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
Олексієнко Микита

Харків 2021

1.Вимоги

1.1 Розробник

Олексієнко Микита Віталійович

студент групи КІТ-120В

10.05.2021

1.2 Загальне завдання

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

1.3 Індивідуальне завдання

У місті відкрилась сучасна бібліотека. Працівники збирають інформацію про книги, які є в бібліотеці. Для цього протрібно розробити методи для роботи з колекцією:

- Знайти всі книги видавництва “Ранок”

- Знайти детективи, що мають електронну версію

- Знайти книгу з найбільшою кількістю сторінок

1.4 Призначення та галузь застосування

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

Вихідні дані — файл з обробленою користувачем колекцією книг, 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%ктахівВ0%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

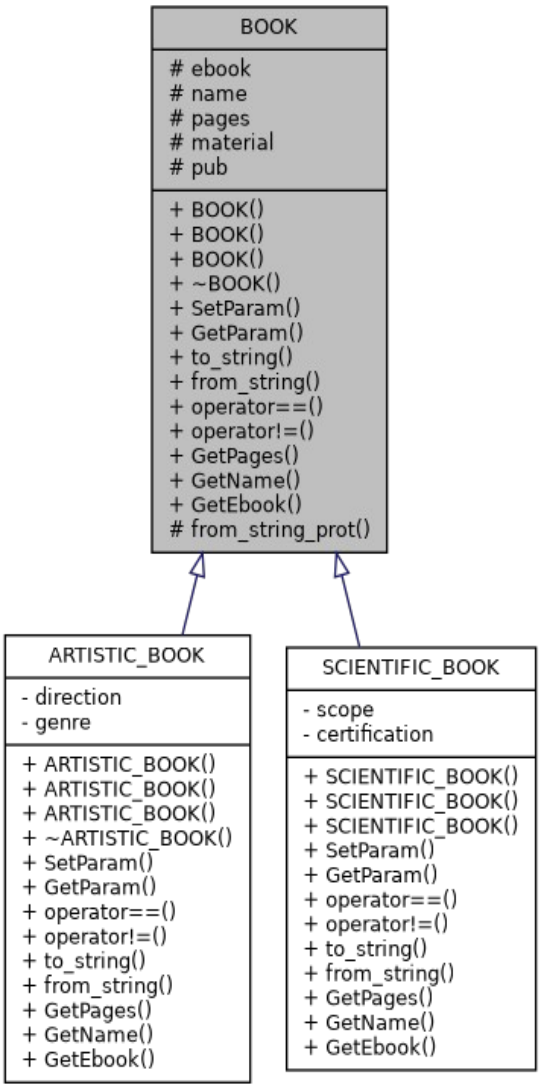

Рисунок 1.1 — *uml-діаграмма успадкувань*

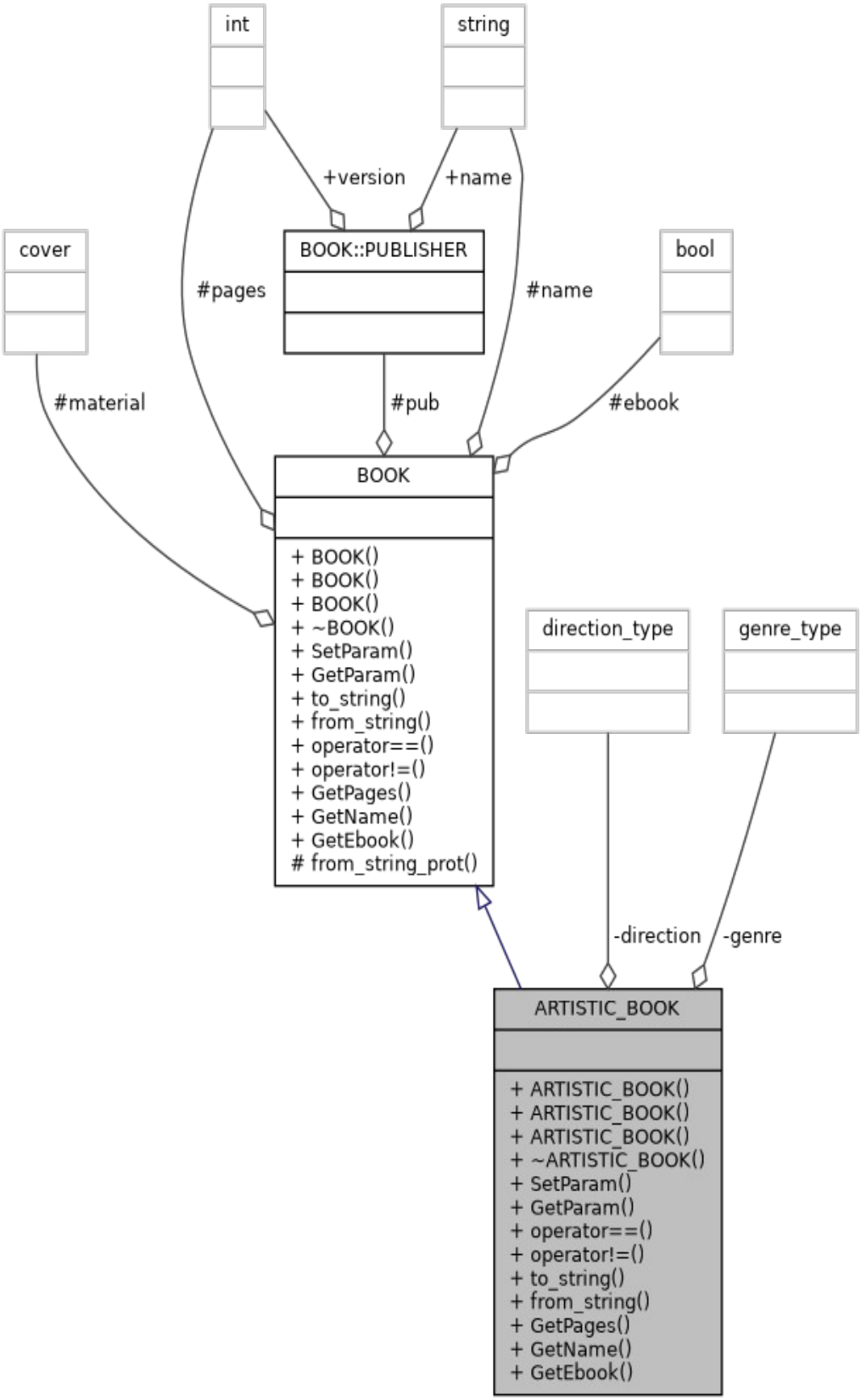

Рисунок 1.2 — uml-діаграмма зв’язків класу ARTISTIC_BOOK

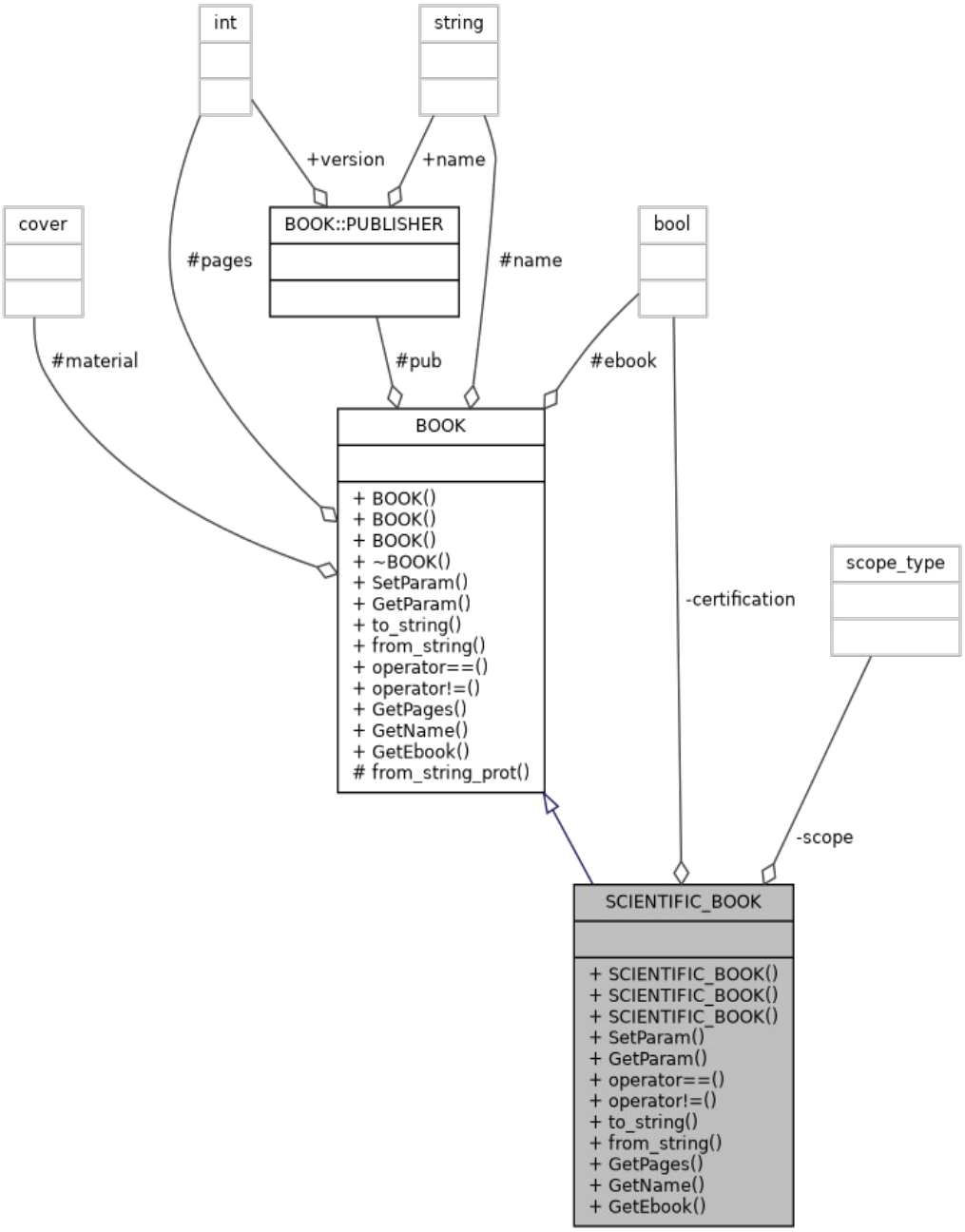

Рисунок 1.3 — uml-діаграмма зв’язків класу SCIENTIFIC_BOOK

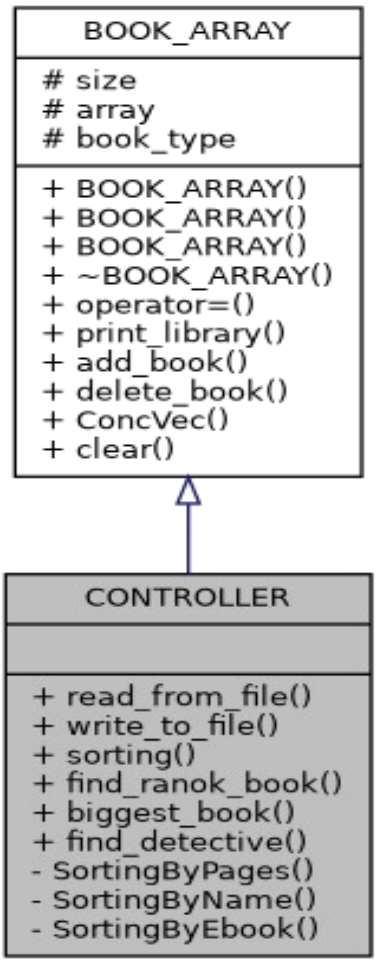

Рисунок 1.4 — uml-діаграмма зв’язків класу BOOK_ARRAY та CONTROLLER

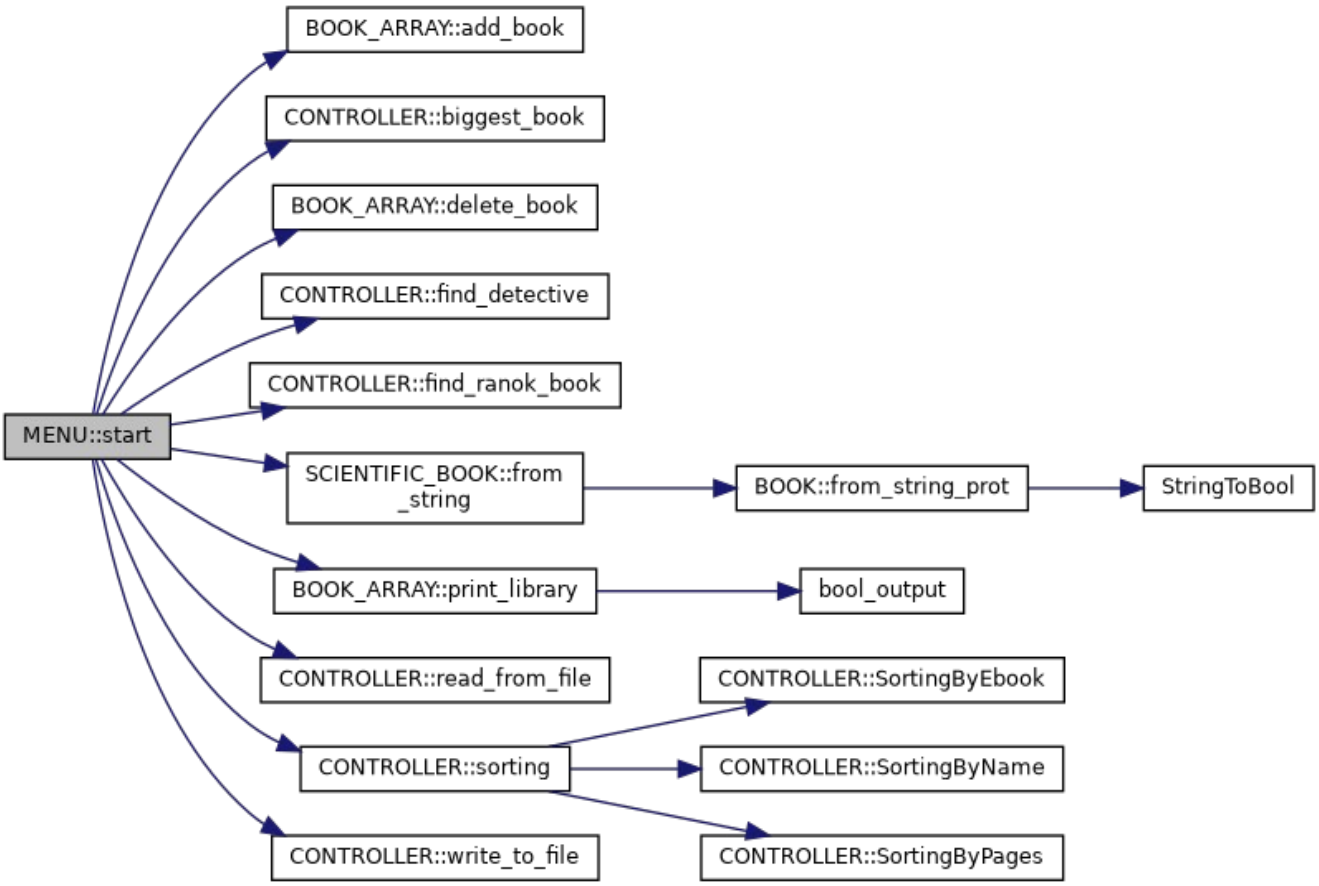

Рисунок 1.6 — граф виклику функцій класу MENU

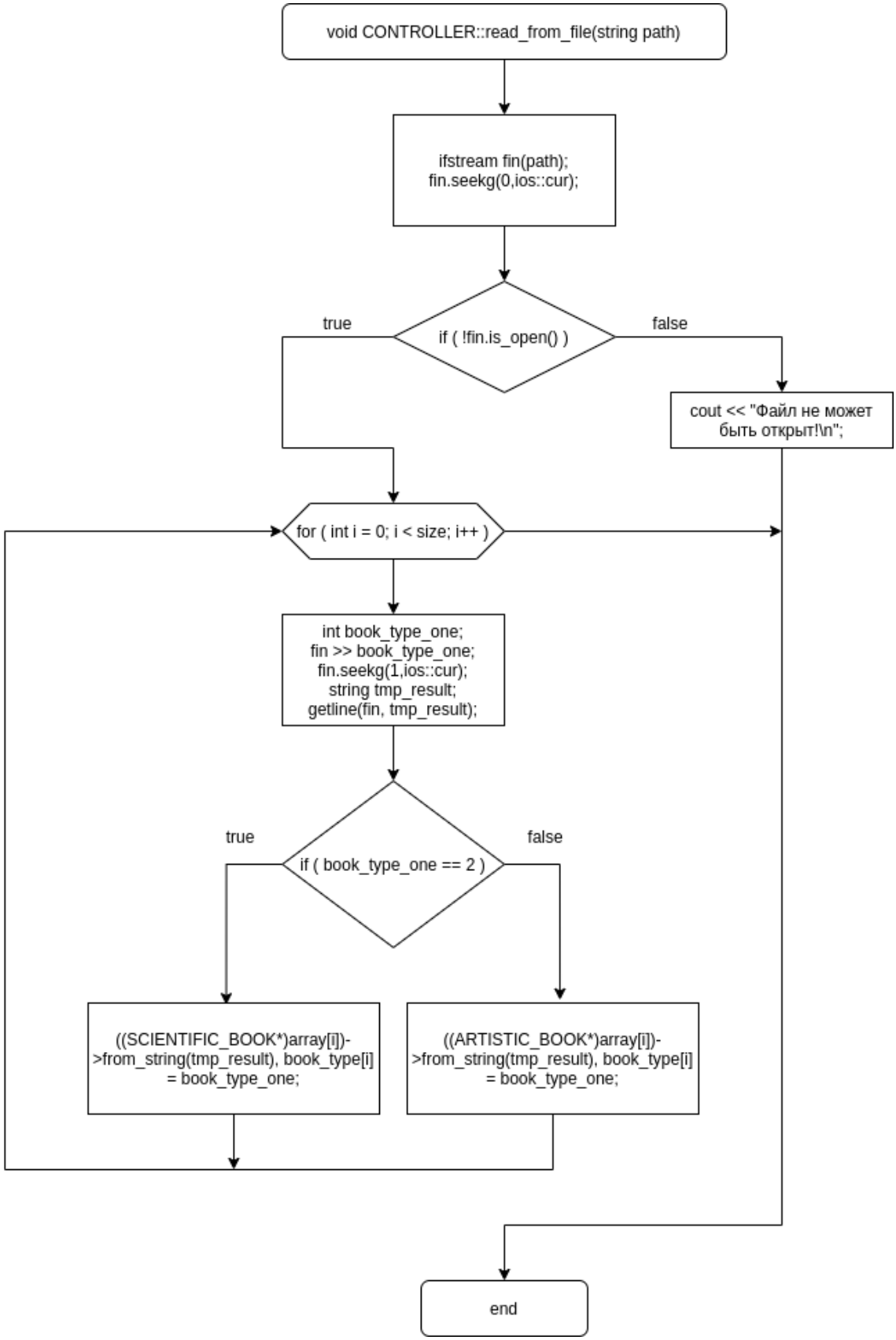

Рисунок 1.7 — метод READ_FROM_FILE

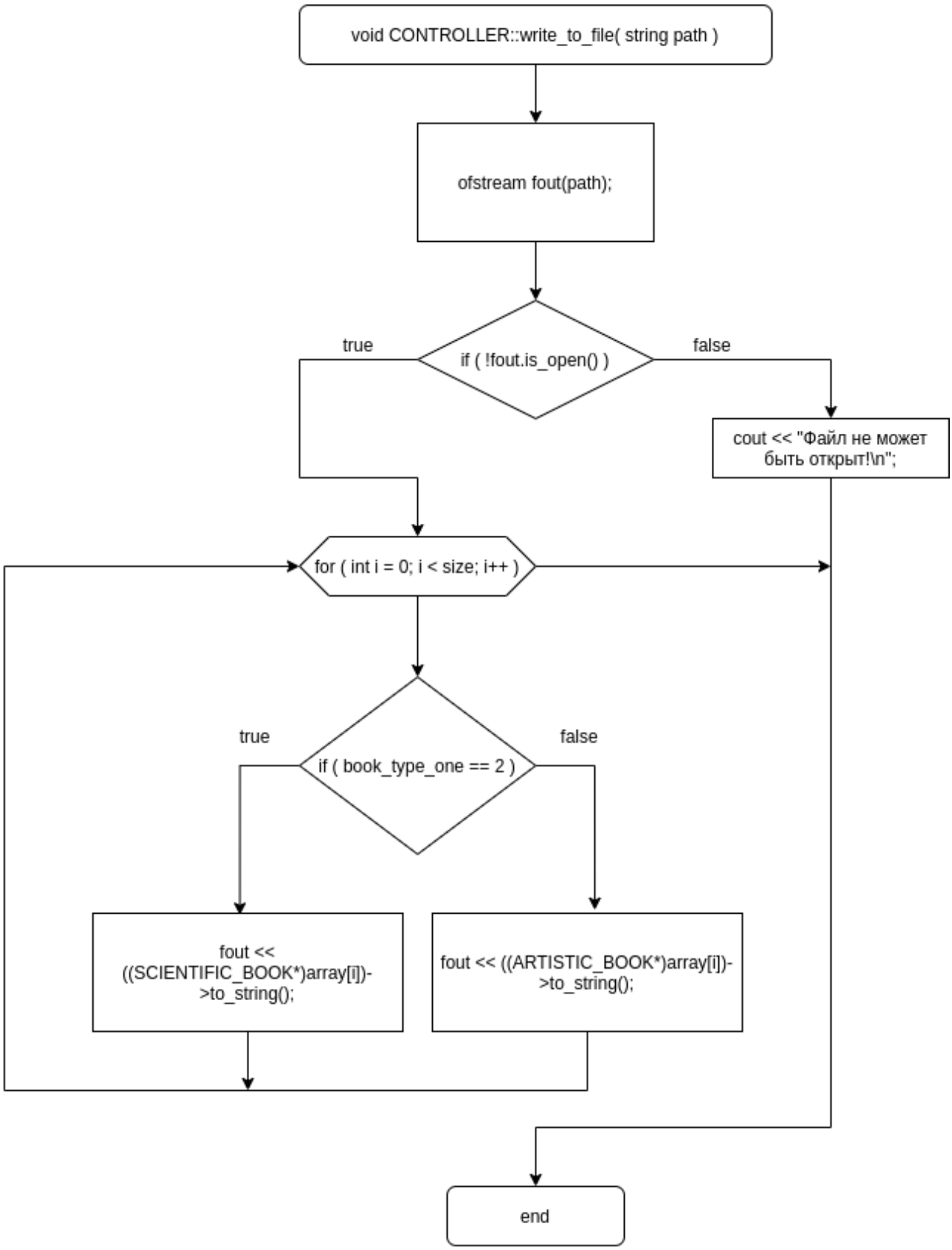

Рисунок 1.8 — метод WRITE_TO_FILE

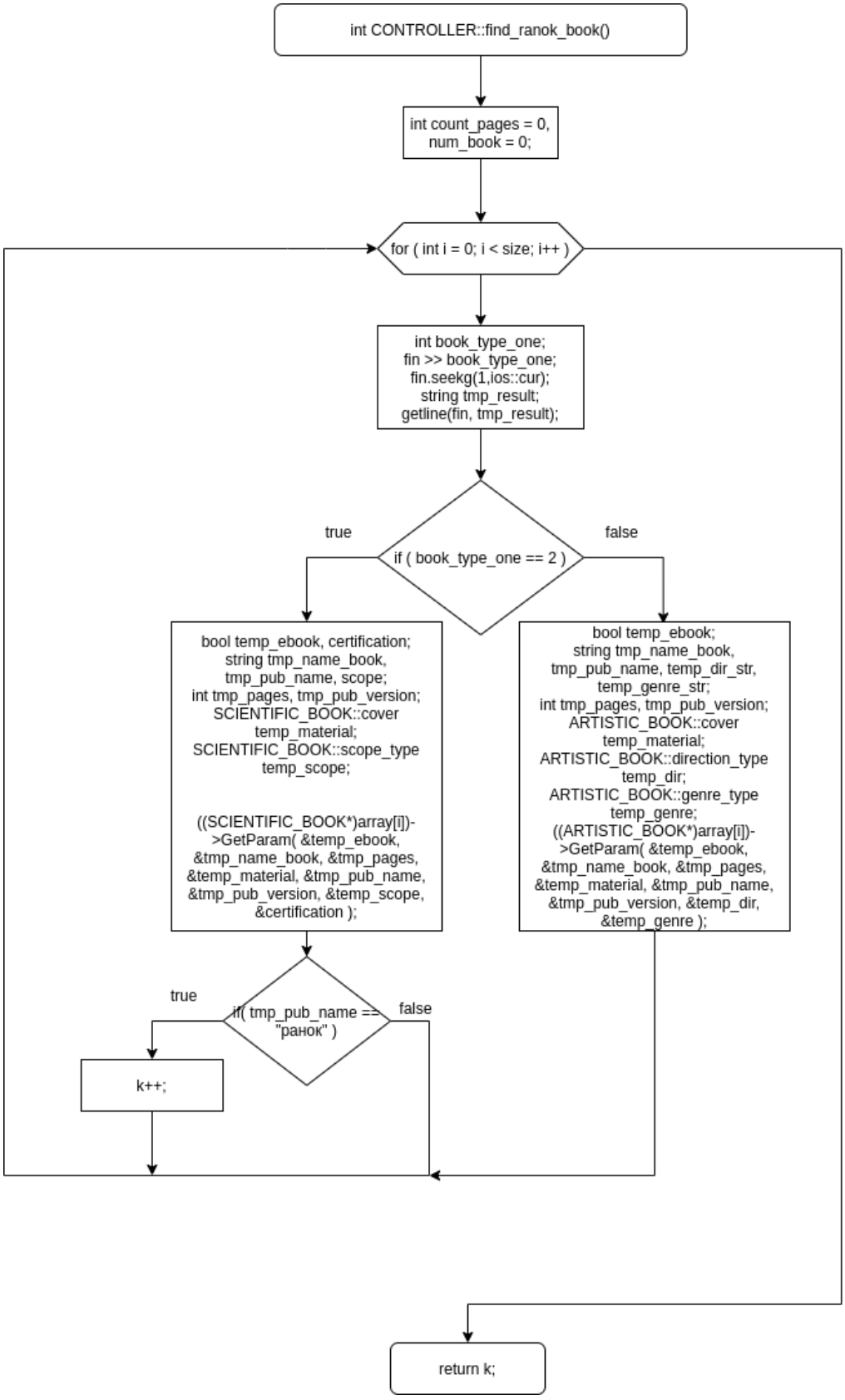

Рисунок 1.9— метод FIND_RANOK_BOOK

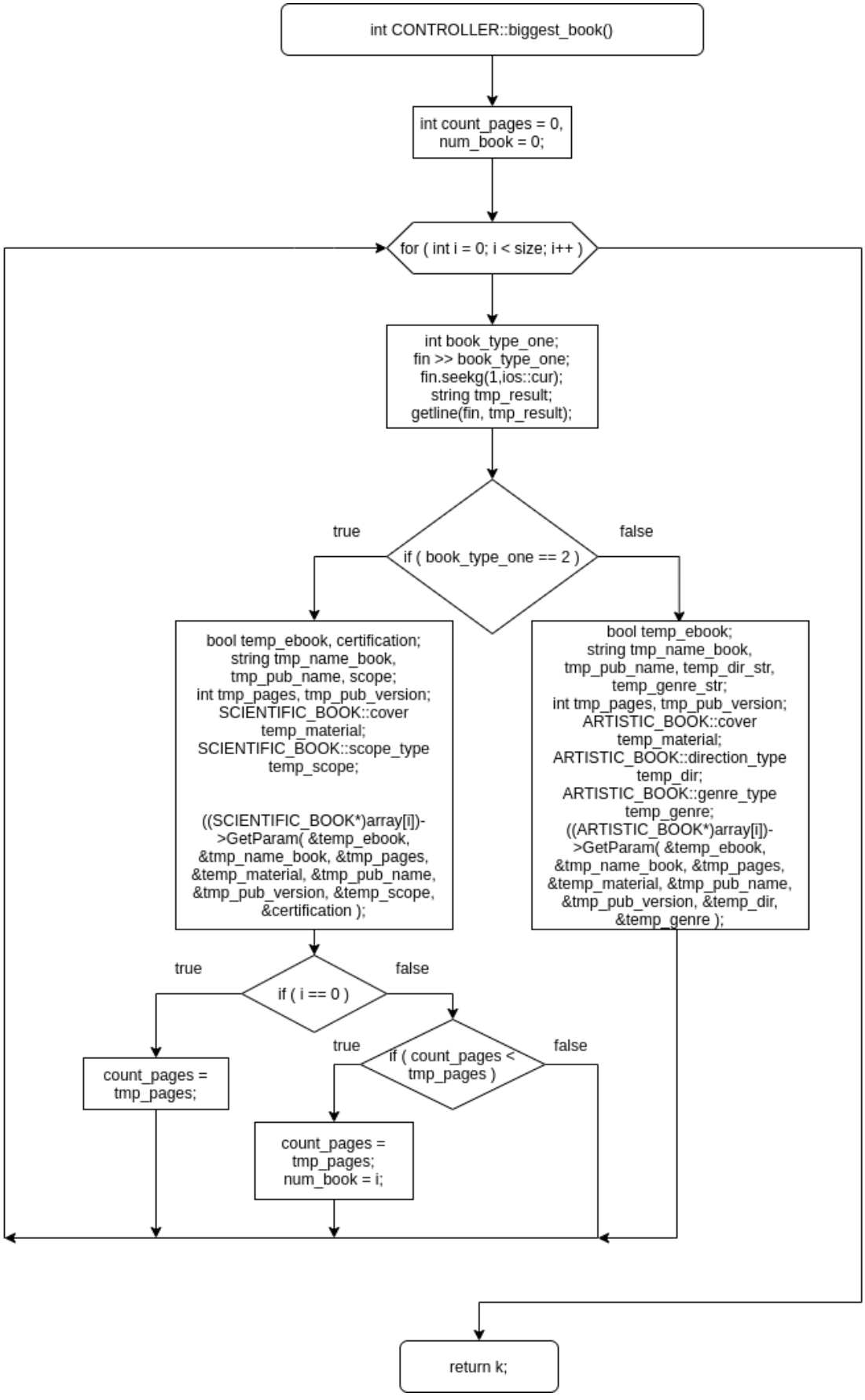

Рисунок 2.0 — метод BIGGEST_BOOK

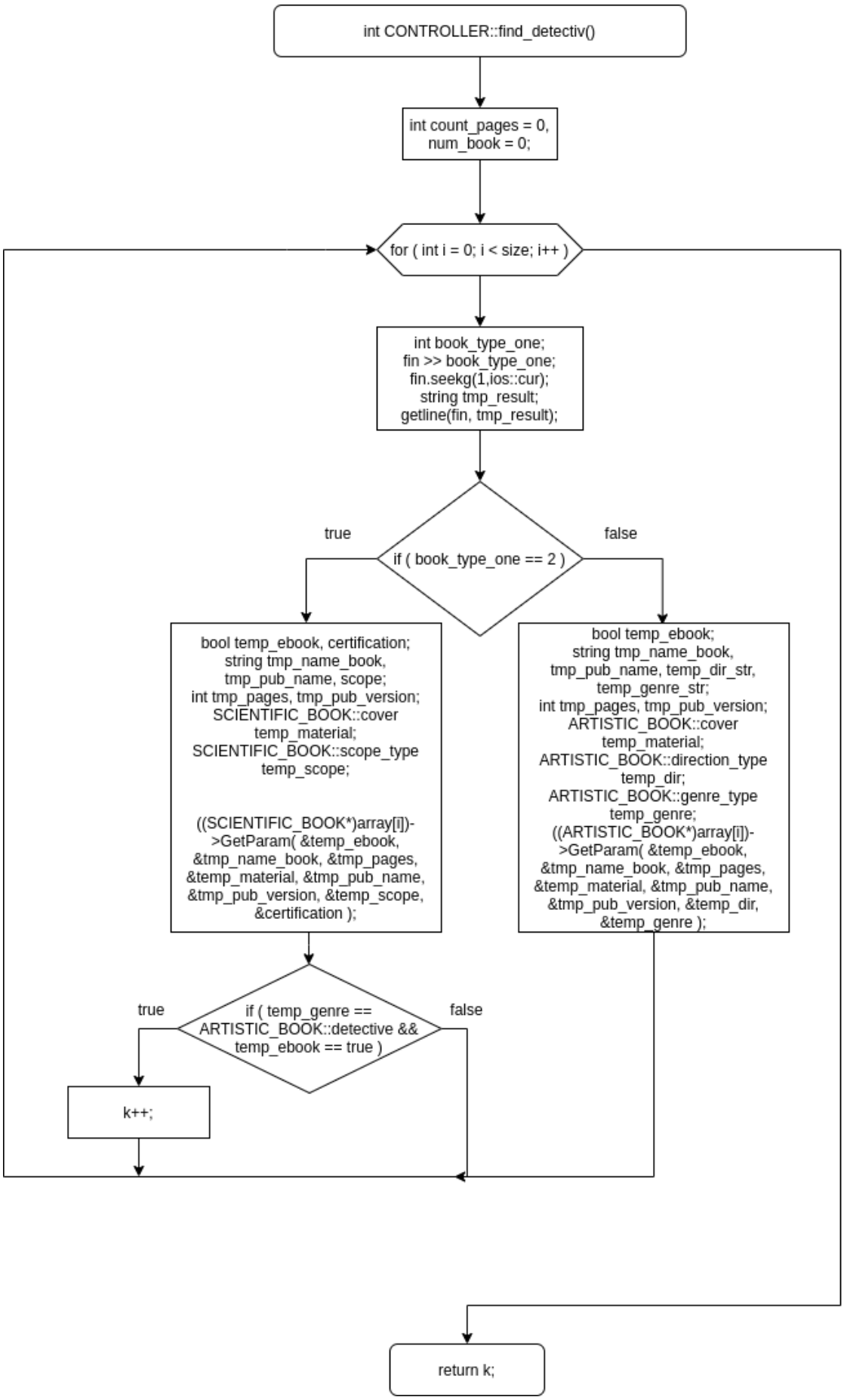

Рисунок 2.1 — метод FIND_DETECTIVE

2.6 Перевірка програми за допомогою утіліти Valgrind

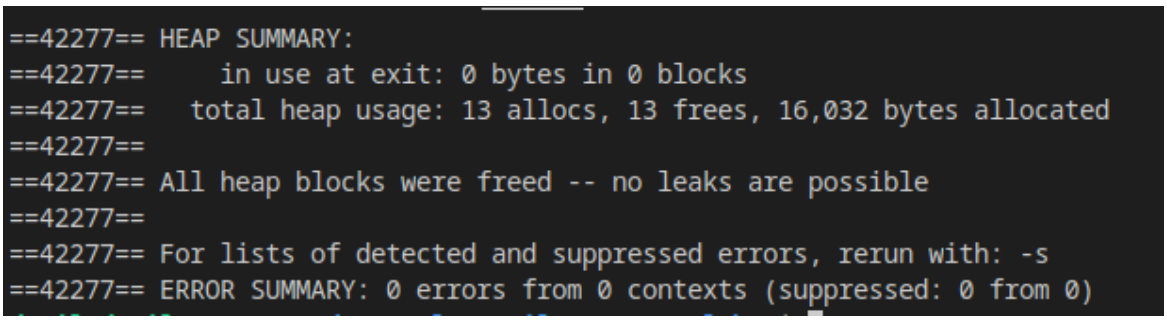

Рисунок 2.2 — утіліта Valgrind

3. Висновки

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

Моє індивідуальне завдання було таким:

У місті відкрилась сучасна бібліотека. Працівники збирають інформацію про книги, які є в бібліотеці. Для цього протрібно розробити методи для роботи з колекцією:
- Знайти всі книги видавництва “Ранок”

- Знайти детективи, що мають електронну версію

- Знайти книгу з найбільшою кількістю сторінок

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

Додаток А

```
void CONTROLLER::read_from_file(string path)
 {
 ifstream fin(path); // откроем поток

 fin.seekg(0,ios::cur); //переместил курсор на начало файла

 cout << "┌──────────────────────────────────────┐\n";
 cout << "| Читаются файлы из " << __FUNCTION__ <<" |\n";
 cout << "└──────────────────────────────────────┘\n";


 if ( !fin.is_open() ) // если файл не открыт
 cout << "Файл не может быть открыт!\n"; // сообщить об этом
 else
 {
 for ( int i = 0; i < size; i++ )
 {
 int book_type_one;

 fin >> book_type_one; // читаем 1 значение в файле. если 1 - это первый
 наследник, если 2 - второй
 fin.seekg(1,ios::cur);

 string tmp_result;
 getline(fin, tmp_result);

 if ( book_type_one == 2 ) // занимаем нужные поля в зависимости от типа
 наследника
 ((SCIENTIFIC_BOOK*)array[i])->from_string(tmp_result), book_type[i] =
 book_type_one;

 else if ( book_type_one == 1 )
 ((ARTISTIC_BOOK*)array[i])->from_string(tmp_result), book_type[i] =
 book_type_one;

 else
 exit(1);
 }
 }

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

 void CONTROLLER::write_to_file( string path )
 {
```

```
ofstream fout(path); // открываем поток

cout << "┌──────────────────────────────────────────┐\n";
cout << "| Записываются файлы в " << __FUNCTION__ <<" |\n";
cout << "└──────────────────────────────────────────┘\n";

if ( !fout.is_open() ) // если файл не открыт
cout << "Файл не может быть открыт!\n"; // сообщить об этом
else
{
for ( int i = 0; i < size; i++ )
{
if ( book_type[i] == 2 ) // занимаем нужные поля в зависимости от типа
наследника
fout << ((SCIENTIFIC_BOOK*)array[i])->to_string();

else if ( book_type[i] == 1 )
fout << ((ARTISTIC_BOOK*)array[i])->to_string();
if ( i != size - 1 )
fout << endl;
}
}
}

//-------------------------------------------------------------

void CONTROLLER::sorting() // функция сортировки
{
BOOK** temp_book = new BOOK* [size]; // создаем временную книгу

for ( int i = 0 ; i < size; i++) // заполняем ее
{
temp_book[i] = array[i];
}

array.clear(); // чистим оригинальную книгу
array.reserve(size);

for ( int i = 0; i < size; i++) // заполняем с временной книги как нужно
{
array.push_back(temp_book[i]);
}
cout << "┌──────────────────────────┐" << endl;
cout << "| Выберете вариант сортировки: |" << endl;
cout << "└──────────────────────────┘" << endl;
cout << "- Эл.версия [1]" << endl;
cout << "- Название [2]" << endl;
cout << "- Страницы [3]" << endl << endl << "- ";

short int var_sort;
cin >> var_sort;
```

```
// выбираем вариант сортировки и переходим в нужный метод

switch( var_sort )
{
case 1:
SortingByEbook();
break;

case 2:
SortingByName();
break;

case 3:
SortingByPages();
break;
}

}

struct sort_pages //функтор сортировки
{
bool operator() ( BOOK* i, BOOK* j )
{
return ( i->GetPages() < j->GetPages() );
}
} sort_pages;

struct sort_name //функтор сортировки
{
bool operator() ( BOOK* i, BOOK* j )
{
return ( i->GetName() < j->GetName() );
}
} sort_name;

struct sort_ebook //функтор сортировки
{
bool operator() ( BOOK* i, BOOK* j )
{
return ( i->GetEbook() < j->GetEbook() );
}
} sort_ebook;

void CONTROLLER::SortingByPages()
{
sort ( array.begin(), array.end(), sort_pages ); //сортировка с какого по
какой элемент
}

void CONTROLLER::SortingByName()
```

```
{
sort ( array.begin(), array.end(), sort_name ); //сортировка с какого по
какой элемент
}

void CONTROLLER::SortingByEbook()
{
sort ( array.begin(), array.end(), sort_ebook ); //сортировка с какого по
какой элемент
}

//-------------------------------------------------------------

// МЕТОД 1 - найти все книги издательства ранок

int CONTROLLER::find_ranok_book()
{
int k = 0;
for ( int i = 0; i < size; i++ )
{
// в зависимости от типа наследника вызываем нужные геттеры и ищем книги
if ( book_type[i] == 2)
{
bool temp_ebook, certification;
string tmp_name_book, tmp_pub_name, scope;
int tmp_pages, tmp_pub_version;
SCIENTIFIC_BOOK::cover temp_material;
SCIENTIFIC_BOOK::scope_type temp_scope;
((SCIENTIFIC_BOOK*)array[i])->GetParam( &temp_ebook, &tmp_name_book,
&tmp_pages, &temp_material, &tmp_pub_name,
&tmp_pub_version, &temp_scope, &certification );

if( tmp_pub_name == "ранок" ) k++;
}

else if ( book_type[i] == 1 )
{
bool temp_ebook;
string tmp_name_book, tmp_pub_name, temp_dir_str, temp_genre_str;
int tmp_pages, tmp_pub_version;
ARTISTIC_BOOK::cover temp_material;
ARTISTIC_BOOK::direction_type temp_dir;
ARTISTIC_BOOK::genre_type temp_genre;

((ARTISTIC_BOOK*)array[i])->GetParam( &temp_ebook, &tmp_name_book,
&tmp_pages, &temp_material, &tmp_pub_name,
&tmp_pub_version, &temp_dir, &temp_genre );

if( tmp_pub_name == "ранок" ) k++;
}
}
```

```
return k;
}

// МЕТОД 2 - найти самую большую по страницам книгу

int CONTROLLER::biggest_book()
{
int count_pages = 0, num_book = 0;

for ( int i = 0; i < size; i++ )
{
// в зависимости от типа наследника вызываем нужные геттеры и ищем самую
большую книгу
if ( book_type[i] == 2)
{
bool temp_ebook, certification;
string tmp_name_book, tmp_pub_name, scope;
int tmp_pages, tmp_pub_version;
SCIENTIFIC_BOOK::cover temp_material;
SCIENTIFIC_BOOK::scope_type temp_scope;
((SCIENTIFIC_BOOK*)array[i])->GetParam( &temp_ebook, &tmp_name_book,
&tmp_pages, &temp_material, &tmp_pub_name,
&tmp_pub_version, &temp_scope, &certification );

if ( i == 0)
count_pages = tmp_pages;
else
{
if ( count_pages < tmp_pages )
{
count_pages = tmp_pages;
num_book = i;
}
}
}

else if ( book_type[i] == 1 )
{
bool temp_ebook;
string tmp_name_book, tmp_pub_name, temp_dir_str, temp_genre_str;
int tmp_pages, tmp_pub_version;
ARTISTIC_BOOK::cover temp_material;
ARTISTIC_BOOK::direction_type temp_dir;
ARTISTIC_BOOK::genre_type temp_genre;

((ARTISTIC_BOOK*)array[i])->GetParam( &temp_ebook, &tmp_name_book,
&tmp_pages, &temp_material, &tmp_pub_name,
&tmp_pub_version, &temp_dir, &temp_genre );

if ( i == 0)
```

```
count_pages = tmp_pages;
else
{
if ( count_pages < tmp_pages )
{
count_pages = tmp_pages;
num_book = i;
}
}
}
}

return num_book + 1;
}

// МЕТОД 3 - найти детективы с эл.версией

int CONTROLLER::find_detective()
{
int count = 0;

for ( int i = 0; i < size; i++ )
{
// так как детективы это 1 тип наследника, 2 тип мы даже не будем
рассматривать// в зависимости от типа наследника вызываем нужные геттеры и
ищем книги
if ( book_type[i] == 1 )
{
bool temp_ebook;
string tmp_name_book, tmp_pub_name, temp_dir_str, temp_genre_str;
int tmp_pages, tmp_pub_version;
ARTISTIC_BOOK::cover temp_material;
ARTISTIC_BOOK::direction_type temp_dir;
ARTISTIC_BOOK::genre_type temp_genre;

((ARTISTIC_BOOK*)array[i])->GetParam( &temp_ebook, &tmp_name_book,
&tmp_pages, &temp_material, &tmp_pub_name,
&tmp_pub_version, &temp_dir, &temp_genre );

if ( temp_genre == ARTISTIC_BOOK::detective && temp_ebook == true ) count +
+;
}
else if ( book_type[i] == 1 ) continue;
}

return count;
}
```