Дәріс 15. Қазіргі C++ мүмкіндіктері

C++ тілі соңғы жылдары елеулі өзгерістерге ұшырап, код жазуды жеңілдететін және қауіпсіздігін арттыратын көптеген жаңа мүмкіндіктерге ие болды.
Бұл дәрісте біз ең маңызды заманауи мүмкіндіктерге тоқталамыз:

- Ақылды көрсеткіштер (*smart pointers*)
- Move semantics
- auto және decltype
- range-based for
- Жаңа стандарттар шолуы (C++11/14/17/20)

1. Ақылды көрсеткіштер (Smart Pointers)

Ақылды көрсеткіштер – жадты автоматты түрде басқаруға арналған класс-объектілер. Олар new және delete қолданбай-ақ динамикалық жадпен қауіпсіз жұмыс істеуге мүмкіндік береді.

Негізгі түрлері:

Ақылды көрсеткіш	Сипаттамасы
std::unique_ptr	Бір ғана иесі бар, көшірілмейді
std::shared_ptr	Бірнеше иесі бар, соңғысы жойғанда жад босайды
std::weak_ptr	shared_ptr-мен бірге жұмыс істейді, санауға қатыспайды

<memory> кітапханасында орналасқан.

unique_ptr – бір ғана иелік

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

int main() {
   unique_ptr<int> ptr = make_unique<int>(42);
   cout << *ptr << endl; // 42

   // unique_ptr көшірілмейді:
   // unique_ptr<int> ptr2 = ptr;

   // Бірақ жылжытуға болады:
   unique_ptr<int> ptr2 = move(ptr);
   cout << *ptr2 << endl;
}
```

shared_ptr – ортақ иелік

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

int main() {
   shared_ptr<int> p1 = make_shared<int>(100);
   shared_ptr<int> p2 = p1; // ортақ пайдалану

   cout << *p1 << " " << *p2 << endl;
   cout << "Санауыш: " << p1.use_count() << endl;
}
```

Нәтиже:

```
100 100
Санауыш: 2
```

weak_ptr – әлсіз сілтеме

weak_ptr – shared_ptr санақ жүйесіне әсер етпейтін жеңіл сілтеме. Бұл тізбекті тәуелділікті (цикликалық сілтеме) болдырмауға көмектеседі.

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

int main() {
   shared_ptr<int> sp = make_shared<int>(50);
   weak_ptr<int> wp = sp;

   if (auto spt = wp.lock()) { // shared_ptr қайтарады
      cout << *spt << endl;
   }
}
```

2. Move Semantics (Жылжыту семантикасы)

Move semantics – ресурстарды көшірудің орнына **иелігін беру**.
Бұл бағдарламаның өнімділігін арттырады, әсіресе үлкен объектілермен жұмыс істегенде.

Екі жаңа ұғым:

- rvalue – уақытша мәндер (std::move көмегімен беріледі)
- && – жылжыту сілтемесі (rvalue reference)

Мысал:

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

string createString() {
   return "Сәлем";
}

int main() {
   string s1 = createString();      // көшіру
   string s2 = move(s1);            // жылжыту (ресурстар иелігі беріледі)

   cout << "s2: " << s2 << endl;
   cout << "s1 бос па? " << s1.empty() << endl;
}
```

Жылжыту барысында ресурстар көшірілмейді, тек **иелік беріледі** – бұл жылдам әрі тиімді

3. auto және decltype

auto – типті автоматты анықтау

Компилятор айнымалының типін оң жағындағы мәннен өзі шығарады.

```
auto x = 10;      // int
auto y = 3.14;    // double
auto s = "Hello";  // const char*
```

Циклдарда жиі қолданылады:

```
for (auto x : {1, 2, 3, 4})
   cout << x << " ";
```

decltype – басқа айнымалының типін көшіру

```
int a = 5;
decltype(a) b = 10; // b де int типінде
```

decltype функция қайтару типін анықтауда пайдалы:

```
template <typename T, typename U>
auto add(T a, U b) -> decltype(a + b) {
   return a + b;
}
```

4. Range-based for (диапазонды цикл)

C++11-де енгізілген range-based for – контейнер элементтерін шарлаудың қарапайым әрі қауіпсіз тәсілі.

Мысалы:

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

int main() {
   vector<int> v = {1, 2, 3, 4, 5};

   for (int x : v) {
      cout << x << " ";
   }

   // сілтеме арқылы өзгерту
   for (auto &x : v) {
      x *= 2;
   }

   cout << "\nКөбейтілген: ";
   for (auto x : v) cout << x << " ";
}
```

Нәтиже:

```
1 2 3 4 5
Көбейтілген: 2 4 6 8 10
```

5. Жаңа стандарттарға шолу

Стандарт	Жыл	Маңызды жаңалықтар
C++11	2011	auto, nullptr, range-for, move semantics, smart pointers
C++14	2014	Лямбда жақсартулары, сандық бөлшектер ('), қайтару типін шығару
C++17	2017	structured bindings, if constexpr, std::optional, std::variant
C++20	2020	Концепциялар (concept), корутиндер (co_await), ranges, модульдер (module)

Қазіргі C++ нұсқалары кодты **жазуды жеңілдетіп**, **өнімділікті арттырады** және **қауіпсіз етеді**.

Қысқаша қорытынды

Мүмкіндік	Сипаттамасы
Smart pointers	Жадты автоматты түрде басқару
Move semantics	Ресурстарды көшірудің орнына иелікті беру
auto / decltype	Типті автоматты шығару және анықтау
range-based for	Контейнерлермен жұмыс істеудің ыңғайлы циклі
Жаңа стандарттар	C++ тілін заманауи, қауіпсіз және икемді етеді

Артықшылықтары:

- Код көлемін азайтады
- Жад ағып кету қатерін төмендетеді
- Өнімділікті арттырады
- Оқуға және қызмет көрсетуге жеңіл код

Есте сақтау:

- unique_ptr, shared_ptr – жадты қолмен босатуға жол бермейді.
- std::move – иелікті жылжыту үшін қолданылады.
- auto – тип жазуды жеңілдетеді.
- range-based for – STL контейнерлерін оңай шарлайды.