

ТЕХНОЛОГИИ В ОБРАЗОВАНИИ
УНИВЕРСИТЕТ
МИКРОЭЛЕКТРОНИКА
ИННОВАЦИИ
КАТАЛИТИЧЕСКИЕ
МАТЕРИАЛЫ
ДИЗАЙН **ТОЧКА**
ЛЕКАРСТВ СБОРКИ
НАУЧНАЯ
ЛАБОРАТОРИЯ
ГЕОХИМИЯ
ИНЖИНИРИНГ
ГЕОФИЗИКА
ГИБРИДНЫЕ
МАТЕРИАЛЫ **НГУ**
ЭНЕРГОСБЕРЕЖЕНИЕ
ВЫСОКИЕ
ЭНЕРГИИ
БИОТЕХНОЛОГИИ
МОДЕЛИРОВАНИЕ
НАНОТЕХНОЛОГИИ
СЕМИОТИКА
НАУКА **МОЗГ** **АРКТИКА**
КОГНИТИВНЫЕ ТЕХНОЛОГИИ
МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ
ЭЛЕМЕНТАРНЫЕ
ЧАСТИЦЫ
ГЕОЛОГИЯ
КВАНТОВЫЕ
ТЕХНОЛОГИИ
БИОЛОГИЯ
ТЕМНАЯ
МАТЕРИЯ
ФОТОНИКА
БИОМЕДИЦИНА
ПРИКЛАДНЫЕ
ИССЛЕДОВАНИЯ
РАЗВИТИЕ
АСТРОНОМИЯ
ГЛОБАЛЬНЫЕ ПРИОРИТЕТЫ
АСТРОФИЗИКА
БИОИНФОРМАТИКА
ЛАЗЕРНАЯ
ФИЗИКА
АРХЕОЛОГИЯ
ЭКОНОМИКА
ЗНАНИЙ
СОТРУДНИЧЕСТВО
ИТ
DEEP
LEARNING
ИЗУЧЕНИЕ

N* Новосибирский
государственный
университет
***НАСТОЯЩАЯ НАУКА**



Разработка алгоритмов генерации и создание генератора LuNA- программ

Автор: Синюков В.К.

Научный руководитель: Перепелкин В.А.

Всероссийская летняя XLI молодежная Школа-
конференция по параллельному программированию
Новосибирск, 2023

* Проблема и Цель

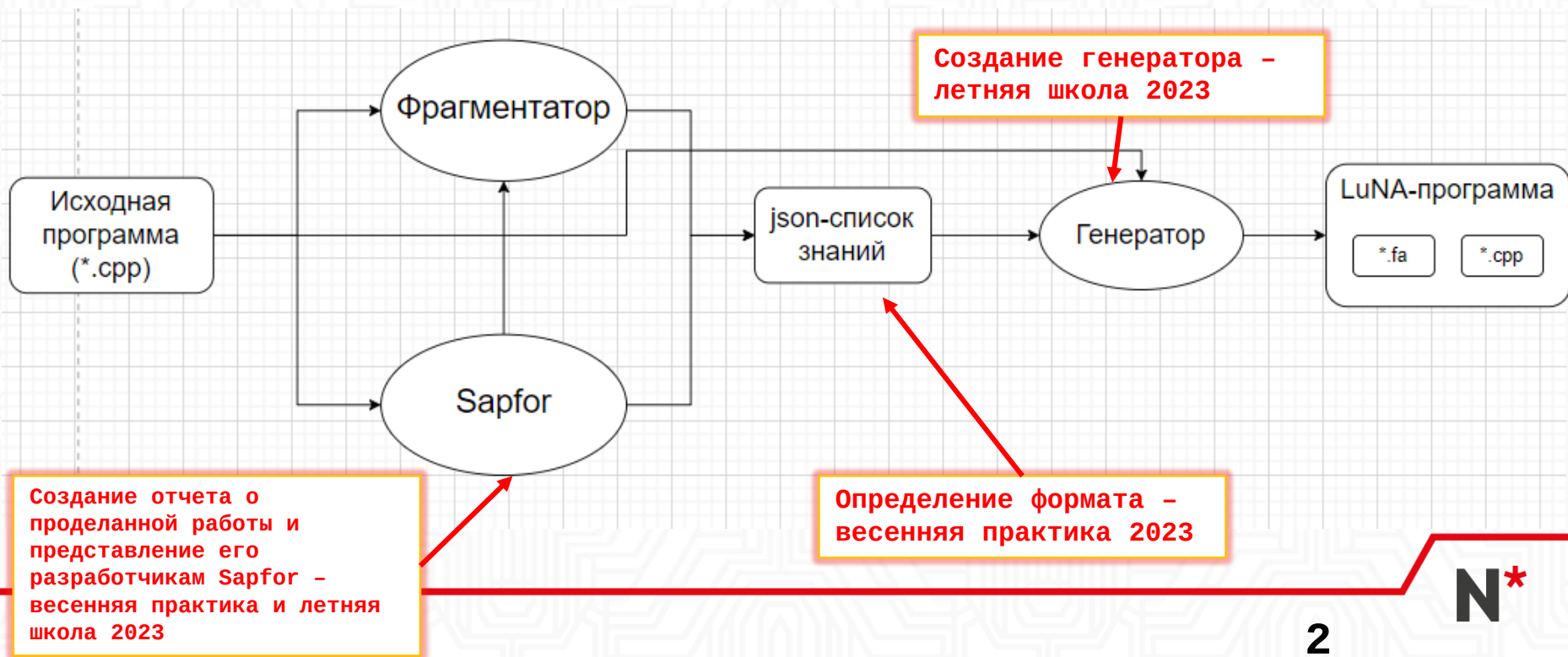
Проблема: автоматическая генерация параллельных программ из последовательных - это в общем случае задача алгоритмически труднорешаемая.

Задача автоматической генерации программ изначально в LuNA не ставилась, тем не менее существует достаточно много готовых наработок и решений по автоматической генерации параллельных программ в конкретных предметных областях (например, Sapfor), которые можно использовать для генерации LuNA-программ.

Объемлющая цель: создание автоматического генератора LuNA-программ из последовательных программ численного моделирования, написанных на языках C/C++/Fortran.

Частная цель (текущая): создание такого автоматического генератора, который сможет сгенерировать данную программу, реализующую численный метод Якоби решения системы дифференциальных уравнений.

*Схема генерации и задачи на школу



* Примеры

```
1
2 int main()
3 {
4     int a, b, c;
5     a = 5;
6     b = 6;
7     c = a * b;
8     return 0;
9 }
10
```



JSON

Генератор



```
2 import init_c(int, int, name);
3 import init_a(name);
4 import init_b(name);
5
6 sub main()
7 {
8     df a, b, c;
9
10    init_a(a);
11    init_b(b);
12    init_c(a, b, c);
13
14 }
15
```

```
15 extern "C"
16 {
17     void init_c(int a, int b, OutputDF& c)
18     {
19         c = a * b;
20     }
21     void init_a(OutputDF& a)
22     {
23         a = 5;
24     }
25     void init_b(OutputDF& b)
26     {
27         b = 6;
28     }
29 }
```

```

1  #define N 1000000
2
3  ∨ int main()
4  {
5      int A[N];
6      for (int i = 0; i < N; ++i)
7          A[i] = i;
8      return 0;
9  }

```

```

10
1  import calc_A(int, int, name);
2
3
4  sub main(int nf)
5  ∨ {
6
7      df A;
8
9  ∨ for f0=0..nf-1 {
10         calc_A(f0, nf, A[f0]);
11     }
12
13 }
14

```

```

1  #define N 1000000
2
3  #include "ucenv.h"
4
5  ∨ int get_size(int f, int nf, int init_size)
6  {
7      return (f * init_size / nf - (f - 1) * init_size / nf);
8  }
9
10 ∨ int get_start(int f, int nf, int init_size)
11 {
12     return ((f - 1) * init_size / nf);
13 }
14
15
16 ∨ extern "C"
17 {
18     void calc_A(int f0, int nf0, OutputDF& A)
19     {
20         int *A_arr = A.create<int>(N);
21         int start0 = get_start(f0, nf0, N);
22         int size0 = get_size(f0, nf0, N);
23         for (int i = 0; i < size0; ++i)
24             {
25                 A_arr[i] = (i + start0);
26             }
27     }
28 }
29 }

```

```

1  #define N 1000
2  #define M 1000
3
4  int main()
5  {
6      int A[N * M];
7      for (int j = 0; j < M; ++j)
8          for (int i = 0; i < N; ++i)
9              A[j * N + i] = j * N + i;
10     return 0;
11 }
12
13 import calc_A(int, int, int, int, name);
14
15 sub main(int nf)
16 {
17     df A;
18
19     for f0=0..nf-1 {
20         for f1=0..nf-1 {
21             calc_A(f0, f1, nf, nf, A[f1][f0]);
22         }
23     }
24 }

```

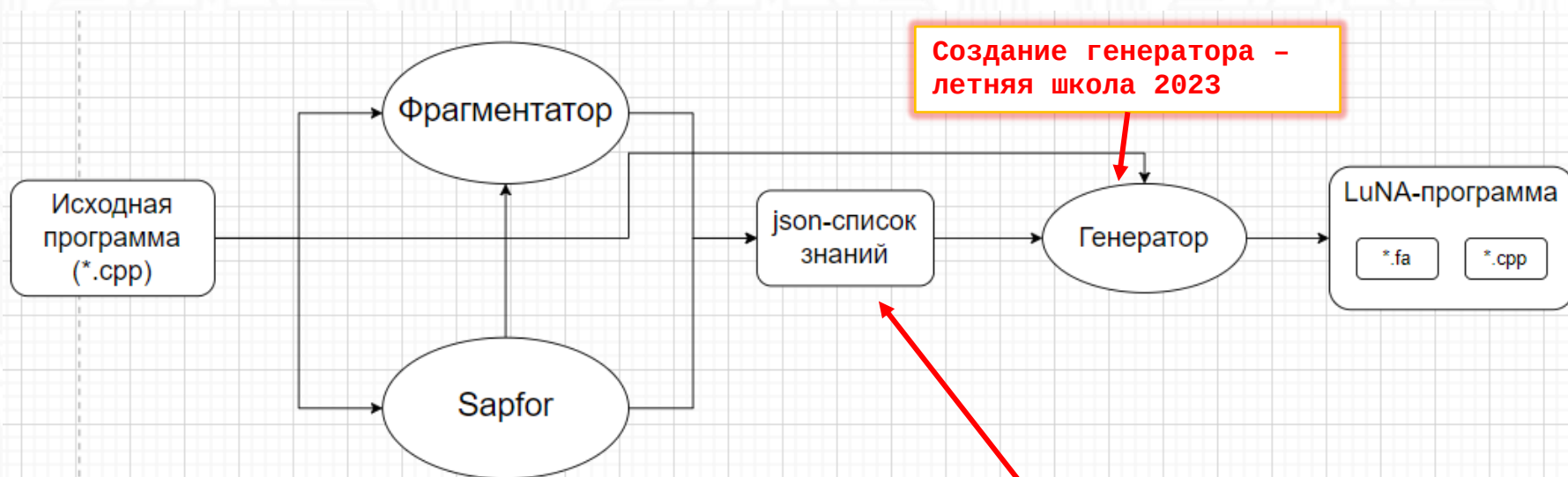
```

1  #define N 1000
2  #define M 1000
3
4  #include "ucenv.h"
5
6  int get_size(int f, int nf, int init_size)
7  {
8      return (f * init_size / nf - (f - 1) * init_size / nf);
9  }
10
11 int get_start(int f, int nf, int init_size)
12 {
13     return ((f - 1) * init_size / nf);
14 }
15
16
17 extern "C"
18 {
19     void calc_A(int f0, int f1, int nf0, int nf1, OutputDF& A)
20     {
21         int *A_arr = A.create<int>(N * M);
22         int start0 = get_start(f0, nf0, M);
23         int size0 = get_size(f0, nf0, M);
24         int start1 = get_start(f1, nf1, N);
25         int size1 = get_size(f1, nf1, N);
26         for (int j = 0; j < size0; ++j)
27         {
28             for (int i = 0; i < size1; ++i)
29             {
30                 A_arr[j * N + i] = (j + start0) * N + (i + start1);
31             }
32         }
33     }
34 }
35
36

```

*

*Схема генерации и задачи на школу

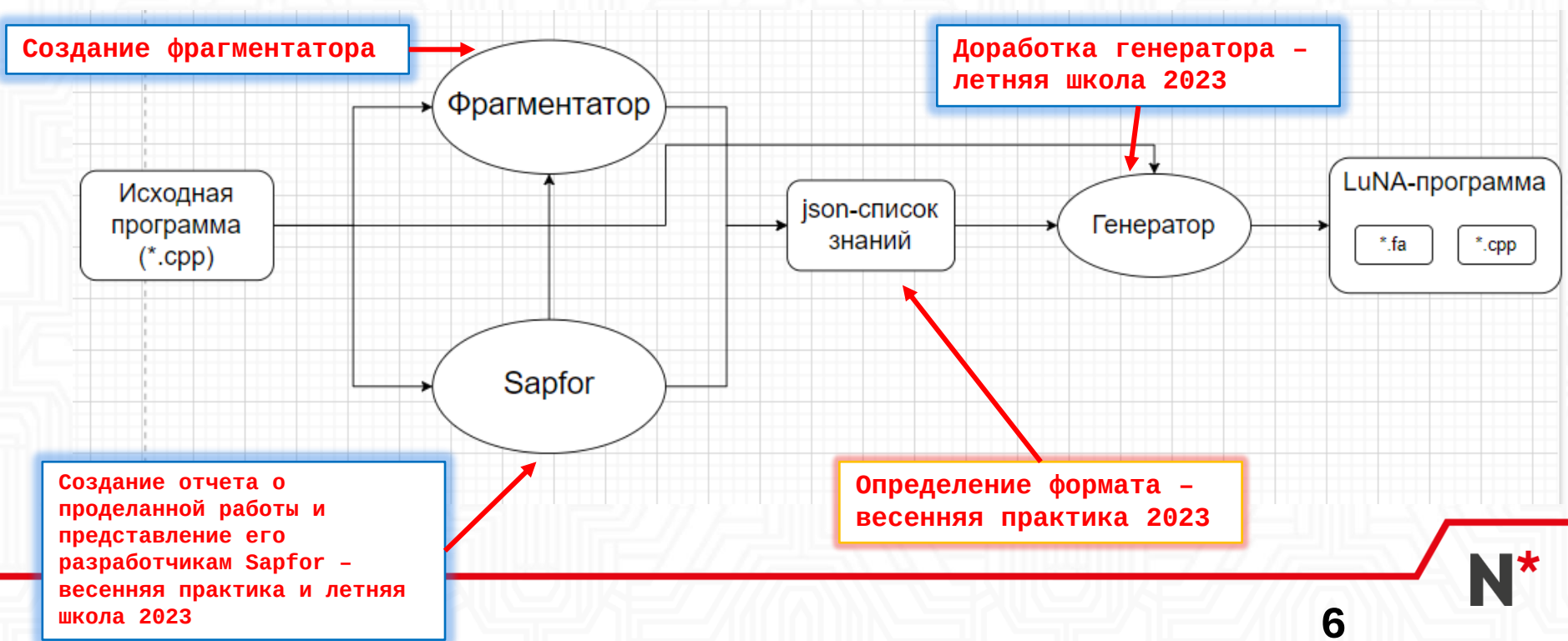


Создание отчета о проделанной работе и представление его разработчикам Sapfor – весенняя практика и летняя школа 2023

Определение формата – весенняя практика 2023

Создание генератора – летняя школа 2023

*Схема генерации и дальнейшая работа



* Заключение

Итоги проделанной работы:

- ✓ Были разработаны и реализованы
 - алгоритмы генерации примитивных LuNA-программ
 - алгоритмы генерации LuNA-программ с разбиением циклов на фрагменты вычислений
- ✓ Были сделаны продвижения в создании отчета для разработчиков системы Sapfor

* Заключение

Дальнейшие планы:

- Завершение отчета и его представление разработчикам Sapfor
- Разработка фрагментатора
- Доработка генератора



N* Novosibirsk
State
University
***THE REAL SCIENCE**

RUSSIA, 630090, NOVOSIBIRSK, PIROGOVA STR., 2



[nsuniversity.official](https://www.facebook.com/nsuniversity.official)



[nsu24](https://www.blogger.com/nsu24)



[@nsuniversity](https://www.instagram.com/nsuniversity)

WWW.NSU.RU/N/