УДК 623.765.4:004.422.63

М.П. Батуринський, Д.С. Комін, Д.Ю. Свистунов

Харківський національний університет Повітряних Сил ім. І. Кожедуба, Харків

ВАРІАНТ ІДЕНТИФІКАЦІЇ ЦІЛЕЙ ПРИ ПОБУДОВІ СИСТЕМИ ОБМІНУ ІНФОРМАЦІЄЮ ПРО ПОВІТРЯНУ ОБСТАНОВКУ МІЖ КЗА ЗРВ ТА МОЖЛИВИМ ДОДАТКОВИМ ДЖЕРЕЛОМ ДАНИХ

В статті розглядається взаємодія системи постачання інформації про повітряну обстановку на КЗА ЗРВ з іншими абонентами, можливо різних типів. При такому обміні необхідно точно ідентифікувати різні цілі від різних джерел інформації. В статті запропонований об'єктно-орієнтований підхід до реалізації цієї задачі за допомогою язика програмування С++. Запропонована ієрархія класів, можливі контейнери для збереження об'єктів та спосіб доступу до даних. Окремо розглянуто питання пошуку цілі з заданим номером від заданого джерела за допомогою алгоритмів бібліотеки стандартних шаблонів. Основними характеристиками запропонованого способу є чітка структурованість та можливість розширення списку джерел інформації без суттєвого ускладнення існуючої системи.

Ключові слова: *структури даних, повітряні цілі, обмін інформацією.*

Вступ

На момент розробки КЗА ЗРВ, що на теперішній час перебувають на озброєнні ЗРВ ЗС України, передбачалось отримання інформації про повітряну обстановку від радіолокаційних засобів, що знаходяться у підрозділах ЗРВ (вищих, підлеглих, або взаємодіючих) та від КЗА РТВ. Нажаль, на протязі досить тривалого часу розвиток у цьому напрямку був досить повільний [1–2]. Тому, на теперішній час, на фоні активного розвитку телекомунікаційних систем в інших галузях господарства ситуація лишає бажати кращого. Але за короткий термін появились успішні спроби використання сучасних обчислювальних засобів для передачі, обробки і узагальнення інформації про повітряну обстановку із використанням сучасних технологій (Ethernet, супутниковий зв'язок, цифрові радіостанції тощо) [3]. Це дало змогу суттєво підвищити швидкість передачі інформації, її точність та обсяг. Наступним етапом є впровадження автоматизованого способу передачі цієї інформації на КП підрозділів ЗРВ [4]. У якості приклада можна уявити систему обміну інформацією між КЗА підрозділів ЗРВ, що включається в штатний канал обміну інформацією з вищестоящим КП. Ця система отримує дані від подібної системи РТВ за допомогою каналу Ethernet. Отримані дані про повітряну обстановку вона може передавати на обидва КП, імітуючи для одного підлеглий КП а для іншого вищестоящий КП. Крім того, при виявлені засобами радіолокації ЗРВ нових цілей ця система може одразу передати цю інформацію на засоби РТВ. З дуже малою затримкою ця інформація може буди доступна іншим споживачам. Крім того, вказана система може дуже легко масштабуватись, добавляючи нові джерела даних та нових споживачів. Наприклад, можливо організувати оперативний ввід даних про БПЛА або інші маловисотні цілі від постів візуального спостереження. Попадаючи до системи вона дуже швидко може бути доступна іншим користувачам. Оскільки кількість джерел даних (напрямків обміну інформацією) може бути декілька і вони можуть бути різного типу, то є необхідність ідентифікації цілі від різних джерел.

В існуючих КЗА ЗРВ це питання вирішувалось досить просто. Для цього використовувалась двомірна матриця в оперативній пам'яті цифрового обчислювального комплексу (ЦОК). Де номер рядка відповідав номеру цілі цього КЗА, а в одному із стовпчиків заносився номер цієї цілі у ЦОК іншого КЗА. Для такого підходу існує кілька причин. По-перше, максимальний номер цілі є доволі невелике число, а кількість абонентів, з якими КЗА проводить обмін інформацією є також постійним і невеликим числом, що обумовлюється конструкцією КЗА. По-друге, язик програмування, що використовувався для опису алгоритмів роботи КЗА, не заохочував використання складних структур даних а швидкодія ЦОК примушувала проводити оптимізацію роботи програми для забезпечення роботи основних алгоритмів.

Основний матеріал

На теперішній час ситуація є майже протилежна. Кількість абонентів, з якими проводиться обмін інформацією заздалегідь не є відомим. А при успішному розповсюдженні системи, скоріше за все, будуть з'являтися нові абоненти або напрямки обміну інформацією, що не передбачались при проектуванні системи. Крім того, немає необхідності обмежувати номер цілі невеликим числом. Навпаки, буде дуже зручно, коли під час роботи системи номери не будуть повторюватись. Це спростить документування роботи системи та аналіз файлів документування.

Тому варіант створення прямокутної матриці для відповідності нумерування не є раціональним.

У статті запропонований можливий підхід до створення структур даних для вирішення означеного питання. Для реалізації обраний універсальний язик програмування C++ за його поєднання в собі швидкості роботи та можливість маніпулювання абстрактними даними [5]. Для нумерації цілей пропонується використовувати класи (структури) даних, що утворюють собою ієрархічне дерево наслідування. Для наглядного відображення буде використовуватись нотація універсального язика програмування UML [6]. Базовий клас CGenerTargIndexInfo буде інкапсулювати загальні ознаки (рис. 1). Це може бути безпосередньо номер, тип абонента, час останнього оновлення інформації по цій цілі. Для кожного типу абоненту, з яким буде проводитись обмін інформацією про повітряну обстановку, буде використовуватись свій клас, породжений від CGenerTargIndexInfo.

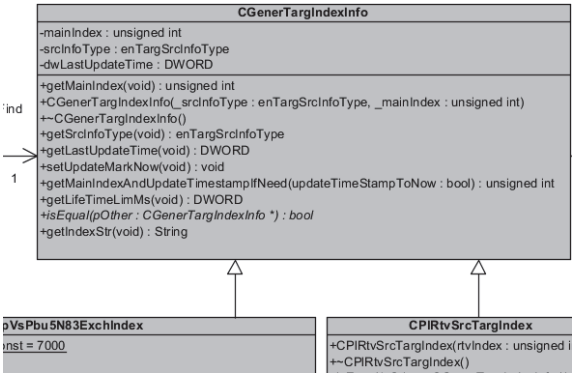

Рис. 1. Структура базового класу

Цей клас є абстрактним класом. Він декларує методи bool isEqual(const CGenerTargIndexInfo* pOther) та String getIndexStr(void). Перший служить для організації уніфікованого пошуку цілі з заданим номером заданого джерела даних. Другий метод getIndexStr відображає номер в тому вигляді, який більше за все підходить для відображення номерів від заданого джерела інформації. Тому під поняттям "номер цілі" буде матися на увазі екземпляр класу, що породжений від базового класу CGenerTargIndexInfo.

Таким чином, стає зрозуміло, що для кожної цілі, що "супроводжується" КЗА має бути набір записів з об’єктами (номерами). Кожен такий об’єкт відповідає нумерації цієї цілі у відповідного абонента. При роботі системи, передбачається, що інформація про ціль від зовнішнього абонента може з'являтися, зникати і навіть змінюватись. Це буде залежати від логіки роботи програми. Тому для утримання всіх номерів добре підійде шаблонний контейнер stl::list<CGenerTargIndexInfo*> з бібліотеки стандартних шаблонів STL [7]. Цей контейнер забезпечує постійний час вставки або видалення елементу контейнера в будь-якій позиції. У якості елементів контейнера мають бути вказівники на об’єкти а не самі об’єкти. В інакшому випадку відбудеться зрізання об’єктів до базового класу CGenerTargIndexInfo. При роботі передбачається активний процес створення нових об’єктів та видалення "застарілих". Крім того, самі об’єкти можуть передаватись у якості аргументів до інших функцій і методів для виконання різноманітних розрахунків. Тому для забезпечення цілісності даних пропонується контролювати тривалість життя об’єктів за допомогою "обгорток" boost::shared_ptr<CGenerTargIndexInfo> з бібліотеки boost [8]. Цей компонент забезпечує "прозорий" доступ до членів даних вказівників, які він "обгортає", що дає змогу більше фокусуватись на логіці роботи алгоритмів.

Таким чином, для кожної цілі, що присутня в програмі обміну інформацією з КЗА ЗРВ створюється контейнер std::list< boost::shared_ptr <CGenerTargIndexInfo>>. Для спрощення сприйняття коду краще перевизначити цей контейнер як targ_index_container_type. Тоді загальний список цілей, що ведуться у програмі може бути оголошений за допомогою контейнеру "карта" [7] як

std::map<unsigned int, CIndexes4Targ>
allTargIndexes;

де CIndexes4Targ – клас, елементом якого є targ_index_container_type$

allTargIndexes – глобальна змінна, що зберігає список всіх номерів цілей.

Додатковими методами цього класу є методи додавання нового номеру, видалення номеру (номерів) та запит часу останнього оновлення.

Окремо треба зупинитись на операціях доступу до даних, що зберігаються в описаному контейнеру. Найбільш частіше використовуваний, мабуть, буде запит на пошук запису цілі від заданого джерела з заданим номером. Для реалізації пошуку скористаємось стандартним шаблонним алгоритмом std::find, що входить до бібліотеки STL. Будемо використовувати варіант алгоритму з предікатом, що дасть змогу локалізувати умови відповідності номерів цілі заданому критерію пошуку. В якості такого предикату буде виступати клас CTargIMFinderSomeIndex (рис. 2). При початку роботи алгоритму пошуку буде створюватись новий екземпляр цього класу і одним із аргументів його конструктору буде номер, який необхідно знайти.

Особливістю роботи алгоритму std::find є те, що у разі позитивного результату пошуку буде повернутий ітератор [7] на елемент контейнеру std::map<unsigned int, CIndexes4Targ>. Тобто отримується вказівний на пару: номер цілі в поточній програмі і список відповідних номерів для інших

абонентів. Для безпосереднього доступу до шуканого номеру використовуються додаткові аргументи при конструюванні предиката CTargIMFinderSomeIndex. Через ці аргументи буде отриманий безпосередній доступ до шуканого номеру та ітератору контейнера std::list< boost::shared_ptr <CGenerTargIndexInfo>>, в якому він зберігається. Останній необхідний для виконання видалення номеру. Це може відбуватись автоматично, при виконанні певних умов, наприклад, перевищення часу, що пройшов після оновлення інформації, певного значення. Або за командую користувача.

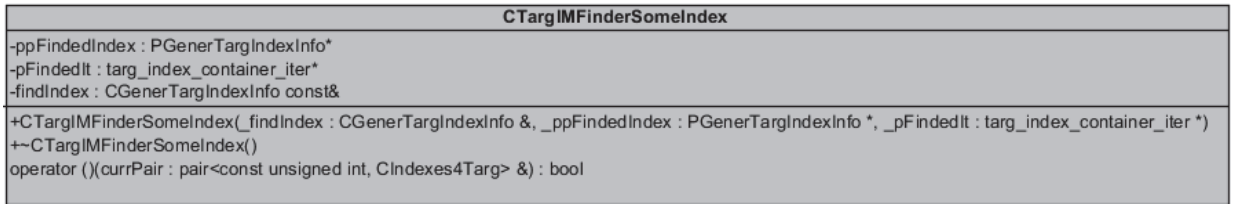

Рис. 2. Клас – предикат для алгоритму пошуку заданої цілі

Розглянемо приклад використання пошуку заданого номеру. Припустимо, система обміну інформацією працює в лінії обміну ПБУ КП ЗРС С-300П та його ВКП. Інформація про повітряну обстановку отримується від системи збору та аналізу інформації, що працює на КП ртбр. В самому простому випадку для цілей існує такий набор номерів. Нова ціль має номер РТВ. Для неї створюється відповідна структура з номером поточної програми. При видачі її до ПБУ використовується номер видачі ПБУ (аналог номеру ВКП при штатному обміні). При створенні зв'язки номерів на ПБУ до цього списку добавляється номер ПБУ, за яким в подальшому проводиться обмін з ПБУ. Для означеної ситуації пошук цілі за номером РТВ буде таким: find_if(allTargIndexes.begin(), allTargIndexes.end(), CTargIMFinderSomeIndex(CPlRtvSrcTargIndex(rtvIndex), NULL, NULL)) != allTargIndexes.end(). Цей пошук просто дає відповідь, чи є вже ціль з заданим номером РТВ. Для доступу до самого об'єкту номер цілі РТВ замість NULL при конструюванні треба передати вказівник, який буде заповнений при позитивному результату пошуку. Пошук цілі з заданим номером видачі на ПБУ буде дуже схожим find_if(allTargIndexes.begin(), allTargIndexes.end(), CTargIMFinderSomeIndex(CVkp4Pbu5N83TargIndex(zrvTargIndex), NULL, NULL)) != allTargIndexes.end(). Для доступу до самого об'єкту номер цілі видачі на ПБУ замість NULL при конструюванні треба передати вказівник, який буде заповнений при позитивному результату пошуку. Такий уніфікований пошук можливий завдяки реалізацій віртуального методу bool isEqual(const CGenerTargIndexInfo* pOther) базового класу CGenerTargIndexInfo в кожному класі що наслідується.

Проаналізуємо час роботи пошуку номеру цілей. Для перевірки реального часу роботи авторами була проведена імітація на базі програми Віраж-ЗРВ, що входить до складу апаратно програмного комплексу Віраж [9]. При замірюванні в програмі "супроводжувалося" 25 цілей. Тестування проводилось на мобільній ПЕОМ під управлінням OS Windows 7 з процесом Intel Core i53210M (початок випуску 2 квартал 2012р.).

Пошук номеру цілей за номером поточної програми займає час, пропорційний $O(\log(N))$. Це обумовлено особливістю побудови контейнеру std::map. При імітації роботи не вдалося заміряти час пошуку номеру цілі, оскільки 100 000 ітерацій пошуку зайняли час, менший ніж дискретність вимірювання 1мс. Причому це не залежало, де розташовувався номер: на початку контейнера, всередині чи наприкінці. При пошуку цілей за номером РТВ результат був інший. Середній час виконання 100 000 ітерацій пошуку цілі, що розташовувалась на початку контейнеру зайняв 40 мсек, всередині контейнеру 220 мсек, і 430 мсек для цілі, що розташовувалась наприкінці контейнеру. В цьому випадку час пошуку зростає лінійно при збільшенні кількості цілей і має величину $O(N)$. Такий час обумовлений роботою шаблонного алгоритму пошуку std::find, який послідовно проходить по елементам контейнеру та виконує процедуру порівняння з аргументом пошуку.

Можливо припустити, що при роботі кількість цілей, що одночасно супроводжуються може досягати 200. Якщо провести апроксимацію часу пошуку, що був затрачений на 25 цілей, і розглянути середній час, як половина від найгіршого випадку, то пошук для всіх 200 цілей займе 3,44 мсек. Оскільки оновлення цілей за даними РТВ передбачається проводити періодично, з інтервалом кілька секунд, то отримане значення часу, на думку авторів, можливо вважати прийнятним.

Висновки

Розвиток обчислювальної техніки дає можливість вирішувати широкий спектр завдань різного ступеню складності [10]. При такому широкому застосуванню основними вимогами, що висуваються до програм є (окрім успішного виконання поставле-

них завдань на момент прийняття роботи): можливість супроводження на прийнятному рівні, подальшого розвитку (масштабування) [11]. При невиконанні цих вимог розвиток проекту має велику ймовірність закінчитися провалом через його велику складність. Запропонований авторами спосіб організації ідентифікації цілей для різних джерел базується на обєктно-орієнтованому підході. Для кожного абоненту, з яким проводиться обмін інформацією про цілі передбачений свій клас. Кожен з таких класів, породжених від базового класу CGenerTargIndexInfo має свій набір даних. Такий підхід дозволяє маніпулювати номерами однаково, за допомогою шаблонних алгоритмів бібліотеки STL. Такий уніфікований підхід спрощує алгоритми роботи з даними та зменшує рівень складності програми. Крім того, такий підхід легко переносить масштабування. При необхідності додавання нових джерел інформації, буде лише створений новий похідний клас від CGenerTargIndexInfo. Це не вплине на роботу інших частин програми, в тому сенсі, що вони не будуть змінюватись.

Список літератури

1. *Карпенко Д.В. Стан та перспективи розвитку зенітного ракетного озброєння Повітряних Сил Збройних Сил України / Д.В. Карпенко // Наука і техніка Повітряних Сил Збройних Сил України. – 2017. – № 2. – С. 75-78.*
2. *Степаненков М.М. Шляхи вдосконалення методів отримання і обробки інформації у засобах повітряної радіотехнічної розвідки / М.М. Степаненков, А.В. Кобзєв, В.В. Романенко // Наука і техніка Повітряних Сил Збройних Сил України. – 2017. – № 2. – С. 121-123.*
3. *У зоні АТО інформація про повітряну обстановку оновлюється за декілька секунд Планшет-РТВ [Електроний ресурс] / Д. Чалий // Народна армія – 11 травня 2016. Режим доступу до публ.: http://na.mil.gov.ua/32708-u-zoni-ato-informaciya-pro-povitryanu-obstanovku-onovlyuyetsya-za-dekilka-sekund.*
4. *Кушнір О.І. Аналіз впливу «гібридної» війни на розвиток автоматизованої системи управління авіацією та ППО Збройних Сил України / О.І. Кушнір, О П. Давикоза, Ю.Ф. Кучеренко // Наука і техніка Повітряних Сил Збройних Сил України. – 2017. – № 2. – С. 116-120.*
5. *SO/IEC 14882:2003(E), Programming Languages - C++ (updated ISO and ANSI C++ standard including the contents of [C++98] plus errata corrections).*
6. *Применение UML 2.0 и шаблонов проектирования / Крэг Ларман. 3-е изд. – М.: Вильямс, 2013. – 736 с. – ISBN: 978-5-8459-1185-8.*
7. *Эффективное использование STL / С. Мейерс, Спб.: Питер, 2002. – 224 с. – ISBN: 5-94723-382-7.*
8. *Mukherjee A. Learning Boost C++ Libraries Packt Publishing, 2015. – 558 p. – ISBN-10: 1783551216, ISBN-13: 978-1-78355-121-7*
9. *Лещенко С. П. Моделирующий комплекс ведения боевых действий Воздушными Силами / С.П. Лещенко, С.И. Бурковский, М.П. Батуринский // Системи озброєння і військова техніка. – 2011. – № 2. – С. 75-79.*
10. *Лупандін В. А. Основні особливості щодо розроблення інформаційно-розрахункових задач з оцінювання радіоелектронної обстановки в інтересах Повітряних Сил ЗС України / В. А. Лупандін, С. В. Закіров, А. О. Феклістов, О. В. Сторожук, А. Г. Леушин // Системи обробки інформації. – 2017. – № 3. – С. 19-23.*
11. *Совершенный код. Мастер-класс / С Макконнелл; пер. с англ. – М.: Издательство «Русская редакция», 2010. – 896 стр.: ил.*

Надійшла до редколегії 9.06.2017

Рецензент: д-р техн. наук проф. С.П. Лещенко, Харківський національний університет Повітряних Сил ім. І. Кожедуба, Харків.

ВАРИАНТ ИДЕНТИФИКАЦИИ ЦЕЛЕЙ ПРИ ПОСТРОЕНИИ СИСТЕМЫ ОБМЕНА ИНФОРМАЦИЕЙ О ВОЗДУШНОЙ ОБСТАНОВКЕ МЕЖДУ КСА ЗРВ И ВОЗМОЖНЫМ ДОПОЛНИТЕЛЬНЫМ ИСТОЧНИКОМ ДАННЫХ

М.П. Батуринский, Д.С. Комин, Д.Ю. Свистунов

В статье рассматривается взаимодействие системы обеспечения информации о воздушной обстановке на КСА ЗРВ с другими абонентами, возможно различных типов. При таком обмене необходимо точно идентифицировать разные цели от различных источников информации. В статье предложен объектно-ориентированный поход к реализации этой задачи при помощи языка программирования С++. Предложена иерархия классов, возможные контейнеры для хранения объектов и способ доступа к данным. Отдельно рассмотрен вопрос поиска цели с заданным номером от заданного источника при помощи алгоритмов библиотеки стандартных шаблонов. Основными характеристиками предложенного способа есть чёткая структурированность и возможность расширения списка источников информации без существенного усложнения существующей системы.

Ключевые слова: *структуры данных, воздушные цели, обмен информацией.*

THE OPTION OF THE AIR TARGETS IDENTIFICATION AT BUILDING A SYSTEM OF EXCHANGE INFORMATION ABOUT THE AERIAL SITUATION BETWEEN AN AUTOMATION COMPLEX OF ANTI-AERIAL TROOPS AND LIKELY OTHERS SOURCES

M. Baturinskij, D. Komin, D. Svistunov

Interaction between a supply system of information about aerial situation of automation complex of anti-aerial troops and other subscribers, maybe different types, is considered in article. It is important to identify the different targets from plenty sources using that exchange. Object-oriented approach to realize this task with help of C++ programming language is offered in the article. The hierarchy of classes, data containers for the objects storing and a data access method is offered. Separately is mentioned a question of finding target with a certain index from a distinct source with help of the algorithms from the standard template library. The main characteristics of the offered approach are obvious structuring and possibility to extend the list of information sources without significant complexity increasing of the present system.

Keywords: *data structures, aerial targets, information exchanging.*