Дәріс-9. Мәліметтердің динамикалық құрылымдары

Жоспар:

1 Мәліметтердің динамикалық құрылымдары
2 Сызықтық тізімдер
3 Стектер
4 Кезектер
5 Бинарлы бұтақтар

1 Мәліметтердің динамикалық құрылымдары

Компилятор айнымалыны анықтау операторын өңдеу кезінде, мысалы, int i =10; , ол типке (int) сәйкес жады аймағын бөледі де, оған айнымалы мәнін жазады (10). Программадағы і айнымалысын пайдалануды компилятор көрсетілген мән сақталатын жады адресіне (нөміріне) ауыстырады.

Программалаушы жады адрестерін есте сақтайтын өз айнымалыларын анықтай алады. Олар нұсқауыштар болып табылады. Сонымен, нұсқауыштар жады аймағы адрестерін сақтау үшін керек.

Java тілінде 3 түрлі нұсқауыштар қолданылады: объектіге нұсқауыш, функцияға нұсқауыш және void типіне нұсқауыш, олар өз қасиеттерімен және орындалатын операциялар жиынымен ерекшеленеді. Нұсқауыш жеке тип емес, ол бір нақты типпен тұрақты байланыс жасайды.

Нұсқауыштар көбінесе компьютердің динамикалық жадымен жұмыс істеу кезінде қолданылады. Ол – программа орындалуы кезінде қажеттілігіне қарай пайдаланылатын жадыдағы белгілі бір бос аймақ (орын). Динамикалық жады аймағын пайдалану динамикалық айнымалылар көмегімен тек нұсқауыштар арқылы атқарылады.

Динамикалық айнымалылардың қолданылу кезеңі – өмірлік мерзімі (время жизни) олар жасалған сәттен бастап, программа аяғына шейін немесе жады әдейілеп босатылғанға дейін жалғасады.

Егер программа жұмысы басталғанға дейін компьютер жадында оның мәліметтерін сақтауға қанша орын керек екендігін анықтай алмасақ, онда жады нұсқауыштар арқылы блоктар түрінде жұмыс барысында біртіндеп бөлініп беріліп отырады.

Java тілінде динамикалық жадымен жұмыс істеудің екі тәсілі бар: алғашқысы – **malloc** функциясын қолдану (ол Java тілінен келген) екіншісі – **new** және **delete** операцияларын пайдалану.

Егер жұмысты бастағанға дейін мәліметтерді сақтауға қанша жады талап етілетінін анықтау мүмкін болмаса, онда жады қажеттілікке байланысты бөлінеді. Мәліметтерді ұйымдастырудың бұл тәсілі мәліметтердің динамикалық құрылымы деп аталады, мұндағы керекті жады мөлшері программаның орындалуы кезінде өзгеріп отырады.

New операциясы арқылы бөлінген жады **delete** арқылы босатылады, ал **malloc** функциясымен берілген жады – **free** функциясымен босайды.

Программаларда динамикалық құрылымдардың сызықтық тізімдер, стектер, кезектер және бинарлық бұтақтар сияқты түрлері қолданылады. Олар жеке элементтердің бір-бірімен байланысы және олармен орындауға болатын

операциялар арқылы ерекшеленеді.

Динамикалық құрылымдар компьютер жадының үзіліссіз аймағын емес, оның әр жерінде де орналаса алады.

Динамикалық құрылымдар көлемдері алдын ала белгілі мәліметтермен тиімді жұмыс істеу үшін жиі қолданылады. Мысалы, егер программада көлемді жиым элементтерімен жұмыс істеу керек болса, оларды сызықты тізім ретінде қарастыруға болады.

1 Сызықтық тізімдер

Программада сызықтық тізімдер, стектер, кезектер, бинарлы бұтақтар жиі қолданылады. Элементтер жиынын байланыстырудың ең қарапайым тәсілі – әр элементтің келесі элементке сілтеме жасауы. Мұндай тізім бір бағытты деп аталады. Егер әр элементке екінші сілтемені (алдыңғы элементке) қоссақ, онда ол екі бағытты болып шығады. Ал егер соңғы элементті нұсқауыш арқылы алғашқы элементпен байланыстырсақ, онда ол сақиналы тізім деп аталады.

Тізімнің әрбір элементінің оны анықтайтын кілті болады. Кілт не бүтін сан, не тіркес, не мәлімет өрісінің белгілі бір бөлігі болуы мүмкін. Мысалы, тізімді алфавит бойынша орналастыру үшін кілт рөлін фамилия атқарса, еңбек ардагерін анықтаудағы кілт – стаж (жұмыс өтілі) бола алады.

Тізіммен келесі операцияларды орындауға болады:

- тізімнің бастапқы қалыптасуы (1-элементті жасау);
- тізімнің соңына элемент қосу;
- кілті берілген элементті оқу;
- элементті тізімнің берілген жеріне кірістіріп қою;
- кілті берілген элементті жою;
- кілт көмегімен тізімді реттеу.

Төменде 5 саннан тұратын тізім жасалып, оған сан енгізіп және тізімнен санды алып тастап, тізімді экранға шығаратын программа мәтіні келтірілген. Тізім басына нұсқауыш pbeg деп аталған, ал тізім соңына нұсқауыш pend деп, қосымша нұсқауыштар pv және pkey болып көрсетілген.

```
#include <iostream.h> struct Node{

int d;
Node *next; Node *prev;};
//---------------------------------
Node * first(int d);
void add(Node **pend, int d);
Node * find(Node * const pbeg, int i);
bool remove(Node **pbeg, Node **pend, int key);
Node * insert(Node * const pbeg, Node **pend, int key, int d); //-------------
-------------------------
int main(){
Node *pbeg = first(1); //Тізімнің 1-элементін қалыптастыру
```

```
Node *pend = pbeg; /* Тізім аяқталды, енді тізім соңына 2, 3, 4, 5 қосамыз:*/ for (int i = 2; i<6; i++) add(&pend, i);
// жаңа элементті (200) 2-элементтен соң енгізу:
insert(pbeg, &pend, 2, 200); // Енді 5-элементті өшіреміз:
if(!remove (&pbeg, &pend, 5)) cout << "табылмады";
Node *pv = pbeg;
while (pv){            // тізімді экранға шығару
cout << pv->d << ' ';
pv = pv->next;  }  return 0; }
```

Бақылау сұрақтар:

1. Java тілінде неше түрлі нұсқауыштар қолданылады?
2. Java тілінде динамикалық жадымен жұмыс істеудің неше тәсілі бар?
3. Тізіммен қандай операцияларды орындауға болады?