2 - AMALIY MASHG‘ULOT. *ABSTRAKT MA’LUMOT TURLARI VA MA’LUMOTLAR TUZILMALARI.*

Ishdan maqsad: Talabalarni ma‘lumot turlari va ma‘lumot tuzilmalari bilan tanishtirish. Turlar va tuzilmalarni o‘rganib chiqish.

Nazariy qism. Algoritmning mohiyatini yanada ixchamroq yetkazish uchun ma’lumotlarni abstraktsiyalash, modullilik va xatolarni qayta ishlash masalalari ko’pincha e’tiborga olinmaydi.

Abstrakt ma’lumotlar turi (ADT) - bu ma’lumotlar turini va ushbu ma’lumotlar turida bajarilishi mumkin bo’lgan operatsiyalarni belgilaydigan matematik ma’lumotlar modeli, lekin bu operatsiyalarning maxsus amalga oshirilishini aniqlamaydi. Ya’ni, ADT interfeysni belgilaydi, lekin amalga oshirishni emas.

Abstrakt ma’lumot turlariga misollar *steklar, navbatlar, ro’yxatlar, daraxtlar* va boshqalarni o’z ichiga oladi. Ularning barchasi ma’lumotlar turini va ushbu ma’lumotlar ustida bajarilishi mumkin bo’lgan operatsiyalar to’plamini belgilaydi.

Abstrakt ma’lumot turlari (Abstrakt Ma’lumotlar Turlar, ADT) va ma’lumotlar tuzilmalari ma’lumotlarni tartibga solish va boshqarishga yordam beradigan informatika fanidagi asosiy tushunchalardir. Ushbu darsda biz abstrakt ma’lumotlar turlari nima ekanligini, ular ma’lumotlar tuzilmalaridan qanday farq qilishini va dasturiy ta’minotni ishlab chiqishda qanday ishlatilishini ko’rib chiqamiz.

Har xil abstrakt ma’lumotlar turlari uchun operatsiyalarga misollar:

1. Stack:

- surish (element qo’shish)
- Pop (elementni olib tashlash)
- Peek (yuqori elementni ko’rish)

2. Navbat:

- Navbatga qo’ying (oxiriga element qo’shing)
- Dequeue (elementni boshidan olib tashlash)
- Old (birinchi elementni ko’rish)
- Orqa (oxirgi elementni ko’rish)

3. Ro‘yxat:

- Insert (elementni kiritish)
- O’chirish (elementni o’chirish)

- Qidiruv (elementni topish)

Ma'lumotlar strukturasi - bu abstrakt ma'lumotlar turini aniq amalga oshirish. U kompyuter xotirasida ma'lumotlar qanday tashkil etilganligini va bu ma'lumotlar bilan qanday operatsiyalarni bajarish mumkinligini belgilaydi. Turli xil ma'lumotlar tuzilmalari bir xil abstrakt ma'lumotlar turini amalga oshirishi mumkinligini tushunish muhimdir, lekin har xil ishlash va xotiradan foydalanish xususiyatlariga ega.

Ma'lumotlar tuzilmalariga *massivlar, bog'langan ro'yxatlar, daraxtlar, xesh jadvallar* va boshqalar kiradi. Abstrakt ma'lumotlar turlarini amalga oshirishga misollar:

1. Stack uchun siz massiv yoki bog'langan ro'yxat ma'lumotlar strukturasidan foydalanishingiz mumkin.

2. Navbat uchun "massiv" yoki "bog'langan ro'yxat" dan ham foydalanishingiz mumkin.

3. Ro'yxat uchun siz "bog'langan ro'yxat" yoki "dinamik massiv" dan foydalanishingiz mumkin.

Abstrakt ma'lumotlar turlari va ma'lumotlar tuzilmalari dasturchilarga ma'lumotlarni samarali tarzda tartibga solish va boshqarish imkonini berish orqali dasturiy ta'minotni ishlab chiqishda muhim rol o'ynaydi. ADT va ma'lumotlar tuzilmalari o'rtasidagi farqlarni tushunish muayyan vazifalar uchun mos dasturlarni tanlashga va dastur ishlashini optimallashtirishga yordam beradi.

Amaliy qism

Bu yerda C++ da abstrakt ma'lumotlar turlari va ma'lumotlar tuzilmalari bo'yicha muammolarning misollari keltirilgan:

1. **Stack**

Stack - abstrakt ma'lumotlar turi bo'lib, u faqat ma'lum bir tartibda kirish mumkin bo'lgan elementlar to'plamidir. Elementni qo'shish (surish), elementni olib tashlash (pop) va yuqori elementga kirish (peek) operatsiyalari oxirgi kirish, birinchi chiqish tartibida amalga oshiriladi (LIFO - Oxirgi In , Birinchi Chiqardi).

Stackni amalga oshirish:

Vazifa: Dinamik massiv yordamida stekni amalga oshirish.

Yechim: C ++ da stekni amalga oshirish uchun int tipidagi elementlar vektorini o'z ichiga olgan Stack klassi qo'llaniladi. Surish operatsiyasi vektor oxiriga element qo'shadi, pop operatsiyasi oxirgi elementni olib tashlaydi va peek operatsiyasi oxirgi elementni olib tashlamasdan qaytaradi.

```
#include <iostream>
using namespace std;
const int MAX_SIZE = 100;
class stack {
private:
int arr [MAX_SIZE];
int top;
public:
Stack() {
top = -1;
}
void push (int val ) {
if (top >= MAX_SIZE - 1) {
            cout << "Stack Overflow" << endl;
return 0;
}
        arr [++top] = val;
}
void pop() {
if (top <0) {
            cout << "Stack Underflow" << endl;
return 0;
}
top --;
}
int peek() {
if (top <0) {
            cout << "Stack bo'sh" << endl;
return -1;
}
return arr [top];
}
bool isEmpty () {
return (top < 0);
        }
};
int main() {
Stack s;
      s.push (1);
      s.push (2);
      s.push (3);
      cout << s.peek () << endl;
      s.pop ();
      cout << s.peek () << endl;
```

```
        s.pop ();
        cout << s.peek () << endl;
        s.pop ();
        cout << s.isEmpty () << endl;
    return 0;
    }
```

2. Navbat

Navbat abstrakt ma'lumotlar turi bo'lib, faqat ma'lum bir tartibda kirish mumkin bo'lgan elementlar to'plamidir. Elementni qo'shish (navbat) va elementni olib tashlash (dequeue) operatsiyalari "birinchi kir, birinchi chiqadi" (FIFO - birinchi) tartibida amalga oshiriladi. In, Birinchi Chiqardi).

Navbatni amalga oshirish:

Vazifa: Bog'langan ro'yxat yordamida navbatni amalga oshiring.

Yechim: C ++ da navbatni amalga oshirish uchun qatorlar navbatini o'z ichiga olgan Queue klassi ishlatiladi (queue < string >). Navbatga qo'yish operatsiyasi navbat oxiriga element qo'shadi va navbatni o'chirish operatsiyasi birinchi elementni olib tashlaydi.

```
#include <iostream>
  using namespace std;
struct Node {
int data;
Node* next;
};
   class Queue {
private:
Node* front;
Node* rear;
    public:
      Queue() {
front = rear = nullptr;
}
Void enqueue (int val) {
Node* newNode = new Node;
        newNode ->data = val;
        newNode -> next = nullptr;
if (rear == nullptr) {
front = rear = newNode;
return;
}
rear->next = newNode;
```

```
rear = newNode;
}
void dequeue() {
if (front == nullptr) {
            cout << "Navbat bo'sh" << endl;
return;
}
Node* temp = front;
front = front->next;
delete temp;
if (front == nullptr) {
rear = nullptr;
        }
}
int peek() {
if (front == nullptr) {
            cout << "Navbat bo'sh" << endl;
return -1;
}
return front->data;
}
bool isEmpty () {
return (front == nullptr);
}
};
int main() {
Queue q;
      q.enqueue (1);
      q.enqueue (2);
      q.enqueue (3);
      cout << q.peek () << endl;
      q.dequeue ();
      cout << q.peek () << endl;
      q.dequeue ();
      cout << q.peek () << endl;
      q.dequeue ();
      cout << q.isEmpty () << endl;
return 0;
}
```

3. **Ikki marta ulangan Ikki marta bog'langan ro'yxat**

Ro’yxat (bog’langan ro’yxat) - har birida qiymat va ro’yxatning keyingi tuguniga havola (ko’rsatkich) bo’lgan tugunlardan tashkil topgan ma’lumotlar strukturasi. Ro’yxatning asosiy operatsiyalariga elementni kiritish, elementni olib tashlash (o’chirish) va elementni qidirish (qidirish) kiradi.

Ro’yxatni amalga oshirish:

Vazifa: Ikki marta bog’langan ro’yxatni amalga oshiring va elementlarni kiritish va olib tashlash usullarini qo’shing.

Yechim: C ++ da yakka bog’langan ro’yxatni amalga oshirish uchun LinkedList sinfidan foydalaning, bu ro’yxatning birinchi elementiga (head) ko’rsatgichni o’z ichiga oladi. Qo’shish operatsiyasi yangi tugunni yaratadi va uni ro’yxatning boshiga qo’yadi, olib tashlash operatsiyasi tugunni qiymat bo’yicha olib tashlaydi va qidiruv operatsiyasi tugunni qiymat bo’yicha qidiradi.

```
#include <iostream>
using namespace std;
struct Node {
        int data;
        Node* prev;
        Node* next;
};
Class DoubleLinkedList {
private:
Node * head;
public:
      DoubleLinkedList () {
        head = nullptr;
}
void insertAtBeginning (int val) {
        Node* newNode = newNode;
         newNode ->data = val;
         newNode -> prev = nullptr;
         newNode -> next = head;
if (head != nullptr ) {
        head-> prev = newNode;
}
head = newNode;
}
void insertAtEnd (int val) {
        Node* newNode = newNode;
         newNode ->data = val;
         newNode -> next = nullptr;
if (head == nullptr) {
```

```
        newNode -> prev = nullptr;
head = newNode;
return;
}
Node* temp = head;
while (temp->next!= nullptr ) {
        temp = temp->next;
}
        temp->next = newNode;
         newNode -> prev = temp;
}
void deleteAtBeginning () {
if (head == nullptr) {
        cout << "Ro'yxat bo'sh" << endl;
return;
}
Node* temp = head;
head = head->next;
if (head != nullptr ) {
        head-> prev = nullptr;
} else {
        delete temp;
}
void deleteAtEnd () {
        if (head == nullptr) {
                cout << "Ro'yxat bo'sh" << endl;
return;
}
if (head->next == nullptr ) {
        delete head;
head = nullptr;
return;
}
Node* temp = head;
while (temp->next!= nullptr ) {
        temp = temp->next;
}
temp-> prev -> next = nullptr;
delete temp;
}
void displey() {
Node* temp = head;
 (temp != nullptr ) {
        cout << temp->data << " ";
```

```
temp = temp->next;
}
        cout << endl;
}
};
int main() {
    DoubleLinkedList dll;
    dll.insertAtEnd (1);
    dll.insertAtEnd (2);
    dll.insertAtEnd (3);
    dll.display ();
    dll.deleteAtBeginning ();
    dll.display ();
    dll.deleteAtEnd ();
    dll.ko'rsatish ();
    return 0;
}
```

Abstrakt ma'lumotlar turlari (ADT) ma'lumotlar modellari bo'lib, ularning ichki ko'rinishi orqali emas, balki ushbu ma'lumotlar ustida bajarilishi mumkin bo'lgan operatsiyalar to'plami orqali aniqlanadi. Ular amalga oshirish tafsilotlaridan abstrakt bo'lishga va ma'lumotlardan foydalanishga e'tibor berishga imkon beradi. Ma'lumotlar tuzilmalari, o'z navbatida, samarali saqlash, qayta ishlash va boshqarish maqsadida ma'lumotlarni tashkil qilish usullaridir. Ular ma'lumotlar qanday tuzilishini va bir-biri bilan o'zaro ta'sirini aniqlaydi.

Abstrakt ma'lumotlar turlari ushbu ma'lumotlar ustida bajarilishi mumkin bo'lgan operatsiyalar to'plami bilan belgilanadi. Ular odatda ma'lumotlarni yaratish, kirish, o'zgartirish va o'chirish operatsiyalarini o'z ichiga oladi.

Umumiy ADTlarga misollar

Abstrakt ma'lumotlar turlariga misollar steklar, navbatlar, ro'yxatlar, daraxtlar va grafiklarni o'z ichiga oladi. Ushbu ma'lumotlar turlarining har biri ma'lumotlar bilan ishlash uchun o'ziga xos operatsiyalar to'plamini taqdim etadi.

ATD bilan bajariladigan operatsiyalar

Abstrakt ma'lumotlar turlari bo'yicha operatsiyalar yangi namuna yaratish, elementlar qo'shish, elementlarga kirish, elementlarni o'zgartirish va elementlarni o'chirishni o'z ichiga oladi.

ADT dan foydalanishning afzalliklari

Abstrakt ma’lumotlar turlaridan foydalanish sizni amalga oshirish tafsilotlaridan abstraktlashtirishga va ma’lumotlardan foydalanishga e’tibor qaratishga imkon beradi. Bu kodni o’qilishi mumkin, modulli va moslashuvchan qiladi.

Ma’lumotlar tuzilmalari ma’lumotlar qanday tashkil etilishini va bir-biri bilan o’zaro ta’sirini belgilaydi. Ular ma’lumotlarni samarali saqlash, qayta ishlash va boshqarishda asosiy rol o’ynaydi.

Ma’lumotlar tuzilmalarining turlari

Ma’lumotlar tuzilmalarining har xil turlari mavjud, jumladan, chiziqli (ro’yxatlar, steklar, navbatlar) va chiziqli bo’lmagan (daraxtlar, grafiklar). Ularning har biri o’ziga xos xususiyatlarga va turli vaziyatlarda qo’llanilishiga ega.

Ma’lumotlar tuzilmalari bilan bajariladigan operatsiyalar

Ma’lumotlar tuzilmalari bo’yicha operatsiyalar elementlarni qo’shish, o’chirish, kirish va o’zgartirishni, shuningdek, turli xil o’tish va qidirish algoritmlarini o’z ichiga oladi.

Turli ma’lumotlar tuzilmalarining afzalliklari va kamchiliklari

Har bir ma’lumotlar strukturasi o’zining afzalliklari va kamchiliklariga ega. Misol uchun, massivlar elementlarga tezkor kirishni ta’minlaydi, lekin ularning o’lchamlari qat’iydir, bog’langan ro’yxatlar esa elementlarni osongina qo’shish va o’chirish imkonini beradi, lekin ko’rsatkichlarni saqlash uchun qo’shimcha xotira talab qiladi.

Mustaqil bajarish uchun topshiriqlar

1. Massiv yordamida stekni amalga oshirish.
2. Bog’langan ro’yxat yordamida navbatni amalga oshirish.
3. Bog’langan ro’yxatda tsikl mavjudligini tekshirish.
4. Ikkilik daraxtning chuqurlik-birinchi o’tishi.
5. Ikkilik daraxtning kengligi-birinchi o’tishi.
6. Ikkilik daraxtdagi elementni kalit bo‘yicha qidiring.
7. Ochiq adreslash bilan xesh-jadvalni amalga oshirish.
8. Zanjirlar yordamida xesh-jadvalni amalga oshirish.
9. Butun sonlar massivini birlashtirish.
10. QuickSort yordamida butun sonlar massivini saralash.
11. Ikki marta bog’langan ro’yxatni amalga oshirish.

12. Floyd algoritmi yordamida bog'langan ro'yxatda tsikl mavjudligini tekshirish.

13. Kenglik-birinchi qidiruv yordamida o'lchovsiz grafikda eng qisqa yo'lni topish.

14. Dijkstra algoritmidan foydalanib, vaznli grafikda eng qisqa yo'lni topish.

15. Bellman-Ford algoritmi yordamida vaznli grafikda eng qisqa yo'lni topish.

16. Uyumga asoslangan ustuvor navbatni amalga oshirish (Heap).

17. Massiv yordamida dek (ikki tomonlama navbat) ni amalga oshirish.

18. Ikkilik daraxtda eng kichik umumiy ajdodni (LCA) topish.

19. Maksimal element uchun qidiruv operatsiyalarini qo'llab-quvvatlovchi stekni amalga oshirish.

20. Maksimal element uchun qidiruv operatsiyalarini qo'llab-quvvatlash bilan navbatni amalga oshirish.

21. Yo'naltirilgan grafikdagi ikkita cho'qqi orasidagi barcha yo'llarni chuqurlik-birinchi o'tish usuli yordamida topish.

22. Ikkilik to'pni amalga oshirish (Binary Uyum).

23. Yo'naltirilmagan grafikdagi barcha bog'langan komponentlarni chuqurlik-birinchi qidiruv yordamida topish.

24. A* algoritmi yordamida vaznli grafikdagi eng qisqa yo‘lni topish.

25. Massiv yordamida halqali navbatni amalga oshirish.

Ushbu muammolarning har biri dasturlash tilida dastur sifatida taqdim etilishi va keyin tegishli abstrakt ma'lumotlar turlari va tuzilmalari yordamida yechilishi mumkin.