## Tarmoqlanuvchi jarayonlarni dasturlash.

Tarmoqlanuvchi hisoblash jarayonlarini algoritmlash va dasturlash. Ko‘pgina masalalarni yechishda ba’zi bir jarayonlar ma’lum shart yoki shartlarning qo‘yilishiga nisbatan bajariladi. Bunday jarayonlar ***tarmoqlanuvchi jarayonlar*** deb yuritiladi va bu jarayonlarning algoritmik tavsiflari bilan avvalgi boblarda tanishgan edik.

Tarmoqlanuvchi hisoblash jarayonlari oddiy va murakkab bo‘lishi mumkin. Bu esa jarayondagi tarmoqlar soniga bog‘liq. Ma’lum bir tarmoqlanuvchi jarayon tarkibida yana tarmoqlanishlar bo‘lishi mumkin. Bunday tarmoqlanishlari bor bo‘lgan hisoblash jarayonlari ***murakkab tarmoqlanuvchi hisoblash jarayonlari*** deb ataladi.

C++ tilida tarmoqlanuvchi jarayonlarni dasturlash uchun shartsiz, shartli o‘tish va tanlash operatorlaridan foydalaniladi: IF, CASE.

### 1.1. Shartsiz o’tish operatori

Dasturda ba'zi bir hollarda boshqaruvni to’g’ridan-to’g’ri biron-bir operatorga uzatishga, ya'ni dasturning bajarilish ketma-ketligini buzishga to’g’ri keladi. Bu jarayon shartsiz o’tish operatori yordamida bajariladi. Shartsiz o’tish operatorining umumiy ko’rinishi quyidagicha:

GOTO <nishon>;

Bu yerda <nishon > identifikator bo’lib, goto yo’riqnomasidan keyin bajariladigan yo’riqning oldida joylashadi va nishon dasturning nishonlarini tafsiflash bo’limida *label* kalit so’zi yordamida tafsiflangan bo’lishi kerak, bu bo’lim dasturda o’zgaruvchilarni tafsiflash bo’limidan oldin turadi. Dastur matnida nishon, bajarilishi kerak bo’lgan operatordan oldin va birdaniga undan keyin ikki nuqta qo’yiladi.

Dasturlashtirishga bag’ishlangan adabiyotlarda ayrim hollarda goto yo’riqnomasidan foydalanmaslik tug’risidagi fikrlarni uchratish muumkin, go’yoki

goto dasturning chalkashishiga olib kelishi mumkin. Bu fikrlarga ot'liq qo'shilish to'g'ri emas. Nishonlarning ko'p bo'lishi, xaqiqatdan ham chalkashtirishi mumkin, ammo ko'pincha nishondan foydalanish qulay, hamda dasturning ixcham bo'lishini taminlaydi.

### 1.2. Shartli o'tish operatori

Dasturda boshqaruvni ma'lum shart asosida u yoki bu tarmoqqa uzatish shartli o'tish operatori yordamida amalga oshiriladi. Shartli o'tish operatori ikki xil ko'rinishda ishlatilishi mumkin: to'liq va qisqa.

C++ tilidagi shart – bu ikki rost (true) yoki yolg'on (false) qiymatlaridan birini qabul qiladigan mantiqiy turdagi (boolean) ifodadir.

Shartli operator berilgan shartni tekshirish imkoniyatini beradi va olingan natijaga qarab u yoki bu harakat bajariladi. Demak shartli operator – xisoblash jarayonini tarmoqlantiruvchi jarayondir.

Shartli o'tishda quyidagi ammaldan foydalaniladi

| Algebraik ifoda | C++ dagi operator | C++ dagi ifoda | Algebraik ma'nosi |
|---|---|---|---|
| tenglik guruhi | == | x==y | x tengdir y ga |
| teng emas | != | x!=y | X teng emas y |
| solishtirish guruhi | > | x>y | X katta y dan |
| Katta teng | >= | x>=y | X katta yoki teng y |

==, !=, >= va <= operatorlarni yozganda oraga bo'sh joy qo'yib ketish sintaksis hatodir. Yani kompilyator dasturdagi hatoni ko'rsatib beradi va uni tuzatilishini talab qiladi. Ushbu ikki belgili operatorlarning belgilarining joyini almashtirish, masalan <= ni =< qilib yozish ko'p hollarda sintaksis hatolarga olib keladi. Gohida esa != ni =! deb yozganda sintaksis hato vujudga ham, bu mantiqiy hato bo'ladi. Mantiqiy hatolarni kompilyator topa olmaydi. Lekin

ular programma ishlash mantig'ini o'zgartirib yuboradi. Bu kabi hatolarni topish esa ancha mashaqqatli ishdir (! operatori mantiqiy inkordir). Yana boshqa hatolardan biri tenglik operatori (==) va tenglashtirish, qiymat berish operatorlarini (=) bir-biri bilan almashtirib qo'yishdir. Bu ham juda ayanchli oqibatlarga olib keladi, chunki ushbu hato aksariyat hollarda mantiq hatolariga olib keladi. Yuqoridagi solishtirish operatorlarini ishlatadigan bir dasturni ko'raylik.

```
//Mantiqiy solishtirish operatorlari
# include <iostream.h>
int main()
{in
t s1, s2;
cout << "Ikki son kiriting: " << endl;
cin >> s1 >> s2; //Ikki son olindi.
if (s1 == s2) cout << s1 << " teng " << s2 << " ga" << endl;
if (s1 < s2) cout << s1 << " kichik " << s2 << " dan" << endl;
if (s1 >= s2) cout << s1 << " katta yoki teng " << s2 << " ga" << endl;
if (s1 != s2) cout << s1 << " teng emas " << s2 << " ga" << endl;
return (0);
}E
kranda:
Ikki sonni kiriting: 74 33
74 katta yoki teng 33 ga
74 teng emas 33 ga
```

Bu yerda bizga yangi bu C++ ning if (agar) struktura-sidir. if ifodasi ma'lum bir shartning to'g'ri (true) yoki noto'g'ri (false)bo'lishiga qarab, dasturning u yoki bu blokini bajarishga imkon beradi. Agar shart to'g'ri bo'lsa, if dan so'ng keluvchi amal bajariladi. Agar shart bajarilmasa, u holda if tanasidagi ifoda bajarilmay, if dan so'ng keluvchi ifodalar ijrosi davom ettiriladi. Bu strukturaning ko'rinishi quyidagichadir: if (shart) ifoda; Shart qismi qavs ichida bo'lishi majburiydir.Eng ohirida keluvchi nuqta-vergul (;) shart qismidan keyin qo'yilsa ( if (shart); ifoda; ) mantiq hatosi vujudga keladi. Chunki bunda if tanasi bo'sh qoladi. Ifoda qismi esa shartning to'g'ri-noto'g'ri bo'lishiga qaramay ijro qilaveradi. C++ da bitta ifodani qo'yish mumkin bo'lgan joyga ifodalar guruhini ham qo'yish mumkin. Bu guruhni {} qavslar ichida yozish kerak. if da bu bunday bo'ladi: if (shart) { ifoda1; ifoda2; ... ifodaN; } Agar shart to'g'ri javobni bersa, ifodalar guruhi bajariladi, aksi taqdirda blokni yopuvchi qavslardan keyingi ifodalardan dastur ijrosi davom ettiriladi

## 1.3. Tanlash operatori

*Tanlash operatori,selektor,parametrlar ro'yxati,tanlash o'zgarmaslari ro'yxati.*

Juda ko'p tarmoqlanish jarayonlarida tarmoqlanish ikki yoki undan ortiq tarmoqqa ajraladi. Umuman olganda, buni bizga tanish shartli o'tish operatori yordamida amalga oshirish mumkin:

IF B1 THEN A1 ELSE

IF B2 THEN A2 ELSE

……

……

IF BK THEN AK ;

Lekin bu hollarda shartli o'tish operatorlarining yozilishi noqulay.

Ko'p hollarda dasturchi uchun shartli operatorning umumiylashgan ko'rinishi - tanlash operatorini ishlatish qulay. Tanlash to'plami uchun 'case' komandasi ishlatiladi.

CASE operatori tarmoqlanish jarayonini berilgan bir necha operatordan birini tanlash yo'li bilan amalga oshiradi. Tanlash operatorida barcha operatorlar, shu jumladan bajarilishi uchun tanlangan operator ham aniq ravishda keltiriladi (berilgan operatorlar ketma-ketligi chegaralangan).

*Tanlash operatori* CASE mavjud variantlardan tanlash imkoniyatini beradi. U har biriga tanlash o'zgarmaslari ro'yxati (ro'yxat bitta o'zgarmasdan iborat bo'lishi mumkin) tegishli *selektor* deb nomlangan ifodadan va *parametrlar ro'yxati*dan iborat.

Formati:

CASE <ifoda-selektor> OF

<ro'yxat1>: <operator1; >

<ro'yxat k2>: <operator2; >

. . .

<ro'yxat N>: <operatorN>

ELSE <operator>

END;

O’zgarmaslar turi doim selektor turiga to’g’ri kelishi kerak. Selektor uchun real va string turlari man etilgan.

CASE operatori quyidagicha ishlaydi. Birinchi navbatda selektor–ifodaqiymati hisoblanadi, keyingi navbatda joriy selektor qiymatiga teng bo’lgan o’zgarmas qatnashgan operator bajariladi. Agar hyech qaysi o’zgarmas selektorning joriy qiymatiga teng bo’lmasa ELSE so’zidan keyingi operator bajariladi. Agar ELSE so’zi bo’lmasa END so’zidan keyingi operator ishga tushadi,ya’ni CASE chegarasidan keyingi operator.

Selektor butun sonli ( -32768..32767 diapazonida bo’lgan) bulev, liter yoki foydalanuvchi turiga bog’liq bo’lishi kerak .

O’zgarmas qiymatlar ro’yxati tasodifiy qiymat yoki diapazondan iborat, ular bir-biridan vergul yordamida ajratiladi. Diapazon chegaralari ikkita biri-biridan ".." belgisi yordamida ajratilgan o’zgaramas sonlar yordamida yoziladi. O’zgarmaslar turi selektor turiga to’g’ri kelishi kerak.

*CASE* so’zidan keyingi ifodaning qiymti hisoblanadi.

Olingan qiymat, ikki nuqtadan oldingi o’zgarmaslar ro’yxatdagi o’zgarmaslar bilan ketma- ket solishtiriladi. Bundan keyin quyidagi harakatlardan biri bajariladi:

- Agar ifoda qiymati ro’yxatdagi o’zgarmaslarning biriga mos kelsa , u holda bu ro’yxatdagi mos kelgan ketma–ket komandalar bajariladi va operator bajarilishi tugaydi;

- Agar ifoda qiymati ro’yxatdagi o’zgarmaslar biriga mos kelmasa, u holda else so’zidan keying komandalar bajariladi va operator bajarilishdan to’xtaydi;

- Agar bu operatorda else bloki bo’lmasa u holda uning bajarilishi yakunlanadi.

Bajarilishi kerak bo’lgan operator yoki operatorlar ketma-ketligi operator selektorining qiymatiga ko’ra aniqlanadi. Operator selektori sifatida haqiqiy bo’lmagan, skalyar ko’rinishdagi har qanday ifoda yoki o’zgaruvchi ishlatilishi mumkin.

Operatorning ishlashida uning tarkibidagi har bir operator tanlash belgisi deb ataluvchi belgi bilan ta'minlanadi. Bu belgi operatorning bajarilishi uchun zarur bo’lgan selektorning maxsus qiymatini qabul qiladigan selektorning tavsifiga mos konstantadir. Operator bir necha mavjud qiymatlar bilan ishlashi uchun, unda tanlash belgilari ro’yxati keltirilishi kerak.

Tanlash operatoridagi belgili operatorlar oddiy belgiga ham ega bo’lishlari mumkin. Bu holda oldin tanlash belgilari, so’ngra oddiy belgilar yoziladi.

Shuni ham inobatga olish lozimki, tanlash operatoriga faqat CASE xizmatchi so’z orqali kirish mumkin, ya'ni tanlash operatoridan tashqaridagi o’tish operatori orqali bu operatorga murojaat qilish mumkin emas.

Tanlash operatorining bajarilishi uning tarkibidagi operatorlar ketma-ketligidagi bitta operatorning bajarilishiga olib keladi. Shuning uchun ularning biridan biriga GOTO operatori yordamida o’tish xato demakdir.

Shartli o’tish operatorining quyidagi

IF B THEN Al ELSE A2

ko’rinishi tanlash operatorining quyidagi ko’rinishiga ekvivalentdir:

```
CASE B OF
TRUE: A1;
FALSE:A2;
END;
```

qisqa ko’rinishdagi shartli o’tish operatorining IF B THEN A ko’rinishi tanlash operatorining quyidagi ko’rinishga ekvivalentdir:

```
CASE B OF
TRUE: A;
FALSE
END;
```

Misol:

```
CASE T OF
'*','/': R:=l;
'+','-': R:=2
End;
```

Bu operatorning bajarilishi natijasida, agar T-belgili o’zga ruvchi "+" yoki "-" belgi qiymatlarni qabul qilsa, R o’zgaruvchi 2 qiymatni, agar T o’zgaruvchi "*" yoki "/" belgini qabul qilsa, R o’zgaruvchi 1 qiymatni qabul qiladi.