Міністерство освіти і науки України
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Обчислювальна техніка та програмування»

ЗВІТ
Про виконання розрахункового завдання
«Розробка інформаційно-довідкової системи»

Керівник викладач:
Давидов В.В.

Виконавець:
студентка гр. КІТ-120А
Зеленець Олена

Харків 2021

1.Вимоги

1.1 Розробник

Зеленець Олена

студентка групи КІТ-120А

26.05.2021

1.2 Загальне завдання

Закріпити отримані знання з дисципліни “Програмування” шляхом виконання типового комплексного завдання.

1.3 Індивідуальне завдання

На ЖД-станції вирішили виконати перепис потягів. Розробити методи для роботи з колекцією:

- Знайти всі потяги з кількістю вагонів більше 10 та які потребують ремонту.

- Знайти всі вантажні потяги, що прямують з України.

- Знайти вантажний потяг з найбільшою масу серед тих, що перевозять дерево.

1.4 Призначення та галузь застосування

Програма призначена для обробки вхідних даних, створення колекції потягів, обробки та роботи з нею. Застосування програми призначено для людей, які працюють з базою даних потягів та їм необхідно цю базу даних впорядковувати, оновлювати та змінювати.

2. Виконання роботи

2.1 Опис вхідних та вихідних даних

Вхідні дані — файл з таблицею впорядкованих вхідних даних стосовно потягів, дані введені користувачем з клавіатури.

Вихідні дані — файл з обробленою користувачем колекцією потягів, текстові дані виведені у консоль.

2.2 Опис складу технічних та програмних засобів

Комп’ютер будь-якої потужності, IDE для написання коду програми, компілятор для обробки коду, монітор для виведення результатів роботи.

2.3 Опис джерел інформації

Офіційна документації стосовно мови розробки C++

https://docs.microsoft.com/en-us/cpp/?view=msvc-160

Інформація щодо стандартної бібліотеки шаблонів

https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%BE%D0%B2

2.4 Фрагменти коду функціональної частини програми програми

Див. Додаток А

2.5 UML-діаграми класів та їх зв’язків

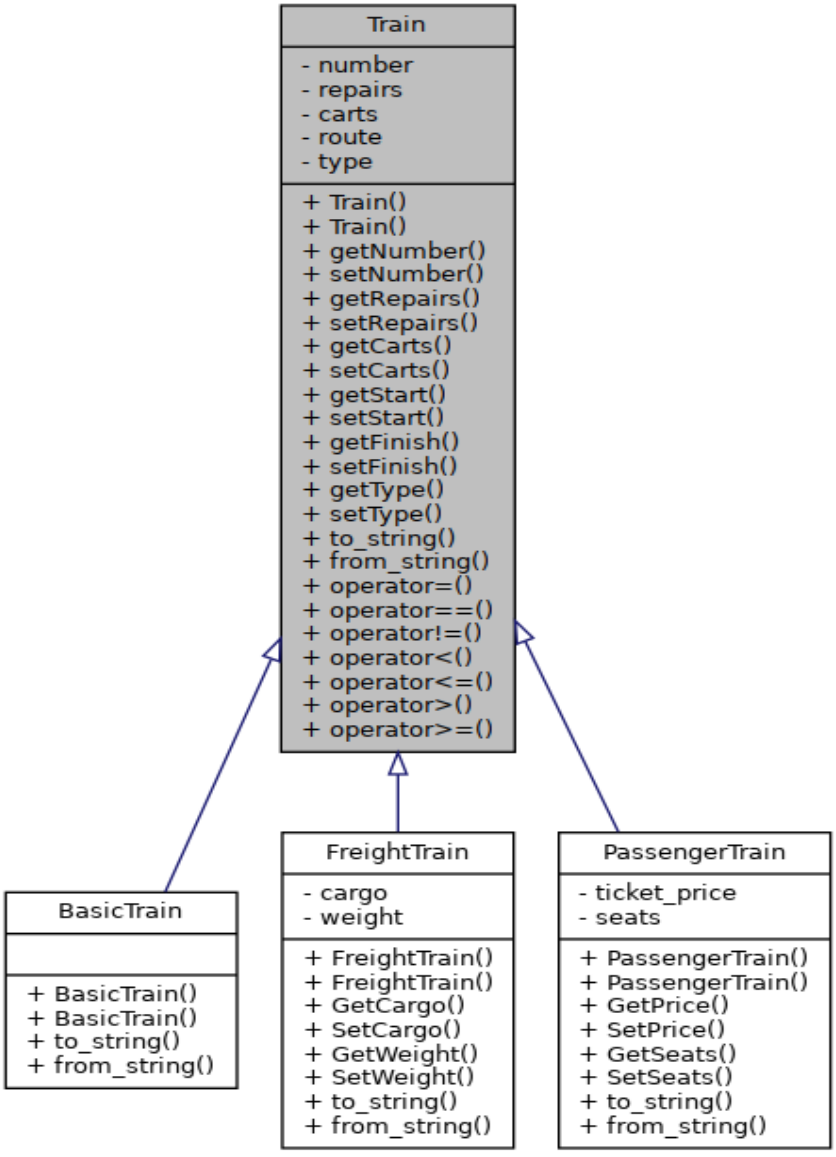

Рисунок 1.1 — *uml-діаграмма успадкувань*

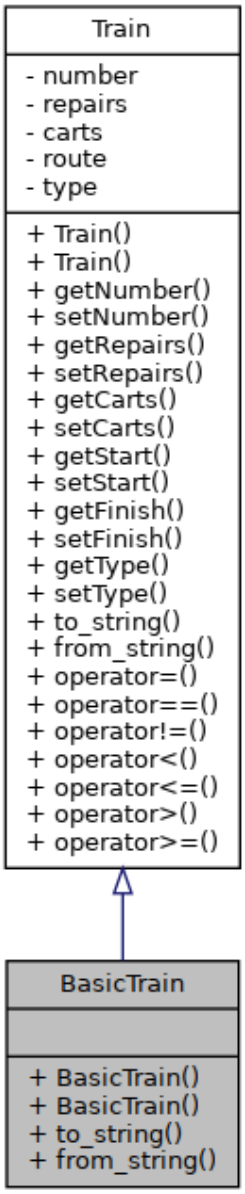

Рисунок 1.2 — uml-діаграмма зв’язків класу BasicTrain

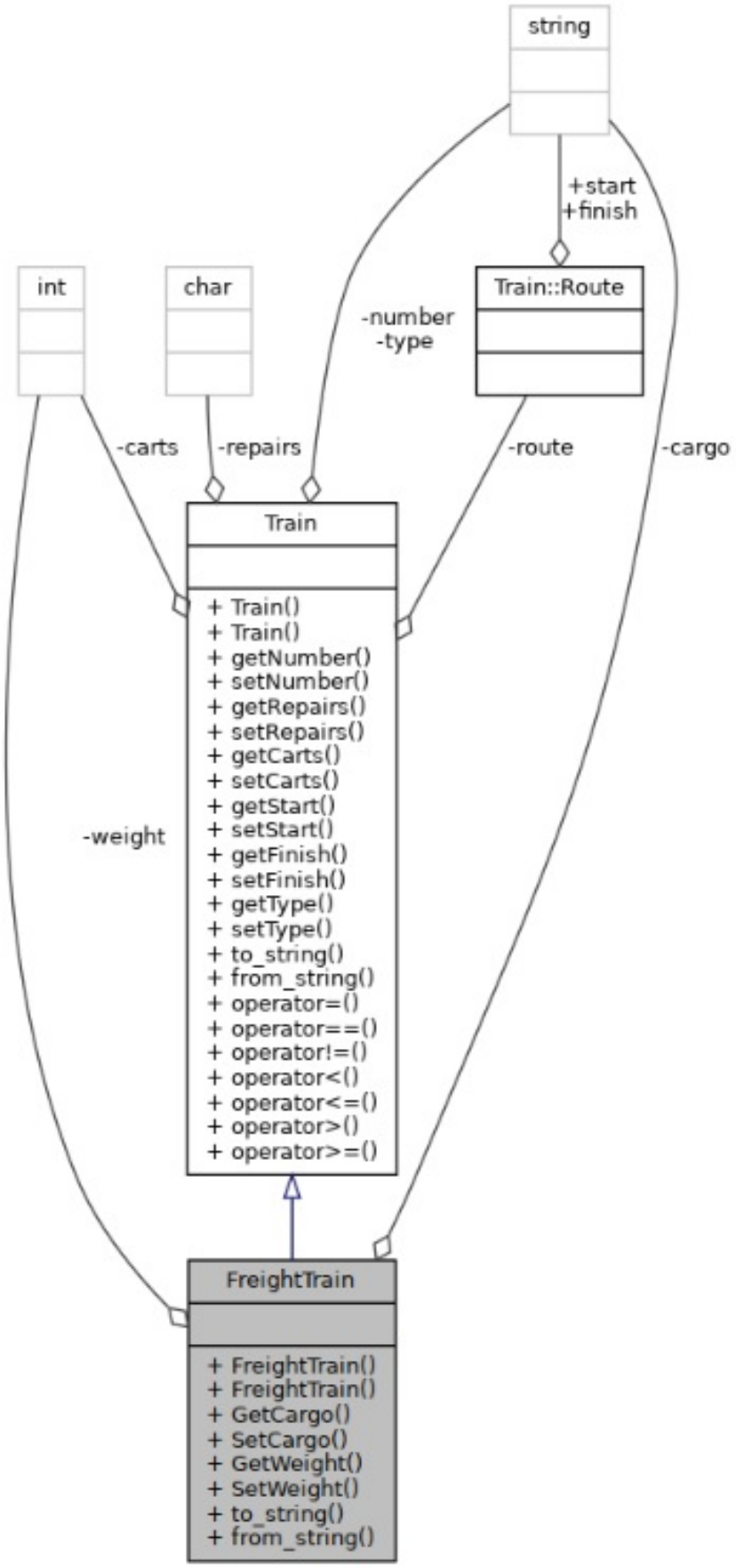

Рисунок 1.3 — uml-діаграмма зв’язків класу FreightTrain

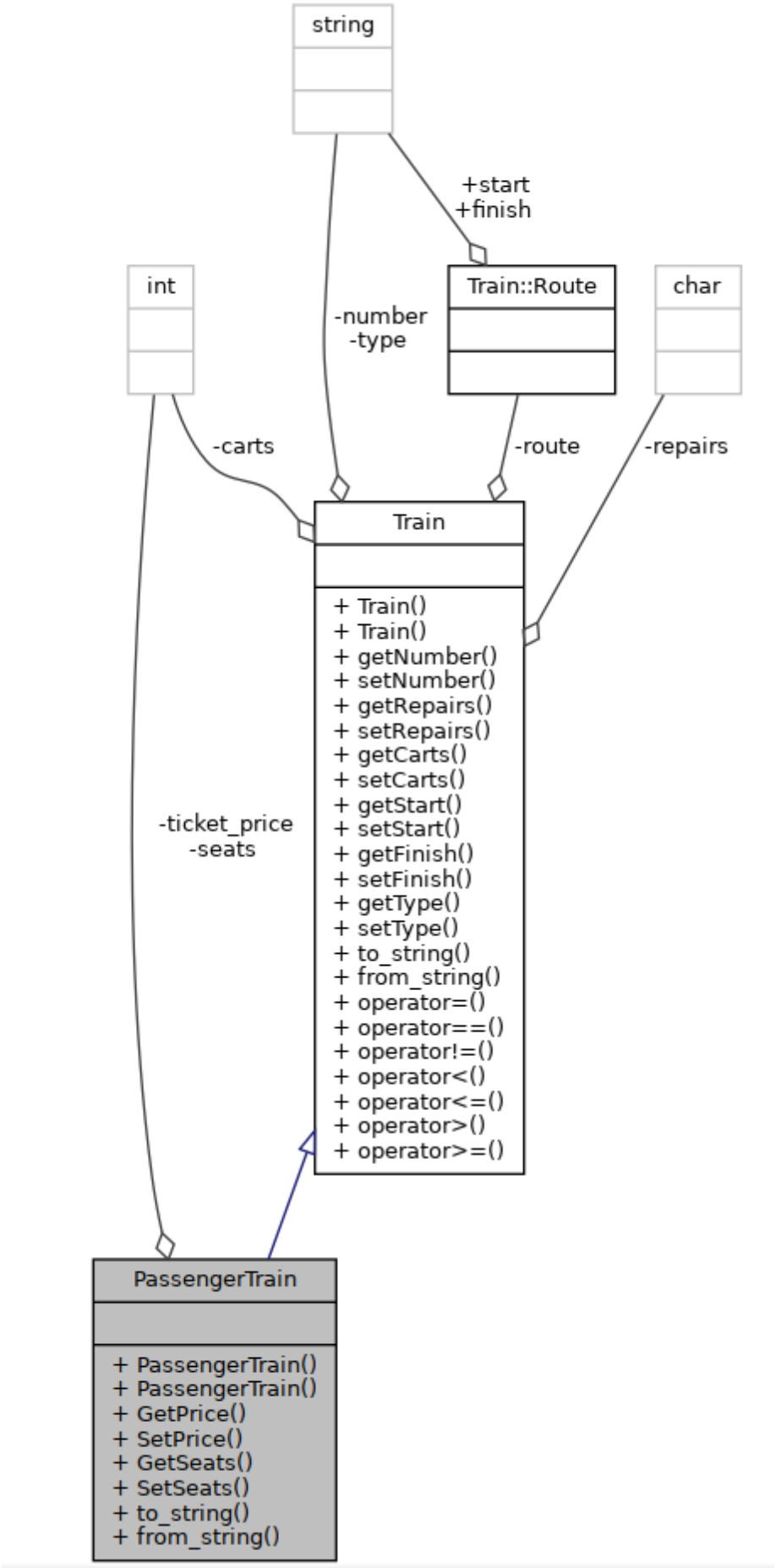

Рисунок 1.4 — uml-діаграмма зв’язків класу PassengerTrain

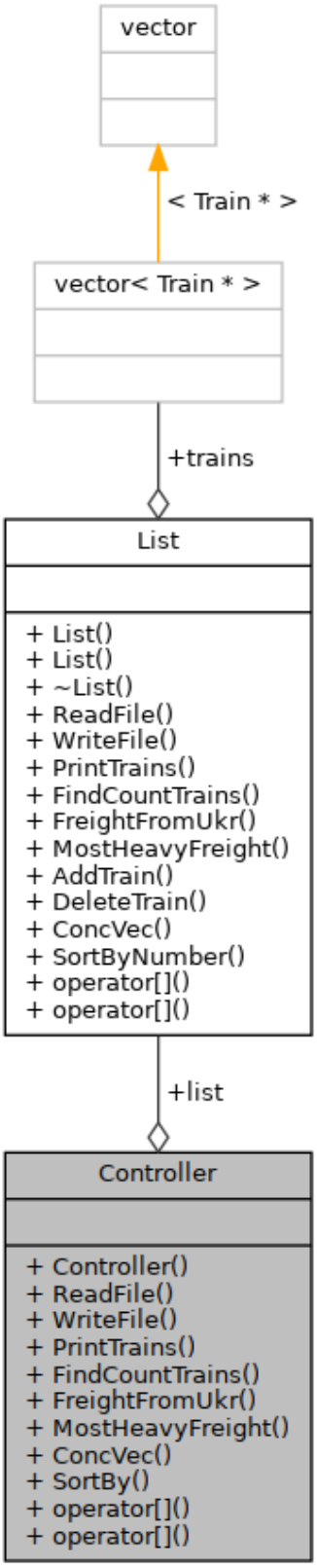

Рисунок 1.5 — uml-діаграмма зв’язків класу List та Controller

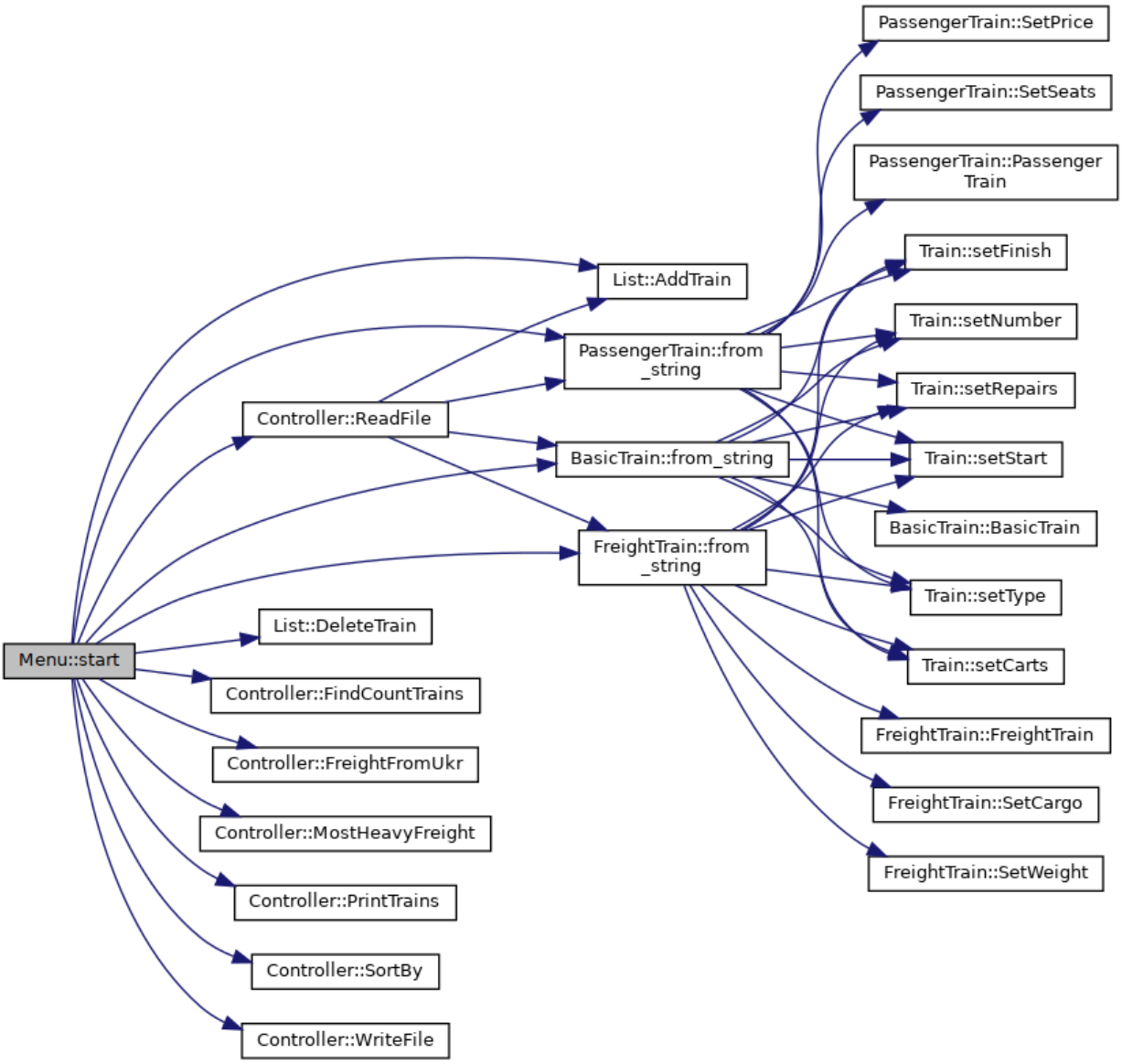

Рисунок 1.6 — Граф виклику функцій класу Menu

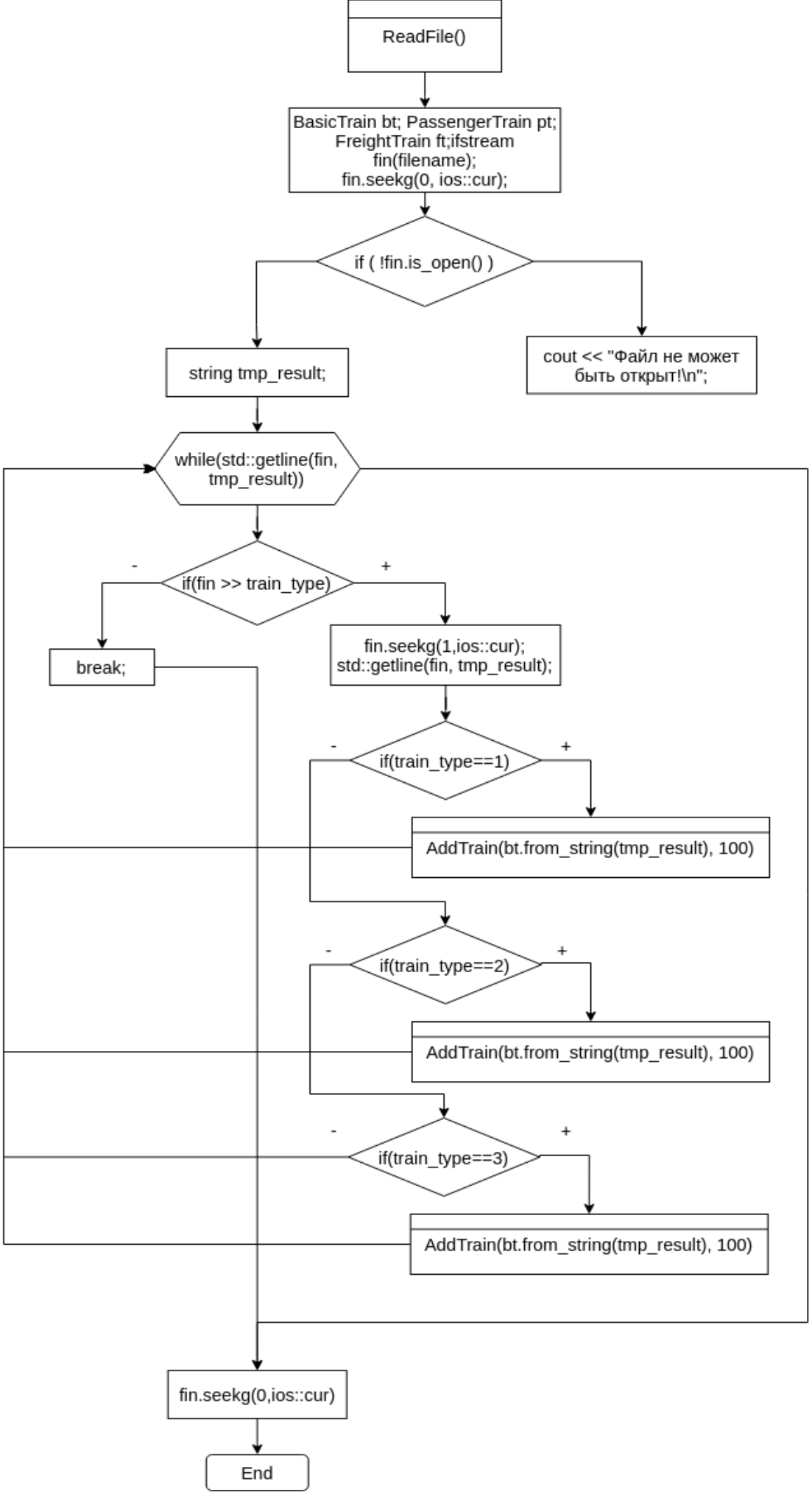

Рисунок 1.7 — Блок-схема функції ReadFile

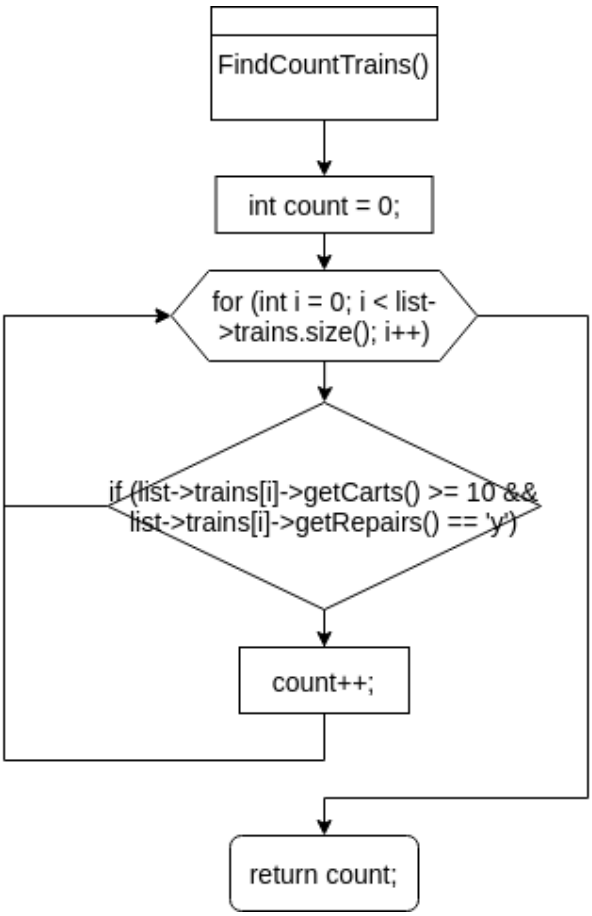

Рисунок 1.8 — Блок-схема функції FindCountTrains

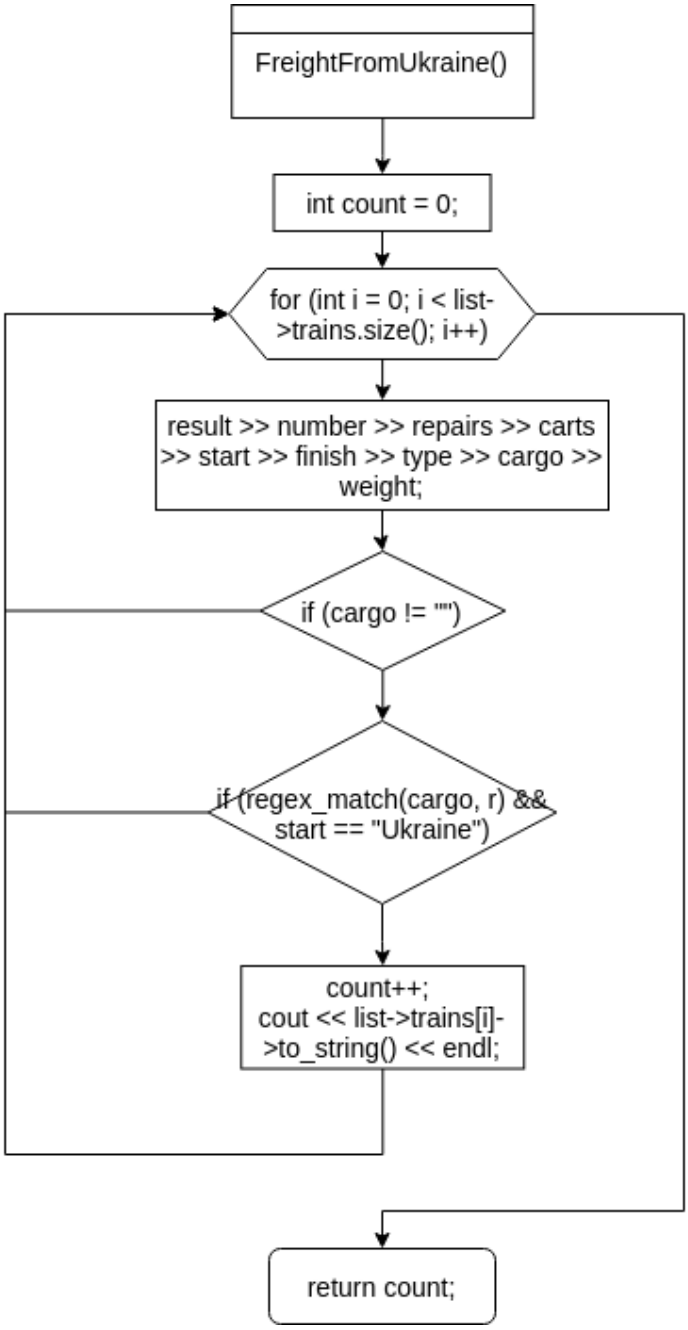

Рисунок 1.9 — Блок-схема функції FreightFromUkraine

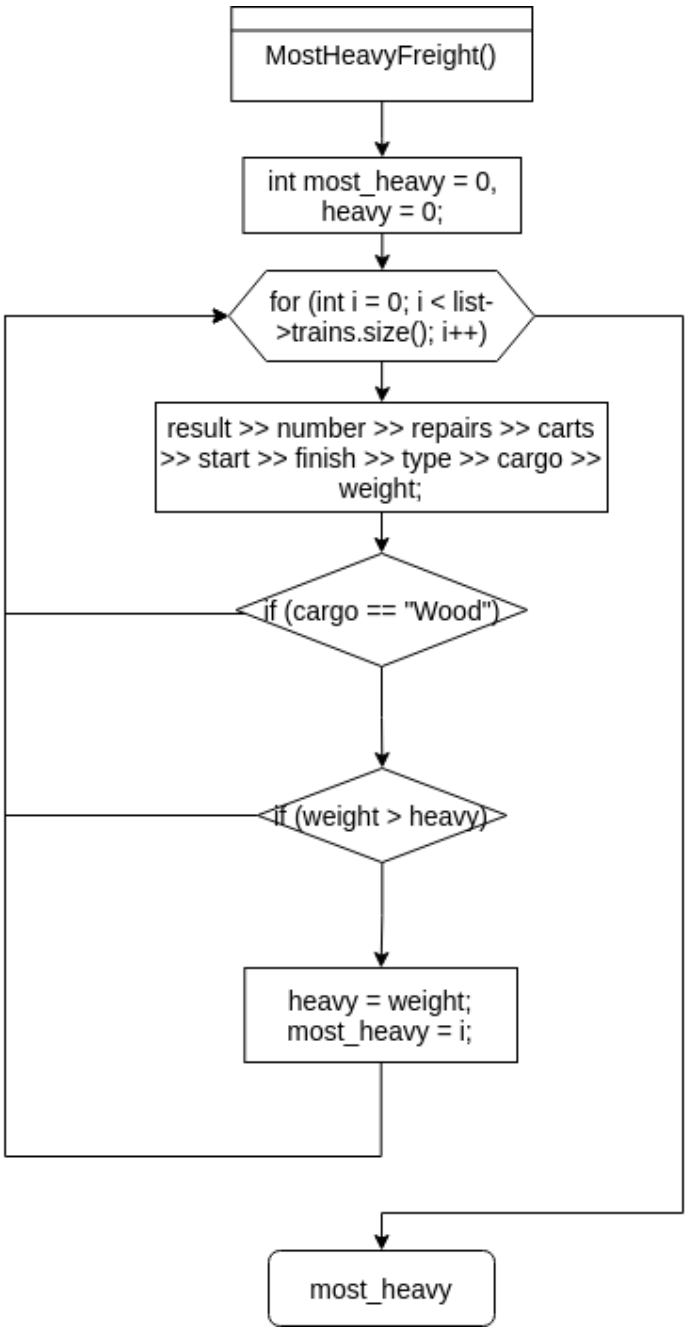

Рисунок 1.10 — Блок-схема функції MostHeavyFreight

2.6 Перевірка програми за допомогою утіліти Valgrind

```
==230415==
==230415== HEAP SUMMARY:
==230415==     in use at exit: 0 bytes in 0 blocks
==230415==   total heap usage: 151 allocs, 151 frees, 88,896 bytes allocated
==230415==
==230415== All heap blocks were freed -- no leaks are possible
==230415==
==230415== For lists of detected and suppressed errors, rerun with: -s
==230415== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Рисунок 1.11 — Утиліта Valgrind

3. Висновки

Я закріпила отримані знання з дисципліни “Програмування” шляхом виконання типового комплексного завдання. Моє індивідуальне завдання було таким:

На ЖД-станції вирішили виконати перепис потягів. Розробити методи для роботи з колекцією:

- Знайти всі потяги з кількістю вагонів більше 10 та які потребують ремонту.
- Знайти всі вантажні потяги, що прямують з України.
- Знайти вантажний потяг з найбільшою масою серед тих, що перевозять дерево.

Я виконала завдання та розробила запропоновані методи по роботі з колекцією. Ця програма призначена для обробки вхідних даних, створення колекції потягів, обробки та роботи з нею. Продемонструвала роботу програми та коректність її виконання.

Додаток А

```
void Controller::ReadFile(string filename)
{
BasicTrain bt;
PassengerTrain pt;
FreightTrain ft;
ifstream fin; //поток для считывания из файла

fin.exceptions(ifstream::badbit | ifstream::failbit); //Возможность самому обрабатывать ошибки потока
try //Попытка открыть файл
{
cout << "Попытка открыть файл..." << endl;
fin.open(filename);
cout << "Файл успешно открыт!" << endl;
}
catch (const std::exception &ex)
{
cout << ex.what() << endl; //Вывод ошибки
cout << "Введите корректное имя файла: ";
cin >> filename;
cout << endl;
Controller::ReadFile(filename); //Будет рекурсировать пока пользователь не введет корректное название файла
}

fin.exceptions(ifstream::goodbit);
fin.seekg(0, ios::cur);

string tmp_result;
while (1) //Пока есть строки в файле
{
int train_type;
if (fin >> train_type)
{ // читаем 1 значение в файле. если 1 - это первый наследник, если 2 - второй
fin.seekg(1, ios::cur);

std::getline(fin, tmp_result);
if (train_type == 1)
list->AddTrain(bt.from_string(tmp_result), 100); //Строку, которую считали, заполняем в базовый класс
else if (train_type == 2)
list->AddTrain(pt.from_string(tmp_result), 100); //Строку, которую считали, заполняем в базовый класс
else if (train_type == 3)
list->AddTrain(ft.from_string(tmp_result), 100); //Строку, которую считали, заполняем в базовый класс
}
else
break;
}

fin.seekg(0, ios::cur);
}



void Controller::PrintTrains() //Вывод на консоль
{
for (int i = 0; i < list->trains.size(); i++) //До конца массива
{
cout << list->trains[i]->to_string() << endl; //Построчный вывод
}
}



int Controller::FreightFromUkr()
{ //2 метод по работе с коллекцией
int count = 0;
for (int i = 0; i < list->trains.size(); i++)
```

```
{
string str = list->trains[i]->to_string(); //Берём в виде строки все параметры обьекта
stringstream result(str);
string number, start, finish, type, cargo = "";
char repairs;
int carts, weight;

result >> number >> repairs >> carts >> start >> finish >> type >> cargo >> weight; //Разбиваем строку по
переменным
static const std::regex r("\\D+");
if (cargo != "")
{ //Если переменная заполнилась (один из двух наследников)
if (regex_match(cargo, r) && start == "Ukraine")
{ //Если переменная заполнена словами(грузовой поезд) и отправляется из Украины
count++; //Увеличиваем счетчик
cout << list->trains[i]->to_string() << endl; //Выводим на экран
}
}
}
return count; //Возвращаем кол-во
}
```