O'zbekiston Respublikasi aloqa, axborotlashtirish va telekommunikatsiya texnologiyalari davlat qo'mitasi

Toshkent axborot texnologiyalari universiteti

Samarqand filiali

Axborot va pedagogik texnologiyalar fakulteti

Informatika va axborot texnologiyalari kafedrasi

KURS ISHI

Mavzu: Identifikatorlar jadvalini rexeshlashtirish asosida tashkillashtirish

Bajaruvchi: 301 guruh talabasi

_______________Miliev O.

Kurs ishi rahbari:

____ Kat. o'qit. Murodov O'.

Samarqand – 2013

Мавзу: Идентификаторлар жадвалини рехешлаштириш билан хеш-адреслаш асосида ташкиллаштириш

Режа: 1. Ишнинг мақсади.
2. Идентифкаторлар жадвалининг тавсифи.
3. Идентификаторлар жадвалини ташкиллаштириш принциплари.
4. Идентификаторлар жадвалини қуришнинг рехешаш билан хеш-адреслаш усули.
5. Ишни бажаришга қўйилган талаблар.
6. Топшириқнинг ечим босқичлари.
6.1. Рехешлаштириш билан хэш-функцияни танлаш ва тасвирлаш.
6.2. Идентификаторлар жадвалларида маълумотларнинг рехешлаштириш тузилмаларини тасвирлаш
6.3. Идентификаторлар жадвалини рехешлаштириш асосида ташкиллаштириш
7. Хулоса.
8. Фойдаланилган адабиётлар.

1. Ишнинг мақсади.

Ишнинг мақсади идентификаторлар жадвалини ташкиллаштиришнинг асосий усулларини ўрганиш, идентификаторлар жадвалини ташкиллаштиришнинг турли усуллари ва бу усулларнинг ютуқ ва камчиликлар ҳақида тасаввур ҳосил қилиш ҳисобланади.

Курс ишини бажариш учун киритишда идентификаторлар тўпламини оладиган, берилган усуллар ёрдамида идентификаторлар жадвалини ташкиллаштирадиган, жадвалларда хусусий идентификаторларни кўп бора излашга рухсат берадиган ва жадвални ташкиллаштиришнинг самарали усулларини солиштирадиган дастур тузиш талаб қилинади.

Идентификаторлар рўйхати матнли файл кўринишида берилган деб ҳисобланади. Идентификатор узунлиги 32 белги билан чегараланган.

2. Идентифкаторлар жадвалининг тавсифи.

Семантик анализ, код генерацияси ва натижавий дастурни оптималлаштиришни бажаришда компилятор жорий дастур асосий элементлари (ўзгарувчилар, ўзгармаслар, функциялар ва тилнинг бошқа лексик бирликлари) характеристикаларига боғланган бўлиши керак. Бу характеристикалар компиляторда кирувчи дастурнинг синтаксис анализи босқичларида олинади (кўпинча ўзгарувчи ва ўзгармасларни тасвирлашнинг блоклар структураси анализида) ҳамда код генерациясига тайёрланиш босқичида тўлдирилади (масалан, хотирани тақсимлашда).

Жорий дастур ҳар бир элементига мос тушадиган характеристикалар тўплами шу элемент типига, унинг маъносига (семантикасига) ва мос ҳолда жорий ва натижавий дастурлар бажарилиши ҳолига боғлиқ бўлади.

Ҳар бир аниқ ҳолда бу характеристикалар тўплами кирувчи тил синтаксиси ва семантикасига, бутун ҳисоблаш тизими архитектурасига боғлиқ бўлади. Аммо жорий дастур у ёки бу элементларига кўпинча мос тушадиган бир неча тур характеристикалар мавжуд. Масалан, ўзгарувчи учун – бу унинг тури ва хотира ячейкаси адреси, ўзгармас учун – унинг қиймати, функция учун – формал аргументлар сони ва турлари, қайтадиган натижалар турлари, функция кодини чақириш адреси. Жорий дастур характеристикалари ҳақида тўлиқ ахборотни, унинг анализини ва бажарилишини махсус адабиётлардан топиш мумкин.

Жорий дастур ихтиёрий элементининг бош характеристикаси унинг исми ҳисобланади.

Кирувчи тилнинг ўзгарувчилар, ўзгармаслар, функциялар ва бошқа элементлар исмлари билан дастур тузувчи шуғулланади – шунинг учун компилятор бу элементларни исмлари бўйича анализ қилиши керак. Ҳар бир элемент исми алоҳида кўринишга эга бўлиши керак. Кўпгина замонавий

дастурлаштириш тиллари жорий дастурнинг кўринувчанлик соҳаси ва бошқа шартларга боғлиқ ҳолда ўзгарувчилар ва функциялар исмларида ўзаро ўхшаш бўлишига рухсат беради. Бу ҳолатда исмлар ягоналигини компиляторнинг ўзи таъминланиши керак, бу ерда жорий дастурнинг элементлари исмлари ягона ҳисобланади.

Шундай қилиб, компилятор вазифаси жорий дастурнинг ҳар элементига боғланган қандайдир ахборотни сақлаш ва элемент исми бўйича бу ахборотга рухсатли кириш ҳисобланади. Бу масалани ечиш учун компилятор идентификатор идентификаторлар жадваллари ёки белгилар жадваллари деб аталадиган маълумотларнинг махсус сақланишини таъминлайди. Идентификаторлар жадвали жорий дастур битта элементига мос тушадиган маълумотлар (ёзувлар) майдони тўпламидан иборат бўлади. Ёзув компиляторда берилган элемент ҳақида барча керакли ахборотни беради ва компилятор иши давомида тўлдирилиб бориши мумкин. Ёзувлар сони идентификаторлар жадвалини ташкиллаштириш усулларига боғлиқ бўлади, аммо ҳар қандай ҳолатда жорий дастур элементларидан кам бўлади. Умуман олганда, компилятор бир эмас, балки бир нечта идентификаторлар жадваллари билан ишлайди – уларнинг сони ва структураси компилятор ишига боғлиқ бўлади [1,2].

3. Идентификаторлар жадвалини ташкиллаштириш принциплари.

Компилятор идентификаторлар жадвалига ёзувларни жорий дастур анализи ва жадвалда жойлашуви талаб қилинадиган янги элементларни топишига кўра тўлдириб боради. Жадвалда ахборотни излаш ҳар доим компиляторга дастурнинг у ёки бу элементи ҳақида маълумот керак бўлганда бажарилади. Шуни эслатиб ўтиш керакки, компилятор билан жадвалда элементни излаш унга янги элементларни қўшишдан кўра кўпроқ бажарилади. Бундай бўлишга сабаб, жорий дастурда янги элементларни ишлатиш уларни тасвирлаш қоидасига кўра кам учрайди. Бундан ташқари, ҳар қандай ҳолатда идентификаторлар жадвалига ҳар бир элементни

қўшишда – шу элементни жадвалда мавжуд эмаслигига амин бўлиш учун излаш операцияси қатнашади. Жадвалда элементни излашнинг ҳар бир операциясида жорий дастурда элементлар сони катта бўлганлиги (дастур ҳажмига боғлиқ ҳолда бирдан то юз минггача) учун вақт сарфланади, бу эса вақт компиляциясининг умумий вақтига таъсир этади. Шунинг учун идентификаторлар шундай ташкилланиши керакки, унда компилятор бу ёзув боғланган элемент исми бўйича жадвалдаги унга керакли ёзувни излашни максимал тез бажариш имконига эга бўлиши керак.

Идентификаторлар жадвалини ташкиллаштиришнинг қуйидаги усулларини кўрсатиш мумкин:

- оддий ва тартибланган рўйхатлар;
- бинар дарахт,
- рехэширлаштириш билан хэш-адреслаш;
- занжир усули бўйича хэш-адреслаш;
- рўйхат ёки бинар дарахт билан хэш-адреслаш комбинацияси.

4. Идентификаторлар жадвалини қуришнинг рехешлаш билан хеш-адреслаш усули.

Амалдаги жорий дастурларда идентификаторлар сони шунчалик кўпки, ҳатто уларнинг сонидан излаш вақтининг мантиқий боғлиқлигини қониқарли деб қабул қилиш мумкин эмас. Идентификаторлар жадвалида ахборотни излашнинг янада самарали усулларига талаб катта.

Агар хэш-функциялар ва хэш-адреслашларни қўллаш билан боғланган усуллар қўлланса, яхши натижаларга эришиши мумкин.

F хэш-функция деб $z : F(r) = n, r \in R, n \in Z$ бутун номанфий сонлар тўпламида қандайдир R кирувчи элементлар тўпламини тасвирлашга айтилади. “Хэш-функция” терминининг ўзи инглизча “hash function” (hash – “аралаштириш”) терминидан олинган.

R рухсат этилган элементлар тўплами хэш-функциянинг аниқланиш соҳаси дейилади. Хэш-функциянинг F қийматлар тўплами деб

$F: \forall rIR: F(r) \in M$ ва $\forall m \in M: \exists r \in R: F(r) = m$ функциядан қайтадиган барча мумкин бўлган қийматлардан иборат $Z: M \subseteq Z$ бутун номанфий сонлар тўпламидан М қисмий тўпламига айтилади. Қийматлар тўпламида хэш-функциянинг аниқланиш соҳасида тасвирланадиган жараён хэшлаштириш дейилади.

Идентификаторлар жадвали билан ишлашда хэш-функция бутун номанфий сонлар тўпламида идентификаторлар исмини тасвирлашни бажариши лозим. Хэш-функциянинг аниқланиш соҳаси барча мумкин бўлган идентификаторлар исмлари тўплами бўлади.

Хэш-адреслаш қандайдир маълумотлар массивидан ячейка адреси сифатида хэш-функциянинг қайтадиган қийматини қўллашдан иборат бўлади. У ҳолда маълумотлар массивининг ўлчами қўлланиладиган хэш-функция қийматларига соҳасига мос бўлиши керак. Бундан эса, реал компиляторда хэш-функция қийматлари соҳаси компьютернинг рухсат этилган адреслаш соҳаси ўлчамидан ошмаслиги керак.

Хэш-адреслашни қўллашга асосланган идентификаторлар жадвалини ташкиллаштириш усули шу элемент учун ҳисобланадиган хэш-функция қайтарадиган адрес ячейкасига жадвалнинг ҳар бир элементини жойлаштиришдан иборат. У ҳолда идеал ҳолатда идентификаторлар жадвалига ихтиёрий элементни жойлаштириш учун фақат унинг хэш-функциясини ҳисоблаши ва маълумотлар массивининг керакли ячейкасига мурожаат қилиши етарли. Жадвалда элементни излаш учун изланаётган элемент учун хэш-функцияни ҳисоблаш ва берилган массив ячейкаси бўш эканлигини текшириш зарур (агар у бўш бўлмаса – элемент топилади, бўш бўлса – элемент топилмайди). Биринчи марта идентификаторлар жадвали ахборот билан тўлдирилиши керак, қаердаки бунинг учун ячейкалар бўш деб хисобланади.

Бу усул самарали ҳисобланади, қаердаки жадвалга элементни жойлаштириш вақти каби, уни излаш вақти жадвал элементларини кўп бора таққослаш учун керак бўладиган кичик вақтга умумий ҳолда мос

тушмайдиган хэш-функцияни ҳисоблашга сарф қилинадиган вақт билан аниқланади.

Усул иккита мумкин бўлган камчиликларга эга. Улардан биринчиси – идентификаторлар жадвалидан хотира ҳажмини қўллаш носамарадорлиги: уни сақлаш учун массив ўлчами хэш-функция қийматларининг барча соҳаларига мос тушиши керак, бу вақтда идентификаторлар жадвалида сақланаётган мумкин қадар кичик бўлиши мумкин. Иккинчи камчилиги – хэш-функцияни мос тўғри танлаш зарурияти. Бу камчилик шунчалик аҳамиятлики, у хэш-адреслашни идентификаторлар жадвалини ташкиллаштириш учун воситаларсиз ишлатишга рухсат бермайди.

Хэш-функцияни танлаш муаммоси универсал ечимга эга эмас. Хэшлаштириш, одатда баъзи бир оддий арифметик ва мантиқий операцияларни белгилар занжирида бажариш ҳисобидан амалга оширилади. Белги учун энг оддий хэш-функция компьютерда белгилар литерининг ички тасвирланиш коди ҳисобланади. Бу хэш-функцияни занжирда биринчи белгини танлаб, белгилар занжири учун ҳам ишлатиш мумкин.

Кўриниб турибдики, бундай примитив хэш-функция қониқарсиз: уни қўллашда муаммо пайдо бўлади – битта ва шу ҳарф билан бошланадиган иккита турли идентификаторларга хэш-функциянинг битта ва шу қиймати мос келади. У ҳолда идентификаторлар жадвалининг битта ва шу ячейкасини хэш-адреслашда иккита турли идентификаторларга жойлаштириш керак, бу эса мумкин эмас. Иккита ва ундан кўп идентификаторларга хэш-функциянинг битта ва фақат шу қиймати мос келган ҳолат коллизия дейилади.

Табиийки, коллизияга рухсат берадиган хэш-функцияни идентификаторлар жадвалида хэш-адреслаш учун ишлатиш мумкин эмас. Барча идентификаторлар жадвалида битта коллизия ҳолатига дуч келиш, бундай хэш-функцияни қўлламаслик учун етарли.

Коллизияга дуч келмайдиган хэш-функцияни қуриш имконияти мавжудми?

Коллизияга тўла дуч келмаслик учун хэш-функция бир қийматли бўлиши керак: хэш-функция аниқланиш соҳасининг ҳар бир элементига унинг қийматлари тўпламининг битта қиймати мос келиши керак ва аксинча – бу бу функция қийматлари тўпламининг ҳар бир қийматига унинг аниқланиш соҳасининг фақат битта элементи мос тушиши керак. У ҳолда хэш-функция аниқланиш соҳасининг ихтиёрий иккита хусусий элементига ҳар доим унинг иккита қийматлари мос келади. Назарий томондан идентификаторлар учун бундай хэш-функцияни қуриш мумкин, қаердаки, хэш-функциянинг аниқланиш соҳаси (идентификаторларнинг барча мумкин бўлган исмлари) ва унинг қийматлари соҳаси (бутун номанфий сонлар) чексиз саналадиган тўплам ҳисобланади, шунинг учун бир тўпламни бошқасига мос ҳолда бир қийматли тасвирлашни ташкиллаштириш мумкин.

Идентификаторлар учун мос ҳолда бир қийматли хэш-функцияни яратиш мумкин эмас, аммо амалиётда чекловлар мавжуд, Гап шундаки, ҳақиқатда ихтиёрий хэш-функция қийматлари соҳаси компьютернинг рухсат этилган адреслаш соҳаси ўлчами билан чегараланган. Берилган архитектурали ихтиёрий компьютернинг адреслар тўплами катта бўлиши мумкин, бундан эса у чегараланган ҳамдир.

Аммо амалиётда чекловлар мавжуд, идентификаторлар учун мос ҳолда бир қийматли хэш-функцияни яратиш мумкин эмас. Гап шундаки, ҳақиқатда ихтиёрий хэш-функция қийматлари соҳаси компьютернинг рухсат этилган адреслаш соҳаси ўлчами билан чегараланган. Берилган архитектурали ихтиёрий компьютернинг адреслаш тўрлами катта бўлиши мумкин, бундан эса у чегараланган ҳамдир.

Элементни жойлаштириш ва излаш алгоритмлари бажариладиган операциялар бўйича ўхшаш. Шунинг учун улар ечиш керак бўладиган бир хил вақтга эга бўлади.

Коллизия пайдо бўлганда идентификаторлар жадвалини ташкиллаштиришда алгоритм элементларни аниқ кўринишда танлаб, жадвалнинг бўш ячейкаларига жойлаштиради. Бу ҳолда элементлар хэш-функция қийматлари билан мос адресли ячейкаларга тушиши мумкин, бу эса янги қўшимча коллизияларнинг пайдо бўлишига олиб келади. Шундай қилиб, жадвалда элементларни излаш ва жойлаштириш учун зарур операциялар сони жадвалнинг тўлдирилганлигига боғлиқ бўлади.

5. Ишни бажаришга қўйилган талаблар.

Курс иши қуйидаги тартибда бажарилиши керак:

1. Талаба ўқитувчидан масала бўйича топшириқ олинади;
2. Идентификаторлар жадвалини ташкиллаштиришда берилган усуллар учун қўлланиладиган маълумотлар структурасини қўллаш амалга оширилади;
3. Ҳисобот тайёрланади ва ҳимоя қилинади;

Топшириқ вариантлари

1-жадвалда масалада қўлланиладиган идентификаторлар жадвалини ташкиллаштириш усуллари берилган.

1-жадвал. Идентификаторлар жадвалини ташкиллаштириш усуллари

Методлар №	**Коллизияларга рухсат бериш усуллари**
1	Оддий рехеширлаш
2	Тасодифий сонларни ишлатиш билан рехеширлаш
3	Кўпайтириш ёрдамида рехеширлаш
4	Занжир усули
5	Оддий рўйхат

6	Тартибланган рўйхат
7	Бинар дарахт

2-жадвалда 1-жадвалда берилган идентификаторлар жадвалини ташкиллаштириш усуллари асосида топшириқлар варианти варианти берилган.

2-жадвал. Масалалар вариантлари

Вариант №	**Жадвални ташкиллаштириш-нинг биринчи усули**	**Жадвални ташкиллаштириш-нинг иккинчи усули**
1	1	1

6. Топшириқнинг ечим босқичлари.

6.1. Рехешлаштириш билан хэш-функцияни танлаш ва тасвирлаш

Хэш-функция сифатида хэш-адреслаш учун киритишда қаторни берадиган функцияни оламиз ва натижада қаторнинг биринчи, ўртадаги ва охирги элементлари кодларининг йиғиндисини беради. Агар қатор уч белгидан кам бўлмаса, у ҳолда битта белги биринчи, ўртадаги, охирги белги сифатида олинади.

Идентификаторда бош ва кичик ҳарфларни турлича деб ҳисоблаймиз. Белгилар коди сифатида Microsoft Windows туридаги ОС базасидаги ҳисоблаш тизимларида қўлланиладиган ASCII жадваллари кодларини оламиз. У ҳолда, агар хэш-функцияни аниқланиш соҳасида қатор инглиз тилининг рақамлари ва ҳарфларидан иборат бўлса, у ҳолда хэш-функциянинг минимал қиймати “0” рақами учта кодининг йиғиндиси, максимал қиймати эса “z” литерасининг учта учта кодининг йиғиндиси бўлади.

Шундай қилиб, Object Pascal тилида танланган хэш-функцияни қийматлари соҳаси қуйидагича ёзилади:

(Ord('0')+ Ord('0')+ Ord('0'))..(Ord('z')+ Ord('z')+ Ord('z'))

қийматлари соҳаси диапазони масала шартларини қаноатлантирадиган 223 элементдан иборат (200 элементдан кам бўлмаган). Берилган ҳолатда киритилувчи элементлар узунлиги чегараланмаган. Қулайлик учун, хэш-функция қийматлари соҳасининг чегараларини берувчи иккита ўзгармасни ёзамиз:

HASH_MIN = Ord('0')+ Ord('0')+ Ord('0');

HASH_ MAX = Ord('z')+ Ord('z')+ Ord('z').

Хэш-функциянинг ўзи рехэширлашни хисобга олмасдан қуйидагиларни ҳисоблайди:

Ord(sName[1] + Ord(sName[(Length(sName)+1) div 2]) + Ord(sName[Length(sName)],

бу ерда *sName* – бу киритилувчи қатор.

Рехэширлаш учун $F = i \cdot H_1 mod H_2$, қаерда H_1 ва H_2 - H_1 $H_2/2$ дан H_2 гача диапазонда танланган оддий сонлар, формула асосида қурилган псевдотасодиф сонлар кетма-кетлигининг оддий генераторини оламиз. Бу генератор *HASH_MIN* дан *HASH_ MAX* гача барча диапазонда кетма-кетликнинг максимал узунлигини беради, H_2 *HASH_MAX-HASH_MIN* +1 га максимал яқин бўлиши керак. Берилган ҳолда диапазон 223 элементдан иборат, бунда 223 – оддий сон, у ҳолда H_2 = 223 деб оламиз (агар диапазон ўлчами оддий сон бўлмаса, у ҳолда H_2 сифатида унга яқин оддий сон олинади). H_1 сифатида 127 ни оламиз: $H_1 = 127$.

Мос ўзгармаслари ёзамиз:

REHASH1 = 127;

REHASH2 = 223;

У ҳолда рехэширлашни ҳисобга олиш билан хэш-функция қуйидаги кўринишга эга бўлади:

```
   function VarHash(const sName:string: iNum:integer):longint:
begin
Result:=(Ord(sName[l])+Ord(sName[(Length(sName)+l) div 2]) +
```

*Ord(sName[Length(sName)]) - HASH_MIN + iNum*REHASHl*

mod REHASH2) mod (HASH_MAX-HASH_MIN+1) + HASH_MIN;

if Result < HASH_MIN then

Result := HASH_MIN;

end;

Бу функциянинг киритилувчи параметрлари: *sName* – хэшлаштирувчи идентификатор исми, *iNum* – рехэширлаш индекси (*iNum=0* бўлса, у ҳолда рехэширлаш мавжуд эмас). Натижа узунлигини текшириш қатори (*Result* < *HASH_MIN*) '0'..'z' диапазондан ташқарида ётган белгилардан иборат қаторни функцияга киритишда хатолар келиб чиқишининг олдини олиш учун қўшилганлар.

Хэш-адреслаш ва бинар дарахт комбинацияси учун бир мунча оддий хэш-функцияни – киритилувчи қаторнинг биринчи ва оралиқ кодлари йиғиндисини қўллаш мумкин. Object Pascal тилида бундай хэш-функциянинг қийматлари диапазони қуйидагича бўлади:

(Ord('0')+ Ord('0').. Ord('z')+ Ord('z'))

Бу диапазон 200 дан кам бўлмаган элементдан иборат, бунда функция масала шартларини қаноатлантириши керак, у бинар дарахт Билан комбинацияда идентификаторларни чегараланмаган қайта ишлашни таъминлайди (идентификаторларнинг максимал сони фақат компьютер оператив ҳажми билан чегараланган).

Бу хэш-функция рехэширлашни қўлламаган ҳолда юқоридагига кўра соддароқ кўринишга эга бўлади:

function VarHash(const sName: string): longin;

begin

Result:=(Ord(sName[1])+Ord(sName[(Length(sName)+1) div 2])

- HASH_MIN) mod (HASH_MAX-HASH_MIN+1) + HASH_MIN:

if Result < HASH_MIN then Result : = HASH_MIN;

end.

6.2. Идентификаторлар жадвалларида маълумотларнинг рехешлаштириш тузилмаларини тасвирлаш

Биринчи навбатда идентификаторлар жадвалида идентификаторлар ҳақида ахборотларни сақлаш учун қўлланиладиган маълумотлар структураларини тасвирлаш керак. Иккита жадвал учун (псевдотасодиф сонлар асосида хэш-адреслаш) битта структурани қўллаймиз. Бу ҳолда жадвалларда қўлланилмайдиган маълумотлар сақланади, аммо дастур коди мустасно. Ўқув мисоли учун бу ёндашиш исботланган.

Идентификаторлар жадвалларининг маълумотлари структураси (уни *TvarInfo* деб атаймиз) идентификаторлар исмлари майдонини (*sName:string* майдони) ҳамда идентификаторлар ҳақида қўшимча ахборот майдонидан иборат бўлиши керак. Лаборатория ишида идентификаторлар ҳақида қандайдир қўшимча ахборотни сақлаш кўриб ўтилмаган, шунинг учун ахборот майдонини кўрсатиш сифатида *TvarInfo* структурасига қўшимча *TAddVarInfo* (*pInfo: TaddVarInfo* майдони) ахборот сруктурасини киритамиз.

Object Pascal тилида *class* сифатида тасвирланган майдонлар ва ўзгарувчилар учун мос структурага фақат ҳаволалар сақланганлиги учун бундай ёндашув хотирани тўғри ишлатишга олиб келмайди, аммо кейинчалик алоҳида маълумотлар структурасида идентификаторлар билан боғланган ихтиёрий ахборотни сақлайди. Берилган ҳолда бошқача ёндашиши мумкин эмас, чунки олдиндан идентификаторлар жадвалида қандай маълумот сақланиши маълум эмас. Компилятор яратувчиси, қоидага кўра, қандай ахборотни сақлаш зарурлигини беради ва бошқача ёндашувни қўллаши мумкин – оралиқдаги маълумотлар структураларини ва ҳаволаларни ишлатмасдан, идентификаторлар жадваллари маълумотлар структурасида барча керакли майдонларни киритиши мумкин.

Биринчи ёндашув, берилган мисолда ишлатилган, оператив хотирани тежамли ишлатишни таъминлайди, аммо мураккаб бўлиб, динамик структура билан ишлашни талаб этади, иккинчи ёндашув ишлатиш учун оддий, аммо хотирани кўпроқ эгаллайди. Икки ёндашувдан қайси бирини танлашни аниқ ҳолатда компилятор яратувчисининг ўзи танлайди.

TvarInfo маълумотлар структураси билан ишлаш учун қуйидаги функциялар керак:

- *constructor Create и destructor Destroy* каби ишлатиладиган маълумотлар структурасини яратиш ва банд қилинган хотирани бўшатиш функциялари;
- қўшимча ахборотга рухсатли кириш функциялари – бу ерда *procedure Setlnfo* ва *procedure CI ear Info* ишлатилади;

Бу функциялар рехэширлаш билан идентификаторлар жадвали учун ва идентификаторлар жадвалини қурилиши учун умумий ҳисобланади.

Қолаверса, *TvarInfo* маълумотлар структурасида идентификаторлар жадвалини қурилиши учун бинар дарахтни ташкиллаштиришни таъминлайдиган маълумотлар ва функцияларнинг қўшимча майдонларини қўшиш керак:

- дарахтнинг чап (“кичик”) ва ўнг (“катта”) шохларига ҳавола - *minE1, maxE1: TvarInfo* маълумотлар майдони каби ишлатилади;
- дарахтга элементларни қўшиш функциялари - *function AddElCnt* ва *function AddElem;*
- дарахтда элементни излаш функциялари - *function FindElCnt* ва *function FindElem;*
- дарахтда ахборот майдонини тозалаш функциялари - *procedure CIearAl1 Info;*
- битта қаторга бинар дарахт таркибини чиқариш функцияси (барча идентификаторлар рўйхатини олиш учун) - *function GetElList.*

Дарахтда элементни излаш ва жойлаштириш функциялари икки экземплярда ишлаб чиқилган, улардан бири таққослашлар сонини ҳисоблайди, иккинчиси эса ҳисобламайди.

(function TVarlnfo.FindElCnt) дарахтда идентификаторларни излаш функциясини ишлатишда бита асосий жойга эътибор бериш керак. Агар Object Pascal тилида қаторларни таққослашнинг стандарт усуллари ёрдамида иккита қаторни таққослаш бажарилса (берилган ҳолда – *sN* изланаётган идентификатор исми ва *sName* дарахтнинг жорий тугунидаги идентификатор исми), у ҳолда дастур коди қуйидаги кўринишга эга бўлади:

```
if sN < sName then begin
…
end
else
if sN > sName then
begin
…
end
else ...
```

Бу фрагментда қаторларни таққослаш икки марта бажарилади: аввал “кичик” *(sN<sName)* муносабат, кейин эса “катта” *(sN>sName)* муносабат текширилади.

Дастур кодида аниқ кўрсатилмаган бўлсада, бу операторларнинг ғар бири учун қаторларни таққослаш функциялари чақирилади (таққослаш операцияси икки марта бажарилади!). Берилган мисолни бажаришда, бунинг олдини олиш учун, қаторларни таққослаш функцияларини очиқ чақириш бажарилади, кейин эса олинган натижа қайта ишланади:

```
i : =  StrComp(PChar(sN), PChar(sName));
if i < 0 then
begin
…
```

```
end
else
if i > 0 then
begin
…
end else ....
```

Бу вариантда бутун сони нол билан таққослаш икки марта бажарилади, қаторларни таққослаш эса фақат бир марта бажарилади, бу эса излаш процедурасининг самарадорлигини оширади.

6.3. Идентификаторлар жадвалини рехешлаштириш асосида ташкиллаштириш

Идентификаторлар жадваллари элементлари *TvarInfo* турли маълумотлар структураси ҳисобланадиган *HASH_MIN...HASH_MAX* ўлчамли статик массивлар кўринишида ташкиллаштирилган. Object Pascal тилида, юқорида айтилганидек, бундай турли структуралар учун ҳаволалар сақланади. Шунинг учун идентификаторлар жадвалида бўш ячейкаларни тасвирлаш учун *nil* – бўш ҳавола ишлатилади.

Хотирада ҳаволалар сақланганлиги учун, тасвирланган массивлар ҳаволалари идентификаторлар жадвалидаги ахборотга воситасиз кўрсатадиган хэш-жадвал ролини ўйнайди.

1.3-расмда идентификаторлар жадвалини ташкиллаштиришни аниқ кўрсатадиган шартли схемалар берилган. 1 схема псевдотасодиф сонлар генератори асосида рехэширлаш билан идентификаторлар жадвали, 2 схема бинар дарахт билан хэш-адреслаш комбинацияси асосида идентификаторлар жадвалини намойиш этади. «*nil*» ёзувли ячейкалар хэш-жадвалнинг тўлдирилмаган ячейкаларига мос келади.

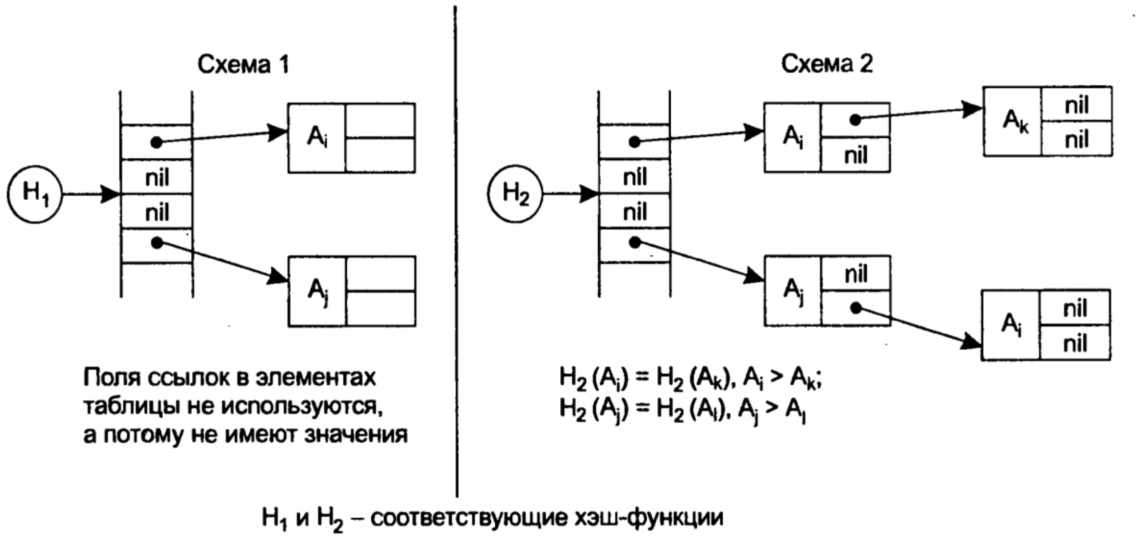

Рис. 1.3. Схемы организации таблиц идентификаторов

1.3-расм. Идентификаторлар жадвалини ташкиллаштириш схемаси.

Ҳар бир идентификаторлар жадвали учун қуйидаги функциялар ишлаб чиқилган:

- \- хэш-жадвални бошланғич инициализация қилиш функцияси - *InitTreeVar* ва *InitHashVar;*
- хэш-жадвалнинг хотирасини бўшатиш функцияси — *ClearTreeVar* ва *ClearHashVar;*
- жадвалдаги қўшимча ахборотни ўчириш функцияси — *ClearTreelnfo* ва *ClearHashlnfo;*
- идентификаторлар жадвалига элементни қўшиш функцияси — *AddTreeVar* и *Add-HashVar;*
- идентификаторлар жадвалида элементни излаш функцияси — *GetTreeVar* и *GetHashVar;*
- жадвалда элементни қўшиш ёки излашда бажариладиган таққослаш операциялари сонини қайтарувчи функциялар — *GetTreeCount* и *GetHashCount.*

Иккита берилган жадвалларни ташкиллаштириш усуллари учун идентификаторларни излаш ва жойлаштириш алгоритмлари “Қисқача назарий маълумотлар” бўлимида берилган, шунинг учун уларни қайтадан ёритиб ўтирмаймиз. Улар юқорида кўриб ўтилган тўртта функциялар кўринишида ишлаб чиқилган (*AddTreeVar* ва *AddHashVar* – элементни жойлаштириш учун, *GetTreeVar* ва *GetHashVar* — элементни излаш учун). Жадвалда элементларни излаш ва жойлаштириш функциялари аъло бажарилган ҳолларда жадвалнинг элементига натижа сифатида ҳаволани қайтаради (*TbElem* модулида ёритилган структурага кўра), акс ҳолда бўш ҳаволани қайтаради.

Шуни эслатиб ўтиш кераккки, жадвлга идентификаторларни жойлаштириш функциялари шундай ташкиллаштирилганки, агар жадвалга янги идентификаторни жойлаштириш вақтида шу исмли идентификатор бўлса, у ҳолда функция жадвалга янги идентификаторни қўшмайди, балки ҳаволани жадвалга яқинда қўшилган идентификаторга натижа сифатида қайтаради. Шундай қилиб, жадвалда бир хил исмли икки ёки ундан кўп идентификаторлар бўлиши мумкин эмас. Бундан эса кирувчи файлдаги бир хил идентификаторлар сони хато сифатида қабул қилинмайди – чунки масаланинг берилишида бир хил исмли идентификаторлар сонига чеклов қўйилмаган.

Барча санаб ўтилган функциялар дастур модулида тасвирланган: *FncHash* — псевдотасодиф сонлар генераторини ишлатиш билан рехэширлаштириш асосида қурилган идентификаторлар жадвали учун ва *FncTree* — хэш-адреслаш ва бинар дарахт комбинацияси асосида қурилган идентификаторлар жадвали учун. Функциялар ва идентификаторлар жадвалларини ташкиллаштириш учун берилган массивлардан ташқари бу модуллар жадвалда идентификаторларни жойлаштириш ва излашда таққослаш операцияларини бажариш сонини ҳисоблаш учун ишлатиладиган ўзгарувчиларни ҳам ўз ичига олади.

Шуни эътиборга олиш керакки, иккала модулнинг инициализация (*initialization*) бўлимида идентикаторлар жадвалини бошланғич тўлдириш функциясини, иккала модулнинг тугатиш (*finalization*) бўлимида эса хотирани бўшатиш функциялари чақирилади. Бу бошқа функцияларни чақириш тартибида модулларни аниқ ишлашини таъминлайди, бунда Object Pascal нинг ўзи модулларни инициализация ва тугатиш бўлимларида дастур кодини вақтинча чақиришни таъминлайди.

6.4. Дастур матни.

Кўрсатиб ўтилган модуллардан ташқари, фойдаланувчи билан интерфейсни таъминлайдиган модул ҳам керак бўлади. Бу модул (*FormLab1*) *VCL* кутубхонада *TForm* синф асосида *TLabForm* график ойнани ишлаб чиқаради. У берилган ОС тизимли кутубхоналаридан бошқарувнинг стандарт аъзолари асосида Windows типли ОС да Graphical User Interface (GUI) воситалари билан интерфейсни таъминлайди. Дастур кодидан (FormLabl.pas файли) ташқари модул ўзига фойдаланувчи интерфейси ресурсларини (FormLabl.dfm файли) тасвирлайди. GUI асосида фойдаланувчи интерфейсини ташкиллаштириш ва интерфейс ресурслари билан дастурлаштириш тизимларида ишлаш принциплари [3, 5,6,7] да келтирилган.
Интерфейснинг шакллари ва унинг аъзоларини бошқаришдан ташқари FormLabl модули жадвалда идентификаторларни жойлаштириш ва излашдан олинадиган статистик натижаларни йиғиш учун хизмат қмладиган учта (iCountNum, iCountHash, iCountTree) ўзгаручидан ҳамда тўпланган статистик ахборотни экранда тасвирлайдиган (procedure ViewStatistic) функциядан ташкил топади.
Модулда тасвирланадиган интерфейс шакли қуйидаги асосий бошқарув аъзоларига эга:

- файл исмини киритиш майдони *(EditFile),* файллар тизимидаги каталогдан файл исмини танлаш тугмаси *(BtnFile),* файлни ўқиш тугмаси *(BtnLoad);*
- ўқилган файлни тасвирлаш учун кўпсатрли майдон *(Listldents)*;
- изланаётган идентификатор исмини киритиш майдони *(EditSearch);*
- киритилган идентификаторни излаш учун тугма *(BtnSearch)* — бу тугма Билан бир марта излаш процедураси чақирилади *(procedure SearchStr);*
- барча идентификаторларни автоматик излаш тугмаси *(BtnAl1Search)* — бу тугма билан идентификаторни излаш процедураси *(procedure SearchStr)* файлдан барча ўқилаган идентификаторлар учун циклик равишда чақирилади (*Listldents* майдонида санаб ўтилганлар учун*);*
- тўпланган статистик ахборотларни жўнатиш тугмаси *(BtnReset);*
- статистик ахборотларни тасвирлаш учун майдон;
- дастур ишини тугатиш тугмаси (BtnExit).

Идентификаторли файлни ўқиш функцияси *(procedure TLablForm. BtnLoadClick) BtnLoad* тугмасини босиш билан чақирилади. У қуйидагича ташкиллаштирилган, аввал файл *Listldents* кўп сатрли майдонида ўқилади, кейин эса барча ўқилган идентификаторлар идентификаторларнинг иккита жадвалига ёзилади. Файлнинг ҳар сатри алоҳида идентификаторлар билан ўқилади, сатрнинг боши ва охиридаги пробеллар ташлаб юборилади. Жадваллардан бирида идентификаторларни жойлаштиришда хатолик бўлса, огоҳлантирувчи хабар чиқарилади (масалан, агар 223 дан кўп турли идентификаторлар ўқилса, у ҳолда рехэшлаштириш мумкин бўлмайди ва хато ҳақида хабар чиқарилади).

Идентификаторларни излаш функцияси *(procedure TLablForm.SearchStr) BtnSearch* (*procedure TLablForm.BtnSearchClick* процедураси) тугмасини бир

марта босиш ёки *BtnAl1Search* (*procedure TLablForm. BtnAllSearchClick* процедураси*)* тугмасини бир неча марта босиш билан чақирилади. Излаш иккита жадвалда бирданига бажарилади, статистик ахборотларни йиғиш ва излаш натижалари мос майдонларда тасвирланади.

6.5. Қилинган ишлар таҳлили.

Бир қатор матнли файллар учун ёзилаган дастур кодини бажариш натижасида псевдотасодиф сонлар генератори асосида рехэшлаштиришни ишлатиш билан идентификаторларни излаш ва жойлаштириш учун 20% гача (45 идентификаторгача) идентификаторларни жадвалга тўлдиришда ўртача таққослашлар сони хэш-адреслашни ишлатганда кичик бўлади. Жадвални 20% дан 40 %гача тўлдиришда (тахминан 45 – 90 идентификатор) иккала усул ҳам тенг кўрсаткичларга эга бўлади, аммо жадвални 40% дан кўп (90 - 223 идентификаторлар) тўлдирилса, комбинацияланган усул самарадорлиги рехэшлаштиришга кўра тез ўсади. Агар киритишда 223 дан кўп идентификаторлар бўлса, рехэшлаштириш тўхтатилади.

Шундай қилиб, хеш-адреслаш усули оддий хэш-функциянинг мавжудлигида ҳам ишлайди ва яхши натижаларни беради (500 – 700 идентификаторлардан иборат кирувчи файллардаги ўртача 3 – 5 таққослашлар), бу вақтда эса реал иш учун рехэшлаштириш усули минглаб ёки ўнг минглаб диапазон қийматли бир неча мураккаб хэш-функцияларни талаб қилади.

7. Х у л о с а.

Идентификаторларни ташкиллаштириш дастури компилятор таркибидаги асосий дастурлардан бири бўлиб, бошланғич дастурни таҳлил қилишда дастурда мавжуд бўлган иднтификаторлар учун идентификаторлар жадвалини ҳосил қилади.

Идентификаторлар жадвалини ташкиллаштиришнинг қуйидаги усулларини кўрсатиш мумкин:

- оддий ва тартибланган рўйхатлар;
- бинар дарахт,
- рехэширлаштириш билан хэш-адреслаш;
- занжир усули бўйича хэш-адреслаш;
- рўйхат ёки бинар дарахт билан хэш-адреслаш комбинацияси.

Курс ишида ана шу усуллардан рехэширлаштириш билан хэш-адреслаш усули ёрдамида идентификаторлар жадвалини ташкиллаштириш масаласи ўрганилган ва дастур тузилган.

8. Фойдаланилган адабиётлар.

8.1. Асосий адабиётлар.

1. Вирт Н. Алгоритмы и структуры программы. М., Мир, 1985.
2. Берзтисс А.Т. Структуры данных. М. Статистика, 1984.
3. Трамбле Ж., Соренсон П. Введение в структуры данных. М., Машиностроение, 1982.
4. Ленгсам Й. Структуры данных для персональных ЭВМ. М., Мир, 1989.

8.2. Қўшимча адабиётлар.

1. Дж Ульман, и др. Введение в системы баз данных. М., Лори , 2000.
2. Д. Мейер, Теория реляционных баз данных, М, Мир,1987.