Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
Институт математики и информационных технологий
Кафедра информационных технологий

ОТЧЕТ

по научно-исследовательской работе

ИССЛЕДОВАНИЕ КОМПИЛЯТОРА ЯЗЫКА D

Студента _ курса очного отделения
группы 02_4_-ДБ
Фамилия Имя Отчество

Руководитель:
к. т. н., доцент
_____ Иванов Иван Иванович

Защищен с оценкой

_____

Иркутск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Реализация компиляторов языков программирования – одно из основных направлений в области системного программирования, включающего разработку трансляторов (в общем смысле, т.е. и компиляторов и интерпретаторов). Трансляторы языков программирования относятся к системам порождающего программирования (ПП), т.е. программным системам, задача которых создать исходный код или какой-либо другой объект по некоторому описанию, модели, исходному информационному объекту. Применение ПП предполагает, что исходный конформационный объект меняется достаточно редко, поэтому имеет смысл повысить производительность целевой системы за счет представления предварительного анализа объекта в виде последовательности инструкций целевого вычислителя, реализующих уже результат анализа. Сама процедура анализа выполняется один раз транслятором.

Разработка трансляторов позволяет решать следующие задачи:

- Разрабатывать новые системы программирования;
- Переносить существующий программный код языка высокого уровня на новые вычислительные платформы, например, микроконтроллеры;
- Разрабатывать языки описания предметной области, представляющие объекты предметной области в удобном для пользователя виде, а, затем, преобразовывать описание в какой-либо другой язык для решения задачи;
- Проводить исследования в области системного программирования и защиты информации;
- Развивать практические аспекты теории формальных языков.

Целью данной курсовой работы является исследование компилятора `LDC` версии 2.0 языка программирования высокого уровня `D`.

В курсовой работе решены следующие **задачи**:

1. Изучен язык программирования `D`, параметры компилятора `LDC2` и система сборки пакетов `dub`.

2. Создана программа, микросервис веб, вычисляющий факториал передаваемого значения.

3. Настроена система сборки проекта, реализована сборка.

4. Исследовано использование аргументов компилятора в процессе сборки проекта.

5. Осуществлен запуск компилятора для исследуемого модуля с целью трансляции исходного кода в промежуточное представление (ПП).

6. Выявлен и проанализирован текст ПП, относящийся к исследуемому методу.

1 Язык D и среда программирования LDC

Разработка D задумывалась как реинжениринг языка C++ с целью избавиться от наиболее существенных недостатков исходного языка и внедрить в него современные архитектурные решения. При создании языка D была сделана попытка соединить производительность компилируемых языков программирования с безопасностью и выразительностью динамических [1].

Первоначально автор предполагал назвать язык «Mars», но из-за преемственности по отношению к C++ в обсуждениях язык постоянно называли «D», в результате именно это название и закрепилось за проектом.

Стабильная версия компилятора 1.0 вышла 2 января 2007. Вскоре после выхода компилятора 17 июня 2007 года автор перевёл версию 1 в режим поддержки и приступил к разработке версии 2.0, которая изначально не гарантировала обратной совместимости. Эта версия (последняя в настоящее время мажорная версия D) развивается и по сей день.

D реализует пять основных парадигм программирования — императивное, ООП, метапрограммирование, функциональное программирование и параллельные вычисления (модель акторов).

D использует сборщик мусора для управления памятью, однако возможно и ручное управление с помощью перегрузки операторов new и `delete`, а также с помощью `malloc` и `free`, аналогично C. Сборщик мусора можно включать и выключать вручную, можно добавлять и удалять области памяти из его видимости, принудительно запускать частичный или полный процесс сборки. Существует подробное руководство, описывающее различные схемы управления памятью в D для тех случаев, когда стандартный сборщик мусора неприменим.

D относится к семейству C-подобных языков, в общих чертах его синтаксис похож на C/C++/C#, Java. При разработке языка соблюдается принцип: код, одинаково допустимый и в C, и в D, должен вести себя одинаково.

«Hello, world!» на D:

```
1 import std.stdio;
2
3 void main()
4 {
5     writeln ("Hello, world!");
6 }
```

Так же, как в C, функция main() является точкой входа.

Конструкции `if`, `for`, `while`, `do-while` выглядят и работают аналогично C/C++. Инструкция множественного выбора `switch` выглядит аналогично C++, но допускает переменные в метках ветвей `case` и требует, чтобы каждая ветвь case завершалась `break` или `return`; для перехода на следующую ветвь после обработки текущей необходимо использовать специальную конструкцию `goto case`. Также запрещены конструкции `switch` без ветви `default`.

Из дополнительных управляющих конструкций можно отметить `static if` — инструкцию для условной компиляции (условие проверяется статически и в код включается содержимое той ветви, которая ему соответствует), оператор полного множественного выбора `final switch` — в отличие от обычного switch, он работает только со значениями enum, а компилятор статически проверяет, что в выборе учтены все возможные варианты и выдаёт ошибку в противном случае. Также есть цикл по коллекции `foreach`.

В D встроена система разбиения программы на модули (пакеты), обеспечивающая раздельную компиляцию и контролируемый импорт-экспорт. Система пакетов напоминает принятую в Java или Go: пакеты образуют иерархическую структуру, естественно отображаемую на дерево файловой системы. В отличие от C++, в D нет глобального пространства имён, каждое имя определяется в каком-либо пакете. С помощью инструкции `import` модуль программы может импортировать пакет, сделав доступными все имеющиеся в нём определения. Обращение к импортированным именам может выполняться с квалификацией: «`имя_пакета.имя_объекта`».

Язык предусматривает ряд средств, направленных на обеспечение удобной работы с импортируемыми именами. Есть возможность переименования пакета при импорте, задания альтернативного имени (алиаса) импортируемого пакета, импорта конкретных имён. Кроме того, язык разрешает без каких-либо дополнительных инструкций использовать импортированные имена без квалификации именем пакета. Однако действует ограничение: если в области видимости есть более одного подходящего определения встреченного в программе имени, то компилятор выдаёт ошибку и требует, чтобы имя было явно квалифицировано. Это предотвращает так называемый «угон имён», когда при добавлении в списки импорта нового пакета компилятор начинает связывать некоторое имя в программе не с тем определением, с которым оно связывалось ранее.

В D юнит-тесты являются частью языка, их можно использовать без подключения дополнительных библиотек или фреймворков.

```d
import std.stdio;

int first (int[] arr) {
    return arr[0];
}

unittest {
    int[] arr1 = [1, 2, 3];
    int[] arr2 = [10, 15, 20];

    assert(first(arr1) == 1);
    assert(first(arr2) == 10);
}

void main() {
    // ...
}
```

1.1 Компилятор LDC

Проект LDC направлен на создание переносимого компилятора языка программирования D с современными возможностями оптимизации и генерации кода. Компилятор использует официальный набор утилит компилятора (так называемый «фронтенд») `DMD` для поддержки последней версии D2 и опирается на библиотеки `LLVM Core` для генерации кода [2].

Исходный код LDC является полностью открытым; части кода, не взятые/адаптированные из других проектов, лицензированы соглашением BSD.

LDC - это полностью общедоступный проект, поэтому любой вклад в разработку приветствуется. Например, если вы хотите заняться разработкой компиляторов, то, по некоторым данным, дополнение LDC новыми полезными функциями – хороший первый шаг, особенно потому, что использованная в компиляторе библиотека LLVM очень удобна в работе. Чтобы начать работу, просмотрите список нереализованных задач, найдите ту, которая понравится (некоторые из них специально помечены как «junior jobs»), и работайте над исправлением - и,

конечно, не стесняйтесь спрашивать совета в списке рассылки или на канале IRC. Список функций/улучшений, которые можно добавить в LDC, можно посмотреть в разделе «Идеи проектов LDC». Также приветствуется разработка и поддержка пакетов LDC в дистрибутивах ОС, в частности, в Debian и его производных.

Компилятор `ldc2` (пакета LDC), в целом, работает аналогично референтному компилятору Digital Mars D, как увидим дальше, удачно компилирует проект, но реализует процедуру компиляции при помощи библиотеки (так называемого «бэкенда») `LLVM` (Low level virtual machine) [3]. Это его свойство позволяет использовать компилятор `ldc2` для создания межъязыковых программных систем, интегрированных на уровне инструкций промежуточного представления (Intermediate representation, IR) скопированной программы, аналогично проекту `.NET` [4]. Но, в отличие от `.NET`, который порождает код виртуальной машины `MSIL`, транслируемый «на лету» (just in time, JIT) во время исполнения программы в машинный код микропроцессора, IR может и интерпретироваться, и транслироваться в код микропроцессора во время компиляции, т.е. до непосредственного исполнения. При этом сборку исполняемого модуля и оптимизацию программного кода можно производить еще в промежуточном представлении, что, в общем случае, улучшает производительность результирующего бинарного файла.

Список ключей компилятора `ldc2` можно получить при помощи команды `ldc2 -help`. Список ключей приведен в Приложении В на стр. 25.

[1]

2 Проектирование и реализация микросервиса

В качестве приложения спроектируем небольшой REST-сервис при помощи библиотеки HUNT. Библиотека позволяет разрабатывать WEB-сервера при помощи определения отображений URL на функции и методы экземпляров классов D. Поддерживается шаблон проектирования MVC (Model-View-Controller) [5].

Пусть сервис реализует три функции:

- Выводит по запросу текущую дату и время,
- Реализует эхо-запрос: возвращает текстовое сообщение, переданное в аргументе,
- На запрос вычисления факториала («полезная» функция) возвращает вычисленное число; аргумент передается в URL.

Для реализации REST необходимо определиться какие на сервере существуют объекты. В нашем случае все три функции можно «приписать» (инкапсулировать в) некоторый анонимный объект-синглтон (имеющийся в единственном экземпляре), соответственно, функции будут его методами.

Следующий этап – это отображение функций на тип запроса (GET, PUT, PUSH и т.д.). Требования к нашему сервису представляют собой две функции с одним параметром, и одно возвращаемое значение (дата/время). Структура входных параметров простая – единичные значения типа Целое и Строка. Следовательно для реализации запросов достаточно использования команды протокола HTTP GET.

Сформируем файл отображения URL на методы класса D, реализующего запросы:

```
1 #
2 # [GET,POST,PUT,*, ... ]    path    controller.method
3 # Symbol* can accept all request method
4 #
5
6 GET   /             index.index
7 POST  /index        index.index
8 *     /home         index.index
9 GET   /api/test     api.testApi
```

```
10 GET   /api/echo/{msg<.+>}/     api.echo
11 GET   /api/fact/{n<\d+>}/      api.fact
```

Здесь в первом столбце представлен тип HTTP-запроса, во втором – шаблоны URL, некоторые включают специальные языковые конструкции для распознавания аргумента запроса. Третий столбец – образ (адрес) метода: `api.*` – запросы на выполнение требуемых функций. Кроме необходимых функции в сервере реализованы также вспомогательные запросы, выполняющие сервисные задания.

Параметр запроса, например, как в строке (10) приведенной выше конфигурации, передается параметром в метод `echo` контроллера `ApiController` (строка 28 в тексте программы ниже). Формат параметра позволяет указывать тип данного и проводить его предварительную верификацию еще до передачи в метод контроллера. В случае, если параметр не подходит к формату, система Hunt выдает ошибку 500 (ошибка сервера).

```d
// файл app/controller/ApiController.d
// ----------------------------------

module app.controller.ApiController;

import hunt.framework;
import std.json : JSONValue;
import std.stdio;
import std.conv;

class ApiController : Controller
{
  mixin MakeController;
  @Action
  JsonResponse testApi() {
    import std.datetime;
    import std.datetime.date : DateTime;

    auto dt = Clock.currTime();
    auto dts = dt.toISOExtString();
    JSONValue js;
    js["currtime"] = dts;
    JsonResponse resp = new JsonResponse(js);
    return resp;
```

```
25    }
26
27    @Action
28    JsonResponse echo(string msg) {
29      JSONValue js;
30      js["echo"] = msg;
31      auto resp = new JsonResponse(js);
32      return resp;
33    }
34
35    @Action
36    JsonResponse fact(string n) {
37      JSONValue js;
38      auto res = fact(to!int(n));
39      js["result"] = to!string(res);
40      auto resp = new JsonResponse(js);
41      return resp;
42    }
43
44    int fact(int n) {
45      if (n==0) return 1;
46      if (n==1) return 1;
47      return n*fact(n-1);
48    }
49  }
```

Представленный выше контроллер решает все поставленные задачи. Каждый метод контроллера обрабатывает GET-запросы, т.е., параметры запроса передаются только через параметры методов контроллера. Возвращаемое значение представляется в виде JSON-структуры. Например, метод `fact(string n)`, возвращает структуру

```
{"result":"6"}
```

Пример сборки и запуска проекта продемонстрированы в следующем фрагменте текста. Сборка осуществляется командой `dub run –compiler=ldc2`, где ключ `–compiler=` используется для указания компилятора, которым собирается проект. В нашем случае – `ldc2`, поддерживающий промежуточное представление..

```
stud@sysrescue:~/projects/webapp$ dub run --compiler=ldc2
  Starting Performing "debug" build using ldc2 for x86_64.
```

```
  Up-to-date hunt 1.7.17: target for configuration [library] is up to date.
  Up-to-date hunt-extra 1.2.3: target for configuration [library] is up to date.
  Up-to-date hunt-net 0.7.1: target for configuration [default] is up to date.
  Up-to-date hunt-http 0.8.2: target for configuration [default] is up to date.
  Up-to-date protobuf 0.6.2: target for configuration [protobuf] is up to date.
  Up-to-date grpc 0.5.0-beta.2: target for configuration [library] is up to date.
  Up-to-date hunt-redis 1.4.1: target for configuration [library] is up to date.
  Up-to-date hunt-cache 0.10.1: target for configuration [library] is up to date.
  Up-to-date hunt-console 0.4.0: target for configuration [hunt-console] is up to
   ↪ date.
  Up-to-date hunt-sql 1.6.0: target for configuration [library] is up to date.
  Up-to-date hunt-database 2.3.6: target for configuration [default] is up to
   ↪ date.
  Up-to-date hunt-validation 0.5.0: target for configuration [library] is up to
   ↪ date.
  Up-to-date hunt-entity 2.8.1: target for configuration [library] is up to date.
  Up-to-date hunt-openssl 1.0.5: target for configuration [library] is up to
   ↪ date.
  Up-to-date hunt-jwt 0.2.0-beta.4: target for configuration [library] is up to
   ↪ date.
  Up-to-date hunt-security 0.6.0: target for configuration [library] is up to
   ↪ date.
  Up-to-date hunt-shiro 1.3.1: target for configuration [library] is up to date.
  Up-to-date poodinis 8.1.3: target for configuration [library] is up to date.
  Up-to-date hunt-framework 3.4.6: target for configuration [library] is up to
   ↪ date.
  Up-to-date webapp ~master: target for configuration [application] is up to
   ↪ date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running webapp


 ___  ___     ___  ___     ________      _________
|\  \|\  \   |\  \|\  \   |\   ___  \   |\___   ___\     Hunt Framework 3.4.6
\ \  \\\  \  \ \  \\\  \  \ \  \\ \  \  \|___ \  \_|
 \ \   __  \  \ \  \\\  \  \ \  \\ \  \  \     \ \  \      Listening: 0.0.0.0:8080
  \ \  \ \  \  \ \  \\\  \  \ \  \\ \  \  \     \ \  \     TLS: Disabled
   \ \__\ \__\  \ \_______\  \ \__\\ \__\      \ \__\
    \|__|\|__|   \|_______|   \|__| \|__|       \|__|
     ↪ https://www.huntframework.com

```

```
35
36 Try to browse http://0.0.0.0:8080
```

Работоспособность сервера проверим в отдельном окне, выполнив команду `ps -FC webapp`, где `-F` требует от программы представить вывод в подробной форме, а `-C` показывает различные статистические данные подсистемы распределения вычислительных ресурсов по отношению к нашему процессу.

```
1 stud@sysrescue:~/projects/webapp/source/app/controller$ ps -FC webapp
2 UID          PID    PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
3 stud       18551   18544  0 163131 37192  0 17:25 pts/1    00:00:00
   ↪ /home/stud/projects/webapp/webapp
```

Таким образом, мы убеждаемся, что процесс сервера запущен, привязан к виртуальному терминалу `pts/1` и находится в режиме ожидания запросов (соединений клиентов).

Далее протестируем работоспособность функций REST, реализованных в сервисе. Для этого используем утилиту `curl`, выполняющую функции HTTP-клиента, но работающую в командной строке. При помощи этой утилиты можно выполнять весь спектр запросов REST, передавать и получать данные и файлы. Выполним три запроса: собственно запрос на сервис, вычисление факториала от аргумента; запрос текущей даты в стандартном интернет-формате; тестовое сообщение (эхо-сервис).

```
1 stud@sysrescue:~/projects/webapp/source/app/controller$ curl
   ↪ http://localhost:8080/api/fact/9
2 {"result":"362880"}
3 stud@sysrescue:~/projects/webapp/source/app/controller$ curl
   ↪ http://localhost:8080/api/test
4 {"currtime":"2023-11-07T17:25:59.8807696"}
5 stud@sysrescue:~/projects/webapp/source/app/controller$ curl
   ↪ http://localhost:8080/api/echo/message-to-test
6 {"echo":"message-to-test"}
```

В приведенном выше примере продемонстрирована работоспособность сервиса при правильных входных данных.

Следующий пример демонстрирует ответ сервера при неправильном входном значении.

```
  <!--
stud@sysrescue : ~/projects/webapp/source/app/controller curl
 ↪  http://localhost:8080/api/fact/9e
  -->
<!doctype html>
<html lang="en">
    <meta charset="utf-8">
    <title>404 Not Found</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>

        * {
            line-height: 3;
            margin: 0;
        }

        html {
            color: #888;
            display: table;
            font-family: sans-serif;
            height: 100%;
            text-align: center;
            width: 100%;
        }

        body {
            display: table-cell;
            vertical-align: middle;
            margin: 2em auto;
        }

        h1 {
            color: #555;
            font-size: 2em;
            font-weight: 400;
        }

        p {
            margin: 0 auto;
            width: 90%;
        }
```

```
41
42      </style>
43  </head>
44  <body>
45      <h1>404 Not Found</h1>
46      <p>Sorry!! Unable to complete your request :(</p>
47
48  </body>
49  </html>
```

Таким образом, разработанное серверное приложение функционирует согласно требованиям.

3 Исследование скомпилированного кода

Рассмотрим фрагмент контроллера, реализующий вычисление факториала.

```
module app.controller.ApiController;

import hunt.framework;
import std.json : JSONValue;
import std.stdio;
import std.conv;

class ApiController : Controller
{
  mixin MakeController;

  // . . . . . .

  @Action
  JsonResponse fact(string n) {
    JSONValue js;
    auto res = fact(to!int(n));
    js["result"] = to!string(res);
    auto resp = new JsonResponse(js);
    return resp;
  }

  int fact(int n) {
    if (n==0) return 1;
    if (n==1) return 1;
    return n*fact(n-1);
  }
}
```

Метод `int fact(int n)` конвертирован в промежуточное представление, в результате получен следующий текст, представленный далее. Интересные моменты прокомментированы в тексте на русском языке.

```
; [#uses = 1]
; Function Attrs: uwtable
define i32 @_D3app10controller13ApiControllerQp4factMFiZi
    (%app.controller.ApiController.ApiController* nonnull %.this_arg, i32 %n_arg)
     ↪ #0 {
```

```
  ; Длинное название функции обусловлено вхождением исходного метода
  ; в контексты пакета app.controller и модуль ApiController.
  ; В метод в качестве первого аргумента передается указатель
  ; на экземпляр класса ApiController - this.
  %this = alloca %app.controller.ApiController.ApiController*, align 8 ; [#uses =
   ↪ 3, size/byte = 8]
  %n = alloca i32, align 4  ; выделение памяти для n [#uses = 5, size/byte = 4]
  store %app.controller.ApiController.ApiController* %.this_arg,
   ↪ %app.controller.ApiController.ApiController** %this, align 8
  store i32 %n_arg, i32* %n, align 4 ; сохранение входного n в локальную
   ↪ переменную n
  %1 = load i32, i32* %n, align 4 ; загрузка локальной n; [#uses = 1]
  %2 = icmp eq i32 %1, 0 ; сравнение ее с 0       ; [#uses = 1]
  br i1 %2, label %if, label %endif ; если больше 0, переход на метку endif

if:                                               ; preds = %0
  ret i32 1 ; если n = 0 то вернуть в качестве результата 1.

dummy.afterreturn:                                ; No predecessors!
  br label %endif ; ненужный код, ужаляется оптимизатором.

endif:                                            ; preds = %dummy.afterreturn,
 ↪ %0
  %3 = load i32, i32* %n, align 4                 ; [#uses = 1]
  %4 = icmp eq i32 %3, 1 ; сравнение n с 1        ; [#uses = 1]
  br i1 %4, label %if1, label %endif2 ; если больше, перейти на метку endif2

if1:                                              ; preds = %endif
  ret i32 1 ; fact(1) = 1

dummy.afterreturn3:                               ; No predecessors!
  br label %endif2

endif2:                                           ; preds = %dummy.afterreturn3,
 ↪ %endif
  %5 = load i32, i32* %n, align 4                 ; [#uses = 1]
  %6 = load %app.controller.ApiController.ApiController*,
   ↪ %app.controller.ApiController.ApiController** %this, align 8 ; [#uses = 1]
  %7 = getelementptr inbounds %app.controller.ApiController.ApiController,
   ↪ %app.controller.ApiController.ApiController* %6, i32 0, i32 0 ; [#uses = 1,
   ↪ type = [38 x i8*]**]
```

```
38    %8 = load [38 x i8*]*, [38 x i8*]** %7, align 8 ; [#uses = 1]
39    %"fact@vtbl" = getelementptr inbounds [38 x i8*], [38 x i8*]* %8, i32 0, i32 36
      ↪  ; [#uses = 1, type = i8**]
40    %9 = load i8*, i8** %"fact@vtbl", align 8       ; загрузка адреса таблицы
      ↪  виртуальных методов
41    %fact = bitcast i8* %9 to i32 (%app.controller.ApiController.ApiController*,
      ↪  i32)* ; вычисление адреса Fact
42    %10 = load %app.controller.ApiController.ApiController*,
      ↪  %app.controller.ApiController.ApiController** %this, align 8 ; [#uses = 1]
43    %11 = load i32, i32* %n, align 4                ; [#uses = 1]
44    %12 = sub i32 %11, 1                            ; n1 = n-1
45    %13 = call i32 %fact(%app.controller.ApiController.ApiController* nonnull %10,
      ↪  i32 %12) ; вычисление fact(n-1)
46    %14 = mul i32 %5, %13                           ; %13 * схраненное значение n
47    ret i32 %14    ; возврат значения факториала.
48  }
```

Как видно из примера, все строки исходной программы преобразовались в промежуточное представление. Так же к ним добавлены команды, например в строках 36-42, где решаются задачи диспетчеризации виртуальных методов. Получается, что все методы в D виртуальные, что одновременно упрощает компилятор, но чуть-чуть замедляет процесс исполнения программы.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы получен интересный факт о компиляторе `ldc2` языка `D`. Оказывается все методы экземпляров некоторого класса – виртуальные. В документации по языку этот факт не описан явным образом, но исследование компилятора, проведенное в данной курсовой работе, показало его наличие. Как язык программирования для пользовательских приложений `D` вполне пригоден, так как использование только виртуальных методов упрощает компилятор, соответственно, делая его более простым. Однако такой подход несколько сказывается на производительности исполняемого кода.

В процессе реализации курсовой работы решены следующие задачи:

1. Изучен язык программирования `D`, параметры компилятора `LDC2` и система сборки пакетов `dub`.

2. Создана программа, микросервис веб, вычисляющий факториал передаваемого значения, в сервисе реализовано четыре полезные функции.

3. Настроена система сборки проекта, реализована сборка.

4. Исследовано использование аргуметов компилятора в процессе сборки проекта.

5. Осуществлен запуск компилятора для исследуемого модуля с целью трансляции исходного кода в промежуточное представление.

6. Выявлен и проанализирован текст промежуточного представления, относящийся к исследуемому методу.

СПИСОК ИПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. D (язык программирования), Материал из Википедии — свободной энциклопедии – URL: https://ru.wikipedia.org/wiki/D_(язык_программирования)
2. LDC Wiki – URL: https://wiki.dlang.org/LDC
3. The LLVM Compiler Infrastructure – URL: https://llvm.org/
4. Build. Test. Deploy. – URL: https://dotnet.microsoft.com/
5. Model–view–controller, Материал из Википедии — свободной энциклопедии – URL: https://en.wikipedia.org/wiki/Model–view–controller

Приложение А Исходный код программ

Здесь приводится main-функция приложения – REST-сервера, реализованного при помощи библиотеки обратного вызова (фреймворка) HUNT.

```
// main.d
//-------
module main;

import hunt.framework;
import hunt.logging;

void main(string[] args)
{
  LogConf conf;
  logLoadConf(conf);
  app().run(args);
}
```

Данный файл – основной, он реализует «полезную» функцию – вычисление факториала. Доступ к функции осуществляется согласно архитектуре REST при помощи GET-запроса.

```
// файл app/controller/ApiController.d
// -----------------------------------

module app.controller.ApiController;

import hunt.framework;
import std.json : JSONValue;
import std.stdio;
import std.conv;

class ApiController : Controller
{
  mixin MakeController;
  @Action
  JsonResponse testApi() {
    import std.datetime;
    import std.datetime.date : DateTime;

```

```
    auto dt = Clock.currTime();
    auto dts = dt.toISOExtString();
    JSONValue js;
    js["currtime"] = dts;
    JsonResponse resp = new JsonResponse(js);
    return resp;
  }

  @Action
  JsonResponse echo(string msg) {
    JSONValue js;
    js["echo"] = msg;
    auto resp = new JsonResponse(js);
    return resp;
  }

  @Action
  JsonResponse fact(string n) {
    JSONValue js;
    auto res = fact(to!int(n));
    js["result"] = to!string(res);
    auto resp = new JsonResponse(js);
    return resp;
  }

  int fact(int n) {
    if (n==0) return 1;
    if (n==1) return 1;
    return n*fact(n-1);
  }
}
```

Данный модуль используется для проверки работоспособности фреймворка Hunt.

```
// файл app/controller/ApiController.d
// -----------------------------------

module app.controller.IndexController;

import hunt.framework;
```

```
7
8  class IndexController : Controller
9  {
10   mixin MakeController;
11
12   @Action
13   string index()
14   {
15     return "Hello world!";
16   }
17
18 }
```

Для запуска сервера необходим конфигурационный модуль его настройки.

```
1  #
2  # файл config/application.conf
3  #
4  http.address = 0.0.0.0
5  http.port = 8080
6  http.allowOrigin=*
7  http.allowMethods=*
8  http.allowHeaders=DNT,X-CustomHeader,Keep-Alive,User-Agent,X-Requested-With,If-Modifie
9  http.enableCors=true
10
11 hunt.log.level= DEBUG
12 hunt.log.path= ./log
13 hunt.log.file= log.txt
14 hunt.log.maxSize = 8M
15 hunt.log.maxNum= 10
```

Интерфейс сервера REST настраивается следующим образом:

```
1  #
2  # Файл config/routes
3  #
4  #
5  # [GET,POST,PUT,*, ... ]    path    controller.method
6  # Symbol* can accept all request method
7  #
8
9  GET   /             index.index
```

```
10 POST  /index        index.index
11 *     /home         index.index
12 GET   /api/test     api.testApi
13 GET   /api/echo/{msg<.+>}/     api.echo
14 GET   /api/fact/{n<\d+>}/      api.fact
```

Приложение В Ключи команды `ldc2`

```
stud@sysrescue:~/go$ ldc2 --help
OVERVIEW: LDC - the LLVM D compiler

USAGE: ldc2 [options] files --run Runs the resulting program, passing the
 ↪  remaining arguments to it

OPTIONS:

General options:

  -D                                      - Generate documentation
  --Dd=<directory>                        - Write documentation file to
   ↪  <directory>
  --Df=<filename>                         - Write documentation file to
   ↪  <filename>
  -H                                      - Generate 'header' file
  --HC                                    - Generate C++ header file
                                            Use -HC=verbose to add comments
                                             ↪  for ignored declarations
                                             ↪  (e.g. extern(D))
  --HCd=<directory>                       - Write C++ 'header' file to
   ↪  <directory>
  --HCf=<filename>                        - Write C++ 'header' file to
   ↪  <filename>
  --Hd=<directory>                        - Write 'header' file to
   ↪  <directory>
  --Hf=<filename>                         - Write 'header' file to
   ↪  <filename>
  --Hkeep-all-bodies                      - Keep all function bodies in .di
   ↪  files
  -I <directory>                          - Look for imports also in
   ↪  <directory>
  -J <directory>                          - Look for string imports also in
   ↪  <directory>
  Setting the optimization level:
      -O                                    - Equivalent to -O3
      --O0                                   - No optimizations (default)
      --O1                                   - Simple optimizations
      --O2                                   - Good optimizations
```

```
      --O3                                  - Aggressive optimizations
      --O4                                  - Equivalent to -O3
      --O5                                  - Equivalent to -O3
      --Os                                  - Like -O2 with extra
       ↪  optimizations for size
      --Oz                                  - Like -Os but reduces code
       ↪  size further
  -P <cppflag>                              - Pass <cppflag> to C preprocessor
  -X                                        - Generate JSON file
  --Xf=<filename>                           - Write JSON file to <filename>
  --allinst                                 - Generate code for all template
   ↪  instantiations
  --betterC                                 - Omit generating some runtime
   ↪  information and helper functions
  --boundscheck=<value>                     - Array bounds check
    =off                                    -   Disabled
    =safeonly                               -   Enabled for @safe functions
     ↪  only
    =on                                     -   Enabled for all functions
  -c                                        - Compile only, do not link
  --cache=<cache dir>                       - Enable compilation cache, using
   ↪  <cache dir> to store cache files
  --cache-prune                             - Enable cache pruning.
  --cache-prune-expiration=<dur>            - Sets the pruning expiration time
   ↪  of cache files to <dur> seconds (default: 1 week). Implies -cache-prune.
  --cache-prune-interval=<dur>              - Sets the cache pruning interval
   ↪  to <dur> seconds (default: 20 min). Set to 0 to force pruning. Implies
   ↪ -cache-prune.
  --cache-prune-maxbytes=<size>             - Sets the maximum cache size to
   ↪  <size> bytes. Implies -cache-prune.
  --cache-prune-maxpercentage=<perc>        - Sets the cache size limit to
   ↪  <perc> percent of the available space (default: 75%). Implies -cache-prune.
  --cache-retrieval=<value>                 - Set the cache retrieval
   ↪  mechanism (default: copy).
    =copy                                   -   Make a copy of the cache file
    =hardlink                               -   Create a hard link to the
     ↪  cache file (recommended)
    =link                                   -   Equal to 'hardlink' on
     ↪  Windows, but 'symlink' on Unix and OS X
    =symlink                                -   Create a symbolic link to the
     ↪  cache file
```

```
54  --checkaction=<value>                          - Action to take when an
     ↪ assert/boundscheck/final-switch fails
55    =D                                           -   Usual D behavior of throwing
       ↪ an AssertError
56    =C                                           -   Call the C runtime library
       ↪ assert failure function
57    =halt                                        -   Halt the program execution
       ↪ (very lightweight)
58    =context                                     -   Use D assert with context
       ↪ information (when available)
59  --conf=<filename>                              - Use configuration file
     ↪ <filename>
60  --cov                                          - Compile-in code coverage
     ↪ analysis and .lst file generation
61                                                   Use -cov=<n> for n% minimum
                                                      ↪ required coverage
62                                                   Use -cov=ctfe to include code
                                                      ↪ executed during CTFE
63  --cov-increment=<value>                        - Set the type of coverage line
     ↪ count increment instruction
64    =default                                     -   Use the default (atomic)
65    =atomic                                      -   Atomic increment
66    =non-atomic                                  -   Non-atomic increment (not
       ↪ thread safe)
67    =boolean                                     -   Don't read, just set counter
       ↪ to 1
68  --cs-profile-generate                          - Perform context sensitive PGO
     ↪ instrumentation
69  --cs-profile-path=<string>                     - Context sensitive profile file
     ↪ path
70  Allow deprecated language features and symbols:
71      -d                                           - Silently allow deprecated
         ↪ features and symbols
72      --dw                                          - Issue a message when
         ↪ deprecated features or symbols are used (default)
73      --de                                          - Issue an error when
         ↪ deprecated features or symbols are used (halt compilation)
74  --d-debug[=<level/idents>]                     - Compile in debug code ⩾
     ↪ <level> or identified by <idents>
75  --d-version=<level/idents>                     - Compile in version code ⩾
     ↪ <level> or identified by <idents>
```

```
76  --deps[=<filename>]                              - Write module dependencies to
     ↪  <filename> (only imports). '-deps' alone prints module dependencies
     ↪  (imports/file/version/debug/lib)
77  --enable-asserts=<value>                         - (*) Enable assertions
78  --disable-d-passes                               - Disable all D-specific passes
79  --disable-gc2stack                               - Disable promotion of GC
     ↪  allocations to stack memory
80  --enable-invariants=<value>                      - (*) Enable invariants
81  --disable-loop-unrolling                         - Disable loop unrolling in all
     ↪  relevant passes
82  --disable-loop-vectorization                     - Disable the loop vectorization
     ↪  pass
83  --disable-red-zone                               - Do not emit code that uses the
     ↪  red zone.
84  --disable-simplify-drtcalls                      - Disable simplification of
     ↪  druntime calls
85  --disable-simplify-libcalls                      - Disable simplification of
     ↪  well-known C runtime calls
86  --dllimport=<value>                              - Windows only: which extern(D)
     ↪  global variables to dllimport implicitly if not defined in a root module
87    =none                                          -   None (default with
       ↪  -link-defaultlib-shared=false)
88    =defaultLibsOnly                               -   Only druntime/Phobos symbols
       ↪  (default with -link-defaultlib-shared and -fvisibility=hidden).
89    =all                                           -   All (default with
       ↪  -link-defaultlib-shared and -fvisibility=public)
90  --dwarf-version=<int>                            - Dwarf version
91  --enable-color=<value>                           - (*) Force colored console output
92  --enable-contracts=<value>                       - (*) Enable function pre- and
     ↪  post-conditions
93  --enable-inlining=<value>                        - (*) Enable function inlining
     ↪  (default in -O2 and higher)
94  --enable-postconditions=<value>                  - (*) Enable function
     ↪  postconditions
95  --enable-preconditions=<value>                   - (*) Enable function
     ↪  preconditions
96  --enable-switch-errors=<value>                   - (*) Enable runtime errors for
     ↪  unhandled switch cases
97  --extern-std=<value>                             - C++ standard for name mangling
     ↪  compatibility
```

```
98    =c++98                                    -   Sets `__traits(getTargetInfo,
       ↪ "cppStd")` to `199711`
99    =c++11                                    -   Sets `__traits(getTargetInfo,
       ↪ "cppStd")` to `201103` (default)
100   =c++14                                    -   Sets `__traits(getTargetInfo,
       ↪ "cppStd")` to `201402`
101   =c++17                                    -   Sets `__traits(getTargetInfo,
       ↪ "cppStd")` to `201703`
102   =c++20                                    -   Sets `__traits(getTargetInfo,
       ↪ "cppStd")` to `202002`
103  --fcf-protection                           - Instrument control-flow
      ↪ architecture protection
104  --fcf-protection=<value>                   - Instrument control-flow
      ↪ architecture protection
105   =none
106   =branch
107   =return
108   =full
109  --fdmd-trace-functions                     - DMD-style runtime performance
      ↪ profiling of generated code
110  --ffast-math                               - Set @fastmath for all functions.
111  --finstrument-functions                    - Instrument function entry and
      ↪ exit with GCC-compatible profiling calls
112  --float-abi=<value>                        - ABI/operations to use for
      ↪ floating-point types:
113   =default                                  -   Target default floating-point
       ↪ ABI
114   =soft                                     -   Software floating-point ABI
       ↪ and operations
115   =softfp                                   -   Soft-float ABI, but hardware
       ↪ floating-point instructions
116   =hard                                     -   Hardware floating-point ABI
       ↪ and instructions
117  --flto=<value>                            - Set LTO mode, requires linker
      ↪ support
118   =full                                     -   Merges all input into a single
       ↪ module
119   =thin                                     -   Parallel importing and codegen
       ↪ (faster than 'full')
120  --flto-binary=<file>                       - Set the linker LTO plugin
      ↪ library file (e.g. LLVMgold.so (Unixes) or libLTO.dylib (Darwin))
```

```
121 --fno-delete-null-pointer-checks             - Treat null pointer dereference
     ↪ as defined behavior when optimizing (instead of _un_defined behavior). This
     ↪ prevents the optimizer from assuming that any dereferenced pointer must not
     ↪ have been null and optimize away the branches accordingly.
122 --fno-discard-value-names                    - Do not discard value names in
     ↪ LLVM IR
123 --fno-plt                                    - Do not use the PLT to make
     ↪ function calls
124 --fp-contract=<value>                        - Enable aggressive formation of
     ↪ fused FP ops
125   =fast                                      -   Fuse FP ops whenever
       ↪ profitable
126   =on                                        -   Only fuse 'blessed' FP ops.
127   =off                                       -   Only fuse FP ops when the
       ↪ result won't be affected.
128 --fprofile-generate[=<filename>]              - Generate instrumented code to
     ↪ collect a runtime profile into default.profraw (overriden by '=<filename>'
     ↪ or LLVM_PROFILE_FILE env var)
129 --fprofile-instr-generate[=<filename>]        - Generate instrumented code to
     ↪ collect a runtime profile into default.profraw (overriden by '=<filename>'
     ↪ or LLVM_PROFILE_FILE env var)
130 --fprofile-instr-use=<filename>              - Use instrumentation data for
     ↪ profile-guided optimization
131 --fprofile-use=<filename>                    - Use instrumentation data for
     ↪ profile-guided optimization
132 --frame-pointer=<value>                      - Specify frame pointer
     ↪ elimination optimization
133   =all                                       -   Disable frame pointer
       ↪ elimination
134   =non-leaf                                  -   Disable frame pointer
       ↪ elimination for non-leaf frame
135   =none                                      -   Enable frame pointer
       ↪ elimination
136 --fsanitize=<checks>                         - Turn on runtime checks for
     ↪ various forms of undefined or suspicious behavior.
137 --fsanitize-address-use-after-return=<value> - Select the mode of detecting
     ↪ stack use-after-return (UAR) in AddressSanitizer: never | runtime (default)
     ↪ | always
138   =never                                     -   Completely disables detection
       ↪ of UAR errors (reduces code size).
```

```
139     =runtime                                -   Adds the code for detection,
         ↪ but it can be disabled via the runtime environment
         ↪ (ASAN_OPTIONS=detect_stack_use_after_return=0). Requires druntime
         ↪ support.
140     =always                                 -   Enables detection of UAR
         ↪ errors in all cases. (reduces code size, but not as much as never).
         ↪ Requires druntime support.
141   --fsanitize-blacklist=<file>              - Add <file> to the blacklist
       ↪ files for the sanitizers.
142   --fsanitize-coverage=<type>               - Specify the type of coverage
       ↪ instrumentation for -fsanitize
143   --fsanitize-memory-track-origins=<int>    - Enable origins tracking in
       ↪ MemorySanitizer (0=disabled, default)
144   --fsave-optimization-record[=<filename>]     - Generate a YAML optimization
       ↪ record file of optimizations performed by LLVM
145   --fsplit-stack                            - Use segmented stack (see Clang
       ↪ documentation)
146   --fthread-model=<value>                   - Thread model
147     =global-dynamic                         -   Global dynamic TLS model
       ↪ (default)
148     =local-dynamic                          -   Local dynamic TLS model
149     =initial-exec                           -   Initial exec TLS model
150     =local-exec                             -   Local exec TLS model
151   --ftime-trace                             - Turn on time profiler. Generates
       ↪ JSON file based on the output filename (also see --ftime-trace-file).
152   --ftime-trace-file=<filename>             - Specify time trace file
       ↪ destination
153   --ftime-trace-granularity=<uint>          - Minimum time granularity (in
       ↪ microseconds) traced by time profiler
154   --fvisibility=<value>                     - Default visibility of symbols
155     =default                                -   Hidden for Windows targets
       ↪ without -shared, otherwise public
156     =hidden                                 -   Only export symbols marked
       ↪ with 'export'
157     =public                                 -   Export all symbols
158   --fwarn-stack-size=<threshold>            - Warn for stack size bigger than
       ↪ the given number
159   --fxray-instruction-threshold=<value>     - Sets the minimum function size
       ↪ to instrument with XRay
160   --fxray-instrument                        - Generate XRay instrumentation
       ↪ sleds on function entry and exit
```

```
Generating debug information:
    -g                                  - Add symbolic debug info
    --gc                                 - Add symbolic debug info,
     ↪ optimize for non D debuggers
    --gline-tables-only                  - Add line tables only
--gdwarf                                - Emit DWARF debuginfo (instead of
 ↪ CodeView) for MSVC targets
--hash-threshold=<uint>                 - Hash symbol names longer than
 ↪ this threshold (experimental)
-i[=<pattern>]                            - Include imported modules in
 ↪ the compilation
--ignore                                - Ignore unsupported pragmas
--lib                                   - Create static library
    --linkonce-templates                 - Use discardable linkonce_odr
     ↪ linkage for template symbols and lazily & recursively define all
     ↪ referenced instantiated symbols in each object file
    --linkonce-templates-aggressive      - Experimental, more aggressive
     ↪ variant
--lowmem                                - Enable the garbage collector for
 ↪ the LDC front-end. This reduces the compiler memory requirements but
 ↪ increases compile times.
--lto-aix-system-assembler=<path>       - Path to a system assembler,
 ↪ picked up on AIX only
--lto-pass-remarks-filter=<regex>       - Only record optimization remarks
 ↪ from passes whose names match the given regular expression
--lto-pass-remarks-format=<format>      - The format used for serializing
 ↪ remarks (default: YAML)
--lto-pass-remarks-output=<filename>    - Output filename for pass remarks
--m32                                   - 32 bit target
--m64                                   - 64 bit target
--mabi=<string>                         - The name of the ABI to be
 ↪ targeted from the backend
--main                                  - Add default main() if not
 ↪ present already (e.g. for unittesting)
--makedeps[=<filename>]                   - Write module dependencies in
 ↪ Makefile compatible format to <filename>/stdout (only imports)
--march=<string>                        - Architecture to generate code
 ↪ for (see --version)
--mattr=<a1,+a2,-a3,...>                - Target specific attributes
 ↪ (-mattr=help for details)
```

```
184 --mcpu=<cpu-name>                                - Target a specific cpu type
     ↪ (-mcpu=help for details)
185 --mdcompute-file-prefix=<prefix>                 - Prefix to prepend to the
     ↪ generated kernel files.
186 --mdcompute-targets=<targets>                    - Generates code for the specified
     ↪ DCompute target list. Use 'ocl-xy0' for OpenCL x.y, and 'cuda-xy0' for CUDA
     ↪ CC x.y
187 --mixin=<filename>                               - Expand and save mixins to
     ↪ <filename>
188 --mtriple=<string>                              - Override target triple
189 --mv=«package.module»=<filespec»                 - Use <filespec> as source file for
     ↪ <package.module>
190 --noasm                                         - Disallow use of inline assembler
191 --nogc                                          - Do not allow code that generates
     ↪ implicit garbage collector calls
192 --o-                                            - Do not write object file
193 --od=<directory>                                - Write object files to
     ↪ <directory>
194 --of=<filename>                                 - Use <filename> as output file
     ↪ name
195 --op                                            - Preserve source path for output
     ↪ files
196 --oq                                            - Write object files with fully
     ↪ qualified names
197 --output-bc                                     - Write LLVM bitcode
198 --output-ll                                     - Write LLVM IR
199 --output-mlir                                   - Write MLIR
200 --output-o                                      - Write native object
201 --output-s                                      - Write native assembly
202 --passmanager=<value>                           - Setting the passmanager
     ↪ (new,legacy):
203   =legacy                                       -   Use the legacy passmanager
       ↪ (available for LLVM14 and below)
204   =new                                          -   Use the new passmanager
       ↪ (available for LLVM14 and above)
205 --plugin=<dynamic_library.so,lib2.so>           - Pass plugins to load.
206 --preview=<name>                                - Enable an upcoming language
     ↪ change identified by <name>, use ? for list
207 --release                                       - Compile release version,
     ↪ defaulting to disabled asserts/contracts/invariants, and bounds checks in
     ↪ @safe functions only
```

```
208 --relocation-model=<value>                  - Choose relocation model
209   =static                                  -   Non-relocatable code
210   =pic                                     -   Fully relocatable, position
       ↪  independent code
211   =dynamic-no-pic                          -   Relocatable external
       ↪  references, non-relocatable code
212   =ropi                                    -   Code and read-only data
       ↪  relocatable, accessed PC-relative
213   =rwpi                                    -   Read-write data relocatable,
       ↪  accessed relative to static base
214   =ropi-rwpi                               -   Combination of ropi and rwpi
215 --revert=<name>                            - Revert language change
     ↪  identified by <name>, use ? for list
216 --run <string>...                          - Runs the resulting program,
     ↪  passing the remaining arguments to it
217 --shared                                   - Create shared library (DLL)
218 --singleobj                                - Create only a single output
     ↪  object file
219 --template-depth=<uint>                    - Set maximum number of nested
     ↪  template instantiations
220 --threads=<int>                            -
221 --transition=<name>                        - Help with language change
     ↪  identified by <name>, use ? for list
222 --unittest                                 - Compile in unit tests
223 -v                                         - Verbose
224 --v-cg                                     - Verbose codegen
225 --vcolumns                                 - Print character (column) numbers
     ↪  in diagnostics
226 --verror-style=<value>                     - Set the style for file/line
     ↪  number annotations on compiler messages
227   =digitalmars                             -   'file(line[,column]): message'
       ↪  (default)
228   =gnu                                     -   'file:line[:column]: message',
       ↪  conforming to the GNU standard used by gcc and clang
229 --verror-supplements=<uint>                - Limit the number of supplemental
     ↪  messages for each error (0 means unlimited)
230 --verrors=<uint>                           - Limit the number of error
     ↪  messages (0 means unlimited)
231 --verrors-context                          - Show error messages with the
     ↪  context of the erroring source line
```

```
232   --verrors-spec                                - Show errors from speculative
      ↪ compiles such as __traits(compiles, ... )
233   --vgc                                         - List all gc allocations
      ↪ including hidden ones
234   --vtemplates                                  - List statistics on template
      ↪ instantiations
235                                                   Use -vtemplates=list-instances
                                                      ↪ to additionally show all
                                                      ↪ instantiation contexts for
                                                      ↪ each template
236   --vv                                          - Print front-end/glue code debug
      ↪ log
237   Warnings:
238       -w                                          - Enable warnings as errors
          ↪ (compilation will halt)
239       --wi                                         - Enable warnings as messages
          ↪ (compilation will continue)
240   --wo                                          - Enable warnings about use of
      ↪ obsolete features
241
242 Generic Options:
243
244   --help                                        - Display available options
      ↪ (--help-hidden for more)
245   --help-list                                   - Display list of available
      ↪ options (--help-list-hidden for more)
246   --version                                     - Display the version of this
      ↪ program
247
248 Linking options:
249
250   -L <linkerflag>                               - Pass <linkerflag> to the linker
251   --Xcc=<ccflag>                                - Pass <ccflag> to GCC/Clang for
      ↪ linking/preprocessing
252   --defaultlib=<lib1,lib2, ... >                - Default libraries to link with
      ↪ (overrides previous)
253   --disable-linker-strip-dead                   - Do not try to remove unused
      ↪ symbols during linking
254   --gcc=<gcc|clang| ... >                       - C compiler to use for linking
      ↪ (and external assembling). Defaults to the CC environment variable if set,
      ↪ otherwise to `cc`.
```

```
255   --link-defaultlib-debug                       - Link with debug versions of
       ↪ default libraries
256   --link-defaultlib-shared                      - Link with shared versions of
       ↪ default libraries. Defaults to true when generating a shared library
       ↪ (-shared).
257   --linker=<lld-link|lld|gold|bfd| ... >        - Set the linker to use. When
       ↪ explicitly set to '' (nothing), prevents LDC from passing `-fuse-ld` to
       ↪ `cc`.
258   --mscrtlib=<libcmt[d]|msvcrt[d]>              - MS C runtime library to link
       ↪ with
259   --platformlib=<lib1,lib2, ... >               - Platform libraries to link with
       ↪ (overrides previous)
260   --static                                      - Create a statically linked
       ↪ binary, including all system dependencies
261
262 Polly Options:
263 Configure the polly loop optimizer
264
265   --polly                                       - Enable the polly optimizer (with
       ↪ -O1, -O2 or -O3)
266   --polly-2nd-level-tiling                      - Enable a 2nd level loop of loop
       ↪ tiling
267   --polly-ast-print-accesses                    - Print memory access functions
268   --polly-context=<isl parameter set>           - Provide additional constraints
       ↪ on the context parameters
269   --polly-dce-precise-steps=<int>               - The number of precise steps
       ↪ between two approximating iterations. (A value of -1 schedules another
       ↪ approximation stage before the actual dead code elimination.
270   --polly-delicm-max-ops=<int>                  - Maximum number of isl operations
       ↪ to invest for lifetime analysis; 0=no limit
271   --polly-detect-full-functions                 - Allow the detection of full
       ↪ functions
272   --polly-enable-simplify                       - Simplify SCoP after
       ↪ optimizations
273   --polly-ignore-func=<string>                  - Ignore functions that match a
       ↪ regex. Multiple regexes can be comma separated. Scop detection will ignore
       ↪ all functions that match ANY of the regexes provided.
274   --polly-isl-arg=<argument>                    - Option passed to ISL
275   --polly-matmul-opt                            - Perform optimizations of matrix
       ↪ multiplications based on pattern matching
```

```
276   --polly-on-isl-error-abort                  - Abort if an isl error is
       ↪ encountered
277   --polly-only-func=<string>                  - Only run on functions that match
       ↪ a regex. Multiple regexes can be comma separated. Scop detection will run
       ↪ on all functions that match ANY of the regexes provided.
278   --polly-only-region=<identifier>            - Only run on certain regions (The
       ↪ provided identifier must appear in the name of the region's entry block
279   --polly-only-scop-detection                 - Only run scop detection, but no
       ↪ other optimizations
280   --polly-optimized-scops                     - Polly - Dump polyhedral
       ↪ description of Scops optimized with the isl scheduling optimizer and the
       ↪ set of post-scheduling transformations is applied on the schedule tree
281   --polly-parallel                            - Generate thread parallel code
       ↪ (isl codegen only)
282   --polly-parallel-force                      - Force generation of thread
       ↪ parallel code ignoring any cost model
283   --polly-pattern-matching-based-opts         - Perform optimizations based on
       ↪ pattern matching
284   --polly-postopts                            - Apply post-rescheduling
       ↪ optimizations such as tiling (requires -polly-reschedule)
285   --polly-pragma-based-opts                   - Apply user-directed
       ↪ transformation from metadata
286   --polly-pragma-ignore-depcheck              - Skip the dependency check for
       ↪ pragma-based transformations
287   --polly-process-unprofitable                - Process scops that are unlikely
       ↪ to benefit from Polly optimizations.
288   --polly-register-tiling                     - Enable register tiling
289   --polly-report                              - Print information about the
       ↪ activities of Polly
290   --polly-reschedule                          - Optimize SCoPs using ISL
291   --polly-show                                - Highlight the code regions that
       ↪ will be optimized in a (CFG BBs and LLVM-IR instructions)
292   --polly-show-only                           - Highlight the code regions that
       ↪ will be optimized in a (CFG only BBs)
293   --polly-stmt-granularity=<value>            - Algorithm to use for splitting
       ↪ basic blocks into multiple statements
294     =bb                                       -   One statement per basic block
295     =scalar-indep                             -   Scalar independence heuristic
296     =store                                    -   Store-level granularity
297   --polly-tc-opt                              - Perform optimizations of tensor
       ↪ contractions based on pattern matching
```

```
  --polly-tiling                                - Enable loop tiling
  --polly-vectorizer=<value>                    - Select the vectorization
    ↪ strategy
    =none                                       -   No Vectorization
    =polly                                      -   Polly internal vectorizer
    =stripmine                                  -   Strip-mine outer loops for the
      ↪ loop-vectorizer to trigger

-d-debug can also be specified without options, in which case it enables all
  ↪ debug checks (i.e. asserts, boundschecks, contracts and invariants) as well
  ↪ as acting as -d-debug=1.

Boolean options can take an optional value, e.g.,
  ↪ -link-defaultlib-shared=<true,false>.
Boolean options marked with (*) also have a -disable-FOO variant with inverted
  ↪ meaning.
```