******

**Отчёт по экзамену «Участие в интеграции программных модулей».**

Билет №12.

Выполнил студент: Константинович Алексей

Группа: П1-18

Преподаватели: Наталья Юрьевна Бобкова,

Попов Вячеслав Николаевич

Королев, 2022

**Оглавление**

[IDEF схема программы 3](#_Toc101100066)

[1 Техническое задание 7](#_Toc101100067)

[1.1 Введение 7](#_Toc101100068)

[1.1.1 Наименование программы 7](#_Toc101100069)

[1.2 Краткая характеристика области 7](#_Toc101100070)

[1.3 Основания для разработки 7](#_Toc101100071)

[1.4 Назначение разработки 7](#_Toc101100072)

[1.4.1 Функциональное назначение 7](#_Toc101100073)

[1.4.2 Эксплуатационное назначение 7](#_Toc101100074)

[1.5 Требования к программе или программному изделию 7](#_Toc101100075)

[1.5.1 Требования к функциональным характеристикам 7](#_Toc101100076)

[1.5.2 Отказы из-за некорректных действий оператора 8](#_Toc101100077)

[1.6 Условия эксплуатации 8](#_Toc101100078)

[1.7 Климатические условия эксплуатации 8](#_Toc101100079)

[1.8 Требования к численности и квалификации 9](#_Toc101100080)

[1.8.1 Требования к численности и квалификации персонала 9](#_Toc101100081)

[1.9 Требования к составу и параметрам технических средств 9](#_Toc101100082)

[1.9.1 Состав технических средств: 9](#_Toc101100083)

[1.9.2 Требования к информационной и программной совместимости 9](#_Toc101100084)

[1.9.3 Требования к маркировке и упаковке 9](#_Toc101100085)

[1.9.4 Требования к транспортированию и хранению 9](#_Toc101100086)

[1.9.5 Специальные требования 9](#_Toc101100087)

[1.9.6 Требования к программной документации 9](#_Toc101100088)

[1.10 Технико-экономические показатели 9](#_Toc101100089)

[1.11 Стадии и этапы разработки 10](#_Toc101100090)

[1.12 Порядок контроля и приемки 10](#_Toc101100091)

[2 Руководство оператора 11](#_Toc101100092)

[2.1 Назначение программы 11](#_Toc101100093)

[2.1.1 Функциональное назначение программы 11](#_Toc101100094)

[2.1.2 Эксплуатационное назначение программы 11](#_Toc101100095)

[2.2 Условия выполнения программы 11](#_Toc101100096)

[2.2.1 Минимальный состав аппаратных средств 11](#_Toc101100097)

[2.2.2 Требования к пользователю 11](#_Toc101100098)

[2.3 Выполнение программы 12](#_Toc101100099)

[2.3.1 Загрузка и запуск мобильного приложения 12](#_Toc101100100)

[2.3.2 Завершение работы программы 12](#_Toc101100101)

[2.4 Сообщения оператору 12](#_Toc101100102)

[3 Отладка и тестирование готового программного средства 13](#_Toc101100103)

# IDEF схема программы

Ниже представлены IDEF-схемы программы:

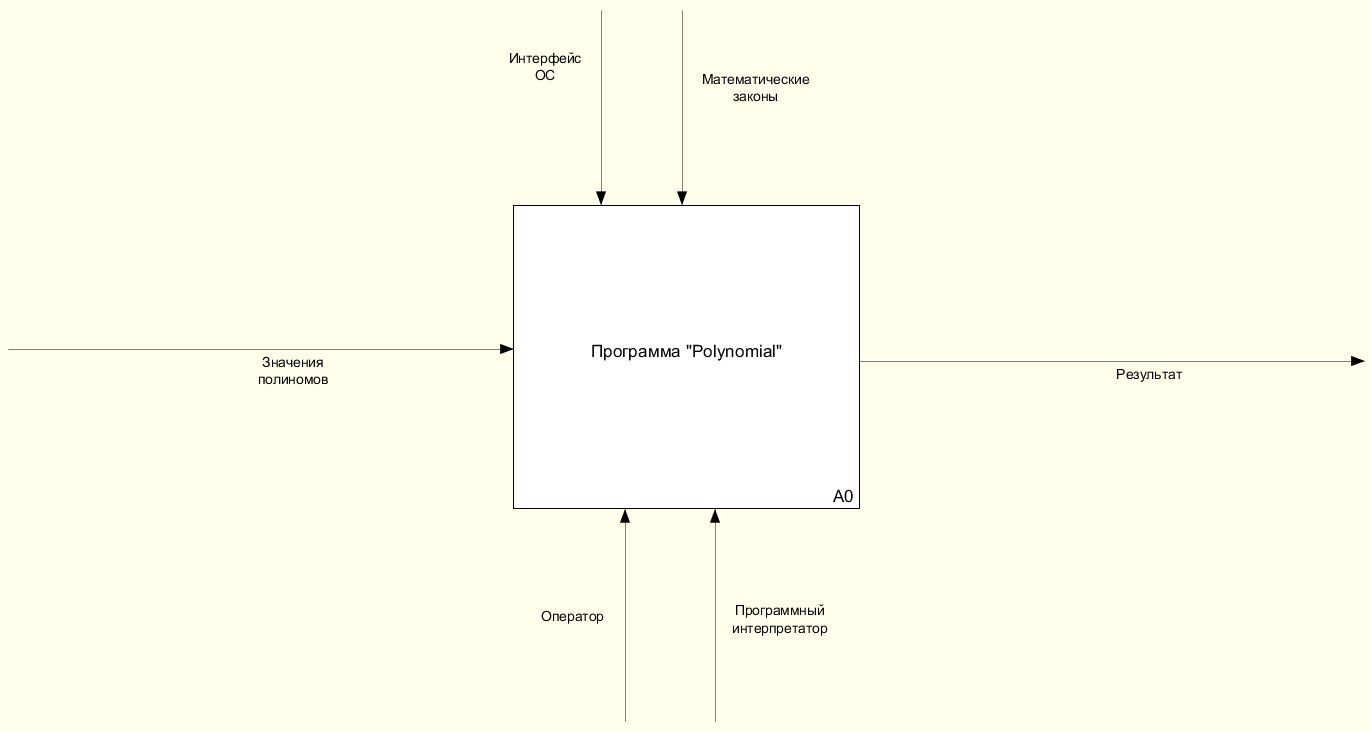


Рисунок 1. Первый уровень IDEF

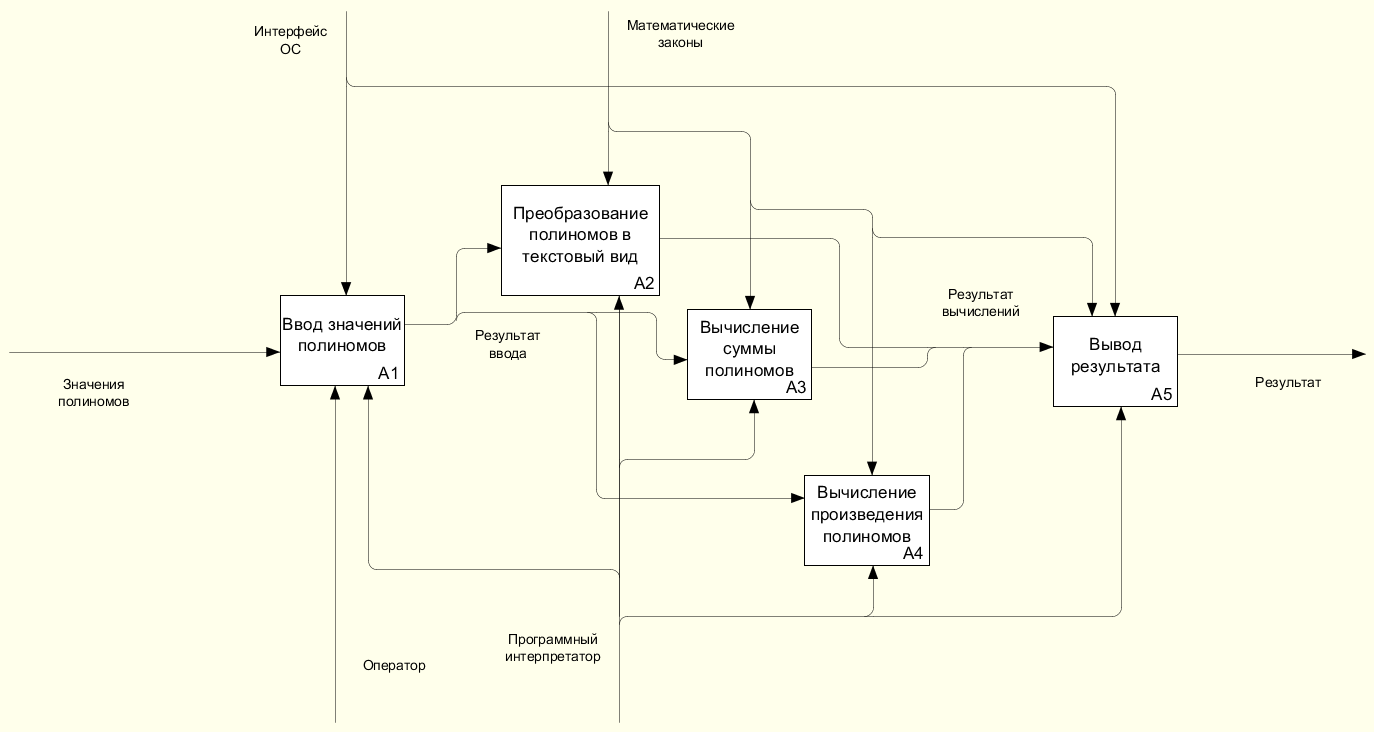


Рисунок 2. Второй уровень IDEF

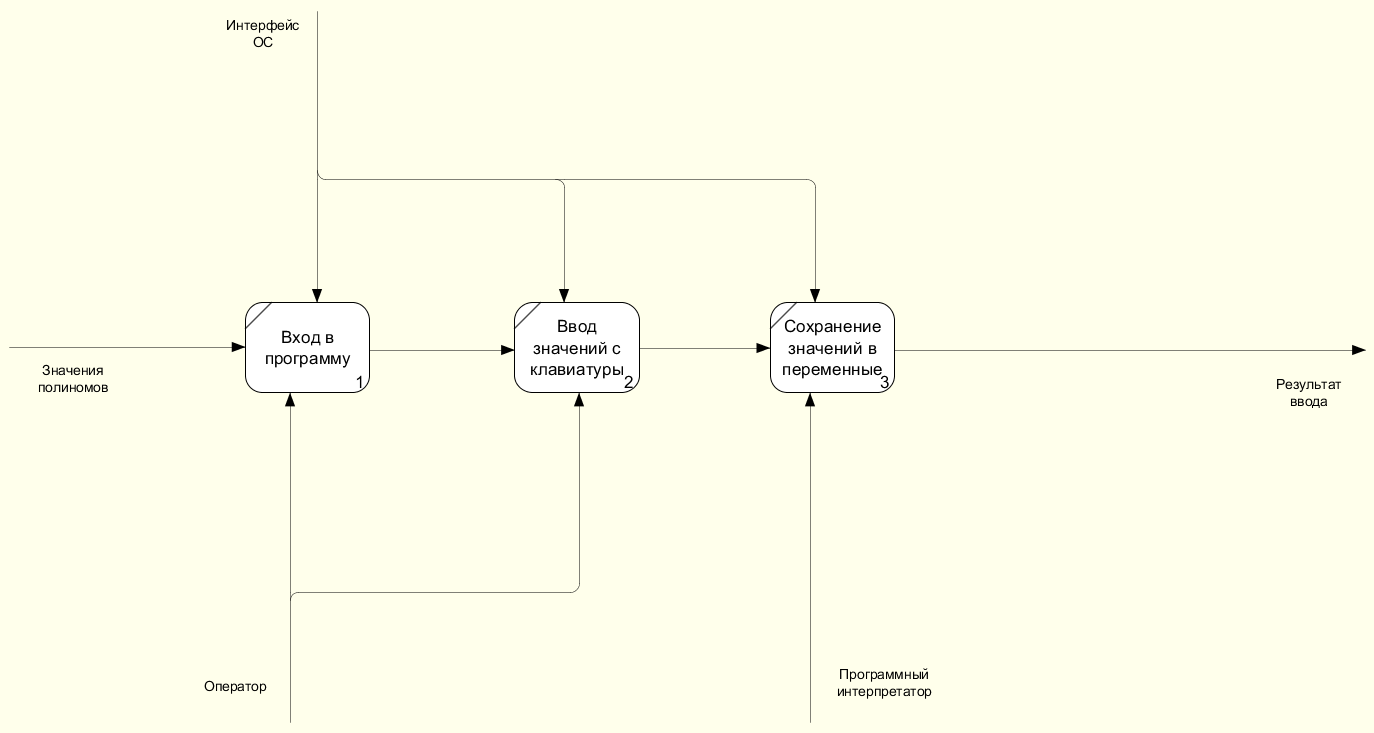


Рисунок 3. Третий уровень IDEF (1)

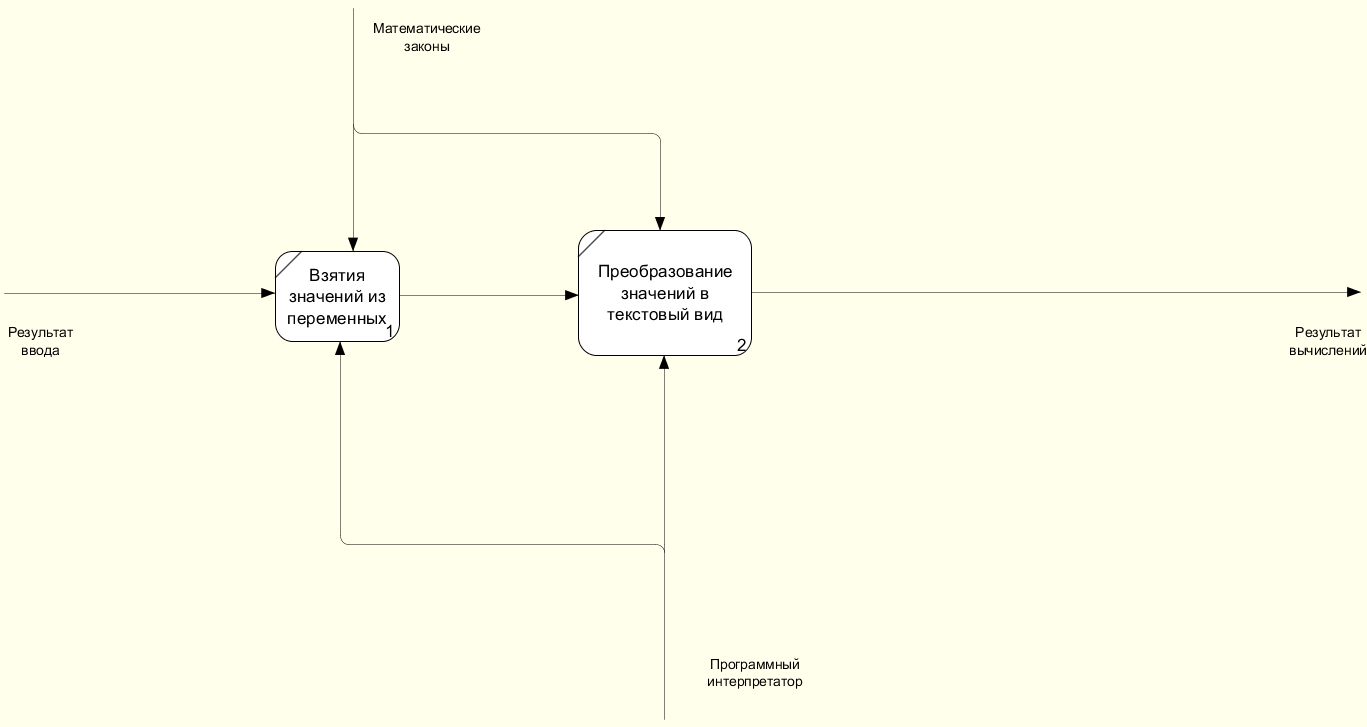


Рисунок 4. Третий уровень IDEF (2)

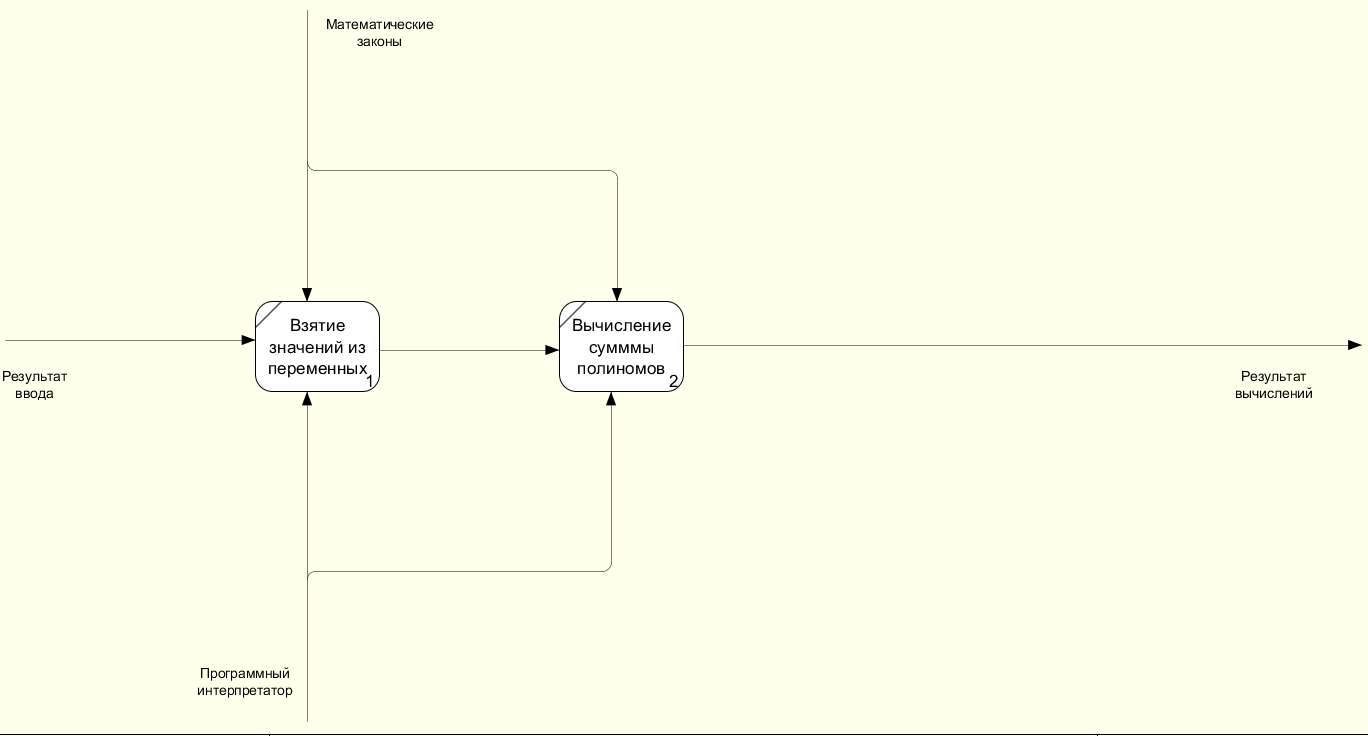


Рисунок 5. Третий уровень IDEF (3)

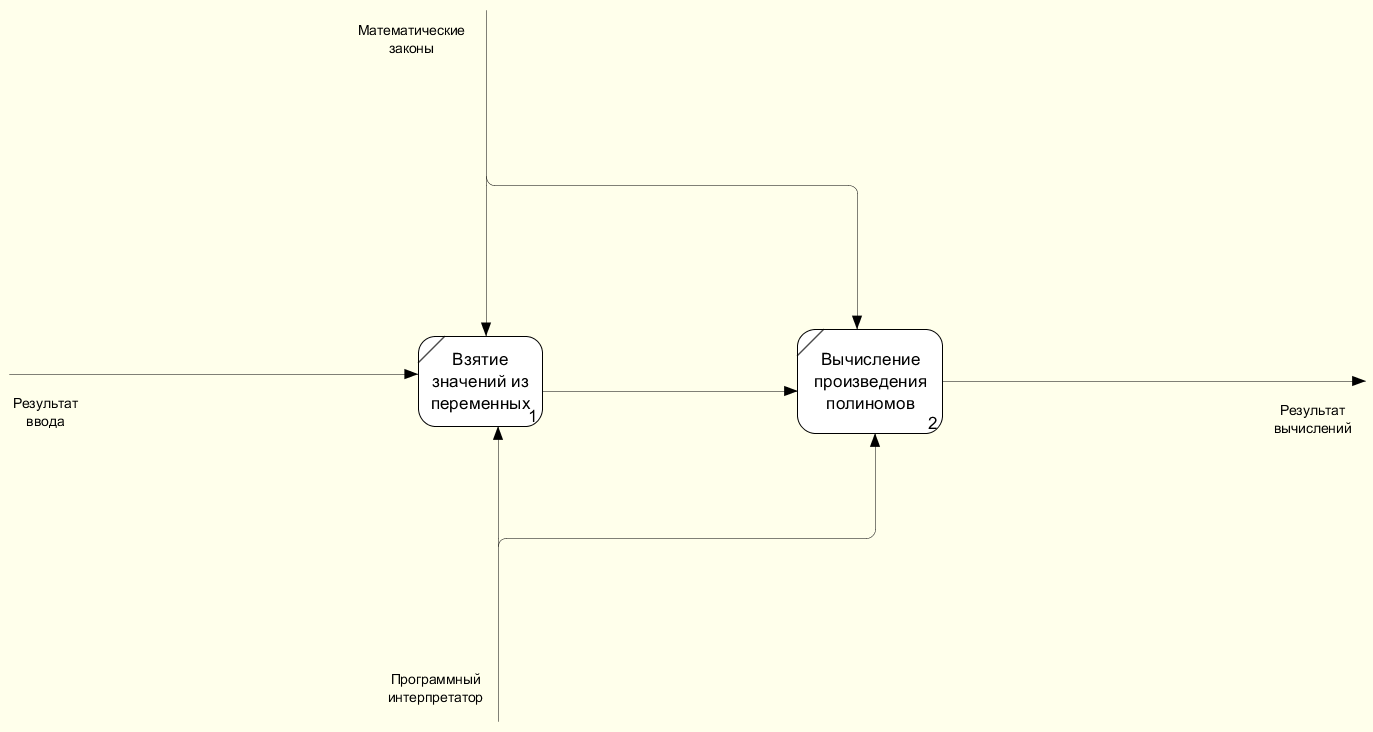


Рисунок 6. Третий уровень IDEF (4)

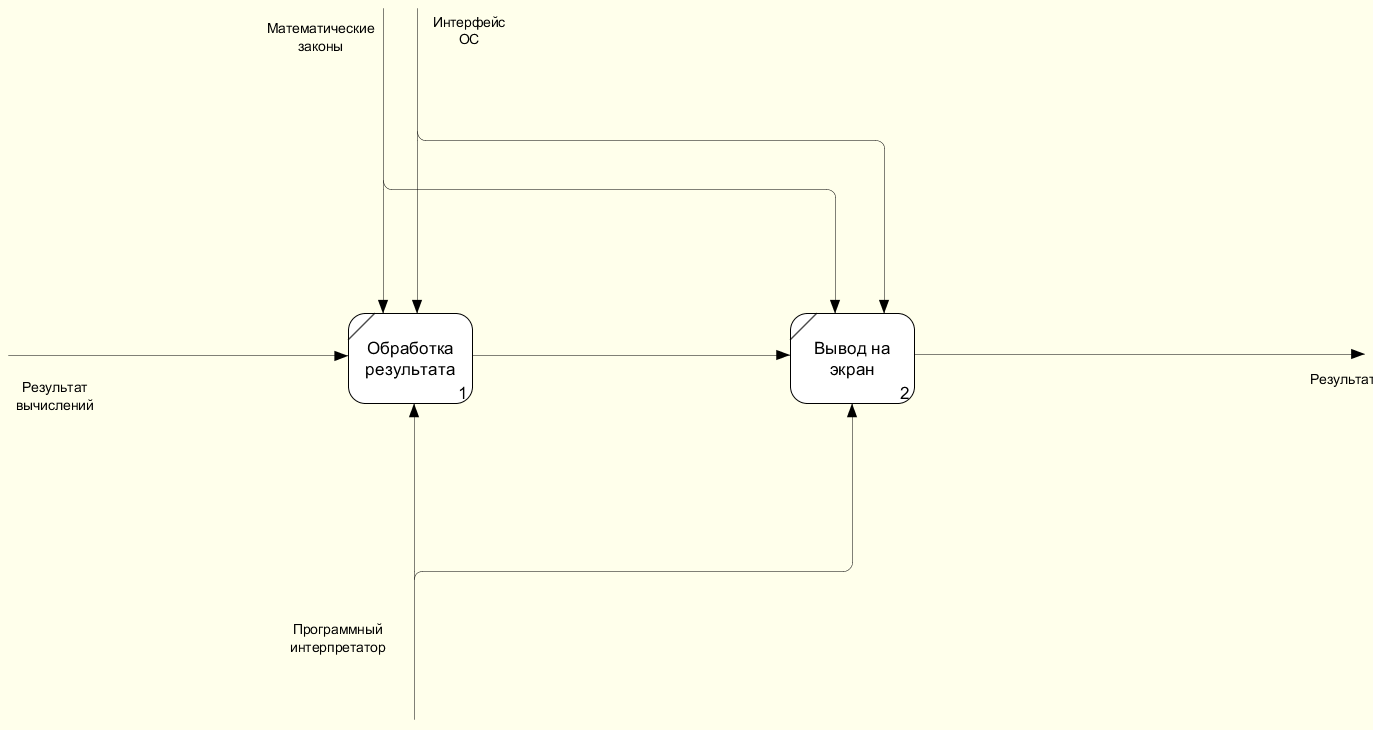


Рисунок 7. Третий уровень IDEF (5)

# Техническое задание

## Введение

### Наименование программы

Наименование программного продукта – «Polynomial»

## Краткая характеристика области

Разработать класс Polynomial для хранения коэффициентов многочлена. Определить методы нахождения суммы двух полиномов, произведения, печати.

## Основания для разработки

Основанием для разработки является автоматизация учебы студентов.

## Назначение разработки

Автоматизация в работе с полиномами.

### Функциональное назначение

Данная программа позволяет работать с полиномами. Она позволяет находить сумму и произведение двух полиномов.

### Эксплуатационное назначение

Программу «Polynomial» планируется использовать в учебных целях.

## Требования к программе или программному изделию

### Требования к функциональным характеристикам

#### Требования к составу выполняемых функций

После запуска программы пользователю предлагаются ввести степень и коэффициенты двух полиномов. После этого можно выбрать их сумму или произведение, а также напечатать их в консоли.

#### Требование к организации входных и выходных данных

Входные и выходные данные организованы в консоли.

#### Требования к временным характеристикам

Требования к временным характеристикам отсутствуют

#### Требования к надёжности

Вероятность безотказной работы системы должна составлять не менее 99.99% при условии исправности версии программы.

#### Требование к обеспечению надёжного функционирования программы

Требования отсутствуют

#### Время восстановления после отказа

Время восстановления после отказа, вызванного неисправностью технических средств, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

### Отказы из-за некорректных действий оператора

Отказы программы вследствие некорректных действий оператора невозможны.

## Условия эксплуатации

Программа открывается в отдельном окне-консоли, где происходит весь ввод-вывод.

## Климатические условия эксплуатации

Специальные условия не требуются.

## Требования к численности и квалификации

Программа не требует проведения каких-либо видов обслуживания.

### Требования к численности и квалификации персонала

Специальные условия не требуются.

## Требования к составу и параметрам технических средств

### Состав технических средств:

Состав технических средств:

* процессор с тактовой частотой, не менее 1 ГГц;
* оперативная память объемом, не менее 512 Мб;
* Свободного места на диске ~10 Мб;

### Требования к информационной и программной совместимости

Нет требований.

### Требования к маркировке и упаковке

Программа передается как скрипт на языке Python.

### Требования к транспортированию и хранению

Для транспортировки программы нужно интернет соединение.

### Специальные требования

Специальные требования отсутствуют.

### Требования к программной документации

Предварительный состав программной документации:

• техническое задание (включает описание применения);

• руководство оператора;

## Технико-экономические показатели

Программа создана с некоммерческой целью.

## Стадии и этапы разработки

Всего этапов разработки:

* Разработка требований
* Проектирование
* Реализация
* Тестирование
* Внедрение

## Порядок контроля и приемки

Приемосдаточные испытания программы должны проводиться согласно разработанной исполнителем и согласованной заказчиком «Программы и методики испытаний».

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в протоколе испытаний. На основании протокола испытаний исполнитель совместно с заказчиком подписывают акт приемки-сдачи программы в эксплуатацию.

# Руководство оператора

## Назначение программы

### Функциональное назначение программы

Программа «Polynomial» предназначена для работы с полиномами.

Данная программа реализует следующие функции:

* Сумма 2-х полиномов
* Произведение 2-х полиномов
* Вывод полиномов на экран

### Эксплуатационное назначение программы

Программу «Polynomial» планируется использовать в учебных заведениях для изучения полиномов.

## Условия выполнения программы

### Минимальный состав аппаратных средств

Минимальный состав используемых технических (аппаратных) средств:

* процессор с тактовой частотой, не менее 1 ГГц;
* оперативная память объемом, не менее 512 Мб;
* Свободного места на диске ~10 Мб;

### Требования к пользователю

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с компьютером и консолью. Персонал должен быть аттестован на II квалификационную группу по электробезопасности.

## Выполнение программы

### Загрузка и запуск мобильного приложения

Для запуска программы требуется запустить скрипт, написанный на языке Python и имеющий расширение .py. После этого откроется консоль программы для ввода-вывода данных.

### Завершение работы программы

Для завершения работы мобильного приложения достаточно закрыть консоль программы.

## Сообщения оператору

Сообщения оператору отсутствуют.

# Отладка и тестирование готового программного средства

Для тестирования функционирования программы были написаны unit-тесты, приведенные в листинге 1:

Листинг 1. Unit-тесты программы.

class TestManager:

TEST\_PASSED\_STRING = "Passed"

TEST\_FAILED\_STRING = "Failed"

def \_\_init\_\_(self):

self.tests = []

def add\_test(self, test\_name, test):

self.tests.append([test\_name, test])

def run\_all\_tests(self):

for test in self.tests:

self.check\_test(test[0], test[1])

def check\_test(self, test\_name, test):

print(test\_name + ": " + (self.TEST\_PASSED\_STRING if test() else self.TEST\_FAILED\_STRING))

def test\_sum():

test\_poly1 = Polynomial([-3, 3], [4, 2], [-5, 1])

test\_poly2 = Polynomial([3, 3], [4, 2], [-5, 1], [4, 4])

test\_poly1.add\_poly(test\_poly2)

expected\_result = [[4, 4], [8, 2], [-10, 1]]

return test\_poly1.factors == expected\_result

def test\_mul():

test\_poly1 = Polynomial([-3, 3], [4, 2], [-5, 1])

test\_poly2 = Polynomial([3, 3], [4, 2], [-5, 1], [4, 4])

test\_poly1.multiply\_poly(test\_poly2)

expected\_result = [[-12, 7], [7, 6], [-20, 5], [16, 4], [-40, 3], [25, 2]]

return test\_poly1.factors == expected\_result

def test\_derivative():

test\_poly = Polynomial([-3, 3], [4, 2], [-5, 1])

result\_poly = test\_poly.derivative()

expected\_result = [[-9, 2], [8, 1], [-5, 0]]

return result\_poly.factors == expected\_result

def test\_str():

test\_poly = Polynomial([-3, 3], [4, 2], [-5, 1])

result\_str = str(test\_poly)

expected\_result = "-3x^3 + 4x^2 - 5x"

return result\_str == expected\_result

Отладка программы производилась в консоли приложения:

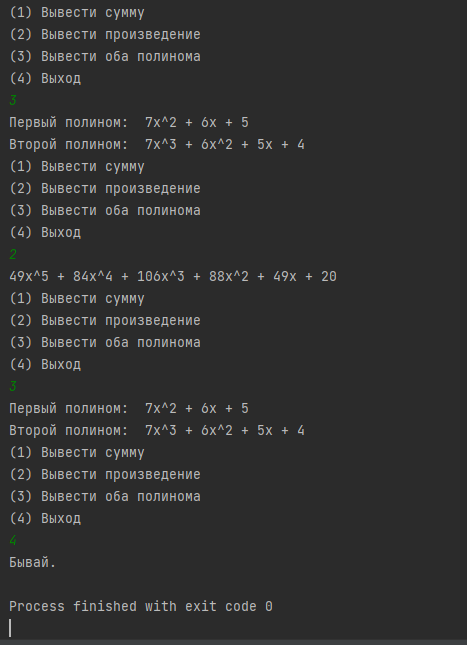


Рисунок 8. Отладка программы