**x`МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**Факультет прикладной математики – процессов управления**

**отчет**

**по лабораторной работе**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Приближенное вычисление площади фигуры методом Монте-Карло»**

**Вариант 1**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22.Б15 |  | Агишев А.Б. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2024 г.**

**Содержание**

[1. Цель работы 1](#_Toc166893501)

[2. Задача 1](#_Toc166893502)

[3. Теоретическая часть 1](#_Toc166893503)

[4. Описание алгоритма 2](#_Toc166893504)

[5. Описание программы 4](#_Toc166893505)

[5.1 Описание классов 4](#_Toc166893506)

[5.2. Описание функций 4](#_Toc166893507)

[6. Рекомендации пользователю 5](#_Toc166893508)

[7. Рекомендации программисту 5](#_Toc166893509)

[8. Контрольный пример 5](#_Toc166893510)

[9. Заключение 6](#_Toc166893511)

# **Цель работы**

Разработать алгоритм и программу для вычисления площади фигуры методом Монте-Карло.

# **Задача**

Приобрести и закрепить знания в области метода Монте-Карло. Написать программу поиска решения. Протестировать программу на примерах.

# **Теоретическая часть**

*Метод Монте-Карло* — это численный метод, основанный на использовании случайных чисел для решения различных задач в физике, математике, экономике, инженерии и других областях. Он был разработан во время Второй мировой войны для применения к военным исследованиям, в частности, для моделирования поведения нейтронов.

*Основная идея метода* Монте-Карло состоит в том, чтобы использовать случайные выборки из пространства параметров задачи для приближенного решения. Этот метод имеет преимущество в том, что он может решать задачи с большим количеством степеней свободы и сложной геометрией, которые могут быть трудны для аналитического решения.

Метод Монте-Карло широко применяется для решения различных задач, включая:

* Интегрирование: вычисление определенных интегралов функций сложной формы.
* Решение уравнений: поиск корней уравнений и систем уравнений.
* Симуляция случайных процессов: моделирование поведения систем с использованием случайных чисел.
* Оптимизация: поиск оптимальных решений в многомерных пространствах параметров.

*Принцип работы метода* Монте-Карло основан на законе больших чисел, который утверждает, что с увеличением числа испытаний (или выборок) статистические оценки будут стремиться к истинным значениям параметров задачи.

Для решения задачи методом Монте-Карло обычно выполняют следующие шаги:

* Определение области параметров задачи.
* Генерация случайных выборок из этой области.
* Выполнение вычислений или проверок для каждой выборки.
* Усреднение результатов для получения приближенного решения.

*Примеры задач*, которые могут быть решены методом Монте-Карло, включают в себя:

* Вычисление значения числа π методом бросания игл на лист бумаги.
* Оценка вероятности событий в сложных системах.
* Моделирование физических процессов в материалах и структурах.
* Определение площадей сложных фигур на плоскости.

Метод Монте-Карло остается одним из наиболее мощных и универсальных численных методов, который находит применение в различных областях науки и техники благодаря своей простоте и эффективности.

# **Описание алгоритма**

1. Получение необходимых параметров через интерфейс.
2. Вычисление площади заданной фигуры или определенного интеграла методом Монте-Карло.
3. Вывод результата и относительной погрешности.

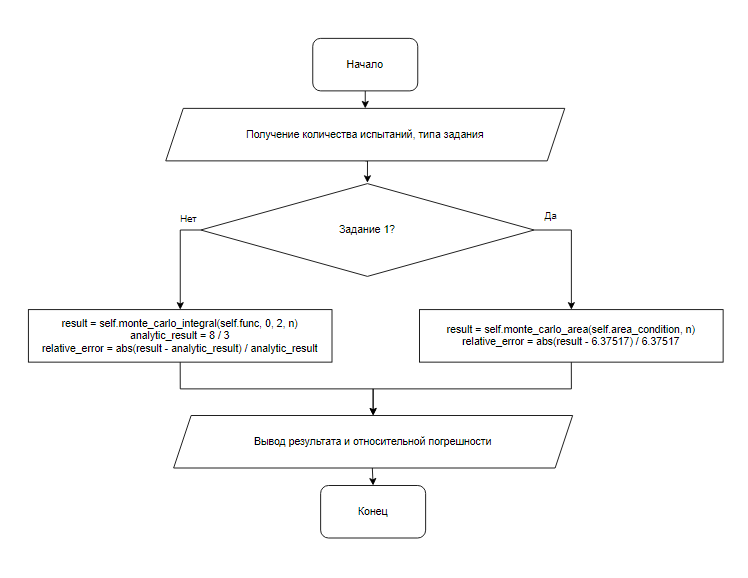


Рисунок 4.1. Блок-схема алгоритма.

# **Описание программы**

## **Описание классов**

В программе используется 1 класс. В таблице 5.1 представлено описание класса.

*Таблица 5.1. Описание класса*

|  |  |
| --- | --- |
| Имя класса | Описание |
| MonteCarloSimulator | Поиск решения задачи |

## **5.2. Описание функций**

Описание функций класса *«MonteCarloSimulator»* представлено в таблице 5.2.

*Таблица 5.2. Описание функций класса «MonteCarloSimulator»*

|  |  |  |
| --- | --- | --- |
| Входные данные | Имя | Описание |
| Экземпляр редактора графа | \_\_init\_\_ | Инициализация класса и создание интерфейса |
| Фигура, число точек | monte\_carlo\_area | Определение площади фигуры методом Монте-Карло |
| Функция, интервал, число точек | monte\_carlo\_integral | Вычисление определенного интеграла методом Монте-Карло |
| Точка | area\_condition | Проверка принадлежности точки к фигуре |
| Число | func | Возвращение значение функции |
| — | calculate | Выполнение выбранного задания |

Полный код программы представлен в Приложении А.

# **Рекомендации пользователю**

Для выполнения программы необходимо выбрать тип задания, а также количество испытаний.

Кнопка *«Вычислить»* отвечает за запуск программы.

Для завершения работы нажмите на крестик в левом верхнем углу.

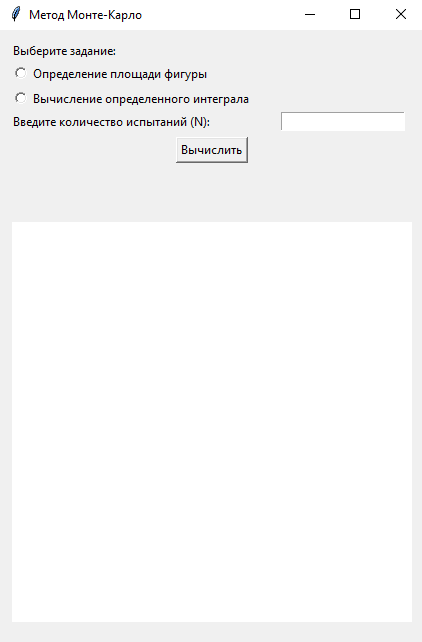
# **Рекомендации программисту**

Для запуска программы необходим Python версии не ниже 3.10.6.

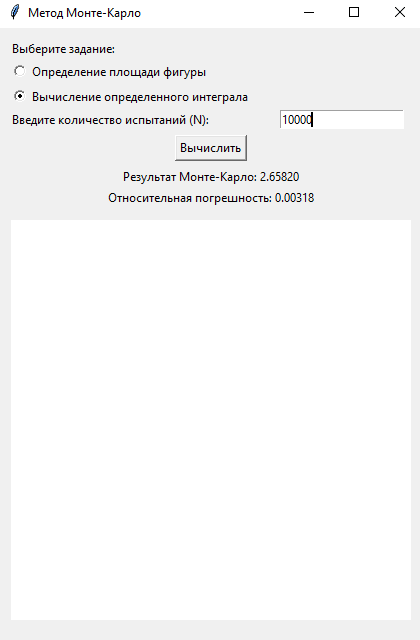
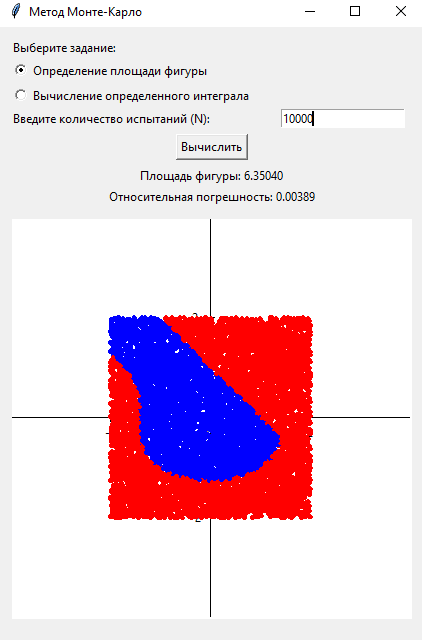
Минимальное необходимое место на диске: 0.5 МБ. Минимальное необходимое количество оперативной памяти: 100 МБ.

# **Контрольный пример**

В данном разделе представлен контрольный пример, демонстрирующий способность программы находить площадь фигуры.



*Рисунок 8.1. Интерфейс программы*



*Рисунок 8.2–8.3. Вывод решения задачи*

# **Заключение**

Эта работа демонстрирует использование метода Монте-Карло для двух задач: вычисления площади фигуры и определенного интеграла. Разработанная программа на Python с интерфейсом на Tkinter позволяет выбрать задачу и провести вычисления с разным числом испытаний. Полученные результаты могут применяться для анализа и решения различных задач в различных областях науки и техники.

1. **Приложение А**

import tkinter as tk  
from random import uniform  
  
  
class MonteCarloSimulator:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Метод Монте-Карло")  
  
 self.frame = tk.Frame(root)  
 self.frame.pack(padx=10, pady=10)  
  
 tk.Label(self.frame, text="Выберите задание:").grid(row=0, column=0, sticky="w")  
  
 self.var = tk.IntVar()  
 tk.Radiobutton(self.frame, text="Определение площади фигуры", variable=self.var, value=1).grid(row=1, column=0, sticky="w")  
 tk.Radiobutton(self.frame, text="Вычисление определенного интеграла", variable=self.var, value=2).grid(row=2, column=0, sticky="w")  
  
 tk.Label(self.frame, text="Введите количество испытаний (N):").grid(row=3, column=0, sticky="w")  
 self.entry\_n = tk.Entry(self.frame)  
 self.entry\_n.grid(row=3, column=1)  
  
 self.calculate\_button = tk.Button(self.frame, text="Вычислить", command=self.calculate)  
 self.calculate\_button.grid(row=4, columnspan=2, pady=5)  
  
 self.result\_label = tk.Label(self.frame, text="")  
 self.result\_label.grid(row=5, columnspan=2)  
  
 self.error\_label = tk.Label(self.frame, text="")  
 self.error\_label.grid(row=6, columnspan=2)  
  
 def monte\_carlo\_area(self, area, n):  
 inside = 0  
 for \_ in range(n):  
 x = uniform(-2, 2)  
 y = uniform(-2, 2)  
 if area(x, y):  
 inside += 1  
 return (inside / n) \* 16 # S = (K / N) \* S0  
  
 def monte\_carlo\_integral(self, func, a, b, n):  
 integral\_sum = 0  
 for \_ in range(n):  
 x = uniform(a, b)  
 integral\_sum += func(x)  
 return (integral\_sum / n) \* (b - a)  
  
 def area\_condition(self, x, y):  
 return x \*\* 2 - y \*\* 3 < 2 and x + y < 1  
  
 def func(self, x):  
 return x \*\* 2  
  
 def calculate(self):  
 try:  
 n = int(self.entry\_n.get())  
 if self.var.get() == 1:  
 result = self.monte\_carlo\_area(self.area\_condition, n)  
 relative\_error = abs(result - 6.37517) / 6.37517  
 self.result\_label.config(text=f"Площадь фигуры: {result:.5f}")  
 self.error\_label.config(text=f"Относительная погрешность: {relative\_error:.5f}")  
 elif self.var.get() == 2:  
 result = self.monte\_carlo\_integral(self.func, 0, 2, n)  
 analytic\_result = 8 / 3 # Аналитическое решение для интеграла x^2 от 0 до 2  
 relative\_error = abs(result - analytic\_result) / analytic\_result  
 self.result\_label.config(text=f"Результат Монте-Карло: {result:.5f}")  
 self.error\_label.config(text=f"Относительная погрешность: {relative\_error:.5f}")  
 except ValueError:  
 self.result\_label.config(text="Ошибка: введите целое число для N.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = MonteCarloSimulator(root)  
 root.mainloop()

*Листинг программы.*