Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

**Отчет по учебной практике**

«АНАЛИТИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ ПОЛИНОМОВ ОТ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ (СПИСКИ)»

**Выполнил:** студент группы 381303-3

Клименко Ксения Евгеньевна

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ Лебедев Илья Геннадьевич

Нижний Новгород

2021

**Содержание**

[Введение 3](#_Toc59276982)

[1. Постановка задачи 4](#_Toc59276983)

[2. Руководство пользователя 5](#_Toc59276984)

[3. Руководство программиста 7](#_Toc59276985)

[3.1. Структура программы 7](#_Toc59276986)

[3.2. Описание алгоритма 7](#_Toc59276987)

[3.2.1. Структура данных Список 7](#_Toc59276988)

[3.2.2. Полином 9](#_Toc59276989)

[3.3. Описание структур данных и функций 12](#_Toc59276990)

[3.3.1. Класс TMList 12](#_Toc59276991)

[3.3.2. Класс TMonomial 13](#_Toc59276992)

[3.3.3. Класс TPolynomial 14](#_Toc59276993)

[3.3.4. Основная программа 17](#_Toc59276994)

[Эксперименты 18](#_Toc59276995)

[Заключение 20](#_Toc59276996)

[Литература 21](#_Toc59276997)

# Введение

Многочленом (полиномом) называется выражение вида где – элементы некоторого поля , буква, коэффициенты полинома, старший коэффициент. Если , то число называется степенью многочлена.

В число основных операций над полиномами входят действия по вычислению значений полинома при заданных значениях переменных, а также большинство известных математических операций (сложение, вычитание, вычисление частных производных, интегрирование и т.п.).

В повседневной жизни человеку постоянно необходимо выполнять вычисления различных арифметических выражений разного уровня сложности как в прикладных, так и в научных целях. Для того чтобы избежать ошибок, можно поручить это компьютеру с помощью специальных приложений.

Таким образом, для этого и было создано консольное приложение, работающее с полиномами от нескольких переменных. При этом используется динамическая структура данных Список, на практическое освоение которой и направлена данная лабораторная работа.

# Постановка задачи

**Задача:** разработать и реализовать приложение, выполняющее алгебраические операции над полиномами.

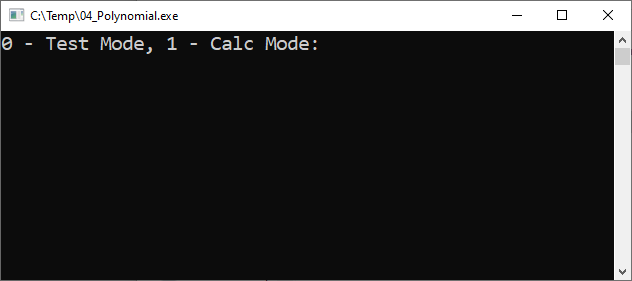
**Входные данные:** полиномы, операции (в виде строки).

**Выходные данные:** полиномы (в виде строки).

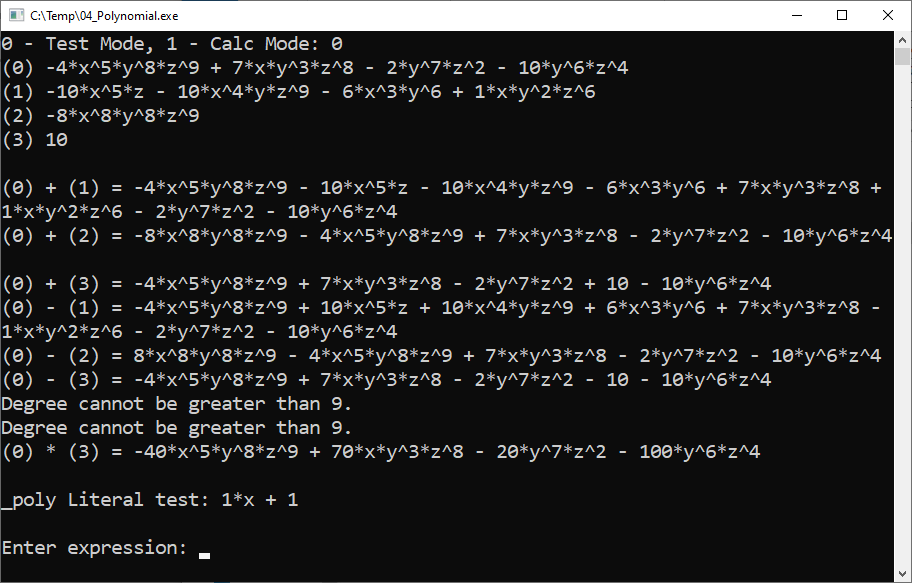
# Руководство пользователя

В данном руководстве содержатся пошаговые инструкции по работе с программой, для того чтобы вы могли как можно быстрее приступить к использованию приложения.

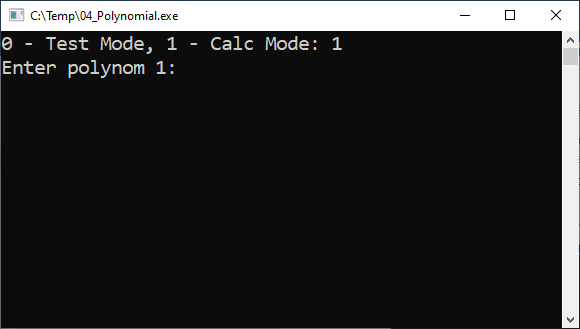
1. Запустите файл **polynomial.exe** из папки с программой. Перед вами отобразится приветственный экран с предложением выбрать режим работы. Введите 0, если хотите запустить автоматическое тестирование, или 1, чтобы перейти в режим ручного ввода, и нажмите Enter.



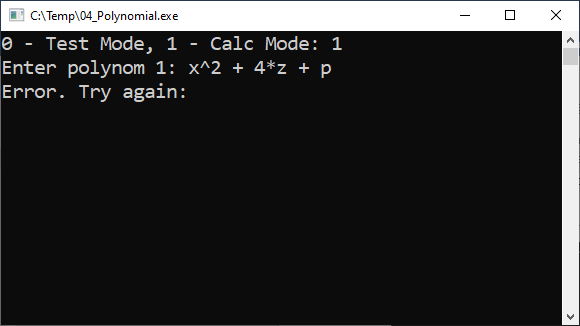
1. Программа после запуска.
2. Если вы выбрали режим автоматического тестирования, то программа сгенерирует несколько полиномов, моном и число и выполнит различные действия над ними, а результат будет выведен на экран (см. Рис. 2). Вы также можете ввести строковое представление полинома с клавиатуры и нажать Enter, чтобы проверить работу алгоритма преобразования строки. Правила ввода следующие: знак \* (между коэффициентом и переменными) не опускать, степени вводить не меньше 1 и не больше 9 (если при какой-либо переменной нужно ввести степень 0, то пропустите эту переменную). Чтобы сменить режим, перезапустите программу.
3. После выбора режима ручного ввода программа предложит вам ввести полиномы. В случае неверного ввода полиномов вы увидите сообщение об ошибке с предложением повторить ввод.
4. Если все введено правильно, программа выведет результаты выполнения операций. Нажмите Enter для выхода из программы.



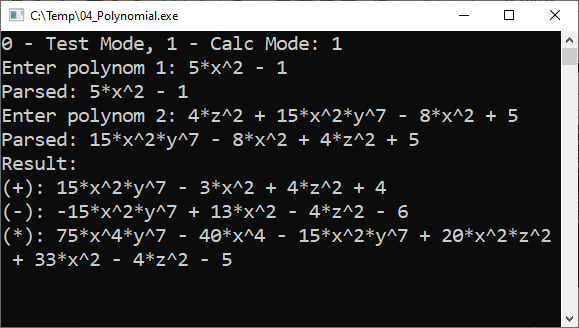
1. Программа в режиме автоматического тестирования.



1. Программа в режиме ручного ввода.



1. Сообщение об ошибке при вводе некорректного выражения (полинома).



1. Отображение результатов вычислений

# Руководство программиста

## Структура программы

* **TMList.h** – модуль Моном (содержит объявление и реализацию класса для работы со структурой данных Список, включая классы узлов;
* **TMonomial.h** – модуль Моном (содержит объявление и реализацию специализации узла списка, используемого в модуле выше, включая специальные классы исключений);
* **TPolynomial.h, TPolynomial.cpp** – модуль Полином (содержит объявление и реализацию класса для работы с полиномами);
* **main.cpp** – основной модуль (консольное приложение).

## Описание алгоритма

### Структура данных Список

Линейный односвязный список – базовая динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и указатель на следующий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Для работы со списками предлагается реализовать следующие операции:

* методы проверки на пустоту и проверки на полноту списка;
* методы навигации по списку (итератор);
* методы вставки в начало, в конец, после звена с заданным ключи и перед звеном с заданным ключом;
* методы удаления звена с заданным ключом и поиск звена с заданным ключом.

#### Вставка звена в начало

temp

pFirst

1. Вставка звена в начало.
2. Создать новое звено temp.
3. Установить указатель на следующее звено у temp на pFirst.
4. Поменять указатель на начало списка pFisrt на temp.

#### Вставка звена в конец

temp

pFirst

1. Вставка звена temp в конец.
2. Создать новое звено temp.
3. По списку дойти до последнего звена.
4. Установить указатель на следующее звено у последнего на temp.
5. Установить указатель на следующее звено у temp на nullptr.

#### Вставка звена после заданного звена

temp

pFirst

1. Вставка звена temp после заданного звена.
2. По списку дойти до заданного звена.
3. Создать новое звено temp.
4. Установить указатель на следующее звено у temp на звено, следующее за заданным.
5. Установить указатель на следующее звено у заданного звена на temp.

#### Вставка звена перед заданным звеном

temp

pFirst

1. Вставка звена temp перед заданным звеном.
2. По списку дойти до звена, предшествующего заданному.
3. Создать новое звено temp.
4. Установить указатель на следующее звено у temp на заданное звено.
5. Установить указатель на следующее звено у звена, предшествующего заданному, на temp.

#### Удаление заданного звена

pFirst

1. Удаление заданного звена.
2. По списку дойти до звена, предшествующего заданному.
3. Установить указатель на следующее звено у звена, предшествующего заданному, на следующее за заданным звено, а заданное звено удалить.

### Полином

Полином можно определить также как выражение из нескольких термов, соединенных знаками сложения или вычитания. Терм включает коэффициент и моном, содержащий одну или несколько переменных, каждая из которых может иметь степень. При выполнении данной лабораторной работы предполагается, что степени переменных в мономах могут принимать значения 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Для организации быстрого доступа может быть использовано упорядоченное хранение мономов. Для задания порядка следования можно принять лексикографическое упорядочивание по степеням переменных, при котором мономы упорядочиваются по степеням первой переменной, потом по второй переменной, и только затем по третьей переменной. В общем виде это правило можно записать как соотношение: моном предшествует моному тогда и только тогда, если

(A1 > A2) ˅ (A1 = A2) & (B1 > B2) ˅ (A1 = A2) & (B1 = B2) & (C1 > C2).

Проверка лексикографического порядка занимает сравнительно много времени. Ее можно существенно упростить при помощи свернутой степени (индекса) монома, образуемой с использованием позиционной системы счисления: для монома со степенями (A, B, C) ставится в соответствие величина

ABC = A \* 100 + B \* 10 + C.

Данное соответствие является взаимно-однозначным. Обратное соответствие определяется при помощи выражений

A= [ABC / 100], B= [ABC % 100 / 10], C=ABC % 10.

Кроме того, введенное соответствие порождает порядок, полностью совпадающий с лексикографическим порядком

> ‎‎↔ ABC1 > ABC2.

Выполненное обсуждение позволяет определить, что наиболее эффективным способом организации структуры хранения полиномов являются линейный (односвязный) список. Тем самым, в рамках лабораторной работы появляется подзадача – разработка структуры хранения в виде линейных списков. Данная разработка должна быть выполнена в некоторой общей постановке с тем, чтобы разработанные программы работы со списками могли быть далее использованы и в других ситуациях, в которых необходимы списковые структуры хранения.

Для работы с полиномами предлагается реализовать следующие операции:

* конструкторы инициализации и копирования;
* метод присваивания;
* методы сложения, вычитания, умножения полиномов.

Ниже более подробно рассматриваются алгоритмы для выполнения указанных методов.

#### Сложение двух полиномов (на линейных списках)

Пусть необходимо выполнить операцию P + Q (P, Q – полиномы от трех переменных, хранящиеся в виде линейного списка). В результирующий полином R копируются мономы из P. Далее для каждого монома в полиноме Q выполняется поиск монома такой же степени в полиноме R. Если такой моном найден, то их коэффициенты складываются, иначе выполняется упорядоченная вставка этого монома в полином R.

#### Вычитание двух полиномов (на линейных списках)

Пусть необходимо выполнить операцию P – Q (P, Q – полиномы от трех переменных, хранящиеся в виде линейного списка). В результирующий полином R копируются мономы из P. Далее для каждого монома в полиноме Q выполняется поиск монома такой же степени в полиноме R. Если такой моном найден, то их коэффициенты вычитаются, иначе выполняется упорядоченная вставка этого монома (с коэффициентом, умноженным на -1) в полином R.

#### Умножение двух полиномов (на линейных списках)

Пусть необходимо выполнить операцию P \* Q (P, Q – полиномы от трех переменных, хранящиеся в виде линейного списка). Создается пустой результирующий полином R. Далее для каждого монома в полиноме P выполняется умножение с каждым мономом из полинома Q, после чего результаты складываются и упорядоченно вставляются в полином R.

## Описание структур данных и функций

### Класс TMList

Объявление класса для работы со списком выглядит следующим образом:

class TMList

{

protected:

TMonomial\* root;

TMonomial\* end;

int count;

public:

TMList();

TMList(const TMList& other);

~TMList();

TMList& operator=(const TMList& other);

bool IsEmpty() const;

void InsFirst(const TMonomial& e);

void InsLast(const TMonomial& e);

void InsAfter(TMonomial\* insAfterThis, const TMonomial& e);

TMonomial\* GetFirst() const;

TMonomial\* GetLast() const;

int GetCount() const;

void DelFirst();

void DelLast();

void Del(TMonomial\* e);

friend std::ostream& operator<<(std::ostream& s, const TMList& list)

{

TMonomial\* i = list.root;

while (i != nullptr)

{

s << i->data << ' ';

i = i->next;

}

return s;

}

};

#### Поля класса

TMonomial\* root; – корневой узел;

TMonomial\* end; – последний узел;

int count; – число узлов.

#### Методы класса

TMList(); – конструктор по умолчанию;

TMList(const TMList& other); – конструктор копирования;

~TMList(); – деструктор;

TMList& operator=(const TMList& other); – оператор присваивания;

bool IsEmpty() const; – проверка на пустоту;

void InsFirst(const TMonomial& e); – вставка элемента в начало списка;

void InsLast(const TMonomial& e); – вставка элемента в конец списка;

void InsAfter(TMonomial\* insAfterThis, const TMonomial& e); – вставка элемента после заданного элемента;

TMonomial\* GetFirst() const; – получение первого узла;

TMonomial\* GetLast() const; – получение последнего узла;

int GetCount() const; – получение количества узлов;

void DelFirst(); – удаление элемента из начала списка;

void DelLast(); – удаление элемента из конца списка;

void Del(TMonomial\* e); – удаление произвольного элемента;

friend std::ostream& operator<<(std::ostream& s, const TMList& list); – оператор вывода.

### Класс TMonomial

class TMonomial

{

void CheckDims(const TMonomial& other) const

public:

int\* data;

int dim;

double k;

TMonomial\* prev;

TMonomial\* next;

TMonomial(double k\_ = 0,

int dim\_ = 0,

TMonomial\* prev\_ = nullptr,

TMonomial\* next\_ = nullptr)

TMonomial(const TMonomial& other);

TMonomial(const TMonomial\* other, TMonomial\* prev\_, TMonomial\* next\_);

~TMonomial();

TMonomial operator+(const TMonomial& other) const;

TMonomial operator-(const TMonomial& other) const;

TMonomial operator\*(const TMonomial& other) const;

TMonomial operator\*(double number) const;

int& operator[](int index);

friend std::ostream& operator<<(std::ostream& s, const TMonomial& m);

};

#### Поля класса

int\* data; – массив степеней монома;

int dim; – размерность (количество переменных в мономе);

double k; – коэффициент при мономе;

TMonomial\* prev; – указатель на предыдущий моном;

TMonomial\* next; – указатель на следующий моном.

#### Методы класса

TMonomial(double k\_ = 0, int dim\_ = 0, TMonomial\* prev\_ = nullptr, TMonomial\* next\_ = nullptr) – конструктор инициализации;

TMonomial(const TMonomial& other); – конструктор копирования;

TMonomial(const TMonomial\* other, TMonomial\* prev\_, TMonomial\* next\_); – конструктор инициализации;

~TMonomial(); – деструктор;

TMonomial operator+(const TMonomial& other) const; – сложение мономов;

TMonomial operator-(const TMonomial& other) const; – вычитание мономов;

TMonomial operator\*(const TMonomial& other) const; – умножение мономов;

TMonomial operator\*(double number) const; – умножение монома на число;

int& operator[](int index); – получение степени переменной по индексу;

friend std::ostream& operator<<(std::ostream& s, const TMonomial& m); – оператор вывода.

### Класс TPolynomial

class TPolynomial

{

TMList m;

bool IsDimMoreOrEqual(const TMonomial\* m1, const TMonomial\* m2) const;

bool IsDimLessOrEqual(const TMonomial\* m1, const TMonomial\* m2) const;

bool IsDimEqual(const TMonomial\* m1, const TMonomial\* m2) const;

TMonomial\* FindSameDim(const TMonomial\* monomial) const;

TMonomial\* FindPrevOrdered(const TMonomial\* monomial) const;

void Add(const TMonomial\* monomial);

void AddNonzero(TMonomial\* primary, const TMonomial\* secondary);

void Reduce();

public:

TPolynomial();

TPolynomial(const TMonomial& monomial);

TPolynomial(const TPolynomial& other);

TPolynomial(double number);

~TPolynomial();

TPolynomial& operator=(const TPolynomial& other);

TPolynomial operator+(const TPolynomial& other);

TPolynomial operator+(const TMonomial& monomial);

TPolynomial operator+(double number);

TPolynomial operator-(const TPolynomial& other);

TPolynomial operator-(const TMonomial& monomial);

TPolynomial operator-(double number);

TPolynomial operator\*(const TPolynomial& other);

TPolynomial operator\*(const TMonomial& monomial);

TPolynomial operator\*(double number);

TPolynomial& operator+=(const TPolynomial& other);

TPolynomial& operator+=(const TMonomial& monomial);

TPolynomial& operator+=(double number);

TPolynomial& operator-=(const TPolynomial& other);

TPolynomial& operator-=(const TMonomial& monomial);

TPolynomial& operator-=(double number);

TPolynomial& operator\*=(const TPolynomial& other);

TPolynomial& operator\*=(const TMonomial& monomial);

TPolynomial& operator\*=(double number);

friend std::ostream& operator<<(std::ostream& s, const TPolynomial& p);

};

#### Поля класса

TMList m; – список мономов.

#### Методы класса

bool IsDimMoreOrEqual(const TMonomial\* m1, const TMonomial\* m2) const; – сравнение степеней мономов (больше или равны);

bool IsDimLessOrEqual(const TMonomial\* m1, const TMonomial\* m2) const; – сравнение степеней мономов (меньше или равны);

bool IsDimEqual(const TMonomial\* m1, const TMonomial\* m2) const; – сравнение степеней мономов (равны);

TMonomial\* FindSameDim(const TMonomial\* monomial) const; – поиск монома с такой же степенью;

TMonomial\* FindPrevOrdered(const TMonomial\* monomial) const; – поиск монома, который должен находиться в упорядоченном списке перед данным в случае, если бы он там был;

void Add(const TMonomial\* monomial); – вставка монома в упорядоченный список;

void AddNonzero(TMonomial\* primary, const TMonomial\* secondary); – вставка ненулевого монома;

void Reduce(); – приведение подобных слагаемых;

TPolynomial(); – конструктор по умолчанию;

TPolynomial(const TMonomial& monomial); – конструктор преобразования из монома;

TPolynomial(const TPolynomial& other); – конструктор копирования;

TPolynomial(double number); – конструктор преобразования из числа;

~TPolynomial(); – деструктор;

TPolynomial& operator=(const TPolynomial& other); – присваивание полинома;

TPolynomial operator+(const TPolynomial& other); – сложение с полиномом;

TPolynomial operator+(const TMonomial& monomial); – сложение с мономом;

TPolynomial operator+(double number); – сложение с числом;

TPolynomial operator-(const TPolynomial& other); – вычитание с полиномом;

TPolynomial operator-(const TMonomial& monomial); – вычитание с мономом;

TPolynomial operator-(double number); – вычитание с числом;

TPolynomial operator\*(const TPolynomial& other); – умножение с полиномом;

TPolynomial operator\*(const TMonomial& monomial); – умножение с мономом;

TPolynomial operator\*(double number); – умножение с числом;

TPolynomial operator+=(const TPolynomial& other); – сложение с полиномом и присваивание;

TPolynomial operator+=(const TMonomial& monomial); – сложение с мономом и присваивание;

TPolynomial operator+=(double number); – сложение с числом и присваивание;

TPolynomial operator-=(const TPolynomial& other); – вычитание с полиномом и присваивание;

TPolynomial operator-=(const TMonomial& monomial); – вычитание с мономом и присваивание;

TPolynomial operator-=(double number); – вычитание с числом и присваивание;

TPolynomial operator\*=(const TPolynomial& other); – умножение с полиномом и присваивание;

TPolynomial operator\*=(const TMonomial& monomial); – умножение с мономом и присваивание;

TPolynomial operator\*=(double number); – умножение с числом и присваивание;

friend std::ostream& operator<<(std::ostream& s, const TPolynomial& p); – вывод строкового представления полинома в поток.

### Основная программа

void testMode(); – запуск режимов автоматического тестирования;

void testMode(); – запуск режима ручного ввода;

int main(); – основная функция (точка входа).

# Эксперименты

Результаты тестирования программы.

1. Результаты экспериментов.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операция | Введённые данные | Введённые данные |
| 1 | Сложение полиномов |  |  |
| Результат | |
|  | |
| 2 | Сложение полинома с мономом | Введённые данные | Введённые данные |
|  |  |
| Результат | |
|  | |
| 3 | Сложение полинома с числом | Введённые данные | Введённые данные |
|  | 10 |
| Результат | |
|  | |
| 4 | Вычитание полиномов | Введённые данные | Введённые данные |
|  |  |
| Результат | |
|  | |
| 5 | Вычитание полинома с мономом | Введённые данные | Введённые данные |
|  |  |
| Результат | |
|  | |
| 6 | Вычитанием полинома с числом | Введённые данные | Введённые данные |
|  | 10 |
| Результат | |
|  | |
| 7 | Умножение полинома на число | Введённые данные | Введённые данные |
|  | 10 |
| Результат | |
|  | |

# Заключение

В ходе выполнения практической работы «Аналитические преобразования полиномов от нескольких переменных (списки)» было разработано и реализовано консольное приложение для работы с полиномами, включая выполнение алгебраических операций над ними, использующее динамическую структуру данных Список.

# Литература

1. Бьерн Страуструп. Язык программирования C++, 2-е изд: Пер. с англ. — М.: Издательский дом «Бином», 2017. — 1136 с.
2. Герберт Шилдт. C++ для начинающих. Шаг за шагом, 2-е изд: Пер. с англ – М.: Издательский дом «ЭКОМ Паблишерз», 2013. – 640 с.
3. Липпман Стенли Б., Лажойе Жози, Му Барбара Э. Язык программирования C++, базовый курс, 5-е изд: Пер. с англ. – М.: Издательский дом «Диалектика», 2018. – 1118 с.
4. Лабораторный практикум. Барышева И. В., Мееров И. Б., Сысоев А. В., Шестакова Н. В. Учебно-методическое пособие. – Нижний Новгород (ННГУ), 2017. – 105 с.
5. Рабочие материалы к учебному курсу «Методы программирования» (часть 1). Гергель В. П. Нижний Новгород, 2015. – 100 с.