Užduočių Atlikimo ir Pateikimo Gidas

Kursas: Objektinis programavimas C++
Semestras: 2026 pavasaris
Versija: 1.0

📋 Turinys

🚀 GitLab projekto sukūrimas

1 žingsnis: Sukurti repo GitLab'e

1. Eikite į fakulteto GitLab: `https://git.mif.vu.lt`
2. Prisijunkite su MIF (!) 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**: `Maintainer`

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/                  ← 1 žingsnis
│   │   └── main.cpp
│   ├── 02/                  ← 2 žingsnis
│   │   └── main.cpp
│   ├── 03/                  ← 3 žingsnis
│   │   ├── 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	☑ TAIP
Užduotis	`/U1/README.md`	Užduoties santrauka, testavimas, pastabos	☑ TAIP
Žingsnis	`/U1/01/README.md`	*Nebūtina*	❌ 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 užduoties U1 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 masyvų įvestis"
git push
```

Žingsnis 4: Atlikti užduoties U1 2 žingsnį

```
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 - užduoties README

```
cd U1
# Sukurti README.md (žr. šabloną ...)
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 įvestis"
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)

Blogi 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!
```

❌ **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

Kiekvienas lygis turi savo šabloną (`README-templates/` direktorijoje). Nukopijuokite šabloną į reikiamą vieta ir užpildykite savo duomenys.

Lygis	Failas	Šablonas	Privaloma?
Projektas	`/README.md`	`README-project.md`	☑ TAIP
Užduotis	`/U1/README.md`	`README-assignment.md`	☑ TAIP
Žingsnis	`/U1/01/README.md`	`README-step.md`	❌ NE

---

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

Šablonas: `README-project.md`

Turi:

- Studentas: vardas, pavardė, grupė, GitLab URL
- Projekto struktūra (direktorijų sąrašas)
- Užduočių būsenos lentelė (atnaujinti po kiekvienos pateiktos užduoties)

---

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

Šablonas: `README-assignment.md`

Turi:

- Žingsnių lentelė su aprašymais ir būsenomis
- Bent 1-2 testai (Input/Output formatu)
- Kompiliavimo instrukcijos (`make` arba `g++`)

Rekomenduojama papildyti:

- Įžvalgos (ką naujo išmokote)
- Problemos ir jų sprendimai

---

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

Šablonas: `README-step.md`

Trumpos pastabos per žingsniui — naudingia debug'inimo metu, bet neprivaloma.

Pateikimas Moodle

1 būdas: Git archive (rekomenduojama)

```
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:

- ☑ Archyvuoja 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
   Commit hash: abc123def456
   ```

5. **Submit**

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

❓ DUK

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

A: Taip, **bet**:

- ☑ Įtraukite `.gitignore` ignoruoti IDE failus
- ☑ Programa turi kompiliuotis iš komandinės eilutės (ne tik IDE)
- ❌ Neįtraukite `.vscode/`, `.idea/` į repo

K: Ką daryti, jei pamiršau commit'inti?

A: Commit'inkite dabar!

```
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 po kiekvieno žingsnio.

K: Ar galiu dirbti grupėje?

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

Bet galite:

- ☑ Diskutuoti idėjas
- ☑ Padėti debug'inti (neduoti 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)

Pradėti **anksčiau**, nelaukti paskutinės dienos! 😊

K: Į ką kreiptis pagalbos?

A:

1. Perskaityti užduoties aprašymą (U1.md, U2.md, ...)
2. Pažiūrėti Stack Overflow, cppreference.com
3. Klausti dėstytojo (auditorijoje, Teams arba email)
4. Klausti kolegos (bet **ne kopijuoti** kodo!)

🔗 Naudingos nuorodos

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