16

O‘N OLTINCHI MA’RUZA

Dasturlash 1 · C++ · 11-mavzu

Inkapsulyatsiya va merosxo‘rlik

Invariant va interfeys · friend va mutable · nomlarni yashirish va kompozitsiya · obyekt kesilishi · ko‘p meros · LSP va final

Invariant

friend

has-a / is-a

virtual meros

ISINISH

“Kvadrat — bu to‘rtburchak.” Dasturda ham shundaymi?

Matematikada to‘g‘ri bo‘lgan gap kodda muammo keltirishi mumkin. Taxmin qiling.

1

To‘rtburchakning enini 5, bo‘yini 4 qilsak, yuzasi nechaga teng bo‘lishini kutamiz?

2

Kvadratning enini o‘zgartirsak, bo‘yi ham o‘zgarishi kerakmi?

3

Avtomobil “dvigatel”dan meros oladimi yoki unda dvigatel bormi?

O‘QUV MAQSADLARI

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

1. **Ta’riflash** — sinf invariantini va interfeys/amalga oshirish farqini
2. **Himoya qilish** — invariantni konstruktor va tekshiruvchi metodlar bilan
3. **Qo‘llash** — friend funksiya/sinf va mutable maydonlarni asosli holatlarda
4. **Tanlash** — merosxo‘rlik (is-a) va kompozitsiya (has-a) orasidan
5. **Tushuntirish** — nomlarni yashirishni va using bilan tiklashni
6. **Oldini olish** — obyekt kesilishining (slicing)
7. **Hal qilish** — ko‘p merosdagi nom to‘qnashuvi va olmos muammosini
8. **Qo‘llash** — Liskov tamoyili, override, final va interfeyslarni

01

Invariant va interfeys

Obyekt doim to‘g‘ri holatda · interfeys va amalga oshirish · istisno bilan himoya

Invariant | Interfeys | throw | private

Inkapsulyatsiyaning asl maqsadi

Tushuncha	Ma’nosi	Misol
Invariant	Obyekt hayoti davomida doim rost bo‘lishi kerak bo‘lgan shart	Sana: 1 ≤ oy ≤ 12
Interfeys	Tashqariga ochiq qism — nima qila oladi	`push()`, `pop()`, `top()`
Amalga oshirish	Yashirin qism — qanday ishlaydi	Stek ichida massiv

Invariant konstruktorda o‘rnatiladi (noto‘g‘ri argument — istisno), har bir public metod uni saqlaydi: avval tekshiradi, keyin o‘zgartiradi.

SANA SINFI

Noto‘g‘ri sana umuman yaratilmaydi

```
class Sana {
    int kun, oy;
    static void tekshir(int k, int o) {
        if (o < 1 || o > 12) throw invalid_argument("oy xato");
        if (k < 1 || k > 31) throw invalid_argument("kun xato");
    }
public:
    Sana(int k, int o) : kun(k), oy(o) { tekshir(k, o); }
    void setOy(int o) { tekshir(kun, o); oy = o; }
    int getOy() const { return oy; }
};
int main() {
    Sana s(15, 3);
    try { s.setOy(13); } catch (const exception &e) { cout
            << e.what() << " "; }
    cout << s.getOy();
    try { Sana x(1, 0); } catch (const exception &e) { cout
            << " " << e.what(); }
}
```

▶ NATIJA

```
oy xato 3 oy xato
```

`tekshir(k, o)`

konstruktor invariantni o‘rnatadi

`setOy(13)`

avval tekshirish — rad etildi, oy o‘zgarmadi (3)

`Sana x(1, 0)`

noto‘g‘ri obyekt yaratilmaydi — istisno

Stek: foydalanuvchi ichkarini bilmaydi

```
class Stek {
    vector<int> d;
public:
    void push(int x) { d.push_back(x); }
    int pop() {
        if (d.empty()) throw underflow_error("bo'sh");
        int x = d.back(); d.pop_back(); return x;
    }
    int size() const { return d.size(); }
};
int main() {
    Stek s;
    s.push(5); s.push(8); s.push(3);
    cout << s.pop() << s.pop() << " " << s.size() << " ";
    s.pop();
    try { s.pop(); } catch (const exception &e) { cout
            << e.what(); }
}
```

▶ NATIJA

```
38 1 bo'sh
```

`push / pop / size`

interfeys — faqat shu uchta amal

`vector<int> d`

amalga oshirish — massiv, ro‘yxat... o‘zgartirish mumkin

`underflow_error`

bo‘sh stekdan olish — invariant himoyasi

TUSHUNISHNI TEKSHIRISH

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

1-bo‘lim yakuni: Invariant va interfeys

1

Sinf invarianti nima?

Savol 1

2

Interfeys va amalga oshirishni ajratishning foydasi?

Savol 2

3

Noto‘g‘ri argument bilan obyekt yaratilmasligini qanday ta’minlaysiz?

Savol 3

AMALIY TOPSHIRIQ

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

1-topshiriq: Sana: invariantni saqlash

`Sana` sinfi har doim **to‘g‘ri** sanani saqlaydi (invariant). `bool ornat(int k, int o, int y)` — sana to‘g‘ri bo‘lsa o‘rnatadi va `true` qaytaradi, aks holda eski qiymat saqlanib qoladi va `false` qaytaradi. `void chiqar() const` — `dd.mm.yyyy`. Boshlang‘ich qiymat — `01.01.2000`.

Kirish: Birinchi qatorda `q` (1 ≤ q ≤ 100), keyin `q` qatorda `kun oy yil` (1 ≤ yil ≤ 9999).

Chiqish: Har bir urinishdan keyin: rad etilsa `XATO`, so‘ng joriy sana `dd.mm.yyyy` ko‘rinishida.

MISOL · KIRISH	MISOL · CHIQISH
3 15 3 2025 30 2 2024 29 2 2024	15.03.2025 XATO 15.03.2025 29.02.2024

Telefon kamerasi bilan skanerlang

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

02

friend va mutable

Do‘st funksiya va sinf · do‘stlik qoidalari · mutable kesh va hisoblagich

friend | Do‘st sinf | mutable | const metod

Yopiq a’zolarga ruxsat — sinfning o‘zi beradi

1 friend funksiya

`friend double masofa(const Vektor&, const Vektor&);` — a’zo emas, lekin private ga kiradi.

2 friend sinf

`friend class Auditor;` — Auditor ning barcha metodlari kiradi.

3 Meros qilinmaydi

Do‘st sinfning hosilasi avtomatik do‘st bo‘lmaydi.

4 O‘zaro emas

A B ni do‘st desa, B A ni do‘st degani emas.

5 Tranzitiv emas

Do‘stning do‘sti — do‘st emas.

6 Qachon?

Operatorlar (14-dars), ikki sinf yaqin hamkorlik qilganda. Kam ishlating.

FRIEND AMALDA

Masofa funksiyasi — do‘st

```
class Vektor {
    double x, y;
public:
    Vektor(double x, double y) : x(x), y(y) {}
    friend double masofa(const Vektor &a, const Vektor &b);
};
double masofa(const Vektor &a, const Vektor &b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}
int main() {
    Vektor p(0, 0), q(3, 4);
    cout << masofa(p, q);
}
```

▶ NATIJA

5

`friend ...`

e’lon sinf ichida, ta’rif tashqarida

`a.x`

private maydonga kirish — do‘st bo‘lgani uchun

Chaqiruv

`masofa(p, q)` — oddiy funksiya kabi, `p.masofa` emas

MUTABLE

const metodda ham o‘zgaradigan maydon

```cpp
class Hujjat {
    string matn;
    mutable int oqishlar = 0;
    mutable int kesh = -1;
public:
    Hujjat(string m) : matn(m) {}
    int sozlar() const {
        oqishlar++;
        if (kesh == -1) {
            istringstream ss(matn); string s; kesh = 0;
            while (ss >> s) kesh++;
        }
        return kesh;
    }
    int getOqish() const { return oqishlar; }
};
int main() {
    const Hujjat h("C++ juda kuchli til");
    cout << h.sozlar() << h.sozlar() << " " << h.getOqish();
}
```

▶ NATIJA

```
44 2
```

mutable

mantiqiy holatga kirmaydigan maydonlar: hisoblagich, kesh

sozlar() const

const obyekt uchun ham chaqiriladi

kesh

birinchi chaqiruvda hisoblanadi, keyin tayyor

TUSHUNISHNI TEKSHIRISH

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

2-bo‘lim yakuni: friend va mutable

1

Do‘stlik meros qilinadimi?

Savol 1

2

A sinfi B ni friend desa, A B ning private iga kira oladimi?

Savol 2

3

mutable maydon qachon kerak?

Savol 3

AMALIY TOPSHIRIQ

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

2-topshiriq: friend funksiya: masofa

`Vektor2` sinfining `x`, `y` maydonlari yopiq. Sinfdan tashqaridagi `double masofa(const Vektor2&, const Vektor2&)` funksiyasini sinfga **do‘st** (`friend`) qilib e’lon qiling va yozing — u ikki nuqta orasidagi masofani qaytarsin.

Kirish: Birinchi qatorda `q` ($1 \le q \le 100$), keyin `q` qatorda `x1 y1 x2 y2` (haqiqiy sonlar).

Chiqish: Har bir juftlik uchun masofa, verguldan keyin 3 xona.

MISOL · KIRISH	MISOL · CHIQISH

`2`
`0 0 3 4`
`1 1 1 1`
	`5.000`
	`0.000`

Telefon kamerasi bilan skanerlang

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

03

Meros turlari va kompozitsiya

Oddiy, ko‘p bosqichli, ierarxik · nomlarni yashirish · using · is-a va has-a

A → B → C

using

is-a

has-a

Sinflar qanday bog‘lanadi?

Turi	Tuzilishi	Misol
Oddiy	A → B	Xodim → Dasturchi
Ko‘p bosqichli	A → B → C	Jonzot → Hayvon → It
Ierarxik	A → B, A → C	Shakl → Doira, Shakl → Kvadrat
Ko‘p (multiple)	A, B → C	Uchuvchi, Suzuvchi → O‘rdak

Nomlarni yashirish: hosila sinfdagi `chop(string)` asosiy sinfdagi BARCHA `chop(...)` larni yashiradi. Tiklash: `using Printer::chop;`

using bilan qaytarish

```
struct Printer {
    void chop(int x) { cout << "int "; }
    void chop(double x) { cout << "double "; }
};
struct Rangli : Printer {
    void chop(const string &s) { cout << "string "; }
};
struct Tuzatilgan : Printer {
    using Printer::chop;
    void chop(const string &s) { cout << "string "; }
};
int main() {
    Rangli r;
    r.chop("a");
    r.Printer::chop(5);
    Tuzatilgan t;
    t.chop(5); t.chop(2.5); t.chop("b");
}
```

▶ NATIJA

```
string int int double string
```

`r.chop(5)`

XATO — Printer::chop(int) yashirilgan

`r.Printer::chop(5)`

aniq ko‘rsatish bilan mumkin

`using Printer::chop`

asosiy versiyalar qaytadi — hammasi ishlaydi

“... bu ...dir” yoki “...da ... bor”?

Merosxo‘rlik (is-a)

- Doira — bu Shakl: `class Doira : public Shakl`
- Elektromobil — bu Avtomobil
- Hosila asosiy sinf o‘rnida ishlatila olishi kerak

Kompozitsiya (has-a)

- Avtomobilda Dvigatel BOR: `Dvigatel dvigatel;` maydoni
- Kompyuterda Protsessor bor
- Sinflar kamroq bog‘lanadi — o‘zgartirish oson

Shubha tug‘ilganda kompozitsiyani tanlang.

TUSHUNISHNI TEKSHIRISH

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

3-bo‘lim yakuni: Meros turlari va kompozitsiya

1

Avtomobil va Dvigatel orasidagi to‘g‘ri munosabat?

Savol 1

2

Hosila sinfda `f(string)` yozilsa, asosiy `f(int)` ga nima bo‘ladi?

Savol 2

3

Yashirilgan asosiy metodlarni qanday qaytarish mumkin?

Savol 3

AMALIY TOPSHIRIQ

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

3-topshiriq: Kompozitsiya: Kompyuter

`Protsessor` (nom, chastota GHz, narx) va `Xotira` (hajm GB, narx) sinflarini yozing. `Kompyuter` sinfi ularni **o‘z ichiga oladi** (kompozitsiya, “has-a”) va ularning narxi ustiga yig‘ish narxini qo‘shadi. `main()` tayyor.

Kirish: Bitta qatorda: protsessor nomi, chastota, protsessor narxi, xotira hajmi, xotira narxi, yig‘ish narxi.

Chiqish: Ikki qator: `nom chastotaGHz hajmGB` (chastota — 1 xona) va umumiy narx.

MISOL · KIRISH

```
Ryzen5 3.7 2100000 16 900000 300000
```

MISOL · CHIQISH

```
Ryzen5 3.7GHz 16GB
3300000
```

Telefon kamerasi bilan skanerlang

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

04

Konstruktorlar va obyekt kesilishi

Ko‘p bosqichli meros tartibi · using Asosiy::Asosiy · slicing

Tartib | Meros konstruktor | Slicing | & / *

Qatlamlar: yuqoridan pastga, teskari o‘chirish

```
struct Jonzot {
    Jonzot() { cout << "J "; }
    ~Jonzot() { cout << "~J "; }
};
struct Hayvon : Jonzot {
    Hayvon() { cout << "H "; }
    ~Hayvon() { cout << "~H "; }
};
struct It : Hayvon {
    It() { cout << "I "; }
    ~It() { cout << "~I "; }
};
int main() {
    It it;
    cout << "| ";
}
```

▶ NATIJA

```
J H I | ~I ~H ~J
```

Konstruktorlar

eng yuqori asosiy sinfdan boshlab

Destruktorlar

teskari tartibda

`using Asosiy::Asosiy;`

C++11: asosiy konstruktorlarni meros olish

SLICING

Hosila qism “kesib tashlanadi”

```
struct Hayvon {
    virtual ~Hayvon() = default;
    virtual string ovoz() const { return "..."; }
};
struct It : Hayvon {
    string ovoz() const override { return "Vov"; }
};
void qiymat(Hayvon h) { cout << h.ovoz() << " "; }
void havola(const Hayvon &h) { cout << h.ovoz() << " "; }
int main() {
    It it;
    Hayvon h = it;
    Hayvon &r = it;
    cout << h.ovoz() << " " << r.ovoz() << " ";
    qiymat(it); havola(it);
}
```

▶ NATIJA

```
... Vov ... Vov
```

`Hayvon h = it;`

qiymat bo‘yicha nusxa — faqat Hayvon qismi

`Hayvon &r = it;`

havola — kesilish yo‘q

`qiymat(it)`

parametr qiymat bo‘yicha — yana kesildi

▶ INTERAKTIV

Kesilish bo‘ladimi?

Har bir ifodani to‘g‘ri qutiga joylashtiring (It — Hayvon ning hosilasi). (Interaktiv slaydlarda bosib bajariladi.)

```
Hayvon h = it;
```

```
Hayvon &h = it;
```

```
Hayvon *h = &it;
```

```
void f(Hayvon h); f(it);
```

```
void f(const Hayvon &h); f(it);
```

```
vector<Hayvon> v; v.push_back(it);
```

```
vector<Hayvon*> v; v.push_back(&it);
```

```
unique_ptr<Hayvon> p = make_unique<It>();
```

Kesilish bor

Kesilish yo‘q

TUSHUNISHNI TEKSHIRISH

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

4-bo‘lim yakuni: Konstruktorlar va obyekt kesilishi

1

Obyekt kesilishi qachon yuz beradi?

Savol 1

2

A → B → C merosda C yaratilganda konstruktorlar tartibi?

Savol 2

3

Kesilishning oldini qanday olish mumkin?

Savol 3

AMALIY TOPSHIRIQ

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

4-topshiriq: Konstruktorlar tartibi

`Shaxs` (ism) asosiy sinfi va undan meros olgan `Talaba` (guruh) sinfini yozing. Konstruktor va destruktorlar xabar chiqarsin: `Shaxs(ism)`, `Talaba(ism, guruh)`, `~Talaba(ism)`, `~Shaxs(ism)`. `Talaba` konstruktori asosiy sinf konstruktorini initsializatsiya ro‘yxatida chaqirsin. `main()` tayyor.

Kirish: Birinchi qatorda `n` ($1 \le n \le 10$), keyin `n` qatorda: ism va guruh.

Chiqish: Konstruktor va destruktorlar chaqirilish tartibi (`main()` obyektlarni blok ichida yaratadi).

MISOL · KIRISH

```
2
Ali 911
Vali 912
```

MISOL · CHIQISH

```
Shaxs(Ali)
Talaba(Ali, 911)
Shaxs(Vali)
Talaba(Vali, 912)
---
~Talaba(Vali)
~Shaxs(Vali)
~Talaba(Ali)
~Shaxs(Ali)
```

Telefon kamerasi bilan skanerlang

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

05

Ko‘p merosxo‘rlik va olmos

Bir nechta asosiy sinf · nom to‘qnashuvi · olmos muammosi · virtual meros

A, B → C

Uchuvchi::

Olmos

virtual public

KO‘P MEROS

O‘rdak ham uchadi, ham suzadi

```
struct Uchuvchi {
    void uch() { cout << "uchdi "; }
    void harakat() { cout << "havoda "; }
};
struct Suzuvchi {
    void suz() { cout << "suzdi "; }
    void harakat() { cout << "suvda "; }
};
struct Ordak : Uchuvchi, Suzuvchi {
    void harakat() {
        Uchuvchi::harakat();
        Suzuvchi::harakat();
    }
};
int main() {
    Ordak o;
    o.uch(); o.suz(); o.harakat();
    o.Suzuvchi::harakat();
}
```

▶ NATIJA

```
uchdi suzdi havoda suvda suvda
```

`Ordak : Uchuvchi, Suzuvchi`

ikki asosiy sinf

`harakat()`

ikkalasida bor — o‘z versiyasi bo‘lmasa noaniq

`o.Suzuvchi::harakat()`

aniq asosiy sinf versiyasi

OLMOS MUAMMOSI

Qurilma → Printer, Skaner → MFU

1

Oddiy meros

MFU ichida Qurilma ikki marta: Printer orqali va Skaner orqali

2

Muammo

`m.id` noaniq; ikki nusxa farq qilib qolishi mumkin

3

Yechim

`class Printer : virtual public Qurilma` — oraliq sinflarda

4

Natija

Qurilma bir marta; uni eng oxirgi hosila (MFU) yaratadi

Virtual meros qo‘shimcha yashirin ko‘rsatkich qo‘shadi — faqat olmos haqiqatan paydo bo‘lganda ishlating.

Bitta umumiy nusxa

```
struct Q { int id = 0; Q() { cout << "Q "; } };
struct P : Q { P() { cout << "P "; } };
struct S : Q { S() { cout << "S "; } };
struct M : P, S {};
struct VQ { int id = 0; VQ() { cout << "VQ "; } };
struct VP : virtual VQ {};
struct VS : virtual VQ {};
struct VM : VP, VS {};
int main() {
    M m;
    m.P::id = 5;
    cout << "| " << m.S::id << endl;
    VM v;
    v.id = 5;
    cout << "| " << v.VS::id;
}
```

▶ NATIJA

```
Q P Q S | 0
VQ | 5
```

Q ikki marta

oddiy merosda ikki nusxa: P::id = 5, S::id = 0

VQ bir marta

virtual merosda bitta nusxa

`v.id = 5`

noaniqlik yo‘q

TUSHUNISHNI TEKSHIRISH

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

5-bo‘lim yakuni: Ko‘p merosxo‘rlik va olmos

1

Olmos muammosi qanday hal qilinadi?

Savol 1

2

Ikkala asosiy sinfda bir xil nomli metod bo‘lsa nima qilinadi?

Savol 2

3

Virtual merosning narxi nimada?

Savol 3

06

Loyihalash tamoyillari

Liskov (LSP) · Kvadrat–To‘rtburchak · override va final · interfeyslar

LSP | override | final | Interfeys

LSP BUZILISHI

Kvadrat to‘rtburchakning o‘rnini bosa olmaydi

```
struct Tortburchak {
    double e, b;
    Tortburchak(double e, double b) : e(e), b(b) {}
    virtual ~Tortburchak() = default;
    virtual void setEni(double x) { e = x; }
    virtual void setBoyi(double x) { b = x; }
    double yuza() const { return e * b; }
};
struct Kvadrat : Tortburchak {
    Kvadrat(double a) : Tortburchak(a, a) {}
    void setEni(double x) override { e = b = x; }
    void setBoyi(double x) override { e = b = x; }
};
void kengaytir(Tortburchak &t) {
    t.setEni(5); t.setBoyi(4);
    cout << t.yuza() << " ";
}
int main() {
    Tortburchak t(2, 3); Kvadrat k(3);
    kengaytir(t); kengaytir(k);
}
```

▶ NATIJA

```
20 16
```

kutilma

eni 5, bo‘yi 4 → yuza 20

Kvadrat

setBoyi(4) enini ham 4 qiladi → 16

LSP

hosila obyekt asosiy kutilgan joyda kutilmagan natija bermasligi kerak

VOSITALAR

override, final va interfeys

Vosita	Vazifasi
`override`	Virtual metodni qayta aniqlayotganini bildiradi; imzo xato bo‘lsa — kompilyatsiya xatosi
`final` (metod)	Keyingi hosilalarda qayta aniqlashni taqiqlaydi
`final` (sinf)	`class Karta final` — undan meros olib bo‘lmaydi
Interfeys	Faqat sof virtual metodlardan iborat abstrakt sinf

Kod aniq sinflarga emas, interfeysga bog‘lansa — yangi turlarni qo‘shish oson bo‘ladi.

INTERFEYS

To‘lov tizimi

```
struct ITolov {
    virtual ~ITolov() = default;
    virtual bool tola(double s) = 0;
};
class Karta final : public ITolov {
    double limit;
public:
    explicit Karta(double l) : limit(l) {}
    bool tola(double s) override {
        if (s > limit) return false;
        limit -= s; return true;
    }
};
void xarid(ITolov &t, double s) { cout << (t.tola(s) ? "OK "
        : "RAD "); }
int main() {
    Karta k(100000);
    xarid(k, 50000); xarid(k, 60000); xarid(k, 45000);
}
```

▶ NATIJA

```
OK RAD OK
```

ITolov

interfeys — majburiyatlar ro‘yxati

`final`

Karta dan boshqa sinf meros ololmaydi

`xarid(ITolov &t, ...)`

funksiya interfeysga bog‘langan

TUSHUNISHNI TEKSHIRISH

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

6-bo‘lim yakuni: Loyihalash tamoyillari

1

LSP nimani talab qiladi?

Savol 1

2

override ning foydasi nima?

Savol 2

3

`class A final` nimani bildiradi?

Savol 3

XULOSA

Asosiy xulosalar

1 Invariant

Konstruktor o‘rnatadi, har bir public metod saqlaydi.

2 Interfeys

Ochiq “nima”, yashirin “qanday” — ichkarini o‘zgartirish xavfsiz.

3 friend / mutable

Do‘stlik meros qilinmaydi, o‘zaro emas; mutable — kesh va hisoblagich.

4 is-a / has-a

Meros — “bu …dir”, kompozitsiya — “…da bor”; shubhada kompozitsiya.

5 Slicing va olmos

Polimorf obyekt — havola/ko‘rsatkich orqali; olmos — virtual meros.

6 LSP, final, interfeys

Hosila asosiyning o‘rnini bosishi kerak; kod interfeysga bog‘lansin.

TAKRORLASH

Yakunlashdan oldin sakkizta savol

1. Sana sinfi uchun invariantga misol?
2. friend funksiya sinfning a’zosimi?
3. mutable maydonni const metodda o‘zgartirish mumkinmi?
4. Kompyuter va Protsessor munosabati?
5. `using Asosiy::f;` nima qiladi?
6. `Hayvon h = it;` nima yuz beradi?
7. Olmos muammosi yechimi?
8. Kvadrat : Tortburchak nima uchun LSP ni buzadi?

LABORATORIYA ISHI

11-laboratoriya ishi: 6 ta dasturlash topshirig‘i

5 **Hisob turlari**
O‘rta · 20 ball

6 **Olmos muammosi**
O‘rta · 20 ball

7 **mutable kesh**
O‘rta · 20 ball

8 **Kesilishsiz hayvonot bog‘i**
O‘rta · 20 ball

9 **Ombor tizimi**
Qiyin · 30 ball

10 **Ko‘p bosqichli meros**
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

Bank hisobi sinfi uchun bitta invariant yozing va uni qaysi metod himoya qilishini ayting.

2

“Talaba” va “Inson” — meros yoki kompozitsiya? “Talaba” va “Reyting daftari”-chi?

3

Bugungi mavzudan qaysi tamoyil sizga eng foydali tuyuldi?