Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного автономного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ДОМАШНЯЯ РАБОТА №2

«Программирование с использованием OPENCL»

ДИСЦИПЛИНА: «Параллельные процессы в информационных системах»

Выполнил: студент гр. ИУК4-31М _____ (Сафронов Н.С.)
(Подпись) (Ф.И.О.)

Проверил: _____ (Корнюшин Ю.П.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель работы: формирование практических навыков по использованию фреймворка OpenCL.

Задачи

Основной задачей выполнения домашней работы является написание программы согласно варианту с использованием фреймворка OpenCL.

Задание

Программа должна быть реализована на объектно-ориентированном языке программирования с использованием фреймворка OpenCL. Разработать программу с использованием возможностей OpenCL в соответствии с вариантами заданий, приведенными в таблице. Во всех заданиях требуется создать матрицу (вектор) или несколько матриц (векторов) одинаковой размерности, указанной в таблице, заполнить их считанными из текстового файла значениями. Текстовые файлы следует предварительно подготовить, заполнив их случайными числами. Затем реализовать задание, используя обработку данных на OpenCL. В конце вывести результаты вычислений опять в текстовый файл.

Интерфейс программы – консольное приложение. В самом начале программа должна спрашивать имена входного и выходного файлов. Затем, после выполнения всех вычислений, печатать общее время работы (в секундах) на экране в виде дробного числа.

Вариант 6

Поменять в одномерном массиве элементы, находящиеся на четных и нечетных местах. Размер массива - 2^{33}, целочисленный.

Результат выполнения работы

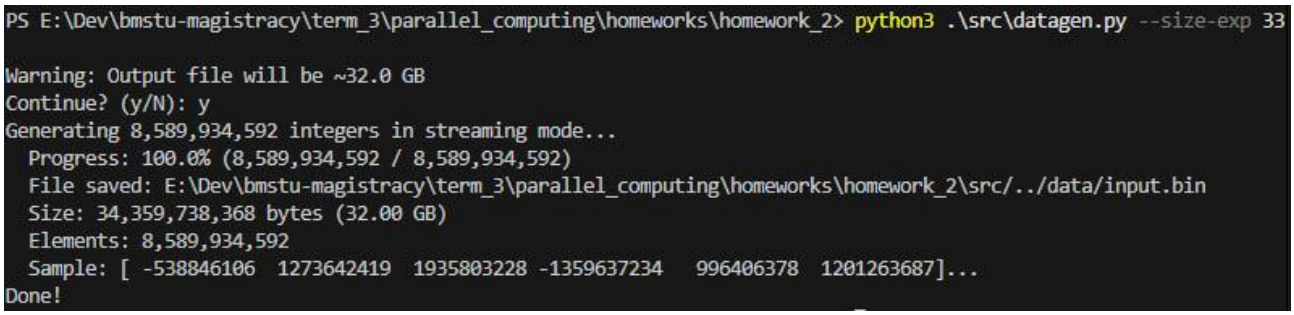


Рисунок 1 – Результат выполнения скрипта генерации входного файла

Листинг программы

```
import numpy as np
import os
import sys
from pathlib import Path
import argparse

def generate_data(output_path: str, size: int, seed: int = 42, block_size: int =
1024**2):
    """
    Генерирует size 32-битных целых чисел и записывает их в файл блоками.

    Args:
        output_path: Путь к выходному файлу
        size: Общее количество чисел
        seed: Сид генератора
        block_size: Количество чисел в одном блоке (по умолчанию 1 млн)
    """
    print(f"Generating {size:,} integers in streaming mode...")

    Path(output_path).parent.mkdir(parents=True, exist_ok=True)

    total_written = 0
    np.random.seed(seed)

    with open(output_path, 'wb') as f:
        while total_written < size:
            remaining = size - total_written
            current_block = min(block_size, remaining)

            block = np.random.randint(
                low=-2_147_483_648,
                high=2_147_483_647,
                size=current_block,
                dtype=np.int32
            )

            block.tofile(f)
            total_written += current_block

            if total_written % max(1, size // 10) == 0 or total_written == size:
                print(f"  Progress: {total_written / size * 100:.1f}%
({total_written:,} / {size:,})")

    with open(output_path, 'rb') as f:
        sample_bytes = f.read(6 * 4)
        sample = np.frombuffer(sample_bytes, dtype=np.int32)

    file_size = os.path.getsize(output_path)
```

```
    print(f"  File saved: {output_path}")
    print(f"  Size: {file_size:,} bytes ({file_size / (1024**3):.2f} GB)")
    print(f"  Elements: {size:,}")
    print(f"  Sample: {sample}...")

def main():
    parser = argparse.ArgumentParser(
        description="Generate binary data for OpenCL array swap (streaming)"
    )
    parser.add_argument(
        "--size-exp",
        type=int,
        default=23,
        help="Exponent N for array size 2^N (default: 23)"
    )
    parser.add_argument(
        "--output",
        type=str,

default=f"{os.path.dirname(os.path.abspath(__file__))}/../data/input.bin",
        help="Output file path (default: ../data/input.bin)"
    )
    parser.add_argument(
        "--seed",
        type=int,
        default=42,
        help="Random seed (default: 42)"
    )
    parser.add_argument(
        "--block-size",
        type=int,
        default=1024**2,
        help="Block size in elements (default: 1048576)"
    )

    args = parser.parse_args()

    size = 2 ** args.size_exp

    if size % 2 != 0:
        size -= 1

    estimated_gb = (size * 4) / (1024**3)
    if estimated_gb > 1.0:
        print(f"Warning: Output file will be ~{estimated_gb:.1f} GB")
        if input("Continue? (y/N): ").lower() != 'y':
            print("Aborted.")
            sys.exit(0)

    try:
        generate_data(args.output, size, args.seed, block_size=args.block_size)
        print("Done!")
    except KeyboardInterrupt:
        print("Interrupted by user")
        sys.exit(1)
    except Exception as e:
        print(f"Error: {e}")
        sys.exit(1)

if __name__ == "__main__":
    main()
```

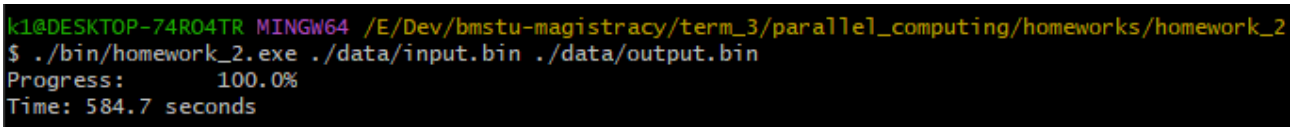


Рисунок 2 – Выполнение программы на GPU

Листинг программы на OpenCL

```
#define CL_TARGET_OPENCL_VERSION 120
#include <CL/cl.h>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <vector>
#include <string>
#include <chrono>
#include <stdexcept>
#include <cstdlib>

#define CL_CHECK(err) do { \
    if (err != CL_SUCCESS) { \
        std::cerr << "OpenCL error at line " << __LINE__ << ": " << err <<
std::endl; \
        throw std::runtime_error("OpenCL error"); \
    } \
} while(0)

class OpenCLArrayProcessor {
private:
    cl_context context = nullptr;
    cl_command_queue queue = nullptr;
    cl_kernel kernel = nullptr;
    cl_program program = nullptr;
    const size_t MAX_BLOCK_SIZE = 64 << 20; // 64M элементов = 256 MB


public:
    OpenCLArrayProcessor() {
        cl_int err;
        cl_platform_id platform;
        cl_device_id device;

        cl_uint numPlatforms;
        CL_CHECK(clGetPlatformIDs(1, &platform, &numPlatforms));
        if (numPlatforms == 0) throw std::runtime_error("No OpenCL platforms");

        cl_uint numDevices;
        CL_CHECK(clGetDeviceIDs(platform, CL_DEVICE_TYPE_GPU, 1, &device,
&numDevices));
        if (numDevices == 0) {
            CL_CHECK(clGetDeviceIDs(platform, CL_DEVICE_TYPE_CPU, 1, &device,
&numDevices));
            if (numDevices == 0) throw std::runtime_error("No OpenCL devices");
        }

        context = clCreateContext(nullptr, 1, &device, nullptr, nullptr, &err);
        CL_CHECK(err);
        queue = clCreateCommandQueue(context, device, 0, &err);
        CL_CHECK(err);

        const char* kernelSource =
            "__kernel void swap_even_odd(__global const int* input, __global
```

```
int* output, int size) {\n"
            "    int gid = get_global_id(0);\n"
            "    if (gid * 2 + 1 < size) {\n"
            "        int even = gid * 2;\n"
            "        int odd = even + 1;\n"
            "        output[even] = input[odd];\n"
            "        output[odd] = input[even];\n"
            "    }\n"
            "}\n";

        program = clCreateProgramWithSource(context, 1, &kernelSource, nullptr,
&err);
        CL_CHECK(err);
        CL_CHECK(clBuildProgram(program, 1, &device, nullptr, nullptr,
nullptr));

        kernel = clCreateKernel(program, "swap_even_odd", &err);
        CL_CHECK(err);
    }

    ~OpenCLArrayProcessor() {
        if (kernel) clReleaseKernel(kernel);
        if (program) clReleaseProgram(program);
        if (queue) clReleaseCommandQueue(queue);
        if (context) clReleaseContext(context);
    }

    void processFile(const std::string& inputFile, const std::string&
outputFile) {
        std::ifstream inFile(inputFile, std::ios::binary);
        if (!inFile) throw std::runtime_error("Cannot open input file");

        std::ofstream outFile(outputFile, std::ios::binary);
        if (!outFile) throw std::runtime_error("Cannot open output file");

        inFile.seekg(0, std::ios::end);
        size_t fileSize = inFile.tellg();
        inFile.seekg(0, std::ios::beg);
        size_t elementCount = fileSize / sizeof(int);
        if (elementCount % 2 != 0) elementCount--;

        size_t processed = 0;
        size_t totalElements = elementCount;

        std::cout << "Progress:\t0.0%" << std::flush;

        while (processed < totalElements) {
            size_t blockSize = std::min(MAX_BLOCK_SIZE, totalElements -
processed);
            size_t blockElements = blockSize - (blockSize % 2);

            std::vector<int> block(blockElements);
            inFile.read(reinterpret_cast<char*>(block.data()), blockElements *
sizeof(int));
            processBlock(block);
            outFile.write(reinterpret_cast<const char*>(block.data()),
blockElements * sizeof(int));
            processed += blockElements;

            double percent = (static_cast<double>(processed) / totalElements) *
100.0;
            std::cout << "\rProgress:\t" << std::fixed << std::setprecision(1)
<< percent << "%" << std::flush;
        }

        std::cout << std::endl;
```

```
    }

private:
    void processBlock(std::vector<int>& block) {
        cl_int err;
        size_t size = block.size();
        if (size == 0) return;

        cl_mem inputBuf = clCreateBuffer(context, CL_MEM_READ_ONLY |
CL_MEM_COPY_HOST_PTR,
                                         size * sizeof(int), block.data(), &err);
        CL_CHECK(err);
        cl_mem outputBuf = clCreateBuffer(context, CL_MEM_WRITE_ONLY, size *
sizeof(int), nullptr, &err);
        CL_CHECK(err);

        CL_CHECK(clSetKernelArg(kernel, 0, sizeof(cl_mem), &inputBuf));
        CL_CHECK(clSetKernelArg(kernel, 1, sizeof(cl_mem), &outputBuf));
        int sizeArg = static_cast<int>(size);
        CL_CHECK(clSetKernelArg(kernel, 2, sizeof(int), &sizeArg));

        size_t globalSize = size / 2;
        CL_CHECK(clEnqueueNDRangeKernel(queue, kernel, 1, nullptr, &globalSize,
nullptr, 0, nullptr, nullptr));
        CL_CHECK(clEnqueueReadBuffer(queue, outputBuf, CL_TRUE, 0, size *
sizeof(int), block.data(), 0, nullptr, nullptr));

        clReleaseMemObject(inputBuf);
        clReleaseMemObject(outputBuf);
    }
};

int main(int argc, char** argv) {
    if (argc < 3) {
        std::cerr << "Usage: " << argv[0] << " <input> <output>" << std::endl;
        return 1;
    }

    auto start = std::chrono::high_resolution_clock::now();
    try {
        OpenCLArrayProcessor proc;
        proc.processFile(argv[1], argv[2]);
    } catch (const std::exception& e) {
        std::cerr << "Error: " << e.what() << std::endl;
        return 1;
    }
    auto end = std::chrono::high_resolution_clock::now();

    std::cout << "Time: " << std::chrono::duration<double>(end - start).count()
<< " seconds" << std::endl;
    return 0;
}
```

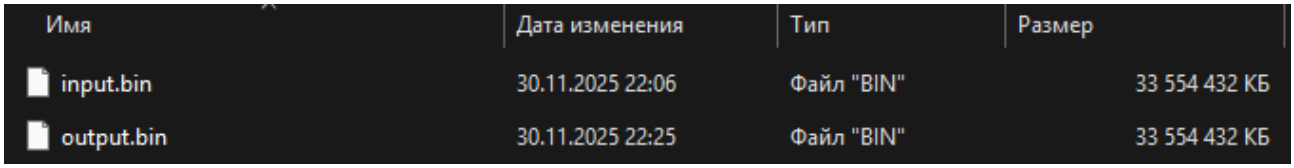
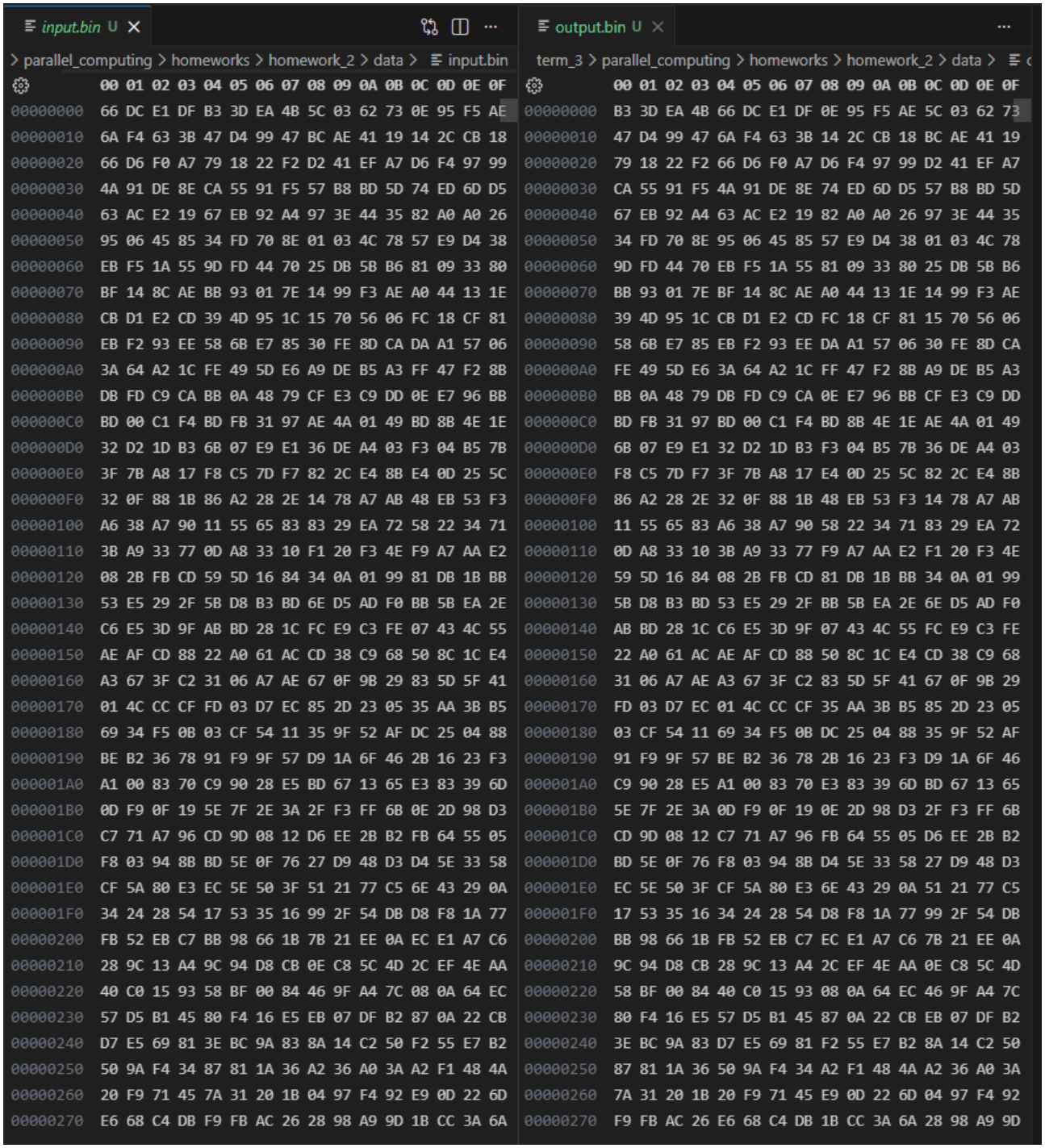


Рисунок 3 – Результирующие файлы



input.bin U

parallel_computing > homeworks > homework_2 > data > input.bin

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	66	DC	E1	DF	B3	3D	EA	4B	5C	03	62	73	0E	95	F5	AE
00000010	6A	F4	63	3B	47	D4	99	47	BC	AE	41	19	14	2C	CB	18
00000020	66	D6	F0	A7	79	18	22	F2	D2	41	EF	A7	D6	F4	97	99
00000030	4A	91	DE	8E	CA	55	91	F5	57	B8	BD	5D	74	ED	6D	D5
00000040	63	AC	E2	19	67	EB	92	A4	97	3E	44	35	82	A0	A0	26
00000050	95	06	45	85	34	FD	70	8E	01	03	4C	78	57	E9	D4	38
00000060	EB	F5	1A	55	9D	FD	44	70	25	DB	5B	B6	81	09	33	80
00000070	BF	14	8C	AE	BB	93	01	7E	14	99	F3	AE	A0	44	13	1E
00000080	CB	D1	E2	CD	39	4D	95	1C	15	70	56	06	FC	18	CF	81
00000090	EB	F2	93	EE	58	6B	E7	85	30	FE	8D	CA	DA	A1	57	06
000000A0	3A	64	A2	1C	FE	49	5D	E6	A9	DE	B5	A3	FF	47	F2	8B
000000B0	DB	FD	C9	CA	BB	0A	48	79	CF	E3	C9	DD	0E	E7	96	BB
000000C0	BD	00	C1	F4	BD	FB	31	97	AE	4A	01	49	BD	8B	4E	1E
000000D0	32	D2	1D	B3	6B	07	E9	E1	36	DE	A4	03	F3	04	B5	7B
000000E0	3F	7B	A8	17	F8	C5	7D	F7	82	2C	E4	8B	E4	0D	25	5C
000000F0	32	0F	88	1B	86	A2	28	2E	14	78	A7	AB	48	EB	53	F3
00000100	A6	38	A7	90	11	55	65	83	83	29	EA	72	58	22	34	71
00000110	3B	A9	33	77	0D	A8	33	10	F1	20	F3	4E	F9	A7	AA	E2
00000120	08	2B	FB	CD	59	5D	16	84	34	0A	01	99	81	DB	1B	BB
00000130	53	E5	29	2F	5B	D8	B3	BD	6E	D5	AD	F0	BB	5B	EA	2E
00000140	C6	E5	3D	9F	AB	BD	28	1C	FC	E9	C3	FE	07	43	4C	55
00000150	AE	AF	CD	88	22	A0	61	AC	CD	38	C9	68	50	8C	1C	E4
00000160	A3	67	3F	C2	31	06	A7	AE	67	0F	9B	29	83	5D	5F	41
00000170	01	4C	CC	CF	FD	03	D7	EC	85	2D	23	05	35	AA	3B	B5
00000180	69	34	F5	0B	03	CF	54	11	35	9F	52	AF	DC	25	04	88
00000190	BE	B2	36	78	91	F9	9F	57	D9	1A	6F	46	2B	16	23	F3
000001A0	A1	00	83	70	C9	90	28	E5	BD	67	13	65	E3	83	39	6D
000001B0	0D	F9	0F	19	5E	7F	2E	3A	2F	F3	FF	6B	0E	2D	98	D3
000001C0	C7	71	A7	96	CD	9D	08	12	D6	EE	2B	B2	FB	64	55	05
000001D0	F8	03	94	8B	BD	5E	0F	76	27	D9	48	D3	D4	5E	33	58
000001E0	CF	5A	80	E3	EC	5E	50	3F	51	21	77	C5	6E	43	29	0A
000001F0	34	24	28	54	17	53	35	16	99	2F	54	DB	D8	F8	1A	77
00000200	FB	52	EB	C7	BB	98	66	1B	7B	21	EE	0A	EC	E1	A7	C6
00000210	28	9C	13	A4	9C	94	D8	CB	0E	C8	5C	4D	2C	EF	4E	AA
00000220	40	C0	15	93	58	BF	00	84	46	9F	A4	7C	08	0A	64	EC
00000230	57	D5	B1	45	80	F4	16	E5	EB	07	DF	B2	87	0A	22	CB
00000240	D7	E5	69	81	3E	BC	9A	83	8A	14	C2	50	F2	55	E7	B2
00000250	50	9A	F4	34	87	81	1A	36	A2	36	A0	3A	A2	F1	48	4A
00000260	20	F9	71	45	7A	31	20	1B	04	97	F4	92	E9	0D	22	6D
00000270	E6	68	C4	DB	F9	FB	AC	26	28	98	A9	9D	1B	CC	3A	6A

output.bin U

term_3 > parallel_computing > homeworks > homework_2 > data > output.bin

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	B3	3D	EA	4B	66	DC	E1	DF	0E	95	F5	AE	5C	03	62	73
00000010	47	D4	99	47	6A	F4	63	3B	14	2C	CB	18	BC	AE	41	19
00000020	79	18	22	F2	66	D6	F0	A7	D6	F4	97	99	D2	41	EF	A7
00000030	CA	55	91	F5	4A	91	DE	8E	74	ED	6D	D5	57	B8	BD	5D
00000040	67	EB	92	A4	63	AC	E2	19	82	A0	A0	26	97	3E	44	35
00000050	34	FD	70	8E	95	06	45	85	57	E9	D4	38	01	03	4C	78
00000060	9D	FD	44	70	EB	F5	1A	55	81	09	33	80	25	DB	5B	B6
00000070	BB	93	01	7E	BF	14	8C	AE	A0	44	13	1E	14	99	F3	AE
00000080	39	4D	95	1C	CB	D1	E2	CD	FC	18	CF	81	15	70	56	06
00000090	58	6B	E7	85	EB	F2	93	EE	DA	A1	57	06	30	FE	8D	CA
000000A0	FE	49	5D	E6	3A	64	A2	1C	FF	47	F2	8B	A9	DE	B5	A3
000000B0	BB	0A	48	79	DB	FD	C9	CA	0E	E7	96	BB	CF	E3	C9	DD
000000C0	BD	FB	31	97	BD	00	C1	F4	BD	8B	4E	1E	AE	4A	01	49
000000D0	6B	07	E9	E1	32	D2	1D	B3	F3	04	B5	7B	36	DE	A4	03
000000E0	F8	C5	7D	F7	3F	7B	A8	17	E4	0D	25	5C	82	2C	E4	8B
000000F0	86	A2	28	2E	32	0F	88	1B	48	EB	53	F3	14	78	A7	AB
00000100	11	55	65	83	A6	38	A7	90	58	22	34	71	83	29	EA	72
00000110	0D	A8	33	10	3B	A9	33	77	F9	A7	AA	E2	F1	20	F3	4E
00000120	59	5D	16	84	08	2B	FB	CD	81	DB	1B	BB	34	0A	01	99
00000130	5B	D8	B3	BD	53	E5	29	2F	BB	5B	EA	2E	6E	D5	AD	F0
00000140	AB	BD	28	1C	C6	E5	3D	9F	07	43	4C	55	FC	E9	C3	FE
00000150	22	A0	61	AC	AE	AF	CD	88	50	8C	1C	E4	CD	38	C9	68
00000160	31	06	A7	AE	A3	67	3F	C2	83	5D	5F	41	67	0F	9B	29
00000170	FD	03	D7	EC	01	4C	CC	CF	35	AA	3B	B5	85	2D	23	05
00000180	03	CF	54	11	69	34	F5	0B	DC	25	04	88	35	9F	52	AF
00000190	91	F9	9F	57	BE	B2	36	78	2B	16	23	F3	D9	1A	6F	46
000001A0	C9	90	28	E5	A1	00	83	70	E3	83	39	6D	BD	67	13	65
000001B0	5E	7F	2E	3A	0D	F9	0F	19	0E	2D	98	D3	2F	F3	FF	6B
000001C0	CD	9D	08	12	C7	71	A7	96	FB	64	55	05	D6	EE	2B	B2
000001D0	BD	5E	0F	76	F8	03	94	8B	D4	5E	33	58	27	D9	48	D3
000001E0	EC	5E	50	3F	CF	5A	80	E3	6E	43	29	0A	51	21	77	C5
000001F0	17	53	35	16	34	24	28	54	D8	F8	1A	77	99	2F	54	DB
00000200	BB	98	66	1B	FB	52	EB	C7	EC	E1	A7	C6	7B	21	EE	0A
00000210	9C	94	D8	CB	28	9C	13	A4	2C	EF	4E	AA	0E	C8	5C	4D
00000220	58	BF	00	84	40	C0	15	93	08	0A	64	EC	46	9F	A4	7C
00000230	80	F4	16	E5	57	D5	B1	45	87	0A	22	CB	EB	07	DF	B2
00000240	3E	BC	9A	83	D7	E5	69	81	F2	55	E7	B2	8A	14	C2	50
00000250	87	81	1A	36	50	9A	F4	34	A2	F1	48	4A	A2	36	A0	3A
00000260	7A	31	20	1B	20	F9	71	45	E9	0D	22	6D	04	97	F4	92
00000270	F9	FB	AC	26	E6	68	C4	DB	1B	CC	3A	6A	28	98	A9	9D

Рисунок 4 – Содержимое исходного файла и результата

Вывод: в ходе выполнения лабораторной работы были сформированы практические навыки по использованию фреймворка OpenCL