МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “ Парадигмы и конструкции языков программирования”

Отчет по ДЗ

Выполнил:
Студент группы ИУ5-36Б
Германенко И.М.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

2. Описание задания

В домашнем заданим был выбран язык программирования **Python**, ранее не изучавшийся мной в полном объёме. Цель работы — глубокое освоение языка через реализацию практического проекта: **системы символьного интегрирования функций одной переменной**.

- Проект должен позволять пользователю:
- вводить математическое выражение функции $f(x)$;
- вычислять **неопределённый интеграл** (с добавлением константы C) при отсутствии пределов;
- вычислять **определённый интеграл**, если заданы оба предела;
- визуализировать функцию и её первообразную (сохранение графика в файл);
- просматривать историю вычислений;
- экспортировать результат в текстовый файл.
- Реализация выполнена в **одном файле** на языке Python с использованием библиотек **SymPy** (для символьных вычислений) и **Matplotlib** (для построения графиков). Особое внимание уделено обработке ошибок, валидации ввода и удобству использования в консольной среде.

3. Описание проекта и экранные формы

Проект представляет собой **интерактивное консольное приложение**, реализованное в одном файле DZ.py. Программа использует **меню-навигацию**, что делает её интуитивно понятной даже для пользователей без опыта работы в терминале.

Основные компоненты программы:

- **Ядро вычислений**: на основе библиотеки SymPy выполняется символьное интегрирование с поддержкой элементарных функций (sin, cos, exp, log, степеней и др.).
- **Обработка ввода**: пользователь вводит функцию в синтаксисе Python (например, x**2 + sin(x)), программа проверяет корректность выражения и выводит понятные сообщения об ошибках.
- **Графическая визуализация**: при выборе соответствующего пункта меню строится график исходной функции и её первообразной; в условиях WSL (без GUI) график автоматически сохраняется в файл integral_graph.png.
- **История и экспорт**: все вычисления сохраняются в памяти, их можно просмотреть или экспортировать в текстовый файл для отчёта.

Листинг кода:

```
# ========================
# ИМПОРТЫ
# ========================
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np
import json
import os
from datetime import datetime

# ========================
# ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ
# ========================
HISTORY = []  # история вычислений (хранится в памяти)
x = sp.symbols('x')  # основная переменная

# ========================
# ФУНКЦИИ ПРОГРАММЫ
# ========================

def clear_screen():
    """Очистка экрана (для Windows/Linux/Mac)"""
    os.system('cls' if os.name == 'nt' else 'clear')

def print_header():
    """Вывод заголовка программы"""
    print("=" * 60)
    print("          СИМВОЛЬНОЕ ИНТЕГРИРОВАНИЕ v2.0")
    print("=" * 60)

def get_function_input():
    """Запрос функции у пользователя с обработкой ошибок"""
    while True:
        try:
            expr_str = input("\nВведите функцию f(x) (например: x**2 + sin(x)):
").strip()
            if not expr_str:
                raise ValueError("Функция не может быть пустой!")
            f = sp.sympify(expr_str, evaluate=True)
            return f, expr_str
        except Exception as e:
            print(f"❌ Ошибка: {e}. Попробуйте снова.")

def get_limits_input():
    """Запрос пределов интегрирования"""
    lower = input("Введите нижний предел (или Enter для неопределённого):
").strip()
    upper = input("Введите верхний предел (или Enter для неопределённого):
").strip()
```

```
    return lower, upper

def compute_indefinite_integral(f, expr_str):
    """Вычисление неопределённого интеграла"""
    try:
        integral = sp.integrate(f, x)
        result_str = str(integral) + " + C"
        print(f"\n✅ Неопределённый интеграл:")
        print(f"∫ {expr_str} dx = {result_str}")
        return result_str
    except Exception as e:
        error_msg = f"Не удалось вычислить интеграл: {e}"
        print(f"❌ {error_msg}")
        return error_msg

def compute_definite_integral(f, expr_str, a_str, b_str):
    """Вычисление определённого интеграла"""
    try:
        a = sp.sympify(a_str)
        b = sp.sympify(b_str)
        integral = sp.integrate(f, (x, a, b))
        result_str = str(integral)
        print(f"\n✅ Определённый интеграл от {a_str} до {b_str}:")
        print(f"∫_{a_str}^{b_str} {expr_str} dx = {result_str}")
        return result_str
    except Exception as e:
        error_msg = f"Ошибка при вычислении: {e}"
        print(f"❌ {error_msg}")
        return error_msg

def plot_function_and_integral(f, integral_expr=None, a=None, b=None):
    """Построение графика функции (сохранение в файл)"""
    try:
        # Устанавливаем бэкенд Agg — для сохранения в файл без GUI
        import matplotlib
        matplotlib.use('Agg')  # Важно: должно быть до импорта pyplot!
        import matplotlib.pyplot as plt

        # Преобразуем выражения в числовые функции
        f_lambdified = sp.lambdify(x, f, "numpy")
        if integral_expr:
            F_lambdified = sp.lambdify(x, integral_expr, "numpy")

        # Создаём диапазон значений
        x_vals = np.linspace(-5, 5, 400)
        y_vals = f_lambdified(x_vals)

        plt.figure(figsize=(10, 6))
        plt.plot(x_vals, y_vals, label=f"f(x) = {f}", linewidth=2)

        if integral_expr and F_lambdified:
```

```
            F_vals = F_lambdified(x_vals)
            plt.plot(x_vals, F_vals, label="F(x) = ∫f(x)dx", linestyle="--",
linewidth=2)

        if a is not None and b is not None:
            # Заливка области под кривой
            a_val = float(a)
            b_val = float(b)
            mask = (x_vals >= a_val) & (x_vals <= b_val)
            plt.fill_between(x_vals[mask], y_vals[mask], color='lightblue',
alpha=0.5, label=f"Площадь от {a_val} до {b_val}")

        plt.axhline(0, color='black', linewidth=0.5)
        plt.axvline(0, color='black', linewidth=0.5)
        plt.grid(True, alpha=0.3)
        plt.legend()
        plt.title("График функции")
        plt.xlabel("x")
        plt.ylabel("y")

        # Сохраняем график в файл вместо show()
        filename = "integral_graph.png"
        plt.savefig(filename, dpi=150, bbox_inches='tight')
        plt.close()  # Закрываем фигуру, чтобы освободить память

        print(f"✅ График сохранён в файл: {filename}")
        print("👉 Откройте файл в проводнике Windows или через VS Code.")

    except Exception as e:
        print(f"⚠️ График не построен: {e}")

def save_to_history(expr_str, result_str, integral_type, limits=None):
    """Сохранение результата в историю (в памяти)"""
    record = {
        "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "function": expr_str,
        "type": integral_type,
        "result": result_str,
        "limits": limits or "N/A"
    }
    HISTORY.append(record)
    print("📌 Результат сохранён в историю!")

def show_history():
    """Показать историю вычислений"""
    if not HISTORY:
        print("📜 История пуста.")
        return
    print("\n📜 ИСТОРИЯ ВЫЧИСЛЕНИЙ:")
    for i, rec in enumerate(HISTORY, 1):
```

```
        print(f"{i}. [{rec['timestamp']}] {rec['type']}: {rec['function']} →
{rec['result']} (пределы: {rec['limits']})")

def export_result(result_str, expr_str, integral_type, limits=None):
    """Экспорт последнего результата в файл"""
    filename = "integral_result.txt"
    content = f"""
==============================
      ЭКСПОРТ РЕЗУЛЬТАТА
==============================
Дата: {datetime.now().strftime("%Y-%m-%d %H:%M:%S")}
Тип: {integral_type}
Функция: {expr_str}
Результат: {result_str}
Пределы: {limits or 'не указаны'}
==============================
"""
    with open(filename, "w", encoding="utf-8") as f:
        f.write(content)
    print(f"✅ Результат экспортирован в файл: {filename}")

def main_menu():
    """Главное меню программы"""
    while True:
        clear_screen()
        print_header()
        print("1. Вычислить неопределённый интеграл")
        print("2. Вычислить определённый интеграл")
        print("3. Показать историю вычислений")
        print("4. Экспортировать последний результат")
        print("5. Построить график функции")
        print("6. Выйти")

        choice = input("\nВыберите действие (1-6): ").strip()

        if choice == '1':
            f, expr_str = get_function_input()
            result = compute_indefinite_integral(f, expr_str)
            save_to_history(expr_str, result, "Неопределённый")
            input("\nНажмите Enter, чтобы продолжить...")

        elif choice == '2':
            f, expr_str = get_function_input()
            a, b = get_limits_input()
            if a and b:
                result = compute_definite_integral(f, expr_str, a, b)
                save_to_history(expr_str, result, "Определённый", f"{a}..{b}")
                input("\nНажмите Enter, чтобы продолжить...")
            else:
                print("❌ Для определённого интеграла нужны оба предела!")
                input("\nНажмите Enter, чтобы продолжить...")
```

```
        elif choice == '3':
            show_history()
            input("\nНажмите Enter, чтобы продолжить...")

        elif choice == '4':
            if HISTORY:
                last = HISTORY[-1]
                export_result(last["result"], last["function"], last["type"],
last["limits"])
            else:
                print("❌ Нет результатов для экспорта.")
            input("\nНажмите Enter, чтобы продолжить...")

        elif choice == '5':
            if HISTORY:
                last_rec = HISTORY[-1]
                f = sp.sympify(last_rec["function"])
                integral_expr = None
                if "C" in last_rec["result"]:
                    integral_expr = sp.sympify(last_rec["result"].replace(" + C",
""))
                a, b = None, None
                if last_rec["limits"] != "N/A":
                    lims = last_rec["limits"].split("..")
                    a, b = lims[0], lims[1]
                plot_function_and_integral(f, integral_expr, a, b)
            else:
                print("❌ Нет функции для построения графика.")
            input("\nНажмите Enter, чтобы продолжить...")

        elif choice == '6':
            print("\n👌 До свидания!")
            break

        else:
            print("❌ Неверный выбор. Попробуйте снова.")
            input("\nНажмите Enter, чтобы продолжить...")


# ========================
# ТОЧКА ВХОДА
# ========================
if __name__ == "__main__":
    main_menu()
```

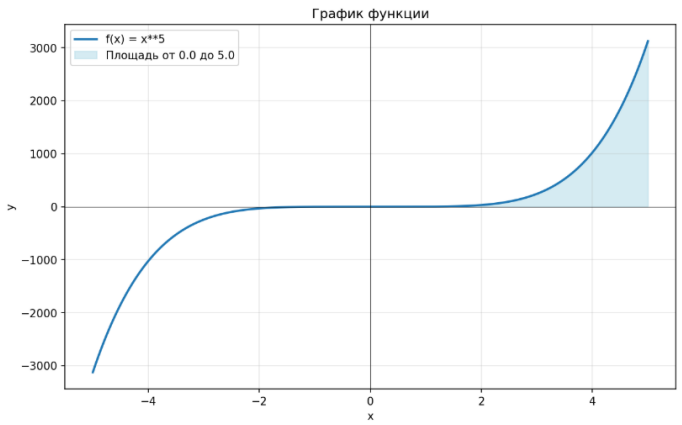
Экранные формы (примеры):

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
============================================================
          СИМВОЛЬНОЕ ИНТЕГРИРОВАНИЕ v2.0
============================================================
1. Вычислить неопределённый интеграл
2. Вычислить определённый интеграл
3. Показать историю вычислений
4. Экспортировать последний результат
5. Построить график функции
6. Выйти

Выберите действие (1-6): 1

Введите функцию f(x) (например: x**2 + sin(x)): x**2-cos(x)

✅ Неопределённый интеграл:
∫ x**2-cos(x) dx = x**3/3 - sin(x) + C
📌 Результат сохранён в историю!

Нажмите Enter, чтобы продолжить...
```

DZ.py U integral_graph.png U

DZ > integral_graph.png



ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ python3 - DZ

```
1. Вычислить неопределённый интеграл
2. Вычислить определённый интеграл
3. Показать историю вычислений
4. Экспортировать последний результат
5. Построить график функции
6. Выйти

Выберите действие (1-6): 5
✅ График сохранён в файл: integral_graph.png
👉 Откройте файл в проводнике Windows или через VS Code.

Нажмите Enter, чтобы продолжить...
```