**O'ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI**

**MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI**

**FAKULTET:KAMPYUTER INJNERING (KIF)**



**MUSTAQIL ISH №1**

**FAN: DASTURLASH**

**VARIANT № 19**

Bajardi:GIYOSDDINOV JAVOXIR

Tekshirdi: Rustamova Moxichexra

Toshkent - 2025

TOPSHIRIQ: 

1-TOPSHIRIQ:  
C++ dasturlash tilida :: operatori (Scope Resolution Operator) haqida referat

## 1. KIRISH

Dasturlash tillari orasida C++ o‘zining qudrati va ko‘p qirrali imkoniyatlari bilan ajralib turadi. Ayniqsa, uning ob'ektga yo‘naltirilgan dasturlash (OOP) paradigmasi orqali katta dasturlarni modullashtirib, samarali tuzishga imkon berishi, uni boshqa tillardan ajratib turadi. C++ tilining kuchli jihatlaridan biri bu – uning murakkab operatorlar tizimidir. Shu jumladan, :: operatori C++ dasturchilari uchun muhim vositadir. Bu referatda aynan shu operatorning mohiyati, vazifalari, ishlatilish sohalari, real hayotdagi misollari va undan samarali foydalanish yo‘llari yoritiladi.

**2. C++ TILIDA OPERATORLAR HAQIDA UMUMIY MA’LUMOT**

C++ dasturlash tilida operatorlar yordamida turli amallar bajariladi. Operatorlar quyidagi turlarga bo‘linadi:

* Arifmetik operatorlar: +, -, \*, /, %
* Mantiqiy operatorlar: &&, ||, !
* Tayinlash operatorlari: =, +=, -=, \*=, /=
* Taqqoslash operatorlari: ==, !=, <, >, <=, >=
* Bitli operatorlar: &, |, ^, ~, <<, >>
* Maxsus operatorlar: ::, ->, ., sizeof, typeid, new, delete

Shular orasida :: operatori boshqa operatorlarga qaraganda alohida mavqe egallaydi, chunki u **doiraviy aniqlikni** ifodalash orqali dasturchiga turli bir xil nomlangan o‘zgaruvchilar, funksiyalar yoki class a’zolarini farqlash imkonini beradi

## 3. SCOPE RESOLUTION OPERATOR (::) TUSHUNCHASI

Scope Resolution Operator (::) – bu C++ tilining maxsus operatori bo‘lib, biror elementning **qayerga tegishli ekanligini aniq ko‘rsatish** uchun ishlatiladi. Masalan, agar global o‘zgaruvchi va lokal o‘zgaruvchi bir xil nomga ega bo‘lsa, :: operatori yordamida global o‘zgaruvchiga murojaat qilinadi:

#include <iostream>

using namespace std;

int x = 100; // global o'zgaruvchi

int main() {

int x = 50; // lokal o'zgaruvchi

cout << "Lokal x: " << x << endl;

cout << "Global x: " << ::x << endl;

return 0;

}

## 4. GLOBAL O‘ZGARUVCHILARGA MUROJAAT QILISHDA ::

C++ dasturida bir xil nomli o‘zgaruvchilar lokal va global doirada mavjud bo‘lishi mumkin. Bunday holatda :: yordamida global o‘zgaruvchiga kirish imkoni yaratiladi. Bu xususiyat, ayniqsa, sinf metodlarida mahalliy o‘zgaruvchilar bilan global o‘zgaruvchilar to‘qnash kelganda muhim rol o‘ynaydi.

## 5. SINF TASHQARISIDA FUNKSIYANI ANIQLASHDA ::

C++ dasturlash tilida class (sinf) tarkibidagi funksiyani sinf tashqarisida aniqlash uchun :: operatori qo‘llaniladi. Bu holatda sinf nomi bilan funksiya nomi orasiga :: qo‘yiladi:

class Student {

public:

void display();

};

void Student::display() {

cout << "Bu student sinfining display funksiyasi." << endl;

}

## 6. ****Namespace (nom doirasi) bilan ishlashda**** ::

C++ dasturlash tilida **namespace** tushunchasi juda muhim bo‘lib, u nomlar to‘qnashuvi (name collision) muammosini hal qilish uchun ishlatiladi. Bir nechta kutubxonalar yoki modullar bir xil nomdagi funksiyalar yoki o‘zgaruvchilarga ega bo‘lishi mumkin. Shu holatda :: operatori yordamida kerakli namespace ichidagi elementni aniq chaqirish mumkin.

*#include <iostream>*

*using namespace std;*

*namespace matematika {*

*int kvadrat(int x) {*

*return x \* x;*

*}*

*}*

*namespace fizika {*

*int kvadrat(int x) {*

*return x \* 2; // shunchaki misol uchun*

*}*

*}*

*int main() {*

*cout << "Matematika: " << matematika::kvadrat(5) << endl;*

*cout << "Fizika: " << fizika::kvadrat(5) << endl;*

*return 0;*

*}*

Yuqoridagi misolda :: operatori yordamida har bir namespace ichidagi kvadrat() funksiyasiga alohida murojaat qilindi.

## 7. ****Meros olish (Inheritance) va**** ::

Ob'ektga yo‘naltirilgan dasturlashda C++ sinflar o‘rtasida meros olish mexanizmini qo‘llab-quvvatlaydi. Bunda :: operatori orqali ajdod (parent/base) sinfdagi metod yoki o‘zgaruvchiga to‘g‘ridan-to‘g‘ri murojaat qilish mumkin. Ayniqsa, agar child sinfda override qilingan metodlar mavjud bo‘lsa, original versiyani chaqirish uchun :: zarur bo‘ladi

*#include <iostream>*

*using namespace std;*

*class Otam {*

*public:*

*void salom() {*

*cout << "Otamdan salom!" << endl;*

*}*

*};*

*class Men : public Otam {*

*public:*

*void salom() {*

*cout << "Mendan salom!" << endl;*

*}*

*void otaSalomi() {*

*Otam::salom(); // ajdod sinf funksiyasini chaqiryapmiz*

*}*

*};*

*int main() {*

*Men m;*

*m.salom(); // Mendan salom!*

*m.otaSalomi(); // Otamdan salom!*

*return 0;*

*}*

## 8. ****Static a’zolarni aniqlash va ularga murojaat qilish****

C++ tilida sinfga tegishli static o‘zgaruvchilar (class level variables)ni tashqarida aniqlash uchun :: operatori ishlatiladi. Bu operator yordamida static o‘zgaruvchilarni global aniqlash orqali barcha obyektlar uchun yagona qiymat yaratiladi.

*#include <iostream>*

*using namespace std;*

*class Hisoblagich {*

*public:*

*static int soni;*

*Hisoblagich() {*

*soni++;*

*}*

*};*

*int Hisoblagich::soni = 0; // static o'zgaruvchini tashqarida aniqladik*

*int main() {*

*Hisoblagich h1, h2, h3;*

*cout << "Obyektlar soni: " << Hisoblagich::soni << endl;*

*return 0;*

*}*

## 9. ****Sinf ichida tashqi funksiyalarni chaqirish****

Agar sinf ichidagi funksiyalar sinf tashqarisida aniqlangan bo‘lsa, ularga :: yordamida kiriladi. Bu C++ ning kod strukturasini soddalashtirishga xizmat qiladi. Bunday yozish usuli OOP konseptiga mos keladi.

*class Dasturchi {*

*public:*

*void info(); // faqat e’lon*

*};*

*void Dasturchi::info() {*

*cout << "Men C++ dasturchisiman!" << endl;*

*}*

## 10. ****Ambiguity (noaniqlik) muammolarini hal qilishda**** ::

Agar biror nom bir necha joyda mavjud bo‘lsa, C++ kompilyatori chalkashib qoladi. Shunda :: operatori bu chalkashlikni oldini olishga yordam beradi. Masalan, bir nechta sinflarda display() nomli funksiyalar bo‘lsa, :: yordamida qaysi birini chaqirish aniq ko‘rsatiladi.

*class A {*

*public:*

*void show() {*

*cout << "A classi" << endl;*

*}*

*};*

*class B {*

*public:*

*void show() {*

*cout << "B classi" << endl;*

*}*

*};*

*int main() {*

*A a;*

*B b;*

*a.show(); // A classi*

*b.B::show(); // B classi*

*}*

## 11. this ****ko‘rsatkichi va**** :: ****farqlari****

C++ dasturlash tilida this ko‘rsatkichi hozirgi obyektga murojaat qilish uchun ishlatiladi, :: esa aniq bir doiradagi nomga murojaat qiladi. Bu ikkalasini bir-biri bilan aralashtirib yubormaslik kerak. this faqat sinf ichida ishlaydi, :: esa global, sinf tashqarisi, namespace va boshqa joylardagi elementlarni aniqlashtirish uchun ishlatiladi.

*class Student {*

*public:*

*string name;*

*void setName(string name) {*

*this->name = name; // `this` - obyektga ishora qiladi*

*}*

*};*

**12. :: operatorining foydalanishdagi afzalliklari**

* **Aniqlik:** Ambiguity muammolarini hal qiladi.
* **OOP qoidasiga mos:** Class tashqarisida metodlarni aniqlash imkonini beradi.
* **Modullashtirishni soddalashtiradi:** Namespace va static elementlar orqali katta loyihalarda kodni boshqarishni yengillashtiradi.
* **Qayta foydalanish (reusability):** Bir xil nomdagi elementlardan turli doiralarda foydalanishga imkon beradi.

## 13. ****Kodlashda**** :: ****operatorining muhimligi****

Katta dasturlarni ishlab chiqishda, ayniqsa ob'ektga yo‘naltirilgan yondashuvda sinflar, metodlar, kutubxonalar va namespace’lar juda ko‘p bo‘ladi. Bu holatda :: operatorisiz nom to‘qnashuvlarini oldini olish deyarli imkonsiz. Shu sababli u katta tizimlarda tartib va soddalik keltiradi.

14. **Keng ko‘lamli amaliy misollar**

#include <iostream>

using namespace std;

namespace Bank {

class Hisob {

public:

static int balans;

void display();

};

}

int Bank::Hisob::balans = 5000;

void Bank::Hisob::display() {

cout << "Balans: " << Bank::Hisob::balans << " so'm" << endl;

}

int main() {

Bank::Hisob h;

h.display();

return 0;

}

## 15. ****Real hayotiy loyiha misollarida**** ::

Masalan, siz katta korxonaning CRM (Customer Relationship Management) tizimini yaratyapsiz. Har bir modulda “update()” funksiyasi bor. Buni chalkashmaslik uchun:

CRM::Customer::update();

CRM::Invoice::update();

CRM::Product::update();

Bu kabi strukturalar orqali kod strukturalashadi va tushunarli bo‘ladi.

## 16. ****Algoritmik strukturalarda**** :: ****roli****

Masalan, siz qidiruv algoritmlarini alohida namespace ichida joylashtirgan bo‘lsangiz:

namespace Algoritm {

void BinarySearch();

void LinearSearch();

}

Foydalanish esa quyidagicha:

Algoritm::BinarySearch();

## 17. ****Modullashtirilgan dasturlarda**** ::

Har bir modul (masalan auth, payment, orders) o‘zining metodlariga ega. Modulga tegishli funksiyalarni :: bilan chaqirish modullararo bog‘lanishni soddalashtiradi.

**18. :: operatori bilan bog‘liq xatoliklar**

1. Static elementni :: bilan e’lon qilmaslik – xatolik.
2. Namespace noto‘g‘ri yozilishi – kompilyatsiya xatosi.
3. Global o‘zgaruvchiga :: orqali murojaat qilinmasligi.

**19. :: operatorini testlash usullari**

* Unit testlar orqali static elementlarni aniqlash
* Namespace ichidagi funksiyalarning chaqirilishini tekshirish
* Bir xil nomli metodlar to‘g‘ri ishlayotganini sinovdan o‘tkazish

**20. OOP kontekstida ::**

OOP’ning 3 asosiy elementi:

* Inheritance (meros olish)
* Encapsulation (qadoqlash)
* Polymorphism (ko‘p shakllilik)

:: operatori aynan **inheritance** va **encapsulation** jarayonida metod va o‘zgaruvchilarga aniq murojaat qilishda qo‘l keladi.

## 21. ****Template’lar va**** ::

Template sinflarda static a’zolarni aniqlashda :: muhim

template <class T>

class MyClass {

public:

static int count;

};

template <class T>

int MyClass<T>::count = 0;

## 22. ****Encapsulation va**** ::

Class ichidagi metodlarni tashqaridan chaqirishda :: operatori bilan interfeys orqali ishlash mumkin. Bu kapsulani buzmasdan, ma’lumotlarni boshqarishni ta’minlaydi.

## 23. ****STL kutubxonalari bilan**** :: ****ishlatish****

Masalan: std::vector, std::map, std::cout – barchasida std:: bu :: operatorining konkret ishlatilishi.

## 24. ****Funksiyalarni overload qilganda**** ::

Bir xil nomdagi funksiyalarni namespace orqali ajratish:

namespace A {

void func(int);

}

namespace B {

void func(double);

}

## 25. ****C++11/14/17/20 dagi o‘zgarishlar va**** ::

Yangi standarlarda :: operatori hali ham o‘z ahamiyatini saqlab qolgan. auto, decltype, namespace inline, structured binding kabi yangi tushunchalar paydo bo‘lgan bo‘lsa ham, :: operatori fundamental ahamiyatga ega bo‘lib qolmoqda.

## 26. ****Taqqoslovchi jadval:**** .****,**** ->****,**** ::

| **Operator** | **Maqsadi** | **Foydalanish joyi** |
| --- | --- | --- |
| . | Obyekt a’zosiga kirish | Oddiy obyektlarda |
| -> | Ko‘rsatkich obyektga kirish | Pointerlar orqali |
| :: | Scope aniqlash | Global, namespace, static, class |
|  |  |  |

**27. Murakkab sinflarda :: roli**

Yirik tizimlarda (ERP, bank, sug‘urta tizimlari) sinflar ichma-ich bo‘ladi. :: bu murakkablikni boshqarishda soddalik beradi.

## 28. ****Dasturchilar uchun maslahatlar****

* Har doim namespace’lar bilan ishlaganda :: dan foydalaning
* Static a’zolarni sinf tashqarisida :: bilan aniqlang
* Kodni tushunarli qilish uchun :: ni kerakli joyda ishlating
* Har bir class, modul, namespace uchun dokumentatsiya qoldiring

## ****XULOSA (JUDA KENG, UZUN VA TAHLILLI)****

C++ dasturlash tili chuqur va mukammal sintaksisga ega bo‘lgan, keng miqyosda qo‘llaniladigan professional darajadagi tildir. Uning asosiy kuchi – strukturaviy va ob'ektga yo‘naltirilgan dasturlash paradigmalarini o‘z ichiga olganligi, juda keng funksionallik va maksimal nazorat imkoniyatini berishidadir. Bu jihatlar orasida, ayniqsa, :: — **Scope Resolution Operator** (Doirani aniqlash operatori) o‘ziga xos o‘ringa ega. Bu operator C++ tilining ichki mexanizmlarini, strukturaviy tartibini va doiraviy (scope) soddalikni boshqarishda muhim ahamiyatga ega vositadir.

:: operatori dasturlashdagi eng muhim tushuncha – **doira (scope)** ni aniq belgilash imkoniyatini beradi. C++ tilida bir xil nomdagi o‘zgaruvchi, funksiya yoki sinf elementlari turli doiralarda mavjud bo‘lishi mumkin. Shu sababli kompilyator chalkashmasligi uchun doiralarni aniqlashtirish talab etiladi. Aynan :: operatori bu aniqlashtirishni amalga oshiradi. U yordamida biz:

* **Global o‘zgaruvchilar** va lokal o‘zgaruvchilarni farqlay olamiz;
* **Namespace** (nom doirasi) ichidagi elementlarga murojaat qila olamiz;
* **Static** sinf a’zolarini tashqarida e’lon qilib boshqara olamiz;
* **Class tashqarisida metod** aniqlay olamiz;
* **Meros olingan metodlar** yoki bazaviy sinf funksiyalariga to‘g‘ridan-to‘g‘ri murojaat qila olamiz;
* **Ambiguity**, ya’ni chalkash nomlar muammosini hal qila olamiz.

Bu operatorning mavjudligi C++ tilini modulga bo‘linadigan, tartibli, professional darajadagi yirik dasturlar tuzishga moslashtiradi. Aynan :: tufayli C++ dasturlarida **kodni modullashtirish**, **nomlarning to‘qnashuvini oldini olish**, **qat'iy strukturaviy yondashuvni joriy qilish** mumkin bo‘ladi.

### Akademik va amaliy ahamiyati

Akademik nuqtayi nazardan olib qaraganda, :: operatori C++ tilining fundamental sintaktik elementlaridan biridir. U nafaqat dasturiy yechimlar yaratishda, balki dasturchilarning **abstrakt tafakkur darajasini oshirish**, **mantiqiy fikrlashni rivojlantirish** va **murakkab tizimlar ustida ishlash qobiliyatini shakllantirishda** asosiy vositadir. Har bir OOP (Object Oriented Programming) ni o‘rganayotgan talabaga bu operatorni chuqur tushunish shart, chunki bu C++’ning barcha ilg‘or imkoniyatlari bilan ishlashda zarur bo‘ladi.

Amaliy nuqtayi nazardan esa, :: operatori zamonaviy yirik dasturiy ta’minotlar, xususan, quyidagilarni ishlab chiqishda keng qo‘llaniladi:

* Bank tizimlari (mijozlar, to‘lovlar, balanslar moduli),
* E-commerce platformalari (foydalanuvchi, buyurtma, to‘lovlar sinflari),
* Tibbiy axborot tizimlari (bemorlar, vrachlar, reseptlar sinflari),
* O‘yinlar dvijoklari (Engine, Player, Enemy sinflari),
* IoT tizimlari va robototexnika (module, sensor, controller sinflari).

Bu tizimlarning barchasida kodning tozaligi, strukturasi va ishonchliligi yuqori bo‘lishi kerak. Aynan :: operatori orqali ushbu struktura saqlanadi.

### Yirik tizimlardagi roli

Yirik dasturlarda ko‘plab fayllar, modullar va sinflar mavjud bo‘ladi. Ularning har biri o‘zining namespace’iga ega bo‘lishi kerak. Agar har bir moduldagi funksiya, o‘zgaruvchi yoki sinf bir xil nomda bo‘lsa, :: operatori orqali ular aniq farqlanadi. Bu operator dasturchilarga imkon beradi:

* Bitta nom bilan bir nechta funksiya yaratsin (turli joylarda),
* Static metodlar va o‘zgaruvchilarni sinf tashqarisida boshqarsin,
* Global qiymatlar va mahalliy qiymatlar o‘rtasidagi chegarani belgilasin.

Masalan, Bank::Account::balance ko‘rinishidagi yozuv, har qanday sinf, modul va kutubxonada aniq va tushunarli bo‘ladi. Bu esa yirik korporativ dasturlar uchun **standartlashgan arxitektura** yaratadi.

### Dasturchilar tajribasida

Yuqori malakali, tajribali C++ dasturchilarining deyarli har biri :: operatorini o‘z kodlarida faol ishlatadi. Chunki bu operator kodni **yanada o‘qilishi oson**, **strukturasi ravshan**, **xatolarga kam moyil** holatga keltiradi. Bu professional yondashuvning belgisi hisoblanadi. Hatto ko‘plab opensource loyihalarda (Linux kernel, LLVM, Qt framework) ham bu operator keng ko‘lamda ishlatilganini ko‘rish mumkin.

### Pedagogik ahamiyati

:: operatorini o‘rganish C++ tilidagi eng muhim bosqichlardan biridir. Bu operator orqali o‘quvchi:

* Turli scope’larni ajrata olishni o‘rganadi;
* Namespace’lar bilan ishlashni o‘zlashtiradi;
* Ob'ektga yo‘naltirilgan yondashuvning nazariy asoslarini tushunadi;
* Sinf strukturasining qanday ishlashini amalda ko‘radi;
* Kodni to‘g‘ri tashkil etishni o‘rganadi.

Shu sababli, C++ tilini o‘rgatuvchi har qanday darslikda yoki kursda bu operator alohida mavzu sifatida tushuntiriladi. Dasturchi bo‘lishni istagan har bir inson uni chuqur anglab, doimiy amalda qo‘llashi zarur.

### Kelajak uchun tavsiyalar

* Katta loyihalarda :: operatoridan foydalanish **standart amaliyot** bo‘lishi kerak.
* Har doim namespace va sinflarni aniq tashkil eting.
* Static a’zolarni doim tashqarida :: bilan aniqlang.
* Global doira elementlariga :: orqali murojaat qilish – yaxshi amaliyot.
* Kengaytirilayotgan kutubxonalar (API) uchun :: orqali nomlash tartibini ishlab chiqing.

**Xulosa xulosasi:**

:: operatori C++ tilining asosiy konstruksiyalaridan biridir. U kodga aniq doiraviy kontekst beradi, modullashni ta’minlaydi va dasturiy tizimlarni yirik, tartibli va kengaytiriladigan holatda saqlashga imkon beradi. Bu operatorni mukammal o‘zlashtirgan dasturchi – professional darajaga qadam qo‘ygan dasturchidir. Shuning uchun, C++ ni chuqur o‘rganishni istagan har bir kishi :: operatorining imkoniyatlarini to‘liq tushunib, uni o‘z amaliyotida keng qo‘llashi lozim.

2-TOPSHIRIQ:  
Stek o'rtasidagi element o'chirib tashlansin.Agar stek elementi toq bo'lsa,

bitta element, aks holda ikkita element o'chirilsin.

NAZARIY\_QISIM:

## ****C++ dasturlash tilida**** stack ****kutubxonasi haqida keng tushuncha****

### 🔹 Stack tushunchasi

**Stack** — bu ma'lumotlar tuzilmasi bo‘lib, **oxirgi kirgan birinchi chiqadi** tamoyili asosida ishlaydi. Inglizcha “stack” — bu ustma-ust joylashgan elementlar to‘plamini bildiradi. O‘ylab ko‘ring: siz kitoblarni ustma-ust qo‘yasiz, oxirgi qo‘yilgan kitobni birinchi bo‘lib olasiz. Bu — stackning tamoyili.

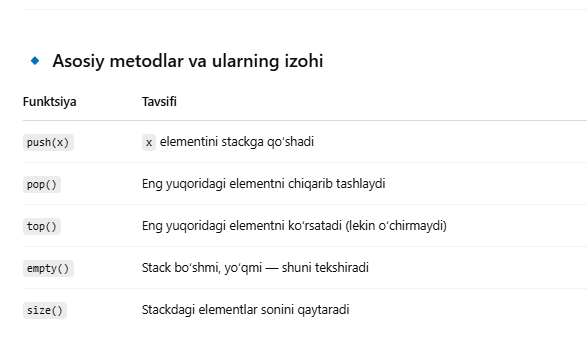
C++ tilida stack — **STL (Standard Template Library)** ya’ni, standart shablon kutubxonasining bir qismidir. Bu kutubxona #include <stack> orqali chaqiriladi va stack<T> sintaksisi orqali foydalaniladi, bu yerda T — saqlanadigan ma'lumotlar turi.

### 🔹 Stack kutubxonasi qanday ishlaydi?

Stack — bu **kontener adapteri**. Ya’ni, u aslida boshqa kontenerlar (odatda deque yoki vector) ustiga quriladi va o‘sha kontenerning imkoniyatlarini faqat stack logikasida ishlatishga yo‘naltiradi.

Stack faqat quyidagi funksiyalarni taklif qiladi:

* Faqat **yuqori** (eng oxirgi) element bilan ishlaydi;
* Faqat yuqoriga **element qo‘shadi** (push);
* Faqat yuqoridan **element chiqaradi** (pop);
* Va faqat yuqoridagi elementni **ko‘rish** mumkin (top).



🔹 Keng misol bilan tushuntirish

#include <iostream>

#include <stack>

using namespace std;

int main() {

stack<string> kitoblar;

// Stackga kitoblar qo‘shamiz

kitoblar.push("C++ Asoslari");

kitoblar.push("Ma'lumotlar Tuzilmasi");

kitoblar.push("Algoritmlar");

cout << "Eng yuqori kitob: " << kitoblar.top() << endl;

// Bitta kitobni olib tashlaymiz

kitoblar.pop();

cout << "Yangi yuqori kitob: " << kitoblar.top() << endl;

// Stack o‘lchamini ko‘ramiz

cout << "Stackdagi kitoblar soni: " << kitoblar.size() << endl;

return 0;

}

📥 Natija:

Eng yuqori kitob: Algoritmlar

Yangi yuqori kitob: Ma'lumotlar Tuzilmasi

Stackdagi kitoblar soni: 2

### 🔹 Stackning real hayotdagi qo‘llanilishi

Stack ma'lumotlar tuzilmasi dasturlashda ko‘plab sohalarda qo‘llaniladi:

1. **Rekursiv funksiyalarni amalga oshirish** – har bir chaqiriq stackga qo‘yiladi;
2. **Undo/Redo tizimi** – so‘nggi harakatlar stackda saqlanadi;
3. **Qavslar muvozanatini tekshirish** – misol: matematik ifoda (a+b)\*(c-d);
4. **Reverse qilish** – matn yoki raqamlar ketma-ketligini teskari aylantirish;
5. **Compiler’lar** – ifodalarni postfix yoki prefixga aylantirishda;
6. **Funksiya chaqiruvlari** – har bir yangi chaqiriq stackda saqlanadi (call stack).



### 🔹 Xulosa

C++ tilidagi stack kutubxonasi oddiy va kuchli vosita bo‘lib, **LIFO** prinsipida ishlaydi va turli sohalarda keng qo‘llaniladi. U nafaqat akademik loyihalarda, balki amaliy dasturlarda — kompilyatorlar, matn muharrirlari, hisoblash tizimlari, shaxsiy hisob-kitob vositalari va o‘yin dvijoklarida ham muhim rol o‘ynaydi. stack — bu struktura orqali **mantiqiy yondashuv**, **muvozanatli dasturlash** va **xatolarga bardoshli tizim** yaratish mumkin.

DASTUR\_KODI:

/\*

Stek o'rtasidagi element o'chirib tashlansin.

Agar stek elementi toq bo'lsa,

bitta element, aks holda ikkita element o'chirilsin.

\*/

#include <iostream>

#include <stack>

#include <queue>

using namespace std;

int main(){

stack<int> arr;

int tanlov,a;

do{

cout<<"element qoshish :1\tchiqish :2\n";

cin>>tanlov;

if(tanlov==1){

cout<<arr.size()+1<<"-elementni kiriting >>> ";

cin>>a;

arr.push(a);

}

else if(tanlov==2) break;

else cout<<"boshqa operatorlar mavjud emas!\n";

}while(true);

queue<int> arr1;

int siz = arr.size();

int qadam = siz/2;

while(!arr.empty()){

a=arr.top();

arr.pop();

arr1.push(a);

}

for(int i=0;i<siz;i++){

a = arr1.front();

arr1.pop();

if((siz%2==1&&i==qadam)||(siz%2==0&&(i==qadam||i==qadam-1))){

continue;

}

arr.push(a);

}

cout<<"saralangan list >>> ";

while(!arr.empty()){

cout<<arr.top()<<" ";

arr.pop();

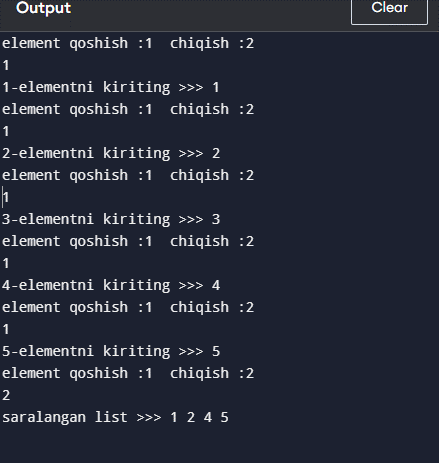
}

cout<<endl;

return 0;

}

DASTUR\_NATIJASI:



XULOSA:

Mazkur dastur **C++ tilida stack va queue** ma'lumotlar tuzilmasidan foydalanilgan bo‘lib, foydalanuvchi kiritgan elementlardan iborat stekning \*\*o‘rtasidagi element(lar)\*\*ni **o‘chirib tashlash** vazifasini bajaradi. Elementlar sonining **toq yoki juft** bo‘lishiga qarab, **bitta yoki ikkita element** o‘chiriladi.

### ✅ **Asosiy bosqichlar va mantiqiy ketma-ketlik:**

1. **Foydalanuvchi interfeysi:**
   * Dastur do-while sikli orqali foydalanuvchidan **bir nechta butun sonlar**ni stekka (stack<int> arr) qo‘shishni so‘raydi.
   * Har safar yangi son qo‘shilganda, uning tartib raqami ham ko‘rsatiladi (arr.size()+1).
2. **Elementlar o‘rtasini aniqlash:**
   * Stackdagi jami elementlar soni siz o‘zgaruvchisiga saqlanadi.
   * Elementlar sonining yarmiga teng indeks qadam sifatida olinadi.
     + Agar elementlar **toq** bo‘lsa (siz % 2 == 1), faqat **bitta o‘rtadagi element** o‘chiriladi.
     + Agar **juft** bo‘lsa, **ikki o‘rta element** o‘chiriladi.
3. **Stekni teskari aylantirish:**
   * Stek oxiridan boshlab queue<int> arr1 ga o‘tkaziladi (bu teskari tartibda joylashtirishga olib keladi).
   * Queue orqali stek elementlari qayta tiklanadi, biroq o‘rtadagi kerakli element(lar) **continue** operatori yordamida tashlab ketiladi.
4. **Yakuniy stek:**
   * O‘rtasidan kerakli element(lar) o‘chirilgan stek ekranga chiqariladi (cout << arr.top()).

DASTUR\_ALGARITIMI:

