# Введение

## Цель лабораторной работы

Изучение и реализация асинхронного взаимодействия потоков на примере конвейерной обработки данных.

## Задачи лабораторной работы

* изучить асинхронное взаимодействие на примере конвейерной обработки данных;
* спроектировать систему конвейерных вычислений;
* реализовать систему конвейерных вычислений;
* протестировать реализованную систему;
* подготовить отчёт по проведенной работе.

# Аналитическая часть

В данном разделе описаны принцип и идея конвейерной обработки данных, а также алгоритм Брезенхема с действительными коэффициентами.

## Конвейерная обработка данных

Конвейерный принцип обработки данных подразумевает, что в каждый момент времени процессор работает над различными стадиями выполнения нескольких команд, причем на выполнение каждой стадии выделяются отдельные аппаратные ресурсы. Такая обработка оптимизирует использование ресурсов для заданного набора процессов, каждый из которых применяет эти ресурсы заранее предусмотренным способом.

Идея конвейерной обработки данных заключается в параллельном выполнении нескольких инструкций процессора. Сложные инструкции процессора представляются в виде последовательности более простых стадий. Вместо выполнения инструкций последовательно (ожидания завершения конца одной инструкции и перехода к следующей), следующая инструкция может выполняться через несколько стадий выполнения первой инструкции. Это позволяет управляющим цепям процессора получать инструкции со скоростью самой медленной стадии обработки, однако при этом намного быстрее, чем при выполнении эксклюзивной полной обработки каждой инструкции от начала до конца.

## Алгоритм Брезенхема с действительными коэффициентами

Работа алгоритма Брезенхема основывается на использовании понятия ошибка. Ошибкой здесь называется расстояние между действительным положением отрезка и ближайшим пикселем сетки растра, который аппроксимирует отрезок на очередном шаге.

На каждом шаге вычисляется величина ошибки и в зависимости от полученного значения выбирается пиксель, ближе расположенный к идеальному отрезку. Поскольку при реализации алгоритма на ЭВМ удобнее анализировать не само значение ошибки, а ее знак, то истинное значение ошибки смещается на -0,5.

Поскольку на первом шаге высвечивается пиксел с начальными координатами, то для него ошибка равняется 0, поэтому задаваемое предварительно значение этой ошибки

Фактически является ошибкой для следующего шага.

В общем алгоритме Брезенхема большее по модулю из приращений принимается равным шагу растра, то есть единице, причем знак приращения совпадает со знаком разности конечной и начальной координат отрезка:

Где *xe, ye –* координаты конца отрезка; *xs, ys –* координаты начала отрезка, а sign - кусочно-постоянная функция действительного аргумента.

Значение другой координаты идеального отрезка для следующего шага определяется как , поскольку приращение ординаты совпадает с величиной одного катета прямоугольного треугольника, а другой катет равен шагу сетки растра, то есть единице.

Ошибка на очередном шаге вычисляется как

В зависимости от полученного значения ошибки выбирается пиксел с той же ординатой (при ) или пиксел с ординатой, на единицу большей, чем у предыдущего пиксела (при ).

Поскольку предварительное значение ошибки вычисляется заранее, то есть вычислено на предыдущем шаге, то во втором случае останется только вычесть единицу из значения ошибки: , так как в этом случае , что не учитывалось при расчете.

## Вывод

Были рассмотрены принцип и идея конвейерной обработки данных, а также алгоритм Брезенхема с действительными коэффициентами.

В данной работе стоит задача реализации системы конвейерной обработки данных для рассмотренного алгоритма.

# Конструкторская часть

В данном разделе представлены схемы алгоритма Брезенхема и реализации конвейерной обработки данных для алгоритма Брезенхема.

## Схема алгоритма Брезенхема

пум-пум

## Схема реализации конвейерной обработки данных для алгоритма Брезенхема

пум-пум

## Вывод

Были представлены схемы алгоритма Брезензема, а также реализации конвейерной обработки данных для данного алгоритма.

# Технологическая часть

В данном разделе приведены требования к программному обеспечению, средства реализации программного обеспечения, а также листинг кода.

## Требования к программному обеспечению

* входные данные – количество выполняемых задач (количество растеризуемых отрезков);
* выходные данные – записи времени прихода и ухода обрабатываемых заявок для каждого реализованного конвейера.

## Средства реализации программного обеспечения

Взять из первой ЛР

## Листинг кода

Ну ты понял

## Тестирование программного продукта

Тестики для алгоритма, что ли… Растеризация трёх отрезков… Ожидаемый результат – класс QImage с заданными пикселами.

## Вывод

Спроектированные алгоритмы были реализованы и (протестированы).

# Исследовательская часть

В данном разделе приведены пример работы программного обеспечения, технические характеристики, а также время выполнения алгоритма.

## Пример работы программного обеспечения

угу

## Технические характеристики

ого

## Время выполнения алгоритма

хех

## Вывод

# Заключение

…