МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ О ПРОХОЖДЕНИИ

УЧЕБНО-ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

на базе Института физико-математических наук и информационных технологий

Выполнил Шатура Евгений Дмитривеич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студент очной формы обучения 1 курса

направления подготовки 01.03.02 Прикладная математика и информатика

профиль обучения «Информатика и программирование»

Руководитель практики

Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Синюхин А. О.

г. Калининград 2022 г.

Оглавление

[Введение 3](#_Toc105004133)

[Глава 1. Базовые понятия ООП 4](#_Toc105004134)

[Инкапсуляция 4](#_Toc105004135)

[Наследование 5](#_Toc105004136)

[Полиморфизм 6](#_Toc105004137)

[Глава 2. Задания на практику 7](#_Toc105004138)

[задача №1 «Рекурсия» 7](#_Toc105004139)

[задача №2 «Сортировки» 7](#_Toc105004140)

[задача №3 «Двусвязный циклический список» 7](#_Toc105004141)

[задача №4 «Стеганография» 7](#_Toc105004142)

[задача №5 «Поиск ошибок» 7](#_Toc105004143)

[Глава 3. Выполнение ДОПОЛНИТЕЛЬНЫх Заданий на практику 9](#_Toc105004144)

[Дополнительная задача №1 9](#_Toc105004145)

[Дополнительная задача №2 10](#_Toc105004146)

[Дополнительная задача №3 11](#_Toc105004147)

[Дополнительная задача №4 16](#_Toc105004148)

[Дополнительная задача №5 17](#_Toc105004149)

[Заключение 18](#_Toc105004150)

[Список литературы 20](#_Toc105004151)

[Приложения 21](#_Toc105004152)

# **Введение**

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

Цель учебной практики: получение первичных профессиональных умений навыков.

Задачи учебной практики:

* Закрепление и углубление теоретических знаний по программированию;
* Изучение основ объектно-ориентированного программирования в C++;
* Понятие принципов проектирования классов, перегрузки операторов, наследования, инкапсуляции данных, деструкторов классов, правила трёх;
* Приобретение опята работы с Git;
* Приобретение и развитие первичных профессиональных навыков и умений по прикладной математике и информатике.

# **Глава 1. Базовые понятия ООП**

## **Инкапсуляция**

Инкапсуляция – одна из ключевых особенностей объектно-ориентированного программирования. Она включает в себя объединение членов данных и функций внутри одного класса. Объединение похожих членов данных и функций внутри класса вместе также помогает в сокрытии данных.

В общем случае инкапсуляция – это процесс упаковки аналогичного кода в одном месте. В C ++ мы можем объединять элементы данных и функции, которые работают вместе внутри одного класса.

В приведенной в приложении 1 программе функция getArea() вычисляет площадь прямоугольника. Чтобы вычислить площадь, ей нужна длина и ширина. Следовательно, члены данных (lengthи breadth) и функция getArea() хранятся вместе в Rectangle классе.

В приведенном в приложении 2 примере мы вычисляем площадь прямоугольника. Чтобы вычислить площадь, нам нужны две переменные: length и breadth и функция: getArea(). Следовательно, мы объединили эти переменные и функции внутри одного класса с именем Rectangle. Здесь переменные и функции могут быть доступны и из других классов. Следовательно, это не скрытие данных, а только инкапсуляция. Мы просто держим похожие коды вместе.

Стоит отметить, что часто рассматривают инкапсуляцию как скрытие данных, но это не совсем так. Инкапсуляция относится к объединению связанных полей и методов вместе. Это можно использовать для скрытия данных. Инкапсуляция сама по себе не скрывает данные.

В C ++ инкапсуляция помогает нам сохранять связанные данные и функции вместе, что делает наш код более чистым и легким для чтения. Это помогает контролировать модификацию наших членов данных.

Рассмотрим ситуацию в приложении 3, когда мы хотим length, чтобы поле в классе было неотрицательным. Здесь мы можем сделать length переменную закрытой и применить логику внутри метода setAge().

Функции getter и setter обеспечивают доступ только для чтения или записи к нашим членам класса. Это помогает развязать компоненты системы. Так мы можем инкапсулировать код в несколько пакетов. Эти несвязанные компоненты (пакеты) могут разрабатываться, тестироваться и отлаживаться независимо и одновременно. И любые изменения в конкретном компоненте не оказывают никакого влияния на другие компоненты.

Мы также можем добиться скрытия данных с помощью инкапсуляции. В приложении 1, если мы изменим length breadth и переменные на privateor или protected, то доступ к этим полям будет ограничен, и они будут скрыты от внешних классов. Это называется скрытием данных.

## **Наследование**

Наследование – одна из ключевых особенностей объектно-ориентированного программирования на C++. Это позволяет нам создать новый класс (производный класс) из существующего класса (базового класса).

Производный класс наследует функции базового класса и может иметь собственные дополнительные функции. Например, в приложении 4 класс Dog является производным от Animal класса. Поскольку Dog является производным от Animal, члены Animal доступны Dog.

Наследование – это отношение is-a. Мы используем наследование только в том случае, если между двумя классами присутствует отношение is-a.

Вот несколько примеров:

* Автомобиль – это транспортное средство;
* Оранж – это фрукт;
* Хирург – это врач;
* Собака – это животное.

Модификатор доступа protected особенно актуален, когда речь идет о наследовании C++. Как и private члены, protected члены недоступны вне класса. Однако они могут быть доступны производными классами и классами или функциями друзей. Protected члены нужны, если мы хотим скрыть данные класса, но при этом хотим, чтобы эти данные наследовались его производными классами.

В приложении 5 переменная type является protectedи, таким образом, доступна из производного класса Dog. Мы можем видеть это, поскольку мы инициализировали type в Dog классе с помощью функции setType(). С другой стороны, private переменная color не может быть инициализирована Dog. Кроме того, поскольку ключевое слово protected скрывает данные, мы не можем получить доступ к типу непосредственно из объекта DogAnimal или класса.

Различные способы получения классов известны как режимы доступа. Эти режимы доступа имеют следующий эффект:

* public:  
  Если производный класс объявлен в public режиме состояния, то члены базового класса наследуются производным классом так же, как они есть;
* private:  
  В этом случае все члены базового класса становятся privateчленами производного класса;
* protected:  
  public члены базового класса становятся protected членами производного класса.

## **Полиморфизм**

Одной из ключевых особенностей наследования классов является то, что указатель на производный класс совместим по типу с указателем на его базовый класс. Полиморфизм – это искусство использования этой простой, но мощной и универсальной функции.

Пример о классах rectangle и triangle, приведённый в приложении 6, написан с помощью указателей с учетом этой особенности. Функция main объявляет два указателя на Polygon (с именем ppoly1и ppoly2). Им присваиваются адреса rect и trgl, соответственно, которые являются объектами типа Rectangleand Triangle. Такие назначения допустимы, так как оба Rectangle и Triangle являются классами, производными от Polygon.

Разыменование ppoly1 и ppoly2(с ppoly1-> and ppoly2->) допустимо и позволяет нам получить доступ к членам их точечных объектов, но поскольку тип обоих ppoly1 и ppoly2 является указателем на Polygon (а не указателем на Rectangle и не указателем на Triangle), можно получить доступ только к членам, унаследованным от Polygon, а не к производным классам Rectangle и Triangle. Вот почему вышеприведенная программа обращается к area членам обоих объектов с помощью rect и trgl напрямую, а не указателей; указатели на базовый класс не могут получить доступ к area членам.

Член area мог бы быть доступен с указателями Polygon, если area бы был членом Polygon вместо члена его производных классов, но проблема в том, что Rectangle и Triangle реализовать разные версии area поэтому нет ни одной общей версии, которая могла бы быть реализована в базовом классе.

# **Глава 2. Задания на практику**

## **задача №1 «Рекурсия»**

*Вариант 2.*

Дано число N. Рассчитать сумму его цифр.

*Задачи во всех вариантах следует выполнять с применением рекурсии. Циклы использовать нельзя.*

## **задача №2 «Сортировки»**

Реализовать сортировку и оцените ее сложность.

*Вариант 2*

Сортировка Шелла.

## **задача №3 «Двусвязный циклический список»**

Реализуйте двусвязный циклический список, предусмотрите в нем функции ***добавления элемента, вставки, удаления, обхода списка в обоих направлениях***.

При выполнении задачи опирайтесь на лекцию 19 про односвязный список, а также на справочные ресурсы, например,

<https://prog-cpp.ru/data-dcs/>

Будьте готовы объяснить вашу реализацию.

## **задача №4 «Стеганография»**

Воспользуйтесь классом для работы с BMP-изображениями и зашифруйте произвольную текстовую информацию.

## **задача №5 «Поиск ошибок»**

Найти все ошибки в приведенном коде. Предложить варианты исправления.

class Foo

{

public:

Foo(int j) { i=new int[j]; }

~Foo() { delete i; }

private:

int\* i;

};

class Bar: Foo

{

public:

Bar(int j) { i=new char[j]; }

~Bar() { delete i; }

private:

char\* i;

};

void main()

{

Foo\* f=new Foo(100);

Foo\* b=new Bar(200);

\*f=\*b;

delete f;

delete b;

}

# **Глава 3. Выполнение ДОПОЛНИТЕЛЬНЫх Заданий на практику**

**Специально для размещения решений мною был создан публичный репозиторий на платформе GitHub (https://github.com/udogaL2/additional\_s2)**

## **Дополнительная задача №1**

**Ниже приведена рекурсивная функция, написанная мной для решения поставленной задачи**

#include <iostream>

**int** sumDig(**int** num) {

**if** (num < 10)

**return** num;

**else** {

**return** num % 10 + sumDig(num / 10);

}

}

**int** **main**() {

**int** num;

// while (true) {

std::**cin** >> num;

std::**cout** << sumDig(num) << '\n';

// }

**return** 0;

}

Полное решение задачи можно увидеть по ссылке: <https://github.com/udogaL2/additional_s2/tree/main/lab1_recursion>

## **Дополнительная задача №2**

**Здесь представлена моя реализация сортировки Шелла**

#include <iostream>

// O(n^2) - худшее

// O(n\*log^2n) - лучшее

**void** sortShell(**int** n, **int** \*mas) {

**int** k = n / 2;

**while** (k > 0) {

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

**int** j = i;

**while** (j >= 0 **and** mas[j] > mas[j + k]) {

**int** temp = mas[j];

mas[j] = mas[j + k];

mas[j+k] = temp;

j--;

}

}

k /= 2;

}

}

**int** **main**() {

**int** m[10] = {1000, -3, 4, 3, 34, 5, -2, 3, 12,-1};

sortShell(10, m);

**for**(**auto** n : m){

std::**cout** << n << ' ';

}

**return** 0;

}

Полное решение задачи можно увидеть по ссылке: https://github.com/udogaL2/additional\_s2/tree/main/lab2\_sort

## **Дополнительная задача №3**

Для решения задачи была использована структура ноды, в которой хранится информация, а также класс списка, в котором реализованы функции добавления, удаления и вывода:

#include <iostream>

#include <exception>

**template**<**typename** T>

**struct** Node {

T data;

Node \*next, \*prev;

};

**template**<**typename** T>

**class** List {

**public**:

List() {

Head = **nullptr**;

Tail = **nullptr**;

**count** = 0;

}

~List() {

**while** (Head) {

Tail = Head->next;

**delete**[] Head;

Head = Tail;

}

}

**void** add(T data) {

**auto** \*temp = **new** Node<T>;

temp->next = temp->prev = **nullptr**;

temp->data = data;

**if** (Head != **nullptr**) {

temp->prev = Tail;

Tail->next = temp;

Tail = temp;

} **else** {

temp->prev = **nullptr**;

Head = Tail = temp;

}

**count**++;

}

**void** addHead(T data) {

**auto** \*temp = **new** Node<T>;

temp->prev = **nullptr**;

temp->data = data;

temp->next = Head;

**if** (Head != **nullptr**)

Head->prev = temp;

**if** (**count** == 0)

Head = Tail = temp;

**else**

Head = temp;

**count**++;

}

**void** insert(**int** pos, T data) {

**if** (pos < 0 **or** pos > **count**) {

**throw** std::**exception**();

}

**if** (pos == 0) {

addHead(data);

**return**;

} **else** **if** (pos == **count**) {

add(data);

**return**;

}

**auto** \*ins\_el = Head;

**if** (**count** / 2 >= pos) {

std::**cout** << "Head\n";

ins\_el = Head;

**int** c = 0;

**while** (c < pos) {

ins\_el = ins\_el->next;

c++;

}

} **else** {

std::**cout** << "Tail\n";

ins\_el = Tail;

**int** c = **count** - 1;

**while** (c > pos) {

ins\_el = ins\_el->prev;

c--;

}

}

**auto** \*prev\_temp = ins\_el->prev;

**auto** \*temp = **new** Node<T>;

temp->data = data;

**if** (**count** != 1)

prev\_temp->next = temp;

temp->next = ins\_el;

temp->prev = prev\_temp;

ins\_el->prev = temp;

**count**++;

}

**void** pop(**int** pos) {

**if** (pos < 0 **or** pos > **count**) {

**throw** std::**exception**();

}

**auto** \*del\_el = Head;

**if** (**count** / 2 >= pos) {

std::**cout** << "Head\n";

del\_el = Head;

**int** c = 0;

**while** (c < pos) {

del\_el = del\_el->next;

c++;

}

} **else** {

std::**cout** << "Tail\n";

del\_el = Tail;

**int** c = **count** - 1;

**while** (c > pos) {

del\_el = del\_el->prev;

c--;

}

}

**auto** \*prev\_del = del\_el->prev;

**auto** \*after\_del = del\_el->next;

// Если удаляем не голову

**if** (prev\_del != **nullptr** **and** **count** != 1)

prev\_del->next = after\_del;

// Если удаляем не хвост

**if** (after\_del != **nullptr** **and** **count** != 1)

after\_del->prev = prev\_del;

**if** (pos == 0)

Head = after\_del;

**if** (pos == **count** - 1)

Tail = prev\_del;

**count**--;

}

T get(**int** pos){

**if** (pos < 0 **or** pos > **count**) {

**throw** std::**exception**();

}

**auto** \*get\_el = Head;

**if** (**count** / 2 >= pos) {

std::**cout** << "Head\n";

get\_el = Head;

**int** c = 0;

**while** (c < pos) {

get\_el = get\_el->next;

c++;

}

} **else** {

std::**cout** << "Tail\n";

get\_el = Tail;

**int** c = **count** - 1;

**while** (c > pos) {

get\_el = get\_el->prev;

c--;

}

}

**return** get\_el->data;

}

**void** print() {

**auto** \*temp = Head;

**while** (temp != **nullptr**) {

std::**cout** << temp->data << '\n';

temp = temp->next;

}

}

**void** invPrint() {

**auto** \*temp = Tail;

**while** (temp != **nullptr**) {

std::**cout** << temp->data << '\n';

temp = temp->prev;

}

}

**private**:

Node<T> \*Head, \*Tail;

**int** **count**;

};

**int** **main**() {

List<**int**> t;

t.add(10);

t.add(11);

t.add(13);

t.print();

std::**cout** << '\n';

t.insert(2, 12);

t.print();

std::**cout** << '\n';

t.invPrint();

std::**cout** << '\n';

std::**cout** << t.get(3) << " \n";

std::**cout** << '\n';

t.pop(2);

t.print();

std::**cout** << '\n';

**return** 0;

}

Полное решение задачи можно увидеть по ссылке: https://github.com/udogaL2/additional\_s2/tree/main/lab3\_dlist

## **Дополнительная задача №4**

**Пример работы программы:**

При шифровании изображения указывается цвет пикселя, с которым необходимо взаимодействовать.  
Исходное изображение:

****

Рисунок 1

Зашифрованное изображение:



Дешифрованное изображение:



Полное решение задачи можно увидеть по ссылке: https://github.com/udogaL2/additional\_s2/tree/main/lab4\_bmp

## **Дополнительная задача №5**

Исправленный класс с использованием полиморфизма:

#include <iostream>

**template** <**typename** T>

**class** Base{

**public**:

**explicit** Base(**int** j){

i = **new** T[j];

}

~Base(){

std::**cout** << "dest\n";

**delete** i;

}

**private**:

T\* i;

};

**int** **main**() {

**auto**\* f =**new** Base<**int**>(100);

**auto**\* g =**new** Base<**char**>(100);

**delete** f;

// f = reinterpret\_cast<Base<int> \*>(g);

**delete** g;

**return** 0;

}

Полное решение задачи можно увидеть по ссылке: https://github.com/udogaL2/additional\_s2/tree/main/lab5\_oop

# **Заключение**

С++ – язык программирования, который поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником – языком C, наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Смысл философии языка C++ можно определить выражением "доверять программисту". Например, компилятор не будет вам мешать сделать что-то новое, что имеет смысл, но также не будет мешать вам сделать что-то такое, что может привести к сбою. Это одна из главных причин, почему так важно знать то, что мы не должны делать, как и то, что мы должны делать, создавая программы на языке С++.

* Закрепление и углубление теоретических знаний по программированию;
* Приобретение и развитие первичных профессиональных навыков и умений по прикладной математике и информатике;
* Получение основ практического применения знаний в сфере ООП;
* Получение практических навыков работы с локальными и онлайн репозиториями путём применения платформы Github;
* Реализация контроля версий с помощью систем Git;
* Получение практических навыков, связанных с отладкой и устранением ошибок путём использования CLion IDE;
* Получение теоретических знаний и практических навыков, напрямую связанных с оптимизацией и рационализацией алгоритмов работы программы на языке C++ ;
* Применение полученных знаний и навыков для решения нетипичных задач;
* Написание, сравнение и использование различных видов сортировки больших массивов элементов для выявление оптимальных методов для дальнейшего использования;
* Создание шаблонов для потенциального использования в последующих проектах;
* Решение стандартных задач профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;
* Структурирование уже имеющихся знаний и выполнение с их использованием различных задач;
* Освоение на начальном уровне графических редакторов для создания графических объектов требующихся для решения задач проекта.

В течение практики все задачи были выполнены, а цели достигнуты.

# **Список литературы**

**Перечень учебной литературы ресурсов сети «Интернет», необходимой для проведения практики**

1. Варфоломеева, Т. Н. Структуры данных и основные алгоритмы их обработки : учебное пособие / Т. Н. Варфоломеева. - Москва : ФЛИНТА, 2017. - 159 с. - ISBN 978-5-9765-3691-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1860018 (дата обращения: 11.01.2022). – Режим доступа: по подписке.
2. Гданский, Н. И. Основы теории и алгоритмы на графах : учебное пособие / Н. И. Гданский. — Москва : ИНФРА-М, 2020. — 206 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-014386-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/978686 (дата обращения: 11.01.2022). – Режим доступа: по подписке.
3. Затонский, А. В. Программирование и основы алгоритмизации. Теоретические основы и примеры реализации численных методов: учебное пособие / А.В. Затонский, Н.В. Бильфельд. — 2-е изд. — Москва: РИОР : ИНФРА-М, 2022. — 167 с. — (Высшее образование). — DOI: https: //www.dx.doi.org/10.12737/20468. - ISBN 978-5-369-01195-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1860435 (дата обращения: 16.02.2022). – Режим доступа: по подписке.
4. Воронцова, Е. А. Программирование на С++ с погружением: практические задания и примеры кода - Москва: НИЦ ИНФРА-М, 2016. - 80 с. ISBN 978-5-16-105159-7. - Текст: электронный. - URL: https://znanium.com/catalog/product/563294 (дата обращения: 16.02.2022). – Режим доступа: по подписке.

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

1. Колдаев, В. Д. Структуры и алгоритмы обработки данных [Электронный ресурс]: учеб. пособие для вузов/ В. Д. Колдаев. - Москва: РИОР; Москва: ИНФРА-М, 2014. - 1 эл. опт. диск (CD-ROM), 294 с.: ил., табл. - (Высшее образование - бакалавриат). - Библиогр.: с. 285. - Лицензия до 23.06.2020 г. - ISBN 978-5-369-01264-2. - ISBN 978-5-16-009012-2: 15100.00 р. Имеются экземпляры в отделах /There are copies in departments: всего /all 2: ЭБС Кантиана(1), ч.з.N1(1) Свободны / free: ЭБС Кантиана(1), ч.з.N1(1)
2. Основы программирования. – режим доступа: http://www.intuit.ru/studies/courses/648/504/info
3. Видеолекции по курсу «Основы программирования». – режим доступа: http://www.youtube.com/watch?v=pxR3UoO9c9w
4. Сайт «Структуры и алгоритмы». – режим доступа: http://www.structur.h1.ru/
5. Структуры и алгоритмы обработки данных: Учебное пособие. – режим доступа: http://window.edu.ru/resource/820/44820

# **Приложения**

**Приложение 1**

class Rectangle {

public:

int length;

int breadth;

int getArea() { return length \* breadth; }

};

**Приложение 2**

#include <iostream>

using namespace std;

class Rectangle {

public:

int length;

int breadth;

Rectangle(int len, int brth) : length(len), breadth(brth) {}

int getArea()

{ return length \* breadth; }

};

int main() {

Rectangle rect(8, 6);

cout << "Area = " << rect.getArea();

return 0;

}

**Приложение 3**

class Rectangle {

private:

int age;

public:

void setLength(int len) {

if (len >= 0)

length = len;

}

};

**Приложение 4**

class Animal {

// eat() function

// sleep() function

};

class Dog : public Animal {

// bark() function

};

**Приложение 5**

#include <iostream>

#include <string>

using namespace std;

// base class

class Animal {

private:

string color;

protected:

string type;

public:

void eat()

{ cout << "I can eat!" << endl; }

void sleep()

{ cout << "I can sleep!" << endl; }

void setColor(string clr)

{ color = clr; }

string getColor()

{ return color; }

};

class Dog : public Animal {

public:

void setType(string tp)

{ type = tp; }

void displayInfo(string c) {

cout << "I am a " << type << endl;

cout << "My color is " << c << endl;

}

void bark()

{ cout << "I can bark! Woof woof!!" << endl; }

};

int main() {

Dog dog1;

dog1.eat();

dog1.sleep();

dog1.setColor("black");

dog1.bark();

dog1.setType("mammal");

dog1.displayInfo(dog1.getColor());

return 0;

}

**Приложение 6**

#include <iostream>

using namespace std;

class Polygon {

protected:

int width, height;

public:

void set\_values (int a, int b)

{ width=a; height=b; }

};

class Rectangle: public Polygon {

public:

int area()

{ return width\*height; }

};

class Triangle: public Polygon {

public:

int area()

{ return width\*height/2; }

};

int main () {

Rectangle rect;

Triangle trgl;

Polygon \* ppoly1 = &rect;

Polygon \* ppoly2 = &trgl;

ppoly1->set\_values (4,5);

ppoly2->set\_values (4,5);

cout << rect.area() << '\n';

cout << trgl.area() << '\n';

return 0;

}

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ДНЕВНИК

УЧЕБНО-ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

г. Калининград 2022 г.

**1.Информационная часть**

Шатура Евгений Дмитриевич, студент очной формы обучения 1 курса группы 03\_ПМ\_21\_О\_ИвП направления подготовки 01.03.02 Прикладная математика и информатика профиль обучения «Информатика и программирование» в соответствии с приказом от 22 декабря 2021 г. № 4888ст направляется на учебно-технологическую (проектно-технологическую) практику в Институт физико-математических наук и информационных технологий.

Период практики – с 17.01.2022 г. по 02.07.2022 г.

Руководитель практики от университета – старший преподаватель Синюхин А. О.

Институт физико-математических наук и информационных технологий

Контактный номер телефона +7 (4012) 338 217

Первый заместитель

директора ОНК «ИВТ» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шпилевой А.А.

**2. Программа практики**

**2.1. План работы**

|  |  |  |
| --- | --- | --- |
| **№**  **п.п.** | **Рабочее место практиканта** | **Продолжительность (в днях)** |
| 1. | Компьютерный класс (аудитория 312) | 18 |

**2.2. Индивидуальное задание по профилю подготовки/специальности**

1. Пройти инструктаж по технике безопасности.
2. Ознакомиться и выполнить задачи на практику.
3. Написать отчет по практике.

Руководитель практики

Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Синюхин А. О.

**3. Ход выполнения практики**

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  **п.п.** | **Дата** | **Описание выполненной работы** | **Отметки руководителя практики** |
| 1 | 17.01.2022 | Инструктаж по технике безопасности | **1** |
| 2 | 24.01.2022 | Дополнительная задача №1 | **0** |
| 3 | 31.01.2022 | Дополнительная задача №1 | **1** |
| 4 | 07.02.2022 | Дополнительная задача №2 | **0** |
| 5 | 14.02.2022 | Дополнительная задача №2 | **0** |
| 6 | 21.02.2022 | Дополнительная задача №2 | **1** |
| 7 | 28.02.2022 | Дополнительная задача №3 | **0** |
| 8 | 07.03.2022 | Дополнительная задача №3 | **0** |
| 9 | 14.03.2022 | Дополнительная задача №3 | **1** |
| 10 | 21.03.2022 | Дополнительная задача №4 | **0** |
| 11 | 28.03.2022 | Дополнительная задача №4 | **0** |
| 12 | 04.04.2022 | Дополнительная задача №4 | **0** |
| 13 | 11.04.2022 | Дополнительная задача №4 | **0** |
| 14 | 18.04.2022 | Дополнительная задача №4 | **1** |
| 15 | 25.04.2022 | Дополнительная задача №5 | **1** |
| 16 | 02.05.2022 | Дополнительная задача №5 | **0** |
| 17 | 09.05.2022 | Дополнительная задача №5 | **0** |
| 18 | 16.05.2022 | Дополнительная задача №5 | **1** |

**3.1 Шкала баллов за выполненные задания работы**

|  |  |  |
| --- | --- | --- |
| **№**  **п.п.** | **Наименование работы** | **Максимально возможное количество баллов** |
| 1 | Дополнительная задача №1 | 1 |
| 2 | Дополнительная задача №2 | 1 |
| 3 | Дополнительная задача №3 | 1 |
| 4 | Дополнительная задача №4 | 1 |
| 5 | Дополнительная задача №5 | 1 |

**Учебная практика (для 1ПМ,** Синюхин А. О.)

Пусть количество задач (папка Additional) равно M. Тогда оценки за учебную практику выставляются следующим образом:

1. Сдано M задач – оценка 5.
2. Сдано M-1 задачи – оценка 4.
3. Сдано M-2, M-3 задач – оценка 3.
4. Сдано <M-3 задач – зачета нет.

**4. Отзыв руководителя практики**

Шатура Евгений Дмитриевич, студент очной формы обучения 1 курса группы 03\_ПМ\_21\_О\_ИвП направления подготовки 01.03.02 Прикладная математика и информатика профиль обучения «Информатика и программирование» в соответствии с приказом от 22 декабря 2021 г. № 4888ст направлен на учебно-технологическую (проектно-технологическую) практику в Институт физико-математических наук и информационных технологий.

Период практики – с 17.01.2022 г. по 02.07.2022 г.

Программа практики и индивидуальное задание на практику выполнены. Отчёт по практике сдан и защищён на отчётной конференции.

Студент Шатура Евгений Дмитриевич в процессе прохождения практики справился с поставленными задачами, приобрёл первичные профессиональные навыки и компетенции, в том числе:

* 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.

Учебная практика оценена на оценку \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики

Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Синюхин А. О.

«18» июня 2022 г.