21

YIGIRMA BIRINCHI MA’RUZA

Dasturlash 1 · C++ · 15-mavzu

Shablonlar bilan ishlash

Funksiya va sinf shablonlari · tursiz parametrlar · kompilyatsiya vaqtidagi hisoblash · ixtisoslashtirish · variadik shablonlar · C++20 konseptlari

template <T>

Stek<T>

template <>

concept

ISINISH

Qolip bilan pishiriq yopasiz: qolip bitta — xamir har xil. Dasturda-chi?

kattasi(int, int), kattasi(double, double), kattasi(string, string)... — har biri uchun alohida yozamizmi?

1

Uchala funksiyaning tanasi nimasi bilan farq qiladi?

2

vector<int> va vector<string> ni kim, qachon yozgan?

3

Qolipga “noto‘g‘ri xamir” solinsa (masalan, solishtirib bo‘lmaydigan tur), qachon bilamiz?

O‘QUV MAQSADLARI

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

1. **Tushuntirish** — shablon, instansiyalash va tur aniqlash (deduction) tushunchalarini
2. **Yozish** — funksiya shablonlarini va aniq argumentni ko‘rsatishni
3. **Yaratish** — sinf shablonlarini (Stek<T>, Juft<A, B>) va CTAD dan foydalanishni
4. **Qo‘llash** — tursiz parametrlar, standart argumentlar va static_assert ni
5. **Hisoblash** — qiymatlarni kompilyatsiya vaqtida (shablon rekursiyasi, constexpr)
6. **Ixtisoslashtirish** — shablonni to‘liq va qisman
7. **Yozish** — variadik shablonlar va fold ifodalarni
8. **Cheklash** — shablon parametrlarini C++20 konseptlari bilan

01

Funksiya shablonlari

template <typename T> · instansiyalash · tur aniqlash · aniq argument · auto

typename T | Deduction | <double> | auto

Shablon — funksiya emas, retsept

Tushuncha	Ma’nosi
Instansiyalash	Shablon aniq tur bilan ishlatilganda kompilyator haqiqiy funksiyani yaratadi
Tur aniqlash	T argumentlardan aniqlanadi; yashirin o‘zgartirishlar hisobga olinmaydi
Aniq argument	`kattasi<double>(3, 2.5)` — T qo‘lda beriladi
Instansiya	`kattasi<int>` va `kattasi<double>` — ikki xil funksiya
Talab	Tanadagi amallar (`>`) T uchun mavjud bo‘lishi shart

Umumlashgan dasturlash (generic programming): algoritm bir marta yoziladi — istalgan tur bilan ishlaydi.

AMALDA

kattasi, almashtir va massiv o‘lchami

```
template <typename T>
T kattasi(T a, T b) { return (a > b) ? a : b; }
template <typename T>
void almashtir(T &a, T &b) { T t = a; a = b; b = t; }
template <typename T, size_t N>
size_t olcham(const T (&m)[N]) { return N; }
int main() {
    cout << kattasi(3, 7) << " " << kattasi<double>(3, 2.5)
         << " ";
    string s = "olma", t = "nok";
    almashtir(s, t);
    int a[] = {4, 8, 15, 16, 23};
    cout << s << " " << olcham(a);
}
```

▶ NATIJA

```
7 3 nok 5
```

kattasi(3, 7)

T = int — argumentlardan

kattasi<double>(3, 2.5)

T aniq — 3 → 3.0

size_t N

massiv o‘lchami ham aniqlanadi

static va auto qaytarish turi

```
template <typename T>
void qayd(const T &x) {
    static int n = 0;
    cout << x << ":" << ++n << " ";
}
template <typename A, typename B>
auto qosh(A a, B b) { return a + b; }
int main() {
    qayd(1); qayd(2); qayd(1.5); qayd(3);
    cout << endl << qosh(2, 3.5) << " " << qosh(string("C"),
            "++");
}
```

▶ NATIJA

```
1:1 2:2 1.5:1 3:3
5.5 C++
```

`static int n`

har bir T uchun o‘z static o‘zgaruvchisi

`qayd(1.5)`

qayd<double> — yangi instansiya, hisob 1 dan

`auto`

natija turi `a + b` ifodasidan: double, string

TUSHUNISHNI TEKSHIRISH

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

1-bo‘lim yakuni: Funksiya shablonlari

1

`kattasi(3, 2.5)` chaqiruvida nima bo‘ladi?

Savol 1

2

Instansiyalash nima?

Savol 2

3

`kattasi<int>` va `kattasi<double>` bitta funksiyami?

Savol 3

AMALIY TOPSHIRIQ

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

1-topshiriq: Funksiya shablonlari

Ikkita funksiya shablonini yozing: `template <typename T> void almashtir(T& a, T& b)` va `template <typename T> T kattasi(T a, T b)`. `main()` ularni `int`, `double` va `string` turlari uchun chaqiradi.

Kirish: Birinchi qatorda `q` ($1 \le q \le 100$). Keyin `q` qatorda: tur (`int`, `double`, `string`) va ikki qiymat.

Chiqish: Har bir so‘rov uchun bitta qatorda: almashtirilgan ikki qiymat va kattasi (double — 2 xona).

MISOL · KIRISH

```
3
int 3 9
double 2.5 -1
string olma anor
```

MISOL · CHIQISH

```
9 3 9
-1.00 2.50 2.50
anor olma olma
```

Telefon kamerasi bilan skanerlang

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

02

Sinf shablonlari

Stek<T> · metodni tashqarida aniqlash · Juft<A, B> · CTAD

Stek<T>

Stek<T>::push

Juft<A,B>

CTAD

Bitta Stek<T> — ko‘p stek

1 E’lon

```
template <typename T> class
Stek { vector<T> d; ... };
```

2 Tashqarida aniqlash

```
template <typename T> void
Stek<T>::push(const T &x) {
... }
```

3 Ishlatish

`Stek<int> s;` — sinf shablonida T odatda aniq yoziladi.

4 Mustaqil turlar

`Stek<int>` va `Stek<string>` — bir-biriga bog‘liq bo‘lmagan ikki sinf.

5 Sarlavha fayli

Shablon kodi to‘liq .h faylda — kompilyator uni instansiyalashda ko‘rishi kerak.

6 CTAD (C++17)

`pair p(1, 2.5);` → `pair<int, double>` — tur konstruktordan.

Qavslar muvozanatini tekshirish

```
template <typename T>
class Stek {
    vector<T> d;
public:
    void push(const T &x) { d.push_back(x); }
    T pop() { T x = d.back(); d.pop_back(); return x; }
    bool bosh() const { return d.empty(); }
};
bool muvozanat(const string &s) {
    Stek<char> st;
    for (char c : s) {
        if (c == '(' || c == '[' || c == '{') st.push(c);
        else if (c == ')' || c == ']' || c == '}') {
            if (st.bosh()) return false;
            char o = st.pop();
            if ((o == '(') != (c == ')')
                    || (o == '[') != (c == ']')) return false;
        }
    }
    return st.bosh();
}
int main() {
    for (string s : {"(a+b)*[c-{d/e}]", "(()]", "{[()()]}",
            "((("})
        cout << muvozanat(s);
}
```

▶ NATIJA

```
1010
```

Stek<char>

ochuvchi qavslar stekka

pop()

yopuvchi qavs oxirgi ochilgani bilan mos bo‘lishi kerak

st.bosh()

oxirida ochiq qavs qolmasligi shart

IKKI PARAMETR

Juft<A, B> va CTAD

```
template <typename A, typename B>
struct Juft {
    A birinchi;
    B ikkinchi;
    Juft(A a, B b) : birinchi(a), ikkinchi(b) {}
};
template <typename A, typename B>
ostream& operator<<(ostream &o, const Juft<A, B> &j) {
    return o << "(" << j.birinchi << ", " << j.ikkinchi << ")";
}
int main() {
    Juft<string, int> a("Aziza", 91);
    Juft b(3.5, 'x');
    cout << a << " " << b;
}
```

▶ NATIJA

```
(Aziza, 91) (3.5, x)
```

`Juft<string, int>`

turlar aniq ko‘rsatilgan

`Juft b(3.5, 'x')`

CTAD: Juft<double, char>

`operator<<`

shablon funksiya — barcha Juft lar uchun

TUSHUNISHNI TEKSHIRISH

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

2-bo‘lim yakuni: Sinf shablonlari

1

Stek<int> va Stek<string> munosabati?

Savol 1

2

Sinf shablonining metodi tashqarida qanday yoziladi?

Savol 2

3

CTAD nima?

Savol 3

AMALIY TOPSHIRIQ

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

2-topshiriq: Stek<T> shablon sinfi

`template <typename T> class Stek` ni `vector<T>` asosida yozing: `push`, `pop` (qiymatni qaytaradi), `top`, `size`, `empty`. `main()` bir vaqtda `Stek<int>` va `Stek<string>` bilan ishlaydi.

Kirish: Birinchi qatorda `q` ($1 \le q \le 1000$). Buyruqlar: `i push x`, `s push so‘z`, `i pop`, `s pop`, `i top`, `s top`, `i size`, `s size`.

Chiqish: `pop` va `top` — qiymat (bo‘sh stekda `BOSH`), `size` — elementlar soni.

MISOL · KIRISH

```
6
i push 5
s push olma
i push 7
i pop
s top
i size
```

MISOL · CHIQISH

```
7
olma
1
```

Telefon kamerasi bilan skanerlang

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

03

Tursiz parametrlar va metadasturlash

template <typename T, size_t N> · standart argumentlar · static_assert · Faktorial<N>

size_t N

Massiv<>

static_assert

constexpr

TURSIZ PARAMETR

O‘lcham turga “yozilgan” massiv

```
template <typename T = int, size_t N = 5>
class Massiv {
    static_assert(N > 0, "o'lcham musbat bo'lsin");
    T d[N] = {};
public:
    T& operator[](size_t i) { return d[i]; }
    constexpr size_t size() const { return N; }
    T yig() const { T s{}; for (auto x : d) s += x; return s; }
};
int main() {
    Massiv<double, 3> a;
    a[0] = 1.5; a[2] = 2;
    Massiv<> b;
    b[4] = 7;
    cout << a.size() << " " << a.yig() << " " << b.size() << " "
            << b.yig();
}
```

▶ NATIJA

```
3 3.5 5 7
```

size_t N

qiymat parametri — kompilyatsiya vaqtida ma’lum bo‘lishi shart

Massiv<>

standart argumentlar: int va 5

static_assert

`Massiv<int, 0>` — kompilyatsiya xatosi

Diqqat!

Massiv<int, 3> va Massiv<int, 4> — har xil turlar

Kompilyatsiya vaqtida hisoblash

```
template <unsigned N>
struct Fakt {
    static const unsigned long long q = N * Fakt<N - 1>::q;
};
template <>
struct Fakt<0> { static const unsigned long long q = 1; };
template <unsigned N>
struct Fib {
        static const unsigned long long q = Fib<N - 1>::q + Fib<N - 2>::q; };
template <> struct Fib<0> {
        static const unsigned long long q = 0; };
template <> struct Fib<1> {
        static const unsigned long long q = 1; };
constexpr long long kvadrat(long long x) { return x * x; }
int main() {
    cout << Fakt<10>::q << " " << Fib<80>::q << " "
            << kvadrat(12);
}
```

▶ NATIJA

```
3628800 23416728348467685 144
```

Fakt<10>

rekursiv instansiyalash — Fakt<0> da to‘xtaydi

Fib<80>

bir zumda — har Fib<k> faqat bir marta yaratiladi

constexpr

zamonaviy usul: oddiy funksiya kompilyatsiyada hisoblanadi

TUSHUNISHNI TEKSHIRISH

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

3-bo‘lim yakuni: Tursiz parametrlar va metadasturlash

1

Tursiz parametr N ga qanday qiymat berish mumkin?

Savol 1

2

Massiv<int, 3> va Massiv<int, 4> bir turmi?

Savol 2

3

static_assert qachon ishlaydi?

Savol 3

AMALIY TOPSHIRIQ

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

3-topshiriq: Massiv<T, N>: tursiz parametr

`template <typename T, int N> class Massiv` — o‘lchami kompilyatsiya vaqtida ma’lum massiv. Metodlar: `size()`, `ol(i, natija)` (indeks to‘g‘ri bo‘lsa `true`), `yigindi()`. `main()` tayyor: `Massiv<int, 5>` va `Massiv<double, 3>`.

Kirish: 5 ta butun son, 3 ta haqiqiy son, so‘ng `q` va `q` ta so‘rov: `i k` yoki `d k` (k — indeks).

Chiqish: Ikki yig‘indi (haqiqiysi 2 xona), so‘ng har bir so‘rovga qiymat yoki `CHEGARADAN TASHQARI`.

MISOL · KIRISH

```
1 2 3 4 5
0.5 1.5 2.25
3
i 4
d 0
i 5
```

MISOL · CHIQISH

```
15
4.25
5
0.50
CHEGARADAN TASHQARI
```

Telefon kamerasi bilan skanerlang

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

04

Ixtisoslashtirish

Asosiy shablon · to‘liq template <> · qisman T* · const char* muammosi

template <>

T*

const char*

Tanlov

Ba’zi turlar uchun alohida amalga oshirish

Turi	Ko‘rinishi	Qachon tanlanadi
Asosiy shablon	`template <typename T> struct Chop { ... };`	Boshqa hech qaysi mos kelmasa
To‘liq	`template <> struct Chop<bool> { ... };`	Aynan shu tur uchun
Qisman (faqat sinf)	`template <typename T> struct Chop<T*> { ... };`	Turlar guruhi: istalgan ko‘rsatkich, vector

Bir nechta variant mos kelsa — eng ixtisoslashgani tanlanadi: to‘liq > qisman > asosiy.

To‘liq va qisman ixtisoslashtirish

```
template <typename T>
struct TurNomi { static string nom() { return "?"; } };
template <> struct TurNomi<int> { static string nom() {
        return "int"; } };
template <> struct TurNomi<char> { static string nom() {
        return "char"; } };
template <typename T>
struct TurNomi<T*> { static string nom() {
        return TurNomi<T>::nom() + "*"; } };
template <typename T>
struct TurNomi<vector<T>> { static string nom() {
        return "vector<" + TurNomi<T>::nom() + ">"; } };
int main() {
    cout << TurNomi<int>::nom() << " " << TurNomi<char**>::nom()
            << " "
         << TurNomi<vector<int*>>::nom() << " "
                << TurNomi<double>::nom();
}
```

▶ NATIJA

```
int char** vector<int*> ?
```

`TurNomi<char**>`

qisman T* ikki marta: char* → char

`vector<int*>`

qisman vector<T>, ichida T* va int

`double`

ixtisoslashtirish yo‘q — asosiy shablon

Manzil emas — matnni solishtirish

```
template <typename T>
T kattasi(T a, T b) { return (a > b) ? a : b; }
template <>
const char* kattasi<const char*>(const char *a, const char *b) {
    return strcmp(a, b) > 0 ? a : b;
}
template <typename T>
void chop(const T &x) { cout << x << " "; }
template <>
void chop<bool>(const bool &x) { cout << (x ? "ha" : "yo'q")
        << " "; }
int main() {
    cout << kattasi<const char*>("olma", "nok") << " ";
    chop(5); chop(true); chop(false);
}
```

▶ NATIJA

```
olma 5 ha yo'q
```

umumiy kattasi

const char* uchun manzillarni solishtirardi — tasodifiy natija

strcmp

ixtisoslashtirish matnni solishtiradi: "olma" > "nok"

chop<bool>

true/false o‘rniga ha/yo‘q

TUSHUNISHNI TEKSHIRISH

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

4-bo‘lim yakuni: Ixtisoslashtirish

1

Ixtisoslashtirish nima uchun kerak?

Savol 1

2

Qisman ixtisoslashtirish qaysi shablonlarda mavjud?

Savol 2

3

`TurNomi<int*>` uchun qaysi versiya tanlanadi?

Savol 3

05

Variadik shablonlar

typename... Args · sizeof... · rekursiya · C++17 fold ifodalar · forward

Args...

sizeof...

(x + ...)

forward

Paketni operator bilan “yig‘ish”

Shakl	Ochilishi
unar o‘ng: `(x op ...)`	x1 op (x2 op (x3 op x4))
unar chap: `(... op x)`	((x1 op x2) op x3) op x4
binar chap: `(init op ... op x)`	(((init op x1) op x2) op x3) — bo‘sh paketda ham ishlaydi

`typename... Args` — turlar paketi, `Args... args` — argumentlar paketi, `args...` — ochish, `sizeof...(args)` — soni.

Istalgan sondagi argument

```cpp
void chop() { cout << endl; }
template <typename B, typename... Q>
void chop(const B &b, const Q&... q) {
    cout << b;
    if (sizeof...(q) > 0) cout << ", ";
    chop(q...);
}
template <typename... T>
auto yigindi(T... x) { return (x + ...); }
template <typename... T>
auto kopaytma(T... x) { return (1 * ... * x); }
int main() {
    chop(1, "ikki", 3.5, '4');
    cout << yigindi(1, 2, 3.5) << " " << kopaytma() << " "
            << kopaytma(2, 3, 4);
}
```

▶ NATIJA

```
1, ikki, 3.5, 4
6.5 1 24
```

`chop(q...)`

birinchisini chiqarib, qolganlari bilan rekursiya

`chop()`

bazaviy holat — paket bo‘sh

`(1 * ... * x)`

binar fold — bo‘sh paketda 1

Argumentlarni o‘zgarishsiz uzatish

```
template <typename T, typename... Args>
unique_ptr<T> yarat(Args&&... args) {
    cout << sizeof...(args) << " arg; ";
    return unique_ptr<T>(new T(std::forward<Args>(args)...));
}
struct Talaba {
    string ism;
    int kurs;
    Talaba(string i, int k) : ism(i), kurs(k) {}
};
int main() {
    auto t = yarat<Talaba>("Aziza", 2);
    auto s = yarat<string>(3, '*');
    cout << t->ism << " " << *s;
}
```

▶ NATIJA

```
2 arg; 2 arg; Aziza ***
```

Args&&... args

“universal havola” — istalgan argument

std::forward

lvalue/rvalue holatini saqlab uzatadi

make_unique

standart kutubxona aynan shunday qurilgan

TUSHUNISHNI TEKSHIRISH

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

5-bo‘lim yakuni: Variadik shablonlar

1

`sizeof...(args)` nimani qaytaradi?

Savol 1

2

`(x + ...)` qanday ochiladi (3 ta argument)?

Savol 2

3

Variadik rekursiyada bazaviy holat nima?

Savol 3

AMALIY TOPSHIRIQ

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

4-topshiriq: Variadik shablonlar

Ikki variadik shablon yozing: `chop(args...)` — barcha argumentlarni `,` bilan ajratib bitta qatorda chiqaradi; `yigindi(args...)` — sonli argumentlar yig‘indisini qaytaradi (fold ifoda). `main()` ularni turli sondagi argumentlar bilan chaqiradi.

Kirish: Bitta qatorda: butun son `a`, haqiqiy son `b` va so‘z `c`.

Chiqish: `main()` chiqaradigan to‘rt qator.

MISOL · KIRISH

```
7 2.5 olma
```

MISOL · CHIQISH

```
7
7, 2.5, olma
olma, 7, olma, 2.5
16.5
```

Telefon kamerasi bilan skanerlang

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

06

C++20 konseptlari va STL

concept · requires · if constexpr · STL — shablonlar to‘plami

- concept
- requires
- if constexpr
- STL

Talablar — imzoda, nom bilan

Konseptsiz shablon

- Talablar tanada “yashiringan”
- Noto‘g‘ri tur — xato shablon ichida, siz yozmagan qatorda
- Xabar o‘nlab qatorga cho‘ziladi

Konsept bilan (C++20)

- `template <typename T> concept Sonli = integral<T> || floating_point<T>;`
- `template <Sonli T> double ortacha(const vector<T> &v);`
- Xato chaqiruv joyida: “constraints not satisfied: Sonli<T>”

`requires` ifodasi: `concept Chiqariladigan = requires(ostream &o, const T &x) { o << x; };`

AMALDA

concept, requires va if constexpr

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

template <typename T>
concept Sonli = integral<T> || floating_point<T>;
template <Sonli T>
double ortacha(const vector<T> &v) {
    double s = 0;
    for (T x : v) s += x;
    return s / v.size();
}
template <typename T>
string turi(const T &) {
    if constexpr (integral<T>) return "butun";
    else if constexpr (floating_point<T>) return "haqiqiy";
    else return "boshqa";
}
int main() {
    cout << ortacha(vector<int>{1, 2, 4}) << " " << turi(5)
         << " " << turi(2.5) << " " << turi("a");
    // ortacha(vector<string>{"a"});  // XATO: Sonli<string> bajarilmadi
    return 0;
}
```

▶ NATIJA

```
2.33333 butun haqiqiy boshqa
```

`template <Sonli T>`

T faqat son bo‘lishi mumkin

`if constexpr`

kompilyatsiya vaqtidagi tarmoqlanish — keraksiz shox umuman kompilyatsiya qilinmaydi

izohdagi qator

ochsangiz — aniq, qisqa xato xabari

Shablon sinfimiz STL ustida: so‘zlar chastotasi

```
template <typename K, typename V = int>
class Chastota {
    map<K, V> m;
public:
    void qosh(const K &k, V n = 1) { m[k] += n; }
    vector<pair<K, V>> engKop(size_t k) const {
        vector<pair<K, V>> v(m.begin(), m.end());
        sort(v.begin(), v.end(), [](auto &a, auto &b) {
            return a.second != b.second ? a.second > b.second
                    : a.first < b.first;
        });
        if (v.size() > k) v.resize(k);
        return v;
    }
};
int main() {
    Chastota<string> c;
    stringstream ss("kod tez kod qisqa shablon kod tez");
    string s;
    while (ss >> s) c.qosh(s);
    for (auto &[soz, n] : c.engKop(2)) cout << soz << "=" << n
            << " ";
}
```

▶ NATIJA

```
kod=3 tez=2
```

`map<K, V>`

STL konteyner — o‘zi ham shablon

`V = int`

standart shablon argumenti

`sort + lambda`

algoritm konteyner turini bilmaydi — iteratorlar bilan ishlaydi

TUSHUNISHNI TEKSHIRISH

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

6-bo‘lim yakuni: C++20 konseptlari va STL

1

C++20 konseptlarining asosiy foydasi?

Savol 1

2

`if constexpr` oddiy if dan nimasi bilan farq qiladi?

Savol 2

3

STL qanday uch qismdan iborat?

Savol 3

XULOSA

Asosiy xulosalar

1 Shablon

Retsept: instansiyalashda T o‘rniga aniq tur qo‘yiladi; har tur — alohida funksiya.

2 Sinf shablonlari

`Stek<T>`; metodlar `Stek<T>::` bilan; CTAD — `pair p(1, 2.5)`.

3 Tursiz parametrlar

`size_t N` — kompilyatsiyada ma’lum; `static_assert` bilan tekshiruv.

4 Ixtisoslashtirish

To‘liq `template <>` va qisman `T*`; eng ixtisoslashgani tanlanadi.

5 Variadik

`Args...`, `sizeof...`, fold `(x + ...)`, `std::forward`.

6 Konseptlar

`concept`, `requires`, `if constexpr` — talablar imzoda, xatolar tushunarli.

TAKRORLASH

Yakunlashdan oldin sakkizta savol

1. Instansiyalash nima?
2. `kattasi<double>(3, 4.5)` da T qanday aniqlanadi?
3. Sinf shabloni metodini tashqarida yozishda nima takrorlanadi?
4. `std::array<int, 5>` dagi 5 — qanday parametr?
5. `template <>` nima?
6. `(... + x)` qanday fold?
7. `sizeof...(args)`?
8. Konsept talabini buzsa xato qayerda chiqadi?

LABORATORIYA ISHI

15-laboratoriya ishi: 6 ta dasturlash topshirig‘i

5 **Ixtisoslashtirish: tavsif<T>**
O‘rta · 20 ball

6 **Umumiy saralash shabloni**
O‘rta · 20 ball

7 **Konsept: Raqamli**
O‘rta · 20 ball

8 **STL: so‘zlar chastotasi**
O‘rta · 20 ball

9 **Navbat<T, N>: halqa bufer**
O‘rta · 20 ball

10 **LRU kesh<K, V>**
Qiyin · 30 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

Ikkita qiymatdan kichigini qaytaruvchi `kichigi` funksiya shablonini yozing.

2

Qachon shablonni ixtisoslashtirish kerak bo‘ladi? Misol keltiring.

3

Kurs davomida o‘rgangan qaysi mavzu sizga eng foydali bo‘ldi?