**АНОТАЦІЯ**

Кваліфікаційна робота включає пояснювальну записку (56с., 40 рис., 4 додатки).

Об’єкт розробки – підсистема оптимізації циклів транслятора мови COBOL на графі залежностей значень і станів.

Розроблена підсистема дозволяє:

* будувати граф залежностей значень і станів вхідної програми;
* відокремлювати обчислення у циклах мови COBOL, що не містять побічних ефектів, від обчислень в процедурній парадигмі;
* генерувати вихідний код програми на мові JAVA з використанням Java Stream Api.

В підсистемі передбачені механізми захисту від введення некоректної програми та від помилок під час виконання. В процесі розробки була використана мова програмування Common Lisp з використанням бібліотек cl-lex, cl-yacc та редактор GNU Emacs з пакетом SLIME. Для компіляції коду мовою Common Lisp було використано високопродуктивний компілятор SBCL (Steel Bank Common Lisp).

В ході виконання дипломного проекту:

* проведено аналіз представлення циклів мови COBOL;
* проведено аналіз представлення циклів за допомогою Java Stream Api;
* проведено аналіз можливих оптимізацій циклів;
* розроблено архітектуру підсистеми.

Використання цієї підсистеми дозволить виявляти цикли, які потім можуть бути оптимізовані при трансляції у Java Stream Api.

Приведені необхідні схеми та документація, підведені підсумки щодо проведеної роботи.

Ключові слова: граф залежностей значень і станів, цикл, COBOL, Java Stream Api, COMMON LISP, CL-LEX, CL-YACC, GNU EMACS, SLIME, SBCL.

**ABSTRACT**

Qualification work includes explanation note (56 pages, 40 drawings, 4 attachments).

The object of development is the cycle optimization subsystem of COBOL language compiler on value state dependency graph.

Developed subsystem enables:

* to build a value state dependency graph of input program;
* to allocate calculations in cycles of language COBOL, which do not contain side effects, from calculations in the procedural paradigm;
* to generate the output code of the JAVA program using Java Stream Api.

The subsystem provides safeguards against the introduction of incorrect input application and runtime errors. In the development process has been used programming language Common Lisp with libraries cl-lex, cl-yacc and GNU Emacs editor with service SLIME. To compile the Common Lisp language code used high-performance compiler SBCL (Steel Bank Common Lisp).

During the development of the diploma project:

* analysis of representation of COBOL language cycles were done;
* analysis of representation of cycles by means of Java Stream Api were done;
* analysis of possible optimization of cycles were done;
* the architecture of the subsystem was developed.

Using of this system will detect cycles, which can then be optimized during the translation into other programming languages.

The necessary schemes, documentation and summed up by the work done were included.

Keywords: values state dependency graph, loop, COBOL, COMMON LISP, CL-LEX, CL-YACC, GNU EMACS, SLIME, SBCL.

**АННОТАЦИЯ**

Квалификационная работа включает объяснительную записку (56 с., 40 рис., 4 приложений).

Объект разработки – подсистема оптимизации циклов транслятора языка COBOL на графе зависимостей значений и состояний.

Разработанная подсистема позволяет:

* строить граф зависимостей значений и состояний входной программы;
* выделять вычисления в циклах языка COBOL, которые не содержат побочных эффектов, от вычислений в процедурной парадигме;
* генерировать выходной код программы на языке JAVA с использованием Java Stream Api.

В подсистеме предусмотрены механизмы защиты от введения некорректной программы и от ошибок во время выполнения. В процессе разработки был использован язык программирования Common Lisp с использованием библиотек cl-lex, cl-yacc и редактор GNU Emacs с пакетом SLIME. Для компиляции кода на языке Common Lisp был использован высокопроизводительный компилятор SBCL (Steel Bank Common Lisp).

В ходе выполнения дипломного проекта:

* проведён анализ представления циклов языка COBOL;
* проведён анализ представления циклов с помощью Java Stream Api;
* проведён анализ возможных оптимизаций циклов;
* разработано архитектуру подсистемы.

Использование данной подсистемы позволит определять циклы, которые потом могут быть оптимизированы при трансляции в Java Stream Api.

Приведены необходимые схемы и документация, подведены итоги по проделанной работы.

Ключевые слова: граф зависимостей значений и состояний, цикл, COBOL, Java Stream Api, COMMON LISP, CL-LEX, CL-YACC, GNU EMACS, SLIME, SBCL.