МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра Математического и компьютерного моделирования

Применение ООП аллокатора для дерева Жане

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы

направление 01.03.02 — Прикладная математика и информатика

механико-математического факультета

Тураева Артема Фирузовича

Научный руководитель
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2020

Введение. На практике часто возникает необходимость решать системы нелинейных алгебраических уравнений с целочисленными коэффициентами. Часто решение разбивается на два этапа: на первом этапе система приводится к некоторому стандартному виду, называемому базисом Гребнера, на втором этапе решается система, полученная на первом этапе. Теоретическая сложность алгоритма пополнения исходной системы до базиса Гребнера, с использованием алгоритма Бухбергера, достаточно высока.

Значительный интерес вызывает альтернативный алгоритм вычисления нередуцированного стандартного базиса, получивший название инволютивного. Инволютивный алгоритм вычисления стандартного базиса пришел в коммутативную алгебру из дифференциальной алгебры и получил большой успех. Полученный инволютивный базис может быть использован при решении систем нелинейных алгебраических уравнений и для их исследования аналогично авторедуцированному базису Гребнера. Основное отличие инволютивного алгоритма вычисления стандартного базиса от алгоритма Бухбергера состоит в модифицированном алгоритме вычисления нормальной формы, который ограничивает набор возможных редукций.

Целью работы является построение алгоритмов, необходимых для вычисления и построения дерева Жане, которое в дальнейшем может быть применено для пополнение систем нелинейных уравнений до базиса Гребнера.

Структура бакалаврской работы. Данная работа включает 7 разделов:

1) Первый раздел посвящен описанию языка C++, разобраны его высокоуровневые возможности, тем самым была показана, эффективность данного языка при решении поставленной задачи.
2) Во втором разделе представлено взаимодействие языка с внутренними компонентами ЭВМ. Было показано, как происходит использование оперативной памяти во время работы программ, написанных на C++.
3) В третьем разделе изучается устройство аллокатора памяти, для более эффективной её организации.
4) В четвёртом разделе изучается средство для сборки и тестировании проектов, написанных на C++.

5) Пятый раздел вводит основные определения, необходимые для понимания базиса Грёбнера. Также производится построение базиса Грёбнера для системы уравнений с использованием алгоритма Бухбергера.
6) Шестой раздел вводит такое понятие, как инволютивное деление, которое является ключевым фактором при построении дерева Жане.
7) В седьмом разделе приводятся определения и алгоритмы, которые нужны для построения дерева Жане и в итоге был получен алгоритм для построения минимального базиса Жане 2.

Основное содержание работы.

Язык C++ — это универсальный высокоуровневый объектно-ориентированный язык программирования. ООП включает в себя такие понятия, как инкапсуляция, наследование, инкапсуляция и полиморфизм.

Формальное определение ООП.

Объектно-ориентированное программирование - это метод реализации, при котором программы организованы, как объединённые коллекции объектов, каждый из которых представляет собой экземпляр некоторого класса, и все классы являются членами объединённой иерархии посредством отношения наследования.

Определения базовых принципов:

Инкапсуляция — механизм языка, позволяющий объединить данные и методы, работающие с этими данными в единый объект, и скрыть детали реализации от других компонент программы.

Наследование в языке появилось в связи с необходимостью переиспользовать и расширить базовый функционал класса предка в дочернем классе. Также этот принцип способствует чистому написанию коду в соответствии с DRY (don't repeat yourself).

Полиморфизм вытекает из наследования, по сути это переопределение базовой функциональности (методов или операторов) родительского класса в классе потомке.

Полезные библиотеки в языке C++:

1) <cassert> - стандартная библиотека, позволяющая производить статическую отладку программы, путём внутреннего контроля допустимых значений при помощи функции assert.

2) <cstddef> - библиотека, содержащая в себе тип данных size_t, который используется для хранения размера объектов любых типов.
3) <cstdlib> - стандартная библиотека, которая содержит в себе функции динамического управления памятью.

Низкоуровневое управление памятью. Указатели — это переменные, в которых хранится адрес области оперативной памяти в шестнадцатеричном формате (обозначаются префиксом *). Ссылки — это тип переменной, который работает как псевдоним другого объекта или значения (обозначаются префиксом &).

Для управления памятью в языке C и C++ используются специальные функции: malloc выделяет ячейку памяти, а функция free её освобождает, оператор new выделяет ячейку, а оператор delete освобождает память. Функция realloc(pointer, size) — выделяет дополнительный участок памяти для хранения объектов.

Организация памяти в программах на языке C++.

Память, которую используют программы, состоит из нескольких частей (сегментов):

1) Сегмент кода (текстовый сегмент), в нём находиться скомпилированная программа.
2) Сегмент bss (неинициализированный сегмент данных), где хранятся глобальные и статические переменные, инициализированные (*null/nullptr*) нулём.
3) Сегмент данных (сегмент инициализированных данных), в нём хранятся инициализированные глобальные и статические переменные.
4) Куча, откуда выделяются динамические переменные.
5) Стек вызовов, где хранятся параметры функции, локальные переменные и другая информация, связанная с функциями.

Стек (stack) — это область оперативной памяти (стековый фрейм), которая создаётся для каждого потока выполняемой программы

Куча (heap) — это хранилище памяти, также расположенное в оперативной памяти компьютера. В отличие от стека, объекты помещённые в кучу хранятся на протяжении выполнения всей программы и могут быть очищены принудительно, используя специальные команды.

Устройство аллокатора.

Аллокатор в языке C++ - это распределитель памяти представленый в виде специализированного класса, который реализует и инкапсулирует детали распределения и освобождения ресурсов компьютерной памяти.

Инструмент CMake

CMake — это кросс-платформенный инструмент (с открытым исходным кодом) для определения и управления сборками кода, прежде всего написанных на языке C++. Основная идея данного программного обеспечения состоит в том, чтобы иметь единое определение того, как строится проект, - который переводится в конкретные определения построения для любой поддерживаемой платформы.

Также в работе рассматривается установка и настройка конфигурационного файла для Linux OS и осуществляется сборка проекта при помощи команд в bash консоли.

Базис Грёбнера и алгоритм Бухбергера.

Введём понятие кольца и идеала:

Пусть $K[x_1, \ldots, x_n]$ — множество всех многочленов от переменных $x_1, \ldots, x_n$ с коэффициентами в поле K (над полем K). Если на этом множестве определены операции сложения и умножения, то такое множество называют кольцом.

Непустое подмножество I кольца K ($\mathrm{I} \subset \mathrm{K}$) называется идеалом в K ($I \lhd K$), если

- $\forall a, b \in I$, выполняется $(a + b) \in I$.
- $\forall a \in I,\ c \in K$, элемент $ac \in I$.

Базис Грёбнера. Для допустимого упорядочивания $\prec$ конечное множество $G \subset R$ называется базисом Грёбнера идеала $I \lhd R$, если

$$(\forall f \in I)(\exists g \in G)\ [lm(g)|lm(f)].$$

Базис Грёбнера может быть построен с помощью алгоритма Бухбергера. Ключевым понятием в данном алгоритме является редукция полиномов и построение S-полиномов.

S-полиномом для $f, g \in R$ называется комбинация

$$S_{poly}(f,g) = \frac{lcm(lm(f), lm(g))}{lt(f)} f - \frac{lcm(lm(f), lm(g))}{lt(g)} g. \quad (1)$$

Следующий алгоритм(алгоритм Бухбергера) 1 строит базис Грёбнера.

Algorithm 1 $GroebnerBasis(F)$

Вход: $F \subset R$ **Выход:** $G \subset R$, такое что $(G) = (F)$

```
1:  G := ∅
2:  while F ≠ ∅ do
3:      G := AutoReduce(G ∪ F)
4:      F := ∅
5:      M := {(f, g) | (f, g ∈ G)(f ≠ g)}
6:      for all (f, g) ∈ M do
7:          h := NormalForm(S_poly(f, g), G)
8:          if h ≠ 0 then
9:              F := F ∪ {h}
10:         end if
11:     end for
12: end while
13: return G
```

На основе этого алгоритма был рассмотрен пример пополнения системы нелинейных уравнений до базиса Грёбнера.

Инволютивное деление.

Ключевым понятием в алгоритме, созданном Бухбергером, является S-полином. Если некоторым самосогласованным способом запретить деление по некоторым переменным, называемыми немультипликативными, и разрешить по другим, называемыми мультипликативными, то построить S-полином можно используя немультипликативные продолжения одного многочлена и его редукцию по другому с использованием мультипликативных переменных.

Будем говорить, что L-инволютивное деление определено на множестве M, если $\forall$ конечного множества $U \subset M$ и $\forall u \in U$ задан подмоноид $L(u, U)$ моноида M, удовлетворяющий следующим свойствам:

- $\forall u \in L(u,U) \ \wedge \ v \mid u \ \Rightarrow v \in L(u,U)$,
- $\forall u, v \in U \ \wedge \ uL(u,U) \cap vL(v,U) \neq \emptyset \Rightarrow u \in vL(v,U) \ \vee \ v \in uL(u,U)$,
- $\forall v \in U \ \wedge \ v \in uL(u,U) \Rightarrow L(v,U) \subseteq L(u,U)$

- $\forall u \in V \ \wedge \ V \subseteq U \Rightarrow L(u,U) \subseteq L(u,V)$

Элементы $L(u,U)(u \in U)$ называются мультипликативными для u. Если $w \in uL(u,U)$, то u называется L-инволютивным делителем w и обозначается $u|_L w$. В свою очередь, моном w называется L-кратным u.

Быстрый поиск делителя с помощью дерева Жане.

Для инволютивного авторедуцированного множества может существовать только один инволютивный делитель. В этом случае возможно построить деревья поиска с различной сбалансированностью, которые ускоряют поиск инволютивного делителя. Одним из таких деревьев является древо Жане.

Разбиение Жане. Конечное множество U разделим на подмножества, маркируемые числами $d_0, d_1, \ldots, d_i \in \mathbb{Z}_{\geq 0}$:

$$[d_0, d_1, \ldots, d_i] = \{u \in U \mid d_0 = 0, d_1 = deg_1(u), \ldots, d_i = deg_i(u)\}.$$

Независимая переменная x_i считается мультипликативной для $u \in U$, если $u \in [d_0, d_1, \ldots, d_i]$ и

$$deg_i(u) = max\{deg_i(v) \mid v \in [d_0, d_1, \ldots, d_i]\}$$

и немультипликативной в противном случае.

Поиск делителя Жане осуществляется на упорядоченных полиномах, имеющих лексикографический порядок.

Рассмотрим структуру дерева Жане общего вида как множество $JT := \cup\{\nu\}$ внутренних узлов и листьев непустого множества мономов. К каждому элементу ν дерева назначим пять элементов $\nu = [d, u, nd, nv]$ со следующей структурой:

- $deg(\nu) = d$ степень текущей переменной,
- $mon(\nu) = u$ указатель на моном,
- $ndg(\nu) = nd$ указатель на следующий узел по степени,
- $nvr(\nu) = nv$ указатель на следующий узел по переменной.

При отсутствии поддерева назначим значение nil соответствующему указателю. Везде, где это не приводит к недоразумению, отождествим указатели nd и nv с узлами, на которые они указывают, а u с мономом. Корень $root(JT)$ дерева JT имеет номер текущей переменной 1 и $deg(root(JT)) = 0$. При

дальнейшем обходе дерева номер текущей переменной при движении направо увеличивается на 1.

Внутренние узлы и листья дерева JT характеризуются следующими состояниями:

- Внутренний узел: $(nv \neq nil \vee (nd \neq nil \wedge d < deg(nd))) \wedge u = nil$
- Лист: $nv = nil \wedge nd = nil \wedge u \neq nil$

Дополнительно для листа дерева, при номере текущей переменной v и числе переменных n, выполнено

$$d = deg_v(u) \wedge (v = n \vee \sum_{i=v+1}^{n} deg_i(u) = 0). \tag{2}$$

По определению делитель Жане $u \in U$ монома w должен быть лексикографически самым старшим среди всех мономов в U или среди мономов в группе $d_1 = deg_1(w), \ldots, d_i = deg_i(w)$ для $1 \leq i < n$. Следовательно $deg_j(w) \geq deg_j(u)$ при $j > i$. После завершения цикла while на линии 3 возможны три ситуации:

- $(ndg(\nu) = nil) \wedge (nvr(\nu) \neq nil)$. В этом случае нужно продолжить поиск делителя по следующей переменной. Переход на нее сделан в строке 6.
- $ndg(\nu) \neq nil$. Выполнение этого условия означает отсутствия делителя Жане. Действительно, переход в строке 8 закончился бы листом с мономом, который имеет более высокую степень в текущей переменной, чем w. Такой моном не может делить w в обычном и инволютивном смысле.
- $(ndg(\nu) = nil) \wedge (nvr(\nu) = nil)$. Это означает, что ν - лист дерева и его моном является искомым делителем Жане.

Вставка монома в дерево Жане осуществляется посредством рекурсивных вызовов, аналогичных рекурсивному определению структур данных.

Созданное поддерево имеет корень w и заканчивается листом с мономом $w \prod_{j=i}^{n} x_i^{deg_j(u)-deg_j(w)} = u$. В случае когда переменная x_i является немультипликативной для всех мономов в этом поддереве, используется алгоритм продления переменной, который применяется к поддереву Жане.

В итоге алгоритм вставки монома в дерево Жане кроме построения новых поддеревьев для вставки u, вычисляет все немультипликативные продолжения, вызванные этой вставкой. Также вычисляются немультипликативные продолжения монома u относительно мономов содержащихся в листьях дерева Жане.

Алгоритм построения мономиального базиса.

Чтобы построить минимальный базис Жане идеала (U) для конечного непустого множества мономов U, необходимо воспользоваться следующим алгоритмом, который включает в себя все предыдущие алгоритмы инволютивного деления и вставки монома в дерево.

Algorithm 2 $MonomialBasis(U, JT)$

Вход: U — конечное множество мономов; JT — дерево Жане
Выход: $\widetilde{U}$ минимальный базис Жане идеала (U)

1: $V := U$
2: $\widetilde{U} := \emptyset$
3: **while** $V \neq \emptyset$ **do**
4: **Выбрать:** $u \in V \wedge (\nexists w \in V\ u)(w \sqsubset u)$
5: $V := V\ \{u\}$
6: **if** $divisor(JT, u) = nil$ **then**
7: $\widetilde{U} := \widetilde{U} \cup \{u\}$
8: $J - insert(JT, u, V)$
9: **end if**
10: **end while**
11: **return** $\widetilde{U}$

Заключение. В бакалаврской работе были получены алгоритмы, которые необходимы при построении базиса Грёбнера: алгоритм Бухбергера и построение дерева Жане. ООП аллокатор позволяет оптимизировать внутреннюю работу программы с оперативной памятью компьютера. Также в работе представлен пример построения базиса Грёбнера для системы нелинейных алгебраических уравнений, используя S-полиномы, которые являются ключевой особенностью алгоритма Бухбергера.