**Аппроксимация**

Аппроксимация — это приближённое представление сложной функции с помощью более простой. Она используется, когда у нас есть набор точек, и мы хотим провести "гладкую" кривую, которая **необязательно проходит через все точки**, но хорошо описывает общее поведение данных.

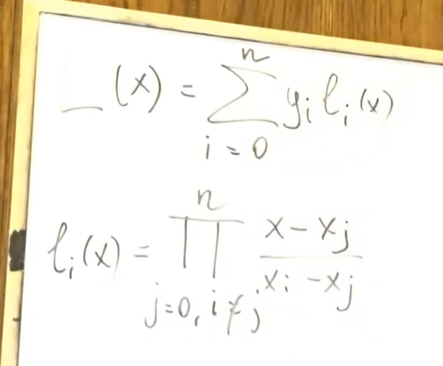
**Интерполяция**

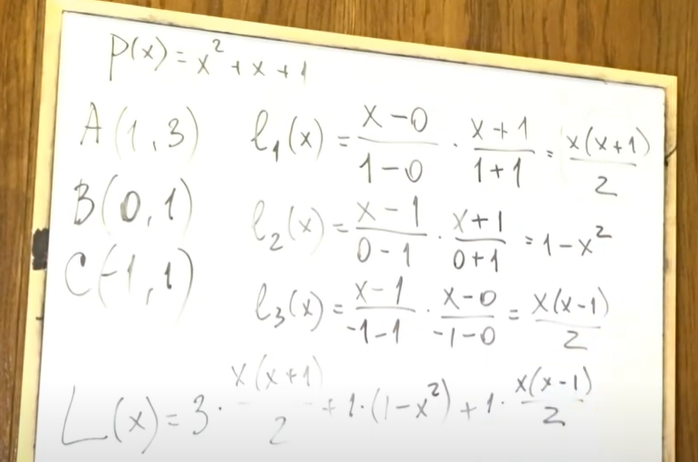
Интерполяция — это построение функции, которая **точно проходит через заданные точки**. Например, если у нас есть несколько известных значений, интерполяция позволяет найти значения между ними.

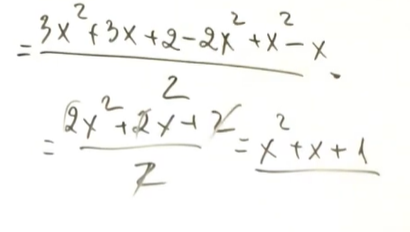
* **Полином Лагранжа** — это метод **интерполяции**, который строит полином, проходящий **через все заданные точки**. Он гарантирует, что результат точно совпадает с известными значениями в узлах.
* **Кривые Безье** — это способ **аппроксимации**. Кривая строится по опорным точкам, но **не обязана проходить через них** (кроме крайних). Используется часто в компьютерной графике для плавных форм.

# Полином Лагранжа

Полином Лагранжа — это способ построить **формулу, которая проходит через заданные точки**.







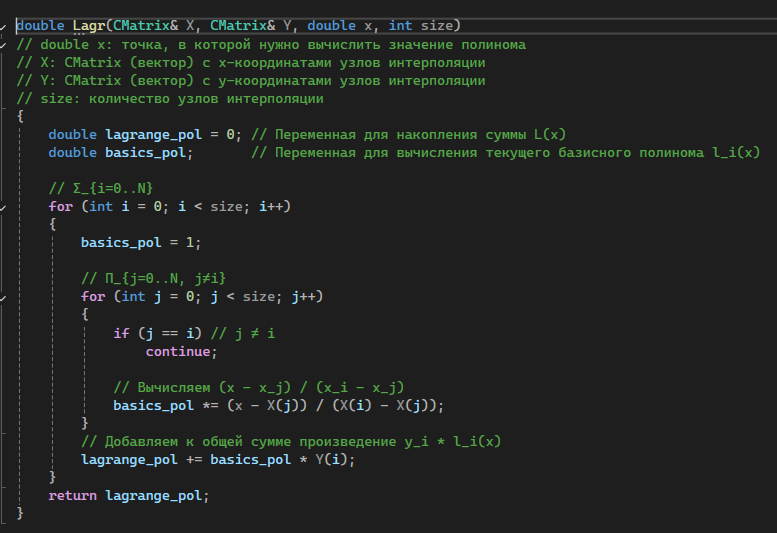
1. Создать функцию

*double Lagr(CMatrix &X, CMatrix &Y, double x)*

которая по множеству точек на плоскости 

, 

вычисляет значение интерполяционного **полинома Лагранжа** в точке .



2. Пункт меню **«Lines ► Lagr».**

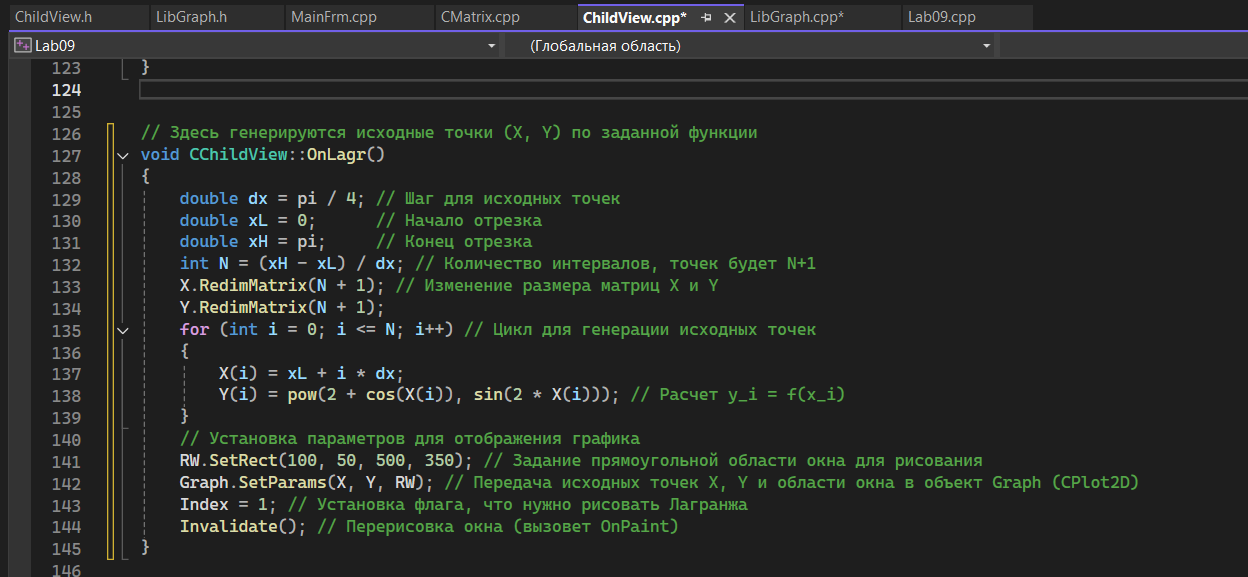
Получить множество точек

, 

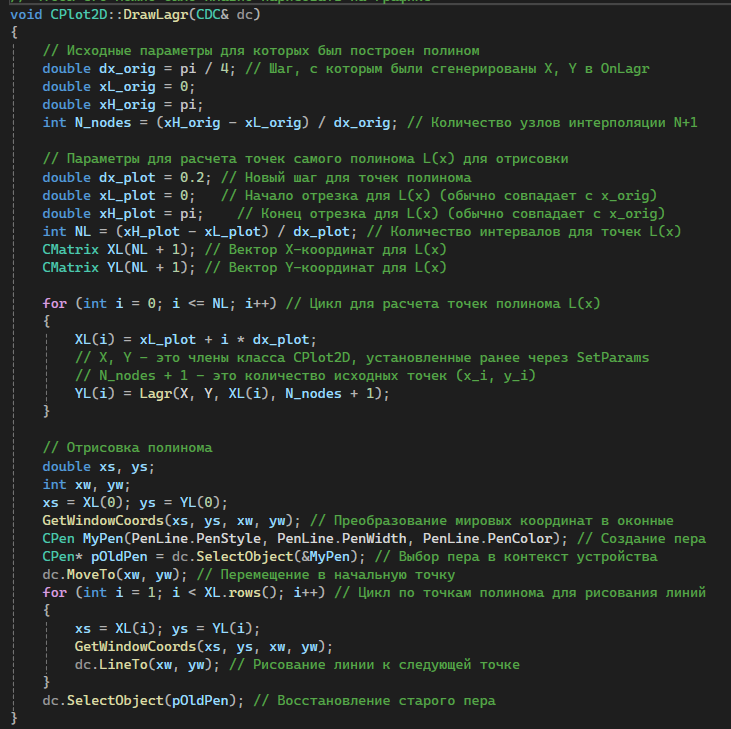


Координаты рассчитанных точек  записать в текстовый файл в формате, удобном для чтения его в пакете Mathcad.

По полученному набору данных рассчитать значения полинома Лагранжа  на отрезке  с шагом . Координаты рассчитанных точек  записать в текстовый файл в формате, удобном для чтения его в пакете Mathcad.



**OnLagr() (в ChildView.cpp)**. **Цель:** Получить **исходный набор опорных точек** (x\_i, y\_i), через которые должен пройти интерполяционный полином Лагранжа.



**DrawLagr() (в CPlot2D.cpp): Цель:** Вычислить значения **самого интерполяционного полинома Лагранжа L(x)** в большом количестве точек, чтобы его можно было плавно нарисовать на графике.

3. Построить график полинома Лагранжа с использованием класса CPlot2D.

В пакете Mathcad в одной системе координат построить зависимости  (значения  отобразить символом «точка») и  (линия).

4. Создать функцию

*void Bezier(CMatrix &X, CMatrix &Y, CMatrix &XB, CMatrix &YB, int M);*

которая по множеству ***заданных*** точек на плоскости

, 

вычисляет координаты **кривой Безье**

, ,

*M* – число отрезков, на которые разбивается параметр *t*, 

Для вычисления координат кривой Безье использовать геометрический алгоритм.

Координаты заданных опорных точек  и координаты рассчитанных точек кривой Безье  записать в текстовый файл в формате, удобном для чтения его в пакете Mathcad.

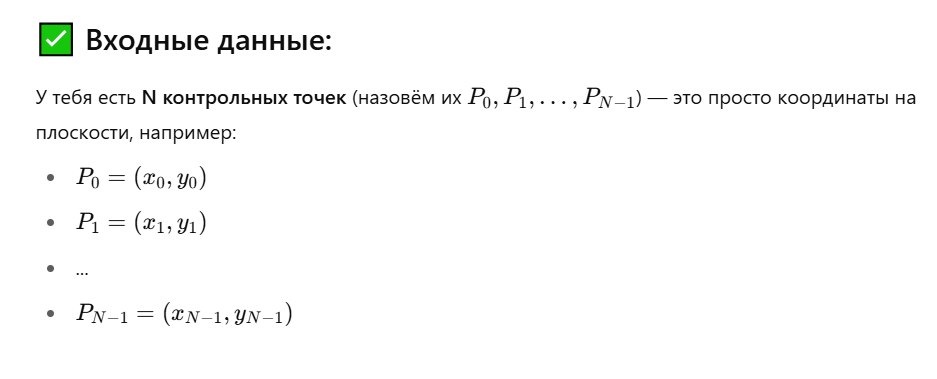
Кривые Безье — это **гладкие кривые**, которые используются для рисования красивых линий в графике, дизайне, анимации и даже шрифтах. Они очень гибкие и легко управляются с помощью **контрольных точек**.

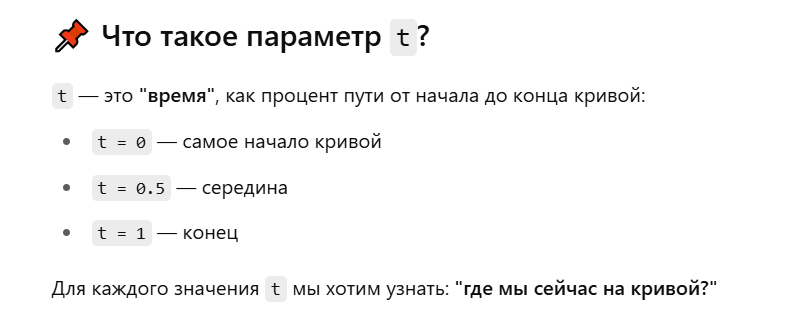
У тебя есть несколько точек на экране — как будто ты ставишь точки карандашом.

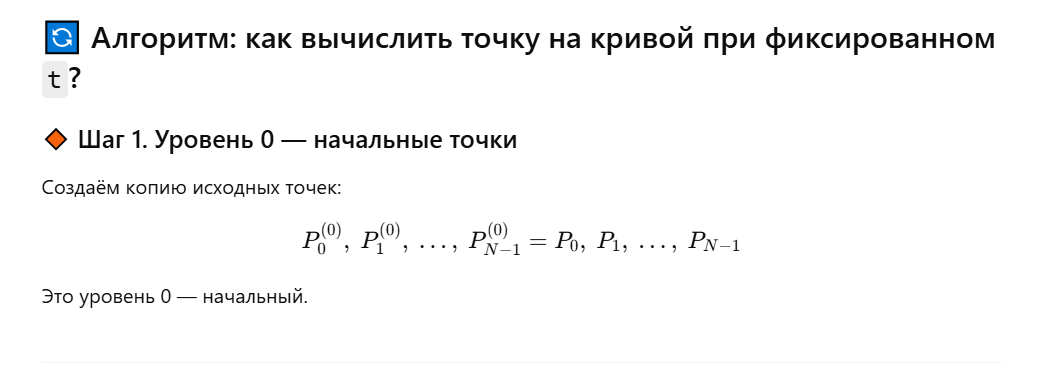
Программа берёт эти точки и **рисует плавную линию**, которая как бы «притягивается» к ним, но не обязательно через них проходит. Это и есть **кривая Безье**.

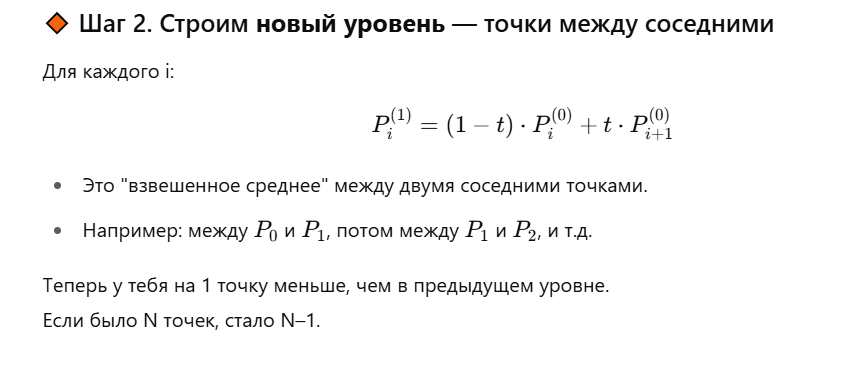
Функция **рисует кривую Безье** по заданным контрольным точкам X(i),Y(i)X(i), Y(i)X(i),Y(i).  
Алгоритм, который используется — **алгоритм де Кастельжо**, реализованный в виде численных операций.

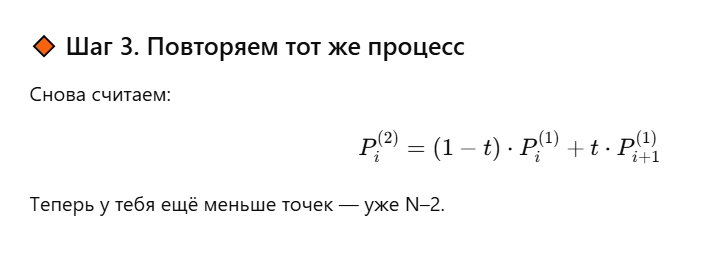
Ты дал программе список точек, она считает промежуточные,  
соединяет их — и рисует кривую. Это **похоже на сглаживание** — но умное, математическое сглаживание.

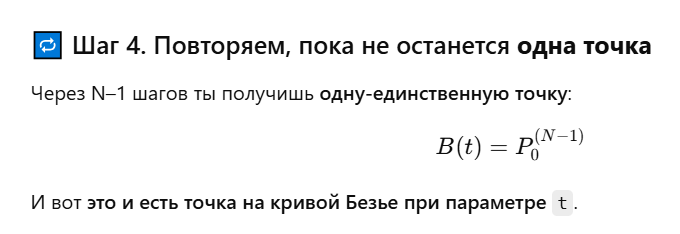


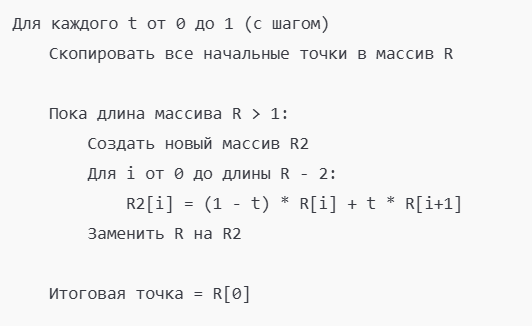


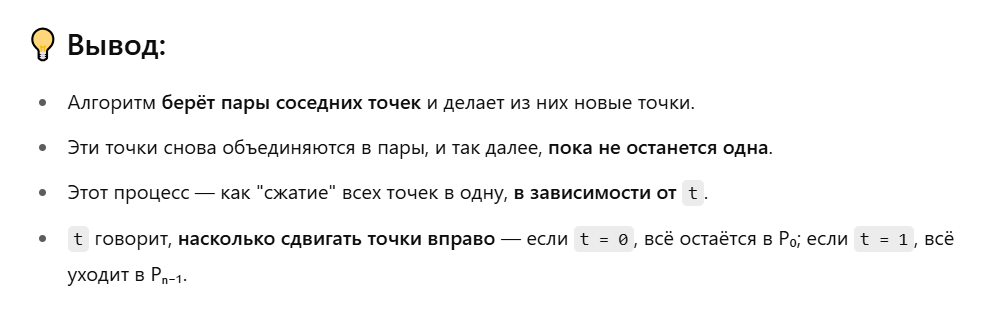


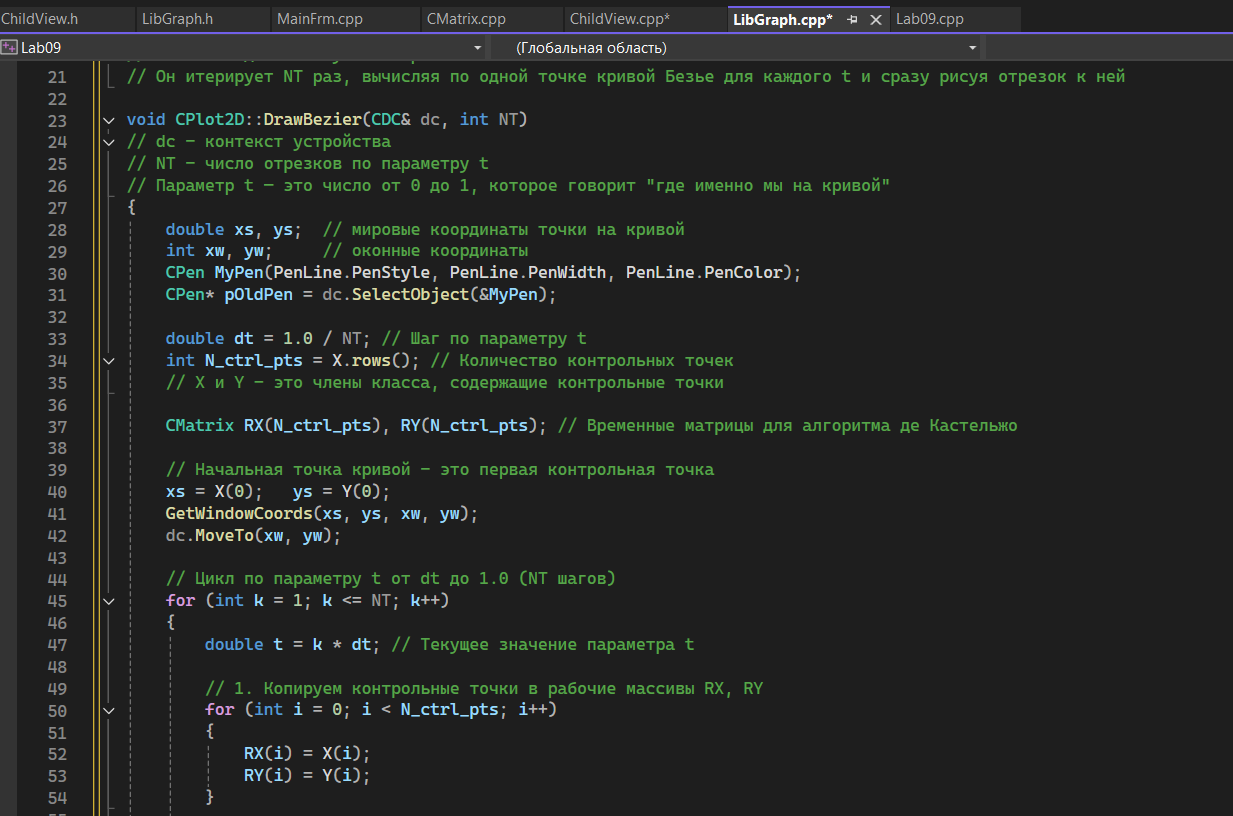












**1. Функции-обработчики меню в CChildView.cpp (например, OnBezier(), OnBezier1(), ... , OnBezier4()):**

* **Когда вызываются?** Когда вы выбираете соответствующий пункт в меню "Lines" (например, "Lines -> Bezier", "Lines -> Bezier (вариант 1)" и т.д.).
* **Что делают?**
  1. **Задают уникальный набор контрольных точек X и Y.** Это главная задача каждой из этих функций. Каждая функция OnBezier, OnBezier1 и т.д. определяет свой, особый набор контрольных точек. Эти точки сохраняются в членах класса CChildView (переменные this->X и this->Y).
  2. Устанавливают this->N\_Bezier = 50; (количество сегментов для построения кривой Безье).
  3. Устанавливают this->RW.SetRect(...) (область окна для рисования).
  4. Вызывают this->Graph.SetParams(this->X, this->Y, this->RW);. Этим действием **контрольные точки X и Y передаются объекту Graph (типа CPlot2D)**. Объект Graph теперь "знает", какие контрольные точки использовать.
  5. Устанавливают this->Index = N; (где N — это уникальный номер для каждого варианта Безье, например, 2 для OnBezier, 3 для OnBezier1 и т.д.). Этот Index будет использован в OnPaint().
  6. Вызывают this->Invalidate();. Это команда системе перерисовать окно. **Это приводит к вызову CChildView::OnPaint()**.

**2. Функция CChildView::OnPaint():**

* **Когда вызывается?** Автоматически, когда окно нужно перерисовать (например, после Invalidate(), при изменении размера окна, при первом отображении).
* **Что делает (для Безье)?**
  1. Проверяет значение this->Index.
  2. Если Index соответствует одному из вариантов Безье (например, if (Index == 2) или if (Index == 3) и т.д.):
     + **Рисуется первая линия (опорная ломаная, обычно зеленая):**
       - PenLine.Set(PS\_SOLID, 0, RGB(0, 255, 0)); // Устанавливаем зеленое перо
       - Graph.SetPenLine(PenLine);
       - Graph.Draw(dc, 0, 1);
         * **Откуда берутся данные для Graph.Draw()?** Она использует контрольные точки, которые были сохранены в объекте Graph на предыдущем шаге через Graph.SetParams() (то есть, Graph.X и Graph.Y).
         * **Что рисует Graph.Draw()?** Она просто соединяет точки (Graph.X(i), Graph.Y(i)) прямыми линиями. Это и есть ваша **опорная ломаная**.
     + **Рисуется вторая линия (сама кривая Безье, обычно красная):**
       - PenLine.Set(PS\_SOLID, 2, RGB(255, 0, 0)); // Устанавливаем красное перо
       - Graph.SetPenLine(PenLine);
       - Graph.DrawBezier(dc, N\_Bezier);
         * **Откуда берутся данные для Graph.DrawBezier()?** Она также использует **те же самые контрольные точки** Graph.X и Graph.Y, которые хранятся в объекте Graph.
         * **Что рисует Graph.DrawBezier()?** Внутри этой функции реализован алгоритм де Кастельжо. Он берет контрольные точки Graph.X, Graph.Y и параметр N\_Bezier (количество сегментов). На основе этого он вычисляет N\_Bezier + 1 точек, которые лежат на сглаженной кривой Безье, и соединяет их линиями. Это и есть **сама кривая Безье**.