Дәріс 9. Құрылымдар (struct) және тізімдер (enum) Пайдаланушы типтері, struct, enum, құрама деректермен жұмыс.

Құрылымдар (struct) және тізімдер (enum)

1. Кіріспе

Бағдарламалау тілдерінде **деректер типтері** негізгі рөл атқарады.
Бірақ кейде қарапайым (int, double, char) типтер жеткіліксіз болады.
Бір объектіге қатысты бірнеше деректерді **бір типке біріктіру** қажет болады.

C++ тілінде мұндай мақсатта **құрылымдар (struct)** және **тізімдер (enum)** қолданылады.

Олар — **пайдаланушы анықтайтын типтер (user-defined data types)**.

---

2. Пайдаланушы анықтайтын типтер (User-defined types)

C++ тілінде деректер типтерін 3 топқа бөлуге болады:

Түрі	Мысал
Қарапайым (basic)	int, float, char, bool
Туынды (derived)	массив, көрсеткіш, функция
Пайдаланушы типтері (user-defined)	struct, enum, class, union, typedef

Пайдаланушы типтері бағдарламашыға нақты өмірлік ұғымдарды модельдеуге мүмкіндік береді.
Мысалы, студент, кітап, қызметкер, нүкте, уақыт т.б.

---

3. Құрылым (struct)

3.1 Анықтамасы

Құрылым (structure) — әртүрлі типтегі бірнеше деректерді **бір атаумен біріктіретін** деректер типі.

Синтаксисі:

```
struct ҚұрылымАты {
    деректер_типі мүше1;
    деректер_типі мүше2;
    ...
};
```

3.2 Мысал:

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

struct Student {
    string name;
    int age;
    double gpa;
};

int main() {
    Student s1;          // құрылымның экземпляры
    s1.name = "Aidos";
    s1.age = 20;
    s1.gpa = 3.7;

    cout << "Аты: " << s1.name << endl;
    cout << "Жасы: " << s1.age << endl;
    cout << "GPA: " << s1.gpa << endl;

    return 0;
}
```

Нәтиже:

```
Аты: Aidos
Жасы: 20
GPA: 3.7
```

3.3 Құрылымды инициализациялау (жариялау кезінде мән беру)

```
Student s2 = {"Dana", 21, 3.9};
```

немесе (C++11 және кейінгі нұсқаларда):

```
Student s3{"Murat", 19, 3.5};
```

3.4 Құрылым мүшелеріне қатынау

Мүшеге қатынау үшін **нүкте (.)** операторы қолданылады:

```
cout << s1.name;
s1.age = 22;
```

3.5 Құрылымды функцияға беру

а) Мән арқылы беру:

```
void print(Student s) {
    cout << s.name << " " << s.age << " " << s.gpa << endl;
}
```

ә) Сілтеме арқылы беру (тиімдірек):

```
void print(const Student& s) {
    cout << s.name << " " << s.age << " " << s.gpa << endl;
}
```

б) Көрсеткіш арқылы беру:

```
void print(const Student* s) {
    cout << s->name << " " << s->age << " " << s->gpa << endl;
}
```

3.6 Құрылым ішінде құрылым (Nested struct)

Кейде құрылымның ішінде басқа құрылым қолдануға болады.

```
struct Date {
    int day, month, year;
};

struct Employee {
    string name;
    double salary;
    Date hired;
};

int main() {
    Employee e = {"Askar", 450000, {15, 3, 2022}};
```

```
    cout << e.name << " hired on " << e.hired.day << "/" << e.hired.month << "/"
<< e.hired.year;
}
```

Нәтиже:

Askar hired on 15/3/2022

---

4. Құрылым массиві

Бір типтегі бірнеше құрылымдардан массив құруға болады.

```
Student group[3] = {
    {"Aidos", 20, 3.7},
    {"Dana", 21, 3.9},
    {"Murat", 19, 3.5}
};

for (int i = 0; i < 3; i++) {
    cout << group[i].name << " - " << group[i].gpa << endl;
}
```

---

5. Құрылым және динамикалық жад

new операторын пайдаланып құрылымды динамикалық түрде бөлуге болады:

```
Student* p = new Student;
p->name = "Aigul";
p->age = 22;
p->gpa = 3.8;

cout << p->name << " " << p->age;
delete p;
```

---

6. typedef және using

Құрылым аттарын қысқарту үшін **typedef** немесе **using** қолданылады.

```
typedef struct {
    string name;
```

```
    int age;
} Person;
```

немесе:

```
using Person = struct {
    string name;
    int age;
};
```

Сонда:

```
Person p1 = {"Aliya", 25};
```

---

7. enum (Тізімдер, санаулы типтер)

7.1 Анықтамасы

enum (enumeration) — бұл шектеулі және нақты мәндер жиынын қамтитын **пайдаланушы анықтайтын тип**.
Ол тұрақтыларды бір топқа біріктіруге мүмкіндік береді.

7.2 Синтаксис:

```
enum ТізімАты { элемент1, элемент2, элемент3, ... };
```

7.3 Мысал:

```
enum Day { Mon, Tue, Wed, Thu, Fri, Sat, Sun };

int main() {
    Day today = Wed;

    if (today == Wed)
        cout << "Бүгін сәрсенбі!";
}
```

Нәтиже:

```
Бүгін сәрсенбі!
```

7.4 Әдепкі мәндер және нөмірлеу

Әр элементке бүтін сан беріледі (әдепкіде 0-ден бастап).

enum Color { Red, Green, Blue }; // Red=0, Green=1, Blue=2

Мәндерді қолмен орнатуға да болады:

enum Color { Red = 10, Green = 20, Blue = 30 };

7.5 enum айнымалыларын пайдалану

```
Color c = Green;

switch (c) {
   case Red: cout << "Қызыл"; break;
   case Green: cout << "Жасыл"; break;
   case Blue: cout << "Көк"; break;
}
```

---

8. enum class (C++11 және кейінгі нұсқалар)

C++11-де жаңа, қауіпсіз нұсқа — **enum class** енгізілді.

Артықшылығы:

- Атаулар **атау кеңістігінде шектеледі** (name conflict болмайды);
- Тип қауіпсіздігі жоғары.

Мысал:

```
enum class Color { Red, Green, Blue };

int main() {
   Color c = Color::Green;

   if (c == Color::Green)
      cout << "Жасыл түсті таңдадыңыз.";
}
```

Color::Green деп жазу қажет, себебі ол енді enum class кеңістігі ішінде.

---

enum class және мән түрі

enum class үшін негіз типті көрсетуге болады:

```
enum class ErrorCode : int { Ok = 0, NotFound = 404, ServerError = 500 };
```

---

9. struct және enum комбинациясы

Кей жағдайларда құрылым ішінде enum қолданылады:

```
struct Order {
   int id;
   enum Status { New, Processing, Completed, Canceled } status;
};

int main() {
   Order o1 = {101, Order::Processing};
   cout << o1.id << " нөмірлі тапсырыс өңделуде.";
}
```

---

10. Құрылым мен enum қолданудың артықшылықтары

Артықшылық	Түсіндірме
Мәліметтерді топтастыру	Бір объектінің сипаттамаларын бір типке біріктіреді
Кодты оқуға жеңілдету	Атаулар мағыналы болады
Қателіктерді азайтады	enum class тип қауіпсіздігін қамтамасыз етеді
Функциялармен біріктіру	struct пен enum функциялармен, класс және массивтермен бірге қолдануға болады

---

11. Қорытынды

- **struct** — әртүрлі типтегі деректерді бір атаумен біріктіруге арналған құрал;
- **enum** — мәндердің шектеулі жиынын сипаттайтын тип;
- **enum class** — қауіпсіз, атау кеңістігімен шектелген жаңа нұсқа;
- Екеуі де **нақты өмірлік объектілерді модельдеуге** және **бағдарламаны құрылымдауға** мүмкіндік береді.

---

12. Өзіндік бақылау сұрақтары

1. Құрылым (struct) дегеніміз не?
2. Құрылым мүшелеріне қалай қатынайды?
3. Құрылым мен кластың айырмашылығы қандай?
4. Құрылымды функцияға қалай беруге болады?
5. enum не үшін қолданылады?
6. enum элементтерінің мәндері қалай беріледі?
7. enum class артықшылығы неде?
8. Құрылым ішінде enum қолдануға бола ма?
9. typedef және using қандай мақсатта қолданылады?
10. struct және enum комбинациясы қай жағдайда тиімді?