# Введение

Цель: Реализация в программе IVP нескольких видов поточечных преобразований изображений и видео.

Задачи:

1. Реализовать инверсию изображения
2. Реализовать бинаризацию изображения

# Процесс выполнения

Результатом выполнения поставленных задач выступает новая вкладка обработки изображённая на рисунке 1.

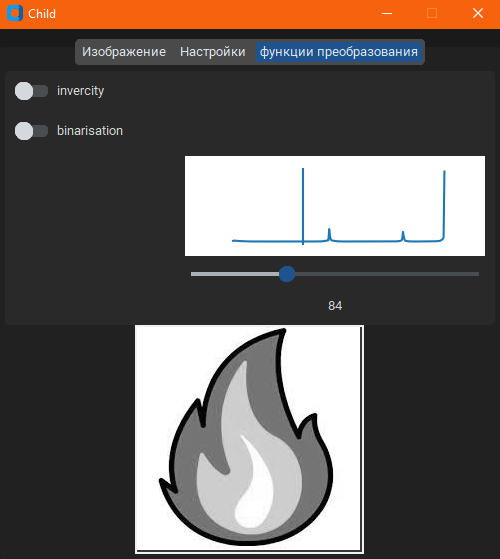
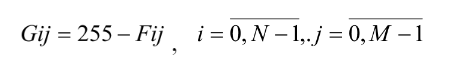


Рисунок 1 - Вкладка обработки изображения

Весь код представлен в приложении А

## Инверсия изображения

Инверсия меняет местами черный и белый цвет по следующей формуле:



На рисунке представлено черно белое (ЧБ) изображение огня и его инверсия.

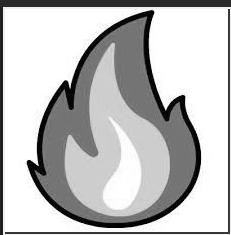
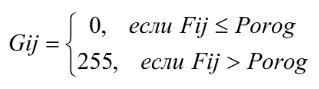
А)Б)

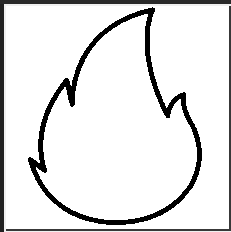
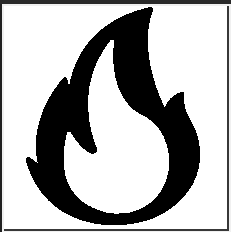
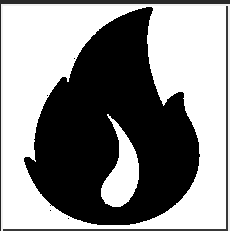
Рисунок 2 - ЧБ изображение. А) нормальное; Б) инверсия

## Бинаризация изображения

Программа рассчитывает бинарное изображение, пикселы которого принимают только два значений : 0 (черный), 255 (белый). По следующей формуле.



На рисунке 3 представлено использование данной функции

А)Б)в)

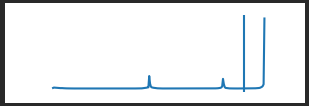
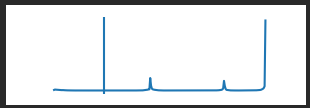


Рисунок 3 - Бинаризованное ЧБ изображение пламени.  
А) с границей на значении 60 Б) 150 в) 230

Важно, что бинаризация работает совместно с инверсией. На рисунке 4 представлено инверсия бинаризованного изображения при значении 230.

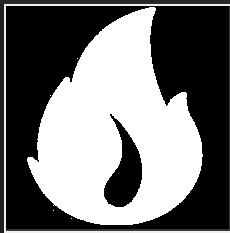


Рисунок 4 - Инверсия бинаризованного при значении 230 изображения ЧБ пламени

# Приложение А

**class** ConversionFunctions:

**def** \_\_init\_\_(self, master = None, preview\_frame=None) -> None:

        if master is None:

            master = CTk()

        self.master = master

        self.preview\_frame = preview\_frame

        self.functions\_frame        = CTkFrame(self.master)

        self.is\_inversion           = StringVar(value='0')

        self.inversion\_switch       = MySwitch(self.functions\_frame,

                                        text='invercity',

                                        variable=self.is\_inversion,

                                        command=self.update\_img)

        self.is\_binarisation        = StringVar(value='0')

        self.binarisation\_switch    = MySwitch(self.functions\_frame,

                                        text='binarisation',

                                        variable=self.is\_binarisation,

                                        command=self.update\_img)

        self.binarisation\_var\_label = CTkLabel(self.functions\_frame, text='100')

        self.binarisation\_var       = tk.IntVar()

        self.binarisation\_var.set(100)

        self.slider                 = CTkSlider(self.functions\_frame, from\_=0, to=255, number\_of\_steps=255, width=300,

                                        variable=self.binarisation\_var, command=self.upadate\_binarisation\_var)

        self.figure = plt.Figure(figsize=(3, 1), dpi=100)

        self.ax = self.figure.add\_subplot(111)

        self.ax.set\_axis\_off()

        histogram = cv2.calcHist([cv2.cvtColor(self.preview\_frame, cv2.COLOR\_BGR2GRAY)], [0], None, [256], [0, 256])

        x = np.linspace(0, 255, 256)

        self.graph, = self.ax.plot(x, histogram)

        self.line = self.ax.axvline(x=100)

        self.hist\_canvas = FigureCanvasTkAgg(self.figure, self.functions\_frame)

        self.preview = Output\_display(self.master, self.preview\_frame, self.conversion)

        self.draw\_widgets()

**def** upadate\_binarisation\_var(self, var):

        self.line.set\_xdata([var])

        self.binarisation\_var\_label.configure(text=**f**'{int(var)}')

        self.update\_img()

**def** conversion(self, img):

        if self.is\_binarisation.get() == '1':

            mask = img <= int(self.binarisation\_var.get())

            img[mask] = 0

            img[~mask] = 255

        if self.is\_inversion.get() == '1':

            img = 255 - img

        return img

**def** update\_img(self):

        histogram = cv2.calcHist([cv2.cvtColor(self.preview\_frame, cv2.COLOR\_BGR2GRAY)], [0], None, [256], [0, 256])

        self.graph.set\_ydata(histogram)

        self.hist\_canvas.draw()

        self.preview.is\_gray = True

        self.preview.on\_change()

**def** draw\_widgets(self):

        self.functions\_frame.pack()

        self.inversion\_switch.grid(             row=0, column=0, sticky=W, padx=10, pady=5)

        self.binarisation\_switch.grid(          row=1, column=0, sticky=W, padx=10, pady=5)

        self.slider.grid(                       row=3, column=1, sticky=W, padx=10, pady=5)

        self.binarisation\_var\_label.grid(       row=4, column=1, sticky=N, padx=10, pady=5)

        self.hist\_canvas.get\_tk\_widget().grid(  row=2, column=1, padx=10, pady=5)

        self.update\_img()

        self.preview.pack()