УДК 004.023
Марченко О.І. к.т.н. доцент, Щербина Б.О. магістрант
Національний технічний університет України «Київський політехнічний інститут»

СПОСІБ ТРАНСЛЯЦІЇ ТИПІВ ДАНИХ МОВИ COBOL В ТИПИ ДАНИХ СУЧАСНИХ МОВ ПРОГРАМУВАННЯ

Марченко О.І., Щербина Б.О. Спосіб трансляції типів даних мови COBOL в типи даних сучасних мов програмування. У статті запропоновано модифікований спосіб трансляції типів даних мови COBOL в типи даних сучасних мов програмування. Він базується на широкому використанні стандартних типів цільової мови, на відміну від більш простого підходу до реалізації семантики типів даних мови COBOL за допомогою системи додаткових класів.
Ключові слова: способи трансляції, типи даних, мова програмування COBOL, стандартні типи сучасних мов програмування.

Марченко А.И., Щербина Б.А. Способ трансляции типов данных языка COBOL в типы данных современных языков программирования. В статье предложен модифицированный способ трансляции типов языка COBOL в типы данных современных языков программирования. Он базируется на широком использовании стандартных типов целевого языка, в отличие от более простого подхода к реализации семантики типов данных языка COBOL с помощью системы дополнительных классов.
Ключевые слова: способы трансляции, типы данных, язык программирования COBOL, стандартные типы современных языков программирования.

Marchenko O.I., Shcherbyna B.O. Technique for COBOL types translation into types of modern programming languages. A modified technique for COBOL types translation into types of modern languages is proposed in this article. It is based on the wide usage of native types of the target language, rather than simplier implementing COBOL data types semantics using a set of additional classes.
Keywords: translation techniques, data types, COBOL programming language, native types of modern languages.

Вступ. Багато програм для управління технологічними та економічними процесами створені з використанням дещо застарілої в сучасних умовах мови COBOL. Для мови COBOL характерна структурна методологія програмування, яка призводить до складного процесу розширення функціональності програми. Найбільш поширеними на даний момент мовами програмування є об'єктно-орієнтовані мови, які забезпечують більш простий і швидкий процес модифікації і нарощення функціональності програмних систем. За допомогою мови COBOL нові програмні продукти вже не створюються, проте існує велика кількість програм, написаних на цій мові, працездатність яких необхідно постійно підтримувати, а в багатьох випадках також і розширювати.

Одним з підходів до розширення функціональності коду, написаного мовою COBOL є створення фактично нового програмного продукту з використанням однієї із сучасних мов програмування. Але такий підхід призводить до значних часових та фінансових витрат. Інший підхід полягає в розробці транслятора, що виконує автоматичне перетворення програм на мові COBOL в програми, що написані сучасними мовами високого рівня. Постановка задачі для таких мовних перетворень достатньо проста: необхідно перевести програму з однієї мови програмування на іншу, залишивши при цьому незмінним її зовнішню поведінку. Проблема такої трансляції залежить від наявності в цільовій мові програмування відповідних вбудованих мовних конструкцій, тобто при перекладі програм з однієї мови на іншу необхідно співставити набори вхідних та вихідних зразків кода, що і складає основну складність мовних перетворень.

Постановка наукової проблеми. Метою даної роботи є аналіз існуючих рішень з трансляції типів даних мови COBOL в типи даних сучасних мов програмування, а також розробка нового способу для такої трансляції.

Аналіз досліджень.

Типи даних мови COBOL

У мові COBOL всі структури даних оголошуються в спеціальному розділі програми, що називається DATA DIVISION або ж розділ даних. Дані можуть бути оголошені ієрархічно, кожне оголошення змінної має число-рівень, що вказує на належність змінної до іншої змінної. Оголошення змінної з більшим числом-рівнем вказує на її належність до змінної з меншим числом-рівнем. Така змінна, що містить в собі інші змінні, має назву RECORD, або "запис".

Саме оголошення змінної, окрім числа-рівня, містить ім'я змінної, її тип або формат, та необов'язкове початкове значення. Крім того, можуть бути вказані також спеціальні індикатори того, що змінна є масивом, або ж перевизначає іншу змінну. Зупинемось докладніше на форматі подання даних, що визначається конструкцією PICTURE.

Тип змінної в мові COBOL визначається описом значення, яке вона може містити, за допомогою спеціальних символів. Кожний символ відповідає за одну комірку пам'яті та позначає формат даних, що може в ній міститися. Найчастіше використовуються наступні позначення:

9 — комірка для цифри;

Z — комірка для цифри, що не відображається, якщо число починається нулем;

X — комірка для символа;

S — комірка для знака змінної;

V — комірка для символа коми в числі.

Наприклад, позначення 99999 означає змінну, що може містити до п'яти цифр, тобто така змінна є числом з п'яти цифр. Аналогічно формат XXXXXXX є поданням змінної, що складається з 7 символів, тобто рядком довжиною до 7 символів.

Важливим фактом в семантиці мови COBOL є те, що значення довільної змінної може бути присвоєне будь-якій іншій змінній, включаючи присвоєння окремого числа в масив. В такому випадку відбувається неявна конвертація даних з формату першої змінної в формат іншої змінної.

Одним з існуючих трансляторів з мови COBOL у мову Java є RES (An Open Cobol To Java Translator). Основною ідеєю трансляції типів даних у цьому трансляторі є створення класів, відповідних кожному типу даних, та організація взаємодії цих класів, наприклад, арифметичні операції реалізовані за допомогою певних методів. Ці згенеровані класи містять бінарне представлення даних, доступ до яких відбувається через методи цього класу. Такий підхід дозволяє повністю змоделювати поведінку типів даних мови COBOL, оскільки класи містять саме бінарне подання даних — подібним чином дані зберігаються і в мові COBOL.

Однак код, що згенерований таким способом, має два суттєвих недоліки. По-перше, такий код є досить громіздким через необхідність генерації всієї системи класів та методів для них. По-друге, змінні навіть найбільш примітивних типів, наприклад такі, що складаються з пари цифр, в такій програмі перетворяться у об'єкти класів із десятком методів для подання цих даних у різних форматах. Це призводить до суттєвого ускладнення потенціально досить простого коду, і як наслідок, більш складної подальшої роботи з ним.

На відміну від існуючих підходів, в роботі пропонується спосіб, який реалізує пряму трансляцію типів даних мови COBOL в стандартні типи даних цільових мов високого рівня. Під стандартними типами маються на увазі базові типи мов високого рівня, що використовуються для зберігання даних елементарних форматів: цілих і дробових чисел, символів і рядків.

У випадку мови COBOL подібні перетворення не можна виконати без додавання спеціальної логіки для деяких операцій, наприклад, округлення до певного знаку після коми, конвертації та приведення різних типів, тощо.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження.

Спосіб трансляції, що пропонується, складається з наступних кроків.

1. Програму, що написана мовою COBOL, транслюють в проміжне представлення, що напряму відображає поведінку та типи даних мови COBOL за допомогою спеціальної системи уніфікованих проміжних класів, таких як:

```
Interface Movable;
Class PicData<...> : Movable;
Class CobolArray<...> : Movable;
Class BinRep : Movable;
```

Тут інтерфейс Movable служить для відтворення семантики присвоєння значення довільної змінної в будь-яку іншу змінну. Параметризований клас PicData призначений для подання простої змінної. Параметром цього класу є "кобольне" подання змінної, тобто подання за допомогою

конструкції PICTURE. Параметризований клас CobolArray слугує для представлення масивів, де параметром класу є власне тип масиву. Клас BinRep (binary representation) слугує для внутрішнього подання даних у єдиному бінарному форматі, тобто саме у такому форматі, який використовується трансляторами мови COBOL.

Всі класи містять методи для конвертації в тип BinRep, а також конструктори, що приймають цей тип. Це необхідно для відтворення семантики присвоєння мови COBOL.

Таким чином, всі прості змінні транслюють в об'єкти класу PicData, а всі змінні-масиви — в об'єкти класу CobolArray. Всі записи ієрархічно транслюють в класи, а змінні, в такому випадку, є полями відповідних класів.

2. Типи змінних мови COBOL перетворюють в типи, що використовуються в заданій мові високого рівня. Відповідність перетворення типів на прикладі мови C# представлена в таблиці 1:

Таблиця 1. Таблиця відповідності типів мов COBOL та C#

PIC clause	Scope	Corresponding Type in C#
S9(n) $1 \le n \le 9$	- 999,999,999 ÷ 999,999,999	int
S9(n) $10 \le n \le 18$	- 999,999,999,999,999,999÷ 999,999,999,999,999,999	long
S9(m)V9(n) $1 \le m + n \le 7$	7 decimal digit precision	float
S9(m)V9(n) $8 \le m + n \le 15$	15 decimal digit precision	double
S9(m)V9(n) $16 \le m + n \le 28$	28 decimal digit precision	decimal
9(n) $1 \le n \le 9$	0 ÷ 999,999,999	unit
9(n) $10 \le n \le 18$	0 ÷ 999,999,999,999,999,999	ulong
9(m)V9(n) $1 \le m + n \le 7$	7 decimal digit precision	float
9(m)V9(n) $8 \le m + n \le 15$	15 decimal digit precision	double
9(m)V9(n) $16 \le m + n \le 28$	28 decimal digit precision	decimal

3. Операції над змінними мови COBOL перетворюють на операції, що визначені для типів заданої мови високого рівня. При цих перетвореннях повністю зберігається семантика роботи з даними мови COBOL. Конвертація типів виконується за допомогою класу BinRep, який має в собі інформацію, необхідну для трансформування даних відповідно до семантики мови COBOL.

4. За допомогою додаткового аналізу виконують трансформації для подальшого спрощення коду, якщо потрібно. Головним чином, ці трансформації спрямовані на заміну конструкцій з використанням класу BinRep на прості конструкції заданої мови високого рівня. Основними конструкціями для покращення є конвертація типів, а також операції з рядками.

Приклад

Для прикладу розберемо трансформацію фрагменту коду, що включає в себе семантичну конструкцію мови COBOL під назвою "модифікація за посиланням". Подібна конструкція передбачає зміну деяких розрядів змінної, не змінюючи при цьому інших. У сучасних мовах програмування подібні дії можна виконати над деякими структурами даних, але для цілочислених типів подібна операція не є визначеною. Відтак, задача трансляції подібної конструкції передбачає використання додаткових конструкцій для відтворення семантики початкової конструкції. Отож, фрагмент коду мовою COBOL:

```
01 date PIC 9(8).
01 year PIC X(4).
01 month PIC X(2).
01 day PIC X(2).

….
```

```
MOVE year TO date(1 : 4).
MOVE month TO date (5 : 2).
MOVE day TO date (7 : 2).
```

В наведеному прикладі у перші чотири розряди восьмирозрядної змінної date записується значення змінної year, в п'ятий та шостий розряди — значення month і в сьомий та восьмий — значення змінної day. Використовуючи наведену схему трансляції згенерований код буде поетапно виглядати наступним чином:

```
{
        BinRep __refmod__ = null;
        __mt_refmod__ =
        CobolUtils.toBinRep("9(08)", date).subrepmod(PicData<S9>(BinRep(-1)),
PicData<9>(BinRep(1)).add(PicData<9>(BinRep(4))).sub(PicData<9>(BinRep(1))), CobolUtils.toBinRep("X(04)", year));
        date =__mt_refmod__.subrep(PicData<9>(BinRep(0)), PicData<9>(BinRep(0)) +
PicData<9>(BinRep(8))).asuint32_t("99999999");
}
{
        BinRep __refmod__ = null;
        __refmod__ =
        CobolUtils.toBinRep("9(08)", date).subrepmod(PicData<9>(BinRep(5)).sub(PicData<9>(BinRep(1))),
PicData<9>(BinRep(5)).add(PicData<9>(BinRep(2))).sub(PicData<9>(BinRep(1))), CobolUtils.toBinRep("X(02)", month));
        date = __mt_refmod__.subrep(PicData<9>(BinRep(0)),
PicData<9>(BinRep(0)).add(PicData<9>(BinRep(8)))).asuint32_t("99999999");
}
{
        BinRep __refmod__ = null;
        __refmod__ =
        CobolUtils.toBinRep("9(08)", date).subrepmod(PicData<9>(BinRep(7)).sub(PicData<9>(BinRep(1))),
PicData<9>(BinRep(7)).add(PicData<9>(BinRep(2))).sub(PicData<9>(BinRep(1))), CobolUtils.toBinRep("X(02)", day));
        date =__mt_refmod__.subrep(PicData<9>(BinRep(0)),
PicData<9>(BinRep(0))).add(PicData<9>(BinRep(8))).asuint32_t("99999999");
}
```

На другому та третьому етапі відбувається заміна типів, що відповідають простим об'єктам на типи мови високого рівня. В результаті цього етапу перетворення фрагмент буде виглядаті наступним чином:

```
{
        BinRep __refmod__ = null;
        __mt_refmod__ =
        CobolUtils.toBinRep("9(08)", date).subrepmod(1 -1,1 +4 -1, CobolUtils.toBinRep("X(04)", year));
        date =__mt_refmod__.subrep(0, 0 + 8).asuint32_t("99999999");
}
{
        BinRep __refmod__ = null;
        __refmod__ =
        CobolUtils.toBinRep("9(08)", date).subrepmod(5 -1,5 +2 -1, CobolUtils.toBinRep("X(02)", month));
        date = __mt_refmod__.subrep(0, 0 + 8).asuint32_t("99999999");
}
{
        BinRep __refmod__ = null;
        __refmod__ =
        CobolUtils.toBinRep("9(08)", date).subrepmod(7 -1,7 +2 – 1, CobolUtils.toBinRep("X(02)", day));
        date =__mt_refmod__.subrep(0, 0 + 8).asuint32_t("99999999");
}
```

За семантикою мови відбувається присвоєння у відповідні розряди через внутрішнє подання даних, що в наведеному прикладі імітується використанням класу BinRep. Таку конструкцію можна істотно спростити, розпізнавши, що присвоєння відбувається у одну і ту ж змінну, а її розряди оновлюються послідовно. Таким чином, можлива заміна на лише одну операцію, що передбачає конкатенацію рядків і їх конвертацію в число. В результаті присвоєння виду:

```
date = CobolUtils.toBinRep(year + month + day).asuint32_t("99999999");
```

У такому варіанті змінні year, month і day транслюються в стороковий тип (string), потім відбувається їх конкатенація і перетворення у цілочислений тип змінної date.

Висновки

Код, що генерується запропонованим способом має більш чітку та інтуїтивно зрозумілу структуру і семантику, а також є суттєво меншим за розміром. Конструкції, що додаються в п.1, в загальному випадку мають бути замінені на конструкції мови високого рівня. Враховуючи широкий спектр можливих комбінацій таких конструкцій, а також істотний семантичний розрив мови COBOL із сучасною методологією програмування, основною задачею для подальшого вдосконалення способу є розробка ефективних трансформацій для їх перетворення на етапі описаному у п.4.

1. RES – An Open Cobol To Java Translator / Sourceforge [Електронний ресурс]. Режим доступу: http://sourceforge.net/projects/opencobol2java/
2. Enterprise COBOL for z/OS / IBM [Електронний ресурс]. Режим доступу: http://www-03.ibm.com/software/products/en/entecoboforzos
3. Свердлов С.З. Языки программирования и методы трансляции: Учебное пособие. - СПб.: Питер, 2007. - 638 с.
4. Описание синтаксиса языков программирования [Электронный ресурс]. Режим доступа: http://wiki.mvtom.ru/index.php
5. Терехов А., Верхув К. Проблемы языковых преобразований [Электронный ресурс]. Режим доступа: http://www.osp.ru/os/2001/05-06/180189/
6. Widmer R. COBOL Migration Planning, Edge Information Group, 1998.
7. Y. Chae and S. Rogers, Successful COBOL Upgrades: Highlights and Programming Techniques, John Wiley and Sons, New York, 1999
8. Молдованова О.В. Языки программирования и методы трансляции: Учебное пособие. – Новосибирск, 2012. – 134 с.
9. Ахо А.В. Компиляторы: принципы, технологии и инструментарий / [Ахо А.В., Лам М.С., Сети Р., Ульман Дж.Д.] 2-е изд. : Пер. с англ. – М.: Изд. дом Вильямс, 2008. –1184 с.
10. Fast Lexical Analyzer. [Электронный ресурс]. Режим доступа: http://flex.sourceforge.net/