ARAGELI

«**AR**ithmetic, **A**lgebra, **GE**ometry, **L**inear and **I**nteger linear programming)

Презентацию подготовила:
Тимошенко Валентина
группа 13604/1

Характеристики

«**Arageli**» (ARithmetic, Algebra, GEometry, Linear and Integer linear programming)

Тип: математическая библиотека
Разработчик: группа разработчиков под научным руководством Николая Юрьевича Золотых; в разное время в библиотеку сделали свой вклад Е. Агафонов, М. Алексеев, А. Бадер, С. Лялин и А. Сомсиков.

Написана на: C++

Операционная система: Кроссплатформенное программное обеспечение

Тестовая версия: 2.2.9.412 prealpha (17 июля 2010)

Лицензия: GNU GPL

Сайт: arageli.org

Алгоритмы и моделируемые объекты

Библиотека поддерживает моделирование следующих алгебраических систем (включая базовое множество применимых операций к каждой из них):

- целые числа неограниченной длины;
- рациональные числа;
- рациональные функции;
- полиномы от одной переменной;
- векторы;
- матрицы;
- конечные поля;
- кольца вычетов (модулярная арифметика);
- алгебраические числа;
- интервальная арифметика.

$$y(x) = -\frac{x^3}{3} + \frac{x^2}{2} + \frac{17}{6}x$$

$$A = \begin{pmatrix} 1 & -1 & 2 \\ 1 & 3 & 1 \\ 0 & 2 & 3 \end{pmatrix}$$

Алгоритмический состав (некоторые алгоритмы)

- алгоритм Евклида для нахождения НОД и коэффициентов Безу;
- бинарный алгоритм Евклида;
- алгоритм Гаусса для нахождения ступенчатой формы матрицы (в том числе и целочисленный аналог);
- разложение целого числа на простые сомножители ρ-методом Полларда;

```
int Rho-Поллард (int N)
{
  int x = random(1, N-2);
  int y = 1; int i = 0; int stage = 2;
  while (Н.О.Д.(N, abs(x - y)) == 1)
  {
    if (i == stage ){
      y = x;
      stage = stage*2;
    }
    x = (x*x + 1) (mod N);
    i = i + 1;
  }
  return Н.О.Д(N, abs(x-y));
}
```

1. НОД(0, n) = n; НОД(m, 0) = m; НОД(m, m) = m;
2. НОД(1, n) = 1; НОД(m, 1) = 1;
3. Если m, n чётные, то НОД(m, n) = 2*НОД(m/2, n/2);
4. Если m чётное, n нечётное, то НОД(m, n) = НОД(m/2, n);
5. Если n чётное, m нечётное, то НОД(m, n) = НОД(m, n/2);
6. Если m, n нечётные и n > m, то НОД(m, n) = НОД((n-m)/2, m);
7. Если m, n нечётные и n < m, то НОД(m, n) = НОД((m-n)/2, n);

Алгоритм Евклида

• решение задачи линейного программирования симплекс методом (прямой и двойственный, в строчечной и столбцовой форме;
• определение простоты числа различными методами.

Все алгоритмы реализованы как шаблоны.

Определение простоты числа методом Соловея-Штрассена

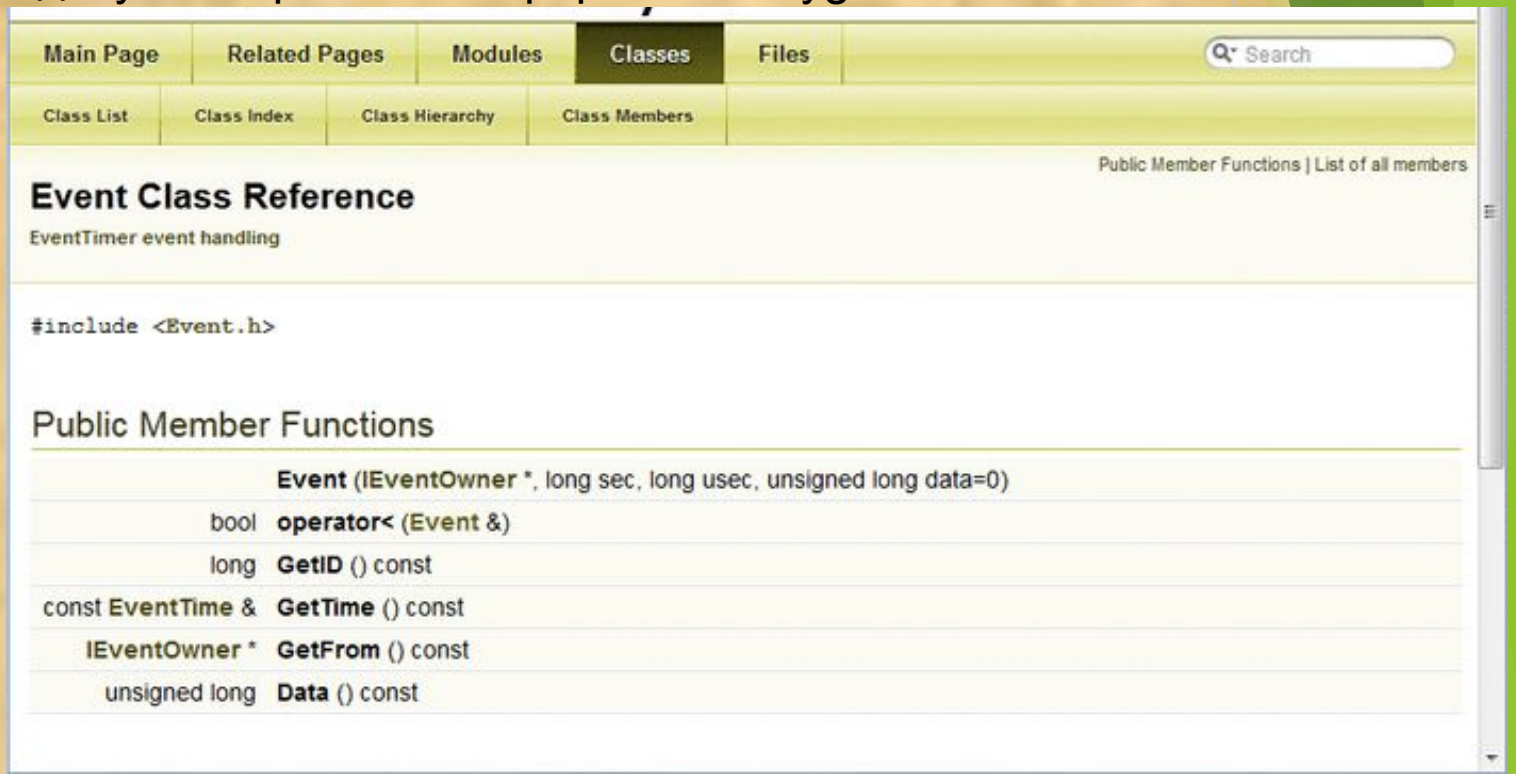
Вход: $n > 2$, тестируемое нечётное натуральное число;
k, параметр, определяющий точность теста.
Выход: *составное*, означает, что n точно составное;
вероятно простое, означает, что n вероятно является простым.

for $i = 1, 2, \ldots, k$:
a = случайное целое от 2 до $n - 1$, включительно;
если НОД(a, n) > 1, **тогда:**
вывести, что n – составное, и **остановиться.**
если $a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \pmod{n}$, **тогда:**
вывести, что n – *составное*, и **остановиться.**

вывести, что n – *простое с вероятностью* $1 - 2^{-k}$, и **остановиться.**

Компонентность и многоуровневость

- базовые алгебраические структуры;
- набор алгебраических алгоритмов;
- параметризуемость;
- дополнительная информация о типе;
- смешанные вычисления;
- подсистема контроля исключительных ситуаций;
- подсистема ввода/вывода;
- контролёры алгоритмов;
- внутреннее документирование в формате doxygen.



Параметризуемые структуры и алгоритмы

Некоторые из основных моделей:

1. big int – класс, моделирующий целое число с произвольным числом бит

2. rational – класс, моделирующий как рациональные числа, так и рациональные функции

   rational <rational>> - так конструируется рациональная функция

3. vector – вектор с элементами типа T

4. sparse polynom – полином от одной переменной с коэффициентами типа T

   sparse polynom <matrix <rational>> - полином с матричными коэффициентами

Примеры рациональных функций:

$$\frac{x^2+1}{x-1},\qquad \frac{ax+b}{cx+d},\qquad \frac{x}{x^2-4},$$

$$\frac{ax+b}{cx^2+dx+f}.$$

Плюсы и минусы статической параметризации*

+ быстродействие
+отсутствие накладных расходов (затраты времени, памяти)
+своевременная диагностика ошибок

-большой объем бинарного кода
-для обеспечения гибкости и полной совместимости типов друг с другом требуется сложная и аккуратная реализация классов

***Параметризация** - очень ценный инструмент, позволяющий за короткое время **проверить различные конструкции и конструктивные схемы и избежать принципиальных ошибок**. Можно изменить несколько параметров и посмотреть, как будет вести себя при этом программа в целом. Параметрическая связь позволяет управлять свойствами или характеристиками объекта посредством управляющего параметра, что значительно усиливает возможности моделирования.

Контролёры алгоритмов

Для некоторых алгоритмов требуется организовать более жёсткий контроль. Для этого существуют специальным образом оформленные объекты-контролёры, которые передаются в “сложную” функцию как дополнительный параметр. Требования к интерфейсу для типа этого объекта полностью определяются функцией; тип должен поддерживать все методы, которые вызывает для него целевая функция. Вызов любой из них является сигналом от алгоритма контролёру.

Контролёр функции — это механизм передачи дополнительной информации в исполняемую функцию или из нее, в процессе её работы. Тип контролёра всегда задаётся как параметр шаблона.

Основным критерием, по которому определяется, делать ли некоторую функцию контролируемой или неконтролируемой, является объём промежуточных результатов и предполагаемая продолжительность работы функции.

Функцию, сконструированную так, что она принимает контролёра, будем называть **контролируемой функцией**.

Для любой контролируемой функции имеется как минимум две её перегруженные версии: с явно указываемым объектом контролёра и без него. Вторая версия просто вызывает первую с контролёром по умолчанию (idler-контролёр). Т. е. для контролируемой шаблонной функции f у нас есть следующие определения:

Класс ctrl::f idler — это класс контролёра, который ничего не делает (в библиотеке все такие классы находятся в пространстве имён Arageli::ctrl).

```
template </*параметры шаблона без контроля */. typename Ctrler>
void f
(
      /* параметры функции без контроля */,
     Ctrler ctrler // контролёр
)
 {   /* код с вызовами методов контролёра */ }
template  </*параметры шаблона без контроля*/>
inline void f
(
     /* параметры функции без контроля */
 )
{   f(/* аргументы функции без контроля */, ctrl::f idler()); }
```

Обработка ошибок

Система контроля и обработки исключительных ситуаций в библиотеке представлены двумя механизмами.

Во-первых, это система классов исключений и описание ситуаций, когда они генерируются. Все функции и методы классов сконструированы таким образом, что они не теряют целостности, когда через них проходит исключение.

Во-вторых, механизм обработки ошибок включает в себя отладочные проверки. Это система предикатов в различных частях функций и методов классов библиотеки вместе с кодом, который проверяет их значение.

Реализованная система проверок может работать в одном из двух режимов.
В первом режиме происходит аварийная остановка, с выводом в стандартный поток сообщения об ошибке.
Во втором режиме проверочным кодом генерируется исключение с исчерпывающей информацией о не сработавшем предикате.

Ввод и вывод

Существует три основных формата ввода/вывода.

- Первый способ ввода/вывода — обратимый; в нём все объекты записываются в виде списков. Для группировки используются скобки, а для отделения элементов при перечислении — запятая.

- Второй формат служит только для вывода — формат с выравниванием при выводе на моноширинную консоль. Особенно удобен при выводе структурированной табличной информации, например, матриц и векторов.

- Третий формат вывода — это вывод на языке LATEX. Такой способ вывода обычно встречается только в больших математических пакетах, подобных MATLAB и Maple.

Решение квадратного уравнения \(ax^2+bx+c=0\):
\begin{equation}\label{eq:solv}
x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a} \end{equation}
Можно сослаться на уравнение~\eqref{eq:solv}.

Примеры программы №1

Найти матрицу, обратную данной:

$$A = \begin{pmatrix} 21 & 3 & 4 \\ 3335 & 6 & 75 \\ 81 & 9 & 10 \end{pmatrix}$$

Программа:

```
#include <iostream>
#include <iomanip>
#include <arageli/arageli.hpp>
using namespace Arageli;
int main ()
{
    matrix<rational<> > A = "((21, 3, 4), (3335, 6, 75), (81, 9, 10))";
    std::cout << "A = \n";
    output_aligned(std::cout, A, "|| ", " ||", "  ");
    std::cout << "\ninversion of A = \n";
    output_aligned(std::cout, inverse(A), "|| ", " ||", "  ");
    std::cout
    << "\n\nthe inversion is valid: "
    << std::boolalpha << (A*inverse(A)).is_unit();
    std::cin.get();
}
```

Вывод программы:

```
A =
||  21   3  4  ||
|| 3335  6  75 ||
||  81   9  10 ||

inversion of A =
||  -205/7792     1/3896     67/7792   ||
|| -27275/23376  -19/3896  11765/23376 ||
||  9843/7792     9/3896    -3293/7792  ||

the inversion is valid: true
```

Данная программа находится в стадии разработки, вывод представлен теоретический

Пример программы № 2

Представление некоторых максимальных значений фундаментальных типов как значений big_int.

Программа:

```
#include <iostream>
#include <limits>
#include <arageli/big_int.hpp>
using namespace Arageli;
int main ()
{
  big_int
  maxint = std::numeric_limits<int>::max(),
  maxfloat = std::numeric_limits<float>::max(),
  maxdouble = std::numeric_limits<double>::max();
  std::cout
  << "maximum int value is " << maxint
  << "\nmaximum float value is " << maxfloat
  << "\nmaximum double value is " << maxdouble;
  std::cin.get();
}
```

Вывод программы:

maximum int value is 2147483647

maximum float value is
340282346638528859811704183484516925440

maximum double value is
179769313486231570814527423731704356798070567525844996598917476803157260780028538760589558632766878171540458953514382464234321326889464182768467546703537516986049910576551282076245490090389328944075868508455133942304583236903222948165808559332123348274797826204144723168738177180919299881250404026184124858368

Данная программа находится в стадии разработки, вывод представлен теоретический

Список использованной литературы

https://ru.wikipedia.org/wiki/Arageli

http://www.arageli.org/

http://www.arageli.org/documentation

http://arageli.org/doc/ArageliUsersGuide.pdf

http://arageli.org/doc/Arageli_overview.pdf

http://arageli.org/doc/Simple_first_examples.doc

Спасибо за внимание!