МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Балтийский федеральный университет имени Иммануила Канта

Институт физико-математических наук и информационных технологий

**ОТЧЁТ**

**по учебной практике**

**(Учебная технологическая (проектно-технологическая) практика)**

**Подготовил:** Шатура Евгений Дмитриевич

Студент 1 курса направления подготовки «Прикладная математика и информатика»

Профиль: Информатика и программирование

Сроки практики: с 01.09.2021 г. по 25.12.2021 г.

**Руководитель:** Синюхин А. О.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**Калининград**

**2021 г.**

**Оглавление**

[**Введение** 3](#_Toc90310619)

[**Индивидуальное задание на практику**: 4](#_Toc90310620)

[**Задача №3 «Функции»** 4](#_Toc90310621)

[**Задача №4 «Счастливые билетики»** 9](#_Toc90310622)

[**Задача №5 «Удаление и дублирование»** 11](#_Toc90310623)

[**Задача №7 «Практикум по векторам»** 14](#_Toc90310624)

[**Задача №8 «Абстракция и инкапсуляция»** 17](#_Toc90310625)

[**Задача №9 «Самодельный вектор»** 22](#_Toc90310626)

[**Задача №10 «Инсталлятор»** 27](#_Toc90310627)

[**Заключение** 29](#_Toc90310628)

[**Перечень учебной литературы ресурсов сети «Интернет»,** 31](#_Toc90310629)

[**необходимой для проведения практики** 31](#_Toc90310630)

[**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»** 31](#_Toc90310631)

[**Отзыв руководителя о прохождении учебной практики** 32](#_Toc90310632)

# **Введение**

**Вид практики** - Учебная технологическая (проектно-технологическая) практика (далее Учебная практика).

**Цель** учебной практики: получение первичных профессиональных умений и навыков.

**Задачи** учебной практики:

* Закрепление и углубление теоретических знаний по программированию;
* Приобретение и развитие первичных профессиональных навыков и умений по прикладной математике и информатике.

# **Индивидуальное задание на практику**:

## **Задача №3 «Функции»**

*При написании кода придерживайтесь рекомендаций, которые обсуждали на лекциях:*

1. *Название функции соответствует тому, что она делает*
2. *Одна задача - одна функция*
3. *Разбитие на файлы*
4. *Используйте пространства имен*

*Оформите ввод, вывод и обработку данных в виде функций и процедур (пример был дан в лекции)*

Дан текст объемом не более 10 кб, найдите в нем самое длинное и самое короткое слово. Отсортируйте слова текста в алфавитном порядке.

**Формат ввода**

Текст на английском языке объемом не более 10 кб.

**Формат вывода**

Вывести два слова: самое длинное и самое короткое, и отсортированный список слов.

**Пример (intput.txt)**

**Вход**

I will go to the shop and you stay at home.

**Выход (output.txt)**

will i

and at go home i shop stay the to will you

**Решение на C++:**

main.cpp:

#include "add3.h"

int main() {

ifstream in("../input.txt");

ofstream out("../output.txt");

unsigned int max\_length = 0;

string max\_word;

unsigned int min\_length = UINT\_MAX;

string min\_word;

vector<string> words;

while (!in.eof()) {

string word = lab3::getWord(words, in);

lab3::compare(word, max\_length, max\_word, min\_length, min\_word);

}

sort(words.begin(), words.end());

lab3::writeAnswer(out, min\_word, max\_word, words);

out.close();

return 0;

}

add3.h:

#include <iostream>

#include <fstream>

#include <string>

#include <climits>

#include <vector>

#include <algorithm>

using namespace std;

namespace lab3 {

string clearingWord(string &word);

void addToList(string &word, vector <string> &words);

string getWord(vector <string> &words, ifstream &in);

void compare(string &word, unsigned int &max\_length, string &max\_word, unsigned int &min\_length, string &min\_word);

void writeAnswer(ofstream &out, string &min\_word, string &max\_word, vector <string> &words);

}

add3.cpp:

#include "add3.h"

namespace lab3 {

string clearingWord(string &word) {

string pure\_word;

for (const auto &symbol: word) {

if (isalpha(symbol))

pure\_word.push\_back((char) tolower(symbol));

}

return pure\_word;

}

void addToList(string &word, vector <string> &words) {

auto it = find(words.begin(), words.end(), word);

if (it == words.end()) {

words.push\_back(word);

}

}

string getWord(vector <string> &words, ifstream &in) {

string word;

in >> word;

word = clearingWord(word);

addToList(word, words);

return word;

}

void compare(string &word, unsigned int &max\_length, string &max\_word, unsigned int &min\_length, string &min\_word) {

unsigned int length;

length = word.size();

if (length > max\_length) {

max\_length = length;

max\_word = word;

}

if (length < min\_length) {

min\_length = length;

min\_word = word;

}

}

void writeAnswer(ofstream &out, string &min\_word, string &max\_word, vector <string> &words) {

out << "Самое длинное слово: " << max\_word << '\n';

out << "Самое короткое слово: " << min\_word << '\n';

for (const auto &w: words) {

out << w << ' ';

}

}

}

**Задача №4  
«Счастливые билетики»**

*Ограничение по времени 2 сек.*

*Ограничение по памяти 256 мб.*

Посчитайте количество всех возможных счастливых трамвайных билетиков. Номер билета содержит 6 цифр.

**Формат ввода (input.txt)**

Отсутствует.

**Формат вывода (output.txt)**

Вывести одно число: количество билетов.

**Решение на C++**:

#include <iostream>

using namespace std;

void newNewTicket(int \*cur\_ticket) {

for (int i = 5; i >= 0; i--) {

if (cur\_ticket[i] < 9) {

cur\_ticket[i]++;

break;

} else

cur\_ticket[i] = 0;

}

}

bool isHappy(const int \*cur\_ticket) {

return cur\_ticket[0] + cur\_ticket[1] + cur\_ticket[2] == cur\_ticket[3] + cur\_ticket[4] + cur\_ticket[5];

}

unsigned int searchWithMas(){

int current\_ticket[6];

for (int &i: current\_ticket) {

i = 0;

}

unsigned int count = 0;

for (int ticket = 1; ticket < 1000000; ticket++) {

newNewTicket(current\_ticket);

if (isHappy(current\_ticket))

count += 1;

}

return count;

}

int basicSearch() {

int count = 0;

int N;

for (int i = 1; i < 1000000; i++) {

if (i % 10 + (i / 10) % 10 + (i / 100) % 10 == (i / 1000) % 10 + (i / 10000) % 10 + (i / 100000) % 10)

count++;

}

return count;

}

int main() {

// простой поиск через остатки и цикл

cout << basicSearch() << '\n';

// поиск с помощью массива и генерацией нового билета

cout << searchWithMas() << '\n';

return 0;

}

**Задача №5  
 «Удаление и дублирование»**

*Реализуйте задачу с использованием простого массива.*

*Сначала введите последовательность в массив. Затем удалите и продублируйте элементы. Затем выведите полученную последовательность (каждый элемент по одному разу). Используйте в программе только один массив, функции и процедуры.*

Дана последовательность натуральных чисел {Aj}j=1...n (n<=10000). Удалить из последовательности числа, сумма цифр которых равна 18, а среди оставшихся продублировать числа, произведение цифр которых кратно 35.

**Формат ввода**

Первое число n – количество чисел в последовательности. Далее на новой строке n чисел последовательности.

**Формат вывода**

Вывести новую последовательность после преобразования.

**Пример**

**Вход (input.txt)**

8

99 990 92 75 571 10 99 5577

**Выход (output.txt)**

92 75 75 571 571 10 10 5577 5577

**Решение на C++**:

#include <iostream>

#include <fstream>

#define N 20000

using namespace std;

int sumDigits(int num) {

int sum = 0;

while (num) {

sum += num % 10;

num /= 10;

}

return sum;

}

int compDigits(int num) {

int comp = 1;

while (num) {

comp \*= num % 10;

num /= 10;

}

return comp;

}

void deleteElement(int mas[], int i, int &n){

for (int j = i; j < n - 1; j++) {

swap(mas[j], mas[j + 1]);

}

n--;

}

void addElement(int mas[], int &i, int &n){

for (int j = n; j >= i + 1; j--) {

swap(mas[j], mas[j - 1]);

}

mas[i] = mas[i + 1];

i += 2;

n++;

}

int read(ifstream &in, int mas[]){

int n;

in >> n;

for (int i = 0; i < n; i++) {

in >> mas[i];

}

return n;

}

void write(ofstream &out, int mas[], int n){

for (int j = 0; j < n; j++) {

out << mas[j] << ' ';

}

}

int main() {

ifstream in("../input.txt");

ofstream out("../output.txt");

int mas[N];

int n = read(in, mas);

int i = 0;

while (i < n) {

if (sumDigits(mas[i]) == 18) {

deleteElement(mas, i, n); // "удалить" элемент с индексом i с количеством чисел n

} else if (compDigits(mas[i]) % 35 == 0) {

addElement(mas, i, n); // добавить элемент с индексом i на место с индексом i+1 с количеством чисел n

} else

i++;

}

write(out, mas, n);

out.close();

return 0;

}

**Задача №7  
«Практикум по векторам»**

*Реализуйте следующую задачу, используя* ***std::vector***

[*https://appdividend.com/2019/05/03/cpp-vector-tutorial-with-example-vector-in-cpp-stl/*](https://appdividend.com/2019/05/03/cpp-vector-tutorial-with-example-vector-in-cpp-stl/)

*Сначала введите последовательность в массив. Затем удалите и продублируйте элементы. Затем выведите полученную последовательность (каждый элемент по одному разу). Используйте в программе только один массив, функции и процедуры.*

Дана последовательность натуральных чисел {Aj}j=1...n (n<=10000). Удалить из последовательности числа, сумма цифр которых равна 18, а среди оставшихся продублировать числа, произведение цифр которых кратно 35.

**Формат ввода**

На одной строке введена последовательность из n чисел.

**Формат вывода**

Вывести новую последовательность после преобразования.

**Пример**

**Вход (input.txt)**

99 990 92 75 571 10 775

**Выход (output.txt)**

92 75 75 571 571 10 775 775

**Решение на C++**:

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

int getSumFigure(int num) {

int sum = 0;

while (num) {

sum += num % 10;

num /= 10;

}

return sum;

}

int getMultFigure(int num) {

int mult = 1;

while (num) {

mult \*= num % 10;

num /= 10;

}

return mult;

}

void initVector(vector<int> &nums, ifstream &in) {

while (!in.eof()) {

int num;

in >> num;

nums.push\_back(num);

}

}

void deleteNums(vector<int> &nums) {

for (auto it = nums.begin(); it != nums.end();) {

if (getSumFigure(\*it) == 18) {

it = nums.erase(it);

} else {

it++;

}

}

}

void copyNums(vector<int> &nums) {

for (auto it = nums.begin(); it != nums.end();) {

if (getMultFigure(\*it) % 35 == 0 and getMultFigure(\*it) != 0) {

nums.insert(it + 1, \*it);

it++;

}

it++;

}

}

void writeAnswer(vector<int> &nums, ofstream &out) {

for (int num: nums) {

out << num << ' ';

}

}

int main() {

ifstream in("../input.txt");

ofstream out("../output.txt");

vector<int> nums;

// заполнение вектора числами

initVector(nums, in);

// удаление чисел по условию

deleteNums(nums);

// дублирование чисел по условию

copyNums(nums);

// запись последовательности в файл

writeAnswer(nums, out);

out.close();

return 0;

}

**Задача №8  
«Абстракция и инкапсуляция»**

*Создайте класс. Разбейте его реализацию на файлы \*.hpp и \*.cpp. Приведите пример использования класса.*

*Принцип ООП №1 (из 4-х) -* ***Абстракция***

*Вы будете самостоятельно придумывать, какие поля и методы разместить классе для решения вашей задачи. Ими вы опишите геометрическую фигуру из вашей задачи. Такое описание называется* ***абстракцией****.*

*Принцип ООП №2 (из 4-х) -* ***Инкапсуляция***

*Вам следует предусмотреть проверку вводимых данных. Например, нельзя ввести отрицательную длину прямоугольника. Если же пользователь это сделал, выведите предупреждение и заменить длину на значение по умолчанию (например, 1, можно выбрать любое).*

*Для вас это будет сделать несложно, т.к. пользователь напрямую не может обратиться к полям класса, они имеют ограничение доступа private.*

*И обращаться он будет только через методы public, либо конструктор. Там и разместите проверку.*

*Из всех полей и методов класса дать пользователю только то, что он не может сломать и что достаточно для выполнения задачи, а к остальному доступ запретить – это есть* ***инкапсуляция****.*

Создайте класс, описывающий круг. Найдите его периметр и площадь.

**Формат ввода**

В класс передается структура точки и радиус.

**Формат вывода**

Отсутствует.

**Решение на C++**:

main.cpp:

#include <iostream>

#include <circle.h>

int main() {

// Создание круга со значениями по умолчанию

std::cout << "circle 1:" << "\n";

gm::Circle circle1;

std::cout << "Координата x центра круга: " << circle1.getOriginPoint().x << '\n' << "Координата y центра круга: " <<

circle1.getOriginPoint().y << '\n' << "Радиус круга: " << circle1.getRadius() << '\n';

// Создание круга с правильными значениями и получение периметра, площади и радиуса

std::cout << "\ncircle 2:" << "\n";

gm::Circle circle2(gm::Point{2, 4}, 10);

std::cout << "Периметр: " << circle2.getPerimeter() << '\n';

std::cout << "Площадь: " << circle2.getSquare() << '\n';

std::cout << "Радиус: " << circle2.getRadius() << '\n';

std::cout << "Диаметр: " << circle2.getDiameter() << '\n';

// Создание круга с неправильными данными и получение радиуса

// В данном случае выводится сообщение, что введенные данные некорректны, и устанавливается значение по умолчанию

std::cout << "\ncircle 3:" << "\n";

gm::Circle circle3(gm::Point{1,1}, -10);

std::cout << "Радиус: " << circle3.getRadius() << '\n';

// Создание круга по умолчанию и попытка установить некорректное значение радиуса

// В данном случае выводится сообщение, что введенные данные некорректны, и устанавливается значение по умолчанию

std::cout << "\ncircle 4:" << "\n";

gm::Circle circle4;

circle4.setRadius(-10);

std::cout << "Радиус: " << circle4.getRadius() << '\n';

// Смена точки центра круга

std::cout << "Координата x центра круга: " << circle4.getOriginPoint().x << '\n';

std::cout << "Координата y центра круга: " << circle4.getOriginPoint().y << '\n';

circle4.setOriginPoint(gm::Point{123, 321});

std::cout << "Смена точки" << '\n';

std::cout << "Координата x центра круга: " << circle4.getOriginPoint().x << '\n';

std::cout << "Координата y центра круга: " << circle4.getOriginPoint().y << '\n';

return 0;

}

circle.h:

#pragma once

#include <iostream>

#include <cmath>

#define PI acos(-1)

namespace gm{

struct Point {

int x, y;

};

class Circle {

public:

Circle();

Circle(Point P, int R);

int getRadius() const;

int getDiameter() const;

Point getOriginPoint() const;

void setRadius(int R);

void setOriginPoint(Point P);

double getSquare() const;

double getPerimeter() const

private:

Point origin\_point{1, 1};

int radius = 1;

static bool verifyIncomingData(int &r);

};

}

circle.cpp:

#include "circle.h"

namespace gm {

Circle::Circle() = default;

Circle::Circle(Point P, int R) {

if (not verifyIncomingData(R)) {

std::cout << "Введены некорректные данные, они заменены на значения по умолчанию. "

"Для замены используйте методы setRadius и setOriginPoint.\n";

} else

radius = R;

origin\_point = P;

}

int Circle::getRadius() const {

return radius;

}

Point Circle::getOriginPoint() const {

return origin\_point;

};

void Circle::setRadius(int R) {

if (verifyIncomingData(R)) {

radius = R;

} else

std::cout << "Введены некорректные данные, они заменены на значения по умолчанию.\n";

}

void Circle::setOriginPoint(Point P) {

origin\_point = P;

};

double Circle::getSquare() const {

return PI \* radius \* radius;

}

double Circle::getPerimeter() const {

return 2 \* PI \* radius;

}

bool Circle::verifyIncomingData(int &r) {

if (r < 0)

return false;

return true;

}

int Circle::getDiameter() const {

return 2 \* radius;

}

}

**Задача №9  
«Самодельный вектор»**

*Создайте класс, который реализует динамический массив наподобие std::vector в соответствии с идиомой RAII.*

*Реализуйте в нем функции:*

1. *Получение размера*
2. *Добавление элемента в конец*
3. *Добавление элемента по индексу*
4. *Получение элемента по индексу*
5. *Согласно варианту*

*Предусмотрите защиту от неправильного использования класса.*

*Разбейте его реализацию на файлы \*.hpp и \*.cpp. Предусмотрите пространство имен. Приведите пример использования класса.*

***Идиома RAII******(Resource Acquisition Is Initialization)*** *– получение ресурса (в данном случае, памяти под динамический массив) связано с инициализацией объекта (вызов конструктора), а освобождение ресурса (очистка памяти) – уничтожением объекта (вызов деструктора).*

Функция поиска минимума в массиве.

**Формат ввода**

Отсутствует.

**Формат вывода**

Отсутствует.

**Решение на C++**:

main.cpp:

#include <iostream>

#include "add9.h"

int main() {

add9::MyVector nums;

nums.pushBack(10);

nums.pushBack(-10);

nums.pushBack(20);

nums.pushBack(30);

nums.pushBack(40);

nums.pushBack(50);

nums.pushBack(60);

nums.print();

std::cout << nums.len() << '\n';

nums.add(345, 2);

nums.print();

std::cout << nums.get(4) << '\n';

std::cout << nums.get(5467) << '\n';

std::cout << nums.min() << '\n';

return 0;

}

add9.hpp:

#pragma once

namespace add9 {

class MyVector {

public:

MyVector();

~MyVector();

unsigned int len() const;

void pushBack(int num);

void add(int num, int index);

int get(int index);

int min();

void print();

private:

// текущее количество элементов в массиве

unsigned int count;

// максимальное количество элементов в массиве

unsigned int capacity;

// сам массив (логично)

int \*array;

void isFit();

};

}

add9.cpp:

#include "add9.h"

#include <iostream>

#include <algorithm>

namespace add9 {

MyVector::MyVector() {

count = 0;

capacity = 1;

array = new int[1]();

}

MyVector::~MyVector() {

delete[] array;

}

unsigned int MyVector::len() const {

return count;

}

void MyVector::pushBack(int num) {

isFit();

count++;

array[count - 1] = num;

}

void MyVector::add(int num, int index) {

if (index < count) {

isFit();

for (int i = count; i >= index + 1; i--) {

std::swap(array[i], array[i - 1]);

}

array[index] = num;

count++;

} else if (index == count)

pushBack(num);

else {

std::cout << "ERROR: Index out of range.\n";

}

}

int MyVector::get(int index) {

if (index <= count) {

return array[index];

} else {

std::cout << "ERROR: Index out of range.\n";

return 0;

}

int MyVector::min() {

if (count > 0) {

int min\_el = array[0];

for (int i = 1; i < count; i++) {

min\_el = std::min(min\_el, array[i]);

}

return min\_el;

}

return 0;

}

void MyVector::isFit() {

if (count + 1 < capacity) {

int \*temp = new int[capacity + 2];

for (int i = 0; i < capacity; i++) {

temp[i] = array[i];

}

delete[] array;

capacity += 2;

array = temp;

}

}

void MyVector::print() {

for (int i = 0; i < count; i++) {

std::cout << array[i] << " ";

}

std::cout << '\n';

}

}

**Задача №10  
«Инсталлятор»**

При помощи программы InnoSetup (<https://habr.com/ru/post/255807/>) создайте инсталлятор для вашей программы. Проверьте, что установленная программа запускается на вашем компьютере.

**Решение**:

#define Name "Darts Game"

#define Version "D0.1"

#define Publisher "ShEvgy"

#define ExeName "Darts.exe"

[Setup]

AppId={{3113320E-C7BE-409C-847E-25E83BD63B8D}

AppName={#Name}

AppVersion={#Version}

AppPublisher={#Publisher}

DefaultDirName={pf}\{#Name}

DefaultGroupName={#Name}

OutputDir="F:\CLionProjects\additional\10. inno setup"

OutputBaseFileName=Darts Game setup

SetupIconFile="F:\CLionProjects\additional\10. inno setup\dart.ico"

Compression=lzma

SolidCompression=ye

[Languages]

Name: "russian"; MessagesFile: "compiler:Languages\Russian.isl";

[Tasks]

Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked

[files]

Source: "F:\CLionProjects\sfml\_Project\cmake-build-debug\sfml\_sample.exe"; DestDir: "{app}"; Flags: ignoreversion; DestName: Darts.exe

Source: "F:\CLionProjects\sfml\_Project\data\\*"; DestDir: "{app}\data"; Flags: ignoreversion recursesubdirs createallsubdirs

[Icons]

Name: "{group}\{#Name}"; Filename: "{app}\{#ExeName}"; IconFilename: "{app}\data\ico\dart.ico"; Tasks: desktopicon

Name: "{commondesktop}\{#Name}"; Filename: "{app}\{#ExeName}"; Tasks: desktopicon

**Заключение**

В процессе прохождения учебной практики, я приобрел первичные практические умения и навыки по прикладной математике и информатике.

А именно:

* закрепление и углубление теоретических знаний по программированию;
* способность приобретать новые научные и профессиональные знания, используя современные образовательные и информационные технологии;
* способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;
* способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения.

Более того, в процессе прохождения учебной практики я:

* ознакомился с основными принципами ООП в языке C++;
* получил навыки конструирования классов и структур;
* научился определять утечки памяти при работе с динамической памятью;
* освоил создание защиты от неправильного использования класса, процедур или функций пользователем.

Также за время прохождения учебной практики были выявлены основные для меня особенности языка C++:

* C++ - чрезвычайно мощный язык, содержащий инструменты для создания мощных программ практически для любых целей, от низкоуровневых инструментов и драйверов до сложных программных систем для различных целей;
* Возможность использования различных namespace’ов, с помощью которых можно проще группировать функции, процедуры и классы;
* Быстрое выполнение кода;
* Более надежное освобождение ресурсов путем автоматического вызова деструкторов при уничтожении объектов;
* Доступ к памяти с помощью указателей.

Однако в то же время в C++ для себя открыл несколько недостатков:

* Отсутствуют средства автоматического управления памятью;
* Программы на С++ обычно длиннее, нежели аналогичные на языке Python, который я использовал ранее в промышленном программировании;
* Написать на С++ портативный код (такой, который бы работал на разных платформах) очень сложно;
* Читаемость кода ниже, чем в некоторых других языках (например, в Python).

В течении практики все необходимые задачи были выполнены, а цели достигнуты.

**Перечень учебной литературы ресурсов сети «Интернет»,**

**необходимой для проведения практики**

1. Кувшинов, Д. Р. Основы программирования [Электронный ресурс]: учеб. пособие для вузов/ Д. Р. Кувшинов, Ю. В. Бидуля; Урал. федер. ун-т им. первого Президента России Б. Н. Ельцина. - Москва: Юрайт, 2019. - 1 on-line, 104 с..
2. Зыков С. В. Программирование. Объектно-ориентированный подход [Электронный ресурс] : учеб. и практикум для акад. бакалавриата / С. В. Зыков, 2019. - 1 on-line, 156 с.

**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

1. Видеолекции по курсу «Основы программирования». – режим доступа: http://www.youtube.com/watch?v=pxR3UoO9c9w
2. Сайт «Структуры и алгоритмы». – режим доступа: http://www.structur.h1.ru/

# **Отзыв руководителя о прохождении учебной практики**

Студент 1 курса направления подготовки бакалавриата «Прикладная математика и информатика» Шатура Евгений Дмитриевич проходил учебную практику в институте физико-математических наук и информационных технологий с 01.09.2021г. по 25.12.2021г.

Задачи данной учебной практики заключаются в закреплении и углублении теоретических знаний и приобретении профессиональных навыков и умений по прикладной математике и информатике. В результате прохождения практики у студента формируются профессиональные компетенции, в том числе:

* способность приобретать новые научные и профессиональные знания, используя современные образовательные и информационные технологии;
* способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;
* способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения.

Студент Шатура Евгений Дмитриевич в процессе прохождения практики справился с поставленными задачами, приобрел первичные профессиональные навыки и умения.

Оценка за учебную практику \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель практики:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_