U1: C++ Primityvai ir Funkcijų Moduliai

Savaitės: 1-2
Svoris: 1 balas
Terminas: Savaitės 2 pabaiga

Prieš pradedant

SVARBU: Jei dar nesukūrėte GitLab projekto, pirmiausia perskaitykite:
Užduočių Atlikimo ir Pateikimo Gidas

Ten rasite:

- GitLab projekto sukūrimo instrukcijas
- Projekto struktūrą
- Git workflow gaires
- README.md šablonus
- Pateikimo į Moodle procesą

Užduoties tikslas

Įsivažiuoti su C++ sintakse, išmokti dirbti su masyvais, konteineriais ir funkcijomis. Suprasti modulinę programų struktūrą.

Mokymosi tikslai

Atlikę šią užduotį, mokėsite:

- ☑ Naudoti C++ įvesties/išvesties srautus (`cin`, `cout`)
- ☑ Dirbti su masyvais ir `vector` konteineriais
- ☑ Rašyti ir kviesti funkcijas
- ☑ Organizuoti kodą į kelių failų struktūrą (`.h`/`.cpp`)
- ☑ Kompiliuoti programą iš kelių failų

📋 Užduoties žingsniai

1 žingsnis. "Hello C++" ir masyvų įvestis

Reikalavimai:

1. Parašykite programą, kuri:
    - Išveda "Hello, C++!" į konsolę
    - Leidžia vartotojui įvesti sveikus skaičius
    - Talpina skaičius į fiksuoto dydžio masyvą (pvz., 100 elementų)
    - Baigia įvedimą, kai vartotojas įveda `0`
    - Atspausdina visus įvestus skaičius

Technikos:

- Naudokite `cout` išvedimui
- Naudokite `cin` įvedimui
- Naudokite `int` tipo masyvą: `int skaiciai[100];`
- Saugokite įvestų skaičių kiekį atskirame kintamajame

Pavyzdys:

```
Hello, C++!
Įveskite sveikus skaičius (0 - baigti):
42
17
99
0
Įvesti skaičiai: 42 17 99
```

---

2 žingsnis. Bubble sort funkcija

Reikalavimai:

1. Parašykite funkciją `rusiuotiMasyva()`, kuri:
    - Priima masyvą ir jo dydį
    - Surūšiuoja masyvo elementus nuo mažiausio iki didžiausio
    - Naudoja **Bubble sort** algoritmą
2. Programos `main()` funkcijoje:
    - Iškvieskite rūšiavimo funkciją
    - Atspausdinkite surūšiuotus skaičius

Technikos:

- Funkcijos deklaracija: `void rusiuotiMasyva(int masyvas[], int dydis);`
- Bubble sort: lyginkite gretimus elementus ir keiskite vietomis

Pavyzdys:

```
Įvesti skaičiai: 42 17 99
Surūšiuoti skaičiai: 17 42 99
```

---

3 žingsnis. Modulinė struktūra (.h + .cpp)

Reikalavimai:

1. Sukurkite **3 failus** direktorijoje `U1/03/`:
   - `main.cpp` - pagrindinė programa (tik `main()` funkcija)
   - `rusiavimas.h` - funkcijų deklaracijos
   - `rusiavimas.cpp` - funkcijų implementacijos
2. `rusiavimas.h` turi turėti:
   - Header guard (`#ifndef`, `#define`, `#endif`)
   - Funkcijos `rusiuotiMasyva()` deklaraciją
   - Funkcijos `spausdintiMasyva()` deklaraciją (nauja!)
3. `rusiavimas.cpp` turi turėti:
   - `#include "rusiavimas.h"`
   - Funkcijų implementacijas
4. `main.cpp` turi turėti:
   - `#include "rusiavimas.h"`
   - Tik `main()` funkciją

Kompiliacija:

```
cd U1/03/
g++ -c main.cpp
g++ -c rusiavimas.cpp
g++ main.o rusiavimas.o -o programa
./programa
```

Arba naudojant Makefile (rekomenduojama).

---

4 žingsnis. Evoliucija: array → vector<int>

Reikalavimai:

1. Pakeiskite masyvą `int skaiciai[100]` į `vector<int> skaiciai`
2. Vietoj `skaiciai[kiekis++] = x` naudokite `skaiciai.push_back(x)`
3. Vektoriaus dydį gaukite su `skaiciai.size()`
4. Adaptuokite rūšiavimo funkciją dirbti su `vector<int>`

SVARBU:

- **Užkomentuokite seną kodą** su masyvais, bet **palikite jį matytis** faile
- Tai leis dėstytojui įsitikinti, kad atlikote visus žingsnius

Technikos:

- `#include <vector>`
- `vector<int>` vietoj `int[]`
- Funkcija gali būti: `void rusiuotiVektoriu(vector<int>& skaiciai)`
- Perduokite vektorių **per nuorodą** (`&`)

---

5 žingsnis. Evoliucija: vector<int> → vector<string>

Reikalavimai:

1. Sukurkite naują programos versiją, kuri:
    - Įveda žodžius (ne skaičius)
    - Baigia įvedimą, kai įvedamas `-`
    - Rūšiuoja žodžius abėcėlės tvarka
    - Atspausdina surūšiuotus žodžius
2. Naudokite `vector<string>` konteinerį
3. Rūšiavimo funkcija turi veikti su `string` tipu

SVARBU:

- Vėl **užkomentuokite seną kodą** (su `int`), bet **palikite matytis**

Technikos:

- `#include <string>`
- `vector<string>` vietoj `vector<int>`
- `string` tipas palaiko `<` operatorių (abėcėlės tvarka)

Pavyzdys:

```
Įveskite žodžius ('-' baigti):
obuolys
bananas
citrina
-
Surūšiuoti žodžiai: bananas citrina obuolys
```

Pateikimas

GitLab direktorijų struktūra:

```
cpp-2026/
├── README.md              ← Projekto README (žr. UzduotiuGidas.md)
├── .gitignore
└── U1/
    ├── 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
    │   ├── main.cpp
    │   ├── rusiavimas.h
    │   ├── rusiavimas.cpp
    │   └── Makefile
    └── 05/                ← 5 žingsnis (FINAL)
        ├── main.cpp
        ├── rusiavimas.h
        ├── rusiavimas.cpp
        └── Makefile
```

Git workflow (tariamas):

Po kiekvieno žingsnio:

```
cd cpp-2026/U1/01/
# ... atlikti darbą ...

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

Moodle pateikimas (tariamas):

1. Sukurti archyvą:

```
cd cpp-2026
git archive --format=zip --output=U1_VardasPavarde.zip HEAD U1/ README.md
.gitignore
```

2. Įkelti į Moodle su GitLab URL

Patarimai

1. **Perskaitykite Užduočių Gidas** prieš pradedant
2. **Git workflow**:
    - Commit'inkite **po kiekvieno žingsnio**
    - Push'inkite **dažnai** (backup!)
3. **Pradėkite paprastai** - pirmiausia paleiskite 1 žingsnį, tada tęskite
4. **Kompiliuokite dažnai** - po kiekvieno žingsnio
5. **Testuokite su skirtingais duomenimis** - teigiami, neigiami skaičiai
6. **Klauskite, jei neaišku** - geriau anksčiau nei vėliau!

Naudingos nuorodos

- C++ vector dokumentacija
- C++ string dokumentacija
- Makefile tutorial

❓ Dažnai užduodami klausimai

K: Ar galiu naudoti `std::sort()` vietoj Bubble sort?
A: Ne, šioje užduotyje privaloma implementuoti Bubble sort patys.

K: Ar privalau naudoti Makefile?
A: Ne, bet rekomenduojama nuo 3 žingsnio. Galite kompiliuoti rankiniu būdu.

K: Kiek testų reikia README faile?
A: Bent 2 - vienas su skaičiais, vienas su žodžiais.

K: Ar galiu naudoti branch'us vietoj subdirektorijų?
A: Taip, bet neprivaloma. Subdirektorijos (`01/`, `02/`, ...) paprastesnės.

Daugiau klausimų? → Žr. Užduočių Gidas - DUK

Sėkmės! 🚀