Программные средства поддержки дистанционного обучения функциональному программированию

Институт Систем Информатики им. А. П. Ершова.
Лаборатория Конструирования и Оптимизации Программ.

Функциональные языки программирования

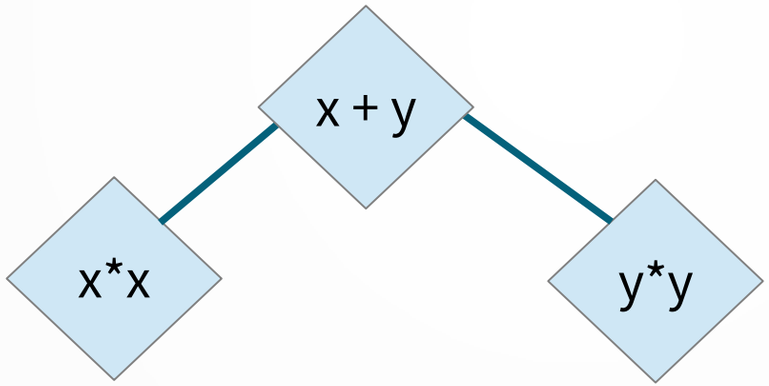
В основе функциональных языков программирования лежит лямбда-исчисление.

Примеры языков ФП:
Lisp (и его диалекты, напр. Clojure), Haskell, Erlang , F#, Ocaml

- Имеют ряд ограничений, принятие которых даёт ряд выгод
- Хорошо подходят для решения специфических задач

Функциональные языки программирования

-Не хранят состояния
-Функции не влияют друг на друга и их результат зависит только от входных данных, что позволяет кэшировать значения функций и легко распараллеливать вычисления.
-Легче составить соответствующую спецификации программу
-Проще отладка



На практике и в императивных языках бывают элементы ФП и в функциональных языках бывают элементы ИП (в парадигме ФП нет циклов, но они бывают реализованы в некоторых языках, в т.ч. в Cloud Sisal).

Обучение функциональному программированию

Для сопровождения обучения функциональному программированию предлагается разработать онлайн-среду разработки программ на языке Sisal.

Среда получается унифицированной
Все участники пользуются одной и той же, всегда самой новой, версией
Нет (значителных) ограничений по ОС, производительности пользовательского оборудования
Доступность высокопроизводительного вычислителя.
Дистанционная работа

Язык Sisal

Cloud Sisal основан на языке Sisal.
Язык программирования Sisal является одним из самых известных потоковых языков промышленного уровня и позиционируется как замена
языка Фортран для научных применений.

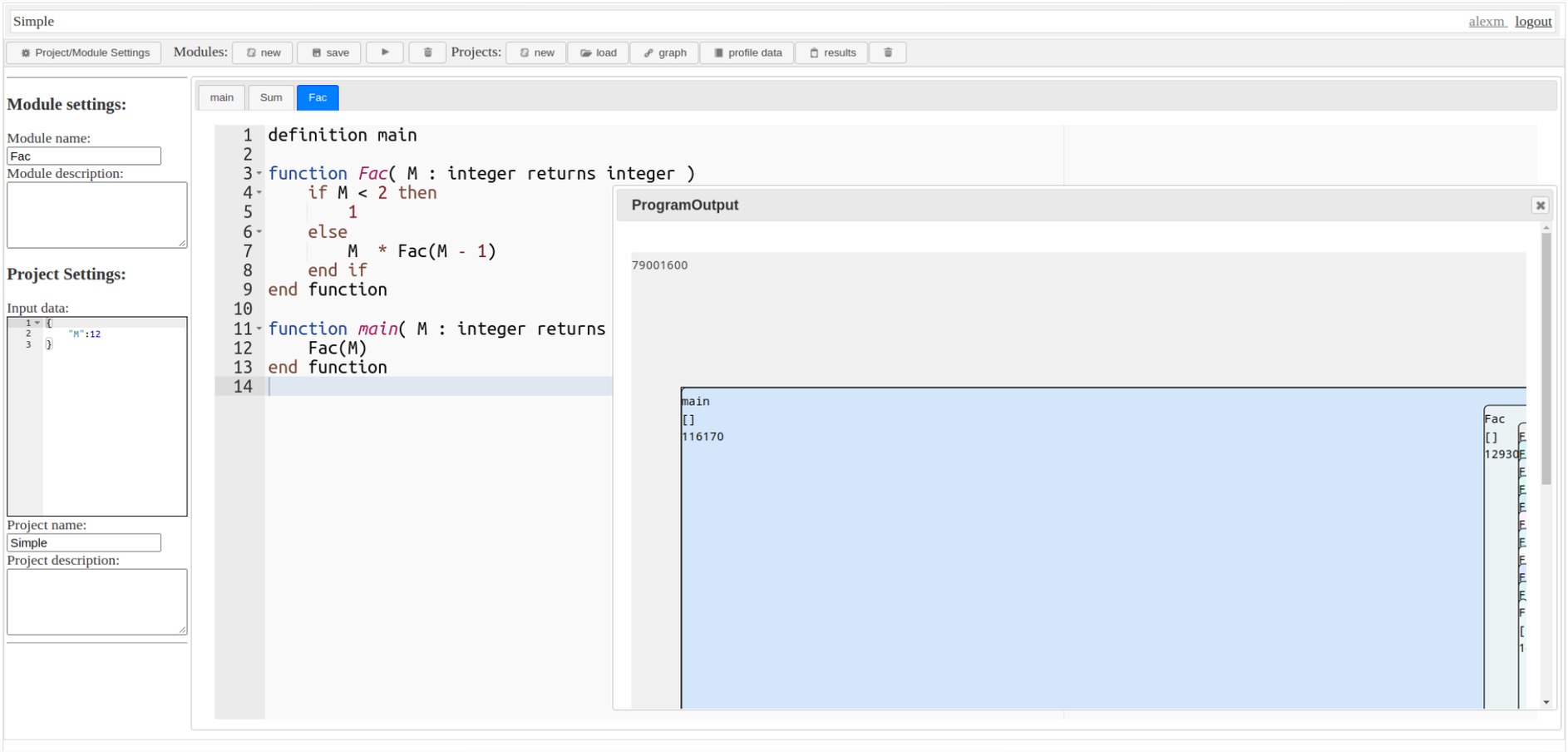
Язык Sisal является результатом сотрудничества Ливерморской национальной лаборатории имени Лоренца, университета штата Колорадо, Манчестерского университета и корпорации DEC. Последняя спецификация языка Sisal версии 2.0 датируется 1991 г. В 1995 г. появилось пользовательское описание языка Sisal 90, не содержащее точных спецификаций языка.

Язык Sisal имеет следующие особенности, облегчающие переход с популярных императивных языков программирования:
приближенный к языку Паскаль синтаксис,
развитую систему типов и явно выделенные циклические выражения.

Язык Sisal имеет следующие основополагающие качества: математическая правильность функций (отсутствие побочных эффектов), прозрачность ссылок имен, задающих значения, а не ячейки памяти, и однократность присваивания.

Система CPPS

Облачная система параллельного программирования CPPS, разрабатываемая в Институте систем информатики СО РАН (ИСИ СО РАН), использует функциональный язык Cloud Sisal для разработки, отладки, верификации и исполнения параллельных программ через веб-браузер. В докладе рассмотрена созданная онлайн среда системы CPPS, которая вместе с созданными компилятором и профилировщиком системы позволяет пользователю на любом устройстве, имеющем выход в Интернет, разрабатывать и исполнять параллельные функциональные программы на языке Cloud Sisal и (в будущем) использовать вычислительную мощность суперычислителя.



Редактор CPPS

Simple alexm logout

Project/Module Settings | Modules: new save | Projects: new load graph profile data results

Module settings:

Module name: Fac

Module description:

Project Settings:

Input data:

```
{
    "M":12
}
```

Project name: Simple

Project description:

main | Sum | Fac

```
definition main

function Fac( M : integer returns integer )
    if M < 2 then
        1
    else
        M  * Fac(M - 1)
    end if
end function

function main( M : integer returns
    Fac(M)
end function
```

ProgramOutput

```
79001600

main
[]
116170
Fac
[]
12930
```

Django

Для редактора использован Python-фреймворк Django.

-Повзолил легко интегрировать имеющиеся средства

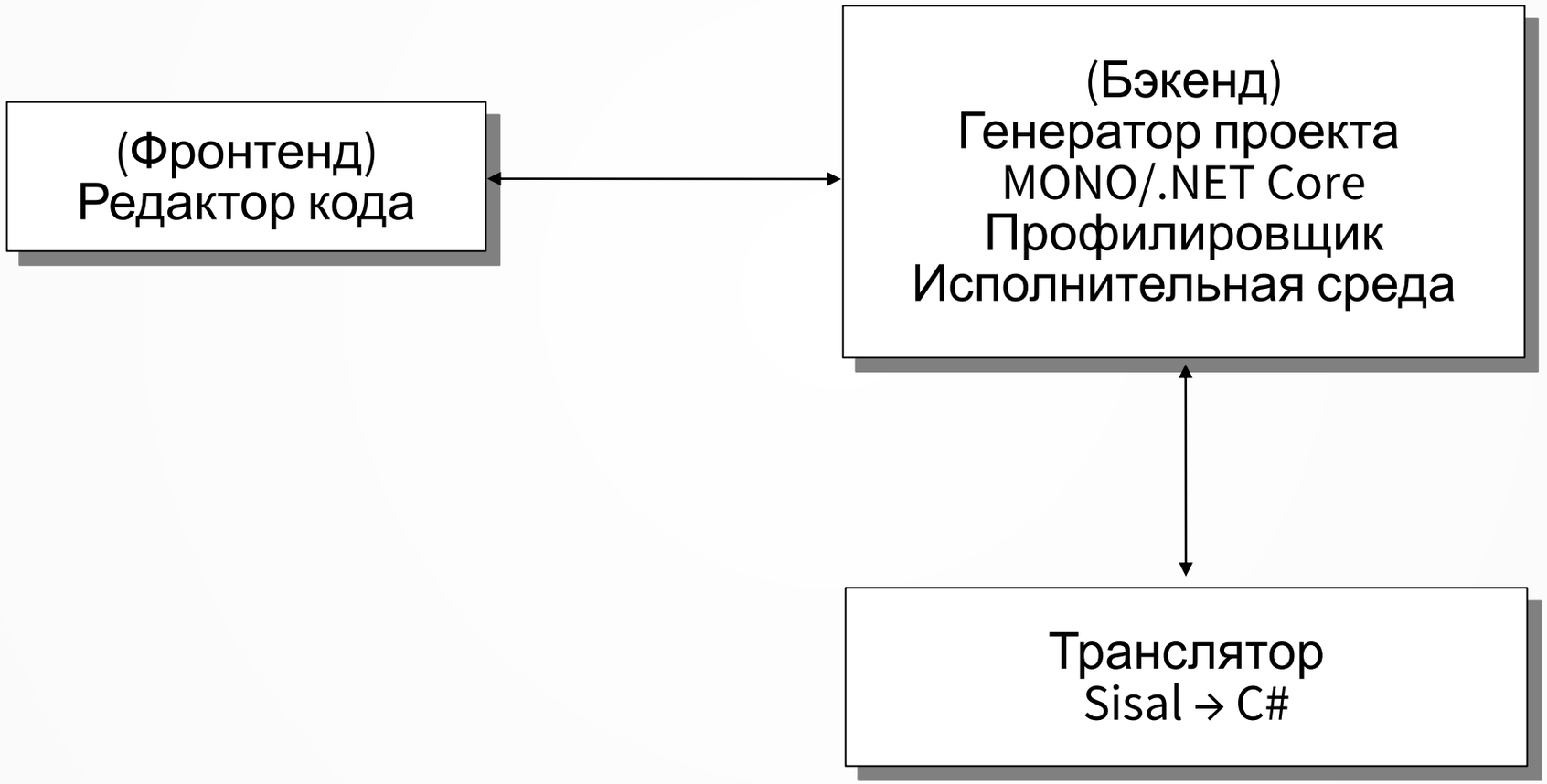
-Использует Python

-Быстр для разработки после первого этапа изучения

-Имеет множество встроенных средств (менеджмент пользователей, модели данных, работа с БД, шаблоны страниц, защита и пр.)

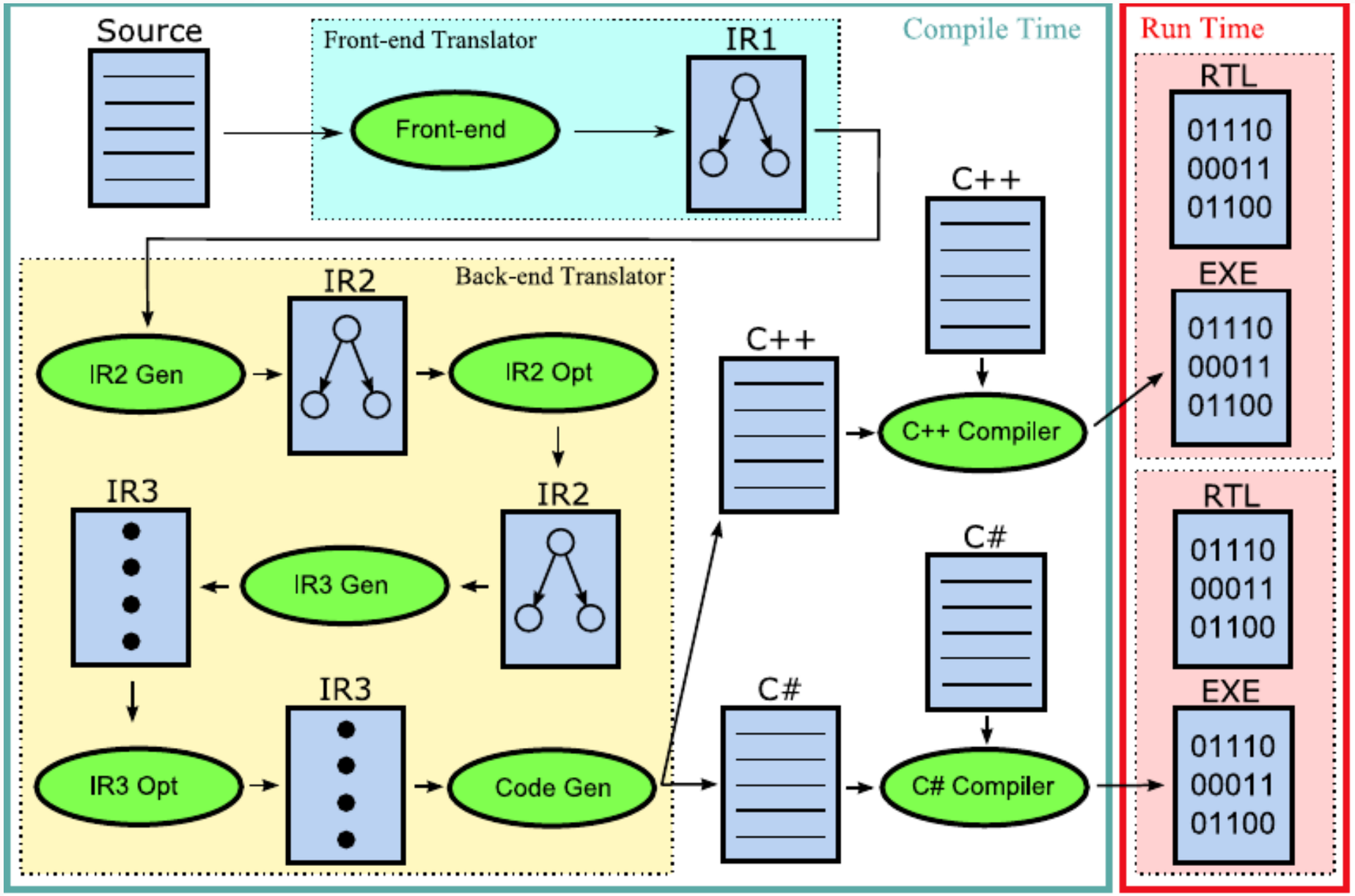
CPPS

Система построена на основе сервисной архитектуры. Блоки общаются между собой посредством HTTP-запросов.



Транслятор Sisal

Транслятор разработан на C++/C#(.NET)



Трансляция

Sisal 3.x

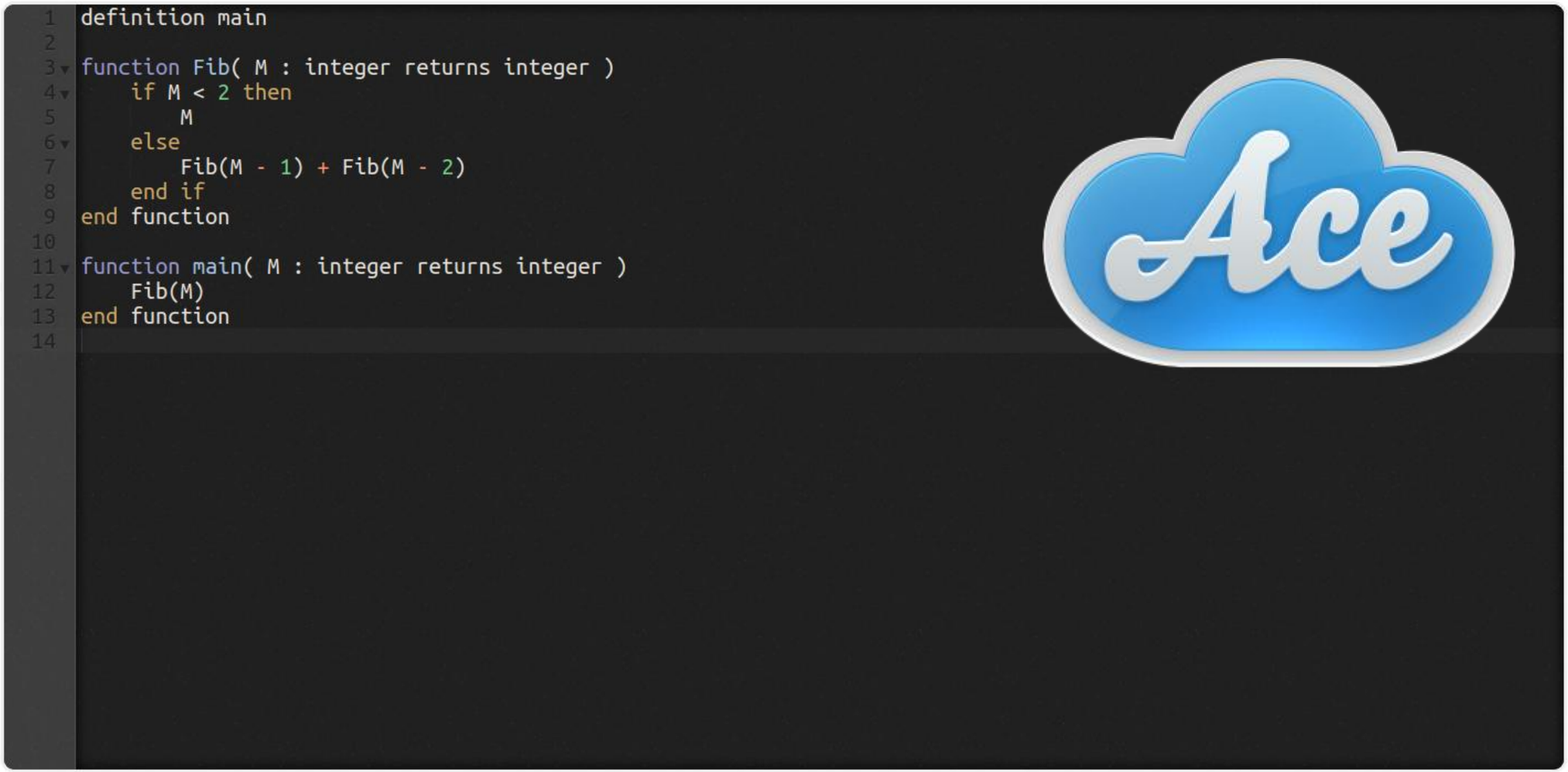
```
definition main

function Fib( M : integer returns integer )
    if M < 2 then
        M
    else
        Fib(M - 1) + Fib(M - 2)
    end if
end function

function main( M : integer returns integer )
    Fib(M)
end function
```

C#

```
// This file is generated by Sisal 3.1 compiler.

using System;
using System.Threading;

namespace Sisal_C_sharp
{

public class SISAL_CODE
{
    public static bool  Fib(CI32 V_1,  out CI32 V_3)
    {
        CBOOL  V_7;
        CI32  V_17;
        CI32  V_21;
        CI32  V_27;
        CI32  V_31;

        V_7 = ( new CBOOL(V_1 < new CI32(0x2)) );
        if ( V_7._Value == true )
        {
            V_3 = V_1;
        }
        else
        {
            V_17 = V_1 - new CI32(0x1);
            V_27 = V_1 - new CI32(0x2);
            Fib(V_17,  out V_21);
            Fib(V_27,  out V_31);
            V_3 = V_21 + V_31;
        }
        return true;
    }
    public static bool  main(CI32 V_33,  out CI32 V_37)
    {

        Fib(V_33,  out V_37);
        return true;
    }
}

} // namespace Sisal_C_sharp
```

ACE (Ajax.org Cloud9 Editor)

Редактор исходного кода с подсветкой синтаксиса, историей правок, поиском, сессиями возможностью создавать свои цветовые схемы, автоотступы, автообнаружение некоторых ошибок и пр. (Доступен по лицензии BSD).



Контейнеризация

Используется LXC

jQuery UI (и jQuery)

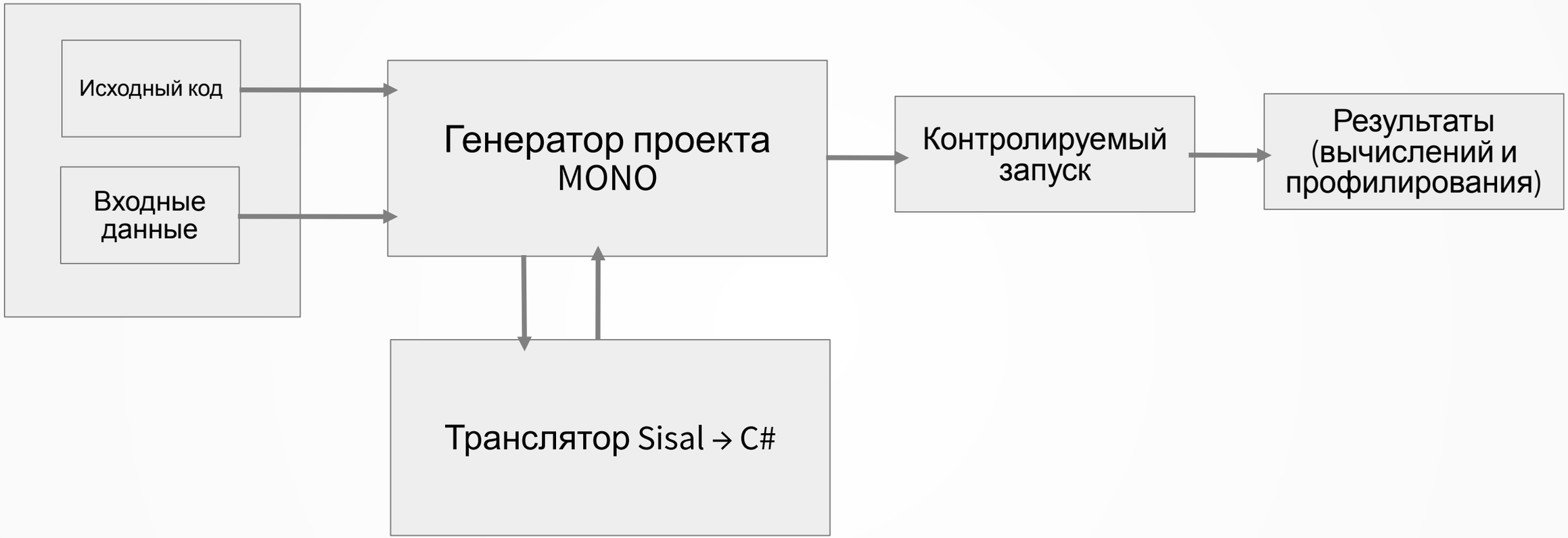
Веб-интерфейс построен с использованием jQuery UI

Сервер генератора проектов и профилирования

Должен решать задачи

- формирования решений (Solution) из получаемых C#-модулей,
- вставки входных данных
- подготовки кода для профилирования,
- запуска итоговой программы,
- выдачи результатов вычислений и профилирования

Генератор проекта



Профилирование

Профилирование — сбор характеристик работы программы, таких как время выполнения отдельных фрагментов (обычно подпрограмм), число верно предсказанных условных переходов, число кэш-промахов и т. д. Инструмент, используемый для анализа работы, называют профилировщиком или профайлером. Обычно выполняется совместно с оптимизацией программы.

```
// This file is generated by Sisal 3.1 compiler.

using System;
using System.Threading;

namespace Sisal_C_sharp
{

public class SISAL_CODE
{
    public static bool Fib(CI32 V_1, out CI32 V_3)
    {
        CBOOL V_7;
        CI32 V_17;
        CI32 V_21;
        CI32 V_27;
        CI32 V_31;

        V_7 = ( new CBOOL(V_1 < new CI32(0x2)) );
        if ( V_7._Value == true )
        {
            V_3 = V_1;
        }
        else
        {
            V_17 = V_1 - new CI32(0x1);
            V_27 = V_1 - new CI32(0x2);
            Fib(V_17, out V_21);
            Fib(V_27, out V_31);
            V_3 = V_21 + V_31;
        }

        return true;
    }
    public static bool main(out int V_33)
    {

        V_33 = 0x1;

        return true;
    }
}

} // namespace Sisal_C_sharp
```

```
// This file is generated by Sisal 3.1 compiler.

using System;
using System.Threading;

namespace Sisal_C_sharp
{

public class SISAL_CODE
{
    struct startup{
        public uint startID;
        public uint parentID;
        public uint startTime;
        public uint endTime;
    }

    public static unsigned int callCounter;

    public static int incCallID()
    {
        return ++callCounter;
    }

    public static void start(String name, uint ID, uint parent)
    {

        history.Add()
    }

    public static void end(uint ID)
    {

    }

    public static bool Fib(CI32 V_1, out CI32 V_3, uint parentCallID)
    {
        int newCallID = incCallID();
        functionStart("Fib", newCallID, parentCallID);
        CBOOL V_7;
        CI32 V_17;
        CI32 V_21;
        CI32 V_27;
        CI32 V_31;

        V_7 = ( new CBOOL(V_1 < new CI32(0x2)) );
        if ( V_7._Value == true )
        {
            V_3 = V_1;
        }
        else
        {
            V_17 = V_1 - new CI32(0x1);
            V_27 = V_1 - new CI32(0x2);
            Fib(V_17, out V_21, newCallID);
            Fib(V_27, out V_31, newCallID);
            V_3 = V_21 + V_31;
        }
        functionEnd(newCallID);
        return true;
    }
    public static bool main(out int V_33, uint parentCallID)
    {
        int newCallID = incCallID();
        functionStart("main", newCallID, parentCallID);

        V_33 = 0x1;
        functionEnd(newCallID);
        return true;
    }
}

} // namespace Sisal_C_sharp
```

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Профилировка

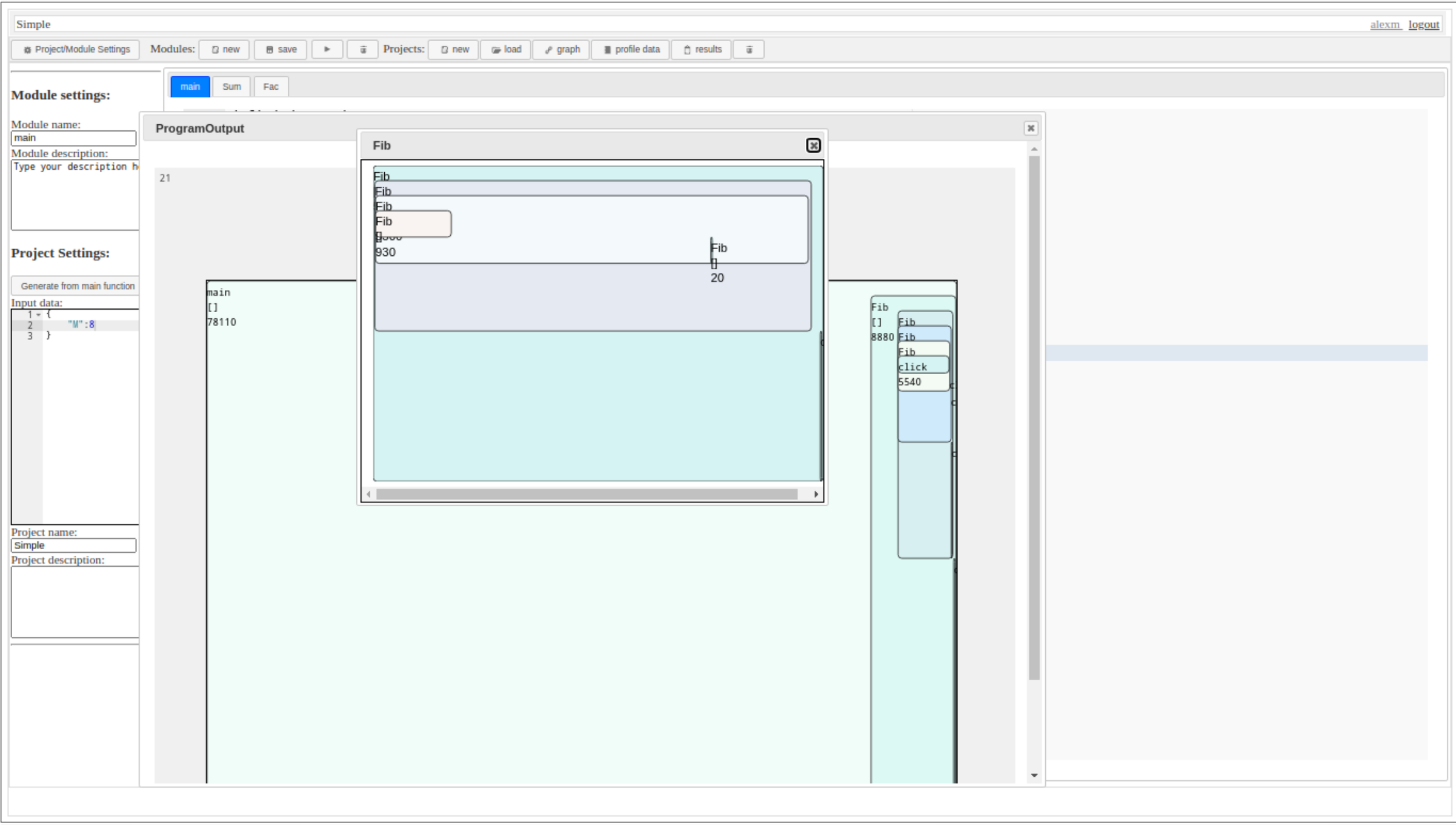
```
1  // This file is generated by Sisal 3.1 compiler.
2
3  using System;
4  using System.Threading;
5
6  namespace Sisal_C_sharp
7  {
8
9  public class SISAL_CODE
10 {
11     public static bool  Fib(CI32 V_1,  out CI32 V_3)
12     {
13         CBOOL  V_7;
14         CI32  V_17;
15         CI32  V_21;
16         CI32  V_27;
17         CI32  V_31;
18
19         V_7 = ( new CBOOL(V_1 < new CI32(0x2)) );
20         if ( V_7._Value == true )
21         {
22             V_3 = V_1;
23         }
24         else
25         {
26             V_17 = V_1 - new CI32(0x1);
27             V_27 = V_1 - new CI32(0x2);
28             Fib(V_17,  out V_21);
29             Fib(V_27,  out V_31);
30             V_3 = V_21 + V_31;
31         }
32         return true;
33     }
34     public static bool  main(CI32 V_33,  out CI32 V_37)
35     {
36
37         Fib(V_33,  out V_37);
38         return true;
39     }
40 }
41
42 } // namespace Sisal_C_sharp
```

```
// This file is generated by Sisal 3.1 compiler.
using System.Collections.Generic;
using System;
using System.Threading;
using System.Web.Script.Serialization;

namespace ConsoleApp1
{
    public class SISAL_CODE
    {
        public struct startup{
            public uint startID;
            public uint parentID;
            public Int64 startTime;
            public Int64 endTime;
            public String name;
            public uint thread;
            public List<String> args;
        }
        public static void init()
        {
            history = new  List<startup>();
            programStart = DateTime.Now.Ticks;
        }
        static SISAL_CODE(){
            init();
        }
        public static bool save_data(){
            var serializer = new JavaScriptSerializer();

            var json = serializer.Serialize(history);


            Console.WriteLine("Finished.");
            System.IO.File.WriteAllText ("output.json", json);
            return true;
        }

        public static List<startup> history;

        public static uint callCounter;
        public static Int64 programStart;
        public static uint incCallID(){
            return ++callCounter;
        }

        public static startup functionStart(String name, List<String> args, uint ID, uint parentID)
        {
            DateTime currentDate = DateTime.Now;
            startup newStartup = new startup();
            newStartup.startTime = currentDate.Ticks - programStart;
            newStartup.startID = ID;
            newStartup.parentID = parentID;
            newStartup.args = args;
            newStartup.name = name;
            return newStartup;
        }
        public static startup functionEnd(startup record)
        {
            record.endTime = DateTime.Now.Ticks - programStart;
            return record;
        }
        public static bool Fib (CI32 V_1,  out CI32 V_3, uint parentCallID)
        {
            List <String> args = new List<String>();
            args.Add("M   = " + V_1._Value.ToString());
            uint newCallID = incCallID();
            startup startRecord =  functionStart("Fib", args, newCallID, parentCallID);
            CBOOL  V_7;
            CI32  V_17;
            CI32  V_21;
            CI32  V_27;
            CI32  V_31;

            V_7 = ( new CBOOL(V_1 < new CI32(0x2)) );
            if ( V_7._Value == true )
            {
                V_3 = V_1;
            }
            else
            {
                V_17 = V_1 - new CI32(0x1);
                V_27 = V_1 - new CI32(0x2);
                Fib(V_17,  out V_21, newCallID);
                Fib(V_27,  out V_31, newCallID);
                V_3 = V_21 + V_31;
            }
            startRecord = functionEnd(startRecord);
            history.Add(startRecord);
            return true;
        }
        public static bool main (CI32 V_33,  out CI32 V_37, uint parentCallID)
        {
            List <String> args = new List<String>();
            args.Add("M   = " + V_33._Value.ToString());
            uint newCallID = incCallID();
            startup startRecord =  functionStart("main", args, newCallID, parentCallID);
            Fib(V_33,  out V_37, newCallID);
            startRecord = functionEnd(startRecord);
            history.Add(startRecord);
            return true;
        }

    }
} // namespace Sisal_C_sharp
```

```
// This file is generated by Sisal 3.1 compiler.
using System.Collections.Generic;
using System;
using System.Threading;
using System.Web.Script.Serialization;

namespace ConsoleApp1
{
    public class SISAL_CODE
    {
        public struct startup{
            public uint startID;
            public uint parentID;
            public Int64 startTime;
            public Int64 endTime;
            public String name;
            public uint thread;
            public List<String> args;
        }
        public static void init()
        {
            history = new  List<startup>();
            programStart = DateTime.Now.Ticks;
        }
        static SISAL_CODE(){
            init();
        }
        public static bool save_data(){
            var serializer = new JavaScriptSerializer();

            var json = serializer.Serialize(history);


            Console.WriteLine("Finished.");
            System.IO.File.WriteAllText ("output.json", json);
            return true;
        }

        public static List<startup> history;

        public static uint callCounter;
        public static Int64 programStart;
        public static uint incCallID(){
            return ++callCounter;
        }

        public static startup functionStart(String name, List<String> args, uint ID, uint parentID)
        {
            DateTime currentDate = DateTime.Now;
            startup newStartup = new startup();
            newStartup.startTime = currentDate.Ticks - programStart;
            newStartup.startID = ID;
            newStartup.parentID = parentID;
            newStartup.args = args;
            newStartup.name = name;
            return newStartup;
        }
        public static startup functionEnd(startup record)
        {
            record.endTime = DateTime.Now.Ticks - programStart;
            return record;
        }
        public static bool Fib (CI32 V_1,  out CI32 V_3, uint parentCallID)
        {
            List <String> args = new List<String>();
            args.Add("M  = " + V_1._Value.ToString());
            uint newCallID = incCallID();
            startup startRecord =  functionStart("Fib", args, newCallID, parentCallID);
            CBOOL  V_7;
            CI32  V_17;
            CI32  V_21;
            CI32  V_27;
            CI32  V_31;

            V_7 = ( new CBOOL(V_1 < new CI32(0x2)) );
            if ( V_7._Value == true )
            {
                V_3 = V_1;
            }
            else
            {
                V_17 = V_1 - new CI32(0x1);
                V_27 = V_1 - new CI32(0x2);
                Fib(V_17,  out V_21, newCallID);
                Fib(V_27,  out V_31, newCallID);
                V_3 = V_21 + V_31;
            }
            startRecord = functionEnd(startRecord);
            history.Add(startRecord);
            return true;
        }
        public static bool main (CI32 V_33,  out CI32 V_37, uint parentCallID)
        {
            List <String> args = new List<String>();
            args.Add("M  = " + V_33._Value.ToString());
            uint newCallID = incCallID();
            startup startRecord =  functionStart("main", args, newCallID, parentCallID);
            Fib(V_33,  out V_37, newCallID);
            startRecord = functionEnd(startRecord);
            history.Add(startRecord);
            return true;
        }

    }
} // namespace Sisal_C_sharp
```

Данные профилирования


```
{
"start": "134010",
"end": "134350",
"name": "Fib",
"callID": "164",
"parentID": "154",
"arg": "m = 3;"
},
```


Simple
alexm logout
Project/Module Settings
Modules: new save
Projects: new load graph profile data results
Module settings:
Module name:
main
Module description:
Type your description h
Project Settings:
Generate from main function
Input data:
1 {
2 "M":8
3 }
Project name:
Simple
Project description:
main
Sum
Fac
ProgramOutput
21
main
[]
78110
Fib
Fib
Fib
Fib
Fib
930
Fib
[]
20
Fib
[]
8880
Fib
Fib
Fib
click
5540

Ссылки и литература

http://pco.iis.nsk.su/CPPS
http://cpps.iis.nsk.su/

1. Касьянов В. Н., Касьянова Е. В. Методы и система облачного параллельного программирования // Проблемы оптимизации сложных систем: Материалы XIV Международной Азиатской школы-семинара (20-31 июля 2018 г.). - Алматы, 2018. - Часть 1. - с. 298-307.
2. Касьянов В. Н., Касьянова Е. В. Язык программирования Cloud Sisal. - Новосибирск, 2018.- 45 с. - (Препринт/ РАН, Сиб. отд-ние, ИСИ; N181).
3. Гордеев Д.С. Модель визуализации изменений в графе внутреннего представления программ языка Cloud Sisal // GraphiCon 2018: труды 28-й Междунар. конф. по компьютерной графике и машинному зрению (Томск, 24–27 сент., 2018 г.). - Томск: Нац. исслед. Том. политех. ун-т., 2018. - с. 115-118.
4. Касьянов В.Н., Золотухин Т.А. Визуализация графовых представлений потоковых программ // GraphiCon 2018: труды 28-й Междунар. конф. по компьютерной графике и машинному зрению Томск, 24-27 сент., 2018 г.). - Томск: Нац. исслед. Том. политех. ун-т., 2018. - с. 138-141.
5. Kasyanov V., Kasyanova E., Zolotuhin T. Visualization of graph presentations of data-flow programs // WSEAS Transactions on Information Science and Applications. - 2018. - Vol. 15. - pp. 140-146.
6. Pankratov S. Automated test generation for optimizing compilers with OpenMP support // MATEC Web Conf. - 2018. - Vol. 210.

?*