15

O‘N BESHINCHI MA’RUZA

Dasturlash 1 · C++ · 10-mavzu

Konstruktorlar va destruktorlar

Konstruktor turlari · initsializatsiya ro‘yxati · explicit · sayoz va chuqur nusxa · Uchlik, Beshlik, Nol qoidalari · RAII

Nusxa | operator= | move | RAII

ISINISH

Kalitning nusxasini yasadingiz. Kalit bilan birga uy ham ikki barobar bo‘ldimi?

Nusxa olishning ikki xil ma’nosi bor — qaysi biri kerakligini o‘ylang.

1

Ikki kishida bitta uyning kaliti bo‘lsa, biri eshikni buzsa nima bo‘ladi?

2

Uyni butunlay qaytadan qurish (haqiqiy nusxa) qanchalik qimmat?

3

Uyni boshqa egaga “ko‘chirib berish”-chi — qurish shart emasmi?

O‘QUV MAQSADLARI

Bu ma’ruza oxirida siz quyidagilarni bajara olasiz...

1. **Tanlash** — vaziyatga mos konstruktor turini: standart, parametrli, nusxa, ko‘chirish, delegatsiya
2. **Qo‘llash** — initsializatsiya ro‘yxatini, ayniqsa const va havola maydonlar uchun
3. **Tushuntirish** — maydonlar e’lon tartibida initsializatsiya qilinishini va explicit ma’nosini
4. **Farqlash** — sayoz va chuqur nusxani, nusxa konstruktori va operator= ni
5. **Yozish** — Uchlik qoidasiga mos sinfni
6. **Qo‘llash** — ko‘chirish semantikasini: T&&, std::move, noexcept
7. **Asoslash** — RAII tamoyili va Nol qoidasini
8. **Taqiqlash** — nusxalashni = delete bilan, standartni = default bilan so‘rash

01

Konstruktor turlari

Standart · parametrli · nusxa · ko‘chirish · delegatsiya · = default / = delete

Nuqta()

Nuqta(int, int)

Delegatsiya

= default

Qaysi konstruktor qachon chaqiriladi?

Turi	Ko‘rinishi	Qachon
Standart	`Nuqta()`	`Nuqta p;`
Parametrli	`Nuqta(int x, int y)`	`Nuqta p(3, 4);`
Nusxa	`Nuqta(const Nuqta &b)`	`Nuqta q = p;`
Ko‘chirish	`Nuqta(Nuqta &&b)`	Vaqtinchalik obyektdan (5-bo‘lim)
Delegatsiya	`Nuqta() : Nuqta(0, 0) { }`	Bir konstruktor ishni boshqasiga beradi

Parametrli konstruktor yozilsa, standart konstruktor avtomatik yaratilmaydi — kerak bo‘lsa `Nuqta() = default;`

AMALDA

Konstruktorlar va delegatsiya

```
class Nuqta {
public:
    int x, y;
    Nuqta() : Nuqta(0, 0) { cout << "S "; }
    Nuqta(int x, int y = 0) : x(x), y(y) { cout << "P "; }
    Nuqta(const Nuqta &b) : x(b.x), y(b.y) { cout << "N "; }
};
int main() {
    Nuqta a;
    Nuqta b(5);
    Nuqta c{1, 2};
    Nuqta d = 7;
    Nuqta e = c;
    cout << endl << d.x << d.y << e.x << e.y;
}
```

▶ NATIJA

```
P S P P P N
7012
```

`Nuqta a;`

delegatsiya: avval P (0, 0), keyin S tanasi

`Nuqta d = 7;`

bitta argument — yashirin o‘zgartirish, y = 0

`Nuqta e = c;`

nusxa konstruktori (N)

MASSIVDA

Konstruktor necha marta chaqiriladi?

```
class H {
public:
    static int son;
    int id;
    H() : id(++son) { cout << "+" << id << " "; }
    ~H() { cout << "-" << id << " "; }
};
int H::son = 0;
int main() {
    { H arr[3]; cout << "| "; }
    H *p = new H[2];
    cout << "| ";
    delete[] p;
}
```

▶ NATIJA

```
+1 +2 +3 | -3 -2 -1 +4 +5 | -5 -4
```

H arr[3]

har element uchun standart konstruktor

blok tugaganda

destruktorlar teskari tartibda: 3, 2, 1

delete[]

har element uchun destruktor

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

1-bo‘lim yakuni: Konstruktor turlari

1

`Nuqta p();` yozuvi nimani bildiradi?

Savol 1

2

Qachon standart konstruktor avtomatik yaratilmaydi?

Savol 2

3

`= default` nima uchun kerak?

Savol 3

AMALIY TOPSHIRIQ

Har kim o‘z qurilmasida · 10 daqiqa · Oson · 10 ball

1-topshiriq: Konstruktor turlari: Nuqta

`Nuqta` sinfiga uchta konstruktor yozing: standart — `(0, 0)`; bitta parametrli — `(a, a)`; ikki parametrli — `(x, y)`. `void chiqar() const` nuqtani `(x, y)` ko‘rinishida chiqarsin. `main()` tayyor.

Kirish: Birinchi qatorda `q` (1 ≤ q ≤ 100). Har bir qatorda: `0`, `1 a` yoki `2 x y` — parametrlar soni va qiymatlari.

Chiqish: Har bir yaratilgan nuqta `(x, y)` ko‘rinishida.

MISOL · KIRISH

```
3
0
1 5
2 3 -4
```

MISOL · CHIQISH

```
(0, 0)
(5, 5)
(3, -4)
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. Kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

02

Initsializatsiya ro‘yxati va explicit

Majburiy holatlar · e’lon tartibi · yashirin o‘zgartirish · explicit

: x(5)

const &

Tartib

explicit

MAJBURIY HOLATLAR

Qachon initsializatsiya ro‘yxatisiz bo‘lmaydi?

1 const maydonlar

Ularga keyinroq qiymat berib bo‘lmaydi: `const string raqam;`

2 Havola maydonlar

Havola yaratilayotganda bog‘lanishi shart: `int &odometr;`

3 Standart konstruktorsiz a’zo

`Dvigatel dv;` — `Dvigatel(int)` faqat ro‘yxatda chaqiriladi.

4 Asosiy sinf

Asosiy konstruktorga argument: `Avtomobil(...) : Transport(n)`

Tanada `x = 5;` — bu initsializatsiya emas, allaqachon yaratilgan maydonga keyinroq qiymat berish.

TUZOQ

Maydonlar e’lon tartibida initsializatsiya qilinadi

```
struct Log {
    Log(const char *n) { cout << n << " "; }
};
class Tartib {
    Log birinchi;
    Log ikkinchi;
    Log uchinchi;
public:
    Tartib() : uchinchi("3"), ikkinchi("2"), birinchi("1") {}
};
int main() {
    Tartib t;
}
```

▶ NATIJA

```
1 2 3
```

Ro‘yxat teskari yozilgan

lekin bu tartibni o‘zgartirmaydi

E’lon tartibi

birinchi, ikkinchi, uchinchi

Xavf

bir maydon boshqasiga tayansa, e’lon tartibi to‘g‘ri bo‘lishi shart

EXPLICIT

Yashirin o‘zgartirishni taqiqlash

```
class Metr {
    double q;
public:
    explicit Metr(double q) : q(q) {}
    double get() const { return q; }
};
class Sekund {
    int q;
public:
    Sekund(int q) : q(q) {}
    int get() const { return q; }
};
void chopM(Metr m) { cout << m.get() << "m "; }
void chopS(Sekund s) { cout << s.get() << "s "; }
int main() {
    chopM(Metr(5.5));
    chopS(90);
    chopS('A');
}
```

▶ NATIJA

```
5.5m 90s 65s
```

chopM(5.5)

XATO — explicit yashirin o‘zgartirishga yo‘l qo‘ymaydi

chopS(90)

int → Sekund jimgina

chopS('A')

char → int → Sekund: 65 sekund (!)

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

2-bo‘lim yakuni: Initsializatsiya ro‘yxati va explicit

1

Maydonlar qaysi tartibda initsializatsiya qilinadi?

Savol 1

2

const maydonga qiymat qayerda beriladi?

Savol 2

3

explicit nimani taqiqlaydi?

Savol 3

AMALIY TOPSHIRIQ

Har kim o‘z qurilmasida · 10 daqiqa · Oson · 10 ball

2-topshiriq: Initsializatsiya ro‘yxati: Oraliq

`Oraliq` sinfida ikkita **o‘zgarmas** maydon bor: `const int a, b`. Ularga faqat initsializatsiya ro‘yxatida qiymat berish mumkin. Konstruktor `Oraliq(int x, int y)` kichigini `a` ga, kattasini `b` ga yozsin. Metodlar: `int uzunlik() const` (`b - a`) va `bool ichida(int t) const` (`a ≤ t ≤ b`).

Kirish: Birinchi qatorda `x y`, ikkinchi qatorda `q` (1 ≤ q ≤ 100), keyin `q` ta son `t`.

Chiqish: Birinchi qatorda uzunlik, keyin har bir `t` uchun `ICHIDA` yoki `TASHQARIDA`.

MISOL · KIRISH

```
8 3
3
5
9
3
```

MISOL · CHIQISH

```
5
ICHIDA
TASHQARIDA
ICHIDA
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. Kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

03

Nusxa konstruktori

Qachon chaqiriladi · sayoz nusxa muammosi · chuqur nusxa yechimi

T(const T&)

Sayoz

Chuqur

double free

QACHON?

Nusxa konstruktori uch holatda chaqiriladi

1 Initsializatsiya

`Massiv b = a;` yoki `Massiv b(a);`

2 Qiymat bo‘yicha uzatish

`void f(Massiv m);` → `f(a)`

3 Qiymat bo‘yicha qaytarish

`return m;` — ko‘pincha kompilyator optimallashtiradi (copy elision).

Kompilyator yaratgan nusxa konstruktori maydonlarni birma-bir nusxalaydi — ko‘rsatkich uchun faqat manzil nusxalanadi.

Ko‘rsatkichli maydonni nusxalash

Xato: sayoz nusxa (kompilyatorniki)

- `b.data = a.data` — ikkalasi bitta blokni ko‘rsatadi
- `b.data[0] = 99` → `a.data[0]` ham 99!
- Blok tugagach `~b()` blokni o‘chiradi, keyin `~a()` yana o‘chiradi
- Natija: “double free detected” — dastur buziladi

To‘g‘ri: chuqur nusxa (o‘zimiz yozamiz)

- `Massiv(const Massiv &b) : data(new int[b.n]), n(b.n)`
- Yangi blok ajratiladi va elementlar nusxalanadi
- a va b mustaqil — biri o‘zgarsa, ikkinchisi o‘zgarmaydi
- Har bir destruktor o‘z blokini o‘chiradi

Qoida: sinf ko‘rsatkich orqali resursga (xotira, fayl) egalik qilsa — o‘z nusxa konstruktoringizni yozing.

Mustaqil nusxa

```
class Massiv {
    int *d;
    int n;
public:
    Massiv(int n) : d(new int[n]), n(n) {
        for (int i = 0; i < n; i++) d[i] = i + 1;
    }
    Massiv(const Massiv &b) : d(new int[b.n]), n(b.n) {
        for (int i = 0; i < n; i++) d[i] = b.d[i];
        cout << "nusxa ";
    }
    ~Massiv() { delete[] d; }
    int &at(int i) { return d[i]; }
};
int main() {
    Massiv a(3);
    Massiv b = a;
    b.at(0) = 99;
    cout << a.at(0) << " " << b.at(0);
}
```

▶ NATIJA

```
nusxa 1 99
```

`new int[b.n]`

nusxa uchun yangi blok

`d[i] = b.d[i]`

elementlarni birma-bir nusxalash

`a.at(0) = 1`

asl obyekt o‘zgarmadi

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

3-bo‘lim yakuni: Nusxa konstruktori

1

Qachon o‘z nusxa konstruktorimizni yozish shart?

Savol 1

2

Sayoz nusxada destruktorlar qanday muammo keltiradi?

Savol 2

3

Funksiyaga obyektni nusxasiz qanday uzatish mumkin?

Savol 3

AMALIY TOPSHIRIQ

Har kim o‘z qurilmasida · 10 daqiqa · O‘rta · 20 ball

3-topshiriq: Chuqur nusxa

`Massiv` sinfi `int* d` dinamik xotirani boshqaradi. **Nusxa konstruktori**ni yozing: u yangi xotira ajratib, elementlarni ko‘chirsin (chuqur nusxa). `main()` nusxani o‘zgartiradi — asl massiv o‘zgarmasligi kerak. Sayoz nusxada ikkala obyekt bitta xotirani ko‘rsatadi va destruktor uni ikki marta o‘chiradi.

Kirish: Birinchi qatorda `n` ($1 \le n \le 1000$), ikkinchi qatorda `n` ta butun son.

Chiqish: Ikki qator: asl massiv va har bir elementi 2 ga ko‘paytirilgan nusxa.

MISOL · KIRISH	MISOL · CHIQISH
3 1 2 3	1 2 3 2 4 6

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. Kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

04

operator= va Uchlik qoidasi

Nusxa konstruktori va nusxalash operatori farqi · o‘z-o‘ziga o‘zlashtirish · = delete

operator=

this == &b

Rule of 3

= delete

Massiv& operator=(const Massiv &b)

1

O‘z-o‘ziga?

```
if (this == &b)
return *this;
```

2

Eskisini bo‘shatish

```
delete[] data;
```

3

Yangi resurs

```
n = b.n; data =
new int[n];
```

4

Nusxalash

```
for (...)
data[i] =
b.data[i];
```

5

Zanjir uchun

```
return *this; →
a = b = c;
```

`Massiv b = a;` — nusxa konstruktori (b yangi); `c = a;` — operator= (c allaqachon mavjud).

UCHLIK QOIDASI

Rule of Three

Destruktor, nusxa konstruktori yoki nusxalash operatoridan birortasini o‘zingiz yozishingiz kerak bo‘lsa, deyarli har doim uchalasini ham yozing.

~T()

Resursni bo‘shatadi — demak sinf resursga ega.

T(const T&)

Yangi obyektga resursning mustaqil nusxasi.

T& operator=(const T&)

Mavjud obyektga: eskini bo‘shatish + nusxalash.

Nusxalash ma’nosiz bo‘lsa (yagona bilet, ochiq fayl): `T(const T&) = delete;` va `T& operator=(const T&) = delete;`

AMALDA

Konstruktor yoki operator= ?

```
struct T {
    T() { cout << "yangi "; }
    T(const T &) { cout << "nusxa "; }
    T& operator=(const T &b) {
        if (this == &b) { cout << "o'zi "; return *this; }
        cout << "op= ";
        return *this;
    }
};
int main() {
    T a;
    T b = a;
    T c;
    c = a;
    a = a;
}
```

▶ NATIJA

```
yangi nusxa yangi op= o'zi
```

`T b = a;`

b hozir yaratilmoqda — nusxa konstruktori

`c = a;`

c mavjud — operator=

`a = a;`

o‘z-o‘ziga o‘zlashtirish tekshiruvi

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

4-bo‘lim yakuni: operator= va Uchlik qoidasi

1

“Uchlik qoidasi” nimani bildiradi?

Savol 1

2

`Massiv c(5); c = a;` da qaysi funksiya chaqiriladi?

Savol 2

3

operator= da nima uchun `this == &b` tekshiriladi?

Savol 3

AMALIY TOPSHIRIQ

Har kim o‘z qurilmasida · 10 daqiqa · O‘rta · 20 ball

4-topshiriq: Uchlik qoidasi: operator=

Oldingi `Massiv` sinfiga **nusxa o‘zlashtirish operatori** `operator=` ni qo‘shing (Uchlik qoidasi: destruktor + nusxa konstruktori + `operator=`). O‘z-o‘ziga o‘zlashtirish (`a = a`) ham to‘g‘ri ishlashi kerak. Massivlar o‘lchami har xil bo‘lishi mumkin.

Kirish: Birinchi qatorda `n` va `n` ta son (a massiv), ikkinchi qatorda `m` va `m` ta son (b massiv). 1 ≤ n, m ≤ 1000.

Chiqish: Uch qator: `a = b; a[0] = 0;` dan keyingi `a`, o‘zgarmagan `b`, va `a = a;` dan keyingi `a`.

MISOL · KIRISH

```
3 1 2 3
2 7 8
```

MISOL · CHIQISH

```
0 8
7 8
0 8
```

Telefon kamerasi bilan skanerlang

1. Saytga kiring (login va parol)
2. Kodni yozing va sinab ko‘ring
3. ✓ Topshirish — natija o‘qituvchiga

05

Ko‘chirish semantikasi

lvalue va rvalue · T&& · std::move · noexcept · Beshlik qoidasi

T&&

std::move

noexcept

Rule of 5

Nusxalash o‘rniga “o‘g‘irlash”

Tushuncha	Izoh
lvalue	Nomi va manzili bor obyekt: `a`, `arr[i]` — nusxalanadi
rvalue	Vaqtinchalik qiymat: `Massiv(5)`, `f()` natijasi — resursini olish mumkin
`T&&`	rvalue havola — faqat vaqtinchalik obyektlarga bog‘lanadi
`std::move(a)`	Hech narsani ko‘chirmaydi — a ni rvalue deb belgilaydi

Ko‘chirishdan keyin manba “bo‘sh, lekin yaroqli” holatda qoladi: `b.data = nullptr; b.n = 0;`

Faqat ko‘rsatkich olinadi

```
class Massiv {
    int *d; int n;
public:
    Massiv(int n) : d(new int[n]), n(n) {}
    Massiv(const Massiv &b) : d(new int[b.n]), n(b.n) { cout
            << "NUSXA "; }
    Massiv(Massiv &&b) noexcept : d(b.d), n(b.n) {
        b.d = nullptr; b.n = 0;
        cout << "KO'CHIRISH ";
    }
    ~Massiv() { delete[] d; }
    int size() const { return n; }
};
int main() {
    Massiv a(1000000);
    Massiv b = a;
    Massiv c = std::move(a);
    cout << endl << a.size() << " " << c.size();
}
```

▶ NATIJA

```
NUSXA KO'CHIRISH
0 1000000
```

`Massiv b = a;`

a — lvalue: chuqur nusxa

`std::move(a)`

a ni rvalue deb belgilaydi → move konstruktor

`a.size() = 0`

a bo‘sh, lekin yaroqli

vector nima uchun noexcept ga qaraydi?

```
struct Tez {
    static int n, k;
    Tez() {}
    Tez(const Tez &) { n++; }
    Tez(Tez &&) noexcept { k++; }
};
struct Sekin {
    static int n, k;
    Sekin() {}
    Sekin(const Sekin &) { n++; }
    Sekin(Sekin &&) { k++; }
};
int Tez::n = 0, Tez::k = 0, Sekin::n = 0, Sekin::k = 0;
int main() {
    vector<Tez> a; vector<Sekin> b;
    for (int i = 0; i < 100; i++) { a.push_back(Tez());
            b.push_back(Sekin()); }
    cout << Tez::n << " " << Sekin::n;
}
```

▶ NATIJA

```
0 127
```

noexcept bor

vector kengayganda eski elementlarni ko‘chiradi: 0 nusxa

noexcept yo‘q

xavfsizlik uchun nusxalaydi: 127 nusxa

Beshlik qoidasi

Uchlik + move konstruktor + ko‘chiruvchi operator=

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

5-bo‘lim yakuni: Ko‘chirish semantikasi

1

`Massiv b = std::move(a);` dan keyin a qanday holatda?

Savol 1

2

std::move nimani ko‘chiradi?

Savol 2

3

Beshlik qoidasidagi beshta metodni ayting.

Savol 3

06

RAII va Nol qoidasi

Destruktor kafolati · istisnoda ham · RAII sinflari · Rule of Zero

RAII | throw | Rule of 0 | string, vector

Resursni olish — bu initsializatsiya

Resurs (xotira, fayl, qulf, ulanish) konstruktorda olinadi va destruktorda bo‘shatiladi — shunda uni bo‘shatishni unutib bo‘lmaydi.

Kafolat

Destruktor blokdan oddiy chiqishda, erta `return` da va hatto istisnoda ham chaqiriladi.

Standart RAII sinflari

`string`, `vector`, `unique_ptr`, `fstream`.

Destruktor qoidalari

Istisno tashlamasin; asosiy sinfda virtual bo‘lsin; qo‘lda chaqirilmasin.

Qo‘lda `new`/`delete` qilingan resurs erta `return` yoki `throw` da “sizib ketadi”.

ISTISNODA HAM

Destruktor kafolatli chaqiriladi

```
struct Resurs {
    string n;
    Resurs(string n) : n(n) { cout << "+" << n << " "; }
    ~Resurs() { cout << "-" << n << " "; }
};
void ishla(bool xato) {
    Resurs r("fayl");
    if (xato) throw runtime_error("uzildi");
    cout << "ok ";
}
int main() {
    ishla(false);
    try { ishla(true); }
    catch (const exception &e) { cout << e.what(); }
}
```

▶ NATIJA

```
+fayl ok -fayl +fayl -fayl uzildi
```

ishla(false)

oddiy tugash — destruktor

throw

istisno uchadi, lekin avval r ning destruktori ishlaydi

catch

istisno ushlanadi — resurs allaqachon bo‘shatilgan

NOL QOIDASI

Resursni o‘zingiz boshqarmang

```
class Talaba {
    string ism;
    vector<int> baholar;
public:
    Talaba(string i, vector<int> b) : ism(i), baholar(b) {}
    void qosh(int x) { baholar.push_back(x); }
    int soni() const { return baholar.size(); }
};
int main() {
    Talaba t1("Aziza", {5, 4, 5});
    Talaba t2 = t1;
    t2.qosh(3);
    cout << t1.soni() << " " << t2.soni();
}
```

▶ NATIJA

```
3 4
```

Maxsus metodlar yo‘q

destruktor, nusxa, operator= — hech biri yozilmagan

`Talaba t2 = t1;`

kompilyator nusxasi — string va vector o‘zi chuqur nusxalanadi

Rule of Zero

maydonlar RAII sinflari bo‘lsa, hech narsa yozish shart emas

TUSHUNISHNI TEKSHIRISH

O‘ylang – Juftlikda – Bo‘lishing · 3 daqiqa

6-bo‘lim yakuni: RAII va Nol qoidasi

1

RAII tamoyilining mohiyati?

Savol 1

2

Istisno yuz berganda lokal obyektlarning destruktori chaqiriladimi?

Savol 2

3

Nol qoidasi nima?

Savol 3

XULOSA

Asosiy xulosalar

1 Konstruktor turlari

Standart, parametrli, nusxa, ko‘chirish, delegatsiya; `= default`, `= delete`.

2 Initsializatsiya ro‘yxati

const, havola va asosiy sinf uchun majburiy; tartib — e’lon tartibi.

3 explicit

Bitta argumentli konstruktorda yashirin o‘zgartirishni taqiqlaydi.

4 Chuqur nusxa

Ko‘rsatkichli resurs → o‘z nusxa konstruktori va operator=.

5 Uchlik va Beshlik

Destruktor + nusxa + operator= (+ move konstruktor + move operator=).

6 RAII va Nol qoidasi

Resurs obyekt bilan yashaydi; string/vector ishlating — hech narsa yozmang.

TAKRORLASH

Yakunlashdan oldin sakkizta savol

1. `Nuqta p(3, 4);` qaysi konstruktor?
2. Delegatsiya qiluvchi konstruktor misoli?
3. Havola maydon qayerda bog‘lanadi?
4. `void f(T t); f(a);` da nima chaqiriladi?
5. Sayoz nusxaning xavfi?
6. `c = a;` (c mavjud) — nima chaqiriladi?
7. Move konstruktor parametri?
8. RAII sinfiga misol?

LABORATORIYA ISHI

10-laboratoriya ishi: 6 ta dasturlash topshirig‘i

5 **Ko‘chirish semantikasi: Bufer**
O‘rta · 20 ball

6 **Obyektlar hayoti: bloklar**
O‘rta · 20 ball

7 **Matritsa (Uchlik qoidasi)**
Qiyin · 30 ball

8 **O‘z satr sinfim**
Qiyin · 30 ball

9 **Nol qoidasi: Guruh**
Oson · 10 ball

10 **RAII: resurs qulfi**
O‘rta · 20 ball

Telefon kamerasi bilan skanerlang

Kod saytda yoziladi va server tomonidan avtomatik tekshiriladi.

O‘qituvchi kodingizni ko‘rib, baho qo‘yadi.

CHIQISH CHIPTASI

Darsni tark etishdan oldin

Javoblaringizni qog‘ozga yozing va o‘qituvchiga topshiring.

1

Sayoz va chuqur nusxa farqini o‘z o‘xshatishingiz bilan tushuntiring.

2

Qaysi holatda nusxalashni `= delete` bilan taqiqlaysiz? Misol keltiring.

3

Beshlik qoidasidan qaysi metod sizga eng qiyin tuyuldi?