###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«Высокоуровневая работа с периферийными устройствами»

студента (ки) 2 курса, 18205 группы

Гайдамака Андрея Владиславовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А. Ю. Власенко

Новосибирск 2019

ЦЕЛЬ 3

ЗАДАНИЕ 3

ОПИСАНИЕ РАБОТЫ 3

ЗАКЛЮЧЕНИЕ 4

Приложение 1. Код программы 5

# ЦЕЛЬ

1. Ознакомиться с программированием периферийных устройств на примере ввода данных с Web-камеры с использованием библиотеки OpenCV.

# ЗАДАНИЕ

1. Реализовать программу с использованием OpenCV, которая получает поток видеоданных с камеры и выводит его на экран.
2. Выполнить произвольное преобразование изображения.
3. Измерить количество кадров, обрабатываемое программой в секунду. Оценить долю времени, затрачиваемого процессором на обработку (ввод, преобразование, показ) видеоданных, получаемых с камеры.

# ОПИСАНИЕ РАБОТЫ

После изучения документации к библиотеке OpenCV, была выбрана версия OpenCV 4.0 в которой по сравнению с версией 2.0 немного отличался синтаксис команд. В качестве преобразования видео был выбран эффект под названием «Негатив», он получается вычитанием из 255 каждой компоненты цвета. В качестве средства измерения времени была выбрана функция times, по причине высокой точности измерения за счет того, что она измеряет время работы только текущего процесса, и так же отлично подходит для измерения отдельных частей программы

При измерении времени работы определенных участков программы из Приложения 1 (считывание, обработка, вывод), были полученны следующие данные:

FPS = 17-18

Input = 11-12%

Processing = 74-76%

Output = 4-6%

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была изучена работа с камерой через библиотеку OpenCV. Так же в ходе этой работы была изучена работа с изображением представленном в виде матрицы пикселей имеющих свой цветовой оттенок. Были так же замерены важные участки работы программы, и с помощью этих измерений удалось получить FPS, а так же долю работы каждого участка: считывание, обработка, вывод

**Приложение 1. Код программы**

#include <opencv2/core.hpp>  
#include <opencv2/videoio.hpp>  
#include <opencv2/highgui.hpp>  
#include <iostream>  
#include <sys/times.h>  
#include <unistd.h>  
  
double get\_time(struct tms start, struct tms end, unsigned long clocks\_per\_sec){  
 return static\_cast<double>(end.tms\_utime - start.tms\_utime) / static\_cast<double>(clocks\_per\_sec);  
}  
  
  
int main(int argc,char \*argv[]){  
 cv::Mat frame;  
 cv::VideoCapture cap;  
 int deviceID = 0;  
 int apiID = cv::*CAP\_ANY*;  
 cap.open(deviceID + apiID);  
 if (!cap.isOpened()){  
 std::cerr << "Cant open" << std::endl;  
 return 1;  
 }  
 struct tms start\_all, end\_all, start\_input, end\_input, start\_proc, end\_proc, start\_out, end\_out;  
 long clocks\_per\_sec = sysconf(\_SC\_CLK\_TCK);  
 unsigned long int frames\_count = 0;  
 double input\_time = 0;  
 double processing\_time = 0;  
 double output\_time = 0;  
 times(&start\_all);  
 while (1){  
 times(&start\_input);  
 cap >> frame;  
 times(&end\_input);  
 input\_time += get\_time(start\_input, end\_input, clocks\_per\_sec);  
  
 if (frame.empty()){  
 std::cerr << "Error frame is empty" << std::endl;  
 break;  
 }  
  
 times(&start\_proc);  
 for (int y = 0; y < frame.cols; ++y){  
 for (int x = 0; x < frame.rows; ++x){  
 frame.at<cv::Vec3b>(x, y)[0] = 255 - frame.at<cv::Vec3b>(x, y)[0];  
 frame.at<cv::Vec3b>(x, y)[1] = 255 - frame.at<cv::Vec3b>(x, y)[1];  
 frame.at<cv::Vec3b>(x, y)[2] = 255 - frame.at<cv::Vec3b>(x, y)[2];  
 }  
 }  
 times(&end\_proc);  
 processing\_time += get\_time(start\_proc, end\_proc, clocks\_per\_sec);  
  
 times(&start\_out);  
 cv::imshow("Live", frame);  
 times(&end\_out);  
 output\_time += get\_time(start\_out, end\_out, clocks\_per\_sec);  
 frames\_count++;  
 if (cv::waitKey(5) >= 0){  
 break;  
 }  
 }  
 times(&end\_all);  
 double total\_time = get\_time(start\_all, end\_all, clocks\_per\_sec);  
 std::cout << "Total time: " << total\_time << " Fps: " << static\_cast<double>(frames\_count) / total\_time << std::endl  
 << "Input: " << input\_time / total\_time \* 100 << "%" << std::endl  
 << "Processing: " << processing\_time / total\_time \* 100 << "%" << std::endl  
 << "Output: " << output\_time / total\_time \* 100 << "%" << std::endl;  
 return 0;  
}