U2: IntList Klasė ir RAII

Savaitės: 3-4
Svoris: 1 balas
Terminas: Savaitės 4 pabaiga

Prieš pradedant

Priminimas: Šiai užduočiai taikomi tie patys reikalavimai kaip ir U1.
👉 Žr. **Užduočių Gidas** dėl:

- GitLab projekto struktūros
- Git commit'ų gairių
- README.md šablonų
- Pateikimo į Moodle

Užduoties tikslas

Sukurti pirmą klasę su dinaminiu atminties valdymu. Išmokti RAII (Resource Acquisition Is Initialization) principą - kad konstruktorius išskiria resursus, o destruktorius juos atlaisvina.

Mokymosi tikslai

Atlikę šią užduotį, mokėsite:

- ☑ Sukurti klasę su private/public nariais
- ☑ Rašyti konstruktorius (default ir su parametrais)
- ☑ Rašyti destruktorių su logging
- ☑ Dirbti su dinaminiu atminties skyrimu (`new`/`delete`)
- ☑ Implementuoti automatinį konteinerio išplėtimą
- ☑ Suprasti RAII principą
- ☑ Organizuoti klasę į `.h`/`.cpp` failus

Kas yra IntList?

IntList - tai **dinaminis sąrašas sveikiems skaičiams** (panašus į `vector<int>`).

Funkcionalumas:

- Saugoja sveikus skaičius
- Automatiškai išsiplečia, kai reikia daugiau vietos
- Leidžia pridėti elementus į pabaigą
- Leidžia gauti elementą pagal indeksą

Skirtumas nuo U1:

- U1: **funkcijos** su masyvais/vektoriais
- U2: **klasė** su inkapsuliuotu dinaminiu masyvu

📋 Užduoties žingsniai

1 žingsnis: Basic klasė su fiksuotu masyvu

Direktorija: `U2/01/`

Reikalavimai:

Sukurkite `IntList` klasę su:

1. **Private nariai:**
   - `int duomenys[100]` - fiksuoto dydžio masyvas
   - `int dydis` - dabartinis elementų skaičius
2. **Public metodai:**
   - `IntList()` - default konstruktorius (inicializuoja `dydis = 0`)
   - `void pridetiGala(int reiksme)` - pridėti elementą į pabaigą
   - `int gautiElementa(int indeksas) const` - gauti elementą pagal indeksą
   - `int gautiDydi() const` - gauti dabartinį dydį
   - `void spausdinti() const` - atspausdinti visus elementus
3. **Modulinė struktūra:**
   - `IntList.h` - klasės deklaracija
   - `IntList.cpp` - klasės implementacija
   - `main.cpp` - testavimo programa
   - `Makefile` - kompiliavimo automatizavimas

Testas:

```
IntList sarasas;
sarasas.pridetiGala(10);
sarasas.pridetiGala(20);
sarasas.pridetiGala(30);
sarasas.spausdinti();  // [10, 20, 30]
```

Pavyzdys:

```
IntList dydis: 3
Elementai: [10, 20, 30]
```

---

2 žingsnis: Dynamic memory + konstruktoriai/destruktorius

Direktorija: `U2/02/`

Reikalavimai:

Modifikuokite `IntList` klasę:

1. **Pakeisti private narius:**
   - `int* duomenys` - **rodyklė** į dinaminį masyvą (ne fixed array!)
   - `int dydis` - dabartinis elementų skaičius
   - `int talpa` - išskirtos atminties talpa
2. **Konstruktoriai:**
   - `IntList()` - default: talpa = 10, išskirti atmintį
   - `IntList(int pradineTalpa)` - su parametru: išskirti nurodytą talpą
   - **Svarbu:** Konstruktoriuje naudoti `new int[talpa]`
3. **Destruktorius:**
   - `~IntList()` - atlaisvinti atmintį su `delete[]`
   - **Logging:** Išvesti pranešimą į `cout` (debug)
   - **Nustatyti rodyklę** `nullptr` (saugumui)
4. **Metodai:**
   - Išlaikyti visus metodus iš 1 žingsnio
   - `pridetiGala()` - dabar prideda į **dinaminį** masyvą
   - **Patikrinti:** `if (dydis >= talpa)` → klaida (kol kas be auto-expand)

Logging pavyzdys:

```
~IntList() {
    cout << "[DEBUG] IntList naikinamas (dydis=" << dydis
         << ", talpa=" << talpa << ")" << endl;
    delete[] duomenys;
    duomenys = nullptr;
}
```

Testas:

```
{
    IntList sarasas(5);  // Talpa = 5
    for(int i = 1; i <= 5; i++) {
        sarasas.pridetiGala(i * 10);
    }
    sarasas.spausdinti();
} // Destruktorius čia iškviečiamas automatiškai!
```

Pavyzdys:

```
[DEBUG] IntList sukurtas (talpa=5)
IntList dydis: 5
Elementai: [10, 20, 30, 40, 50]
[DEBUG] IntList naikinamas (dydis=5, talpa=5)
```

---

3 žingsnis: Automatinis išplėtimas

Direktorija: `U2/03/`

Reikalavimai:

Pridėkite automatinį atminties išplėtimą:

1. **Private metodas:**
   - `void isplesti()` - išplečia masyvo talpą
   - Algoritmas:

     ```
     1. Apskaičiuoti naują talpą (pvz., talpa + 5)
     2. Išskirti naują didesnį masyvą
     3. Nukopijuoti senus duomenis
     4. Atlaisvinti seną masyvą
     5. Priskirti naują masyvą
     ```

2. **Modifikuoti `pridetiGala()`:**

   ```
   void pridetiGala(int reiksme) {
       if (dydis >= talpa) {
           isplesti();  // Automatinis išplėtimas!
       }
       duomenys[dydis++] = reiksme;
   }
   ```

3. **Logging `isplesti()` metode:**

   ```
   cout << "[DEBUG] IntList isplesta (sena talpa=" << talpa
        << ", nauja talpa=" << naujaTalpa << ")" << endl;
   ```

Testas su mažu `talpa`:

```
IntList sarasas(3);  // Pradinis talpa = 3
for(int i = 1; i <= 10; i++) {
    sarasas.pridetiGala(i * 10);  // Turi išsiplėsti!
}
sarasas.spausdinti();
```

Pavyzdys:

```
[DEBUG] IntList sukurtas (talpa=3)
[DEBUG] IntList isplesta (sena talpa=3, nauja talpa=8)
[DEBUG] IntList isplesta (sena talpa=8, nauja talpa=13)
IntList dydis: 10
Elementai: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[DEBUG] IntList naikinamas (dydis=10, talpa=13)
```

Pateikimas

GitLab direktorijų struktūra:

```
cpp-2026/
├── README.md
├── .gitignore
├── U1/
│   └── ...
└── U2/
    ├── README.md          ← Užduoties santrauka (PRIVALOMA)
    ├── 01/                ← 1 žingsnis (basic class)
    │   ├── IntList.h
    │   ├── IntList.cpp
    │   ├── main.cpp
    │   └── Makefile
    ├── 02/                ← 2 žingsnis (dynamic memory)
    │   ├── IntList.h
    │   ├── IntList.cpp
    │   ├── main.cpp
    │   └── Makefile
    └── 03/                ← 3 žingsnis (auto-expand) - FINAL
        ├── IntList.h
        ├── IntList.cpp
        ├── main.cpp
        └── Makefile
```

Git workflow:

Po kiekvieno žingsnio:

```
git add U2/01/
git commit -m "U2: 1 žingsnis - Basic IntList klasė"
git push

git add U2/02/
git commit -m "U2: 2 žingsnis - Dynamic memory + RAII"
git push

git add U2/03/
git commit -m "U2: 3 žingsnis - Automatinis išplėtimas"
git push
```

U2/README.md šablonas:

```
# U2: IntList Klasė ir RAII

**Būsena**: ☑ Atlikta
**Pateikta**: 2026-02-28

---

## 📝 Žingsniai

| Žingsnis | Direktorija | Aprašymas |
|----------|-------------|-----------|
| 1 | `01/` | Basic klasė (fiksuotas masyvas) |
| 2 | `02/` | Dynamic memory + RAII |
| 3 | `03/` | Automatinis išplėtimas |

---

## 🧪 Testavimas

**Testas 1 (be išplėtimo)**:
```

Input: talpa=10, pridėti 5 elementus Output: [10, 20, 30, 40, 50] ☑ VEIKIA

```

**Testas 2 (su išplėtimu)**:
```

Input: talpa=3, pridėti 10 elementų Output: [10, 20, ... 100], automatiškai išsiplėtė 2 kartus ☑ VEIKIA

```

---

## 💭 Pagrindinės įžvalgos
```

```
1. RAII principas - konstruktorius išskiria, destruktorius atlaisvina
2. Automatinis išplėtimas - panašiai kaip `vector`
3. Destruktoriaus logging padeda debug'inti
```

Moodle pateikimas:

```
cd cpp-2026
git archive --format=zip --output=U2_VardasPavarde.zip HEAD U2/ README.md
.gitignore
```

Detalios instrukcijos: Žr. Užduočių Gidas

☑ Vertinimo kriterijai

Kriterijus	Balai
Programa kompiliuojasi be klaidų	15%
Basic klasė (1 žingsnis) veikia	15%
Dynamic memory + RAII (2 žingsnis) veikia	30%
Automatinis išplėtimas (3 žingsnis) veikia	20%
Destruktoriaus logging	5%
Modulinė struktūra (`.h`/`.cpp`)	5%
Git commit'ai po kiekvieno žingsnio	5%
README.md su testais	5%
TOTAL	**100%**

Patarimai

1. **Pradėkite nuo 1 žingsnio** - basic klasė su fiksuotu masyvu
2. **Logging yra svarbus** - matysite, kada destruktorius kviečiamas
3. **Testuokite su mažu** `talpa` (pvz., 3) - lengviau pamatyti išplėtimą
4. `nullptr` **po** `delete[]` - gera praktika (safety)
5. **Išsaugokite seną kodą** - naujas žingsnis = nauja direktorija
6. **Commit'inkite dažnai!**

Naudingos nuorodos

- C++ konstruktoriai
- C++ destruktoriai

- RAII idioma
- new/delete operators

❓ Dažnai užduodami klausimai

K: Kodėl `delete[]` o ne `delete`?
A: `delete[]` naudojamas masyvams, `delete` - vienam objektui. **Labai svarbu!**

K: Kodėl reikia `nullptr` po `delete[]`?
A: Safety - jei atsitiktinai bandysime `delete[]` dar kartą, nekris programa.

K: Kaip testuoti destruktorių?
A: Logging (`cout` destruktoriuje) + scope (`{ ... }` blokais).

K: Kodėl automatinis išplėtimas reikalingas?
A: Tai `vector` elgsena - dinaminė talpa. Iš pradžių nežinome, kiek elementų bus.

Daugiau klausimų? → Žr. Užduočių Gidas - DUK

Sėkmės! 🚀