

масив об'єктів базового класу та має в собі методи додавання, видалення елемента, отримання елемента по індексу (або ідентифікатору), вивід усіх елементів на екран. Рекомендовані сигнатури методів: - додавання:

```
void CList::addPhone(Phone& phone);
```

- видалення:

```
void CList::removePhone(int index);
```

- отримання по індексу:

```
CPhone& CList::getPhone(int index);
```

- виведення усіх елементів (при цьому цей метод повинен викликати метод CList::getPhone(int index), щоб не було дублювання коду):

```
void CList::showAll();
```

• Ви

• Р

•

•

•

Індивідуальне завдання.

• С

Обрати прикладну галузь за варіантом відповідно до номера у журналі групи.

• Опис роботи

Програма працює за тим же принципом, що і колишні, однак використовує ООП, полегшуючи роботу.

• Функціональне призначення

Дана програма може бути використана для маніпуляцій з нашим динамічним масивом, який має в собі класи. Таким чином ми отримуємо великий функціонал на увазі більше кількості можливих методів для роботи з нашим масивом.

• Опис логічної структури

Данна програма працює за принципом створення класу, в якому зберігається динамічний масив іншого класу, а також за допомогою методів, які контролюється цей масив.

Функція main містить в собі виклик функцій, а також корисних властивостей для функцій

class List - клас, який зберігає в собі динамічний масив іншого класу, а також які всередині себе має всі методи роботи з даним класом.

class Agency - базовий клас.

void List::expand() - метод, який використовується для розширення обсягу нашого динамічного масиву

void List::add_ag - метод, який використовується для розширення нашого

масиву, а точніше до
рахунок спеціального

Agency& List::get_
індексом.

void List::remove_

int List::get_oldest_
в масиві (індекс).

зого елемента за

нт масиву за

агенство з масиву.

старший елемент

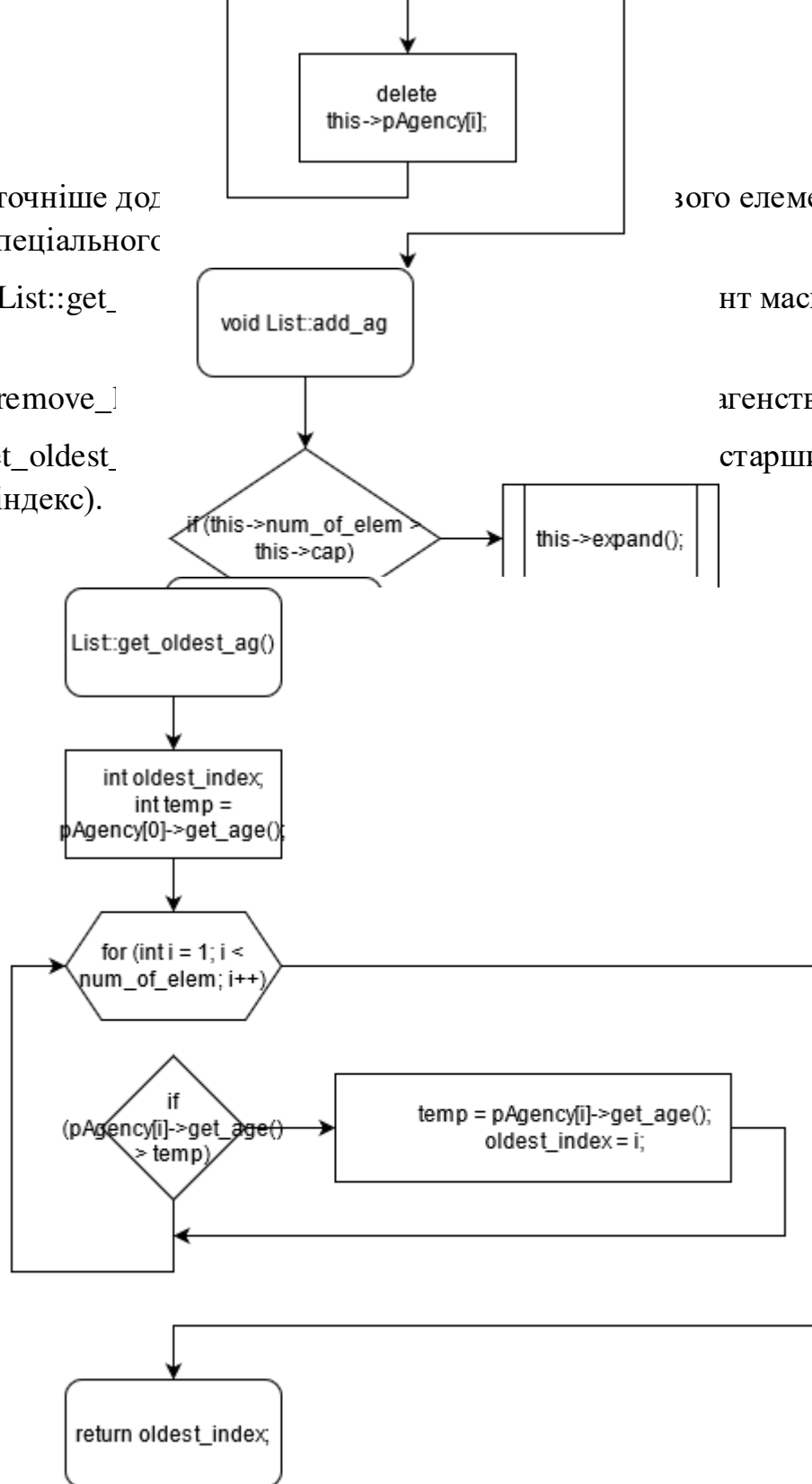


рис. 5 - int List::get_oldest_ag()

- **Важливі елементи програми**

```
List my_list;

my_list.add_ag(Agency("Kostya", "Gladkov", "Email", "help", "London", 10));
my_list.add_ag(Agency("Someone", "Noone", "Email", "nope", "London", 15));

for (int i = 0; i < my_list.get_num(); i++){
    cout << my_list.get_ag(i).get_info() << "\n";
}

this->cap *= 2;

Agency **temp_arr = new Agency*[this->cap];

for (int i = 0; i < this->num_of_elem; i++)
{
    temp_arr[i] = new Agency(*this->pAgency[i]);
}

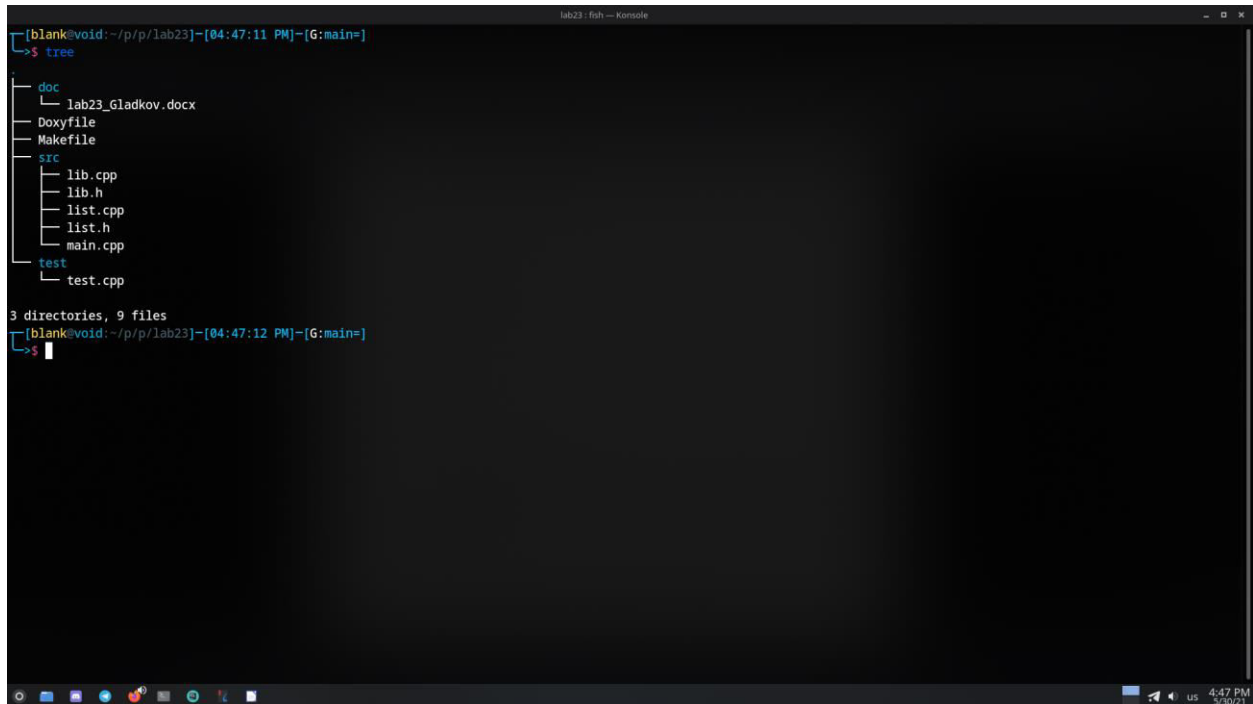
for (int i = 0; i < this->num_of_elem; i++)
{
    delete this->pAgency[i];
}

delete [] this->pAgency;

this->pAgency = temp_arr;

this->initialize(this->num_of_elem);
```

- Структура

A terminal window titled 'lab23 / fish -- Konsole' showing the output of the 'tree' command. The directory structure is as follows:

```
.
├── doc
│   └── lab23_Gladkov.docx
├── Doxyfile
├── Makefile
├── src
│   ├── lib.cpp
│   ├── lib.h
│   ├── list.cpp
│   ├── list.h
│   └── main.cpp
├── test
└── test.cpp
```

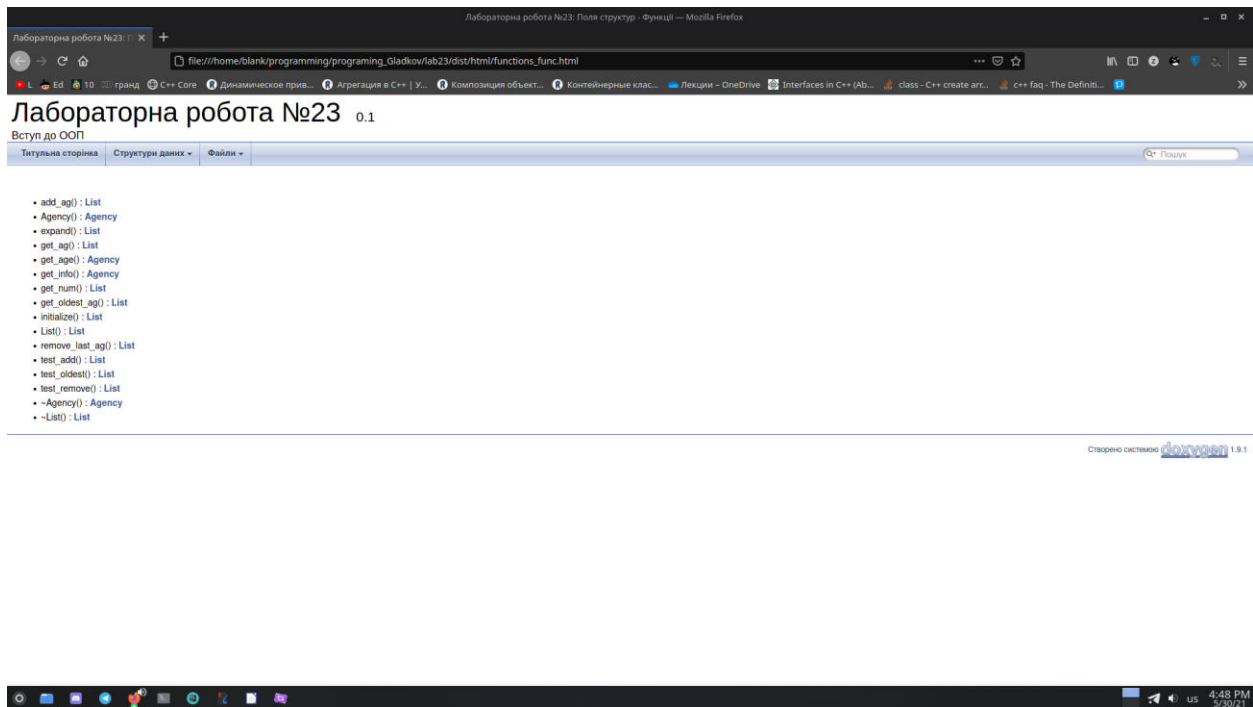
Below the tree output, it says '3 directories, 9 files'. The prompt is '[blank@void:~/p/p/lab23]--[04:47:11 PM]--[G:main=]' and the user has entered '\$' followed by a cursor.

```
[blank@void:~/p/p/lab23]--[04:47:11 PM]--[G:main=]
>$ tree

.
├── doc
│   └── lab23_Gladkov.docx
├── Doxyfile
├── Makefile
├── src
│   ├── lib.cpp
│   ├── lib.h
│   ├── list.cpp
│   ├── list.h
│   └── main.cpp
├── test
└── test.cpp

3 directories, 9 files
[blank@void:~/p/p/lab23]--[04:47:12 PM]--[G:main=]
>$
```

- Doxygen



• Варіанти використання

Дана програма може бути використана для роботи з базовим класом, а також класом-списком, який буде зберігати в собі динамічний масив базового класу.

Висновки

У даній лабораторній роботі були придбані знання роботи з ООП.