

C++ TILIDA TAKRORLANISH OPERATORLARI (WHILE, DO WHILE , FOR

Shermatova Hilola Mirzayevna

Farg‘ona davlat universitet

Fizika –matematika fakulteti

Axborot texnologiyalari kafedrası dotsenti

Shermatovahilola1978@gmail.com

Zokirova Shaxlo Sarvar qizi

Axborot tizimlari va

texnologiyalar yo‘nalishi I kurs talabasi

shaxlozakirova7@gmail.com

Annotatsiya : Ushbu maqolada C++ dasturlash tilida takrorlanish operatorlari – while, do-while va for – ning ishlash prinsiplari, ularning farqlari va qaysi hollarda qaysi tsikl turi qulayroq ekani yoritilgan. Takrorlanish operatorlari dastur kodini soddalashtirish va bir xil amallarni bir necha marta bajarishga imkon beradi. Maqolada har bir tsiklning sintaksisi, ishlash mexanizmi, qo‘llash sohalari va amaliy misollar bilan tushuntirib berilgan. While tsikli shart tekshirilganidan keyin bajariladigan tsikl bo‘lsa, do-while tsikli kamida bir marta bajariladigan tsikl hisoblanadi. For tsikli esa iteratsiyalar soni oldindan ma‘lum bo‘lganda eng qulay tanlovdir. Shuningdek, maqolada cheksiz tsikllar, ichma-ich joylashgan tsikllar, break va continue operatorlari, hamda tsikllarning samarali ishlatilishi bo‘yicha maslahatlar berilgan. Ushbu maqola C++ dasturlash tili bilan tanish bo‘lgan yoki uni o‘rganayotgan dasturchilar uchun foydali qo‘llanma bo‘lib xizmat qiladi.

Annotation: This article discusses iteration operators in the C++ programming language—while, do-while, and for—their working principles, differences, and the scenarios in which each loop type is most suitable. Iteration operators simplify code and allow the execution of repetitive tasks multiple times. The article explains the syntax, working mechanisms, application areas, and practical examples for each loop. The while loop executes only if the condition is met, whereas the do-while loop executes at least once before checking the condition. The for loop is

the most convenient choice when the number of iterations is known in advance. Additionally, the article covers infinite loops, nested loops, the break and continue statements, and provides tips for the efficient use of loops. This article serves as a useful guide for programmers who are familiar with or are learning the C++ programming language.

Аннотация: В данной статье рассматриваются операторы циклов в языке программирования C++ — while, do-while и for, их принципы работы, различия и наиболее удобные случаи их использования. Операторы циклов упрощают код и позволяют выполнять одинаковые действия несколько раз. В статье объясняется синтаксис каждого цикла, механизм его работы, области применения и приводятся практические примеры. Цикл while выполняется только при выполнении условия, тогда как цикл do-while выполняется как минимум один раз. Цикл for является наилучшим выбором, когда количество итераций известно заранее. Кроме того, в статье рассматриваются бесконечные циклы, вложенные циклы, операторы break и continue, а также даются советы по эффективному использованию циклов. Данная статья является полезным руководством для программистов, знакомых с языком программирования C++ или изучающих его.

Kalit so‘zlar : stikl, shart, qiymat, qadam, takrorlanish operatorlari ,while, do while, for, stikl turi, amallar, dastur kodi, break .

Keywords : loop ,condition, value ,step ,looping constructs ,while ,do while , for, loop type, operations ,program code , break.

Ключевые слова : цикл , условие, значение, шаг, операторы повторения, while, do while, for, тип цикла, операции ,программный код, break.

KIRISH

Dastur kodining biror qismining ko‘p marta bajarishi sikl hisoblanadi. Dastur kodining qandaydir qismini qandaydir shart asosida birnecha marta bajartirish uchun dasturlashda sikldan foydalaniladi. Agar shart rost bo‘lsa sikl davom qiladi. Aks holda to‘xtatiladi. Agar shart hamisha rost bo‘lsa bunday sikl cheksiz sikl deb ataladi.

C++ da siklni tashkil qilish uchun while, do while va for operatorlari mavjud. Siklni o‘rganish uchun eng oson misol bu 1 dan n gacha natural sonlarning

yig‘indisini($1+2+3+\dots+n$) topish dasturini tuzish. Bu yig‘indini takrorlanish jarayoni orqali hisoblash uchun 1 dan n gacha sonlarni birma-bir qo‘shib chiqish lozim. Yig‘indining dastlabki qiymatini 0 ga tenglaymiz. Siklning har bir qadamida quyidagi amallar bajariladi:

$s=0;$

1-qadam. $s=s+1=0+1=1;$

2-qadam. $s=s+2=1+2=3;$

3-qadam. $s=s+3=3+3=6;$

4-qadam. $s=s+4=6+4=10;$

5-qadam. $s=s+5=10+5=15;$

i -qadam. $s=s+i;$

n -qadam. $s=s+n;$

Har bir qadamda bir xil amal bajariladi, yig‘indining yangi qiymatini hosil qilish uchun uning avvalgi qadamdagi qiymatiga navbatdagi natural son qo‘shiladi.

1) while sikli.

Bu siklda shart oldindan qo‘yiladi. Agar shart rost bo‘lsa sikl tanasi bajariladi. Aks holda sikl to‘xtab undan keyingi qadamga o‘tiladi.

while (shart) {sikl tanasi}

1 dan n gacha sonlar yig‘indisini topish uchun har bir qadamda navbatdagi sonni qo‘shib borish uchun i o‘zgaruvchi e’lon qilamiz.

#include

using namespace std;

int main()

{**int** $s = 0, i = 1, n;$

cout<<"n=";

cin>>n;

while ($i \leq n$) {

$s += i;$

$i++;$ }

cout<<"s="<<s;}

Dastur kodini bir boshdan qarab chiqamiz. Bizga uchta o‘zgaruvchi kerak. Birinchi o‘zgaruvchi n soni, ikkinchi o‘zgaruvchi sanab borish uchun ishlatiladigan i o‘zgaruvchisi, uchinchi yig‘indining qiymatini saqlash

uchun s o‘zgaruvchi. Siklni boshlashdan oldin yig‘indining qiymatini nolga tenglaymiz, shunda unga qandaydir sonni birinchi marta qo‘shganimizda uning o‘zi hosil bo‘ladi. i o‘zgaruvchining dastlabki qiymatini 1 ga tenglaymiz, chunki 1 dan boshlab yig‘indiga qo‘shib borishimiz lozim. Agar $i \leq n$ shart bajarilsa u holda i ni yig‘indiga qo‘shamiz ($s += i$ bu $s = s + i$ ning qisqacha yozilishi) va i ning qiymatini orqali birga oshiramiz ($i++$ bu inkrement).

Cheksiz sikl.

while yordamida cheksiz sikl hosil qilish uchun shart ifodaga hamisha rost qiymat qabul qiladigan mantiqiy ifoda, o‘zgaruvchi yoki rost konstanta qiymatini yozishimiz mumkin.

while (1)

```
{ cout<<"Cheksiz sikl\n"; }
```

2) do while sikli.

do while sikli while sikliga o‘xshash, farqi shart sikl oxirida tekshiriladi va shart bajarilsin yoki bajarilmasin kamida bir marta(1-sikl) sikl bajariladi.

1 dan n gacha sonlar yig‘indisi quyidagicha yoziladi:

#include

```
using namespace std;
```

```
int main() {
```

```
int s = 0, i = 1, n;
```

```
cout<<"n=";
```

```
cin>>n;
```

```
do { s += i; i++; }
```

```
while (i <= n);
```

```
cout<<"s="<<s;}
```

Bu siklda i o‘zgaruvchinig qiymati qanday bo‘lishidan qat’iy nazar sikl bir marta aylanadi. Bu siklni sonni kiritishda uning to‘g‘riligini tekshirish va toki to‘g‘ri kiritilmaguncha kiritishni davom qildirish uchun foydalanishimiz mumkin.

Masalan, yuqoridagi masalamizda n soni natural bo‘lishi kerak, agar natural son kiritilmasa yana kiritishni so‘rash lozim:

```
do { cout<<"n=";cin>>n; }
```

```
while (n < 1);
```

3) for sikli.

For sikli sintaksisi quyidagicha:

for(sikl boshlanishidan oldingi amallar; sikl davom etish sharti; siklning har bir iteratsiyasi oxiridagi amallar) {sikl tanasi;}

Iteratsiya deb siklning bir marta bajarilishiga aytiladi. Agar ma’lum qadam bilan bitta o‘zgaruvchining qiymatini o‘zgartirib takrorlanuvchi jarayon amalga oshirish lozim bo‘lsa, u holda uni quyidagicha xususiy holda yozishimiz mumkin:

for(<o‘zgaruvchi tipi> o‘zgaruvchi =boshlang‘ich qiymat; o‘zgaruvchi <=oxirgi qiymat; o‘zgaruvchi +=sikl qadami) { sikl tanasi;}

1 dan n gacha sonlar yig‘indisini topish uchun quyidagicha sikl amalga oshirishimiz mumkin:

```
#include
using namespace std;
int main()
{int s = 0, n;
cout<<"n=";
cin>>n;
for (int i = 1; i <= n; i++)
{ s += i; }
cout<<s;}
```

Bu siklda *i* ning qiymati sikl boshlanishidan avval 1 ga teng qiymatni qabul qiladi. Yana bitta iteratsiya qilish uchun bajarilishi kerak bo‘lgan shart $i \leq n$, agar shart rost bo‘lsa, yana bitta iteratsiya bajariladi, iteratsiya oxirida *i* ning qiymati birga oshiriladi($i++$). Keyingi har bir iteratsiyada for siklining ikkinchi va uchinchi qismlari bajariladi, 1-qismi boshqa bajarilmaydi. Eng oxirgi iteratsiyadan oxirida *i* ning qiymati oshirilgach $n+1$ ga teng bo‘ladi va keyingi iteratsiyada shart yolg‘on qiymat qabul qilganligi sababli ($n+1 \leq n$ yolg‘on qiymat qabul qiladi) sikl aylanishi tugaydi.

Sikl o‘zgaruvchisi *i* haqiqiy son ham bo‘la oladi. Masalan 1 dan 10 gacha sonlarni 0.01 qadam bilan chiqarish uchun ya’ni 1, 1.01, 1.02, 1.03, ..., 10 sonlarini chiqarish uchun quyidagicha sikl yoziladi.

```
for (double x = 1; x <= 10; x += 0.01) {
cout<<x<<" "; }
```

x sikl parametri bu safar haqiqiy qiymatni qabul qiladi va har bir iteratsiya oxirida qiymati 0.01 ga oshiriladi.

for siklining uchta qismidan istalgan qismini yozmaslik mumkin:

```
double x = 1
```

```
for (; x <= 10; x += 0.01) {
```

```
cout<<x<<" "; }
```

bu kod avvalgi yozilgani bilan bir xil, faqat x=1 dastlabki qiymatni o‘zlashtirish for ichida yozilmadi.

```
double x = 1;
```

```
for (; ; x += 0.01)
```

```
{ cout<<x<<" "; }
```

bu kod qismida x ning qiymati 1 dan boshlab 0.01 qadam bilan oshirib boriladi, lekin to‘xtash sharti yozilmadi, shuning uchun cheksiz sikl hosil bo‘ladi.

```
double x = 1;
```

```
for (; ;)
```

```
{ cout<<x<<" "; }
```

Bu holatda esa x ning qiymati iteratsiya oxirida o‘zgartirilmadi shuning uchun cheksiz ko‘p marta x ning dastlabki qiymati 1 chirariladi.

break operatori.

break operatori siklni uning bajarilish sharti rost qiymat qabul qilishiga qaramasdan to‘xtatish uchun qo‘llaniladi. Yuqoridagi x ning qiymati 1 dan 100 gacha 0.01 qadam bilan oshirib boradigan misolda

```
double x = 1;
```

```
for (; ;) {
```

```
if (x > 100.000001)
```

```
break;
```

```
cout<<x<<" ";
```

```
x += 0.01; }
```

break operatorining ishlatishga misollardan biri berilgan sonning tub yoki tub emasligini aniqlaydigan dastur yozish.

Sonning tub ekanligini aniqlash uchun uni 2 dan gacha bo‘lgan sonlarga bo‘linishini tekshiramiz. Agar ulardan biriga qoldiqsiz bo‘linadigan bo‘lsa, u holda

bu son tub emas. 103 sonining tub ekanligini aniqlash uchun 2,3,4,5,6,7,8,9 va 10 sonlariga bo‘linishini tekshiramiz.

```
#include
using namespace std;
int main() {
    int n;
    cin>>n;
    bool is_prime = true;
    for (int i = 2; i*i <= n; i++)
        if (n % i==0) {
            is_prime = false;
            break;    }
    if (n==1)
        is_prime = false
    if (is_prime)
        cout<<"Tub";
    else
        cout<<"Tub emas";}
```

Dastur kodini taxlil qilib chiqamiz. `cin>>n` – `n` sonini kiritish. `is_prime` o‘zgaruvchisi - berilgan sonning tub ekanligining rost yoki yolg‘onligini saqlovchi qiymat. Dastlab sonni tub deb tasavvur qilamiz(`is_prime = true`). 2 dan gacha sonlarni ko‘rib chiqish uchun `for (int i=2; i<=sqrt(n); i++)` ko‘rinishida siklni amalga oshirish lozim. `i<=sqrt(n)` shartning ikkala tamonini kvadratga ko‘tarib, uning o‘rniga `i*i <= n` shartni yozish mumkin.

`n` soni `i` ga qoldiqsiz bo‘linishi uchun `n` ni `i` ga bo‘lgandagi qoldiq qiymati nolga teng bo‘lishi kerak(`if (n % i==0)`). Agar bunday shart bajarilsa, u holda tekshiilayotgan son tub emas degan xulosaga kelinadi, ya’ni uning 1 dan kata va o‘ziga teng bo‘lmagan birorta bo‘luvchisi bor. Endi qolgan sonlarga bo‘linishini tekshirishning zaruriyati yo‘q, siklni to‘xtatish mumkin. Berilgan son tub emas degan xulosaga kelamiz(`is_prime = false`) va siklni to‘xtatamiz(`break`).

Agar `n=1` bo‘lsa `n` soni 2 dan boshlab hech bir songa bo‘linmaydi va `is_prime` `true` qiymatini saqlab qoladi. Buni alohida tekshirish lozim: agar `n` birga teng bo‘lsa u

holda u tub emas. Agar berilgan son tub bo‘lsa is_prime o‘zgaruvchisi **true** qiymatni saqlab qoladi.

continue operatori.

continue operatori siklni to‘xtatmasdan, uni keyingi iteratsiyadan davom qildirib ketish uchun ishlatiladi. Masalan a dan b gacha sonlar yig‘indisi va ular ichidan n ga qoldiqsiz bo‘linmaydigan sonlar sonini topish dasturini for sikli yordamida quyidagicha yozish mumkin:

```
#include
using namespace std;
int main() {
    int a, b, n;
    cin>>a>>b>>n;
    int sum = 0, cnt = 0;
    for (int i = a; i <= b; i++)
    {sum += i
    if (i % n != 0)
    cnt++; }
    cout<<a<<" dan "<<b<<" gacha sonlar yig ‘indisi: "<<sum<<endl;
    cout<<n<<" ga bo ‘linmaydigan sonlar soni: "<<cnt<<endl;}
```

a dan b gacha barcha sonlarni ko‘rib chiqamiz, sum += i summaga barcha i larni qo‘shib boramiz, agar navbatdagi son i ga qoldiqsiz bo‘linsa **if** (i % n != 0), i ga bo‘linadigan sonlar sonini birga oshiramiz(cnt++). Siklni **continue** operatori bilan quyidagi shaklda ham yozish mumkin:

```
for (int i = a; i <= b; i++)
{sum += i;
if (i % n==0)
continue;
cnt++;}
```

Bu shaklda yozilganda sum += i hamisha bajariladi. Agar n %i==0 shart bajarilsa u holda siklning navbatdagi iteratsiyasiga o‘tiladi. Ya’ni bizga n ga bo‘linmaydigan sonlar soni kerak. Agar n %i==0 shart bajarilmasa, u holda sikl tanasining navbatdagi amali ya’ni cnt++ bajarilib bo‘linmaydigan sonlar soni birga oshiriladi.

Xulosa :

C++ dasturlash tilida takrorlanish operatorlari — while, do-while va for tsikllari dastur kodining samaradorligi va takrorlanuvchi jarayonlarni avtomatlashtirishda muhim rol o‘ynaydi. Har bir tsikl o‘ziga xos jihatlarga ega bo‘lib, ular dasturchiga vazifani eng optimal tarzda hal qilish imkonini beradi.

while tsikli shart tekshirilgandan keyin bajariladi va noma’lum takrorlanishlar uchun mos keladi.

do-while tsikli kamida bir marta bajarilishini ta’minlaydi, chunki shart oxirida tekshiriladi.

for tsikli esa takrorlanishlar soni oldindan ma’lum bo‘lgan holatlarda eng qulay variant hisoblanadi.

Dasturchi sifatida ushbu tsikllarning ishlash mantig‘ini va ularni qachon qo‘llash kerakligini tushunish muhim. To‘g‘ri tanlangan tsikl kodni yanada tushunarli, samarali va optimallashtirilgan qiladi. Shu bois, takrorlanish operatorlarini puxta o‘rganish va ulardan unumli foydalanish har bir dasturchining mahoratini oshiradi.

FOYDALANILGAN ADABIYOTLAR:

- 1.Stroustrup B. The C++ Programming Language—Addison-Wesley, 2013.
- 2.Lippman S.B., Lajoie J., Moo B.E. C++ Primer—Addison-Wesley, 2012.
- 3.Meyers S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14—O'Reilly Media, 2014.
- 4.Josuttis N.M. The C++ Standard Library: A Tutorial and Reference—Addison-Wesley, 2012.
- Kerrisk M. The Linux Programming Interface—No Starch Press, 2010.
- 6.Overland B. C++ Without Fear: A Beginner's Guide That Makes You Feel Smart Addison-Wesley, 2015.
- 7.Koenig A., Moo B.E. Accelerated C++: Practical Programming by Example—Addison Wesley, 2000.
- ISO/IEC 14882:2017. Programming Languages—C++—International Organization for Standardization, 2017.

9.EckelB.ThinkinginC++–PrenticeHall,2000.

10.BalagurusamyE.Object-OrientedProgrammingwithC++–McGraw-HillEducation,2013.

11.SutterH.ExceptionalC++:47EngineeringPuzzles,ProgrammingProblems,andSolutions–Addison-Wes