Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Фундаментальные науки
КАФЕДРА Прикладная математика

ОТЧЕТ ПО ПРАКТИКЕ

Студент Брегадзе Зураб Гелаевич
фамилия, имя, отчество

Группа ФН2-32Б

Тип практики: Ознакомительная практика

Название предприятия: Научно-учебный комплекс «Фундаментальные науки» МГТУ им. Н.Э. Баумана

Студент _____ Брегадзе З.Г.
подпись, дата *фамилия и.о.*

Руководитель практики _____ Попов А.Ю.
подпись, дата *фамилия и.о.*

Оценка _____

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Кафедра «Прикладная математика»

З А Д А Н И Е
на прохождение ознакомительной практики

на предприятии Научно-учебный комплекс «Фундаментальные науки» МГТУ им. Н.Э. Баумана

Студент Брегадзе Зураб Гелаевич
фамилия, имя, отчество

Во время прохождения ознакомительной практики студент должен

1. Изучить на практике основные возможности языка программирования C++ и систем компьютерной алгебры, закрепить знания и умения, полученные в курсах «Введение в информационные технологии», «Информационные технологии профессиональной деятельности».
2. Изучить способы реализации методов решения задачи идентификации положения точки в системе ячеек.
3. Реализовать «метод полос», называемый также «Slab decomposition method».

Дата выдачи задания «26» сентября 2024 г.

Руководитель практики _____ Попов А.Ю.
подпись, дата *фамилия и.о.*

Студент _____ Брегадзе З.Г.
подпись, дата *фамилия и.о.*

Оглавление

Постановка задачи

Плоская многоугольная сетка задана следующим образом: дан список точек, каждая из которых задана двумя своими координатами, и список ячеек, для каждой из которых записаны последовательно против часовой стрелки номера вершин из первого списка точек. При этом гарантируется, что любые две ячейки либо не имеют общих точек, либо имеют одну общую вершину, либо примыкают к одному общему отрезку (т.е. имеют общую сторону и две общих вершины).

Требуется идентифицировать положение, т.е. установить принадлежность конкретной ячейке, системы точек, про которые известно лишь то, что они лежат строго внутри ячеек.

а) решить задачу «перебором»;

б) решить задачу эффективно, используя «метод полос», называемый также «Slab decomposition method».

Структура исходного файла данных:

n	<< количество узлов сетки
x1 y1	<< координаты первого узла сетки
…	
xn yn	<< координаты n-го узла сетки
p	<< количество ячеек сетки
m1	<< количество углов в первой ячейке
k11 k12 … k1(m1)	<< номера узлов, образующих первую ячейку
…	<< количество углов в p-й ячейке
mp	<< номера узлов, образующих p-ю ячейку
kp1 kp2 … kp(m1)	<< количество точек для идентификации
q	<< координаты первой точки
x1 y1	
…	<< координаты q-й точки
xq yq	

Структура файла результата:

q	<< количество точек для идентификации
c1	<< ячейка, в которую попадает первая точка
…	
cq	<< ячейка, в которую попадает q-я точка

Введение

Основной целью ознакомительной практики 3-го семестра, входящей в учебный план подготовки бакалавров по направлению 01.03.04 – Прикладная математика, является знакомство с особенностями осуществления деятельности в рамках выбранного направления подготовки и получение навыков применения теоретических знаний в практической деятельности.

В ранее пройденном курсе «Введение в специальность» произошло общее знакомство с возможными направлениями деятельности специалистов в области прикладной математики и получен опыт оформления работ (реферата), который полезен при оформлении отчета по практике.

В рамках освоенного курса «Введение в информационные технологии» изучены основные возможности языка программирования C++ и сформированы базовые умения в области программирования на C++. Задачей практики является закрепление соответствующих знаний и умений и овладение навыками разработки программ на языке C++, реализующих заданные алгоритмы. Кроме того, практика предполагает формирование умений работы с системами компьютерной алгебры и уяснение различий в принципах построения алгоритмов решения задач при их реализации на языках программирования высокого уровня (к которым относится язык C++) и на языках функционального программирования (реализуемых системами компьютерной алгебры).

1. История рассматриваемой задачи

Локализация точки на плоскости является одной из ключевых задач вычислительной геометрии. Она находит широкое применение в областях, связанных с обработкой геометрической информации, таких как компьютерная графика, географические информационные системы (ГИС), планирование движения и системы автоматизированного проектирования (САПР). В 1976 году Дэвид П. Добкин и Ричард Д. Липтон впервые предложили алгоритм, решающий эту задачу за время $O(\log_2 n)$. Однако данный метод требовал значительных объемов памяти для хранения структур данных — до $O(n^2)$. Позднее Питер Сарнак и Роберт Тарьян смогли оптимизировать этот подход, используя структуру данных, известную как персистентное красно-черное дерево (persistent red-black tree). Это позволило сократить объем необходимой памяти до $O(n)$ [1].

2. Обзор методов решения задачи

2.1. Метод перебора

Метод перебора является одним из самых простых способов решения задачи. Он предполагает проверку каждой тестовой точки на принадлежность каждой из возможных ячеек. Для определения принадлежности точки многоугольнику используется метод бросания луча (ray casting method).

Суть метода заключается в следующем: из заданной точки проводится луч в фиксированном направлении, например, вдоль положительного направления оси *x*. Затем подсчитывается количество пересечений этого луча со сторонами многоугольника. Если оно нечётное, точка находится внутри многоугольника, если чётное – снаружи (рис. 1) [2].

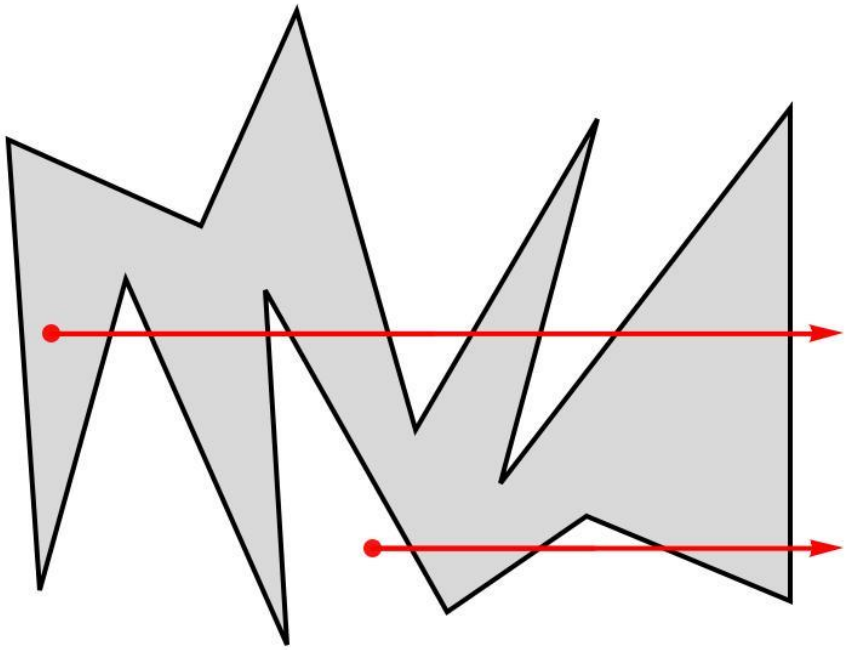

Рис. 1. Метод бросания луча

Главное преимущество этого метода в его универсальности: он применим как для выпуклых, так и для невыпуклых многоугольников.

С точки зрения сложности, алгоритм требует $O(n)$ времени для проверки одной точки в отношении одного n-угольника, так как пересечения вычисляются для каждой из сторон. Для m точек и k многоугольников общая сложность метода составит $O(m \cdot k \cdot n)$, где n – среднее число вершин многоугольника в системе ячеек.

2.2. «Метод полос» («Slab decomposition method»)

Более эффективным алгоритмом решения поставленной задачи является «метод полос». Данный алгоритм, помимо непосредственного определения положения точки в системе ячеек, включает в себя этап предварительной обработки. Он состоит в том, что заданную изначально сетку необходимо разбить на полосы так, чтобы их границы проходили через вершины ячеек. Для определённости будем считать, что разбиение производится вдоль оси абсцисс (слева направо) (рис. 2).

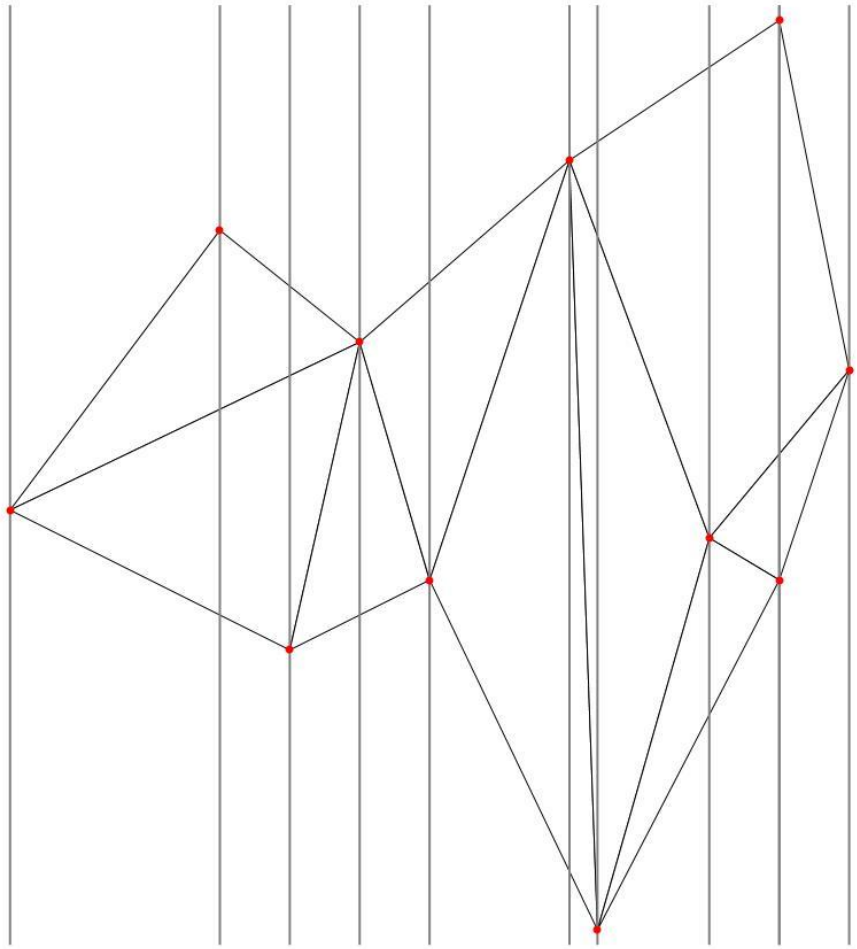

Рис. 2. «Метод полос»

После подготовки плоскости для каждого многоугольника определяются полосы, в которые он попадает. Делается это с помощью рассмотрения пересечения полосы и диапазона абсцисс текущего многоугольника. Если полученное множество не пустое, то в список, соответствующий данной полосе, добавляется номер ячейки, принадлежащей этой полосе.

На этапе определения номера ячейки, которой принадлежит точка, с помощью бинарного поиска находится полоса, в которой находится данная точка, а затем снова с помощью бинарного поиска среди ячеек, которые попали в данную полосу, определяется та из них, которая содержит данную точку. Для того, чтобы бинарный поиск работал корректно, массивы должны быть отсортированы. Для определения полосы эта задача не представляет трудностей, так как все они упорядочены по *x*-координатам вершин. Для осуществления бинарного

поиска по ячейкам в полосе выполняется их сортировка по минимальной ординате ячейки для данной полосы.

В момент определения номера ячейки, которой принадлежит точка, возникает неопределённость в связи с тем, что её ордината может лежать как выше, так и ниже ребра, проходящего через точку с минимальной *y*-координатой ячейки в этой полосе. Для разрешения этой проблемы составляется уравнение прямой:

$$\frac{x_0 - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}, \quad (1)$$

где (x_0, y_0) – координаты тестовой точки, (x_0, y) – координаты точки на ребре, $(x_1, y_1), (x_2, y_2)$ – координаты точек ребра ячейки с минимальной ординатой в данной полосе, принадлежащих непосредственно полосам. Из уравнения (1) выражается *y*, после чего проводится сравнение:

- Если $y_0 > y$, то полученный индекс ячейки в полосе остаётся неизменным;
- Если $y_0 < y$, то полученный индекс ячейки в полосе уменьшается на единицу.

Оценим сложность алгоритма. Введём переменные: *m* – количество уникальных *x*-координат точек сетки, *ncells* – количество ячеек, *p* – среднее количество ячеек в полосе, *k* – среднее число вершин в ячейке, *test* – количество тестовых точек. Основные затраты ресурсов во время работы алгоритма идут на определение ячеек, попадающих в полосу, сортировку ячеек в полосе и на вычисление координат пересечения ячеек с границами полос. Таким образом, сложность алгоритма – $O(ncells \cdot m + (m-1) \cdot p \cdot \log_2 p + test \cdot p \cdot k)$.

2.3. Описание основных функций алгоритмов с использованием псевдокода

Algorithm 1 Метод перебора

Функция *ISINTERSECTING(p, a, b)*

Входные данные: *p* – проверяемая точка, из который выпускается горизонтальный отрезок, *a, b* – точки границ отрезка.

Выходные данные: *bool T* – пересечение выпущенного луча с отрезком *[a; b]*.

1. *low = a, high = b* – нижняя и верхняя точки отрезка соответственно;

2. **if** *low.y > high.y* **then** меняем местами *a* и *b*;

3. **if** *p.y < low.y* **OR** *p.y > high.y* **then** *return False*;

4. **if** *high.y = low.y* **then** *return False;*

5. Определим потенциальную точку пересечения точки *p* с отрезком *[a; b]*:

$$x = low.x + \frac{(p.y - low.y) * (high.x - low.x)}{high.y - low.y};$$

6. *return p.x < x*.

Algorithm 2 Slab decomposition

Функция *YCellInSlab(cell, slab)*

Входные данные: *cell* – ячейка, *slab* – полоса, в которой лежит ячейка.

Выходные данные: *Y* – набор ординат точек ячейки, принадлежащим границам полосы.

1. **for** каждая последовательная пара вершин *(a, b)* из *cell*:

    if вершины не принадлежат полосе **OR** вершины лежат на границах полосы:

        add *(a, b)* to *pair*

    end if;

end for;

3. **for** каждая пара *p = (a, b)* в *pair*:

    for *i:*=1..2:

        $y = \frac{(slab[i] - p.a.x) * (p.b.y - p.a.y)}{(p.b.x - p.a.x)} + p.a.y$;

        add *y* to *Y*;

    end for;

end for;

4. **return** *Y*.

Входные данные: *P* – набор точек, образующих сеть ячеек на плоскости, *PNums* – набор номеров точек, образующих каждую ячейку, *Pts* – набор тестовых точек.

Выходные данные: *CellNums* – номера ячеек, которым принадлежат проверяемые точки.

1. *cellsInSlabs = dict(i→{}), i := 1..Slabs.size()* – словарь, в котором ключ – номер полосы, значение – список из номеров ячеек, лежащих в этой полосе;
2. **for** *i := 1..cellsInSlabs.size():*

   SortBy(*cellsInSlabs[i]*, **MIN**(*YCellInSlab(cells[a], slabs[i])) &);*

   end for;
3. **for** каждая точка *pt* из *Pts*:

   slabInd = BinSearch(setxslab, pt.x);

   cellInd = BinSearch(MinY, pt.y);

   $y = \frac{(pt.x - slabs[slabInd][1]) * (y2 - y1)}{(slabs[slabInd][2] - slabs[slabInd][1])} + y1;$

   if *pt.y > y* **then** *CellNums[i] = cellsInSlabs[slabInd][cellInd*];

   else *CellNums[i] = cellsInSlabs[slabInd][cellInd - 1];*

   end for;
4. **return** *CellNums*.

3. Особенности реализации на языке C++

Основным объектом в задаче является точка, поэтому для её хранения была создана структура *Point*, её полями стали координаты точки. Также была создана структура *Cell*, которая содержит в себе массив из вершин ячейки. Для динамического хранения большого массива данных было использовано такое средство стандартной библиотеки, как *std::vector*. В рамках реализации «метода полос» использовался контейнер *std::map* с целью создания словаря, ключом которого являлся номер полосы, а значением – вектор из номеров ячеек, которые попадают в каждую из полос. Для сортировки использовалась функция *std::sort()* из заголовочного файла <*algorithm*>.

Исходные данные считываются из файла, во время считывания выполняется проверка на корректность введённых данных. В результате работы программы в файл записывается количество проверяемых точек и номера ячеек, в которые они попали.

3.1. Реализация метода перебора

Для реализации метода перебора были реализованы две функции, обе из которых необходимы для реализации метода «бросания лучей». Логическая функция *isIntersecting* проверяет, пересекается ли горизонтальный луч со стороной многоугольника. Логическая функция *ray_cast* позволяет узнать, принадлежит ли точка многоугольнику. Во время работы в цикле *for* создаётся переменная *flag*, изначальное значение которой 0. Она предназначена для того, чтобы отслеживать, попадает ли точка в какую-либо ячейку или нет. Если точка окажется в ячейке, значение переменной *flag* изменится на 1, и программа выйдет из цикла, отвечающего за проход по всем ячейкам. Если же значение переменной *flag* останется равным 0, это будет означать, что точка не попала ни в одну ячейку, и в таком случае в вектор результата добавляется -1.

3.2. Реализация «метода полос»

Для реализации метода полос с помощью функции *xslab* заполняется вектор *setxslab*, предназначенный для хранения *x*-координат всех точек. Также создаётся вектор из векторов *slabs*, который заполняется с помощью функции *slabsBoundaries* и хранит координаты левой и правой границы полосы в каждом своём векторе. В векторе *rangeTable* хранятся начальные и конечные абсциссы каждой ячейки, которые позже позволяют понять, в какие полосы попал многоугольник и заполнить соответствующий массив *cellsRange*. Затем заполняется словарь *cellsInSlabs*, в котором ключи – номера полос, а значения – векторы с номерами ячеек, которые принадлежат данной полосе. После производится сортировка ячеек в полосах по их минимальной *y*-координате в этой полосе.

После предварительной обработки данных начинается этап поиска положения точки. Для каждой точки вычисляется номер полосы, которой она принадлежит, а затем в данной полосе определяется ячейка, в которой находится эта точка. Все эти действия выполняются с помощью бинарного поиска. Стоит отметить, что в случае, когда искомая точка по *y* находится выше, чем минимальная *y*-координата данной ячейки в полосе, но сама точка при этом не принадлежит указанной ячейке, полученный индекс для массива из ячеек в полосе уменьшается на единицу.

4. Особенности реализации в Wolfram Mathematica

4.1. Реализация метода перебора

Система компьютерной алгебры Wolfram Mathematica представляет широкий спектр возможностей для решения огромного класса задач с помощью встроенных средств математического пакета. Не менее значимыми являются и средства визуализации данной системы. Так, например, с помощью функции *Graphics*, были отрисованы системы ячеек с проверяемыми точками, что позволило наглядно представить задачу.

В случае метода перебора исходная задача сводится к задаче определения принадлежности точки многоугольнику. Для этого была использована встроенная функция *RegionMember*, что позволило нам не прибегать к написанию дополнительных блоков кода.

4.2. Реализация «метода полос» («Slab decomposition method»)

В рамках данного алгоритма были созданы две пользовательские функции: *search* и *YCellInSlab*. Функция *search* реализует бинарный поиск, принимая на вход отсортированный массив и одну из координат точки (абсциссу или ординату). Функция *YCellInSlab* определяет *y*-координаты ячейки при пересечении с заданной полосой, принимая в качестве аргументов ячейку и границы полосы по *x*-координате.

Для хранения списка номеров ячеек, соответствующих каждой полосе, используется ассоциативный массив *cellsInSlabs*.

При реализации алгоритма активно использовались встроенные инструменты Wolfram Mathematica, такие как *Table* (для создания и хранения данных), а также *Sort* и *SortBy* (для сортировки списков, включая сортировку с использованием лямбда-функций).

5. Результаты тестовых примеров

Пример 1.

Исходные данные:

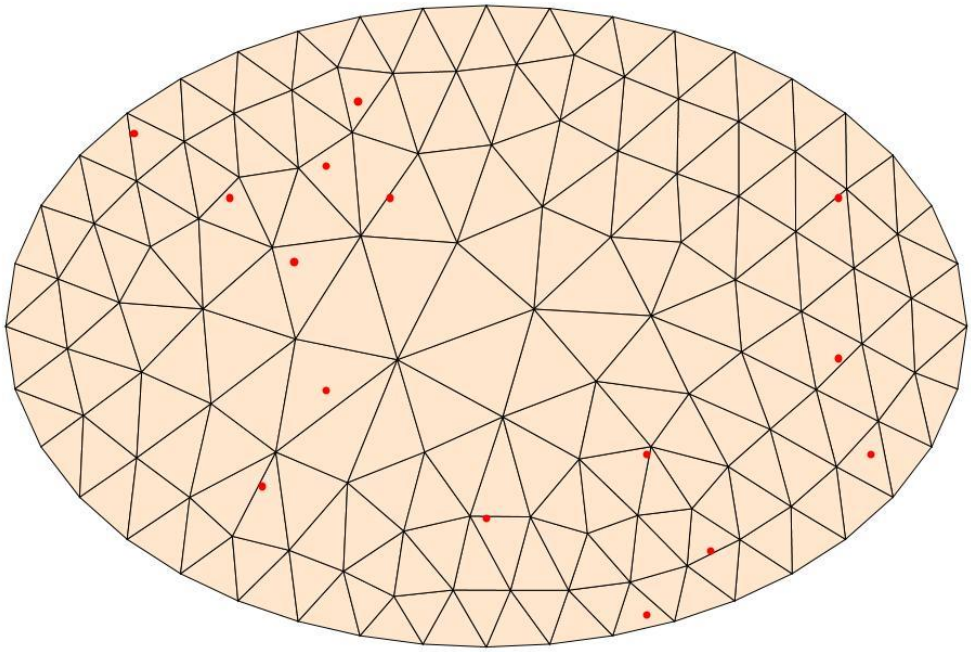

Рис. 3. Сеть ячеек для примера №1

Пример 2.

Исходные данные:

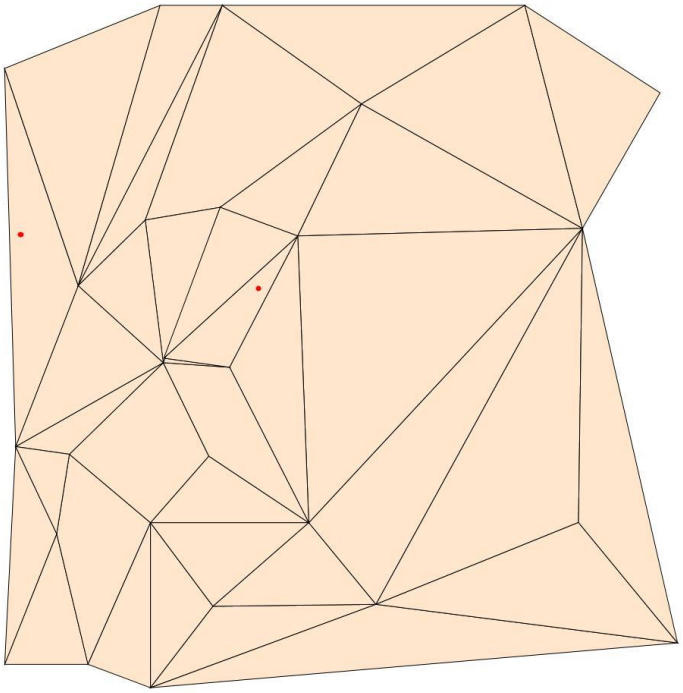

Рис. 4. Элемент сети ячеек для примера №2

Таблица. Сравнение результатов измерений времени

	Перебор на WM, с	Перебор на C++, с	Slab Decomposition на WM, с	Slab Decomposition на C++, с
Сеть 1	0,967	0,00006	0,229	0,001
Сеть 2	395,649	0,00071	37,988	0,056

Из таблицы видно, что в системе компьютерной алгебры Wolfram Mathematica алгоритм полного перебора работает медленнее, чем Slab Decomposition, в то время как на C++ происходит расхождение результатов измерений с теоретическими данными.

Заключение

В ходе выполнения работы были решены все поставленные задачи. Были изучены на практике основные возможности языка программирования C++ и систем компьютерной алгебры, полученные в курсах «Введение в информационные технологии», «Информационные технологии профессиональной деятельности». Были изучены такие способы реализации методов решения задачи идентификации положения точки в системе ячеек, как полный перебор и «метод полос», называемый также «Slab decomposition».

Список использованных источников

1. Point location // Wikipedia. The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Point_location (дата обращения: 26.10.2024).
2. Задача о принадлежности точки многоугольнику // Википедия. Свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Задача_о_принадлежности_точки_многоугольнику (дата обращения: 19.10.2024).
3. Прата С. Язык программирования C++. Лекции и упражнения. М.: Диалектика-Вильямс. 2020. 1248 с.