Дәріс 10. ООП негіздері. кластар және объектілер Инкапсуляция, класс мүшелері, әдістер, конструкторлар және деструкторлар.

Объектіге бағытталған бағдарламалау (ООП) негіздері: кластар және объектілер

---

1. Кіріспе

C++ тілі — **объектіге бағытталған бағдарламалау (Object-Oriented Programming, OOP)** тілдерінің бірі.
ООП – бұл бағдарламаны нақты өмірдегі **объектілер мен олардың өзара әрекеттесуін** модельдеу тәсілі.

Оның басты идеясы:

Бағдарлама – өзара байланысқан **объектілер жиынтығы**.

---

2. ООП ұғымы

Объектіге бағытталған бағдарламалау **деректер мен оларды өңдейтін функцияларды біріктіреді**.
Бұл тәсіл **модульдік**, **икемді** және **қайта пайдалануға болатын** бағдарламалар жазуға мүмкіндік береді.

ООП-тың негізгі төрт қағидасы:

№	Қағида	Түсіндірме
1	**Инкапсуляция (Encapsulation)**	Деректер мен оларды өңдейтін әдістерді бір модульге (классқа) біріктіру
2	**Мұрагерлік (Inheritance)**	Бір кластың қасиеттерін басқа классқа беру
3	**Полиморфизм (Polymorphism)**	Бір интерфейстің әртүрлі жүзеге асырылуы

№	Қағида	Түсіндірме
4	**Абстракция (Abstraction)**	Тек маңызды қасиеттерді көрсету, артық мәліметті жасыру

Бұл дәрісте біз ең басты екі қағиданы — **инкапсуляция** және **кластар/объектілермен жұмысты** қарастырамыз.

---

3. Класс және объект ұғымдары

3.1 Класс дегеніміз не?

Класс (class) — бұл **объектінің үлгісі (template)** немесе **типі**.
Класс деректерді (fields) және олармен жұмыс істейтін әдістерді (methods) біріктіреді.

3.2 Объект дегеніміз не?

Объект (object) — бұл кластың **нақты экземпляры (instance)**, яғни, кластан жасалған нақты дерек.

3.3 Мысал:

```
#include <iostream>
using namespace std;

class Car {
public:
    string brand;
    int year;

    void show() {
        cout << "Марка: " << brand << ", Жылы: " << year << endl;
    }
};

int main() {
    Car car1; // объект жасау
```

```
    car1.brand = "Toyota";
    car1.year = 2020;
    car1.show();

    Car car2;
    car2.brand = "Hyundai";
    car2.year = 2023;
    car2.show();
}
```

Нәтиже:

Марка: Toyota, Жылы: 2020
Марка: Hyundai, Жылы: 2023

---

4. Класс құрылымы

Класс келесідей бөлімдерден тұрады:

```
class ClassName {
private:
    // жеке (private) деректер мен функциялар
protected:
    // мұрагерлерге қолжетімді элементтер
public:
    // жалпыға қолжетімді әдістер мен өрістер
};
```

Мүшелердің қолжетімділік модификаторлары

Кілт сөз	Қол жетімді аймақ	Сипаттама
private	Тек осы кластың ішінде	Инкапсуляция үшін қолданылады
protected	Класс ішінде және туынды кластарда	Мұрагерлік кезінде пайдаланылады
public	Барлық жерден қолжетімді	Интерфейс элементтері үшін

---

5. Инкапсуляция

Инкапсуляция — бұл деректерді қорғау механизмі.
Ол деректер мен оларды өңдейтін әдістерді **бірге жабу** арқылы жүзеге асады.

Мақсаты:

- Деректерді **сыртқы тікелей өзгертуден қорғау**;
- Тек **арнайы әдістер** арқылы ғана қатынауға мүмкіндік беру.

Мысал:

```
#include <iostream>
using namespace std;

class BankAccount {
private:
    double balance;

public:
    BankAccount(double initial) {
        balance = initial;
    }

    void deposit(double amount) {
        balance += amount;
    }

    void withdraw(double amount) {
        if (amount <= balance)
            balance -= amount;
        else
            cout << "Жеткілікті қаражат жоқ!" << endl;
    }

    double getBalance() {
        return balance;
    }
};

int main() {
```

```
    BankAccount acc(1000);
    acc.deposit(500);
    acc.withdraw(200);
    cout << "Қалдық: " << acc.getBalance() << endl;
}
```

Нәтиже:

Қалдық: 1300

Мұнда balance өрісіне тікелей қатынау болмайды — тек әдістер арқылы ғана.

---

6. Кластың әдістері (methods)

Кластың ішінде анықталған функциялар **әдіс (method)** деп аталады.
Әдіс **класс деректеріне** (this көрсеткіші арқылы) қатынай алады.

Мысал:

```
class Rectangle {
private:
    double width;
    double height;

public:
    void setValues(double w, double h) {
        width = w;
        height = h;
    }

    double area() {
        return width * height;
    }
};
```

Қолдану:

```
int main() {
    Rectangle r;
    r.setValues(5, 3);
```

```
    cout << "Ауданы: " << r.area();
}
```

Нәтиже:

Ауданы: 15

Конструктордың түрлері:

Түрі	Сипаттамасы
Әдепкі (default)	Параметрсіз
Параметрлі (parameterized)	Параметр қабылдайды
Көшірме (copy constructor)	Басқа объектіден көшіреді
Инициализация тізімі бар (initializer list)	Мүшелерді тиімді инициализациялау

Инициализация тізімі мысалы:

```
class Student {
    string name;
    int age;
public:
    Student(string n, int a) : name(n), age(a) {}
};
```

---

10. this көрсеткіші

this — ағымдағы объектіге сілтеме.
Ол әдіс ішінде осы объектінің мүшелеріне қатынау үшін қолданылады.

Мысал:

```
class Point {
    int x, y;
```

```
public:
   Point(int x, int y) {
      this->x = x;
      this->y = y;
   }
};
```

---

11. Объектілер массиві

Бір типтегі бірнеше объектіден массив жасауға болады:

```
class Student {
public:
   string name;
   int age;
   void show() { cout << name << " (" << age << " жаста)" << endl; }
};

int main() {
   Student group[3] = {
      {"Aidos", 20},
      {"Dana", 21},
      {"Murat", 19}
   };

   for (auto& s : group)
      s.show();
}
```

---

12. Кластардың артықшылықтары

Артықшылық	Түсіндірме
Деректерді қорғау	Инкапсуляция арқылы деректерді сыртқы араласудан сақтау
Қайта пайдалану	Кодты қайта қолдануға мүмкіндік
Модульдік құрылым	Бағдарлама модульдерге бөлінеді
Оңай кеңейту	Жаңа қасиеттерді оңай қосуға болады

Артықшылық	Түсіндірме
Нақты модельдеу	Нақты өмірдегі объектілерді модельдеуге мүмкіндік

---

13. Кластар мен struct айырмашылығы

Ерекшелік	struct	class
Әдепкі қолжетімділік	public	private
Мұрагерлік кезінде	public әдепкі	private әдепкі
Қолданылу саласы	Деректер тобы	Деректер мен логика бірлігі

---

14. Қысқаша қорытынды

Түсінік	Анықтама
Класс	Объектінің үлгісі, өрістер мен әдістер жиыны
Объект	Кластың экземпляры
Инкапсуляция	Деректерді қорғау және біріктіру тәсілі
Конструктор	Объект құрылған кезде шақырылатын әдіс
Деструктор	Объект жойылғанда шақырылатын әдіс

---

15. Өзіндік бақылау сұрақтары

1. Объектіге бағытталған бағдарламалау дегеніміз не?
2. Класс пен объекттің айырмашылығы қандай?
3. Инкапсуляцияның мәні неде?
4. Қол жетімділік модификаторларын атаңыз.
5. Конструктор не үшін қажет және қалай анықталады?
6. Деструктордың рөлі қандай?
7. Кластың ішінде және сыртында әдіс анықтаудың айырмашылығы неде?
8. this көрсеткіші не үшін қолданылады?

9. struct пен class арасындағы айырмашылық неде?
10. Кластарды қолданудың артықшылықтарын атаңыз.