|  |  |
| --- | --- |
| ***Вариант задания №7*** | |
| *Код программы:* | *Код оптимизированной программы:* |
| ***# Программа 1: Сравнение методов оптимизации (элементарные конструкции) # Реализация методов Фибоначчи, дихотомии и золотого сечения  import math  print("Программа для сравнения методов оптимизации")  # Функция, которую мы минимизируем print("Введите данные для функции f(x) = a\*x^2 + b\*x + c") a = float(input("Введите коэффициент a: ")) b = float(input("Введите коэффициент b: ")) c = float(input("Введите коэффициент c: "))  # Входные параметры для всех методов print("\nВведите границы интервала поиска и точность") left = float(input("Левая граница: ")) right = float(input("Правая граница: ")) eps = float(input("Точность (epsilon): "))  # Счетчики вычислений функции для каждого метода count\_dichotomy = 0 count\_golden\_section = 0 count\_fibonacci = 0   # Вычисление значения функции def f(x):  return a \* x \*\* 2 + b \* x + c   print("\n1. Метод дихотомии") # Метод дихотомии a\_dichot = left b\_dichot = right delta = eps / 2 # Параметр для дихотомии  while b\_dichot - a\_dichot > eps:  x1 = (a\_dichot + b\_dichot - delta) / 2  x2 = (a\_dichot + b\_dichot + delta) / 2   f1 = f(x1)  f2 = f(x2)  count\_dichotomy += 2 # Два вычисления функции   if f1 <= f2:  b\_dichot = x2  else:  a\_dichot = x1  x\_min\_dichotomy = (a\_dichot + b\_dichot) / 2 f\_min\_dichotomy = f(x\_min\_dichotomy) count\_dichotomy += 1 # Дополнительное вычисление для f\_min  print(f"Результат: x\_минимум = {x\_min\_dichotomy:.6f}, f(x\_минимум) = {f\_min\_dichotomy:.6f}") print(f"Количество вычислений функции: {count\_dichotomy}")  print("\n2. Метод золотого сечения") # Метод золотого сечения a\_golden = left b\_golden = right golden\_ratio = (math.sqrt(5) - 1) / 2 # ≈ 0.618  x1 = a\_golden + (1 - golden\_ratio) \* (b\_golden - a\_golden) x2 = a\_golden + golden\_ratio \* (b\_golden - a\_golden) f1 = f(x1) f2 = f(x2) count\_golden\_section += 2 # Два начальных вычисления функции  while b\_golden - a\_golden > eps:  if f1 <= f2:  b\_golden = x2  x2 = x1  f2 = f1  x1 = a\_golden + (1 - golden\_ratio) \* (b\_golden - a\_golden)  f1 = f(x1)  count\_golden\_section += 1  else:  a\_golden = x1  x1 = x2  f1 = f2  x2 = a\_golden + golden\_ratio \* (b\_golden - a\_golden)  f2 = f(x2)  count\_golden\_section += 1  x\_min\_golden = (a\_golden + b\_golden) / 2 f\_min\_golden = f(x\_min\_golden) count\_golden\_section += 1 # Дополнительное вычисление для f\_min  print(f"Результат: x\_минимум = {x\_min\_golden:.6f}, f(x\_минимум) = {f\_min\_golden:.6f}") print(f"Количество вычислений функции: {count\_golden\_section}")  print("\n3. Метод Фибоначчи") # Метод Фибоначчи a\_fib = left b\_fib = right  # Вычисление чисел Фибоначчи fib = [1, 1] # Первые два числа Фибоначчи n = 2 # Уже вычислены 2 числа while fib[n - 1] <= (b\_fib - a\_fib) / eps:  fib.append(fib[n - 1] + fib[n - 2])  n += 1  n = n - 1 # Индекс последнего вычисленного числа print(f"Используемое число Фибоначчи: {fib[n]}")  # Инициализация точек x1 = a\_fib + (fib[n - 2] / fib[n]) \* (b\_fib - a\_fib) x2 = a\_fib + (fib[n - 1] / fib[n]) \* (b\_fib - a\_fib) f1 = f(x1) f2 = f(x2) count\_fibonacci += 2 # Два начальных вычисления функции  # Итерации метода Фибоначчи k = 1 while k < n - 2:  if f1 > f2:  a\_fib = x1  x1 = x2  f1 = f2  x2 = a\_fib + (fib[n - k - 1] / fib[n - k]) \* (b\_fib - a\_fib)  f2 = f(x2)  count\_fibonacci += 1  else:  b\_fib = x2  x2 = x1  f2 = f1  x1 = a\_fib + (fib[n - k - 2] / fib[n - k]) \* (b\_fib - a\_fib)  f1 = f(x1)  count\_fibonacci += 1  k += 1  # Последняя итерация if f1 > f2:  a\_fib = x1 else:  b\_fib = x2  x\_min\_fibonacci = (a\_fib + b\_fib) / 2 f\_min\_fibonacci = f(x\_min\_fibonacci) count\_fibonacci += 1 # Дополнительное вычисление для f\_min  print(f"Результат: x\_минимум = {x\_min\_fibonacci:.6f}, f(x\_минимум) = {f\_min\_fibonacci:.6f}") print(f"Количество вычислений функции: {count\_fibonacci}")  print("\nИтоговое сравнение методов:") print(f"1. Дихотомия: x = {x\_min\_dichotomy:.6f}, f(x) = {f\_min\_dichotomy:.6f}, вычислений: {count\_dichotomy}") print(f"2. Золотое сечение: x = {x\_min\_golden:.6f}, f(x) = {f\_min\_golden:.6f}, вычислений: {count\_golden\_section}") print(f"3. Фибоначчи: x = {x\_min\_fibonacci:.6f}, f(x) = {f\_min\_fibonacci:.6f}, вычислений: {count\_fibonacci}")*** | ***# Программа 2: Сравнение методов оптимизации (оптимизированная версия с функциями) # Реализация методов Фибоначчи, дихотомии и золотого сечения  import math   def input\_data():  """Получение входных данных от пользователя"""  print("Программа для сравнения методов оптимизации")  print("Введите данные для функции f(x) = a\*x^2 + b\*x + c")   a = float(input("Введите коэффициент a: "))  b = float(input("Введите коэффициент b: "))  c = float(input("Введите коэффициент c: "))   print("\nВведите границы интервала поиска и точность")  left = float(input("Левая граница: "))  right = float(input("Правая граница: "))  eps = float(input("Точность (epsilon): "))   return a, b, c, left, right, eps   def target\_function(x, a, b, c):  """Целевая функция f(x) = a\*x^2 + b\*x + c"""  return a \* x \*\* 2 + b \* x + c   def dichotomy\_method(f, left, right, eps, params):  """  Метод дихотомии для поиска минимума функции   Args:  f: целевая функция  left, right: границы интервала  eps: точность  params: параметры для целевой функции   Returns:  tuple: (x\_min, f\_min, количество вычислений)  """  a, b = left, right  delta = eps / 2 # Параметр для дихотомии  count = 0   while b - a > eps:  x1 = (a + b - delta) / 2  x2 = (a + b + delta) / 2   f1 = f(x1, \*params)  f2 = f(x2, \*params)  count += 2 # Два вычисления функции   if f1 <= f2:  b = x2  else:  a = x1   x\_min = (a + b) / 2  f\_min = f(x\_min, \*params)  count += 1 # Дополнительное вычисление для f\_min   return x\_min, f\_min, count   def golden\_section\_method(f, left, right, eps, params):  """  Метод золотого сечения для поиска минимума функции   Args:  f: целевая функция  left, right: границы интервала  eps: точность  params: параметры для целевой функции   Returns:  tuple: (x\_min, f\_min, количество вычислений)  """  a, b = left, right  golden\_ratio = (math.sqrt(5) - 1) / 2 # ≈ 0.618  count = 0   x1 = a + (1 - golden\_ratio) \* (b - a)  x2 = a + golden\_ratio \* (b - a)  f1 = f(x1, \*params)  f2 = f(x2, \*params)  count += 2 # Два начальных вычисления функции   while b - a > eps:  if f1 <= f2:  b = x2  x2 = x1  f2 = f1  x1 = a + (1 - golden\_ratio) \* (b - a)  f1 = f(x1, \*params)  count += 1  else:  a = x1  x1 = x2  f1 = f2  x2 = a + golden\_ratio \* (b - a)  f2 = f(x2, \*params)  count += 1   x\_min = (a + b) / 2  f\_min = f(x\_min, \*params)  count += 1 # Дополнительное вычисление для f\_min   return x\_min, f\_min, count   def generate\_fibonacci(n):  """Генерирует последовательность чисел Фибоначчи до n-го числа"""  fib = [1, 1]  for i in range(2, n):  fib.append(fib[i - 1] + fib[i - 2])  return fib   def find\_fibonacci\_number(ratio, eps):  """Находит подходящее число Фибоначчи для заданной точности"""  fib = [1, 1]  n = 2  while fib[n - 1] <= ratio / eps:  fib.append(fib[n - 1] + fib[n - 2])  n += 1  return fib, n - 1   def fibonacci\_method(f, left, right, eps, params):  """  Метод Фибоначчи для поиска минимума функции   Args:  f: целевая функция  left, right: границы интервала  eps: точность  params: параметры для целевой функции   Returns:  tuple: (x\_min, f\_min, количество вычислений, использованное число Фибоначчи)  """  a, b = left, right  count = 0   # Вычисление чисел Фибоначчи  fib, n = find\_fibonacci\_number(b - a, eps)   # Инициализация точек  x1 = a + (fib[n - 2] / fib[n]) \* (b - a)  x2 = a + (fib[n - 1] / fib[n]) \* (b - a)  f1 = f(x1, \*params)  f2 = f(x2, \*params)  count += 2 # Два начальных вычисления функции   # Итерации метода Фибоначчи  k = 1  while k < n - 2:  if f1 > f2:  a = x1  x1 = x2  f1 = f2  x2 = a + (fib[n - k - 1] / fib[n - k]) \* (b - a)  f2 = f(x2, \*params)  count += 1  else:  b = x2  x2 = x1  f2 = f1  x1 = a + (fib[n - k - 2] / fib[n - k]) \* (b - a)  f1 = f(x1, \*params)  count += 1  k += 1   # Последняя итерация  if f1 > f2:  a = x1  else:  b = x2   x\_min = (a + b) / 2  f\_min = f(x\_min, \*params)  count += 1 # Дополнительное вычисление для f\_min   return x\_min, f\_min, count, fib[n]   def print\_results(method\_name, x\_min, f\_min, count, additional\_info=None):  """Вывод результатов метода"""  print(f"\n{method\_name}")  if additional\_info:  print(additional\_info)  print(f"Результат: x\_минимум = {x\_min:.6f}, f(x\_минимум) = {f\_min:.6f}")  print(f"Количество вычислений функции: {count}")  return f"{x\_min:.6f}", f"{f\_min:.6f}", count   def main():  """Основная функция программы"""  # Получение входных данных  a, b, c, left, right, eps = input\_data()  params = (a, b, c) # Параметры для целевой функции   # Применение метода дихотомии  x\_min\_dichotomy, f\_min\_dichotomy, count\_dichotomy = dichotomy\_method(  target\_function, left, right, eps, params  )  dichotomy\_results = print\_results("1. Метод дихотомии", x\_min\_dichotomy, f\_min\_dichotomy, count\_dichotomy)   # Применение метода золотого сечения  x\_min\_golden, f\_min\_golden, count\_golden = golden\_section\_method(  target\_function, left, right, eps, params  )  golden\_results = print\_results("2. Метод золотого сечения", x\_min\_golden, f\_min\_golden, count\_golden)   # Применение метода Фибоначчи  x\_min\_fibonacci, f\_min\_fibonacci, count\_fibonacci, fib\_number = fibonacci\_method(  target\_function, left, right, eps, params  )  additional\_info = f"Используемое число Фибоначчи: {fib\_number}"  fibonacci\_results = print\_results("3. Метод Фибоначчи", x\_min\_fibonacci, f\_min\_fibonacci, count\_fibonacci,  additional\_info)   # Итоговое сравнение  print("\nИтоговое сравнение методов:")  print(f"1. Дихотомия: x = {x\_min\_dichotomy:.6f}, f(x) = {f\_min\_dichotomy:.6f}, вычислений: {count\_dichotomy}")  print(f"2. Золотое сечение: x = {x\_min\_golden:.6f}, f(x) = {f\_min\_golden:.6f}, вычислений: {count\_golden}")  print(f"3. Фибоначчи: x = {x\_min\_fibonacci:.6f}, f(x) = {f\_min\_fibonacci:.6f}, вычислений: {count\_fibonacci}")   if \_\_name\_\_ == "\_\_main\_\_":  main()*** |
| *Расчет метрики:* | *Расчет метрики:* |
|  | |