Дәріс 11. Мұрагерлік және полиморфизм (C++)

1. Мұрагерлік (Inheritance) дегеніміз не?

Мұрагерлік – бұл жаңа класты бұрыннан бар кластың негізінде құру механизмі.
Ол арқылы базалық кластың (ата-аналық) қасиеттері мен әдістерін жаңа туынды класс (балалық) **қайта қолдана**алады.

Артықшылықтары:

- Кодты қайта қолдану
- Бағдарламаны жеңілдету және құрылымдау
- Кеңейтуге оңай болу

2. Базалық және туынды кластар

- **Базалық класс (Base class)** – бастапқы класс, оның қасиеттері мұрагерлікке беріледі.
- **Туынды класс (Derived class)** – базалық кластың қасиеттерін мұрагерлікке алып, жаңа қасиеттер қоса алады.

Синтаксис:

```
class Derived : public Base {
   // жаңа мүшелер
};
```

Мұндағы public – мұрагерліктің түрі (жиі қолданылатын түрі).

Мысал:

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

// Базалық класс
class Animal {
public:
   void eat() {
      cout << "Жануар тамақ ішеді" << endl;
   }
};

// Туынды класс
class Dog : public Animal {
public:
   void bark() {
```

```
        cout << "Ит үреді" << endl;
    }
};

int main() {
    Dog d;
    d.eat();   // базалық кластан мұрагерлік алған әдіс
    d.bark();  // өз әдісі
    return 0;
}
```

Нәтиже:

Жануар тамақ ішеді
Ит үреді

3. Мұрагерліктің түрлері

Түрі	Сипаттамасы
public	Базалықтың public → public, protected → protected
protected	Базалықтың public & protected → protected
private	Барлық мұрагерлік мүшелер → private

Ең жиі қолданылатыны – **public мұрагерлік**.

4. Полиморфизм (Polymorphism)

Полиморфизм – “көптүрлілік” дегенді білдіреді.
C++ тілінде бұл бір интерфейс арқылы әртүрлі әрекеттерді орындау қабілеті.

Түрлері:

- **Компиляция уақыты полиморфизмі** – функцияны жүктеу (overloading)
- **Орындау уақыты полиморфизмі** – виртуалды функциялар арқылы

5. Виртуалды функциялар (Virtual Functions)

Виртуалды функция – базалық класта жарияланып, туынды кластарда қайта анықталатын әдіс.
Бұл полиморфизмді іске асыруға мүмкіндік береді.

Синтаксис:

virtual қайтару_типі функция_аты();

Мысал:

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

class Animal {
public:
   virtual void sound() { // виртуалды функция
      cout << "Жануар дыбыс шығарады" << endl;
   }
};

class Dog : public Animal {
public:
   void sound() override { // қайта анықтау
      cout << "Ит үреді" << endl;
   }
};

class Cat : public Animal {
public:
   void sound() override {
      cout << "Мысық мияулайды" << endl;
   }
};

int main() {
   Animal* a;  // базалық көрсеткіш
   Dog d;
   Cat c;

   a = &d;
   a->sound();  // Ит үреді

   a = &c;
   a->sound();  // Мысық мияулайды

   return 0;
}
```

Егер virtual болмаса, әрдайым Animal нұсқасы шақырылады.

6. Абстрактілі кластар (Abstract Classes)

Абстрактілі класс – өзі арқылы объект құруға болмайтын, тек мұрагерлік үшін арналған класс.
Міндетті түрде **таза виртуалды функциясы (pure virtual function)** болуы тиіс.

Синтаксис:

virtual void функция_аты() = 0;

Мысал:

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

class Shape {
public:
   virtual void draw() = 0;  // таза виртуалды функция
};

class Circle : public Shape {
public:
   void draw() override {
      cout << "Шеңбер салу" << endl;
   }
};

int main() {
   // Shape s; Қате: абстрактілі кластан объект жасай алмаймыз
   Shape* s = new Circle();
   s->draw(); // Шеңбер салу
   delete s;
   return 0;
}
```

Абстрактілі кластар – интерфейс рөлін атқарады.

Қысқаша салыстыру

Түсінік	Мағынасы
Мұрагерлік	Жаңа класс бұрынғының қасиеттерін алады
Базалық класс	Негізгі класс (ата-аналық)
Туынды класс	Базалықтан мұрагерлік алған жаңа класс
Полиморфизм	Бір интерфейс арқылы әртүрлі әрекеттер орындау
Виртуалды функция	Туынды кластарда қайта анықталатын әдіс
Абстрактілі класс	Объект жасалмайтын, тек мұрагерлік үшін арналған класс

Артықшылықтары

- Кодты қайта пайдалану
- Кеңейтуге ыңғайлы архитектура
- Интерфейстер арқылы икемділік

- Полиморфизм арқасында бірыңғай интерфейспен жұмыс

Есте сақтау:

- Виртуалды функцияны базалық класта жарияласаң, туындыда оны қайта анықтап полиморфизмге қол жеткізесің.
- Абстрактілі класты объект ретінде қолдануға болмайды.
- Мұрагерлік – «is-a» қатынасына негізделеді (Dog **is an** Animal).

Қорытынды:
Мұрагерлік пен полиморфизм – ООП-тің ең маңызды принциптері. Олар кодты қайта пайдалануға, кеңейтуге және бір интерфейс арқылы түрлі әрекеттер орындауға мүмкіндік береді.