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
Бульба С. С.

Виконавець: студент гр. КІТ-120в
Бабенко А. П.

Харків 2021

1. Вимоги

1.1 Розробник

- Бабенко Антон Павлович;
- Студент групи КІТ-120в;
- 20 травня 2021;

1.2 Загальне завдання

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

1.3 Індивідуальне завдання

01. Файл

- Поля базового класу:
  - Прихований файл (наприклад: так, ні)
  - Назва файлу (наприклад: “Лаба01”)
  - Розмір файлу, Кб (наприклад: 10, 123.56)
  - Доступи (https://www.tutorialspoint.com/unix/unix-file-permission.htm) (структура, що даватиме можливість вказати можливість читання, запису, виконання)
  - Формат файлу (один з переліку: txt, docx, pdf, mp3, avi, mp4, mkv, exe, bat, jar)
- Спадкоємець 1 - Відеофайл. Додаткові поля:
  - Роздільна здатність (https://animoto.com/blog/news/hd-video-creation-sharing) (один з переліку: 360, 480, 720, 1080)
  - Частота кадрів (https://uk.wikipedia.org/wiki/Частота_кадрів) (наприклад: 24, 72)
- Спадкоємець 2 - Файл зображення. Додаткові поля:
  - Розмір зображення (структура, що має значення кількості пікселей по ширині та по висоті)
  - Кількість точок на дюйм (https://en.wikipedia.org/wiki/Dots_per_inch) (наприклад: 200, 300)
- Методи для роботи з колекцією:
  1. Обрати з каталогу всі файли більше 50 кБ
  2. Відсортувати за назвою та обрати другий файл, що матиме всі доступи (rxw)
  3. Знайти зображення, що має найменшу кількість пікселів

1.4 Призначення та галузь застосування

Програма призначена для обробки вхідних даних, створення колекції файлів обробки та роботи з нею. Застосування програми призначено для людей, які працюють з базою даних файлів 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 файлів, дані введені користувачем з клавіатури.

Вихідні дані – файл з обробленою користувачем колекцією файлів, 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

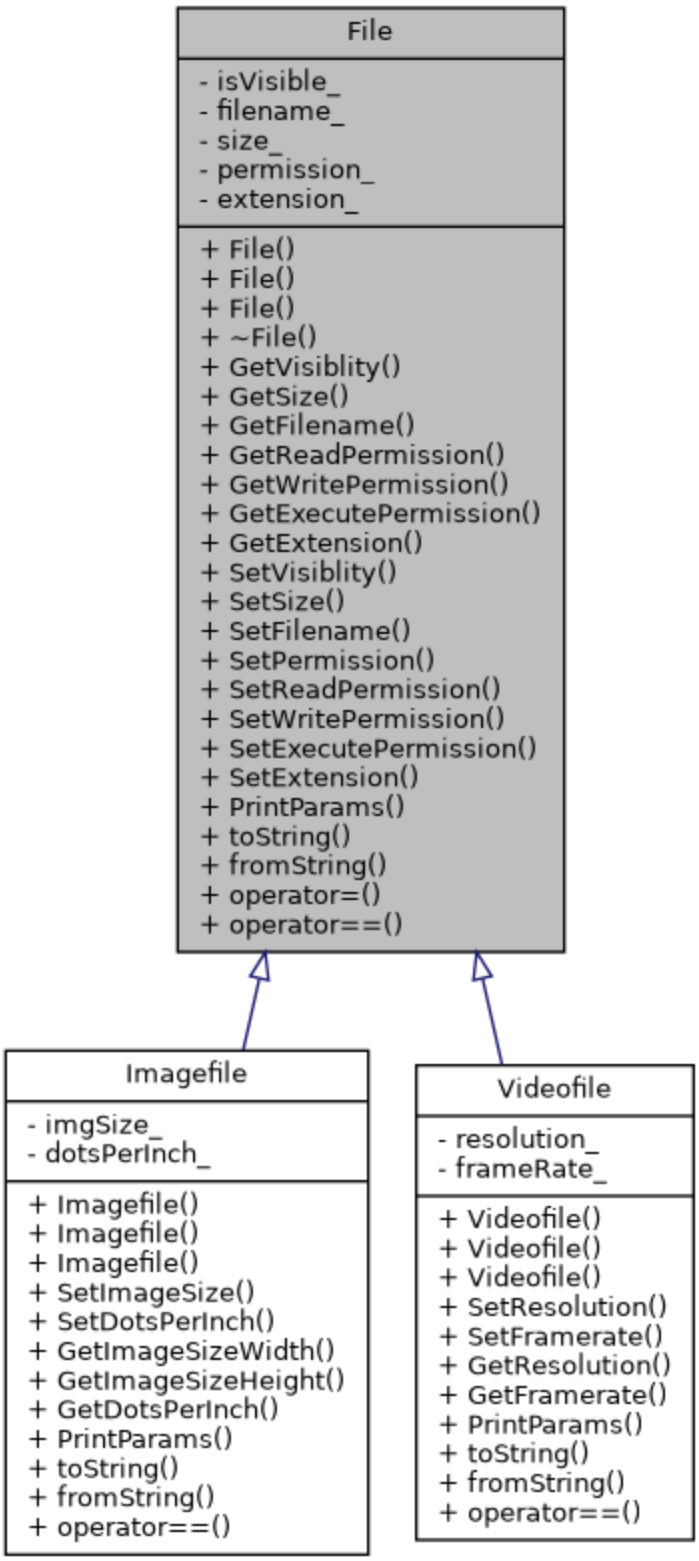

Рисунок 1.1 – uml-діаграмма успадкувань

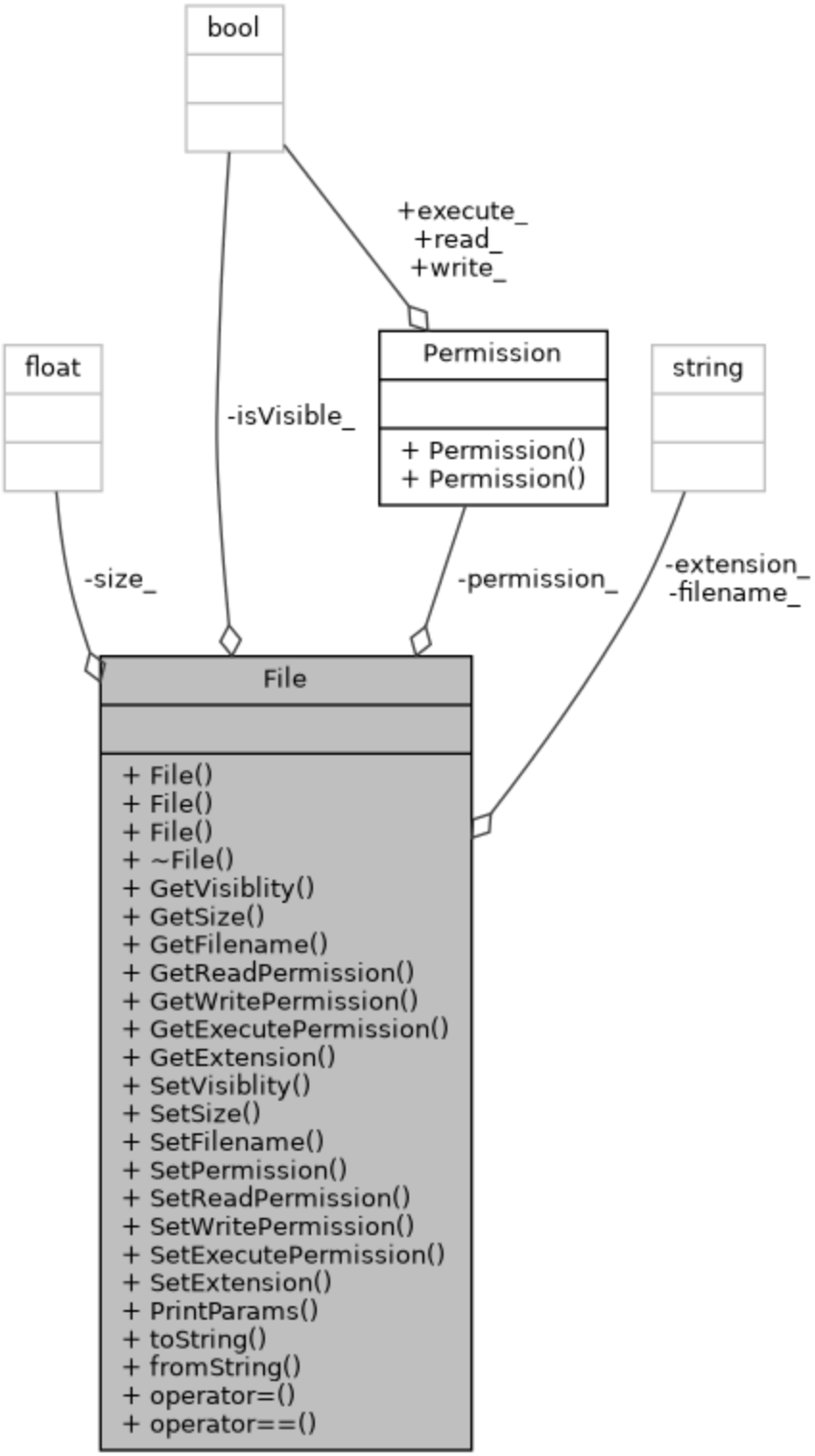

Рисунок 1.2 – uml-діаграмма зв’язків класу File

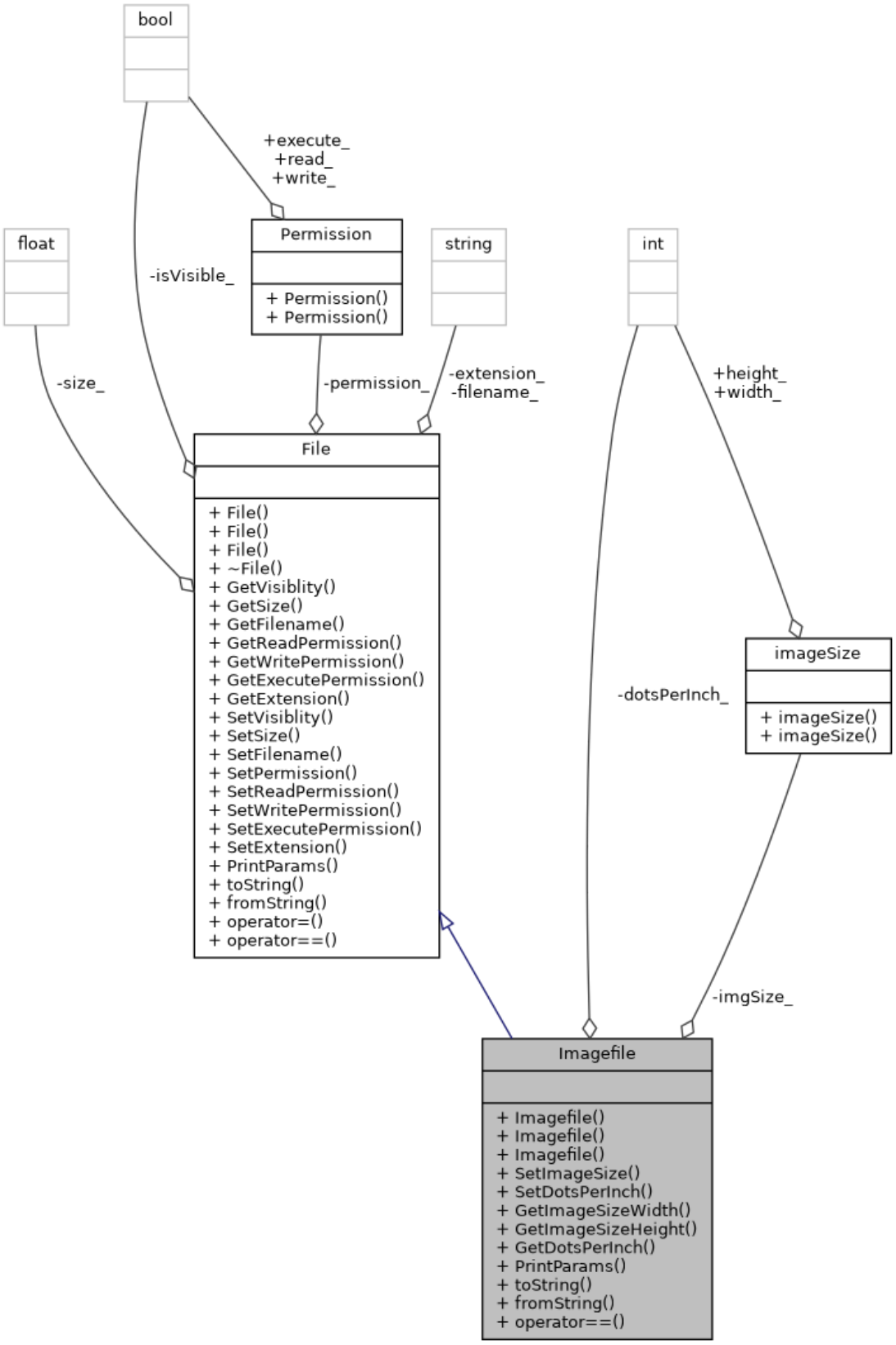

Рисунок 1.3 – uml-діаграмма зв’язків класу Imagefile

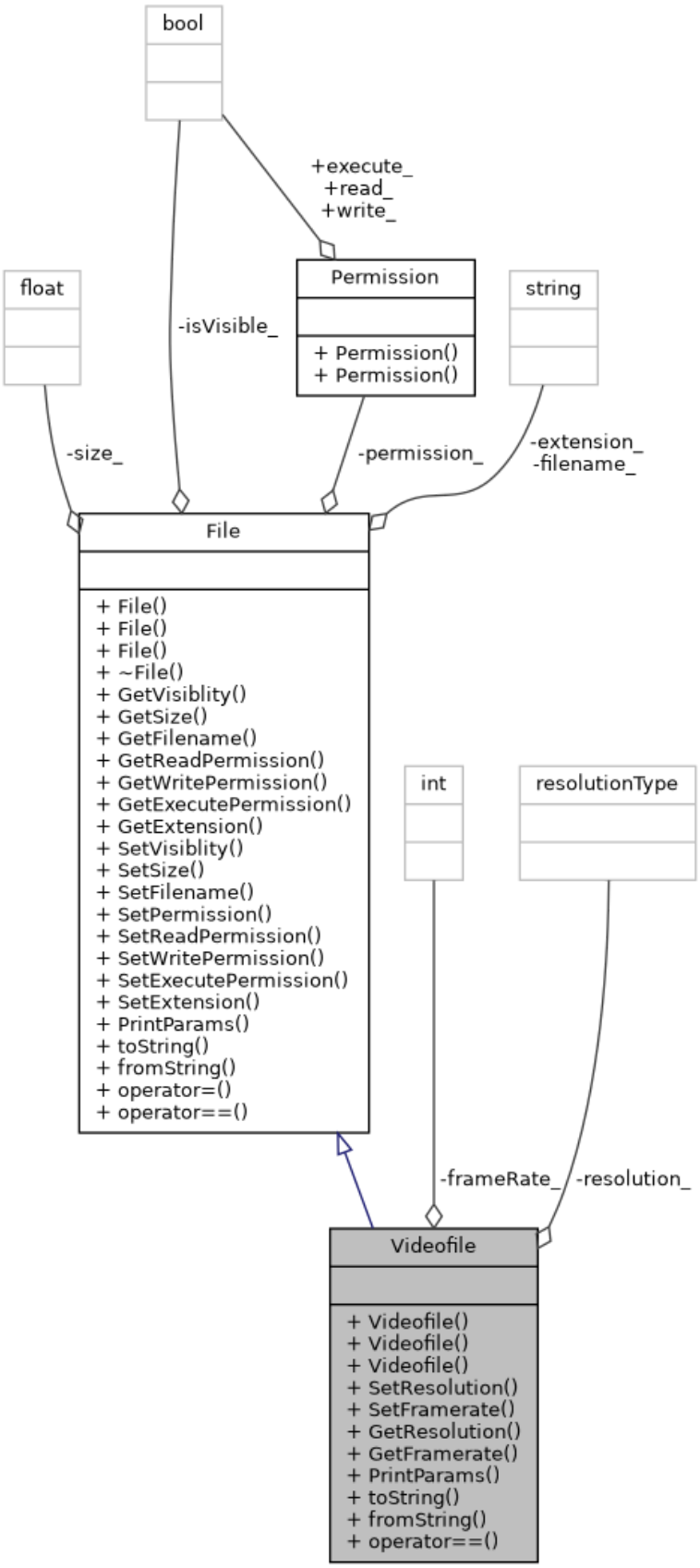

Рисунок 1.4 – uml-діаграмма зв’язків класу Videofile

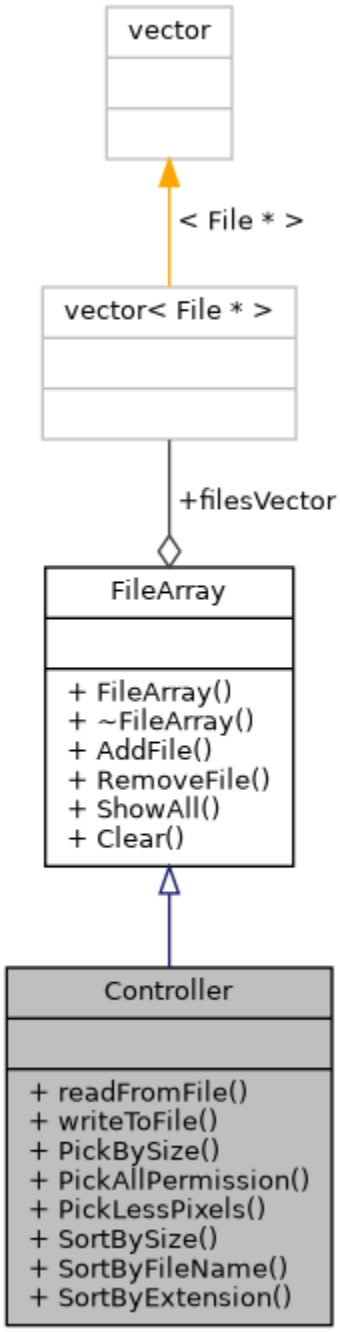

Рисунок 1.5 – uml-діаграмма зв’язків класу FileArray та Controller

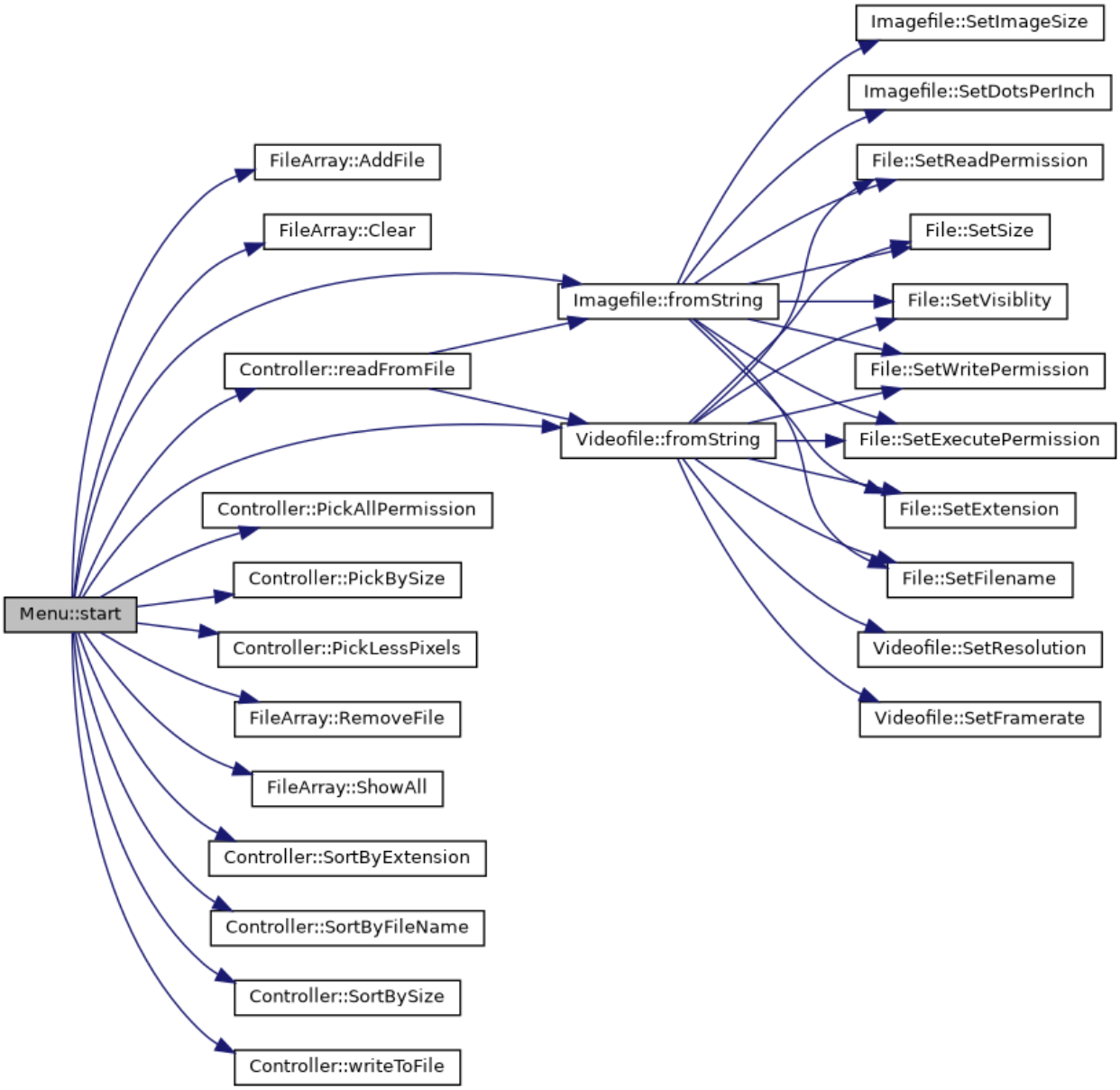

Рисунок 1.6 – граф виклику методів класу Menu

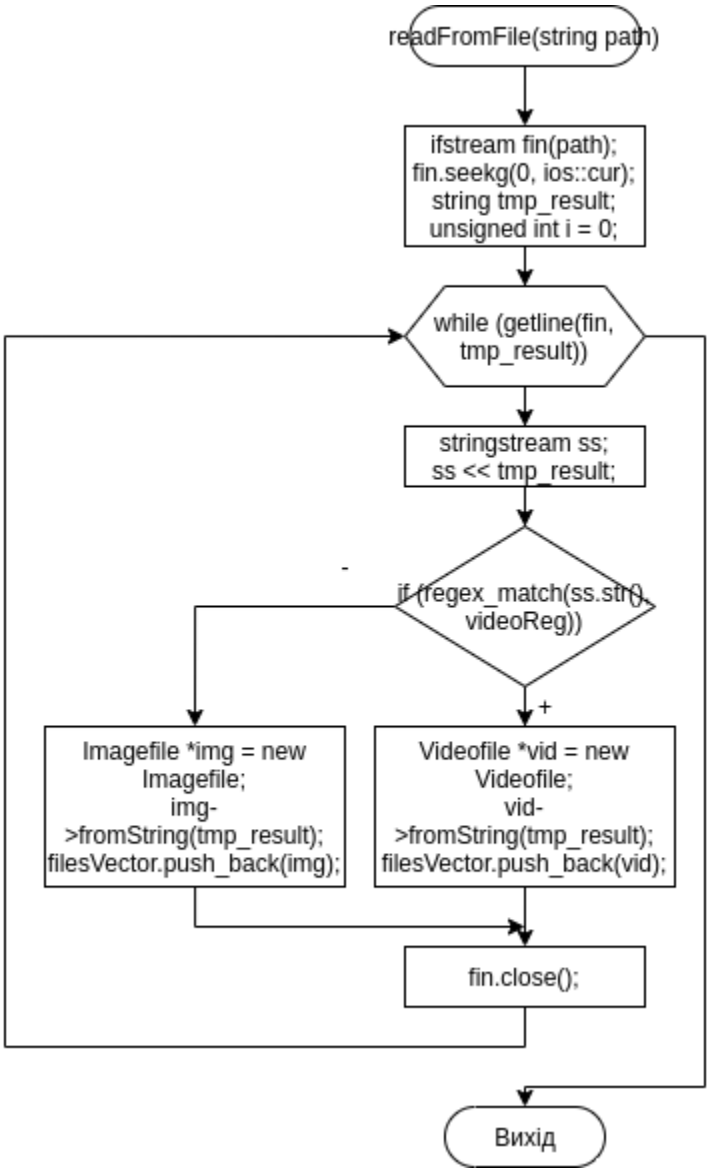

Рисунок 1.7 – блок-схема методу readFromFile

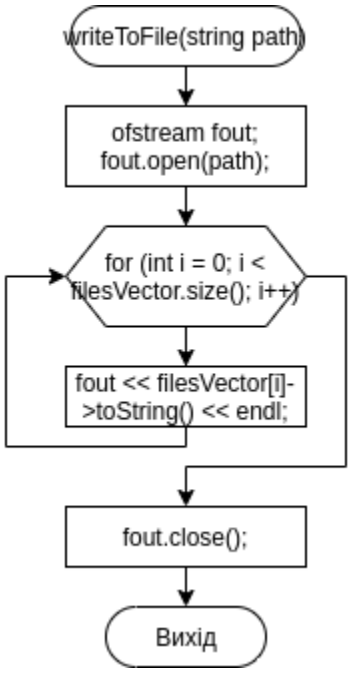

Рисунок 1.8 – блок-схема методу writeToFile

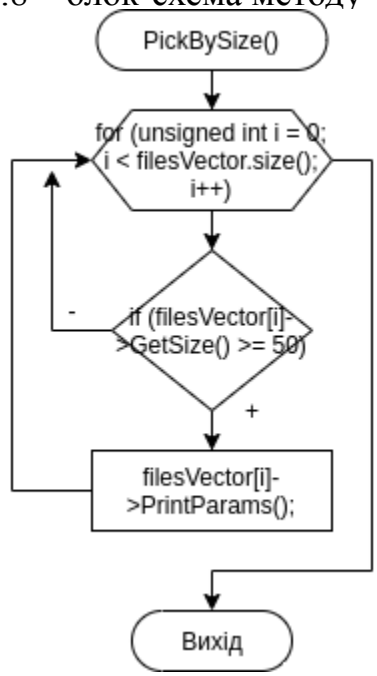

Рисунок 1.9 – блок-схема методу PickBySize

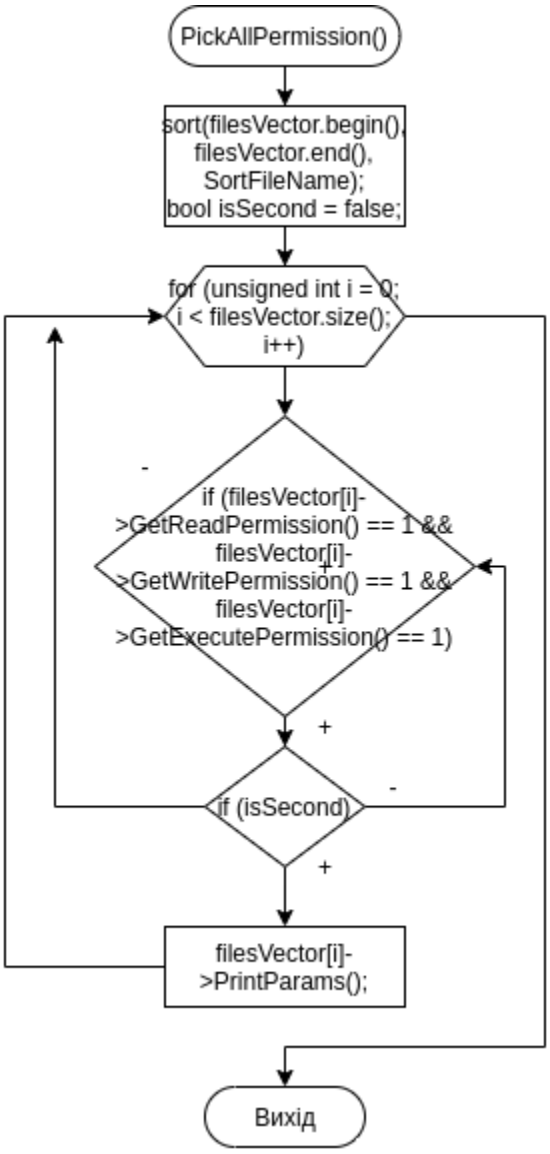

Рисунок 1.9 – блок-схема методу PickAllPermission

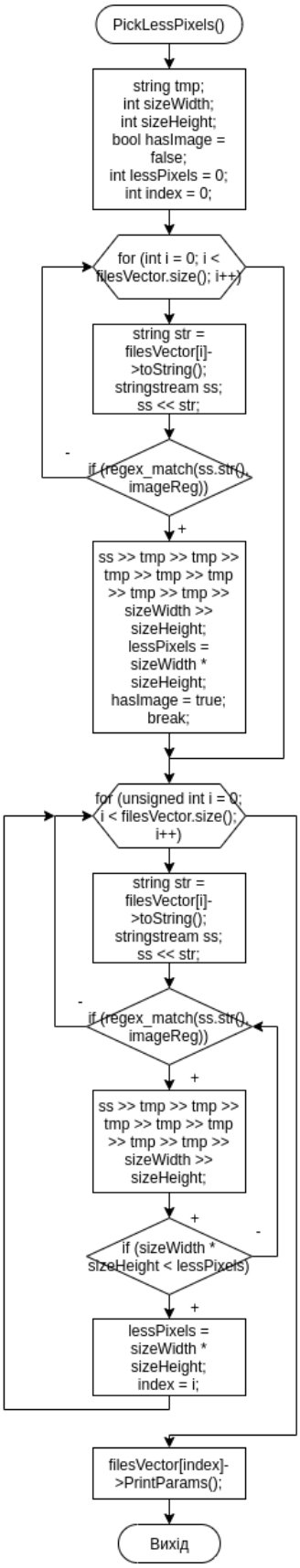

Рисунок 1.9 – блок-схема методу PickLessPixels

2.6 Перевірка програми за допомогою утиліти Valgrind

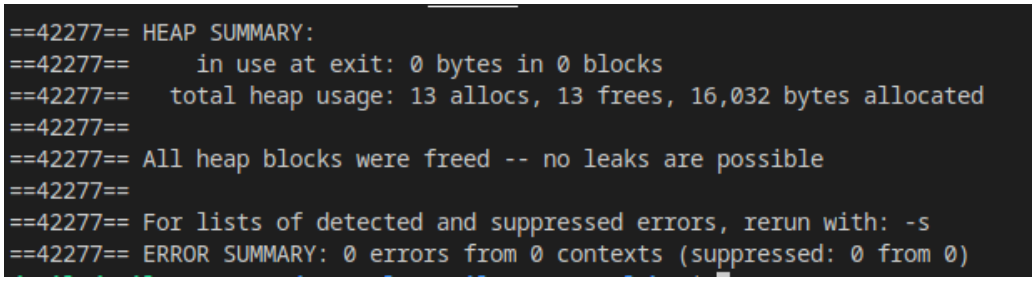

Рисунок 1.11 — утіліта Valgrind

3. Висновки

Я закріпив отримані знання з дисципліни “Програмування” шляхом виконання типового комплексного завдання. Моє індивідуальне завдання було таким: Розробити методи для роботи з колекцією файлів :

- Обрати з каталогу всі файли більші 50 кБ
- Відсортувати за назвою та обрати другий файл, що матиме всі доступи (rxw)
- Знайти зображення, що має найменшу кількість пікселів

Я виконав завдання та розробив запропоновані методи по роботі з колекцією. Ця програма призначена для обробки вхідних даних, створення колекції файлів обробки та роботи з нею. Продемонстрував роботу програми та коректність її виконання.

Додаток А

```
void Controller::readFromFile(string path)
{
 ifstream fin(path);
 fin.seekg(0, ios::cur);
 string tmp_result;

 if (!fin.is_open())
   cout << "Файл не может быть открыт!\n";
 else
 {
   unsigned int i = 0;
   while (getline(fin, tmp_result))
   {
     int fileType;
     const regex videoReg("^((true)|(false)) [\\w\\d]+
[\\d\\.\\d]+ ((true)|(false)) ((true)|(false)) ((true)|(false))
[\\w\\d]+ [\\d]+ [\\d]+$");
     const regex imageReg("^((true)|(false)) [\\w\\d]+
[\\d\\.\\d]+ ((true)|(false)) ((true)|(false)) ((true)|(false))
[\\w\\d]+ [\\d]+ [\\d]+ [\\d]+$");

     stringstream ss;
     ss << tmp_result;

     if (regex_match(ss.str(), videoReg))
     {
       Videofile *vid = new Videofile;
       vid->fromString(tmp_result);
       filesVector.push_back(vid);
     }
     else if (regex_match(ss.str(), imageReg))
     {
       Imagefile *img = new Imagefile;
       img->fromString(tmp_result);
       filesVector.push_back(img);
     }
   }
```

```
 }

 fin.close();
}

void Controller::writeToFile(string path)
{
 ofstream fout;
 fout.open(path);
 for (int i = 0; i < filesVector.size(); i++)
 {
   fout << filesVector[i]->toString() << endl;
 }
 fout.close();
}

void Controller::SortBySize()
{
 sort(filesVector.begin(), filesVector.end(), SortSize);
}

void Controller::SortByFileName()
{
 sort(filesVector.begin(), filesVector.end(), SortFileName);
}

void Controller::SortByExtension()
{
 sort(filesVector.begin(), filesVector.end(), SortExtension);
}

void Controller::PickBySize() const
{
 for (unsigned int i = 0; i < filesVector.size(); i++)
 {
   if (filesVector[i]->GetSize() >= 50)
   {
     filesVector[i]->PrintParams();
```

```
    }
  };
}

void Controller::PickAllPermission()
{
  sort(filesVector.begin(), filesVector.end(), SortFileName);
  bool isSecond = false;
  for (unsigned int i = 0; i < filesVector.size(); i++)
  {
    if (filesVector[i]->GetReadPermission() == 1 &&
        filesVector[i]->GetWritePermission() == 1 &&
        filesVector[i]->GetExecutePermission() == 1)
    {
      if (isSecond)
      {
        filesVector[i]->PrintParams();
      }
      isSecond = true;
    }
  };
}

void Controller::PickLessPixels()
{
  const regex imageReg("^((true)|(false)) [\\w\\d]+ [\\d\\.\\d]+
((true)|(false)) ((true)|(false)) ((true)|(false)) [\\w\\d]+
[\\d]+ [\\d]+ [\\d]+$");

  string tmp;
  int sizeWidth;
  int sizeHeight;
  bool hasImage = false;
  int lessPixels = 0;
  int index = 0;

  for (int i = 0; i < filesVector.size(); i++)
  {
```

```
  string str = filesVector[i]->toString();
  stringstream ss;
  ss << str;

  if (regex_match(ss.str(), imageReg))
  {
    ss >> tmp >> tmp >> tmp >> tmp >> tmp >> tmp >> tmp >>
        sizeWidth >> sizeHeight;

    lessPixels = sizeWidth * sizeHeight;
    hasImage = true;
    break;
  }
 }

 for (unsigned int i = 0; i < filesVector.size(); i++)
 {
  string str = filesVector[i]->toString();
  stringstream ss;
  ss << str;

  if (regex_match(ss.str(), imageReg))
  {
    ss >> tmp >> tmp >> tmp >> tmp >> tmp >> tmp >> tmp >>
        sizeWidth >> sizeHeight;

    if (sizeWidth * sizeHeight < lessPixels)
    {
      lessPixels = sizeWidth * sizeHeight;
      index = i;
    }
  }
 }
 filesVector[index]->PrintParams();
};
```