100$) – Oybek aka sotib olgan tovarlar soni va “har bir k-chi tovar bepul” aksiyasini parametri. Keyingi satr Oybek aka sotib olgan butun sonli n ta a_i ($1 \le a_i \le 10\ 000$) tovar narxlari.

Chiquvchi ma'lumotlar: Oybek aka to‘lagan minimum narxni chiqaring

Kiritishga misol	Chiqarishga misol
5 2 200 100 1000 400 100	1300
6 4 100 11 56 22 43 62	251

7-keys. Umumiy yig‘indi(acm.tuit.uz dagi masala shifri **0040)**

Ketma-ket berilgan sonlar yig‘indisini hisoblang.

Kiruvchi ma'lumotlar: N ta son ketma-ket berilgan $1 \le n \le 10^5$.

Chiquvchi ma'lumotlar: Sizdan ketma-ket berilgan sonlar yig‘indisini hisoblovchi dastur tuzish talab etiladi. Barchasining absolyut qiymati 10^9 dan oshmaydi.

Kiritishga misol	Chiqarishga misol
1 2 3 4 -1 -2 -3 -4	0

8-keys. Tarqatish(acm.tuit.uz dagi masala shifri **0044)**

Butun sonlardan iborat bo‘lgan massiv berilgan. Sizni vazifangiz ularni kamaymagan tartibda saralashingiz kerak. Massivning eng minimal va maximal sonlari oralig‘i 107 dan oshmaydi.

Kiruvchi ma'lumotlar: Birinchi satrda massiv elemntlari *N (1 < N < 3 000 000)* soni berilgan. Ikkinchi satrda moduli 2^{31} - 1 dan oshmaydigan *N* ta son joylashtirilgan.

Chiquvchi ma'lumotlar: Ushbu massivni kamaymaydigan tartibda chiqaradigan dastur tuzing. Ixtiyoriy ikkita son orasida bitta probel tursin.

Kiritishga misol	Chiqarishga misol
3 5 3 8	3 5 8

2 1559168841 1559168839	1559168839 1559168841

9-keys. Oddiy sortirovka(acm.tuit.uz dagi masala shifri **0045)**

Butun sonlardan iborat bo‘lgan massiv berilgan. Sizni vazifangiz ularni kamaymagan tartibda sortirovka qilishingiz kerak.

Kiruvchi ma'lumotlar: Birinchi satrda massiv elementlari *N (1 < N < 100000)* soni berilgan. Ikkinchi satrda moduli 10^9 dan oshmaydigan *N* ta son joylashtirilgan.

Chiquvchi ma'lumotlar: Ushbu massivni kamaymaydigan tartibda sortirovka qiladigan dastur tuzing. Ixtiyoriy ikkita son orasida bitta probel bor.

Kiritishga misol	Chiqarishga misol
10 1 8 2 1 4 7 3 2 3 6	1 1 2 2 3 3 4 6 7 8

10-keys. Ikkilik qidiruv(acm.tuit.uz dagi masala shifri **0049)**

Ikkita *A* va *B*massivlar berilgan. *B* massiv elementlarini *A* massivda bor yoki yo‘qligini aniqlovchi dastur tuzing.

Kiruvchi ma'lumotlar: Birinchi satrda *N va K* $(0 \le N,K\ 10^5)$ sonlari berilgan. Ikkinchi satrda *A* massivning *N* ta va uchunchi satrda *B* massivning *K* ta elementi berilgan. Elementlarining moduli $2 \cdot 10^9$ dan oshmaydi. *A* massivning elementlari kamaydigan qilib saralangan.

Chiquvchi ma'lumotlar: B massivning har bir *K* ta elementi *A* massivda bor bo‘lsa alohida satrda «YES», aks holda «NO» chiqaring.

Kiritishga misol	Chiqarishga misol
5 4 1 4 5 8 9 5 6 1 9	YES NO YES YES
10 10 -8 -6 -4 -4 -2 -1 0 2 3 3 8 3 -3 -2 2 -1 2 9 -8 0	NO YES NO YES YES YES

	YES NO YES YES

11- keys. Anagramma (acm.tuit.uz dagi masala shifri **2020)**

S1 satr S2 satrning anogrammasi deyiladi qachonki u S2 satrning belgilarini o‘rin almashtirishlaridan tashkil topgan bo‘lsa. Sizga S1 va S2 satrlar berilgan. S1 satr S2 satrning anogrammasi ekanligini aniqlovchi dastur tuzing.

Kiruvchi ma'lumotlar: Birinchi qatorda S1, ikkinchisida S2 berilgan. Har ikkala satr ham faqat katta lotin harflaridan tashkil topgan. Satrlar bo‘sh emas va belgilar soni 100000 dan oshmaydi.

Chiquvchi ma'lumotlar: Agar S1 qator S2 qatorning anogrammasi bo‘lsa YES aks holda NO chiqaring.

Kiruvchi	**Chiquvchi**
ABAA ABBA	NO
ABBA BABA	YES

12-keys. Satrni ochish (acm.tuit.uz dagi masala shifri **2030)**

Faqat katta lotin harflaridan tashkil topgan satrlarni ko‘rib chiqamiz. Masalan, AAAABCCCCCDDDD satrni qaraymiz. Bu satrni uzunligi 14 ga teng. Satr faqat katta harflardan tashkil topgan. Takrorlangan simvollarni o‘chirish va ularni son bilan almashtirish mumkin. Bu sonlar albatta takrorlanishlar sonini bildiradi. Shunday qilib, berilgan satrni 4AB5C4D ko‘rinishda tasvirlash mumkin. Bu satrning uzunligi 7 ga teng. Bu usulni biz satrlarni qadoqlash deymiz. Berilgan qadoqlangan satrni oling va undan berilgan satrni qayta tiklovchi dastur tuzing.

Kiruvchi ma'lumotlar: Bitta satrda qadoqlangan satr berilgan. Satrda faqatgina quyidagi ko‘rinishda beriladi. Masalan, nA. Bu yerda n ($2 \le n \le 99$)– takrorlanuvchi simvollar soni, A esa katta lotin harfi. Yoki A, ya’ni sonsiz simvol.Satrning maksimal uzunligi 80 dan oshmaydi.

Chiquvchi ma'lumotlar: Bitta satrda qayta tiklangan satrni chiqaring.

Kiruvchi	Chiquvchi
3A4B7D	AAABBBBDDDDDDD
22D7AC18FGD	DDDDDDDDDDDDDDDDDDDDDDAA AAAAACFFFFFFFFFFFFFFFFGD
4ABC	AAAABC

13-keys. Satrlarni solishtirish (acm.tuit.uz dagi masala shifri **2031)**

A va B satrlari berilgan, shu ikki satrni o‘zaro solishtiruvchi dastur tuzing, bunda katta va kichik belgilar bir xil deb hisoblansin.

Kiruvchi ma'lumotlar: Birinchi qatorda A satri, ikkinchi qatorda B satri kiritiladi, A va B satrlari katta va kichik lotin harflaridan iborat.

Chiquvchi ma'lumotlar: Agar A > B bo‘lsa “A>B”, agar A < B bo‘lsa “A<B”, agar A = B bo‘lsa “A=B” chiqaring.

Kiruvchi	Chiquvchi
Abdukarimov Hojiyev	A<B
Sunatullo Sirojiddin	A>B
Shaxzod sHAXzod	A=B

14-keys. Sevgi xatining sehri (acm.tuit.uz dagi masala shifri **2035)**

Jeyms do‘sti sevgilisiga atab yozgan xatni topib oldi. Jeyms hazilkash edi, va xatni o‘zgartirishga qaror qildi. U barcha so‘zlarni palindromga aylantirdi. Har bir so‘zda u harflarni faqatgina o‘zidan kichigigagina alishtirishi mumkin edi, masalan 'd' harfini u 'c' harfiga alishtirishi mumkin edi va bu bitta amal hisoblanar edi (u harflarni faqatgina 'a' harfigacha alishtirishi mumkin, 'a' harfini 'z' ga alishtira olmaydi). So‘zlarni palindrom qilish uchun sarflangan eng kam amallar sonini toping.

Kiruvchi ma'lumotlar: Birinchi qatorda T – testlar soni berilgan. Keyingi T ta qatorda bittadan so‘z berilgan. $1 \le T \le 10$ $1 \le$ длина слова $\le 10^4$

Chiquvchi ma'lumotlar: Har bir test uchun bitta son, masala shartini qanoatlantiruvchi qiymatni chiqaring.

Kiruvchi	Chiquvchi
4 Abc Abcba Abcd Cba	2 0 4 2

15-keys. So‘z (acm.tuit.uz dagi masala shifri **2039)**

Adhamni internetdagi bazi odamlar kata va kichik harflarni aralashtirib yozishi juda hayron qoldiradi. Shuning uchun u o‘zining brauzeriga kengaytma yaratishga qaror qildi. Bu kengaytma har bir so‘zdagi harflarni shunday o‘zgartiradiki, unda so‘z faqat katta harflardan yoki faqat kichik harflardan tashkil topgan bo‘lishi zarur. So‘zlarni o‘zgartirishda iloji boricha kam almashtirishlardan foydalanish zarur. Masalan, HoUse so‘zi house ga, ViP so‘zi esa VIP ga almashishi zarur. Agar so‘zda katta va kichik harflar soni bir hil bo‘lsa unda so‘zdagi harflarni kichik harflarga almashtirish zarur. Masalan maTRIx so‘zini matrix so‘ziga alishtirish zarur.

Kiruvchi ma'lumotlar: Bir qatorda s so‘z berilgan, u katta va kichik lotin harflardan iborat va 1 dan 100 gacha uzunlikka ega.

Chiquvchi ma'lumotlar: To‘g‘irlangan s so‘zini chiqaring.

Kiruvchi	Chiquvchi
HoUse	House
ViP	VIP
matrix	matrix

VI. BO`LIM

MUSTAQIL TA`LIM MAVZULARI

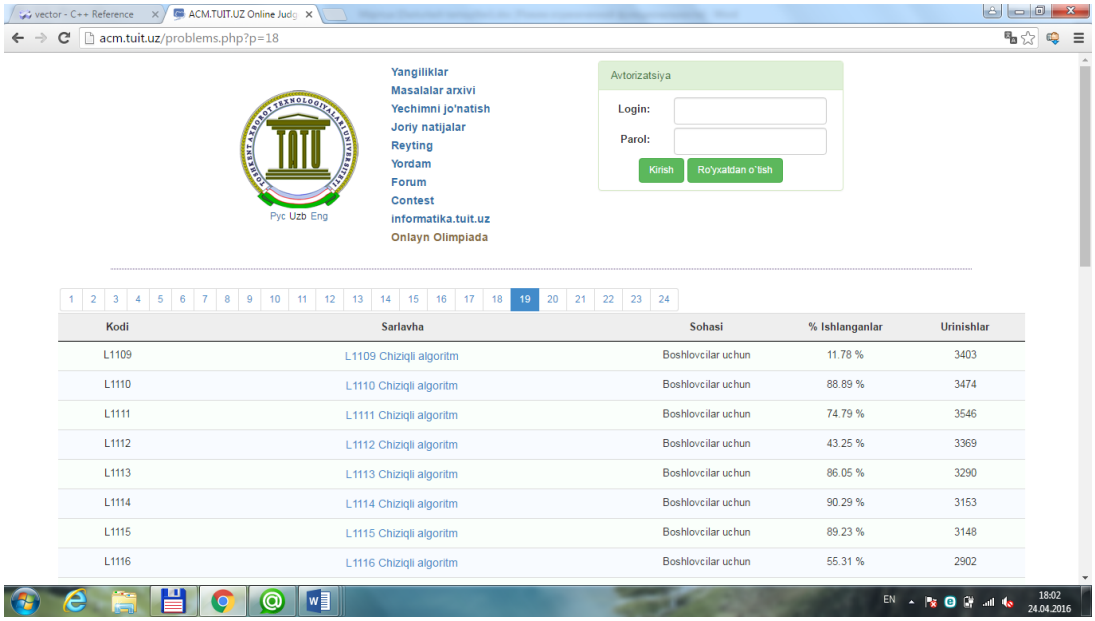
VI. MUSTAQIL TA’LIM MAVZULARI

Mustaqil ishni tashkil etishning shakli va mazmuni

Tinglovchi mustaqil ishni muayyan modulni xususiyatlarini hisobga olgan xolda quyidagi shakllardan foydalanib tayyorlashi tavsiya etiladi:

- me’yoriy xujjatlardan, o‘quv va ilmiy adabiyotlardan foydalanish asosida modul mavzularini o‘rganish;
- tarqatma materiallar bo‘yicha ma’ruzalar qismini o‘zlashtirish;
- avtomatlashtirilgan o‘rgatuvchi va nazorat qiluvchi dasturlar bilan ishlash;
- maxsus adabiyotlar bo‘yicha modul bo‘limlari yoki mavzulari ustida ishlash;
- tinglovchining kasbiy faoliyati bilan bog‘liq bo‘lgan modul bo‘limlari va mavzularni chuqur o‘rganish.

Mustaqi ta’lim uchun har bir tinglovchi dasturiy yechim to‘g‘riligini avtomatik testlovchi tizim (http://acm.tuit.uz) dan ro‘yxatdan o‘tgan xolatda, masalalar arxividagi L1101 dan L1130 gacha, L1201 dan L1230 gacha, L1301 dan L1326 gacha, L1401 dan L1430 gacha va 2S LabIsh N1 01 dan 2S LabIsh N1 23 gacha variant bo‘yicha majburiy va undan tashqari ixtiyoriy masalani tanlab ishlash, dasturni tizimga yuborish (tizim qabul qilguniga qadar).



VII. BO`LIM

GLOSSARIY

VII. GLOSSARIY

Termin	O‘zbek tilidagi sharhi	Ingliz tilidagi sharhi
!=	Teng emas operatori; mantiqiy inkor amali qiymati bilan birhil.	The inequality operator; compares values for inequality returning a bool.
#define	Makro deriktivalarni belgilash	a directive that defines a macro.
#include	Bir source fayl ichida boshqa bir faylag murojatni amalga oshirish mexanizmi	a mechanism for textual inclusion of one source file into another.
+=	add-and-assign operatori; masalan a+=b vazifazi jihatdan a=a+b bilan bir xil	add-and-assign operator; a+=b is roughly equivalent to a=a+b.
.c file	Dastur jodini o‘zida jamlovchi fayl	file containing definitions.
.cpp file	Dastur jodini o‘zida jamlovchi fayl	file containing definitions.
.h file	Sarlavha fayli	header file
Address	Hotira manzili	a memory location
Aggregate	Konstruktorsiz massiv yoki struktura	an array or a struct without a constructor
Algorithm	Hisoblashning aniq ketma-ketligi	a precise definition of a computation.
And	&& mantiqiy “va” (ko‘paytirish) operatori sinonimi	synonym for &&, the logical and operator.
ANSI	Amerika milliy standart agentligi.	The American national standards organization.
Application	Umumiy maqsadga ega dasturlar to‘plami	a collection of programs seen as serving a common purpose (usually providing a common interface to their users)
Bit	0 yoki 1 qiymatga ega birlik hotira	a unit of memory that can hold 0 or 1.
Bool	Mantiqiy tip. Bu tip faqat rost yoki yolg‘on qiymat qabul qiladi	the built-in Boolean type. A bool can have the values true and false.
Borland C++ Builder	C++ tilida visual dasturlashga ihtisoslashtirilgan IDE muhiti.	Borland's implementation of C++ together with proprietary libraries for Windows

		programming in an IDE
Bug	Xatolik termini	colloquial term for error.
Byte	Xotiradagi bir nechta belgilar yig‘indisi	a unit of memory that can hold a character of the C++ representation character set.
C++	Tizimli dasturlashni protsedurali qo‘llab quvvatlovchi dasturlash tili.	a general-purpose programming language with a bias towards systems programming that supports procedural programming, data abstraction, object-oriented programming, and generic programming.C++ was designed and originally implemented by Bjarne Stroustrup.
Char	Belgili tip. Har bir belgi 8 bit, ya’ni baytga teng.	character type; typically an 8-bit byte.
char*	Char massiviga ko‘rsatkich	pointer to a char or an array of char. Typically assumed to point to a C-style string.
Cin	Standart kiritish oqimi	standard istream.
Class	Foydalanuvchi belgilaydigan tur, sinf. Sinf foydalanuvchi funksiyasi, foydalanuvchi ma’lumotlari va kontentlari bo‘lishi mumkin.	a user-defined type. A class can have member functions, member data, member constants, and member types.
Compiler	C++ da yozilgan buyruqlarni mashina tiliga o‘girib beruvchi vosita.	the part of a C++ implementation that produces object code from a translation unit.
const	Faqatgina bir marta qiymat berish mumkin bo‘lgan o‘zgaruvchilarni e’lon qilsh.	attribute of a declaration that makes the entity to which it refers readonly.
copy()	Nusxa olish operatori	Copy operator
UML	Унификация қилинган моделлаштириш тили	Unified Modeling Language
OMG	Объектларни бошқариш гуруҳи	Object Management Group
4GL	Тўртинчи авлод тили	Fourth-Generation Language
ANSI	Америка миллий стандартлаш	American National Standards

	институти	Institute
AMPS		Advanced Mobile Phone Service
ERP	Корхона ресурсларини режалаштириш	Enterprise Resource Planning
CRM	Мижозлар билан ўзаро муносабатларни бошқариш	Customer Relations Management
SQL	Тузилмалашган сўровлар тили	Structured Query Language
OLAP	Хақиқий вақтда маълумотларга аналитик ишлов бериш	On-Line Analytical Processing
OLTP	Хақиқий вақтда транзакцияларга ишлов бериш	On-Line Transaction Processing
TCO	Эгалик қилишнинг ялпи қиймати	Total Cost of Ownership
JIT	Айни вақтида	Just-In-Time
LAN	Локал ҳисоблаш тармоғи	Local Area Network
MAN	Махаллий ҳисоблаш тармоғи	Metropolitan Area Network
WAN	Худудий ҳисоблаш тармоғи	Wide Area Network
ISO	Ҳалқаро стандартлаштириш ташкилоти	International Organization for Standardization
API	амалий дастурлаштириш махсус интерфейси	Application Programming Interface
WWW	Умумжаҳон ўргамчак тўри	World Wide Web
ASCII	Ахборот алмашишнинг Америка стандарти	American Standard Code for Information Interchange
LIFO	«Охирида келди, биринчи кетди» принципи	Last In, First Out
FIFO	«Биринчи келди, биринчи кетди» принципи	First In, First Out
PDA	персонал рақамли котиб	Personal Digital Assistant

VIII. BO`LIM

ADABIYOTLAR RO`YXATI

VII. ADABIYOTLAR RO‘YXATI

Maxsus adabiyotlar.

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. Bjarne Stroustrup. The C++ Programming Language, 4th Edition. Person Education, Inc. 2013. Third printing, April 2014.
4. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013. 488 b.

Internet resurslar.

1. http://www.stroustrup.com/4th.html
2. http://www.cplusplus.com/
3. http://acm.tuit.uz/ - дастурий ечим тўғрилигини автоматик тестловчи тизим.
4. http://acm.tuit.uz/forum/
5. http://acm.timus.ru/ – дастурларни тестловчи тизим.
6. http://codeforces.com/ - дастурий ечим тўғрилигини автоматик тестловчи тизим

EXPERT CONCLUSION

TO THE EDUCATIONAL-METHODOLOGICAL COMPLEX FOR THE COURSE OF RETRAINING PEDAGOGUE CADRES OF HIGHER EDUCATION ORGANIZATIONS IN THE DIRECTION OF "INFORMATICS AND INFORMATION TECHNOLOGIES"

These educational-methodological complexes were developed in accordance with defined requirements.

It consists of the: syllabus; theoretical and practical materials; assessment; presentations on every topic; glossary; tests; list of references.

The syllabus is written correctly. The sequence of topics proposed for study, focused on high-quality learning. Calendar-thematic plan corresponds to its content of the working program on discipline. Tests vary, allow to adequately assess the level of listeners' knowledge on the subject. Methodical recommendations for practical exercises provide the formation of basic skills to carry out research in the process of scientific knowledge and the theoretical foundation of professional tasks.

Presentation for lecture material is accurate and specific, it promotes better assimilation of discipline. The presented educational-methodical complexes in the direction of "Informatics and information technologies" informative, has a practical orientation, includes a sufficient number of diverse elements aimed at developing the mental and creative abilities of listeners.

In the content of the educational-methodological complexes, prospects for the development of computer science and information technologies as well as reforms and foreign experience in the field of information and communication technologies in the Republic of Uzbekistan have been taken into consideration and actual topics have been added.

Assignments of assessment are composed in the form of collection of cases.

Glossary includes reviews (components) in both Uzbek and English languages.

As the educational-methodological complexes is useful for learners in the direction of "Informatics and information technologies" it totally matches with the requirements of the course of teacher training and retraining and it is expedient to be used in education process.

Vice rector of ICT, TUIT Chul Soo LEE