

# Užduočių Atlikimo ir Pateikimo Gidas

**Kursas:** Objektinis programavimas C++

**Semestras:** 2026 pavasaris

**Versija:** 1.0

## Turinys

1. [GitLab projekto sukūrimas](#)
2. [Projekto struktūra](#)
3. [Užduoties atlikimo workflow](#)
4. [Git commit'ų gairės](#)
5. [README.md reikalavimai](#)
6. [Pateikimas Moodle](#)
7. [Vertinimas](#)
8. [DUK](#)

## GitLab projekto sukūrimas

### 1 žingsnis: Sukurti repo GitLab'e

1. Eikite į fakulteto GitLab: <https://gitlab.mif.vu.lt>
2. Prisijunkite su VU kredencialais
3. Sukurkite **naują projektą**:
  - **Project name:** [cpp-2026](#)
  - **Visibility:** [Private](#) (svarbu!)
  - **Initialize with README:**  (pažymėti)

### 2 žingsnis: Suteikti prieigą dėstytojui

1. **Settings → Members**
2. **Add member:** [\[dėstytojo username\]](#)
3. **Role:** [Reporter](#) (skaitymo teisės)

### 3 žingsnis: Clone repo į savo kompiuterį

```
git clone https://gitlab.mif.vu.lt/[jūsų-username]/cpp-2026.git  
cd cpp-2026
```

## Projekto struktūra

### Pilna struktūra:

```

cpp-2026/           ← GitLab repo

    ├── README.md      ← Pagrindinis projekto README (PRIVALOMA)
    └── .gitignore      ← Git ignoruojami failai (PRIVALOMA)

    └── U1/             ← Užduotis 1
        ├── README.md    ← Užduoties santrauka (PRIVALOMA)
        ├── 01/
        |   └── main.cpp
        ├── 02/
        |   └── main.cpp
        ├── 03/
        |   ├── main.cpp
        |   ├── rusiavimas.h
        |   ├── rusiavimas.cpp
        |   └── Makefile
        ├── 04/             ← 4 žingsnis
        └── 05/             ← 5 žingsnis (FINAL)

    └── U2/             ← Užduotis 2
        ├── README.md
        ├── 01/
        ├── 02/
        └── 03/

    └── U3/             ← Užduotis 3
        └── ...

    ... (U4-U9)

```

## README.md hierarchija:

| Lygis     | Failas           | Turinys                                     | Privaloma?                               |
|-----------|------------------|---|--|
| Projektas | /README.md       | Bendras projekto aprašymas, užduočių būsena | <input checked="" type="checkbox"/> TAIP |
| Užduotis  | /U1/README.md    | Užduoties santrauka, testavimas, pastabos   | <input checked="" type="checkbox"/> TAIP |
| Žingsnis  | /U1/01/README.md | Nebūtina (tik jei norite)                   | <input checked="" type="checkbox"/> NE   |

## Užduoties atlikimo workflow

### Bendra schema:

1. Perskaityti užduotį (pvz., U1.md)  
↓
2. Sukurti direktorijas žingsniams (U1/01/, U1/02/, ...)  
↓
3. Atlikti žingsnį → compile → test  
↓

4. Commit (po kiekvieno žingsnio!)  
↓
5. Push į GitLab (backup!)  
↓
6. Kartoti 3-5 kiekvienam žingsniui  
↓
7. Užpildyti U1/README.md  
↓
8. Final commit + push  
↓
9. Sukurti archyvą  
↓
10. Pateikti Moodle

## Detalus pavyzdys (U1):

### Žingsnis 1: Sukurti direktorijas

```
cd cpp-2026
mkdir -p U1/01 U1/02 U1/03 U1/04 U1/05
```

### Žingsnis 2: Atlikti 1 žingsnį

```
cd U1/01
# Rašyti kodą (main.cpp)
g++ main.cpp -o programa
./programa
# Testuoti
```

### Žingsnis 3: Commit

```
git add U1/01/
git commit -m "U1: 1 žingsnis - Hello World ir masivų ivestis"
git push
```

### Žingsnis 4: Tęsti su 2 žingsniu

```
cd ../02
# Kopijuoti iš 01/ (jei reikia)
cp ../01/main.cpp .
# Modifikuoti kodą
# ...
git add U1/02/
```

```
git commit -m "U1: 2 žingsnis - Bubble sort funkcija"  
git push
```

## Žingsnis 5: Po visų žingsnių - README

```
cd U1  
# Sukurti README.md (žr. šabloną žemiau)  
git add README.md  
git commit -m "U1: Užduoties README"  
git push
```

## Git commit'ų gairės

### Geri commit'ai:

#### Po kiekvieno žingsnio

```
git commit -m "U1: 1 žingsnis - Hello World ir masyvų ivestis"  
git commit -m "U1: 2 žingsnis - Bubble sort funkcija"  
git commit -m "U1: 3 žingsnis - Modulinė struktūra"
```

#### Aprašomieji pranešimai

```
git commit -m "U2: Pridėtas copy constructor su deep copy"  
git commit -m "U3: Pataisyta memory leak destruktoriuje"
```

#### Dažni commit'ai (po kiekvienos reikšmingos modifikacijos)

### Bogi commit'ai:

#### Vienas commit visai užduočiai

```
git commit -m "U1 done" # Blogai!
```

#### Neaprašomieji pranešimai

```
git commit -m "fix"      # Blogai!  
git commit -m "asdf"     # Blogai!  
git commit -m "commit"   # Blogai!
```

## X Reti commit'ai (tik pradžioje ir pabaigoje)

### Commit pranešimų formatas:

U[numeris]: [Trumpas aprašymas]

Pavyzdžiai:

- U1: 1 žingsnis - Hello World ir masyvų įvestis
- U1: 3 žingsnis - Modulinė struktūra (.h/.cpp)
- U2: IntList konstruktorius su dynamic allocation
- U3: Copy constructor - deep copy implementacija

## 📄 README.md reikalavimai

### 1. Projekto README (/README.md) - PRIVALOMAS

#### Šablonas:

```
# C++ Objektinis Programavimas (2026)

**Studentas**: Vardas Pavardė
**Grupė**: XXXXXX
**GitLab**: https://gitlab.mif.vu.lt/\[username\]/cpp-2026
```

---

#### ## 📁 Projekto struktūra

- \*\*U1/\*\* - C++ primityvai ir funkcijų moduliai
- \*\*U2/\*\* - IntList klasė (RAII)
- \*\*U3/\*\* - Kompozicija (Langas/Kambarys)
- \*(bus papildoma...)\*

---

#### ## ✅ Užduočių būsena

| Užduotis | Būsena                                      | Terminas   | Pateikta   |
|----------|---|------------|------------|
| U1       | <input checked="" type="checkbox"/> Atlikta | 2026-02-14 | 2026-02-13 |
| U2       | <input checked="" type="checkbox"/> Vykdoma | 2026-02-28 | -          |
| U3       | <input checked="" type="checkbox"/> Laukia  | 2026-03-14 | -          |

---

#### ## 💻 Kompiliavimas

Kiekviena užduotis turi savo Makefile:

```
```bash
cd U1/05/
make
./programa
```

## Pastabos

- Commit'ai daroma po kiekvieno žingsnio
- Jei klausimai - rašyti dėstytojui

---

### \*\*2. Užduoties README (`/U1/README.md`)\*\* - PRIVALOMAS

\*\*Šablonas:\*\*

```markdown

# U1: C++ Primityvai ir Funkcijų Moduliai

\*\*Būsena\*\*:  Atlikta

\*\*Pateikta\*\*: 2026-02-13

---

##  Žingsniai

| Žingsnis | Direktorija | Aprašymas                    |
|----------|-------------|------------------------------|
| 1        | `01/`       | Hello World + masyvas        |
| 2        | `02/`       | Bubble sort funkcija         |
| 3        | `03/`       | Modulinė struktūra (.h/.cpp) |
| 4        | `04/`       | array → vector<int>          |
| 5        | `05/`       | vector<int> → vector<string> |

---

##  Testavimas

\*\*Testas 1 (skaičiai)\*\*:

Input: 42 17 99 5 0 Output: 5 17 42 99  VEIKIA

\*\*Testas 2 (žodžiai)\*\*:

Input: obuolys bananas citrina - Output: bananas citrina obuolys  VEIKIA

---

## ## ☁ Pagrindinės ižvalgos

1. Modulinė struktūra - `.`h`/.`cpp` separacija patogu
2. Vector daug lankstesnis už masyvą (dinaminis dydis)
3. Tas pats algoritmas veikia su int ir string!

---

## ## ⚠ Problemos (jei buvo)

\*(Nepriivaloma, bet naudinga)\*

\*\*Problema 1\*\*: Makefile TAB vs spaces

\*\*Sprendimas\*\*: Pakeisti spaces į TAB simbolius

---

## ## 📂 Kompiliavimas

```
```bash
```

```
cd 05/
```

```
make
```

```
./programa
```

\*\*Minimumas\*\* (jei tingite):

- Žingsnių lentelė
- Bent 1-2 testai
- Kompiliavimo instrukcijos

---

### \*\*3. Žingsnio README (`/U1/01/README.md`)\*\* - NEPRIVALOMAS

Jei \*\*tikrai\*\* norite, galite pridėti trumpas pastabas kiekviename žingsnyje, bet \*\*ne būtina\*\*!

---

## ## 📂 Pateikimas Moodle

### \*\*1 būdas: Git archive (rekomenduojama)\*\*

```
```bash
```

```
cd cpp-2026
```

```
# Sukurti archyvą tik su U1 užduotimi
```

```
git archive --format=zip --output=U1_VardasPavarde.zip HEAD U1/  
  
# ARBA visa repo archyvas  
git archive --format=zip --output=cpp2026_VardasPavarde.zip HEAD
```

**Pliusai:**

- Archyvojoja tik commit'intus failus (ne "junk" failus)
- Automatiškai ignoruoja `.o`, `programa`, ir kt.

**2 būdas: Rankinis zip**

```
cd cpp-2026  
  
# Išvalyti compiled failus  
cd U1/05  
make clean  
cd ../../  
  
# Sukurti archyvą  
zip -r U1_VardasPavarde.zip U1/ README.md .gitignore
```

**Minusai:**

- Reikia rankiniu būdu išvalyti
- Galite įtraukti "junk" failus

**Archyvo vardas:**

U[numeris]\_VardasPavarde.zip

Pavyzdžiai:

- U1\_JonasJonaitis.zip
- U2\_PetrasPetraitis.zip

**Kas turi būti archyve:** **Privaloma:**

- `/U1/` direktorija su visais žingsniais
- `/U1/README.md`
- `/README.md` (projekto root README)
- `/.gitignore`

 **Neturi būti:**

- \*.**o** failai (compiled object files)
  - Executable failai (**programa**, **a.out**, etc.)
  - Editor junk (**.vscode/**, **.idea/**, **\*~**)
- 

## Moodle pateikimo workflow:

1. **Sukurti archyvą** (žr. aukščiau)
2. **Eiti į Moodle** → C++ kursas → Užduotis U1
3. **Upload failą:** **U1\_VardasPavarde.zip**
4. **Pridėti GitLab URL** (comment/text field):

GitLab repo: [https://gitlab.mif.vu.lt/\[username\]/cpp-2026](https://gitlab.mif.vu.lt/[username]/cpp-2026)  
Commit hash: abc123def456

## 5. Submit

**Terminas:** Žiūrėkite užduoties aprašyme (pvz., U1.md)

---

## Vertinimas

### Kas vertinama:

| Kriterijus                                     | %           |
|--|-------------|
| <b>Programos funkcionalumas</b>                | 50%         |
| └ Programa kompiliuojasi be klaidų             | 15%         |
| └ Teisingai atlieka užduotį                    | 25%         |
| └ Testai praeina                               | 10%         |
| <b>Kodo kokybė</b>                             | 30%         |
| └ Aiškūs komentarai                            | 10%         |
| └ Modulinė struktūra                           | 10%         |
| └ Geros praktikos (header guards, const, etc.) | 10%         |
| <b>Git ir dokumentacija</b>                    | 20%         |
| └ Commit'ų kokybė                              | 10%         |
| └ README.md kokybė                             | 10%         |
| <b>TOTAL</b>                                   | <b>100%</b> |

---

### Commit'ų vertinimas:

| Commit stilus                                 | Balai |
|---|-------|
| Po kiekvieno žingsnio, aprašomieji pranešimai | 100%  |
| Keli commit'ai, bet ne visi žingsniai         | 70%   |
| Vienas commit "U1 done"                       | 30%   |
| Nėra commit'ų / tik vienas pradinis           | 0%    |

## README.md vertinimas:

| README kokybė                        | Balai |
|--------------------------------------|-------|
| Pilnas (žingsniai, testai, įžvalgos) | 100%  |
| Minimalus (tik žingsniai + testai)   | 70%   |
| Tik failų sąrašas                    | 40%   |
| Tuščias arba nėra                    | 0%    |

## ?

## DUK

### K: Ar galiu naudoti branch'us vietoj subdirektorijų?

A: Taip, **galite**, bet **neprivaloma**.

### Pavyzdys su branch'ais:

```
git checkout -b u1-step1
# ... darbas ...
git commit -m "U1: 1 žingsnis"

git checkout -b u1-step2
# ... darbas ...
git commit -m "U1: 2 žingsnis"

# Galutinis merge į main
git checkout main
git merge u1-step5
```

Bet subdirektorijos (01/, 02/, ...) **paprastesnės ir labiau atitinka paskaitų medžiągą** (Stack Evolution stilius).

### K: Ar reikia Makefile kiekviename žingsnyje?

A: Ne, **tik nuo 3 žingsnio** (kai turite kelis .cpp failus).

- Žingsniai 1-2: g++ main.cpp -o programa pakanka
- Žingsniai 3-5: **Makefile rekomenduojamas** (daug failų)

## K: Ar galiu naudoti IDE (VS Code, CLion)?

A: Taip, bet:

- **Itraukite .gitignore** ignoruoti IDE failus
- **Programa turi kompiliuotis su Makefile** (ne tik IDE)
- **Nejtraukite .vscode/, .idea/** į repo

## K: Ką daryti, jei pamiršau commit'inti?

A: Commit'inkite dabar!

```
# Jei jau padarėte kelis žingsnius be commit'ų:  
git add U1/01/  
git commit -m "U1: 1 žingsnis (late commit)"  
  
git add U1/02/  
git commit -m "U1: 2 žingsnis (late commit)"
```

**Geriau vėliau nei niekada!** Bet **ateityje** commit'inkite **dažnai**.

## K: Ar senasis kodas turi būti užkomentuotas ar ištrinti?

A: Priklauso nuo užduoties:

- **U1:** Užkomentuoti (žr. U1.md reikalavimus)
- **U2-U9: Ištrinti** (senasis kodas - tai praeitų žingsnių direktoriujos)

**Git saugo visą istoriją**, todėl galite ištrinti seną kodą - jis vis tiek matomas commit'uose!

## K: Ar .gitignore privalomas?

A: **Taip!** Itraukite šį minimalų .gitignore:

```
# Compiled files  
*.o  
*.out  
programa  
a.out  
  
# Editor files  
*~  
.vscode/  
.idea/  
*.swp
```

```
# OS files  
.DS_Store  
Thumbs.db
```

---

## K: Ką daryti, jei GitLab "permission denied"?

A: Patikrinkite SSH raktus:

```
# Sugeneruoti SSH raktą (jei neturite)  
ssh-keygen -t ed25519 -C "your.email@mif.vu.lt"  
  
# Nukopijuoti public key  
cat ~/.ssh/id_ed25519.pub  
  
# Ištraukti į GitLab:  
# Settings → SSH Keys → Add key
```

Arba naudokite HTTPS:

```
git clone https://gitlab.mif.vu.lt/[username]/cpp-2026.git
```

---

## K: Ar galiu dirbti grupėje?

A: Ne, kiekvienas studentas turi **savo repo**.

Bet galite:

- Diskutuoti idėjas
- Padėti debug'inti (ne duoti kodo!)
- Kopijuoti kodą (plagiatas!)

---

## K: Kiek laiko užtrunka užduotis?

A: Priklauso nuo užduoties ir jūsų patirties:

- **U1:** 3-5 valandos (pradedantiesiems), 2-3 val (patyrusiems)
- **U2-U3:** 4-6 valandos
- **U4-U7:** 5-8 valandos
- **U8-U9:** 8-12 valandų (projektas)

**Patariu:** Pradėti **anksčiau**, ne laukti paskutinės dienos! 😊

---

## K: Kam kreiptis pagalbos?

**A:**

1. **Pirmiausia:** Perskaityti užduoties aprašymą (U1.md, U2.md, ...)
  2. **Antra:** Pažiūrėti Stack Overflow, cppreference.com
  3. **Trečia:** Klausti dėstytojo (Moodle arba email)
  4. **Paskutinis būdas:** Klausti kolegos (bet **ne kopijuoti** kodą!)
- 

## Naudingos nuorodos

- **GitLab dokumentacija:** <https://docs.gitlab.com/>
  - **Git tutorial:** <https://git-scm.com/book/en/v2>
  - **Markdown sintaksė:** <https://www.markdownguide.org/>
  - **C++ reference:** <https://en.cppreference.com/>
  - **Makefile tutorial:** <https://makefiletutorial.com/>
- 

## Kontaktai

**Dėstytojas:** [Vardas Pavardė]

**Email:** [email@mif.vu.lt]

**Konsultacijos:** [Laikas ir vieta]

**Moodle:** [Nuoroda]

---

**Sėkmės!** 