ЎЗБЕКИСТОН АЛОҚА ВА АХБОРОТЛАШТИРИШ АГЕНТЛИГИ
ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ
АХБОРОТ ТЕХНОЛОГИЯЛАРИ ФАКУЛЬТЕТИ

АТДТ кафедраси

МАЪЛУМОТЛАР ТУЗИЛМАСИ ВА АЛГОРИТМЛАР
фанидан
МАЪРУЗА МАТНЛАРИ

Тузувчи: т.ф.н., доц.Б.Б.Акбаралиев

ТОШКЕНТ - 2011

МУНДАРИЖА

1-маъруза. “Маълумотлар тузилмаси ва алгоритмлар” фанига кириш 3
2-маъруза. Маълумотларнинг оддий турлари 8
3-маъруза. Статик турдаги маълумотлар тузилмаси 14
4-маъруза. Рўйхат кўринишидаги маълумотлар тузилмаси 20
5-маъруза. Динамик маълумотлар тузилмаси 30
6-маъруза. Чизиқли боғланган рўйхатлар. ОХКТни боғламли рўйхатлар кўринишида тасвирлаш. 33
7-маъруза. Чизиқсиз маълумотлар тузилмаси 43
8-маъруза. Рекурсив маълумотлар тузилмаси 48
9-маъруза. Бинар дарахтлар ва ундаги амаллар 52
10-маъруза. Қидирув алгоритмларини тадқиқ қилиш 60
11-маъруза. Саралаш алгоритм ва усулларини тадқиқ қилиш 74
12-маъруза. Калитларни акслантириш (жойлаштириш) 82

1-маъруза. “Маълумотлар тузилмаси ва алгоритмлар” фанига кириш

Режа:

1. Асосий тушунчалар ва таърифлар
2. Маълумотларни тасвирлаш босқичлари
3. Маълумотлар тузилмасини классификация қилиш

Калитли сўзлар: *маълумотлар тузилмаси, маълумотлар турлари, абстракт босқич, мантиқий босқич, физик босқич, тўплам, кетма-кетлик, матрица, дарахт, граф, сўз.*

“Маълумотлар тузилмаси ва алгоритмлар (МТА)” фанининг мақсади - турли дастурлаш тизимларида лойиҳалаш усуллари, маълумотлар тузилмасини ишлаб чиқиш ҳамда алгоритмлар бўйича назарий ва амалий билимлар беришдан иборатдир.

МТА вазифаси – талабаларни турли хил маълумотлар тузилмалари билан таништириш, янги тузилмаларни ишлаб чиқиш хамда уларни ўқув жараёнларига тадбиқ этиш усулларини ўргатишдан иборатдир.

Маълумот - бу бирор бир объект, жараён, ходиса ёки воқеликни ифодалаб (таснифлаб) берувчи белги ёки белгилар мажмуасидир. Берилган маълумот (белги)лар қандай қиймат қабул қилишига қараб маълумотларнинг бир қанча турлари мавжуд.

Маълумотлар тузилмаси - бу тузилмани ташкил қилувчи элементлар(маълумотлар) ва улар орасидаги боғлиқликни кўрсатиб берувчи муносабатлар мажмуасидир.

Маълумотлар тузилмаси (МТ) – информацион объектнинг умумий хоссаси бўлиб, мазкур хосса билан бирор бир дастур ўзаро алоқадор бўлади. Ушбу умумий хосса қуйидагилар орқали тавсифланади:

- мазкур тузилманинг мумкин (қабул қилиши мумкин) бўлган қийматлари тўплами;
- мумкин бўлган амаллар (операциялар) мажмуаси;
- ташкил этилганлик таснифи.

Оддий маълумотлар тузилмасини баъзан маълумотлар турлари деб ҳам аталади.

Одатда, маълумотларни таснифлаш қуйидаги кўринишдаги босқичларга ажратилади:

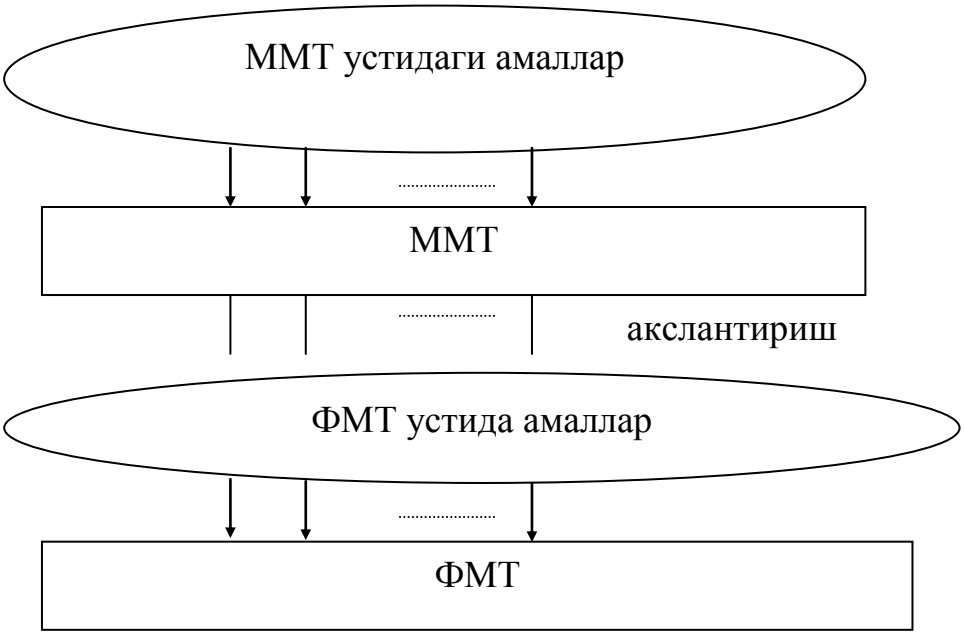
1) абстракт (математик) босқич;
2) мантиқий босқич;
3) физик (жисмоний) босқич.

Маълумки, ихтиёрий объект, ходиса ёки бирор бир жараён тадқиқ қилинаётганда унинг модели қуриб олинади. Модел турлича бўлиши мумкин, масалан, математик модел, физик модел ва бошқа моделлар. Объект, ходиса ёки бирор бир жараённи математик модел қурилди дегани ўша қаралаётган тизимни маълум бир математик қонуниятлар орқали, яъни математик формулалар орқали ифодаланишидир.

Мантиқий босқичда маълумотлар тузилмасини бирор бир дастурлаш тилида ифодаланиши тушунилади.

Физик (жисмоний) босқичда эса информацион объектни мантиқий тавсифланишига мос равишда ЭХМ хотирасида акслантирилиш тушинилади. ЭХМ хотираси чекли бўлганлиги сабабли, хотирани тақсимлаш ва уни бошқари муаммоси юзага келади.

Юқоридан кўриниб турибдики, мантиқий босқич билан физик босқичлар бир биридан фарқ қилади. Шу сабабли, ҳисоблаш тизимларида мантиқий босқични физик босқичга ва аксинча, физик босқични мантиқий босқичга акслантириш муамоси вужудга келади.



Бу ерда ММТ – мантиқий маълумотлар тузилмаси; ФМТ – физик маълумотлар тузилмаси;

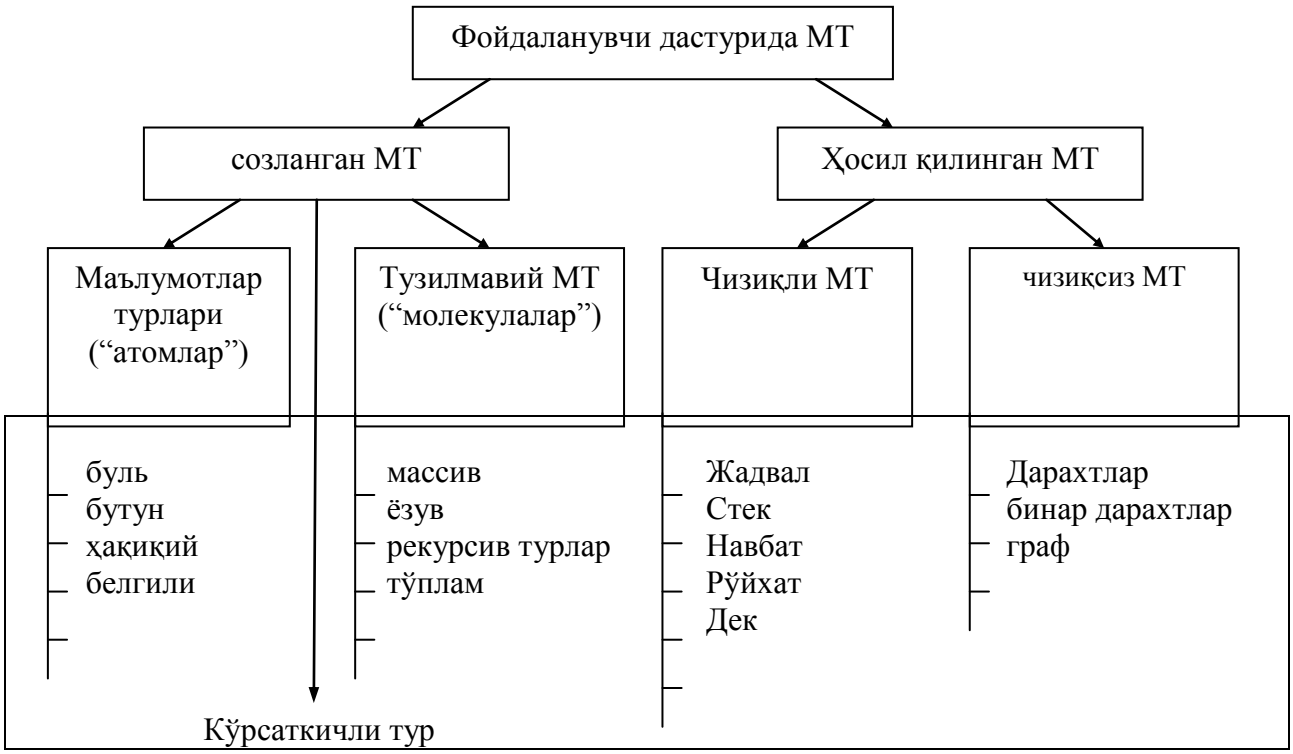
Абстракт босқичда ихтиёрий тузилмани <D,R> жуфтлик кўринишда ифодалаш мумкин, бу ерда D – элементларнинг чекли тўплами бўлиб, улар, яъни элементлар маълумотлар турлари ёки маълумотлар тузилмаси бўлиши мумкин, R – эса муносабатлар тўплами бўлиб, мазкур муносабатлар хусусиятлари абстракт босқичда маълумотлар тузилмаларини турларини аниқлайди.

Маълумотлар тузилмасини асосий кўринишлари (турлари):

1) Тўплам – муносабат тўплами бўш R=∅ бўлган элементлар мажмуаси.
2) Кетма-кетлик – шундай абстракт тузилмаки, бунда R тўплам фақатгина битта чизиқли муносабатдан иборат (яъни, биринчи ва оҳирги элементдан ташқари ҳар бир элемент учун ўзидан олдин ва кейин келадиган элемент мавжуд.

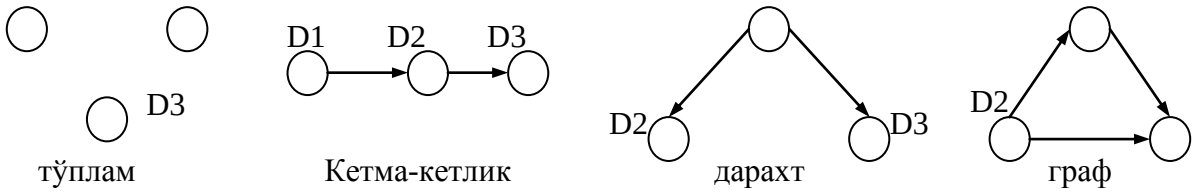
3) Матрица – шундай тузилмаки, бунда R муносабатлар тўплами иккита чизиқли муносабатдан ташкил топган бўлади.
4) Дарахт – бунда R тўплам иерархик тартибдаги битта муносабатдан ташкил топган бўлади.
5) Граф – бунда R муносабатлар тўплами фақатгина битта бинар тартибли муносабатдан ташкил топган бўлади.
6) Гиперграф – бу шундай маълумотлар тузилмасики, бунда R тўплам икки ёки ундан ортиқ турли тартибдаги муносабатлардан ташкил топган бўлади.

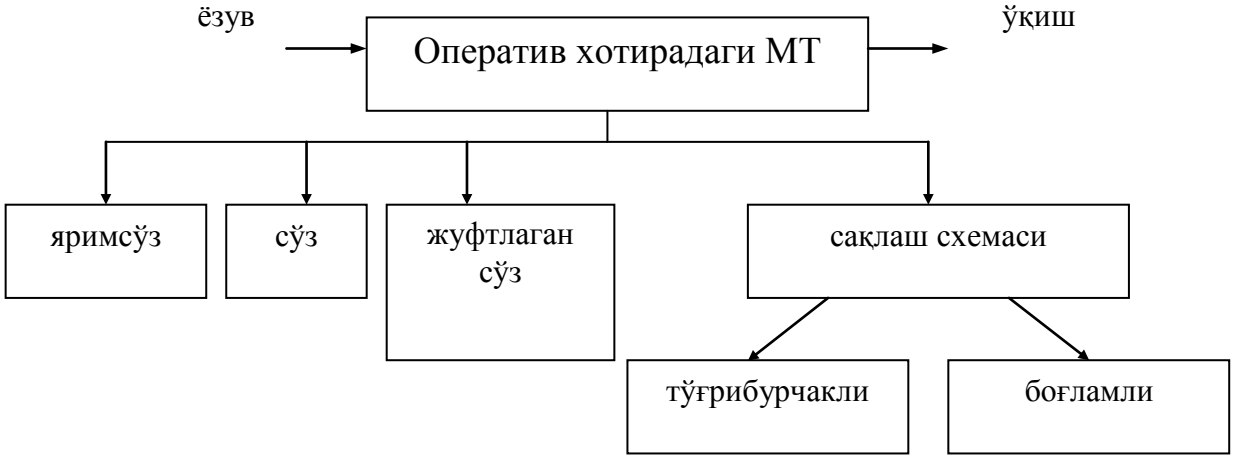
Фойдаланувчи дастурида ва ЭҲМ ҳотирасида МТ классификация қилиш



МТ классификация қилишда асосий белги бу маълумотлар тузилмасини дастур ишлаши мобайнида ўзгариши ҳисобланади. Масалан, агар дастур бажарилиши мобайнида элементлар сони ва/ёки улар орасидаги муносабатлар ўзгарса, у ҳолда бундай МТ динамик маълумотлар тузилмаси, акс ҳолда статистик маълумотлар тузилмаси дейилади.

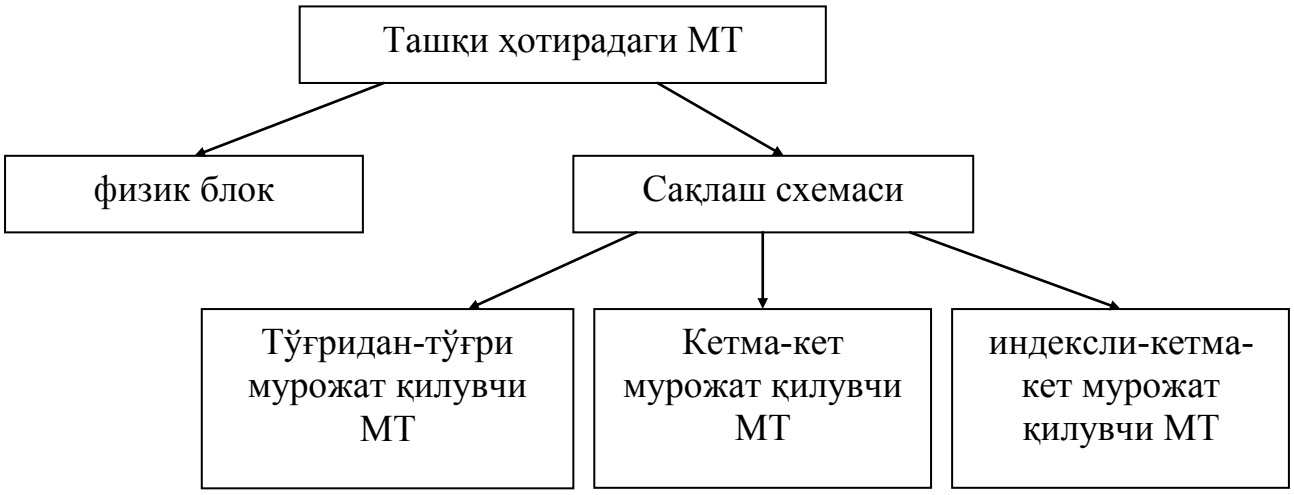
МТга мисоллар:





Оператив хотира массивни ифодалайди.

Сўз – бир вақтнинг ўзида қайта ишланиши мумкин бўлган минимал сондаги битдир.



Назорат саволлари

1. Маълумотлар тузилмаси деганда нимани тушунасиз?
2. Маълумотларни тасвирлаш босқичларини келтириб ўтинг.
3. Маълуотлар тузилмаси классификацияси қандай амалга оширилади?
4. Маълумотлар тузилмасини фойдаланувчи дастуридаги классификацияси.

5. Маълумотлар тузилмасини оператив хотирадаги классификацияси.
6. Маълумотлар тузилмасини ташқи хотирадаги классификацияси.
7. Қандай маълумотлар динамик ёки статик турдаги маълумотлар тузилмаси дейилади?

2-маъруза. Маълумотларнинг оддий турлари

Режа:
1. Маълумотларни стандарт турлари;
2. Фойдаланувчи томонидан аниқланадиган турлар.

Калитли сўзлар: *бутун тур, ҳақиқий тур, мантиқий тур, белгили тур, кўрсаткичли тур, саналадиган тур, диапазонли тур*

Маълумки, математикада ўзгарувчиларни, уларнинг баъзи бир керакли тавсифларига мос равишда классификация қилиш қабул қилинган. Ўзгаувчиларга мисол сифатида қуйидагиларни келтириб ўтиш мумкин: ҳақиқий ўзгарувчилар, комплекс ўзгарувчилар, мантиқий ўзгарувчилар, бундан ташқари баъзи бир қийматларни қабул қилувчи ўзгарувчилар ва бошқалар. Маълумотларни қайта ишлашда уларни классификация қилиш ҳам катта аҳамиятга эга. Бу ерда ҳам классификация қилинаётганда ҳар бир константа, ўзгарувчи, ифода ёки функция бирор бир турга тегишли бўлади деган тамойилга асосланади.

Умуман олганда турлар ўзгарувчи ёки ифода қабул қилиши мумкин бўлган қийматлар тўплами орқали тавсифланиб, улар функцияларни шакллантириши мумкин.

Кўплаб дастурлаш тилларида маълумотлар стандарт ва фойдаланувчи томонидан бериладиган турларга ажратилади. Маълумотларни стандарт турларига қуйидаги 5 та тур ўзгарувчилари киради:

a) бутун;
b) ҳақиқий;
c) мантиқий;
d) белгили (символ) (CHAR);
e) кўрсаткичли (POINTER).

Фойдаланувчи томонидан аниқланадиган турлар эса 2 та:

a) саналадиган;
b) диапазонли (оралиқли).

Маълумотларнинг ихтиёрий тури қийматлар соҳаси ва улар устида бажарилиши мумкин бўлган амаллар орқали тавсифланади.

БУТУН ТУР

Мазкур тур бутун сонлар тўпламини қандайдир қисм тўплами бўлиб, унинг ўлчами машина, яъни ЭҲМ конфигурациясига боғлиқ равишда ўзгариб туради. Агар бутун сонни машинада тасвирлаш учун п та разряддан фойдаланилса (бунда қўшимча коддан фойдаланилганда), у ҳолда х бутун соннинг қиймат қабул қилиш оралиғи қуйидагича бўлиши зарур, яъни қуйидаги шартни қаноатлантириши лозим: $-2^{n-1} <= x < 2^{n-1}$.

Бутун турдаги маълумотлар устида бажариладиган барча амаллар тўғри амалга оширилади деб ҳисобланиб, ушбу амаллар арифметикада қабул қилган қоидаларига бўй сунади. Агар ушбу турда амаллар бажарилганда

натижа рухсат этилган ораликдан чиқиб кетса, у ҳолда ҳисоблаш тўхтатилади. Бундай ҳол тўлиб кетиш деб аталади.

Мазкур турга кирувчи сонлар иккитага бўлинади: ишорали ва ишорасиз. Уларнинг ҳар бир учун мос равишда қиймат қабул қилиш оралиғи мавжуд:

a) ишорасиз сонлар учун $(0..2^{n}-1)$;

b) ишоралилар учун $(-2^{N-1}.. 2^{N-1}-1)$.

Сонлар машинада қайта ишланаётганда уларнинг ишорали кўринишидан фойдаланилади. Агар машина сўзи ёзув, командарани қайта ишлаш ва кўрсаткичлар учун фойдаланилаётган бўлса, у ҳолда соннинг ишорасиз кўринишидан фойдаланилади.

Бутун сонлар устида – қўшиш, айриш, кўпайтириш, бутунсонли бўлиш (қолдиқни ташлаб юбориш орқали), берилган модул бўйича ҳисоблаш (бўлишда қолган қолдиқни ҳисоблаш), берилган сонлар тўпламининг энг катта ва энг кичик элементини аниқлаш, бутун даражага ошириш, соннинг қийматига қараб ўзидан олдинги ёки кейинги сонни аниқлаш. Бу операцияларнинг натижалари ҳам бутун сонлар бўлади.

Мисол (Паскал тилида):

A div B = C
A mod B = D
C * B + D = A
7 div 3 = 2
7 mod 3 = 1

Бутун сонлар устида =,~=, <, <=, >, >= операторлар билан бинар амалларни ҳам бажариш мумкин. Аммо бу операцияларнинг натижалари бутун турга кирмайди, улар мантиқий турга киради.

Ҳақиқий тур

Ҳақиқий турга каср қисмлари бор чекли сонлар тўплами киради. Тўпламни чекли бўлиш шарти ЭХМда сонларни ифодалаш чегараланганлиги билан боғлиқ. Ҳақиқий сонлар устида қуйидаги амалларни бажариш мумкин: қўшиш, айриш, бўлиш, кўпайтириш, тригонометрик функцияларини хисоблаш, даражага ошириш, квадрат илдиз чиқариш, логарифмлаш, минимум ва максимум элементларни топиш ва бошқалар. Буларнинг натижалари ҳам ҳақиқий турга киради. Бу ерда ҳам бинар амалларга нисбатан масаланинг ечимлари мантиқий турга тегишли бўлади.

ЭҲМ хотирасида ҳақиқий сонлар асосан қўзғалувчан нуқта форматида сақланади. Бу форматда х ҳақиқий сон қуйидаги кўринишда ифодаланади:

$\mathbf{x = +/- M * q^{(+/-P)}}$ – сонинг яримлогарифмик шаклдаги ифодаланиши қуйидаги чизмада келтирилган.

$$\mathbf{937,56 = 93756 * 10^{-2} = 0,93756 * 10^{3}}$$

Мантиқий тур.

Мазкур тур мантиқий мулоҳазаларни тўғрилигини аниқлаш учун, турли ҳил дастурлаш тилларида турлича ифодаланиладиган ифодаларни 2 та кўринишда аниқлайди. Мантиқий маълумотлар устида қуйидаги мантиқий операцияларни бажариш мумкин: конъюнкция (ва), дизъюнкция (ёки) и инкор (йўқ), ҳамда қийинроқ бўлган эквивалентлик, импликация, чиқариб ташлаш, ёки ва бошқа операциялар. Юқорида келтирилган ихтиёрий операциянинг натижаси – мантиқий қийматга эга бўлади. Мантиқий қийматни хотирада сақлаш учун битта бит етарли.

Асосий мантиқий функцияларнинг чинлик жадвали.

A	B	not A	A or B	A and B
1	1	0	1	1
1	0	0	1	0
0	1	1	1	0
0	0	1	0	0

Белгили тур.

Белгили турга белгиларнинг чекли тўплами ёки литер, уларга лотин алифбосидаги харфлар ва унда йўқ кирилл харфлар, ўнлик рақамлар, математик ва махсус белгилар киради. Белгили маълумотлар ҳисоблаш техникаси билан инсон ўртасидаги алоқани ўрнатишда катта аҳамиятга эга. Кўпинча, дастурлашнинг ҳар бир тизимида белгилар тўплами фиксирланган бўлиб, улар турли тизимларда турли ҳил бўлиши мумкин. Бундан ташқари улар тартибланган бўлиб, ҳар бир унинг элементига аниқ бир сонли код мос қўйилиб, у тўпламдаги тартиб рақамини аниқлайди. Белгини сонли кодига ўтиб, реляцион операторлардан фойдаланиб, символларни таққослаш мумкин. Бундай таққослашларнинг натижалари BOOLEAN турига киради. Белгини кодини қийматини аниқлаш учун ёки аксинча, кўпгина дастурлаш тилларида 2 та стандарт функциядан фойдаланилади. Масалан Паскал тилида символлар қўштирноқ орасида берилиши мумкин ёки коди орқали ифодаланаётган бўлса # белги қўйилиб кейин сонли коди ёзилади. Сатр (қатор, string) – бу қандайдир белгилар кетма-кетлиги. Сатр битта, бўш ёки бир нечта белгилар бирлашмасидан иборат бўлиши мумкин. Паскал тилида сатр 0 дан то 255 тагача узунликка эга бўлиши мумкин. Агар ўзгарувчи сатр турига тегишли бўлса, у ҳолда ўзгарувчи тури ёзилаётганда string деб аниқланади.

Белгили турдаги амаллар:

a) Ўзлаштириш;
b) Таққослаш;
c) Кодлаш тизимидаги рақамини (тартибини) аниқлаш: ORD(W_i);
d) Белгини рақами бўйича топиш: CHR(i)
e) Навбатдаги белгини аниқлаш: SUCC(W_i)
f) Битта олдинги белгини аниқлаш: PRED(W_i)

Кўрсаткичли тур.

Pointer тури маълумотларни кўрсаткичлари ёки манзиллари (адрес) тўпламини намоён қилади, яъни кўрсаткичлар маълумотларни эмас балки бу маълумотлар жойлашган хотирадаги манзилни ўз ичига олади. Кўрсаткичлар хотирада бори йўғи 4 байт жойни эгаллаб, у кўрсатаётган маълумотлар анча катта жойни эгаллаган бўлиши мумкин. Pointer тури маълумоти ихтиёрий бошқа бирор маълумот ёки маълумотлар гуруҳига йўналтирилган бўлади. Кўрсаткичга мумкин бўлган у ёки бу қийматни ўзлаштириб, ушбу кўрсаткич орқали керакли маълумотга мурожатни амалга ошириш мумкин. Pointer туридаги маълумотларни қийматлар тўпламида битта махсус қиймат бўлиб, уни ўзлаштириш ҳеч қаерга йўналтирилмаганлигини кўрсатади, яъни нол ёки бўш кўрсаткич хисобланади. Масалан, Паскаль тилида бундай қиймат сифатида nil символи фойдаланилади. Кўрсаткичлар устида амаллар қуйидагича бўлиши мумкин: бирор бир кўрсаткичга бошқа кўрсаткич қийматини ўзлаштириш мумкин ёки бошқа маълумот эгаллаб турган хотира соҳаси адресини ўзлаштириш мумкин. Кўрсаткичлар ўзаро боғланган маълумотлар тузилмасини яратишда ва қайта ишлашда катта аҳамиятга эга. Хотирада кўрсаткичларни ифодалаш учун учун асосан дастурлаш тизимига мос равишда манзилни максимал узунлигича жой ажратилади. Кўрсаткичларни қиймати номанфий бутун сонлар сифатида соҳада битларни кетма-кетлиги кўринишида сақланади.

Фойдаланувчи томонидан аниқланадиган турлар

Саналадиган

Қийматларнинг ўзгарувчан турлари стандартлардан фарқлирок янги турларни яратишга имкон беради. Бу гуруҳга саналадиган ва чегараланган турлар киради.

Қийматларнинг саналадиган туринг бундай аталишига сабаб, улар қатъий аниқланган тартибда саналадиган кўринишда берилади ва хамма қийматларнинг сони қатъий чегараланган хамда кўрилаётган турдаги қийматларни қабул қилиши мумкин. Саналадиган тур ечилаётган масалага қараб фойдаланувчи томонидан берилиши мумкин.

Саналадиган тур константалар рўйхатидан ташкил топади. Бу турдаги ўзгарувчилар рўйхатидаги ихтиёрий қийматни қабул қилиши мумкин. Саналадиган турнинг умумий ёзилиш шакли қуйидагича:

TYPE_ турнинг номи = (константалар рўйхати);
VAR_ ўзгарувчи номи: турнинг номи;

Бу ерда константа тушунчаси фойдаланувчи томонидан берилаган махсус константа кўриниши тушунилади. Константалар рўйхати бир-биридан вергул билан ажратилади ва улар оддий қавслар ичига олинади. Масалан:

TYPE ЙИЛ =(қиш, баҳор, ёз, куз);
VAR A: ЙИЛ;

бу ерда ЙИЛ -саналадиган турнинг номи; қиш, ёз, куз-константалар.

А - ўзгарувчи номи бўлиб у юқоридаги константалардан ихтиёрийсини қабул қилиши мумкин.

Паскал тилида саналадиган турнинг константалари тўғридан-тўғри ўзгарувчилар бўлимида TYPE бўлимисиз хам берилиши мумкин, яъни VAR A: (киш, бахор, ёз, куз) ;

Ҳар бир константа тартиб рақамига эга бўлиб, хисоб **0** дан бошланади, яъни қиш-**0,** баҳор-1, ёз-2, куз-3 рақамларига эга. Константалар тартиблангани учун уларга солиштириш амаллари <, <=, =,<>, >=, > ва ORD, PRED, SUCC шунингдек стандарт функцияларни қўллаш мумкин.

Мисол.

Бизга хайвонларнинг номи берилган. Бу рўйхатдаги йўлбарснинг тартиб номери ва тулкидан кейин турган хайвоннинг тартиб номери аниклансин.

```
PROGRAM_АНИКЛАШ (INPUT, OUT);
 TYPE_хайвон=(тулки, бўри, қуён, зебра, йўлбарс, арслон, айиқ, бегемот, бургут, барс)
  VAR P1, P2 : хайвон
  N1, N2 : INTEGER;
BEGIN
  P1: = йўлбарс ;
  P2: = SUCC (тулки) ;
  N1: = ORD (P1) +1 ;
  N2: = ORD (P2) +1;
  WRITELN (`ТАРТИБ номери йулбарс =` , N1: 2) ;
  WRITELN (` тулкидан кейин турган хайвоннинг тартиб номери = ` , N2:2)
END.
```

Ҳайвонларнинг тартибланиш рақами нолдан бошлангани сабабли, уларнинг ҳақиқий тартиб рақамини аниқлаш учун N1 ва N2 га 1 қўшиш керак бўлади.

Чегараланган ёки диапазонли турлар.

Агарда бирорта ўзгарувчи ўзининг туридаги қийматларнинг хаммасини қабул қилмасдан, фақат бирор чегараланган оралик қийматларини қабул қилса, у ҳолда бу ўзгарувчини чегараланган тур деб қараш мумкин бўлади.

Масалан, турларни тасвирлаш қисмида йилнинг ойлари тасвирланган бўлса, яъни TYPE ОЙ = (январь, февраль, март, апрель, май, июнь, июль, август, сентябрь, октябрь, ноябрь, декабрь); аммо масалани ечишда фақат йилнинг ёз ойи иштирок этса, у ҳолда чегараланган турни қуйидагича ёзиш мумкин

TYPE ЁЗ = ИЮНЬ..АВГУСТ ;

ва у йил ойларининг бир қисмини ташкил қилади.

Бу ҳолда Ёз тури ОЙ туридан «кесиб» олинган бўлиб, Ёз турига нисбатан ОЙ база ҳисобланади.

Мисолдан кўриниб турибдики, чегараланган турларда константаларнинг бошланғич ва охирги қийматлари кўрилаётган диапазон учун берилади ва бир-биридан (..) горизонтал икки нуқта билан ажратилади .

Чегараланган турнинг умумий кўриниши қуйидагича.

TYPE тур номининг константа;

Бундай тур қуйидаги қоидаларга амал килиш керак :

1) Икала чегара константалари бир хил турли бўлиши керак;
2) База тури сифатида оддий турларнинг ихтиёрийсидан фойдаланиш мумкин, фақат хақиқий тур (REAL) бўлмасин.
3) Чегараланган турда бошланғич константа охириги константадан катта бўлмаслиги керак .
4) Чегараланган турдаги ўзгарувчилар, ўзгарувчиларни тасвирлаш қисмида уларнинг номи билан тасвирланиши керак.

Назорат саволлари

1. Маълумотлар тузилмасини асосий тавсифи нимадан иборат?
2. Маълумотларнинг қандай турларини биласиз?
3. Бутун турдаги маълумотлар устида қандай амалларни бажариш мумкин?
4. Маълумотларнинг бул турида қандай амаллар мавжуд?
5. CHAR туринг тузилмаси қандай?
6. Белгили турдан қандай амалларни бажариш мумкин?
7. Кўрсаткичли тур маълумоти ёрдамида нимани ҳисоблаш мумкин?
8. Маълумотларнинг саналадиган тури дегани нима?
9. Диапазонли тур қандай берилади?

3-маъруза. Статик турдаги маълумотлар тузилмаси

Режа:

1. Тўплам тушунчаси ва унинг устидаги амаллар.
2. Массивлар ва улар устидаги амаллар.
3. Ёзувлар ва улар устидаги амаллар.
4. Жадваллар ва уларни эълон қилиш.

Калитли сўзлар: *статик тур, вектор, массив, ёзув, жадвал, калит, қўшма калит.*

1. Тўплам тушунчаси.

Маълумки, тўплам тушунчаси билан қайсидир маънода мактаб даврларидан танишмиз. Тўплам математикада бошланғич, фундаментал тушунчалардан бири бўлиб, элементлар мажмуасидан ташкил топади.

Таъриф. Тўплам бу бир турга тегишли бўлиб, такрорланмайдиган элементлар мажмуасидир.

Тўплам базавий турга тегишли бўлган барча қийматларни қабул қилиши мумкин. Шуни эслатиб ўтиш лозимки, базавий 256 тадан ортиқ қийматни қабул қилмаслиги лозим. Шу сабабли тўпламнинг базавий тури byte, char ва улар орқали ҳосил қилинган турлар бўлиши мумкин**.**

Тўплам тури маълумотлари учун ажратилган байтлар сони:

ByteSize = (max div 8)-(min div 8) + 1, бу ерда max ва min – тўплам базавий турининг юқори ва қуйи чегараси.

Аниқ бир E элементнинг байт рақами қуйидагича аниқланади:

ByteNumber = (E div 8)-(min div 8).

Тўпламни Паскал тилида эълон қилиш:

var S : set of T_0;

Тўплам устидаги амаллар:

Қўшиш (тўпламлар йиғиндиси, A+B);

Айриш (тўпламлар айримаси, A-B);

Кўпайтмаси(A*B);

Элементни тўпламга тегишли ёки тегишли эмаслигини аниқлаш (a in A).

2. Векторлар

Вектор – бу энг содда статик ва чизиқли тартибланган тузилмадир. Бу тузилмадаги элементлар орасидаги муносабат уларнинг қатъий кетма-кетлик кўринишида ифодаланишидир (қаранг, чизма).

N	N+1	N+2	…	N+M

Векторнинг ҳар бир элементи унинг вектордаги ўрнини аниқловчи (кўрсатувчи) ўзининг индексига эга бўлади. Индекслар бутун сон бўлганлиги туфайли улар устида бир неча амалларни амалга ошириш мумкин ҳамда мурожаатни мантиқий босқичида элементни тузилмадаги ўрнини аниқлаш

мумкин. Векторнинг элементига мурожаат қилиш учун векторни номини ва элементнинг индексини кўрсатиш кифоя бўлади.

Дастурда векторни эълон қилиш учун унинг номини, элементлар сонини ва уларнинг турини кўрсатиш лозим

Масалан:

```
var
  M1: Array [1..100] of integer;
  M2: Array [1..10] of real;
```

Вектор элементларнинг барчаси фақатгина битта турга тегишли маълумотлардан иборат ҳамда уларнинг сони олдиндан аниқ бўлиши лозим.

3. Массивлар

Умуман олганда массив элементи бу вектор элементи бўлиб, унинг элементи ҳам тузилма элементи бўлиб ҳисобланади(қаранг. чизма).

B_{11}	B_{12}	...	B_{1N}
B_{21}	B_{22}	...	B_{2N}
B_{31}	B_{32}	...	B_{3N}
B_{41}	B_{42}	...	B_{4N}

Икки ўлчамли массив элементига мурожаатни амалга ошириш учун унинг индекси қийматларии зарур бўлади. Физик босқичда икки ўлчамли массив ҳам худди бир ўлчамли (вектор) массив каби кўринишга эга бўлади ҳамда трансляторлар массивни қатор ёки устун кўринишида ифодалайди.

Массив хоссалари

- Массив элементлари бир турга тегишли, шу сабабли, уларнинг ҳар бири хотирада бир ҳил хажмни эгаллайди;
- Массив ташқи қурилмада эмас, балки оператив хотирада жойлашади;
- Массив элементлари кетма-кет келган ячейкаларни эгаллайди.

C++да массивни 2 ҳил усулда бериш мумкин:

- инициализация қилинмаган (масалан, 4 та элементдан иборат бутун турли массив: int a[4]);
- инициализация қилинган (int a[]={2, 3, 4, 5}).

Эслатма: Массивлар билан ишлаётганда эълон қилинган чегарадан чиқиб кетмаслик лозим, сабаби бу ҳақида компилятор огоҳлантирмайди.

Массивга доир мисоллар

Фараз қилайлик, бизга A={aij} ва B={bij} матрицалар берилган бўлиб, A матрица энг катта элементини ва ушбу матрицалар йиғиндисини топиш талаб қилинган бўлсин.

```
# include <iostream.h>
# include <conio.h>
main()
{clrscr(); int n,m,i,j,p,t;
```

```
float  a[40][50], b[50][60], c[40][50];
cout<<"A matrisa o'lchamini kirit"<<endl;
cin>>n>>m;
cout<<"B matrisa o'lchamini kirit"<<endl;
cin>>p>>t;
cout<<"A matrisa elementlarni kirit"<<endl;
for(i=0; i<n; i++)
for(j=0; j<m;j++)
cin>>a[i][j];
cout<<"B matrisa elementlarni kirit"<<endl;
for(i=0; i<p; i++)
for(j=0; j<t;j++)
cin>>b[i][j];
float x=a[0][0]; int k=0;int l=0;
for(i=0; i<n; i++)
for(j=0; j<m;j++)
if (x<a[i][j]){ x=a[i][j];
k=i+1;l=j+1;}
cout <<"A matrisa max="<<x<<" "<<"k="<<k<<"l="<<l<<endl<<endl;
if((n!=p) && (m!=t)) cout<<"A va B matrisalarni qo'shib bo'lmaydi"<<endl;
else
{ for(i=0; i<n; i++)
for(j=0; j<m;j++)
c[i][j]=a[i][j]+b[i][j];
}
return 0;
}
```

4. Ёзув

Ёзув – майдон деб аталувчи чекли сондаги маълумотлар тузилмасидир. Ёзув кетма-кет турдаги маълумотлар тузилмасини ифодалаб, мантиқий тасвирланишда ҳам физик тасвирланишда ҳам тузилма элементлари кетма-кет жойлашган бўлади. Ёзувнинг массивдан фарқи шундан иборатки, унинг элементлари турли турларга тегишли бўлган тўпламдан иборат бўлиши мумкин. Ёзувда маълумот элементларини кўпинча ёзув майдонлари деб хам аталади.

Ёзувни эълон килиш мантиқий кўриниши қуйидагича:

<тур номи> = record <майдонлар рўйхати> end

майдонлар орасига ; белгиси қўйилади.

Масалан:

```
Type
    BirthDay = Record
    Day, Month:Byte;
    Year:Word;
End;
```

```
Var
    a,b:BirthDay;
    Begin
        a.day:=27;
        b.year:=1939;
    End.
```

Майдонларга мурожаатни соддалаштириш учун with операторидан фойдаланилади.

Масалан:

```
With a Do
Begin
    Day:=27;
    Month:=4;
    Year:=1985;
    End.
```

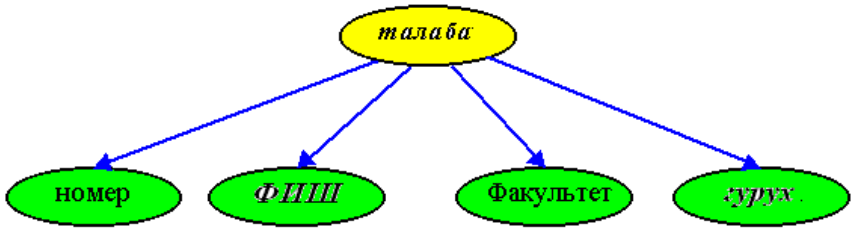
Қуйидаги масалани таҳлил қилиб чиқайлик:

Ёзувнинг мантиқий тузилмасини график кўринишда хам жадвал кўринишида хам ифодалаш мумкин, яъни

Мантиқий тузилма

Номер	Фамилия	Факультет	Гуруҳ

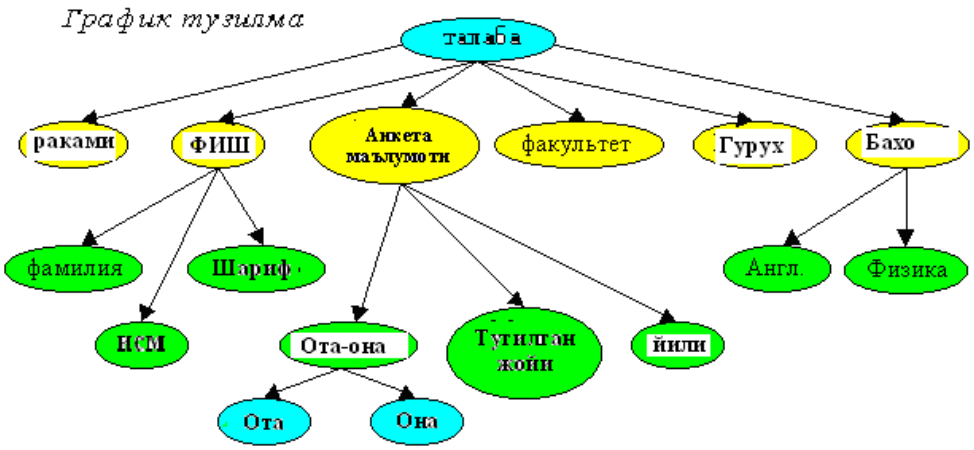
График тузилма



Ёзув элементларини ўзи хам ёзувдан иборат бўлиши мумкин. Бу ҳолатда мураккаб иерархик маълумотлар тузилмаси вужудга келади.

Талаба ҳақида қуйидаги маълумотларни ўз ичига олувчи ёзувни тўлдириш талаб қилинган бўлсин: N – талаба тартиб рақами; талаба Исми, бу ерда талаб фамилияси, исми, отасининг исми бўлсин; талабанинг анкета маълумотлари, яъни туғилган йили, туғилган жойи, ота-онаси: онаси, отаси; Факультети; Гуруҳи; фанлардан сессияда олган баҳолари, масалан чет тили, информатика, математика ва бошқ.

Қуйида ушбу ёзув тузилмасини иккита мантиқий ифодаланиши келтириб ўтилган.



Юқоридаги маълумотни ифодалаш натижасида тўрт босқичли иерархик маълумотлар тузилмасига эга бўламиз. Информация тармоқларда жойлашган бўлиб, қолган тугунлари тармоқларга йўлни кўрсатади.

1-чи босқич	Талаба = ёзув
2-чи босқич	Рақам
2- чи босқич	Исм = ёзув
3- чи босқич	Фамилия
3- чи босқич	Исм
3- чи босқич	Отасиниг исми
2- чи босқич	Анкет маълумотлари = ёзув
3- чи босқич	Туғилган жойи
3- чи босқич	Туғилган йили
3- чи босқич	Ота-онаси = ёзув
4- чи босқич	Онаси
4- чи босқич	Отаси
2- чи босқич	Факультет
2- чи босқич	Гуруҳ
2- чи босқич	Баҳолар =ёзув
3- чи босқич	Чет тили
3- чи босқич	Физика

Ушбу тузилма ичма-ич жойлашган ёзув деб аталади.

Ёзув устидаги амаллар:

1. Ёзув майдони маълумотларни ўқиш.
2. Ёзув майдонига информация киритиш.
3. Турга мос келувчи, ёзув майдони устида бажариши мумкин бўлган барча амаллар.

5. Жадваллар

Жадвал - бу ёзувнинг чекли мажмуасидир.

N	ФИШ	ГУРУХ	ФАКУЛЬТЕТ	АНГЛ.	ФИЗИКА
1					
2					
…					
n					

Жадвал берилаётганда унда иштирок этадиган ёзувлар сони кўрсатиб ўтилади.

Масалан:

```
Type ST = Record
            Num: Integer;
            Name: String[15];
            Fak: String[5];
            Group: String[10];
            Angl: Integer;
            Physic: Integer;
var
  Table: Array [1..19] of St;
```

Жадвал маълумотлари элементи ёзув ҳисобланади. Шунинг учун жадвал устида бажариладиган амаллар бу ёзув устида бажариладиган амаллардир.

Жадвал устида бажариладиган амаллар:

1. Берилган калит бўйича ёзувни қидириш.
2. Жадвалга янги ёзувни киритиш.

Калит – бу ёзув идентификатори. Ушбу идентификаторни сақлаш учун махсус майдон ажратилади.

Қўшма калит – бу шундай калитки, у иккидан ортиқ майдонни ўз ичига олади.

Назорат саволлари

1. Қайси статик тузилма энг оддий хисобланади?
2. Вектор деб нимага айтилади?
3. Ёзув деганда нимани тушунасиз?
4. Ёзувни эълон қилиш қандай амалга оширилади
5. Жадвални асосий элементларини санаб беринг.
6. Уларнинг асосий хусусиятларини айтиб беринг.
7. Статик турдаги маълумотлар тузилмаси устида бажарилиши мумкин бўлган амаллар.

4-маъруза. Рўйхат кўринишидаги маълумотлар тузилмаси

Режа:

1. Рўйхатлар ҳақида тушунча.
2. Оммавий хизмат кўрсатиш турлари ҳақида тушунча.
3. ОХКТдаги асосий амаллар.
4. ОХКТни массив кўринишда ифодалаш.

Калитли сўзлар: *яримстатик тузилма, рўйхат, навбат, стек, халқасимон навбат, дек.*

Рўйхатлар

Таъриф 1. Рўйхат деб бир турга тегишли бўлган элементлар кетма-кетлигига айтилади.

Рўйхатга мисол:

$E_1, E_2, ..., E_n$ $(n\geq 0)$, $E_i \in T$, $i=1\ldots n$, бу ерда T маълум бир тур.

Таъриф 2. Рўйхат элементлари сони **n** га рўйхат узунлиги дейилади.
Изоҳ. Агар $n\geq 1$ бўлса, у ҳолда E_1 рўйхат биринчи элементи, E_n эса охирги элементи ҳисобланади.
Таъриф 3. Агар n=0 бўлса, у ҳолда рўйхат бўш дейилади.
Эслатма: Умумий ҳолда рўйхат элементлари сони чегараланмаган ва дастур бажарилиши мобайнида ўзгариб туриши мумкин мумкин.

Рўйхат элементларини уларни рўйхатдаги позициясига қараб тартиблаш мумкин.

Рўйхат туридаги маълумотлар тузилмасини мантиқий тасвирлашни икки ҳил кўринишда амалга ошириш мумкин:
1). Ошкормас (ноаниқ) кўринишда боғланган рўйхат. Бунда рўйхат массив асосида амалга оширилади.
2). Ошкор кўринишда берилган рўйхат. Бунда рўйхат элементлари кўрсаткичлар орқали боғланган.

Рўйхат кўринишидаги маълумотлар тузилмасига:

- Стек;
- Дек;
- Навбат.

Кундалик хаётда деярли ҳар куни ҳар бир инсон навбат тушунчаси билан дуч келади. Умуман олганда рўйхат элементи бирор бир хизмат кўрсатишга буюртма бўлиб хисобланади: масалан, маълумотлар бюросидан керакли маълумотни олиш, кинотеатрларда чипта олиш, дўконда харид қилиб олинган маҳсулотларга кассада пул тўлаш ва бошқ.

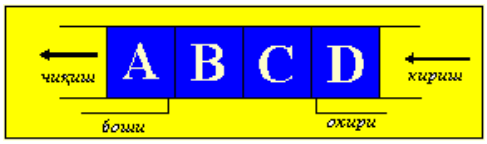
Умумий олганда, юқорида келтириб ўтилган тузилмалар, яъни стек, навбат ва дек оммавий хизмат кўрсатиш турларини ифодалаб, улар тузилма элементларига хизмат кўрсатилиш тартибини аниқлаб беради.

Дастурлашда шундай маълумотлар тузилмаси мавжудки, у навбат деб аталади. Бу турдаги маълумотлар тузилмасида келиб тушган буюртмаларга хизмат кўрсатиш тартиби аниқланади.

Навбатлар яримстатик тузилма хисобланиб, вақт ўтиши ва навбат узунлигига қараб, уни ташкил этувчи элементлар ўзгариб туриши мумкин.

Навбатни ташкил қилувчи элементларга хизмат кўрсатилишига қараб, навбатнинг асосий иккита кўриниши мавжуд:

1. Навбатнинг биринчи кўринишида, навбатга келиб тушган биринчи элементга биринчи бўлиб хизмат кўрсатилади ва навбатдан чиқарилади. Мазкур кўринишдаги хизмат кўрсатишни **FIFO** (*First input-First output*, яъни биринчи келган – биринчи кетади) номлаш қабул қилинган. Навбат ҳар иккала томондан очиқ бўлади.



2. Иккинчи кўринишни **LIFO** (*Last input - First output*, яъни охирги келган – биринчи кетади) дейилиб, навбатга келиб тушган охирги буюртма (элемент)га биринчи бўлиб хизмат кўрсатилади. Мазкур кўринишдаги навбатни дастурлашда СТЕК деб номлаш қабул қилинган.

1. Стеклар

LIFO, яъни навбатнинг охирги бўлиб кирган элементига биринчи бўлиб хизмат кўрсатилади. Бу энг кўп ишлатиладиган маълумотлар тузилмаларидан бири бўлиб, турли хил масалаларни ҳал қилишда анча қулай ва самарали хисобланади.

Хизмат кўрсатишни келтирилган тартибига кўра, стекда фақатгина битта позицияга мурожаат қилиш мумкин. Бу позиция стекнинг учи дейилиб унда стекка вақт бўйича энг охирги келиб тушган элемент назарда тутилади. Биз стекга янги элемент киритсак, бу элемент олдинги стек учида турган элемент устига жойлаштирилади хамда стекни учида жойлашиб қолади. Элементни фақатгина стек учидан танлаш мумкин; бунда танланган элемент стекдан чиқариб ташланади ва стек учини эса чиқариб ташланган элементдан битта олдин келиб тушган элемент ташкил қилиб қолади. (бундай тузилмага маълумотларга чекланган мурожаат тузилмаси дейилади).

Стекни график кўринишида қуйидагича тасвирлаш мумкин:

Биринчи элемент стек пастига киритилади.

Стек устида амалга ошириладиган амаллар:

1. PUSH(s , i) – стекга элемент киритиш, бу ерда s – стек номи, i - стекга киритиладиган элемент;
2. POP (s) – стекдан элементни танлаш. Элемент танланаётганда ўзи эгаллаб турган ишчи хотирага жойлаштирилади;
3. EMPTY (s) – стекни бўш ёки бўш эмаслигини текшириш (true - бўш, false – бўш эмас);
4. STACKTOP (s) – стек юқори элементини ўчирмасдан ўқиш.

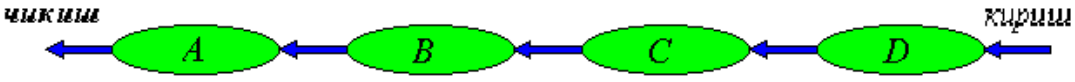
Стек яратиш дастури фрагменти (зарур процедурлар)

```
Program STACK;
  const
     max_st=50;
var
    st,st2: array[1..max_st] of integer;
    n:integer;

  function empty:boolean;  {Стекда элемент борлигини текшириш}
    begin
       empty:=n=0
    end;

  procedure push(a:char);  {стекга элементни жойлаштириш}
    begin
      inc(n);
      st[n]:=a;
    end;

  procedure pop(var a:char);  {стекдан элементни ажратиб олиш}
    begin
      a:=st[n];
      dec(n);
    end;

  function full:boolean;   {тўлаликка текшириш}

    begin
      Full:=n=max_st
    end;

  procedure stacktop(var a:char);  {юқори элементни аниқлаш}
    begin
      a:=st[n];
    end;
```

```
begin  {асосий дастур}
   .
   .
end.
```

2. Навбат

Дастурлашда шундай маълумотлар тузилмаси мавжудки, у навбат дейилади. Бундай маълумотлар тузилмаси реал навбатни моделлаштиришда катта ахамиятга эга. Бунда хизмат кўрсатишга келиб тушган талаб, унинг ижроси, яъни хизмат кўрсатиш тартибини аниқлашда зарур бўлади. Кундалик ҳаётимиздан барчамизга маълум бўлган навбат тури, дастурлашда **FIFO** (*First input-First output*, яъни биринчи келган – биринчи кетади) деб номланади. Қуйида 4 та элементдан иборат навбат келтирилган.



Бу ердан кўриниб турибдики, стекдан фарқли равишда хизмат кўрсатилиш биринчи келган элементга биринчи бўлиб хизмат кўрсатилади. Стекдан яна бир фарқи, бунда навбатнинг ҳар иккала томони очиқ бўлади, яъни бир томондан келиб иккинчи томондан чиқиб кетади.

Демак, навбатда элементни олиш рўйхат бошидан, ёзиш эса охиридан амалга оширилади.

ЭХМ хотирасида реал навбат эементлари сони чекли бўлган бир ўлчамли массив кўринишида яратилади. Албатта, бунда навбат элементи турини кўрсатиш ва навбат билан ишлашни кўрсатувчи ўзгарувчи зарур бўлади.

Навбат физик босқичда хотира соҳасини рўйхат кетма-кетлиги бўйича тўлалигича эгаллайди.

Навбат устида амалга ошириладиган амаллар:

Навбат учун 3 та оддий амал аниқланган.

1. Навбатга янги элемент жойлаштириш: insert (q,x), бу ерда q-навбат, x – элемент.
2. Навбат бошидан элементни ўчириш: remove(q)
3. Навбатни бўш ёки бўш эмаслигини аниқлаш: empty (q)

Бундан ташқари, навбат бир ўлчамли массив кўринишида ифодаланганлиги учун массивни тўла ёки тўла эмаслигини кузатиб туриш лозим бўлади. Шу мақсадда, full(q) амали киритилади.

Умуман олганда, insert амалини ҳар доим бажариш мумкин. Сабаби, навбатни ташкил қилувчи элементлар сонига чекланишлар қўйилмаган. Remove амали эса фақатгина навбат бўш бўлмагандагина ишлайди**.** Empty амали эса ҳар доим ўринли.

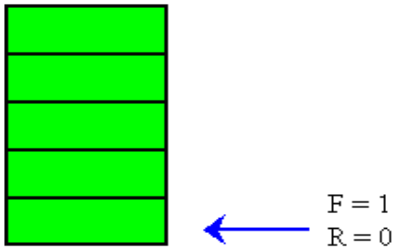
***Паскал тилида навбатни бир ўлчамли массив кўринишда амалга оширишга мисол*:**

```
const
  MaxQ = ...
```

```
type
  E = ...
  Navbat = Array [1..MaxQ] of E;
var
  Q: Navbat;
  F, R: Integer;
  {F кўрсаткич навбат бошини кўрсатадаи. R эса навбат охирини. Агар навбат
  бўш бўлса, у ҳолда F = 1 ва R = 0 бўлади. (яъни R < F – навбатнинг
  бўшлик шарти).}
Procedure Insert(I: Integer; var Q: Navbat);
begin
  Inc(R);
  Q[R]:=I;
end;

Function Empty(Q: Navbat): Boolean;
begin
    If R < F then Empty:=True Else Empty:=False;
end;

Procedure Remove(var I: Integer; Q: Navbat);
begin
  If Not Empty(Q) then
    begin
      I:=Q[F];
      Inc(F);
    end;
end;
```

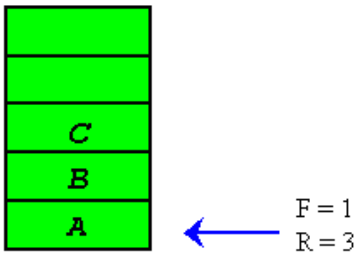
Стандарт процедуралар ёрдамида навбат билан ишлашга доир мисол.

MaxQ = 5



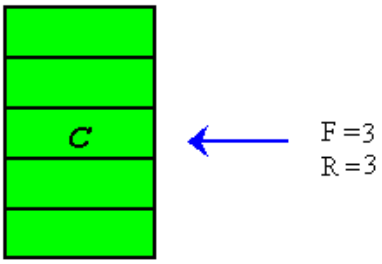
A, B ва C элементларни навбатга қўялик.

```
Insert(q, A);
Insert(q, B);
Insert(q, C);
```



A ва B элементларни навбатдан чиқарамиз.

```
Remove(q);
Remove(q);
```



Афсуски, навбатни бундай кўринишда ифодалаш, беъмани, мантиқсиз ҳолатга олиб келиши мумкин. Чунки, бундай ифодаланганда, фараз қилайлик, навбат бўш бўлсин, аммо унга янги элемент қўйиб бўлмайди (элементларни қўшиш ва чиқариш амалларини бажариб ушбу ҳолатга олиб келинг). Кўриниб турибдики, навбатни массив кўринишда ифодалаш номақбул (неприемлемо) экан.

Юзага келган муаммони ҳал қилишнинг ечимларидан бири remove амалини қуйидагича модификация қилиш бўлиб ҳисобланади. Навбатдаги элемент ўчирилганда, навбатнинг барча элементлари массив бошига сурилади (жойлаштирилади). Бундай ҳолда remove (q) амалини қуйидагича амалга ошриш мумкин.

```
X = q[1]
For I =1 to R-1
        q[I] =q[I+1]
next I
R =R-1
```

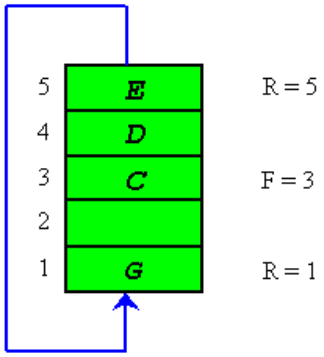
Бунда F ўзгарувчига эҳтиёж қолмайди, сабаби, массивнинг биринчи элементи ҳар доим навбат боши бўлиб ҳисобланади. Агар навбат бўш бўлса, у ҳолда R нинг қиймати ноль бўлади.

Бироқ бу усул анча самарасиз. Чунки, навбатдан элементини ўчириш доимо навбатнинг қолган элементларини силжтишни талаб қилади. Агар навбат элементлари сони қанча катта бўлса, уни амалга ошириш анча самарасиз бўлади. Бундан ташқари, элементни навбатдан ўчириш амали, мантиқан олганда, фақатгина битта элементни бошқаришдан иборат бўлади, яъни навбат бошида жойлашган элементни. Бундай усулдан, фақатгина ўчириш ишини бажариб, бошқа қўшимча ишларни амалга оширмаганда фойдаланса бўлади халос. Фараз қилайлик, навбат бошидаги элементни эмас,

балки бошқа элементни ўчирмоқчимиз, у ҳолда юқоридаги усулдан фойдалниб бўлмайди.

Навбатни ифодалашга яна бошқача ёндашув, бу массивни чизиқли кетма-кетлик кўринишида эмас, балки боши ва охирига эга бўлган чизиқсиз ёпиқ ҳалқа кўринишда ифодалашдир. Бундан келиб чиқадики, навбатнинг биринчи элементи, охирги элементидан кейиноқ келади. Бу эса агар сўнги элемент банд бўлган тақдирда ҳам биринчи элемент бўш бўлса, унинг ўрнига оҳирги элементдан кейин жойлашади.

Ҳалқасимон навбатни ташкил қилиш.



Мисол кўрамиз. Фараз қилайлик, навбат 5 элементли массивда 3 та элементга эга бўлсин: 3-чи, 4-чи ва 5-чи ўринларда. Мазкур ҳолат юқоридаги чизмада келтирилган. Гарчи, массив тўлган бўлмасада, навбатнинг сўнги элементи банд. Энди, агар навбатга G элемент жойлаштирилмоқчи бўлинса, у ҳолда бу элемент массивнинг биринчи ўрнига (позициясига) ёзилади. Навбатнинг биринчи элементи бу ерда Q(3), ундан кейин Q (4), Q(5) ва Q(l) элементлар келади.

Афсуски, навбатнинг бундай ифодаланишида, унинг бўшлигини аниқлаш етарлича қийин бўлади. Навбатни бўшлигини текшириш учун энди R<F шарт бошқа ярамайди.

Мазкур муаммони ҳал қилишнинг ечимларидан бири “келишув”ни киритишдир, бунда F нинг қиймати навбатнинг биринчи элементидан кейин келувчи массив индекси бўлади, лекин бу индекс энг биринчи элементи индекси эмас. Бундай ҳолатда, R навбатнинг энг сўнги элемент индексини билдирганлиги сабабли, F = R шарт навбатни бўшлигини англатади.

Эслатиб ўтамиз, навбат билан ишлашда F ва R га 0 ва 1 эмас, балки навбатнинг охирги элемент индекси қийматлари белгиланади, чунки, навбатнинг бундай кўринишда ифодалаганимизда, массивнинг сўнги элементи тезда биринчи элементнинг ортида бўлади. Сабаби, R = F бўлгани учун навбат бошида бўш бўлади.

Ҳалқасимон навбатдаги асосий амаллар*:*

1. Навбатга элемент қўйиш.

Insert(q,x)

2. Навбатдан элементни чиқариб ташлаш.

Remove(q)

3. Навбатни бўшлигини текшириш.

Empty(q)

empty (q) амалини қуйидагича ёзиш мумкин:

```
if F = R
    then empty = true
    else empty = false
       endif
return
```

remove (q) амалини қуйидагича ёзиш мумкин:

```
empty (q)
if empty = true
      then print «бўш навбатдан танлаш»
      stop
endif
if F =maxQ
      then F =1
      else F = F+1
endif
x = q(F)
return
```

Шуни эсда тутиш лозимки, F нинг қиймати элементни ажратиб олингунча модификация қилиниши лозим.

insert (q,x) элемент қўйиш амали.

Қўйиш амалини дастурга киритилаётганда, массив тўлиб қолиши ҳолатини таҳлил қилиш лозим бўлади. Маълумки, тўлиб қолиш агар массив навбат элементлари билан банд ҳолда яна бошқа бирор бир элемент киритмоқчи бўлсак вужудга келади. Масалан, юқоридаги чизмага яна қайтайлик. Кўриниб турибдики, бошида келтирилган массивда 3 та элемент жойлашган, улар мос равишда Q (3), Q (4) ва Q (5) позицияларда турибди. Навбатнинг сўнги элементи Q (5) позицияда турганлиги сабабли, R нинг қиймати 5 га тенг бўлади. Навбатнинг биринчи элементи Q (3) ўринда тургани учун эса, F нинг қиймати 2 бўлади. Мазкур навбатга янги G элементни қўйсак, R нинг қийматини мос равишда ўзгаришига олиб келади. Агарда навбатдаги элементни киритмоқчи бўлсак, у ҳолда массив бутунлай тўлиб қолади. Чунки, бу ҳолатда F = R бўлиб, у нарса массивни тўлалигини англатади. Кўриниб турибдики, навбатни бундай кўринишда амалга оширганимизда бўш навбат билан тўла навбат орасидаги фарқни аниқлаш имкони бўлмай қолади. Албатта, бундай ҳолат бизни қаноатлантирмайди.

Бундай ҳолатдан чиқиб кетишнинг ечимларидан бири массивда битта элементдан воз кечиш бўлиб ҳисобланади. Бунда навбатни энг максимал

ҳажмдан битта кам бўлган ҳолатгача ортириб бориш имкони вужудга келади. Масалан, агар 100 та элементли массив навбат сифатида эълон қилинган бўлса, у ҳолда бу массивда навбат 99 тагача элемент қабул қила олади. 100-чи элементни навбатга жойлаштирмоқчи бўлсак, у ҳолда тўлиб кетиш ҳолати юзага келади. Юқорида келтириб ўтилган ҳол учун insert қисм дастурини қуйидагича ифодалаш мумкин:

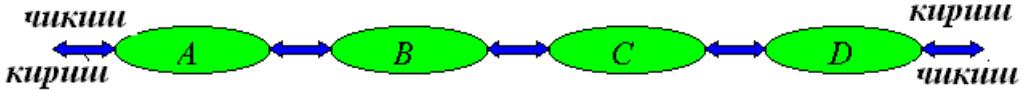
```
if R = max(q)
    then R = 1
    else R = R+1
endif
'тўлаликка текшириш
if R = F
    then print «навбат тўлиб кетган»
    stop
endif
q (r) =x
return
```

insert қисм дастурида тўлаликка текшириш R нинг янги қиймати аниқланганидан кейин амалга оширилади. Remove қисм дастурида эса бундай текшириш қисм дастурга кириб F ни қиймати эълон қилингунча.

3. Дек

Дек сўзи (DEQ - Double Ended Queue) инглиз тилидан олинган бўлиб, иккита четга эга навбат деган маънони билдиради.

Декнинг ўзига хос хусусияти шундан иборатки, элементларни ёзиш ва ўқишни ҳар иккала четидан хам амалга ошириш мумкин



Декни қуйи чегаралари бирлаштирилган иккита стек кўринишда қараш мумкин.

Дек устида бажариладиган амаллар:

1. Insert – элемент қўйиш.
2. Remove – декдан элементни чиқариб ташлаш.
3. Empty – бўш ёки бўш эмаслигини текшириш.
4. Full – тўлаликка текшириш.

Назорат саволлари.

1. Навбат билан стек қандай маълумотлар тузилмасига киради?
2. Стекдан элементни танлаш қандай амалга оширилади?
3. Стекни юқори элементини ўчирмасдан ўқиш қай йўсинда амалга оширилади?
4. Қандай хизмат кўрсатиш турига FIFO, қайси бирига LIFO деб аталади?
5. Ҳалқасимон навбатининг тўлаганлиги белгиси нимадн иборат?

6. Навбатнинг бўшлиги белгиси?
7. Рўйхат деб нимага айтилади?
8. Рўйхат турларини айтиб ўтинг.
9. Навбат элементларини санаб ўтинг.
10. Ҳалқасимон навбат қандай ташкил қилинади?
11. Декнинг ўзига хослиги нимадан иборат?

5-маъруза. Динамик маълумотлар тузилмаси

Режа:

1. Боғланган рўйхатлар ҳақида тушунча.
2. Боғламли рўйхатлар классификацияси.
3. Боғланган рўйхатларни мантиқий тасвирлаш.
4. Боғламли рўйхатлар устидаги амаллар.

Калитли сўзлар: *динамик маълумотлар тузилмаси, боғланган рўйхатлар, информацион майдон, кўрсаткичли майдон, кўп боғламли, чизиқли ва чизиқсиз боғланган рўйхат, рўйхатга элемент қўшиш, рўйхатдан элементни чиқариш.*

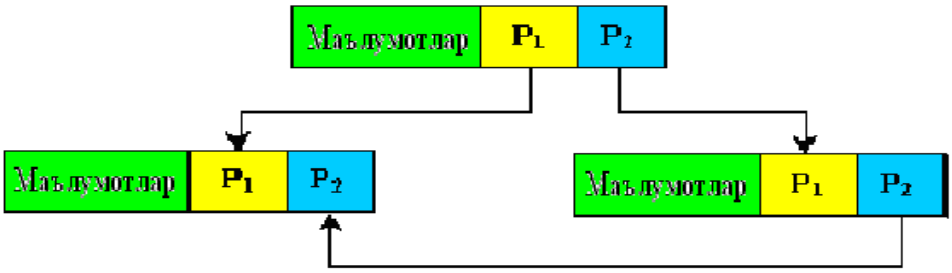
Дастур бажарилаётганда вужудга келадиган ёки ўлчамлари дастур бажарилиши мобайнида аниқланадиган объектлар динамик объектлар дейилади.

Динамик маълумотлар тузилмаси 2 та хусусиятга эга:

1) Тузилмада элементлар сони олдиндан аниқланмаган.
2) Динамик тузилма элементлари қатъий чизиқли тартибланмаган.

Улар хотирада тартибсиз жойлашган бўлиши мумкин.

Динамик тузилма элементларини ўзаро боғлаш учун тузилма элементлари таркибига информацион майдондан ташқари кўрсаткичлар майдони хам киради (қаранг, чизма) (тузилмани бошқа элементлари билан боғлиқлик).



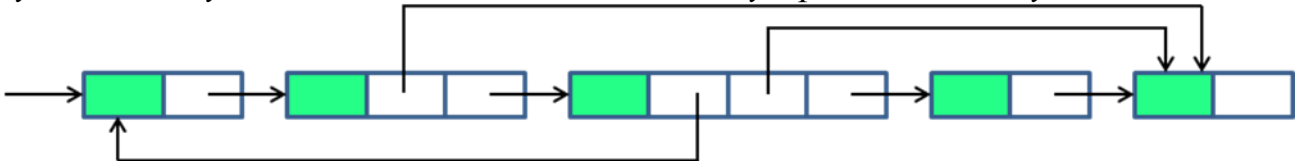
P_1 ва P_2 – ўзаро боғланган элементларни адресларини ўз ичига олувчи кўрсаткичлардир. Кўрсаткичлар слот рақамини ўз ичига олади.

Боғланган рўйхатлар

Таъриф. *Агар рўйхат элементлари кўрсаткичлар орқали боғланган бўлса, у ҳолда бундай тузилмага боғланган рўйхат деб аталади.*

Изоҳ. *Боғланган рўйхат элементининг кўрсаткичлари майдони сони бир нечта ва турли ҳил бўлиши мумкин.*

Таъриф. *Рўйхат m боғламли дейилади, агар рўйхатнинг элементлари кўпи билан тузилманинг m та элементи билан ўзаро боғланган бўлса.*



Таъриф. *Агар боғланган рўйхат элементлари мавжуд бўлмаса, у ҳолда бундай рўйхат бўш рўйхат деб аталади.*

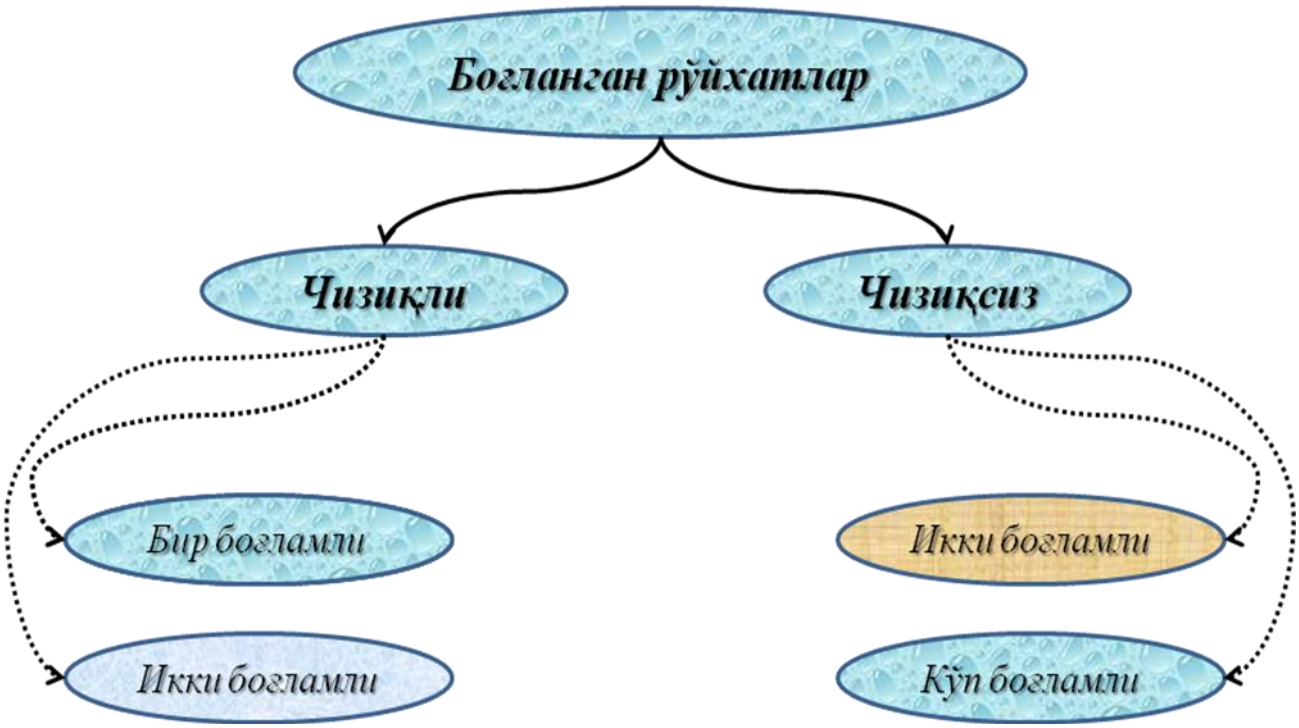
Боғламли рўйхатлар классификацияси

Боғланган рўйхатлар энг кўп тарқалган динамик тузилмалардан хисобланади. Маълумотларни мантиқий тасвирлаш нуқтаи назаридан рўйхатлар иккитага ажратилади: чизиқли ва чизиқсиз.

Чизиқли рўйхатларда элементлар орасидаги боғлиқлик қатъий тартибланган бўлиб, элемент кўрсаткичи ўзидан навбатдаги ёки олдинги элемент адресини ўз ичига олади.

Чизиқли рўйхатларга бир ва икки боғламли рўйхатлар киради. Чизиқсиз рўйхатларга эса кўп боғламли рўйхатлар киради.

Умуман олганда рўйхат элементи бир ёки бир неча кўрсаткичлар ёзуви майдонини намойиш қилади.



Боғланган рўйхатларни мантиқий тасвирлаш

Боғланган рўйхат элементлари мантиқий тасвирланишда ёзув каби ифодаланади.

Паскалда:

```
type   PNode = ^TNode;
       TNode = record
       Info: BT;    Next1, Next2, …, NextN : PNode;
       end;
var
       Lst: PNode;  {рўйхат бошини кўрсатувчи кўрсаткич}
```

P: PNode;

Боғланган рўйхат устидаги амаллар

- Рўйхатга элемент қўшиш;
- Рўйхатдан элементни ўчириш;
- Рўйхатда элементни қидириш;
- Рўйхат элементларини чоп этиш мумкин.

Эслатма: Рўйхатнинг ихтиёрий элементини ўчириш, ихтиёрий жойига элемент қўшиш мумкин.

6-маъруза. Чизиқли боғланган рўйхатлар. ОХКТни боғламли рўйхатлар кўринишида тасвирлаш.

Режа:

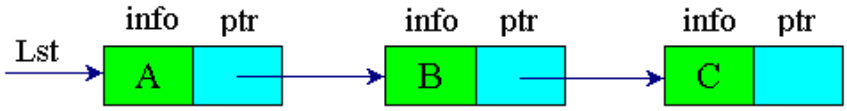
1. Бир боғламли рўйхатлар ҳақида тушунча.
2. Икки боғламли рўйхатлар ҳақида тушунча.
3. Бир ва икки боғламли рўйхатлар устидаги амаллар.
4. ОХКТни боғламли рўйхатлар кўринишида тасвирлаш.

Бир боғламли рўйхатлар

Таъриф. Агар рўйхат элементлари (тугуни) фақатгина битта кўрсаткичлар майдонига эга бўлса, у ҳолда бундай тузилмага бир боғламли ёки бир томонлама йўналтирилган рўйхат деб аталади.

Таъриф. Рўйхат элементларига мурожаат фақат рўйхат бошидан амалга оширилади. Тескари алоқа йўқ.

Бир боғламли рўйхат элементи иккита майдонга эга (чизма): информацион майдон (INFO) ва кўрсаткич майдони (PTR).

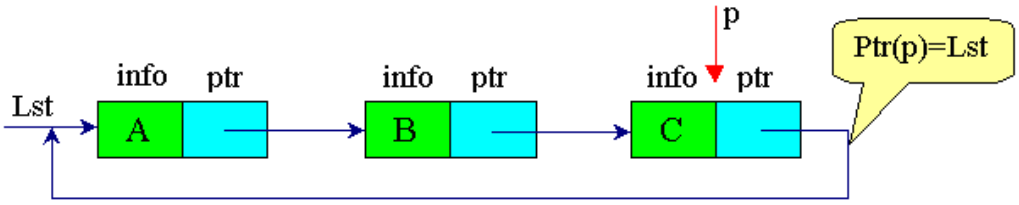


Бир боғламли рўйхатда кўрсаткични ўзига хослиги шундан иборатки, бунда фақатгина ўзидан кейин келувчи рўйхат элементи адресини кўрсатади. Рўйхат энг сўнги элементининг кўрсаткич майдони бўш бўлади (NIL). LST – рўйхат бошига кўрсаткич. Умуман олганда рўйхат бўш хам бўлиши мумкин, бу ҳолда LST NIL билан устма-уст тушади, яъни тенг бўлади.

Рўйхат элементига мурожат фақатгина рўйхат бошидан амалга оширилади, яъни бу рўйхатда тескари алоқа йўқ.

Ҳалқасимон бир боғламли рўйхат

Ҳалқасимон бир боғламли рўйхат оддий бир боғламли рўйхатда энг сўнги элемент кўрсаткичига рўйхат боши элементи кўрсаткичи қийматини ўзлаштириш орқали хосил қилинади (чизма).



Бир боғламли рўйхатлар устида бажариладиган оддий амаллар

1) Бир боғламли рўйхат бошига элемент қўйиш.

Фараз қилайлик бир боғламли рўйхат бошига информацион майдони D бўлган элемент қўйиш талаб қилинган бўлсин. Бунинг учун бўш элемент киритиш лозим (P=GetNode). Ушбу элементнинг информацион майдонига D (INFO(P)=D)нинг қиймати ўзлаштирилади, кўрсаткич қийматига эса рўйхат бошини англатувчи кўрсаткич қиймати ўзлаштирилади (Ptr(P) = Lst), рўйхат

боши кўрсаткичи қийматига эса P кўрсаткич қиймати ўзлаштирилади (Lst = P).

Паскал тилида амалга ошириш қуйидагича:

```
type
  PNode = TNode;
  TNode = record
    Info: Integer;  {рўйхат элементлари тури – ихтиёрий бўлиши мумкин}
    Next: PNode;
  end;
var
  Lst: PNode;  {рўйхат бошини кўрсатувчи кўрсаткич}
  P: PNode;
Бошига қўйиш
New(P);   {янги элемент яратиш}
P^.Info:=D;
P^.Next:=Lst; {P рўйхат бошни кўрсатади, лекин Lst  P ни кўрсатмайди – янги бошланиш}
Lst:=P;  {Lst рўйхатнинг янги бошини кўрсатади }
```

2) Бир боғламли рўйхатнинг бошидан элементни ўчириб ташлаш.

Фараз қилайлик, рўйхатнинг биринчи элементини ўчириб, лекин ушбу элемент тўғрисидаги маълумотни Info майдонида сақлаб қолиш талаб қилинсин. Бунинг учун ўчирилаётган элементни кўрсатувчи P кўрсаткич киритиб оламиз (P = Lst). X ўзгарувчига ўчирилаётган элемент информацион майдонининг қийматини берамиз (X=Info(P)). Рўйхат бошини кўрсатувчи кўрсаткичга навбатдаги элемент кўрсаткичи ўзлаштирилади (Lst = Ptr(P)) хамда элементни ўчирамиз (FreeNode(P)).

Паскал тилида амалга ошириш қуйидагича:

```
Элементни рўйхатдан ўчириш
P:=Lst;
X:=P^.Info;
Lst:=P^.Next;
Dispose(P);       {Элемент динамик хотирадан ўчирилади}
```

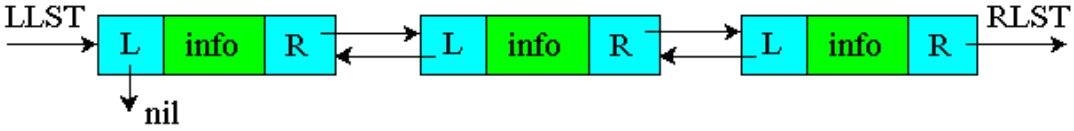
Икки боғламли рўйхат

Кўпгина масалаларни ҳал қилишда бир томонга йўналтирилган рўйхатлардан фойдаланиш маълум бир қийинчиликларни келтириб чиқаради. Сабаби, бир томонга йўналтирилган рўйхатда ҳар доим рўйхатда бош бўғимдан рўйхатнинг сўнги бўғими томонига харакатланиш мумкин холос. Лекин кўпгина масалалар ҳал қилинаётганда маълум бир элементни қайта ишлаш учун ундан олдин келган элементга мурожаат қилиш зарурати пайдо бўлади. Ушбу ҳолатда берилган элементдан олдин келган элементга мурожаат қилиш бир боғламли рўйхатда ноқулай ва анча секин амалга оширилади хамда уни амалга ошириш алгоритми мураккаблашади.

Ушбу ноқулайликларни йўқотиш мақсадида рўйхатнинг ҳар бир бўғимига яна битта майдон қўшилади. Ушбу майдон қиймати ўзидан олдин

келган бўғимга мурожаатдан иборат бўлади. Ушбу кўринишдаги элементлардан ташкил топган динамик тузилмага иккитомонлама йўналтирилган ёки икки боғламли рўйхат дейилади.

Икки боғламли рўйхатнинг ҳар бир элементи иккита кўрсаткичга эга. Биттаси олдинги элементга кўрсатади (тескари), иккинчиси навбатдаги элементни кўрсатади (тўғри) (чизма).



Умуман олганда, икки боғламли рўйхат бу элементлари сони бир ҳил фақатгина тескари кетма-кетликда ёзилган иккита бир боғламли рўйхатдир.

Ҳалқасимон икки боғламли рўйхат

Дастурлашда иккибоғламли рўйхатларни кўпинча қуйидагича умумлаштирилади: сўнги бўғин майдони қиймати Rptr сифатида бош бўғинга мурожат олинади, Lptr майдони қиймати сифатида сўнги бўғинга мурожаат қаралади

Икки бўғимли рўйхат устидаги амаллар:

- рўйхат элементини яратиш;
- рўйхатда элементни қидириш;
- рўйхатнинг кўрсатилган жойига элементни қўйиш;
- берилган элементни рўйхатдан ўчириш.

Стекларни бир боғламли рўйхатлар ёрдамида амалга ошириш

Ихтиёрий бир боғламли рўйхатни стек деб қараш мумкин. Лекин, рўйхат бир ўлчамли массив кўринишида ифодаланганда, стекга нисбатан устунликка (афзалликка) эга бўлади. Сабаби, стекда массив ўлчами олдиндан берилади, рўйхатда эса ўлчам олдиндан берилмайди.

Рўйхатга тадбиқ қилиш мумкин бўлган стек амаллари

1). Стекга элемент қўшиш учун, рўйхат алгоритмида Lst кўрсаткични Stack кўрсаткичига ўзгартириш лозим (Push(S, X) амали).

```
P = GetNode
Info(P) = X
Ptr(P) = Stack
Stack = P
```

2) Стекни бўш ёки бўш эмаслигини текшириш (Empty(S))

```
If Stack = Nil then Print “Стек бўш”
Stop
```

3) Стекдан элементни танлаш (POP(S))

```
P = Stack
X = Info(P)
Stack = Ptr(P)
FreeNode(P)
```

Паскалда амалга ошириш:

Стек

Lst кўрсаткич ўрнига Stack кўрсаткичидан фойдаланилади.

Элемент қўшиш (Push (S, X)) процедураси

```
procedure Push(var Stack: PNode; X: Integer);
var
  P: PNode;
begin
  New(P);
  P^.Info:=X;
  P^.Next:=Stack;
  Stack:=P;
end;
```

Бўшлигини текшириш (Empty)

```
function Empty(Stack: PNode): Boolean;
begin
  If Stack = nil then Empty:=True Else Empty:=False;
end;
```

Элемент танлаш (Pop) процедураси

```
procedure Pop(var X: Integer; var Stack: PNode);
var
  P: PNode;
begin
  P:=Stack;
  X:=P^.Info;
  Stack:=P^.Next;
  Dispose(P);
end;
```

Рўйхатга тадбиқ қилиш мумкин бўлган навбат амаллари.

Рўйхат боши кўрсаткичи сифатида навбат боши F кўрсаткичи, рўйхат охирги элементини кўрсатувчи кўрсаткич сифатида навбат охири R кўрсаткичи қаралади.

1) Навбатдан элементни ўчириш амали (Remove(Q, X)).

Биламизки, навбатда элементни ўчириш амали унинг бошидан амалга оширилиши лозим.

```
P = F
F = Ptr(P)
X = Info(P)
FreeNode(P)
```

2) Навбатни бўшлигини тешириш амали (Empty(Q))

```
If F = Nil then Print “Навбат бўш”
Stop
```

3) Навбатга элемент қўйиш амали (Insert(Q, X)).

Навбатга элемент қўйиш унинг охиридан амалга оширилиши лозим.

```
P = GetNode
```

Info(P) = X
Ptr(P) = Nil
Ptr(R)= P
R = P

Паскалдаги реализацияси:

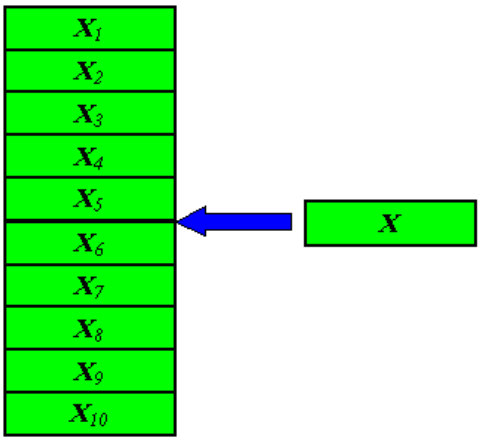
```
procedure Remove(var X: Integer; Fr: PNode);
var
  P: PNode;
begin
  New(P);
  P:=Fr;
  X:=P^.Info;
  Fr:=P^.Next;
  Dispose(P);
end;
function Empty(Fr: PNode): Boolean;
begin
  If Fr = nil then Empty:=True Else Empty:=False;
end;
procedure Insert(X: Insert; var Re: PNode);
var
  P: PNode;
begin
  New(P);
  P^.Info:=X;
  P^.Next:=nil;
  Re^.Next:=P;
end;
```

Бир боғламли рўйхат мустақил маълумотлар тузилмаси сифатида

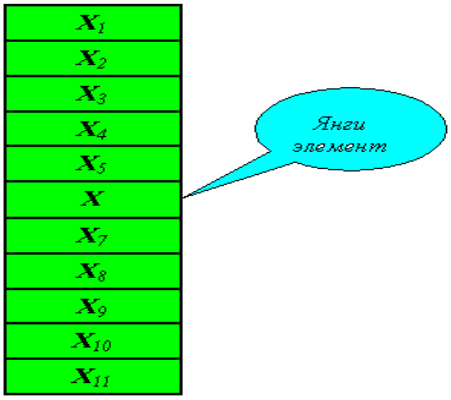
Бир боғламли рўйхатни кўриш фақатгина рўйхат бошидан бошлаб кетма-кет тарзда амалга оширилади. Агарда қайсидир позицияга келиб, ундан олдинги элементни кўрмоқчи бўлсак, у ҳолда яна рўйхат бошига қайтиш лозим бўлади. Бу эса бир боғламли рўйхатни массив турдаги статистик маълумотлар тузилмасига нисбатан камчилигини билдиради. Сабаби, массив туридаги статистик тузилмаларда ихтиёрий элементга мурожаатни амалга ошириш мумкин. Агарда рўйхат элементлари сони кўп бўлиб, рўйхат ичига элемент қўшиш лозим бўлса, у ҳолда бундай тузилма анча афзалликга эга эканлигини кўрамиз.

Масалан:

Фараз қилайлик, мавжуд массивнинг 5–чи ва 6-чи элементлари орасига янги X элемент қўйиш талаб қилинган.



Мазкур ишни массивда амалга ошириш учун, X6 дан бошлаб массив барча элементлари индексларини бир бирликга ортириш лозим. Натижада қуйидагича массивга эга бўлами (қаранг, чизма):



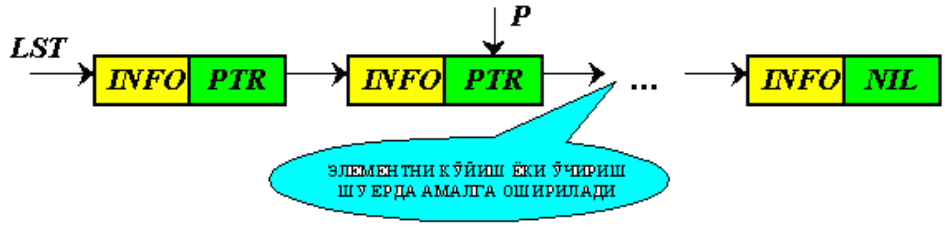
Мазкур жараён сезиларли вақтни олиши мумкин.

Бир боғламли рўйхатда эса бу жараён 2 та кўрсаткич қийматини ўзгартириш ва янги элементни ҳосил қилишдан иборат бўлиб, мазкур амалларга кетган вақт ўзгармас бўлади хамда у массив элементлари сонига боғлиқ бўлмайди.

Рўйхатга элемент қўйиш ва чиқариш

Фараз қилайлик, рўйхатга элемент қўйиш лозим. Бунинг учун биринчи навбатда, янги элементни рўйхатнинг қайси жойига қўйилиши аниқланади, яъни шундай элемент аниқланадики, янги элемент ундан кейин қўйилади. Янги элементни қўйиш InsAfter(Q, X) процедураси ёрдамида, ўчириш эса DelAfter(Q, X) процедура орқали амалга оширилади.

Бунда P ишчи кўрсаткич шундай элементни кўрсатадики, ундан кейин янги элемент қўйилади ёки кейинги элемент ўчирилади (қаранг, чизма).



Мисол:

Фараз қилайлик, рўйхатга ишчи кўрсаткичи P бўлган элементдан кейин информацион майдони X бўлган янги элемент қўйиш талаб қилинган бўлсин.

Бунинг учун:

1) Янги элементни ҳосил қилиш зарур.

Q = GetNode

2) Ҳосил қилинган элементнинг информацион майдонига X нинг қийматини ўзлаштириш.

Info(Q) = X

3) Ҳосил қилинган элементни қўйиш.

Ptr(Q) = Ptr(P)

Ptr(P) = Q

Юқорида келтирилган процедура InsAfter(Q, X) бўлиб ҳисобланади.

Фараз қилайлик, рўйхатга ишчи кўрсаткичи P бўлган элементдан кейинги элемент ўчирилиши талаб қилинган бўлсин.

Бунинг учун:

1) Ўчирилаётган элемент кўрсаткичига Q қийматни ўзлаштирамиз.

Q = Ptr(P)

2) X ўзгарувчида ўчирилаётган элемент информацион майдони қийматини сақлаймиз.

X = Info(Q)

3) Ўчирилаётган элемент кўрсаткичи қийматини ундан кейин келувчи элемент қийматига ўзгартирамиз ва ўчиришни амалга оширамиз.

Ptr(P) = Ptr(Q)

FreeNode(Q)

Мазкур процедура - DelAfter(P, X).

Юқорида келтирилган процедуралар Паскал тилида қуйидаги кўринишга эга бўлади:

```
procedure InsAfter(var Q: PNode; X: Integer);
var
  Q: PNode;
begin
  New(Q);
  Q^.Info:=X;
  Q^.Next:=P^.Next;
  P^.Next:=Q;
procedure DelAfter(var Q: PNode; var X: Integer);
var
  Q: PNode;
begin
  Q:=P^.Next;
  P^.Next:=Q^.Next;
  X:=Q^.Info;
  Dispose(Q);
end;
```

Рўйхатлар устидаги амалларга доир масалалар

1 масала.

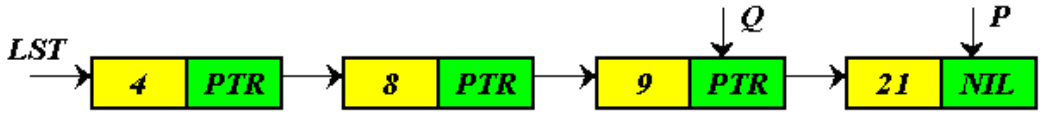
Фараз қилайлик рўйхатни текшириб, унда информацион майдони 4 га тенг бўлган элементларини ўчириш талаб қилинган бўлсин.

Р орқали ишчи кўрсаткични белгилаб оламиз, процедура бошида Р = Lst. Шундай Q кўрсаткич киритамизки, у Р дан бир элемент орқада юрсин. Р кўрсаткич керакли элементни топганда ушбу элемент Q га нисбатан навбатдаги элемент сифатида ўчирилади.

```
Q = Nil
P = Lst
While P <> Nil do
  If Info(P) = 4 then
    If Q = Nil then
      Pop(Lst)
      FreeNode(P)
      P = Lst
    Else
      DelAfter(Q, X)
    EndIf
  Else
    Q = P
    P = Ptr(P)
  EndIf
EndWhile
```

2 масала.

Информацион майдони ўсиш тартибида тартибланган рўйхат берилган бўлсин. Ушбу рўйхатга информацион майдони Х га тенг бўлган шундай элемент қўйиш талаб қилинадики, ҳосил бўлган рўйхатда тартибланганлик бузилмасин.



Фараз қилайлик Х = 16 бўлсин. Бошланғич шартлар: Q = Nil, P = Lst. Янги элемент 3-чи ва 4-чи элементлар орасига қўйилиши лозим бўлсин (юқоридаги чизма).

```
Q =Nil
P = Lst
While (P <> Nil) and (X > Info(P)) do
  Q = P
  P = Ptr(P)
EndWhile
if Q = Nil then
  Push(Lst, X)
EndIf
InsAfter(Q, X)
```

Келтирилган мисолларни Паскал тилида реализация қилиш:

1-масала:

```
Q:=nil;
P:=Lst;
While P <> nil do
  If P^.Info = 4 then
    begin
      If Q = nil then
        begin
          Pop(Lst);
          P:=Lst;
        end
      Else Delafter(Q,X);
    End;
  Else
    begin
      Q:=P;
      P:=P^.Next;
    end;
  end;
```

2-масала:

```
Q:=nil;
P:=Lst;
While (P <> nil) and (X > P^.Info) do
  begin
    Q:=P;
    P:=P^.Next;
  end;
{Мазкур жойга элемент қўйилади}
If Q = nil then Push(Lst, X);
InsAfter(Q, X);
End;
```

6-маъруза бўйича назорат саволлари

1. Чизиқли боғланган рўйхатлар.
2. Бир боғламли рўйхатлар ҳақида тушунча.
3. Бир боғламли ҳалқасимон рўйхатлар.
4. Икки боғламли рўйхатлар ҳақида тушунча.
5. Икки боғламли ҳалқасимон рўйхатлар.
6. Чизиқли рўйхатларнинг ҳалқасимон рўйхатлардан фарқи?
7. Нима сабабдан икки боғламли рўйхатлар керак?
8. Боғланган рўйхатларга элемент киритиш қандай амалга оширилади?
9. Боғланган рўйхатларга элементни ўчириш қандай амалга оширилади?
10. Боғланган рўйхатларга элементни қидириш қандай амалга оширилади?
11. Боғланган рўйхат элементларини чоп этиш қандай амалга оширилади?

12. Массивга нисбатан бир боғламли рўйхатнинг камчилиги нимадан иборат?

7-маъруза. Чизиқсиз маълумотлар тузилмаси

Режа:

1. Чизиқсиз маълумотлар тузилмаси ҳақида тушунча
2. Чизиқсиз маълумотлар тузилмаси классификацияси
3. Чизиқсиз маълумотлар тузилмасини мантиқий тасвирлаш
4. Кераксиз элементларни тузилмадан чиқариб ташлаш усуллари

Чизиқсиз маълумотлар тузилмаси ҳақида тушунча

Таъриф. Агар тузилмани ташкил этувчи элементлар қатъий тартибланмаган бўлса, у ҳолда бундай тузилмага чизиқсиз маълумотлар тузилмаси деб аталади.

Изоҳ. Чизиқсиз маълумотлар тузилмасида элементлар орасидаги муносабатлар ихтиёрий бўлиши мумкин.

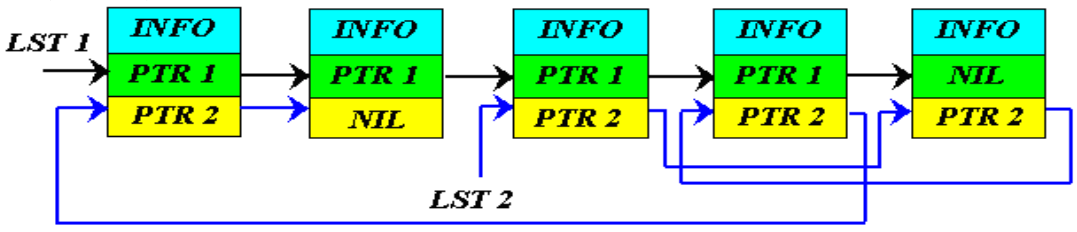
Чизиқсиз тузилмани 3 та фарқли белгисини ажратиш мумкин:

- Тузилмани хар бир элементи бошқа ихтиёрий элементга мурожаат қилиш мумкин;
- Тузилмани берилган элементига мазкур тузилманинг ихтиёрий сондаги элементи мурожаат қилиши мумкин;
- Мурожаатлар оғирликга, яъни мурожаатлар иерархик кўринишга эга бўлиши мумкин.

Чизиқсиз маълумотлар тузилмаси классификацияси

- **Рўйхатлар:**
  - ✓ чизиқсиз икки боғламли;
  - ✓ кўп боғламли;
- **Дарахтлар:**
  - ✓ бинар дарахтлар;
  - ✓ кўпўлчамли дарахтлар;
- **Графлар:**
  - ✓ йўналтирилган граф (орграф);
  - ✓ йўналтирилмаган граф (граф);
  - ✓ гиперграф.

Агар иккинчи кўрсаткичлар элементларни ихтиёрий тартибини аниқласа, у ҳолда икки боғламли рўйхатлар хам чизиқсиз боғланган тузилма бўлади (чизма).

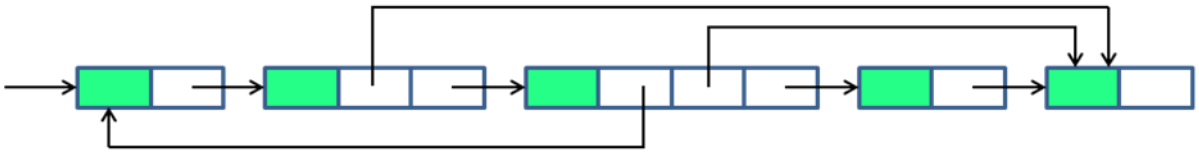


LST1 – 1-чи рўйхат бошига кўрсаткич (P1 кўрсаткич билан йўналтирилган). У чизиқли бўлиб, 5 та элементдан ташкил топган.

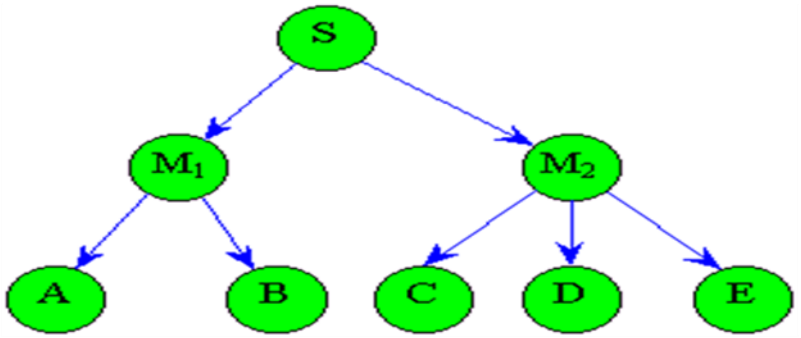
2-чи рўйхат худди шу элементлардан ташкил топган бўлиб, лекин улар тартиби ихтиёрий кетма—кетлик шаклида. 2-чи рўйхат боши 3-чи элемент бўлиб охири 2-чи элемент хисобланади.

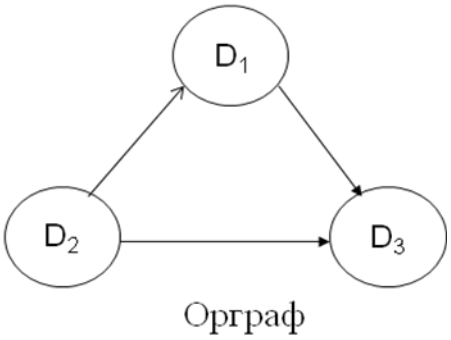
Чизиқсиз тузилмаларга мисоллар:

Чизиқсиз рўйхат



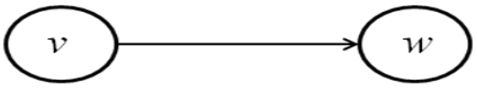
Дарахт

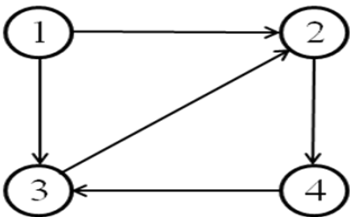




Орграф

Таъриф. G=(V,E) жуфтликка йўналтирилган граф (орграф) дейилади, бунда V - учлари тўплами (тугун), E - эса ёйлар (йўналтирилган ёқлар).





Таъриф. Орграфда йўл деб шундай v_1, v_2,…, v_n тугунлар кетма-кетлиги айтиладики, бунда $v_1 \to v_2$, $v_2 \to v_3$, … , $v_{n-1} \to v_n$ ёйлар мавжуд бўлиши шарт.

Таъриф. Йўл узунлиги деб йўлни ташкил этувчи ёйлар сонига айтилади.

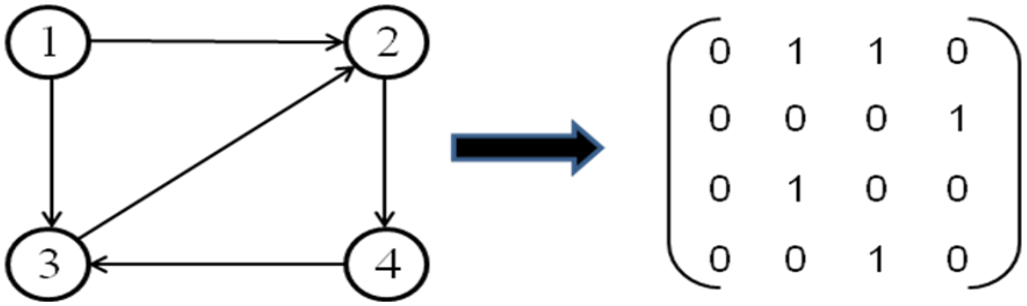
Таъриф. Йўл оддий дейилади, агар биринчи ва сўнги тугундан ташқари барча тугунлар турли ҳил бўлса.

Чизиқсиз маълумотлар тузилмасини мантиқий тасвирлаш

Ихтиёрий кўринишдаги чизиқсиз маълумотлар тузилмасини икки ҳил кўринишда мантиқий тасвирлаш мумкин:

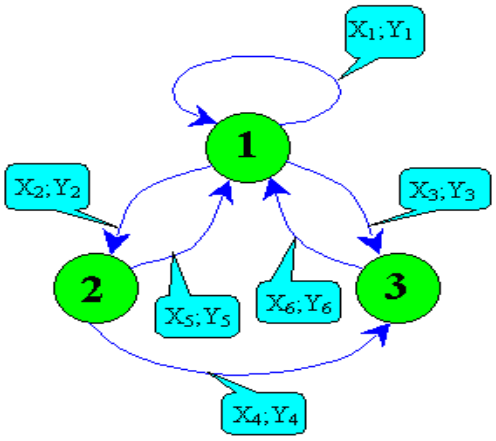
- қўшма матрица;
- кўрсаткичли боғланган рўйхат.

Масалан, қўшма матрица орқали ифодалаб олиш



Фараз қилайлик, граф кўринишидаги дискрет тизим берилган бўлсин. Бу ерда граф тугунлари бу ҳолатлар, ёқлари бир ҳолатдани иккинчи ҳолатга ўтиш (қаранг, чизма).

Тизимга кириш сигнали бу X. Кириш сигналига таъсир (реакция) Y чиқиш сигналини ҳосил қилиш ва мос ҳолатга ўтиш бўлиб ҳисобланади.

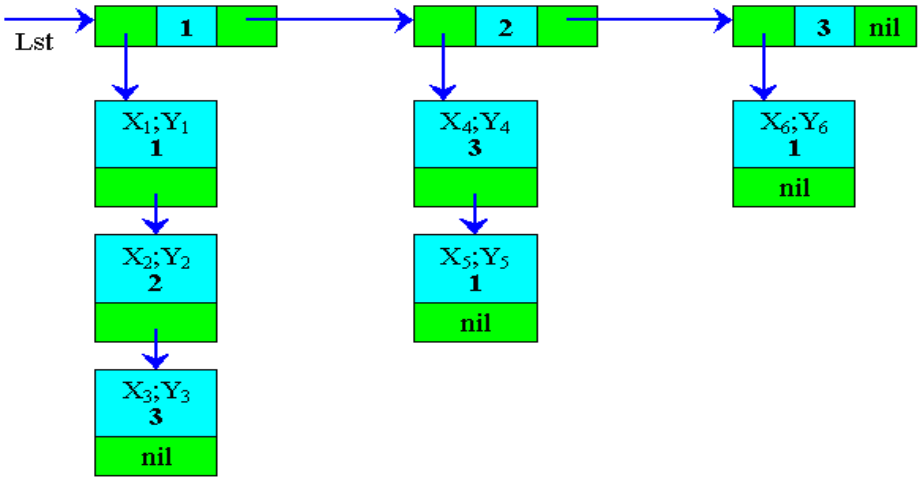


Дискерт тизим граф ҳолатини чизиқсиз икки боғламли рўйхат кўринишда тасвирлаш мумкин. Бунда информацион майдонга тизим ҳолати ва ёқлари ҳақидаги маълумотлар ёзилади. Элемент кўрсаткичлари тизим ёқларини мантиқий шакллантириши лозим (қаранг, чизма).

Юқорида келтирилганларни рўёбга ошириш учун қуйидагилар амалга оширилиши лозим:

1) Тизим (1, 2, 3) ҳолатини акслантирувчи рўйхатлар яратилиши лозим;

2) мос ҳолатлардан ёқлар бўйича ўтишни акслантирувчи рўйхатлар яратилиши лозим.



Умумий ҳолда кўпбоғламли тузилмаларни амалга ошириш натижасида тўр (сеть) ҳосил бўлади.

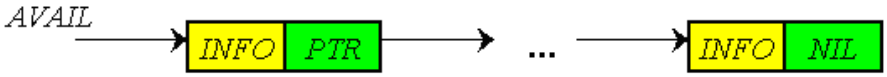
Бўшаган элементларни утилизация қилиш

Рўйхат билан ишлаётганда компьютер хотирасидан самаралироқ фойдаланиш учун, яъни кераксиз, фойдаланилмайдиган элементларни чиқариб ташлаш учун, тадқиқ қилинаётган рўйхатга ўхшаш эркли рўйхат яратилиб олинади. Бунда тадқиқ қилинаётган рўйхат майдони билан яратиб олинган рўйхат майдонлари формати бир ҳил бўлиши лозим.

Агар тадқиқ қилинаётган рўйхатлар кўп бўлиб уларни формати турли ҳил бўлса, у ҳолда ҳар бир тадқиқ қилинаётган рўйхат учун алоҳида эркли рўйхат яратилиб олиниши лозим.

Эркли рўйхатдаги элементлар сони, дастур ҳал қилаётган масалага қараб аниқланиб олиниши лозим. Одатда, эркли рўйхат машина хотирасида стек кўринишида ҳосил қилинади. Бунда янги элементни яратиш (*GetNode*) бўш стекдан элементни танлашга эквивалент бўлади, *FreeNode* амали эса бўш стекга бўшаган элементни қўшишга эквивалент.

Фараз қилайлик, биздан рўйхат боши кўрсаткичи AVAIL бўлган эркли рўйхатни стек кўринишида яратиш талаб қилинган бўлсин (Расмга қаранг). Рўйхат бўш элементини яратиш ва рўйхатдан элементни бўшатиш процедурасини ишлаб чиқайлик.



Кўп боғламли рўйхатларда бўшаган элементни утилизация қилиш

Агарда чизиқсиз кўп боғламли рўйхатдан фойдаланилаётган бўлса, у ҳолда рўйхатда бўшаган элементни бўш элементлар мажмуасига стандарт амаллар орқали қайтариш ҳар доим ҳам самарали натижа бермайди.

Кўп боғламли рўйхатларда бўш элементларни утилизация қилишнинг бир неча йўллари мавжуд. Қуйида шуларни келтириб ўтамиз.

Биринчи йўли – **хисоблагичлар (счетчик) усули.** Бунда кўп боғламли рўйхатнинг ҳар бир элементига мазкур элементга мурожатни ҳисобловчи

хисоблагич (счетчик) майдони қўйилади. Агар элемент ҳисоблагичи кўрсаткичи нол ҳолатда хамда элемент кўрсаткичи майдони nil бўлса, у ҳолда ушбу элемент бўш элементлар мажмуасига (кераксиз элементлар қаторига) қайтарилади.

Иккинчи усул – **кераксиз элементларни йиғиш усули (маркер усули).** Агарда қайсидир элемент билан алоқа ўрнатилган бўлса, у ҳолда элементнинг бир битли майдонига (маркер) “1”, акс ҳолда “0” ёзилади. Рўйхат тўлганлиги тўғрисида сигнал келганда, маркери нол бўлган элементлар қидирилади, яъни кераксиз элементларни йиғиш дастури ишга туширилади. Бунда дастур ажратилган хотирани барча қисмини қараб чиқиб, маркер билан белгиланмаган барча элементларни эркли элементлар рўйхатига қайтаради.

Назорат саволлари

1. Чизиқсиз маълумотлар тузилмасининг ўзига хослиги нималардан иборат?
2. Чизиқсиз маълумотлар тузилмасининг классификацияси.
3. Граф турлари: Орграф, граф, гиперграф.
4. Чизиқсиз маълумотлар тузилмаларини мантиқий тасвирлаш йўллари.
5. Кераксиз элементларни утилизация қилиш йўллари

8-маъруза. Рекурсив маълумотлар тузилмаси

Режа:

1. Рекурсия хақида тушунча. Рекурсив алгоритмлар.
2. Дарахтлар хақида тушунча.
3. Дарахтлар классификацияси.
4. Дарахтларни тасвирлаш.

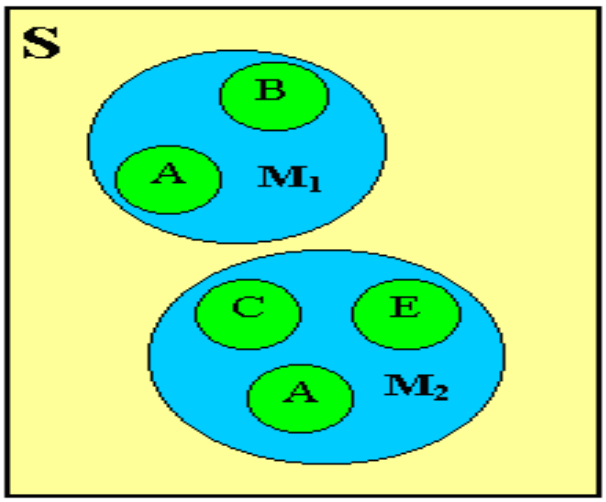
Калитли сўзлар: *рекурсия, рекурсив алгоритм, рекурсив тузилма, дарахт, кўп ўлчамли дарахт, бинар дарахт, илдиз, ота-ўғил, терминал (барг), дарахт баландлиги, чиқиш даражаси, мувозанатланган дарахт, калит, дарахт қуриш, тугун қўшиш, ўчириш, қидириш, дарахт кўруви, дарахтларни тасвирлаш.*

Рекурсив алгоритм ва рекурсив маълумотлар тузилмаларини кўриб чиқайлик.

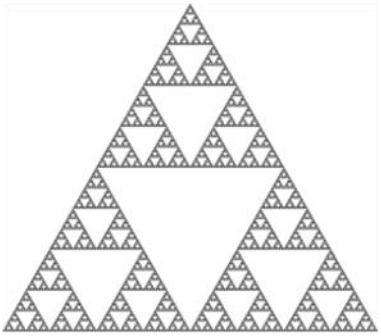
Таъриф. Рекурсия (умуман)– бу шундай жараёнки, унда тадқиқ қилинаётган жараённи аниқлаш мазкур жараёнга мурожаат қилиш орқали амалга оширилади.

Изоҳ. Рекурсив алгоритм – бу алгоритмни аниқлашда ўзига бевосита ёки билвосита мурожаат қилишдир.

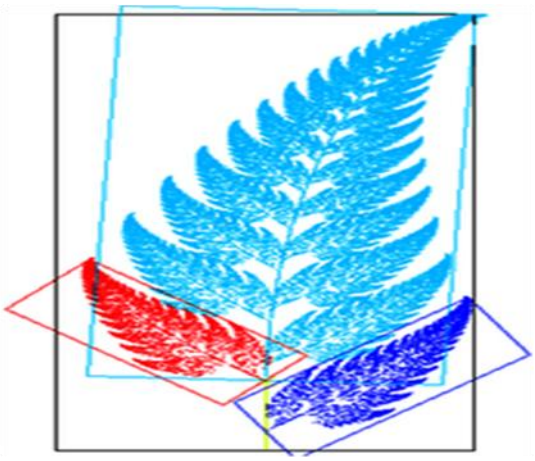
Таъриф. Агар маълумотлар тузилмаси элементлари ҳам мазкур тузилмага ўхшаш тузилма бўлса, у ҳолда бундай тузилмага рекурсив маълумотлар тузилмаси дейилади (қаранг, чизма).

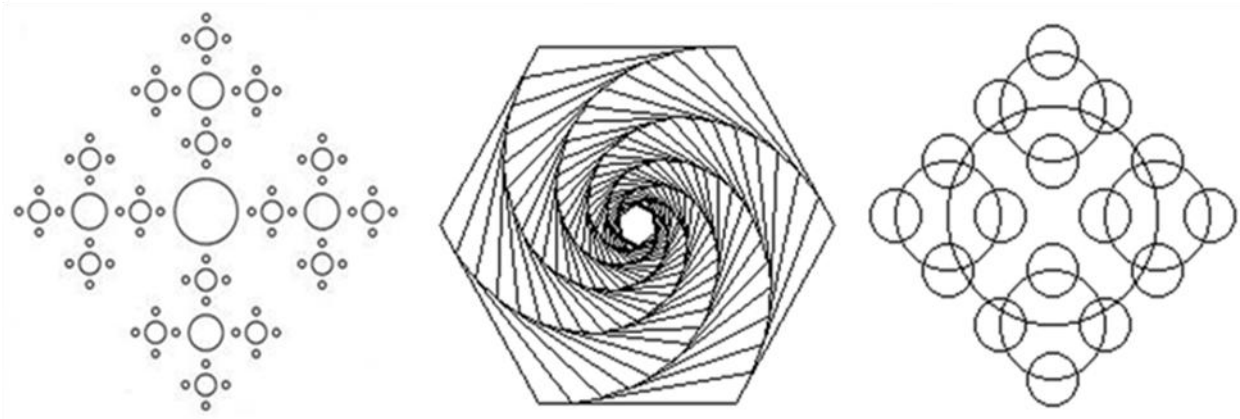


Рекурсив объектларга мисоллар



Серпинский учбурчаги





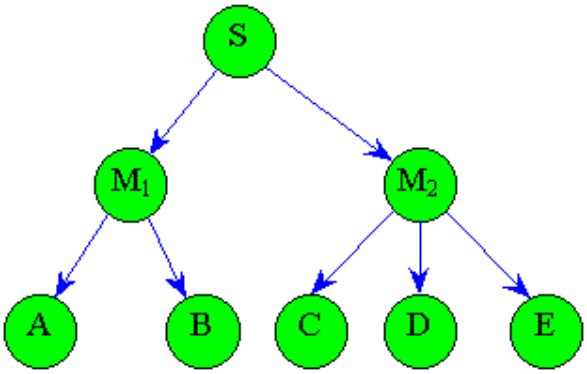
Тадқиқ қилинаётган масалани рекурсив усулда ҳал қилиш учун қуйидаги ишларни амалга ошириш лозим бўлади. Баъзан уларни масалан ҳал қилиш босқичлари ёки рекурсив триада деб ҳам аташади.

Рекурсив триада қуйидагилардан иборат:

- **параметризация қилиш** – масала шартини таснифлаш ва уни ҳал этиш учун параметрлар аниқланади;
- **рекурсия базаси (асоси)** – масала ечими аниқ бўлган тривиал ҳолат аниқланади, яъни бу ҳолатда функцияни ўзига мурожаат қилиши талаб этилмайди;
- **декомпозиция** – умумий ҳолатни нисбатан анча оддий бўлган ўзгарган параметрли қисм масалалар орқали ифодалайди.

Дарахтлар ҳақида тушунча

Дарахт – бу чизиқсиз боғланган маълумотлар тузилмасидир (қаранг, чизма).



Дарахт ўзининг қуйидаги белгилари билан таснифланади:

- дарахтда шундай битта элемент борки, унга бошқа элементлардан мурожаат йўқ. Мазкур элементга дарахт илдизи дейилади;

- дарахтда ихтиёрий элементга чекли сондаги кўрсаткичлар ёрдамида мурожаат қилиш мумкин;

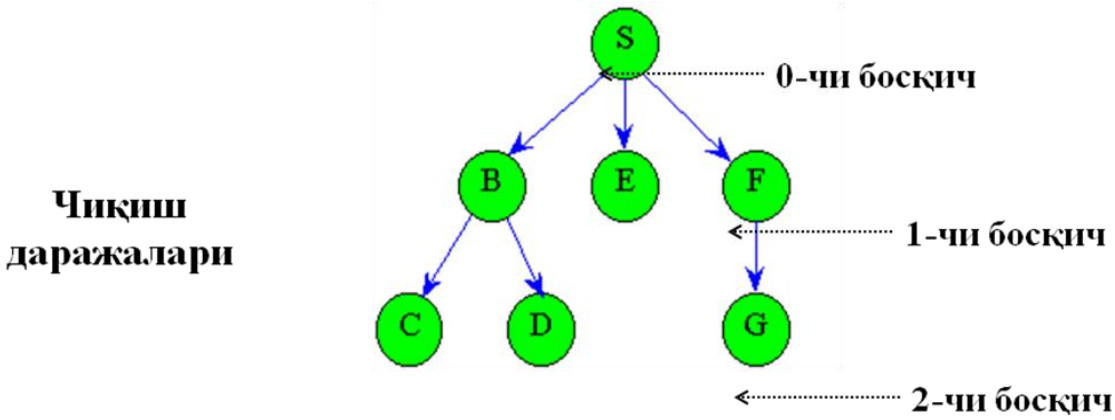
- дарахтнинг ҳар бир элементи фақатгина ўзидан олдинги келган битта элемент билан боғланган. Дарахтнинг ҳар бир тугуни оралиқ ёки терминал (барг) бўлиши мумкин. Юқоридаги чизмада M1, M2 - оралиқ, A, B, C, D, E - барглардир. Терминал тугуннинг ўзига хос таснифи унинг шохлари йўқлигидир.

Тугунлар орасидаги боғлиқликни тавсифлаш учун яна қуйидагича терминдан фойдаланилади: M1 – A ва B элементлар учун “ота” . A ва B – эса M1 тугун “ўғиллари”.

Таъриф. Дарахт босқичлари сонига дарахт баландлиги дейилади.

Юқоридаги чизмадаги дарахт баландлиги иккига тенг.

Таъриф. Дарахт тугунларидан чиқаётган шоҳлар сони тугундан чиқиш даражаси дейилади.



Дарахтлар классификацияси

Дарахтлар чиқиш даражаси бўйича синфларга ажратилади:

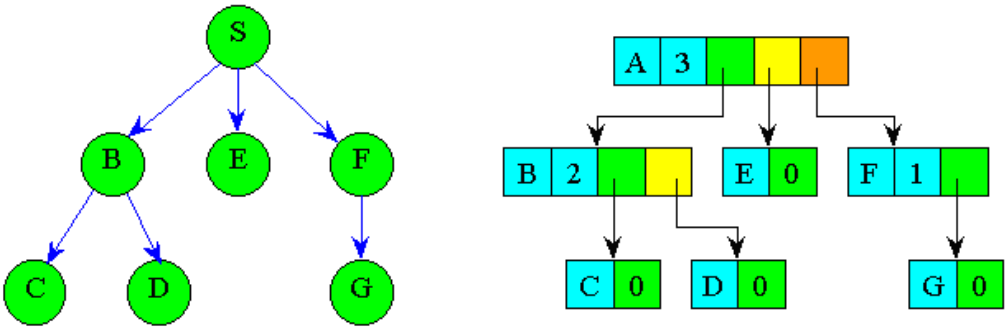
1) агар максимал чиқиш даражаси m бўлса, у ҳолда бундай дарахт m-чи тартибли дарахт дейилади;

2) агар чиқиш даражаси 0 ёки m бўлса, у ҳолда тўлиқ m-чи тартибли дарахт бўлади;

3) агар максимал чиқиш даражаси 2 бўлса, у ҳолда бундай дарахт бинар дарахт дейилади;

4) агар чиқиш даражаси 0 ёки 2 бўлса, у ҳолда тўлиқ бинар дарахт дейилади.

Дарахтларни тасвирлаш



Дарахтни график шаклдаги ва унинг чизиқсиз рўйхат шаклидаги ифодаланиши

ЭХМ хотирасида дарахтни ифодалашанинг энг қулай усули бу уни боғланган рўйхатлар кўринишида ифодалашдир. Рўйхат элементи тугун қиймати ва чиқиш даражасини ўз ичига олувчи информацион майдонга хамда чиқиш даражасига тенг бўлган кўрсаткичлар майдонига эга бўлиши лозим (юқоридаи чизма), яъни элементнинг ҳар бир кўрсаткичи ушбу элементни тугун ўғиллари бўлган тугунларга йўналишини аниқлайди.

Назорат саволлари

1. *Рекурсия нима?*
2. *Рекурсив объект, алгоритм, функция тушунчаси.*
3. *Рекурсив триада.*
4. *Рексурсив алгоритм самарадорлигини аниқлаш ва ошириш йўллари.*
5. *Дарахт тушунчаси: баландлиги, чиқиш даражаси.*
6. *Дарахтлар классификацияси.*
7. *Дарахтларни мантиқий тасвирлаш.*

9-маъруза. Бинар дарахтлар ва ундаги амаллар

Режа:

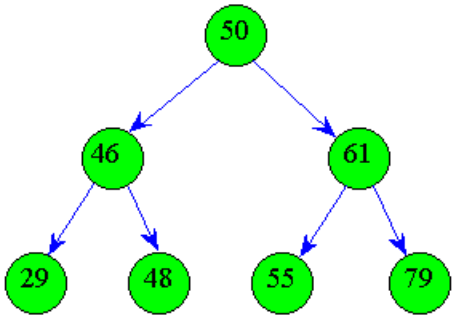
1. Бинар дарахтлар ҳақида тушунча.
2. Кўп ўлчамли дарахтни бинар кўринишга келтириш.
3. Дарахтлар устидаги амаллар.

Бинар дарахтлар

Бинар дарахтлар энг кўп фойдаланиладиган дарахтлар тури хисобланади.

Дарахтларни ЭХМ хотирасида тасвирланишига кўра хар бир элемент тўртта майдонга эга ёзув хисобланади. Мазкур майдонлар қиймати мос равишда ёзув калити бўлиб, бошқа элементларга мурожаатни ифодалайди, яъни чапга-пастга, ўнга-пастга ва ёзув матнига.

Шуни эсда тутиш лозимки, дарахт хосил қилинаётганда, отага нисбатан чап томондаги ўғил қиймати кичик калитга, ўнг томондаги ўғил эса катта қийматли калитга эга бўлади. Масалан, қуйидаги элементлардан бинар дарахт қурамиз: 50, 46, 61, 48, 29, 55, 79. У қуйидаги кўринишга эга бўлади:



Натижада, ўнг ва чап қисм дарахтлари бир хил босқичли тартибланган бинар дарахт хосил қилдик. Агар дарахтнинг ўнг ва чап қисм дарахтлари босқичлари фарқи бирдан кичик бўлса, бундай дарахт идеал мувозанатланган дарахт дейилади. Юқорида хосил қилган бинар дарахтимиз идеал мувозанатланган дарахтга мисол бўлади.

Бинар дарахтни хосил қилиш учун ЭХМ хотирасида элементлар қуйидаги турда бўлиши лозим:

V = MakeTree(Key, Rec) амали иккита кўрсаткичли (калит) ва иккита майдонли (информацион) элемент яратади (дарахт тугуни)

MakeTree процедурси кўриниши:

Паскаль

```
New(p);
p^.r := rec;
p^.k := key;
v := p;
```

```
p^.left := nil;
p^.right := nil;
```

Бошида калит биринчи қиймати киритилади. Ундан сўнг элементни ўзини maketree процедураси орқали ҳосил қиламиз. Кейин эса кўрсаткич бўш қийматни кўрсатгунча циклни давом эттирамиз.

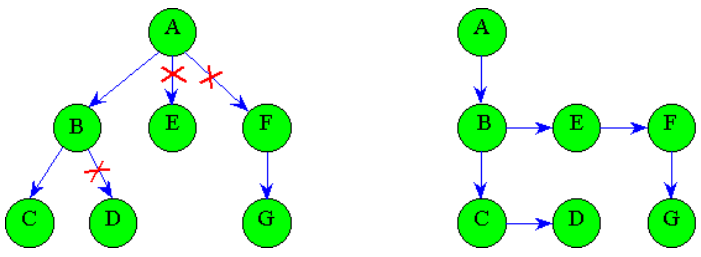
```
READ(key,rec)
tree=maketree(key,rec)
WHILE not eof DO
  READ(key,rec)
  V=maketree(key,rec)
  WHILE P<>nil DO
     Q=P
     IF key=k(P)
       THEN P=left(P)
       ELSE P=right(P)
    END IF
 END WHILE
 IF P=nil
  THEN WRITELN(' Бу илдиз');
  tree=V
  ELSE IF key<k(q)
          THEN left(P)=V
          ELSE right(P)=V
         END IF
   END IF
END WHILE
```

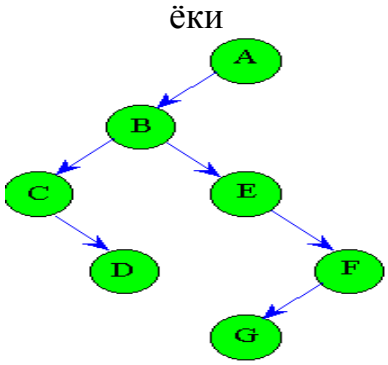
m-ўлчамли дарахтни бинар кўринишга келтириш

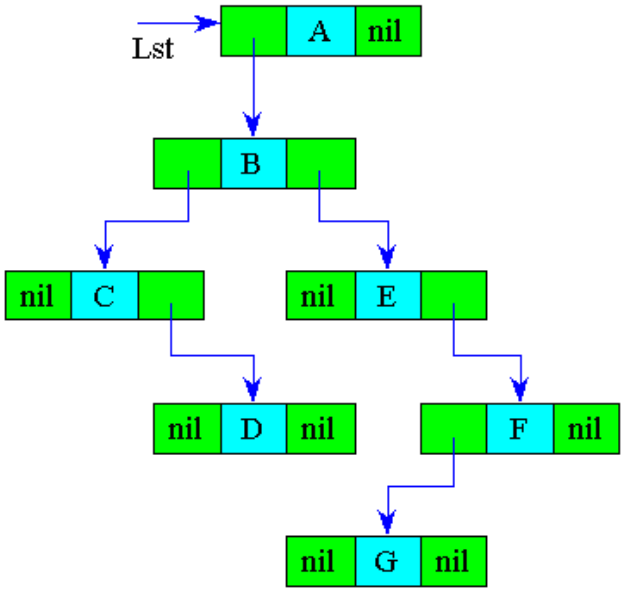
Ноформал алгоритм:

1. Дарахтнинг хар бир тугунида катта ўғилга мос четки чап шохидан ташқари барча шохлари кесиб ташланади.
2. Битта ота барча ўғиллари горизонтал чизиқ билан уланади.
3. Хосил қилинган тузилманинг хар бир тугунида катта ўғил мазкур тугун пастида турган тугун хисобланади (агар у мавжуд бўлса).

Алгоритм амаллар кетма-кетлиги қуйида келтирилган.







m-ўлчовли дарахтни бинар кўринишга келтириш.

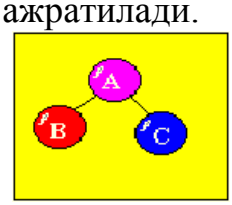
Дарахтлар устида бажариладиган амаллар

1. Дарахт кўруви (Обход дерева).
2. Қисм дарахтни ўчириш.
3. Қисм дарахт қўйиш.

Дарахт кўрувини амалга ошириш учун қуйидаги учта процедурани бажариш лозим:

1. Илдизни қайта ишлаш.
2. Чап тармоқ(шох)ни қайта ишлаш.
3. Ўнг тармоқ(шох)ни қайта ишлаш.

Юқоридаги процедура қандай кетма-кетликда амалга оширилишига қараб кўрувни учта кўринишга ажратилади.

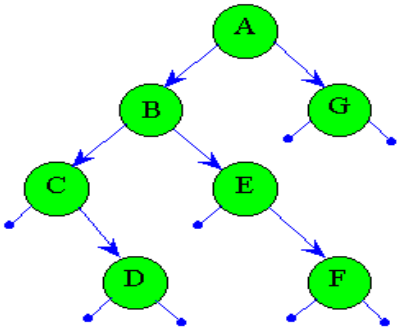


1. Юқоридан қуйига. Процедур қуйидаги кетма-кетликда бажарилади A-B-C.
2. Чапдан ўнг. Процедур қуйидаги кетма-кетликда бажарилади

B-A-C.

3. Қуйидан юқорига. Процедур қуйидаги кетма-кетликда бажарилади B-C-A.

Масалан қуйидаги дарахтда кўрув ўтказайлик.



Дарахат кўруви тартиби:
Юқоридан пастга: A,B,C,D,E,F,G.
Чапдан ўнга: C,D,B,E,F,A,G.
Пастдан юқорига: D,C,F,E,B,G,A.

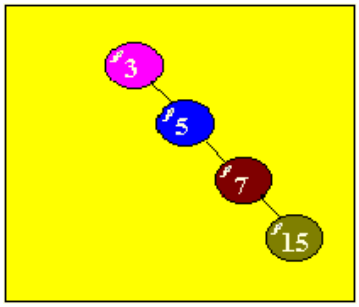
Дарахт кўригини рекурсив процедурлари:

```
1) PROCEDURE pretrave (tree) {юқоридан пастга кўрик}
      IF tree<>nil
       THEN PRINT info (tree)
            pretrave (left (tree))
            pretrave (right (tree))
      END IF
    RETURN
2) PROCEDURE intrave (tree)

      IF tree<>nil
       THEN intrave (left (tree))
            PRINT info (tree)
            intrave (right (tree))
      END IF
      RETURN
       3) PROCEDURE postrave (tree)

      IF tree<>nil
       THEN postrave (left (tree))
            postrave (right (tree))
            PRINT info (tree)
      END IF
      RETURN
```

Бинар дарахт бўйича қидирув процедураси

Мазкур процедуранинг вазифаси шундан иборатки, у берилган калит бўйича дарахт тугуни қидирувини амалга оширади. Қидирув операциясининг

давомийлиги дарахт тузилишига боғлиқ бўлади. Ҳақиқатдан, агар элементлар дарахтга калит қийматлари ўсиш (камайиш) тартибида келиб тушган бўлса, у ҳолда дарахт бир томонга йўналган рўйхат ҳосил қилади (чиқиш даражаси бир бўлади, яъни ягона шоҳга эга), масалан:



Бу ҳолда дарахтда қидирув вақти, бир томонлама йўналтирилган рўйхатдаги каби бўлиб, ўртача қараб чиқишлар сони N/2 бўлади.

Агар дарахт мувозанатланган бўлса, у ҳолда қидирув энг самарали натижа беради. Бу ҳолда қидирув $\log_2 N$ дан кўп бўлмаган элементларни кўриб чиқади.

Қидирув процедурасини кўриб чиқамиз. search ўзгарувчисига топилган бўғин кўрсаткичи ўзлаштирилади:

```
p=tree
  WHILE p<>nil DO
     IF key=k (p)
       THEN search=p
      RETURN
     END IF
     IF key<>k (p)
       THEN p=left (p)
       ELSE p=right (p)
     END IF
  END WHILE
  search=nil
  RETURN
```

Дарахтга янги элемент қўшиш процедураси

Дарахтга бирор бир элементни қўшишдан олдин дарахтда берилган калит бўйича қидирувни амалга ошириш лозим бўлади. Агар берилган калитга тенг калит мавжуд бўлса, у ҳолда дастур ўз ишини якунлайди, акс ҳолда дарахтга элемент қўшиш амалга оширилади.

Дарахтга янги ёзувни киритиш учун, аввало дарахтни шундай тугунини топиш лозимки, натижада мазкур тугунга янги элемент қўшиш мумкин бўлсин. Керакли тугунни қидириш алгоритми ҳам худди берилган калит бўйича тугунни топиш алгоритми каби бўлади. Бироқ берилган калит бўйича қидирув процедурасидан тўғридан-тўғри (бевосита) фойдаланиб бўлмайди, сабаби, қидирув процедурасида, қайси тугунда мурожаат NIL (search = nil) бўлгани фиксирланмайди.

Қидирув процедурасини шундай модификация қиламизки, қўшимча самара сифатида янги процедурамиз берилган калит турган тугунни фиксирласин (қидирув мувофақиятли бўлса), ёки шундай тугунники, ушбу тугунни қайта ишлагандан кейин қидирув якунлансин (қидирув мувофақиятли бўлса).
Дарахтда қўшилаётган элемент калитига тенг калитли элемент йўқ бўлган ҳолда элементни қўшиш процедурасини келтириб ўтамиз.

```
q=nil
p=tree
WHILE p<>nil DO
        q=p
    IF key=k (p)
      THEN search=p
      RETURN
    END IF
    IF key<k (p)
      THEN p=left (p)
      ELSE p=right (p)
    END IF
END WHILE
```

{Берилган калитга тенг тугун топилмади, элемент қўшиш талаб қилинади. Ота бўлиши мумкин тугунга q кўрсаткич берилади.}

```
V=maketree (key, rec)
```

{Қўйилаётган V элемент чап ёки ўнг ўғил бўлишини аниқлаш лозим.}

```
 IF key<k (q)
   THEN left (q)=V
   ELSE right (q)=V
 END IF
 search=V
RETURN
```

Бинар дарахтдан элементни ўчириш процедураси

Тугунни ўчириб ташлаш натижасида дарахтнинг тартибланганлиги бузилмаслиги лозим. Тугун дарахтда ўчирилаётганда 3 ҳил вариант бўлиши мумкин:

1) Топилган тугун терминал (барг). Бу ҳолатда тугун шунчаки ўчириб ташланади.

2) Топилган тугун фақатгина битта ўғилга эга. У ҳолда ўғил ота ўрнига жойлаштирилади.

3) Ўчирилаётган тугун иккита ўғилга эга. Бундай ҳолатда шундай қисм дарахтлар звеносини топиш лозимки, уни ўчирилаётган тугун ўрнига қўйиш мумкин бўлсин. Бундай звено ҳар доим мавжуд бўлади:

- бу ёки чап қисм дарахтнинг энг ўнг томондаги элементи (ушбу звенога эришиш учун кейинги учига чап шоҳ орқали ўтиб, навбатдаги учларига эса, мурожаат NIL бўлмагунча, фақатгина ўнг шоҳлари орқали ўтиш зарур).

- ёки ўнг қисм дарахтнинг энг чап элементи (ушбу звенога эришиш учун кейинги учига ўнг шоҳ орқали ўтиб, навбатдаги учларига эса, мурожаат NIL бўлмагунча, фақатгина чап шоҳлари орқали ўтиш зарур).

Ўчирлаётган элемент чап қисм дарахтининг энг ўнгидаги элемент ўчирилаётган элемент учун “Предшественник” бўлади (12 учун – 11 бўлади). Меросхўр эса ўнг қисм дарахтнинг энг чапидаги тугуни (12 учун - 13).

Меросхўрни топиш алгоритмини ишлаб чиқайлик (5.9 чизма).

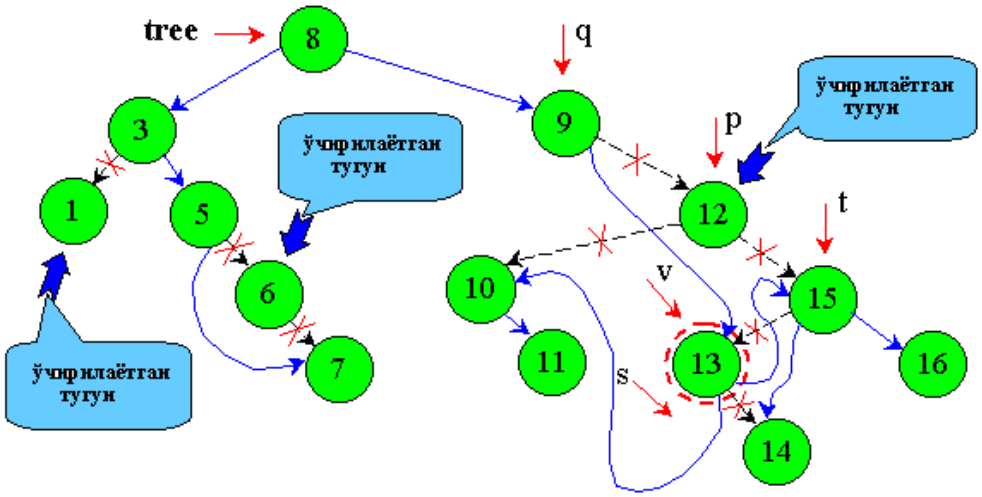
p – ишчи кўрсаткич;

q - p дан бир қадам орқадаги кўрсаткич;

v – ўчиралётган тугун меросхўрини кўрсатади;

t - v бир қадам орқада юради;

s - v дан бир қадам олдинда юради (чап ўғилни ёки бўш жойни кўрсатиб боради).



5.9-чизма. Дарахт тугунини ўчириш

Юқоридаги дарахт бўйича қарайдиган бўлсак, охир оқибатда, v кўрсаткич 13 тугунни, s эса бўш жойни кўрсатиши лозим.

1) Элементни қидириш процедураси орқали ўчирилаётган элементни топамиз. p кўрсаткич ўчириланётган элементни кўрсатади.

2) Ўчириладиган элементни ўрнига қўйилувчи тугунга v кўрсаткич қўямиз.

```
IF left (p)=nil
  THEN v=right (p)
  ELSE IF right (p)=nil
          THEN v=left (p)
          ELSE t=p
                  v=right (p)
                  s=left (v)
```

```
WHILE s<>nil
          t=v
          v=s
          s=left (v)
END WHILE
     IF t<>p
      THEN WRITE (v элемент p нинг ўғли эмас)
               left (t)=right (v)
              right (v)=right (p)
     END IF
left (v)=left (p)
      IF q=nil
       THEN WRITE (v илдиз)
                tree=v
       RETURN
      END IF
      IF p=left (q)
       THEN left (q)=v
       ELSE right (q)=v
      END IF
   END IF
END IF
```
FREENODE (p) (ушбу процедура бўш тугун ҳосил қилади, яъни ўчирилган элемент жойлашган хотира ячейкасини тозалайди.)
RETURN

Назорат саволлари

1. Бинар дарахт тушунчаси.
2. Бинар дарахт қандай хосил қилинади?
3. Кўп ўлчамли дарахтни қандай қилиб бинар дарахт кўринишига келтириш мумкин?
4. Дарахтда қандай амалларни бажариш мумкин?
5. Дарахтда кўрув қандай амалга оширилади?

10-маъруза. Қидирув алгоритмларини тадқиқ қилиш

Режа:

1. Қидирув тушунчаси
2. Қидирув алгоритмлари
3. Қидирув алгоритмларининг самарадорлиги
4. Қидирувни мукаммаллаштириш усуллари

Калитли сўзлар: *қидирув, жадвал, файл, калит, ноёб калит, иккиламчи калит, ташқи калит, ички калит, кетма-кет қидирув, индексли кетма-кет қидирув, қидирув самарадорлиги, жадвални қайта тартиблаш, транспозиция, бинар қидирув, мукаммал дарахт.*

Маълумки, ахборот технологиялари жадал суратлар билан

ЭХМда маълумотларни қайта ишлашда қидирув асосий амаллардан бири бўлиб ҳисобланади. Унинг вазифаси берилган аргумент бўйича массив маълумотлари ичидан мазкур аргументга мос маълумотларни топишдан иборат.

Ихтиёрий маълумотлар мажмуаси жадвал ёки файл деб аталади. Ихтиёрий маълумот (ёки тузилма элементи) бошқа маълумотдан бирор бир белгиси орқали фарқ қилади. Мазкур белги калит деб аталади. Калит ноёб бўлиши, яъни мазкур калитга эга маълумот жадвалда ягона бўлиши мумкин. Бундай ноёб калитга бошланғич (биринчи) калит дейилади. Иккинчи калит бир жадвалда такрорлансада у орқали хам қидирувни амалга ошириш мумкин. Маълумотлар калитини бир жойга йиғиш (бошқа жадвалга) ёки ёзув сифатида ифодалаб битта майдонга калитларни ёзиш мумкин. Агар калитлар маълумотлар жадвалидан ажратиб олиниб алоҳида файл сифатида сақланса, у ҳолда бундай калитлар ташқи калитлар дейилади. Акс ҳолда, яъни ёзувнинг бир майдони сифатида жадвалда сақланса ички калит дейилади.

Калитни берилган аргумент билан мослигини аниқловчи алгоритмга берилган аргумент бўйича қидирув деб аталади. Қидирув алгоритми вазифаси керакли маълумотни жадвалда топиш ёки йўқлиги аниқлашдан иборатдир. Агар керакли маълумот йўқ бўлса, у ҳолда иккита ишни амалга ошириш мумкин:

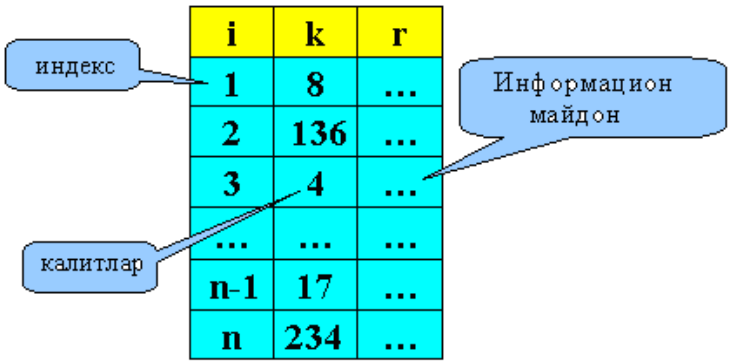
1. маълумот йўқлигини индикация (белгилаш) қилиш
2. жадвалга маълумотни қўйиш.

Фараз қилайлик, k – калитлар массиви. Ҳар бир k(i) учун r(i) – маълумот мавжуд. Key – қидирув аргументи. Унга rec - информацион ёзув мос қўйилади. Жадвалдаги маълумотларнинг тузилмасига қараб қидирувни бир неча турлари мавжуд.

1. Кетма-кет қидирув

Мазкур кўринишдаги қидирув агар маълумотлар тартибсиз ёки улар тузилиши ноаниқ бўлганда қўлланилади. Бунда маълумотлар бутун жадвал бўйича оператив хотирада кичик адресдан бошлаб, то катта адресгача кетма-кет қараб чиқилади.

Массивда кетма-кет қидирув (search ўзгарувчи топилган элемент рақамини сақлайди).



C++ тилида дастур қуйидагича бўлади:

```
#include <iostream.h>
#include <conio.h>
int search(int a[], int N, int key)
{
    int i=0;
    while (i!=N)
if (a[i]==key) return i;
    else i++;
return -1;
}
main ()
{
int i, N, mas[1000], key, P;
cout<<”Masiv uzunligini kiriting!”<<endl;
cin>>N;
cout<<”Massiv elementlarini kiriting!”<<endl;
for (i=0; i<N; i++)
cin>>mas[i];
cout<<”Qidiruv elementini kiriting!”<<endl;
cin>>key;
P=search(mas,N,key);
if (P==-1) cout<<”Bunday element massivda yoq”;
else cout<<”Bunday element bor”<<P+1;
getch();
return 0;
}
```

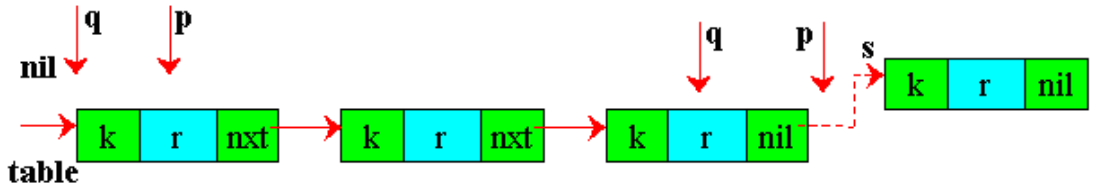
Массивда кетма-кет қидирув алгоритми самарадорлигини бажарилган таққослашлар сони M билан аниқлаш мумкин. M_{min} = 1, M_{max} = n. Агар маълумотлар массив ячейкасида бир ҳил эхтимоллик билан тақсимланган бўлса, у ҳолда $M_{ср} \approx (n + 1)/2$ бўлади.

Агар керакли элемент жадвалда бўлмаса ва мазкур элементни жадвалга қўшиш лозим бўлса, у ҳолда юқоридаги дастурдаги охирги иккита оператор қуйидагига алмаштирилади.

Паскалда

```
n:=n+1;
k[n]:=key;
r[n]:=rec;
search:=n;
exit;
```

Агар маълумотлар жадвали бир боғламли рўйхат кўринишида берилган бўлса, у ҳолда кетма-кет қидирув рўйхатда амалга оширилади.



Алгоритмлар варианти:

```
#include <iostream.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
struct TNode {
  int value;
  TNode* pnext;
  TNode(int val): pnext(0), value(val) {}
};

//Ro'yhatga element qo'shish
void add2list(TNode **pphead, int val) {
  TNode **pp = pphead, *pnew;
  //while(*pp) {
    //if(val < (*pp)->value)
        // break;
    //else
   //    pp = &((*pp)->pnext);
  //}
  pnew = new TNode(val);
  pnew->pnext = *pp;
  *pp = pnew;
}

//Ro'yhat elementlarini ekranga chiqarish
void print(TNode *phead) {
```

```
  TNode* p = phead;
  while(p) {
    cout <<""<< p->value<<"|" <<" "<<"|"<<p<<"|"<< "--> ";
    p = p->pnext;
  }
  cout << endl;
}
// Ro'yhatda element qidirish, C++
TNode* Find(TNode *phead, int x)
{
TNode *p=phead;
 while(p)
 if (p->value==x) return p;
        else p = p->pnext;
        return 0;

}

//Ro'hat elementini o'chirish

void deleteList(TNode *phead) {
  if(phead) {
    deleteList(phead->pnext);
    if(phead)
        delete phead;
  }
}
//Asosiy qism
int main() {int mas[1000], N, key; TNode* T;
  clrscr();
  TNode *phead = 0;
  //srand(time(0));
  cout<<"Royhat uzunligini kirit"<<endl;
  cin>>N;
  cout<<"Elementlarni kirit!"<<endl;
  for(int j=0; j<N; j++)
  cin>>mas[j];
  for(int i = 0; i < N; ++i)
    add2list(&phead,mas[i]);
    cout<<"Qidiruv elementni kiriting!"<<endl;
    cin>>key;
    // rand() % 100);

  cout << "Elements of the list:" << endl;
  print(phead);
```

```
  T=Find(phead,key);
  if (T==0) cout <<"Bunday element yoq"<<endl;
  else cout <<"Bunday element bor"<<" "<<T->value<<" "<<T<<endl;
  //deleteList(phead);
  getch();
}
```

Паскалда:

```
    q:=nil;
    p:=table;
    while (p <> nil) do
        begin
        if p^.k = key then
                begin
                        search: = p; exit;
                end;
        q := p;
        p := p^.nxt;
        end;
    New(s);
    s^.k:=key;
    s^.r:=rec;
    s.^nxt:= nil;
    if q = nil then table = s
                else q.^nxt = s;
    search:= s;
    exit
```

Рўйхатли тузилманинг афзаллиги шундан иборатки, рўйхатга элементни қўшиш ёки ўчириш тез амалга ошади, бунда қўшиш ёки ўчириш элемент сонига боғлиқ бўлмайди, массивда эса элементни қўшиш ёки ўчириш тахминан барча элементларни яримини силжитишни талаб қилади. Рўйхатда қидирувни самарадорлиги тахминан массивники билан бир ҳил бўлади.

Умуман олганда кетма-кет қидирув самарадорлигини ошириш мумкин.

Фараз қилайлик, кун давомида маълумотлар йиғилиб, кечқурун улар қайта ишлансин. Маълумотлар тўплангандан кейин улар сараланади.

2. Индексли кетма-кет қидирув

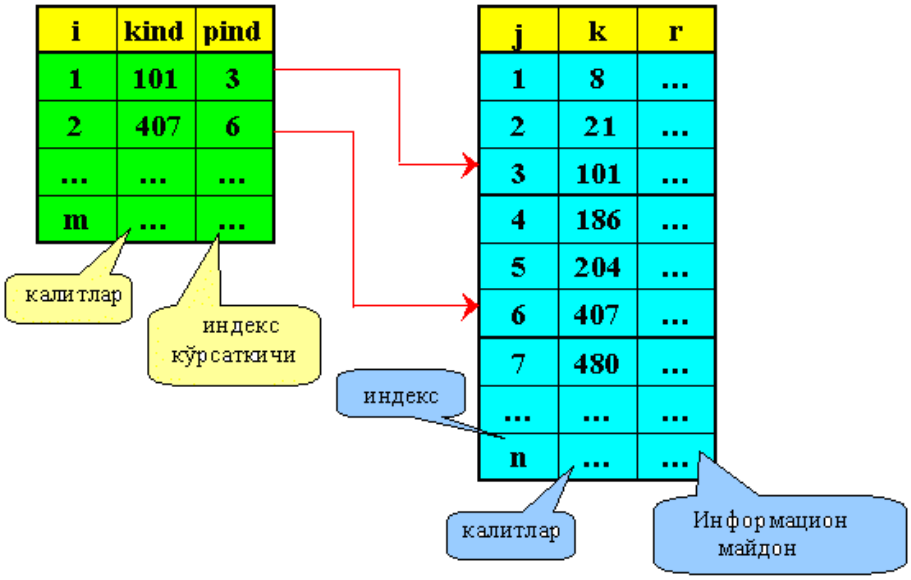
Мазкур кўринишдаги қидирув амалга оширилаётганда иккита жадвал ташкил қилинади: ўз калитига эга маълумотлар жадвали (ўсиш тартибида тартибланган) ва индекслар жадвали, бу хам маълумотлар калитидан иборат-у, лекин бу калитлар асосий жадвалдан аниқ бир интервал орқали олинган. (2-чизма).

Бошида берилган аргумент бўйича кетма-кет қидирув индекслар жадвалида амалга оширилади. Қачонки, биз берилган калитдан кичик

калитни аниқлаганимизда, асосий жадвалда қидирувни қуйи чегарасини ўрнатамиз - low, кейин эса юқори чегарани - hi, яъни (kind > key).

Масалан, key = 101.

Қидирув тўла жадвал бўйича эмас, балки low дан hi гача давом этади.



2-Чизма.

Индексли кетма-кет қидирув функцияси ва дастури.(C++да)

```
#include <iostream.h>
#include <conio.h>
int InSeqsearch(int realArray[], int N, int kind[2][1000],int m,int key, int *t)
{
  int i=0,
      low = 0,
      hi = 0;

  while ((i<m) && (kind[0][i]<=key))
        {
         i++;
         (*t)++;
        }
 (*t)++;

  if (i==0)
    low=0;
  else
    low=kind[1][i-1];

  if (i==m)
    hi=N;
  else
    hi=kind[1][i]-1;
```

```
  for (int j=low; j<=hi; j++)
  {
        (*t)++;
        if( key==realArray[j] )
        {
          return j;
        }
   }
   return -1;
}
main ()
{

int i = 0 ,
   N = 0,
   mas[1000] = {0},
   kind[2][1000] = {0},
   key = 0,
   P = 0,
   index = 0,
   kindIndex = 0,
   t = 0;

cout<<endl<<"Masiv uzunligini kiriting!"<<endl;
cin>>N;

cout<<"Massiv elementlarini kiriting!"<<endl;

for (i=0; i<N; i++)
  cin>>mas[i];

cout<<"Qidiruv elementini kiriting!"<<endl;
cin>>key;

cout<<"Boshlangich qadamni kiriting! "<<endl;
cin>>P;

i = P-1;
while(i<N)
{
 kind[0][kindIndex] = mas[i];
 kind[1][kindIndex++] = i;
 i += P;
}
```

```
  index = InSeqsearch(mas,N,kind,kindIndex,key, &t);

  if (index == -1)
    cout<<"Bunday element massivda yuq "<< index <<"  "<<t<<endl;
  else
    cout<<"Bunday element bor"<<" "<<index+1<<"  "<<t<<endl;

  getch();
  return 0;
}
```

3. Кетма-кет қидирувни самарадорлиги

Ихтиёрий қидирувнинг самарадорлиги жадвалдаги маълумотларнинг калитлари билан солиштириш сони – ***C*** билан бахоланиши мумкин. Агар таққослашлар (солиштириш) сони қанча кичик бўлса, қидирув алгоритми самарадорлиги шунча яхши бўлади.

Массивда кетма-кет қидирувнинг самарадорлиги қуйидагича бўлади:

$C = 1 \div n$, $C = (n + 1)/2$.

Умуман олганда рўйхатда хам самарадорлик юқоридаги каби бўлади. Гарчи массивда хам боғланган рўйхатда хам қидирув самарадорлиги бир хил бўлсада, маълумотларни массив ва рўйхат кўринишда тасвирлашнинг ўзига хос камчилик ва афзалликлари мавжуд. Қидирувнинг мақсади - қуйидаги жараёнларни бажарилишидан иборат:

1) Топилган ёзувни ўқиш.
2) Қидирилаётган ёзув топилмаса, уни жадвалга қўйиш.
3) Топилган ёзувни ўчириш.

Биринчи жараён (қидирувнинг ўзи) массив учун ҳам рўйхат учун ҳам бир хил бўлади. Иккинчи ва учинчи жараёнда эса қидирув рўйхатли тузилмада самараликроқ бўлади (сабаби массивда элементларн силжитиш лозим).

Агар k массивда элементларни силжитишлар сони бўлса, у холда $k = (n + 1)/2$ бўлади.

4. Индексли кетма-кет қидирувни самарадорлиги

Агар бўлиши мумкин барча холатлар тенг эхтимолли деб олинса, у ҳолда қидирув самарадорлигини қуйидагича хисоблаш мумкин:

Белгилашлар киритиб оламиз: m – индекс ўлчови; $m = n / p$; p – қадам ўлчови

$$Q = (m+1)/2 + (p+1)/2 = (n/p+1)/2 + (p+1)/2 = n/2p+p/2+1 \qquad (*)$$

Q ни p бўйича дифференциаллаб уни нолга тенглаштирамиз:

$$dQ/dp=(d/dp)\,(n/2p+p/2+1)= - n / 2\,p^2 + 1/2 = 0$$

Бу ердан

$$p^2=n\,; \quad p_{опт} = \sqrt{n}$$

(*) ифодада p ўрнига $p_{опт}$ ни қўйиб қуйидаги таққослашлар сонини оламиз:

$$Q = \sqrt{n} + 1$$

Демак, индексли кетма-кет қидирувни самарадорлиги тартиби ***O*** ($\sqrt{n}$) бўлади.

5. Қидирувни мукаммаллаштириш усуллари

Умуман олганда, жадвалда хар бир элементни қидириш эхтимоллигини қандайдир бир қиймат билан изоҳлаш мумкин. Фараз қилайлик жадвалда қидирилаётган элемент мавжуд. У ҳолда қидирув амалга оширилаётган барча жадвални дискрет холатга эга тизим сифатида қараш мумкин хамда унда қидирилаётган элементни топиш эхтимоллиги – бу тизим i-чи холати эхтимоллиги *p(i)* деб олиш мумкин.

$$\sum_{i=1}^{n} p(i) = 1$$

Жадвални дискрет тизим сифатида қараганимизда, ундаги таққослашлар сони дискрет тасодифий миқдорлар қийматларини математик кутилмасини ифодалайди.

$Z=Q=1p(1)+2p(2)+3p(3)+...+np(n)$

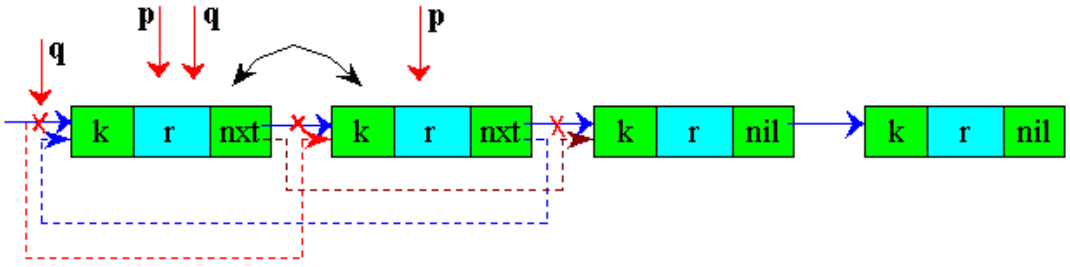
Иложи борича $p(1) \geq p(2) \geq p(3) \geq ... \geq p(n)$ бўлса, мақсадга мувофиқ бўлади.

Бу шарт таққослашлар сонини камайтириб, самарадорликни оширади. Сабаби, кетма-кет қидирув биринчи элементдан бошланганлиги учун энг кўп мурожаат қилинадиган элементни биринчига қўйиш лозим.

Қидирув жадвалини қайта тартиблашни энг кўп ишлатиладиган иккита усули мавжуд. Уларни бир боғламли рўйхатлар мисолида кўриб чиқамиз.

5.1 Топилган элементни рўйхат бошига қўйиш орқали жадвални қайта тартиблаш

Мазкур усулни мағзи шундан иборатки, берилган калитга тенг калитли элемент рўйхатда биринчи элемент деб ўзлаштирилади, қолганлари эса сурилади.



Келтирилган алгоритм рўйхат учун ҳам массив учун хам ўринли. Бироқ бу алгоритм массив учун тавсия қилинмайди, сабаби элементларни ўринлаштиришга кўрсаткичларни ўринлаштиришдан кўра анча кўп вақт талаб қилади.

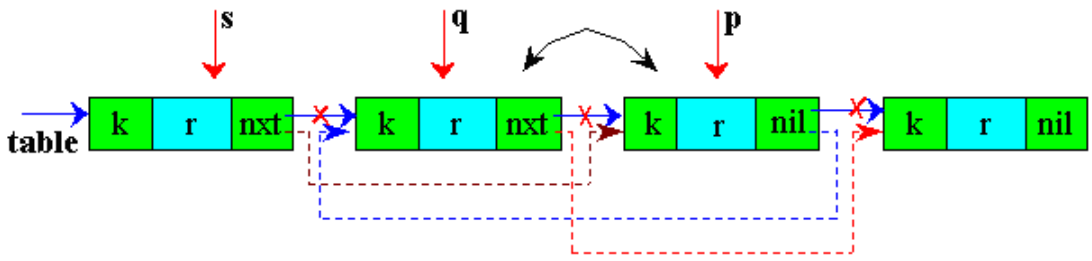
Рўйхатни қайта тартиблаш алгоритми:

Паскаль:

```
q:=nil;
p:=table;
while (p <> nil) do
 begin
   if key = p^.k then
     begin
       if q = nil
       then 'ўринлаштириш шарт эмас'
      search := p;
        exit;
        end;
      q^.nxt := p^.nxt;
      p^.nxt := table;
      table := p;
      exit;
     end;
   q := p;
   p := p^.nxt;
 end;
search := nil;
exit;
```

Транспозиция усули

Ушбу усулда топилган элемент рўхатда битта олдинги элемент билан ўрин алмаштирилади. Агарда мазкур элементга кўп мурожаат қилинса, биттадан олдинга сурилиб бориб натижада рўйхат бошида бўлади.



Чизма. Қўшни элементларни ўрнини алмаштириш

p – ишчи кўрсаткич
q – ёрдамчи кўрсаткич, p дан битта қадам орқада бўлади
s - ёрдамчи кўрсаткич, q дан иккита қадам орқада бўлади

Транспозиция усули алгоритми:

Паскаль:

```
s:=nil;
q:=nil;
p:=table;
while (p <> nil) do
```

```
 begin
   if key = p^.k then
     ‘транспонерлаймиз
     begin
        if q = nil then
          begin
          ‘ўринлаштирилмайди
          search:=p;
          exit;
        end;
        q^.nxt:=p^.nxt;
        p^.nxt:=q;
        if s = nil then
          table := p;
        else
           begin
              s^.nxt := p;
        end;
        search:=p;
        exit;
  end; end;
search:=nil; exit;
```

Ушбу усул нафақат рўйхатда, балки массивда хам қулай (сабаби фақатгина иккита ёнма-ён турган элемент ўрин алмаштирилади).

Мукаммал қидирув дарахти

Агар ажратиб олинган элементлар қандайдир ўзгармас тўпламни ташкил қилишса, келгуси қидирувлар самаралироқ бўлиши учун уларни бинар дарахт кўринишида ифодалаш мақсадга мувофиқ бўлиши мумкин.

Қуйида келтирилган дарахтларда бинар қидирувни кўриб чиқайлик (а)ва б) чизма). Иккала дарахт хам учтадан элементга эга - *к1, к2, к3* бўлиб бу ерда *к1<к2<к3*. *к3* элементни қидириш а) чизмада иккита таққослашни талаб қилса, б) чизмада эса битта.

Бирор бир ёзувни ажратиб олиш учун зарур бўлган калитларни таққослашлар сони бинар қидирув дарахтидаги ушбу ёзув босқичига бирни қўшганига тенг.

Фараз қилайлик:

қидирув аргументи *key = к1* бўлиши эхтимоллиги - *p1*,

қидирув аргументи *key = к2* бўлиши эхтимоллиги – *p3*,

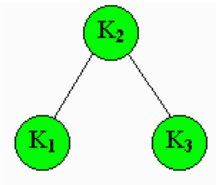
қидирув аргументи *key = к3* бўлиши эхтимоллиги – *p3*,

key < к1 бўлиши эхтимоллиги – *q0*,

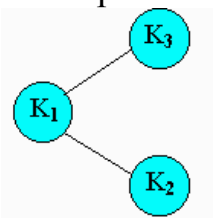
к2 > key > к1 бўлиши эхтимоллиги – *q1*,

к3 > key > к2 бўлиши эхтимоллиги – *q2*,

key > *к3* бўлиши эхтимоллиги – *q3*,
C1 - а) чизмадаги таққослашлар сони,
C2 - б) чизмадаги таққослашлар сони.



Чизма а)



Чизма б)

У ҳолда $\cdot p1+\cdot p2+\cdot p3+\cdot q0+\cdot q1+\cdot q2+\cdot q3 = 1$
Бирор бир қидирувда кутилаётган таққослашлар сони қуйидагича бўлади:

$$C1 = 2\cdot p1+1\cdot p2+2\cdot p3+2\cdot q0+2\cdot q1+2\cdot q2+2\cdot q3$$
$$C2 = 2\cdot p1+3\cdot p2+1\cdot p3+2\cdot q0+3\cdot q1+3\cdot q2+1\cdot q3$$

Бирор бир берилган калитлар ва эхтимолликлар тўпламида кутилаётган таққослашлар сонини минималлаштирувчи бинар қидирув дарахти мукаммал дейилади. Гарчи дарахт яратиш алгоритми анча сермашаққат иш бўлсада, бироқ у яратган дарахтда қидувни амалга ошириш анча самарали бўлади. Афсуски, кўпинча, қидирув аргументи эхтимоллиги олдиндан аниқ бўлмайди.

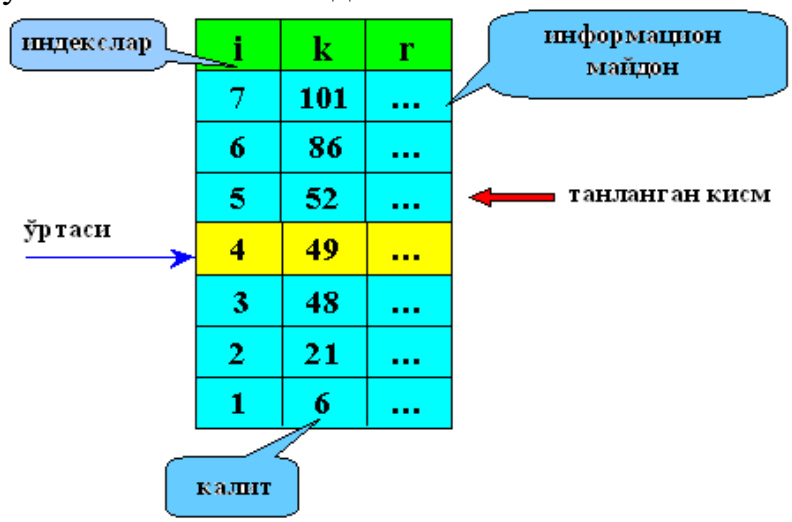
Бинар қидирув (тенг иккига бўлиш усули)

Фараз қилайлик, ўсиш тартибида тартибланган сонлар массиви берилган бўлсин. Ушбу усулни асосий ғояси шундан иборатки, тасодифий қандайдир A_M элемент олинади ва у X қидирув аргументи билан таққосланади. АгарA_M=X бўлса, у ҳолда қидирув якунланади; агар A_M <X бўлса, у ҳолда индекслари M дан кичик ёки тенг бўлган барча элементларни келгуси қидирувдан чиқариб юборилади. Худди шунингдек, агар A_M >X бўлса.

M ихтиёрий танланганда ҳам таклиф қилинаётган алгоритм коррект ишлайди. Шу сабабали M ни шундай танлаш лозимки, тадқиқ қилинаётган алгоритм самаралироқ натижа берсин, яъни уни шундай танлайликки, иложи борича келгуси жараёнларда иштирок этувчи элементлар сони кам бўлсин. Агар биз ўртача элементни, яъни массив ўртасини танласак ечим мукаммал бўлади.

Қуйидаги чизма кўринишида берилган массивни қараб чиқайлик. Фараз қилайлик, биздан калити 52 га тенг бўлган элементни топиш талаб қилинсин. Дастлабки таққосланадиган элемент 49 бўлади. 49<52 бўлгани учун сабабли, кейинги қидирув 49 дан юқорида турган элементлар орасида амалга оширилади. Янги хосил бўлган массив ўртаси 86. Агар берилган калит билан ушбу калитни таққосласак 86>52 бўлади. Демак, навбатдаги қидирувлар 86 билан 49 орасидаги элементлар ичида амалга оширилиши лозим. Кейинги қадамда маълум бўлдики, қаралаётган элементлар

ўртасидаги элемент калити 52 га тенг. Шундай қилиб, массивда берилган калитга тенг бўлган элементни топдик.



Юқоридаги алгоритмни амалга ошириш дастури C++ тилида қуйидагича бўлади:

```
#include <iostream.h>
#include <conio.h>
int Binsearch(int a[], int N, int key, int *t)
{
int l=0, r=N-1, mid=(l+r)/2;
        while (l<=r)
{ *t+=1;
if (a[mid]==key) return mid;
if (a[mid]>key) r=mid-1;
        else l=mid+1;
        mid=(l+r)/2;
}
a[N]=key;
return N;
}
main ()
{
int i, N, mas[1000], key, P, t=0;
cout<<endl<<"Masiv uzunligini kiriting!"<<endl;
cin>>N;
cout<<"Massiv elementlarini kiriting!"<<endl;
for (i=0; i<N; i++)
cin>>mas[i];
cout<<"Qidiruv elementini kiriting!"<<endl;
cin>>key;
P=Binsearch(mas,N,key, &t);
```

```
if (P==N) cout<<"Bunday elementni massivga qo'shis lozim"<<" "<<P+1<<" "<<t<<endl;
else cout<<"Bunday element bor"<<" "<<P+1<<"  "<<t<<endl;
getch();
return 0;
}
```

Агар key = 101 бўлса, керакли ёзув 3 марта таққослашларда аниқланади (кетма-кет қидирувда таққослашлар сони 7 та бўлар эди).

Агар C – таққослашлар сони ва n – жадвалдаги элементлар сони бўлса, у ҳолда

$C = log_2 n$.

Масалан, $n = 1024$.

Кетма-кет қидирувда $C = 512$, бинар қидирувда $C = 10$.

Агар катта хажмдаги маълумотлар ичида қидирув амалга оширилаётган бўлса, у ҳолда бинар ва индексли кетма-кет қидирувни умумлаштириб олиб бориш мумкин. Сабаби, ҳар иккала қидирув ҳам тартибланган массивда амалга оширилади.

Назорат саволлари

1. Қидирув вазифаси нимадан иборат?
2. Ноёб калит деганда нимани тушунасиз?
3. Рўйхатда берилган калитли элемент йўқ бўлганда қайси амал бажарилади?
4. Кетма-кет қидирув ва индексли кетма-кет қидирувларнинг фарқи нимадан иборат?
5. Улардан қайси бири самаралироқ ва нима сабабдан?
6. Жадвални қайта тартиблашнинг қандай усулларини биласиз?
7. Топилган элементни бошига қўйиш усулининг транспозиция усулидан асосий фарқлари нималардан иборат?
8. Мазкур усуллар маълумотлар қандай кўринишда берилганда тезроқ ишлайди?
9. Улар қандай рўйхатларда ишлайди, яъни тартибланган ёки ихтиёрий?
10. Бинар қидирувнинг мазмун ва моҳияти нимадан иборат?
11. Қандай қилиб бинар дарахтни четлаб ўтиш мумкин?
12. Бинар қидирувни массивда ишлатиш мумкинми?
13. Агар бўш бўлмаган дарахтда илдиз ўчириладиган бўлса, у ҳолда унинг ўрнига қайси элемент ўтади?

11-маъруза. Саралаш алгоритм ва усулларини тадқиқ қилиш

Режа:

1. Саралаш тушунчаси, унинг турлари.
2. Тўғридан тўғри қўшиш орқали саралаш.
3. Тўғридан тўғри танлаш орқали саралаш.
4. Тўғридан тўғри алмаштириш орқали саралаш.
5. Саралашнинг яхшиланган усуллари.

Калитли сўзлар: *саралаш, ички саралаш, ташқи саралаш, регулярлик, саралаш самарадорлиги, қўшиш орқали саралаш, танлаш орқали саралаш, алмаштириш орқали саралаш (пуфаксимон саралаш), тез саралаш, шелл саралаши..*

Саралаш – бу берилган тўплам элементларини бирор бир тартибда жойлаштириш жараёнидир. Саралашни мақсади тартибланган тўпламда керакли элементни топишни осонлаштиришдан иборат. Саралаш дастурларни трансляция қилинаётганда, маълумотлар мажмуасини ташқи хотирада ташкил қилинаётганда, кутубхоналар, каталоглар, маълумотлар базаси яратилаётганда тадбиқ қилинади. Маълумки, саралашнинг турли ҳил алгоритмлари мавжуд. Сабаби, битта масалани саралаш учун жуда кўплаб турли ҳил алгоритмлардан фойдаланиш мумкин. Берилган масалани ҳал қилишда баъзилари мукаммал бўлиши мумкин. Шунинг учун саралаш масаласида алгоритмларни қиёсий таҳлилини ўтказиш зарурати пайдо бўлади.

Саралаш масаласини қўйилишини қуйидагича ёзиш мумкин.

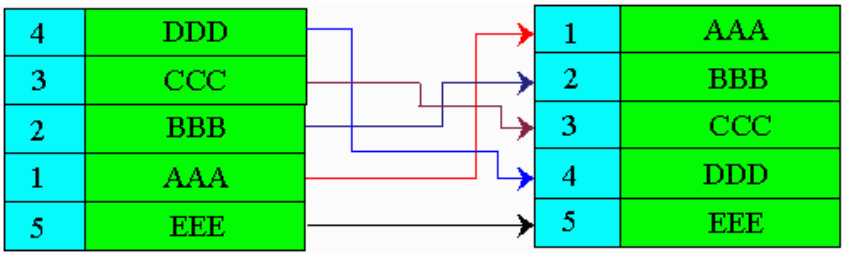
Фараз қилайлик, a_1, a_2 ,..., a_n, элементлар кетма-кетлиги берилган бўлсин. У ҳолда саралаш алгоритми элементларни массивга шундай жойлаштирадики, натижада улар қандайдир муносабатга нисбатан $f(a_{к1}) \leq f(a_{к2}) \leq \ldots \leq f(a_{кn})$ тартибга эга бўлади. Одатда f тартиблаш функцияси қандайдир махсус қоида билан ҳисобланмасдан, балки элементни калит қиймати бўйича массив элементлари тартибланади.

Маълумотларга қайта ишлов берилаётганда маълумотни информацион майдонини ҳамда уни машинда жойлашишини (адресини) билиш зарур.

Саралашни иккита тури мавжуд: ички ва ташқи:

- ички саралаш бу оператив хотирадаги саралаш;
- ташқи саралаш – ташқи хотирада саралаш.

Саралаш бу маълумотларни калитлари бўйича хотирада регуляр кўринишда жойлаштиришдир. Регулярлик деганда маълумотлар калит қийматлари бўйича массивда бошидан охиригача ўсиши ёки камайиши тушинилади.



Агар сараланаётган ёзувлар хотирада катта хажмни эгалласа, у ҳолда уларни алмаштиришлар катта сарф (вақт ва хотира маъносида) талаб қилади. Ушбу сарфни камайтиши мақсадида, саралаш калитлар адреси жадвалида амалга оширилади. Бунда фақатгина маълумот кўрсаткичлари алмаштирилиб, массив ўз жойида қолади. Юқоридаги усул адреслар жадвалини саралаш усули дейилади.

Сараланаётганда бир ҳил калитлар учраши мумкин, бу ҳолда сараланагандан кейин бир ҳил калитлилар бошланғич тартибда қандай жойлашган бўлса, ушбу тартибда қолдирилиши мақсадга мувофиқ бўлади (Бир ҳил калитлилар ўзларига нисбатан). Бундай усулга турғун саралаш дейилади.

Саралаш самарадорлигини бир неча мезонлар бўйича баҳолаш мумкин:

– саралашга кетган вақт;
– саралаш учун талаб қилинган оператив хотира;
– дастурни ишлаб чиқишга кетган вақт.

Биринчи мезонни қараб чиқайлик. Саралаш бажарилганда таққослашлар ёки алмаштиришлар сони ҳисоблаш мумкин.

Фараз қилайлик, $H = 0{,}01n^2 + 10n$ – таққослашлар сони. Агар $n < 1000$ бўлса, у ҳолда иккинчи қўшилувчи катта, акс ҳолда яъни, $n > 1000$ бўлса, биринчи қўшилувчи катта бўлади.

Демак, кичкина n ларда таққослашлар сони n га тенг бўлади, катта n ларда эса n^2 га тенг бўлади.

Саралашда таққослашлар сони қуйидаги оралиқларда бўлади:

$0(n \log n)$ дан 0 (n^2) гача; 0 (n) – идеал ҳолатда.

Саралашни қуйидагича усуллари бор:

– қатъий (тўғридан-тўғри) усуллар;
– яхшиланган усуллар.

Қатъий усуллар:

1. тўғридан-тўғри қўшиш усули;
2. тўғридан-тўғри танлаш усули;
3. тўғридан-тўғри алмаштириш усули.

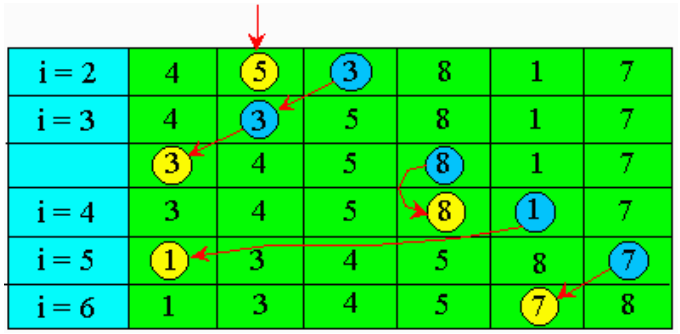
Юқорида келтирилган учала усулда ҳам алмаштиришлар сони деярли бир ҳил бўлади.

Тўғридан-тўғри қўшиш усули билан саралаш

Бундай усул карта ўйинида кенг қўлланилади. Элементлар (картлар) ҳаёлан “тайёр” a(1),...,a(i-1) ва бошланғич кетма-кетликларга бўлинади. Ҳар бир қадамда (i=2 дан бошланиб, ҳар бир қадамда бир бирликка ошириб борилади) бошланғич кетма-кетликдан i-чи элемент ажратиб олиниб тайёр кетма-кетликнинг керакли жойига қўшилади.

Таклиф қилинаётган усулни қуйидаги мисолда кўриб чиқамиз.

Фараз қилайлик, калит қиймати 4, 5, 3, 8, 1, 7 бўлган элементлар берилган бўлсин.



Керакли жойни қидириш жараёнини қуйидаги тартибда олиб бориш қулай бўлади. Таққослашлар амалга ошириш мобайнида, навбатдаги a(j) элемент билан солиштирилади, кейин эса x бўш жойга қўйилади ёки a(j) ўнга сурилади ва жараён чапга “кетади”. Шуни эътиборга олиш лозимки, саралаш жараёни қуйидаги шартларни бирортаси бажарилганда якунланади:

1. x элементи калитидан кичик калитли a(j) элемент топилди.
2. тайёр кетма-кетликнинг чап томони охирига етиб борилди.

Таклиф этилаётган усул алгоритми қуйидагича бўлади:

```
Procedure StraightInsertion
 Var
  i,j:index; x:item;
 begin
  for i:=2 to n do
    x:=a[i]; a[0]:=x; j:=1;
    while x<a[j-1] do a[j]:=a[j-1]; j:=j-1; end;
    a[j]:=x
    end
 end StraightInsertion
```

алгоритм самарадорлиги

Фараз қилайлик, таққослашлар сони C, ўринлаштиришлар сони M бўлсин. Агар массив элементлари камайиш тартибида бўлса, у ҳолда тақққослашлар сони энг катта бўлиб, у $C_{\max} = \frac{n(n-1)}{2}$ га тенг бўлади, яъни $O(n^2)$. Ўринлаштиришлар сони эса $M_{\max} = C_{\max} + 3(n-1)$ га тенг бўлади, яъни $O(n^2)$. Агар берилган массив ўсиш тартибида сараланган бўлса, у ҳолда

таққослашлар ва ўринлаштиришлар сони энг кичик бўлади, яъни $C_{\min} = n-1$, $M_{\min} = 3(n-1)$.

Тўғридан-тўғри танлаш усули билан саралаш

Фараз қилайлик, a_1, a_2, … , a_n элементлар кетма-кетлиги берилган бўлсин.

Мазкур усул қуйидаги тамойилларга асосланган:

1. Берилган элементлар ичидан энг кичик калитга эга элемент танланади.
2. Ушбу элемент бошланғич кетма-кетликдаги биринчи элемент a_1 билан ўрин алмашади.
3. Ундан кейин ушбу жараён қолган n-1 та элемент, n-2 та элемент ва хоказо, токи битта энг “катта” элемент қолгунча давом эттирилади.

Таклиф қилинаётган усул алгоритми қуйидагича бўлади:

Паскал тилидаги дастури:

```
Procedure StraightSelection
    Var
        i,j,k: index; x:item;
    begin
      for i:=1 to n-1 do
          k:=I; x:=a[i];
          for j:=i+1 to n do
           if a[j]<x then k:=j; x:=a[k]
    end;
    end;
    a[k]:=a[i];
    a[i]:=x
    end;
    end StraightSelection
```

Алгоритм самарадорлиги:

Таққослашлар сони $M = \frac{n}{2}(n-1) = \frac{n^2 - n}{2}$.

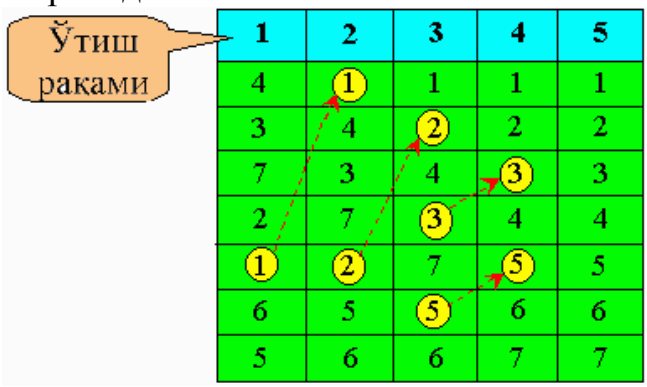
Алмаштиришлар сони $C_{min} = 3(n - 1)$, $C_{max} = 3(n - 1)\frac{n}{2}$

(n^2 тартиб).

Ушбу усул бўйича саралаш бажарилса, энг ёмон холда таққослашлар ва алмаштиришлар сони тартиби n^2 бўлади.

Тўғридан-тўғри алмаштириш усули билан саралаш (пуфаксимон)

Ушбу усулни ғояси қуйидагича: n - 1 марта массивда қуйидан юқорига қараб юриб калитлар жуфти-жуфти билан таққосланади. Агар пастки калит қиймати юқоридаги жуфти калитидан кичик бўлса, у ҳолда улар ўрни алмаштирилади.



Паскал тилидаги дастури:

```
for i := 2 to n do
  for j := n downto i do
    if a[j - 1] > a[j] then
      begin
        x := a[j - 1];
        a[j - 1] := a[j];
        a[j] := x;
      end;
```

Бизнинг холатда битта ўтиш “бекор” бўлди. Элементларни ортиқча ўринлаштирмаслик учун байроқча киритиш мумкин.

Пуфаксимон усулни яхшиланган усули бу шейкер саралаш усули бўлиб, ҳар бир ўтишдан кейин цикл ичида йўналиш ўзгартирилади.

Алгоритм самарадорлиги:

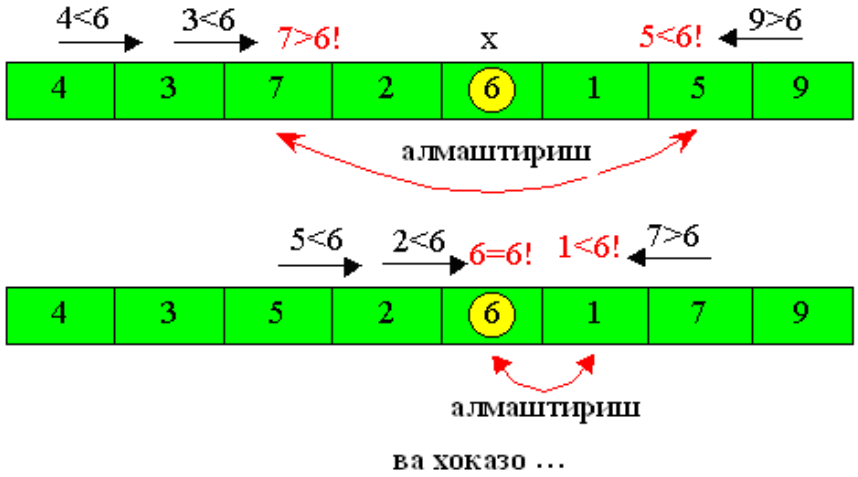
таққослашлар сони $M = \frac{n}{2} \cdot \frac{n}{2} = \frac{n^2}{4}$,

алмаштиришлар сони $C_{max} = 3\frac{n^2}{4}$.

Саралашнинг яхшиланган усуллари

Quiksort – тез саралаш усули

Ғояси: Бу усул алмаштириш усулидаги саралашга тегишли бўлиб унинг асосини калитларни танланган калитга нисбатан ажратиш ташкил қилади.



6 дан чап томонда калитлари кичик, ўнг томонда эса калитлари 6 дан катта бўлган элементлар жойлашади (юқоридаги чизма).

```
procedure Sort (L, R: integer);
begin
  i := L;
  j := r;
  x := a[(L + r) div 2];
  repeat
    while a[i] < x do
      i := i + 1;
    while a[j] > x do
      j := j - 1;
    if i <= j then
      begin
        y := a[i];
        a[i] := a[j];
        a[j] := y;
        i := i + 1;
        j := j - 1
      end;
  until i > j;
  if L < j then sort (L, j);
  if i < r then sort (i, r);
end;

procedure QuickSort;
begin
  sort (1, n);
end;
```

Алгоритм самарадорлиг:

O(n log n) – энг самарали усул.

Шелл саралаши (қисқариб борувчи қадамлар орқали саралаш)

Тўғри қўшиш усулини 1959 йилда Д. Шелл томонидан мукаммаллаштириш таклиф қилинган. Қуйидаги чизмада ушбу усул тасвирланган:

саралаш	44	55	12	42	94	18	6	67
кейин тўртлик	44	18	6	42	94	55	12	67
иккилик	6	18	12	42	44	55	94	67
яккалик	6	12	18	42	44	55	67	94

Бошида бир биридан 4 қадамда жойлашган элементлар ўзаро гуруҳланиб саралаш амалга оширилади. Бундай жараён тўртлик саралаш деб аталади. Биринчи ўтишдан кейин элементлар қайта гуруҳланиб, энди ҳар икки қадамдаги элементлар таққосланади. Бу эса иккилик саралаш деб номланади. Ва ниҳоят, учинчи ўтишда оддий ёки яккалик саралаши амалга оширилади.

Бир қарашда мазкур усул билан саралаш амалга оширилганда саралаш жараёни камайиш ўрнига ортиб борадигандек туюлсада, элементларни ўрин алмаштиришлар нисбатан кам амалга оширилади.

Кўриниб турибдики, бу усул натижасида тартибланган массив ҳосил бўлиб, ҳар бир ўтишдан кейин саралашлар камайиб боради. Энг ёмон ҳолатда охирги ишни яккалик саралаш амалга оширади.

Барьер усулидан фойдаланилганда ҳар бир саралаш ўзининг барьерига эга бўлиши лозим ҳамда дастур унинг жойини аниқлаши учун уни иложи борича осонлаштириш лозим. Шунинг учун массивни [-h1..N] гача кенгайтириш лозим бўлади.

h[1..t] – қадамлар ўлчами массиви
a[1..n] - сараланаётган массив
k – саралаш қадами
x – қўшилаётган элемент қиймати

Subroutine ShellSort

```
const   t = 3;
          h(1) = 5;
          h(2) = 3;
          h(3) = 1;
var
  h: array [1..t] of integer;
  a: array [1..n] of integer;
   k, x, m, t, i, j: integer;
 begin
   for  m = 1 to t do
```

```
    begin
      k:= h(m);
      for i = k + 1 to n do
        begin
         x:= a(i);
         for j = i-k  downto k   do
           begin
            if  x < a(j ) then
              a(j+k):= a(j);
             else
               goto 1;
            end ;
           end;
          end;
1:         a(j+1) = x ;
       end;
   end .
```

Умуман олганда, қандай қадамлар танланганда энг яхши натижа олиниши исботланмаган бўлсада, лекин бу қадамлар бири иккинчисини кўпайтувчилари бўлмаслиги лозимлиги аниқланган.

Д. Кнут қадамларни қуйидагича кетма-кетлигини таклиф қилган (тескари тартибда): 1,3,7,15,31,…,яъни: $h_{m-1}=2h_m+1$, $h_t=1$, $t = [\log_2 n] - 1$. Агар қадамлар ушбу кўринишда аниқланса, алгоритм самарадорлиги тартиби $\mathbf{O}(n^{1.2})$.

Назорат саволлари

1. Саралаш деганда нимани тушунасиз?
2. Саралашнинг асосий усулларини айтиб беринг.
3. Саралашнинг қайси усулари қатъий усулга тегишли?
4. Саралашнинг яхшиланган усулларини айтиб беринг.
5. Қандай саралаш турғун дейилади?
6. Тўғридан-тўғри қўшиш усули ғояси нимадан иборат?
7. Тўғридан-тўғри танлаш усули ғояси нимадан иборат?
8. Тўғридан-тўғри алмаштириш усули ғояси нимадан иборат?
9. Юқоридаги усулларнинг бир биридан фарқини айтиб беринг.
10. Қайси саралаш усули энг самарали бўлиб ҳисобланади?
11. Шелл усули қайси асосий саралаш усулига тегишли?

12-маъруза. Калитларни акслантириш (жойлаштириш)

Режа

1. **Калитларни акслантириш.**
2. **Акслантириш функциясини танлаш.**
3. **Зиддиятни ҳал қилиш алгоритмлари**

Калитли сўзлар: *хешлаштириш, хеш функция, калитларни акслантириш, зиддият, зиддиятни ҳал қилиш.*

Жойлаштириш усули (**хешлаштириш**) маълумотлар тузилмасида элемент жойлашган ўринни тез аниқлашга йўналтирилган усулдир. Жойлаштириш усулида маълумотлар оддий массив сифатида ифодаланган бўлади. Ушбу усулда маълумотлар калитлари *H* акслантириш ёрдамида массив индексларига акслантирилади. Шу сабабли мазкур акслантириш *«калитларни акслантириш»* деб номланади.

Элементни жадвалга қўшишдан олдин унинг адреси хеш-функция орқали аниқланади: *A = h(K)*, бу ерда K – калит, A – жадвалдаги элемент адреси бўлиб, $0 \le A \le N-1$, шарт ўринли бўлади.

Ушбу усулдан кўпинча дарахтларда ҳамда уни тадбиқ этиш мувоффақият келтирувчи масалаларда фойдаланилади.

Калитларни акслантириш билан боғлиқ бўлган асосий муаммо бу калитларни қабул қилиши мумкин бўлган қийматлар тўпламини хотира адреси (массив индекслари) қабул қилиши мумкин бўлган тўпламдан анча кенглигидир. Қуйидагича мисол кўриб чиқайлик. Фараз қилайлик, 1000 та одам бор. Ҳар бир одамни аниқлаш (идентификация қилиш) учун калит сифатида исмларини қарайлик. Фараз қилайлик ҳар бир исм 16 тагача ҳарфдан ташкил топган бўлсин. У ҳолда биз бўлиши мумкин бўлган калит қийматлари тўпламини (бу ерда бўлиши мумкин бўлган калитлар тўплами 26^{16} та элментдан иборат бўлади, агар алифбо 26 та харфдан иборат бўлса), 10^3 та индексли массивга акслантириш лозим бўлади. Юқоридаги мисолдан кўриниб турибдики, H функция ичига акслантирувчи акслантириш бўлади, яъни «кўпгина қиймат бир қийматга акслантирилади». Агар бирор бир калит берилган бўлса, у ҳолда қидирув амалининг биринчи қадами - калит билан боғланган индексни ҳисоблаш, яъни h = H(k), иккинчи ва асосийси – текшириш бўлиб ҳисобланади, яъни бунда ҳақиқатдан ҳам h функция T массивда (жадвалда) k калитли элементни идентификация қилаётгани текширилади.

Акслантириш функциясини танлаш

Ўз-ўзидан маълумки, акслантиришнинг ихтиёрий яхши функцияси калитларни берилган индекслар оралиғига иложи борича тенг тақсимлайди. Агарда ушбу талаб ўринли бўлса, у ҳолда қўшимча шарт ва чекланишлар бўлмайди. Агарда акслантириш мутлақо тасодифий бўлса яна ҳам яхшироқ

бўлади. Мазкур усулга “майдалаш” (хешлаштириш), яъни аргументни бўлаклаш деб аталади. H функция эса “жойлаштириш функцияси” деб аталади. Равшанки, H функция иложи борича самарали хисобланиши лозим, яъни унча кўп бўлмаган асосий арифметик амаллардан ташкил топган бўлиши лозим.

Фараз қилайлик, k калит тартиб рақамини барча мумкин бўлган калитлар тўпламида ифодаловчи ORD(k) функция берилган бўлсин. Бундан ташқари, i массив индекси 0 **...** N — 1 оралиқда жойлашган деб фараз қиламиз, бу ерда N — массив ўлчами. Бундай ҳолда “жойлаштириш функцияси” сифатида қуйидаги функцияни олиш мумкин:

H(k) = ORD(k) MOD N

Агарда “жойлаштириш функцияси” юқоридаги каби аниқланадиган бўлса, у ҳолда калит қийматлари индекслар ўзгариши оралиғига текис тақсимланади. Шу сабабли, калитларни акслантириш амалга оширилаётган вақтда кўпинча уни асос қилиб олишади. Бундан ташқари, агар масив узунлиги N иккининг қандайдир даражасига тенг бўлса, у ҳолда функция самарали ҳисобланади. Лекин, агар калит харфлар кетма-кетлигидан ташкил топган бўлса, у ҳолда айнан бундай функциядан воз кечишга тўғри келади. Сабаби, бундай ҳолатда барча калитлар тенгэхтимолликга эга деб қараш нотўғри бўлади. Натижада, фақатгина бир неча белгилар билан фарқланувчи сўзларни битта индексга акслантириш эхтимоллиги юқори бўлади, бу эса калитларни нотекис тақсимланишига олиб келиши мумкин. Шу сабабли, амалий масаллар ҳал қилинаётганда N сифатида туб сонларни олиш тавсия этилади.

Зиддиятни ҳал қилиш алгоритмлари

Агар берилган калитга мос жадвал қатори керакли (қидирилаётган) элементга эга эмаслиги маълум бўлса, у ҳолда зиддият (“конфликт”) юзага келди дейилади. Бундай ҳолат, агарда бир неча элемент битта индексга акслантириладиган калитларга эга бўлса юзага келади. Бундай ҳолатда мазкур берилган калит орқали тўлиқ аниқланувчи индекс бўйича иккинчи уриниш амалга оширилади (Муқобил индекс шакллантириш орқали). Иккинчи индексни шакллантиришнинг бир қанча усуллари мавжуд. Энг содда йўлларидан бири бу – биринчи H(k) индекслари бир ҳил бўлган барча қаторни бир-бирига боғлаш, яъни боғланган рўйхат каби. Бундай усулга тўғридан-тўғри боғлаш (direct chaining) деб аталади. Ҳосил бўлган рўйхат элементлари асосий жадвалда жойлашиши ҳам жойлашмаслиги ҳам мумкин.

Бундай ҳолатда рўйхат элементлари жойлашган хотира тўлалик (тўлиб-тошиш, переполнение) соҳаси дейилади. Ушбу усулни камчилиги, иккиламчи рўйхатларни кузатиб бориш ҳамда зиддиятга борувчи элементлар рўйхати ҳар бир қаторида мурожаат учун жой ажратиш лозим бўлади.

Зиддиятни ҳал қилишнинг яна бир усулида эса берилган жадвални берилган қаторида керакли элемент мавжуд бўлмаса, токи керакли элемент топилгунча ёки бўш қаторга боргунча бошқа қаторларини кўриб чиқилади. Агар қараб чиқиш бўш қаторгача бориб етса, у ҳолда кўрсатилган калит берилган жадвалда йўқ деб ҳисобланади. Зиддиятни бундай ҳал қилиш усулига очиқ адресли деб аталади. Табиийки, ихтиёрий берилган калит учун индекслар кетма-кетлиги иккинчи уринишда бир ҳил бўлиши лозим. Бундай ҳолатда қараб (кўриб) чиқиш алгоритми қуйидаги схемада ишлайди:

```
h = H(k)
i = 0
repeat
  if T(h)  =  k
        then  элемент топилди
        else if T(h) = free
                    then элемент жадвалда йўқ
                    else {зиддият}
                    i := i + 1
                    h := H(k) + G(i)
              endif
          endif
until  топилди, ёки жадвалда йўқ.
```

Назорат саволлари

1. Калитларни алмаштириш нима?
2. Акслантириш функцияси вазифаси нимадан иборат?
3. Қандай ҳолатларда зиддият юзага келади?
4. Зиддиятни ҳал қилишнинг қандай усулларини биласиз?

Фойдаланилган адабиётлар

1. ***Алфред В. Ахо., Джон Э. Хопкрофт, Джефри Д. Ульман.*** Структура данных и алгоритмы//Учеб.пос., М. : Изд.дом: "Вильямс", 2000, — 384 с.
2. ***Бакнелл Джулиан М.*** Фундаментальные алгоритмы и структуры данных в Delphi//СПб: ООО «ДиаСофтЮП», 2003. 560с.
3. ***Роберт Седжвик.*** Фундаментальные алгоритмы на C++. Анализ, Структуры данных, Сортировка, Поиск//К.: Изд. «ДиаСофт», 2001.- 688 с.
4. ***Динман М.И.*** C++. Освой на примерах//СПБ.:БХВ-Петербург, 2006, 384.
5. ***Шилдт, Герберт.*** Полный справочник по C#//М. : Изд. дом "Вильямс", 2004, 752 с.
6. ***Вирт Н.*** Алгоритмы и структуры программы//М., Мир, 1985.