

OBJEKTINIS PROGRAMAVIMAS C++, 2026

Kurso struktūra: Teminiai skyriai -> Paskaitos

TEMINIAI SKYRIAI (TEMINĖ STRUKTŪRA)

1 skyrius. EVOLIUCIJA: Nuo C iki Modern C++

1.1. Iš Monolito į Modulį

- Programos dekompozicija
- Kompiliavimo vienetai (*compilation units*)
- Antraštės/implementacijos atskyrimas (*header/implementation*)

1.2. Informacijos slėpimas. Pradžia

- Statiniai kintamieji (`static`)

1.3. Inkapsuliacija. Pradžia

- Vartotojo apibrėžti tipai (*User-Defined Types*)
- Inkapsuliacija (*Encapsulation*) C būdu

1.4. Gyvavimo ciklas

- Nepermatomos rodyklės (*opaque pointers*)
- Rankinis resursų valdymas
- RAII. Pradžia

1.5. Pažangios C technikos (Priedas 10A-D) - perdaryti

- Tipų alijavimas (`typedef aliasing`)
- Konstantiškumo taisyklės (`const correctness`)
- Funkcijų rodykliai (`function pointers`) polimorfizmui
- Bendrinio rodyklio technika (`void pointer generics`)

1.6. Šuolis į C++ - perdaryti

- `class` kaip sprendimas
- `Constructor/Destructor` automatizacija
- `private/public` vykdymas kompiiatoriaus lygmenje

2 skyrius. KLASĖS ANATOMIJA

2.1. KLASĖS pagrindai

- `struct` vs `class`
- Funkcijos-nariai (*member functions*)
- `this` rodyklė
- Konstruktoriai: numatytais (`default`), parametrizuotas
- Destruktorius
- Inicializacija vs Priskyrimas

2.2. Prieigos kontrolė

- `public/private/protected`
- Draugiškos funkcijos/klasės (`friend`)
- Getterių/Setterių šablonas (*getter/setter*)

2.3. RAII principas

- Resursų gavimas yra inicializacija (*Resource Acquisition Is Initialization*)
 - Destruktoriaus atsakomybė
 - Automatinis valymas (*automatic cleanup*)
-

3 skyrius. GERESNIS C

3.1. Nuorodos (*References*)

- *lvalue* nuorodos
- Nuoroda vs rodyklė
- Nuorodos kaip funkcijų parametrai
- Gražinimas per nuorodą (*return by reference*)

3.2. Auto ir tipo išvedimas

- `auto` raktažodis; kada naudoti/nenaudoti
- `decltype`

3.3. Ciklas su diapazonu (*Range-based loops*)

- `for (auto& elem : container)`; kada efektyvu

3.4. Nullptr

- vs `NULL` vs `0`

3.5. Vieninga inicializacija (*Uniform initialization*)

- Riestinių skliaustų inicializacija `{}`
 - Tiesioginė inicializacija `()`
-

4 skyrius. KOPIJAVIMO SEMANTIKA

4.1. Numatytais kopijavimo elgesys

- Pavaršinio kopijavimo (*shallow copy*) problema
- Bitinis kopijavimas (*bitwise copy*)

4.2. Kopijavimo konstruktorius (*Copy Constructor*)

- Kada kviečiamas
- Giliojo kopijavimo (*deep copy*) implementacija

4.3. Kopijavimo priskyrimo operatorius (*Copy Assignment Operator*)

- `operator=` perkrovimas (*overloading*)
- Savęs priskyrimo patikra (*self-assignment check*)
- "Kopijuok-ir-keisk" idioma (*copy-and-swap idiom*)

4.4. Trijų taisyklė (Rule of 3)

- Destruktorius + Kopijos konstruktorius + Kopijos priskyrimas
 - Kada reikalinga
-

5 skyrius. OOP PRINCIPAI (4 kolonos)

5.1. Inkapsuliacija (*Encapsulation*)

- Apibendrinimas iš Klasės anatomijos

5.2. Paveldėjimas (*Inheritance*)

- `public/protected/private` paveldėjimas
- Konstruktorių/destruktorių grandinė
- `is-a` ryšys
- Bazinės klasės inicializacija (*base class initialization*)
- Apsaugotų narių (*protected members*) prieiga

5.3. Polimorfizmas (*Polymorphism*)

- Funkcijų perkrovimas (*function overloading*) - kompiliavimo metu
- Virtualios funkcijos (*virtual functions*) - vykdymo metu (*runtime*)
- Virtualių funkcijų lentelė (*vtable*)
- `override` raktažodis
- Grynai virtuali (*pure virtual*) = 0
- Abstrakčios klasės (*abstract classes*)
- Sąsajos šablonas (*interface pattern*)

5.4. Abstrakcija (*Abstraction*)

- Sąsaja (*interface*) vs Implementacija

- Priklausomybių inversija (*dependency inversion*)
-

6 skyrius. MODERN C++ (C++11/14/17/20/23/26)

6.1. rvalue nuorodos

- *lvalue vs rvalue*
- `&&` sintakė
- `std::move`

6.2. Perkėlimo konstruktorius (*Move Constructor*)

- Kada kviečiamas
- Resursų "vagystė" (*resource stealing*)
- `noexcept`

6.3. Perkėlimo priskyrimo operatorius (*Move Assignment*)

- "Perkelk-ir-sukeisk" šablonas (*move-and-swap pattern*)

6.4. Penkių taisyklė (Rule of 5)

- Trijų taisyklė + Perkėlimo operacijos
- Nulio taisyklė (*Rule of 0*) - kada nereikia nieko

6.5. Išmaniosios rodyklės (Smart Pointers)

- `unique_ptr` (nuosavybė)
 - `shared_ptr` (nuorodų skaičiavimas, *reference counting*)
 - `weak_ptr`
 - `make_unique/make_shared`
 - **RAII** "tobulumas"
-

7 skyrius. OPERATORIŲ PERKROVIMAS

7.1. Aritmetiniai operatoriai

- `+, -, *, /` kaip narys/draugas (`member/friend`)
- Sudėtinis priskyrimas `+=, -=`

7.2. Palyginimo operatoriai

- `==, !=, <, >, <=, >=`
- Erdvėlaivio operatorius (`spaceship operator`) `<=>` (C++20)

7.3. Srauto operatoriai

- `operator<< / operator>>`

- `friend` reikalavimas

7.4. Indeksavimo operatorius []

- `const/ne-const` versijos

7.5. Funkcijos iškvietimo operatorius ()

- Funktoriai (`functors`)
-

8 skyrius. ŠABLONŲ PAGRINDAI

8.1. Funkcijų šablonai (*Function templates*)

- Šablono sintaksė (`template` syntax)
- Tipo išvedimas (*type deduction*)
- Šablono specializacija (*template specialization*)

8.2. Klasių šablonai (*Class templates*)

- Šablono parametrai (*template parameters*)
 - Šablono instanciacija (*template instantiation*)
 - Atskiro kompiliavimo problema
-

9 skyrius. STL PAGRINDAI

9.1. Konteinerių pagrindai

- `std::array` (fiksuoto dydžio)
- `std::vector` (dinaminis)
- Bendra konteinerių sąsaja

9.2. Iteratoriai

- Iteratoriaus koncepcija
- `begin()/end()`
- Iteratorių kategorijos

9.3. Algoritmai

- `std::sort, std::find`
- *Lambda* išraiškos (pagrindinės)
- Algoritmas + Iteratorius šablonas

9.4. Kiti konteineriai (apžvalga)

- `std::string`
- `std::list`

- `std::map`
 - `std::set`
 - Kada ką naudoti
-

10 skyrius. IŠIMČIŲ APDOROJIMAS

10.1. `try/catch/throw`

- Išimčių sklidimas (*exception propagation*)
- Steko atsukimas (*stack unwinding*)

10.2. Standartinės išimtys

- `std::exception` hierarchija

10.3. RAII ir išimtys

- Kodėl RAII kritiškas
- Išimčių saugumo lygiai (*exception safety levels*)

10.4. Noexcept

- Kada ir kodėl
-

PASKAITŲ PROJEKCIJA Į TEMINES DALIS (siekiamybė

#	Paskaitos pavadinimas	Objektas	Teminės dalys	Fokusas
1	Ivadas + C evoliucija I dalis	Stack (C, stages 1-5)	Intro + I.1, I.2	Modularumas, kompliacija, info hiding pradžia. Isibėgėjimas su C
2	C evoliucija II + šuolis į C++	Stack (C, 6-9) → Stack-C++ "v0"	I.3, I.4, I.5	Factory, lifecycle problema → CLASS sprendimas. Wow! Fixed array dar.
3	Klasės anatomija: paprasti objektai	Student, Point	II.1, II.2, II.3 (basic)	class sintaksė, konstruktoriai, getters/setters , this , access control
4	Geresnis C	Student su refs	III (visa)	References, auto, range-loops, nullptr, uniform init
5	STL ivadas: konteineriai	Student + vector	IX.1, IX.2 (dalis)	std::array, std::vector , iteratoriai, begin/end, range-based praktikoje
6	RAII ir destruktorius	String	II.3 (full) + IV.1	char* dinamika, destruktorius, shallow copy problema, "žemiškas" pavyzdys
7	Copy Constructor	String	IV.2	Deep copy , kada kvičiamas, String(const String&)

#	Paskaitos pavadinimas	Objektas	Teminės dalys	Fokusas
8	Copy Assignment + Rule of Three	String	IV.3, IV.4	<code>operator=, self-assignment, copy-and-swap, Rule of Three</code>
9	Kompozicija "ant pirštų"	Course (has Students)	V.1 (extend)	Shallow/deep lifecycle, <code>has-a</code> , agregacija vs kompozicija
10	Paveldėjimas	Shape → Circle/Rect	V.2	<code>Is-a</code> , konstruktorių chain, protected, base <code>init</code>
11	Polimorfizmas: virtualios funkcijos	Shape su <code>draw()</code>	V.3 (dalis)	<code>Virtual, vtable, override</code> , runtime binding
12	Abstrakčios klasės ir sasajos	IDrawable, Animal	V.3 (finish), V.4	Pure <code>virtual (=0)</code> , interface pattern, dependency inversion
13	Move semantika	String su <code>move</code>	VI.1, VI.2, VI.3, VI.4	Rvalue refs, <code>std::move</code> , Move constructor/assign, Rule of Five
14	Protingi rodykliai + Rule of Zero	Stack-C++ su <code>unique_ptr</code>	VI.5 + Rule of Zero	Smart pointers, RAI ^I tobulumas, compiler-generated methods
15	Šablonai + Kulminacija	Stack<T> → <code>std::stack</code>	VIII.1, VIII.2 + IX.3	Templates, <code>Stack<T></code> , connection to STL, exceptions, best practices

OBJEKTŲ HIERARCHIJA PAGAL SUDĖTINGUMĄ

1. PAPRASTAS objektas (tik duomenys + prieiga)

- **Student** arba **Point** arba **Date**
- Nėra dinamininių resursų

Fokusas: `class` sintaksė, constructors, getters/setters, access control

2. KOMPOZICINIS objektas (turi kitų objektų)

- **Course** (turi `Student`'ų array/vector)
- **Rectangle** (turi 2 `Point`'us)

Fokusas: shallow vs deep, objektų santykiai, kompozicija

3. RESURSUS VALDANTIS objektas

- **String** (dinaminis `char` masyvas)
- **Stack** (dinaminis masyvas)

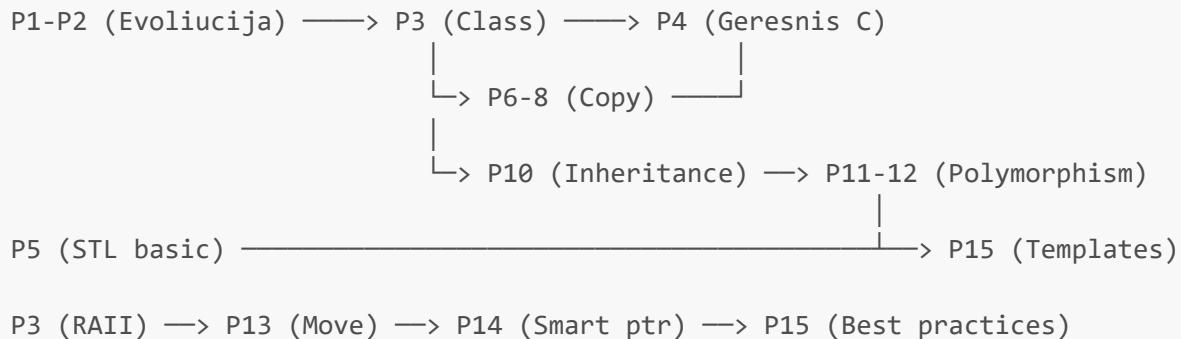
Fokusas: RAI_II, destrukturis, copy/move semantika

4. POLIMORFINIS objektas

- **Shape** → Circle, Rectangle ([inheritance hierarchy](#))
- **Animal** → Dog, Cat

Fokusas: virtual functions, abstract classes

KRITINIAI RYŠIAI TARP PASKAITŲ



PAPILDOMI OBJEKTAI SEMINARAMS/UŽDUOTIMS

Objektas	Kam	Sudėtingumas
Date	Operatorių perkrovimas (+, -, ==, <, <<)	Paprastas
Fraction	Aritmetiniai operatoriai, normalizacija	Vidutinis
Matrix	<code>operator[]</code> , 2D dinaminis masyvas, copy/move	Sudėtingas
BankAccount	Kompozicija (turi <code>Transaction</code> 'ų vector)	Vidutinis
Employee → Manager	Inheritance, virtual salary calculation	Vidutinis

VERSIJA

Data: 2026-02-01

Autorius: Viktoras Golubevas

Statusas: Preliminari struktūra: 13-15 paskaitų iš 10 teminių skyrių