

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
Левочкин В.В.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

2. Описание задания

В домашнем задании 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:

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

```

HISTORY=[]
x=sp.symbols('x')

defclear_screen():
    os.system('cls' if os.name=='nt' else 'clear')

defprint_header():
    print("="*60)
    print(" СИМВОЛЬНОЕ ИНТЕГРИРОВАНИЕ v2.0")
    print("="*60)

defget_function_input():
    whileTrue:
        try:
            expr_str=input("\nВведите функцию f(x) (например: x**2 + sin(x)): ")
            expr_str=expr_str.strip()
            ifnotexpr_str:
                raiseValueError("Функция не может быть пустой!")
            f=sp.sympify(expr_str, evaluate=True)
            returnf, expr_str
        exceptExceptionase:
            print(f"❌Ошибка: {e}. Попробуйте снова.")

defget_limits_input():
    lower=input("Введите нижний предел (или Enter для неопределённого): ").strip()
    upper=input("Введите верхний предел (или Enter для неопределённого): ").strip()
    returnlower, upper

defcompute_indefinite_integral(f, expr_str):
    try:
        integral=sp.integrate(f, x)
        result_str=str(integral) + " + C"
        print(f"\n✅Неопределённый интеграл:")
        print(f"∫ {expr_str}dx = {result_str}")
        returnresult_str
    exceptExceptionase:
        error_msg=f"Не удалось вычислить интеграл: {e}"
        print(f"❌{error_msg}")
        returnerror_msg

defcompute_definite_integral(f, expr_str, a_str, b_str):
    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}")
        returnresult_str
    exceptExceptionase:
        error_msg=f"Ошибка при вычислении: {e}"
        print(f"❌{error_msg}")
        returnerror_msg

defplot_function_and_integral(f, integral_expr=None, a=None, b=None):

```

```

try:
    import matplotlib
    matplotlib.use('Agg')
    import matplotlib.pyplot as plt

    f_lambdified = sp.lambdify(x, f, "numpy")

    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:
        F_lambdified = sp.lambdify(x, integral_expr, "numpy")
        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:
        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")

    filename = "integral_graph.png"
    plt.savefig(filename, dpi=150, bbox_inches='tight')
    plt.close()

    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):
    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():
    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):
    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: {limitsor'не указаны'}

```

```

=====

```

```

"""

```

```

    with open(filename, "w", encoding="utf-8") as f:

```

```

        f.write(content)

```

```

    print(f"✓Результат экспортирован в файл: {filename}")

```

```

def main_menu():

```

```

    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

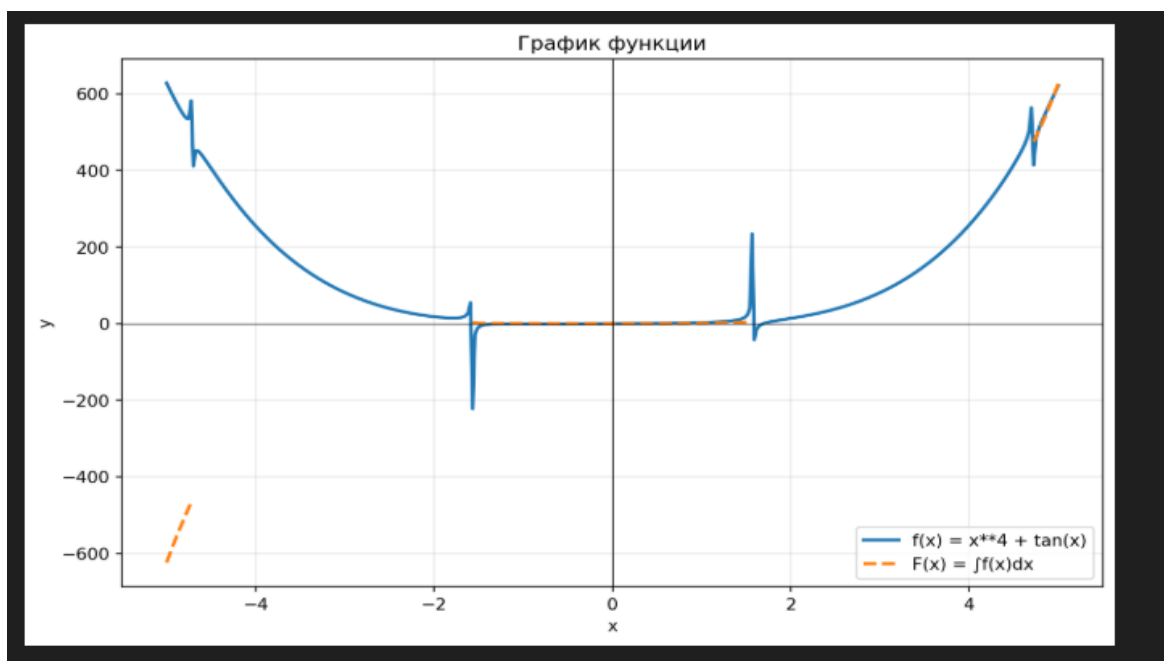
else:
    print("✗ Неверный выбор. Попробуйте снова.")
    input("\nНажмите Enter, чтобы продолжить...")

if __name__ == "__main__":
    main_menu()

```

\

Примеры выполнения кода :



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: $x^{**2} + \sin(x)$): $x^{**4} + \tan(x)$

✅ Неопределённый интеграл:

$$\int x^{**4} + \tan(x) \, dx = x^{**5}/5 - \log(\cos(x)) + C$$

📌 Результат сохранён в историю!

Нажмите Enter, чтобы продолжить...

