Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 3 «Итерационные алгоритмы»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Комбаев И. Д.

студент группы 609-01

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2020 г.

Цель работы:

* закрепление знаний конструкции повторения структурного программирования;
* получение практических навыков разработки циклических алгоритмов;
* построение инициализаторов и итераторов, использования операторов цикла

Задание: требуется решить задачу в соответствии с индивидуальными заданием, выданным преподавателем, с использованием в реализации алгоритма операторов цикла while в первом случае и for – во втором

Вариант 20. Заданы числа A, ω, α ∈ [0, π] и количество итераций. Найти площадь фигуры, ограниченной осями координат и первым полупериодом функции y = A⋅sin(ω⋅t + α), методом Монте-Карло.

**Вариант 20**

**Формальное описание задачи.**

Метод Монте-Карло — это способ вычисления площади фигуры, которая находится в границах какой-либо фигуры, площадь которой известна. Пусть A – фигура, которая нам известна, а Б – фигура, площадь которой мы ищем. Б помещается «внутрь» фигура А, то есть никакая часть фигуры Б не выходит за границы фигуры А. Под фигурой А мы обычно берем квадрат или прямоугольник. Чтобы вычислить площадь фигуры Б мы случайным образом ставим точки в область фигуры А. После того, как некоторое кол-во точек будет поставлено –, мы считаем сколько точек попало в фигуру, а сколько не попало. Далее мы можем найти соотношение непопавших точек к попавшим. Так как точки были поставлены случайно, можно считать, что это отношение приблизительно равно отношение площади фигуры А к Б. Из этого равенства мы можем найти приблизительную площадь фигура Б. Чем больше точек было поставлено – тем выше точность этого метода.

Функции y = A⋅sin(ω⋅t + α) – синусоида, площадь которой нужно найти. У этой синусоиды мы ищем часть площади, которая ограничена осями плоскости и первым полупериодом. Стоит учесть, что при А < 0 – следует считать площадь части синусоиды, которая находится в IV четверти. Это следует делать потому, что значение А < 0 инвертирует синусоиду относительно оси Х.

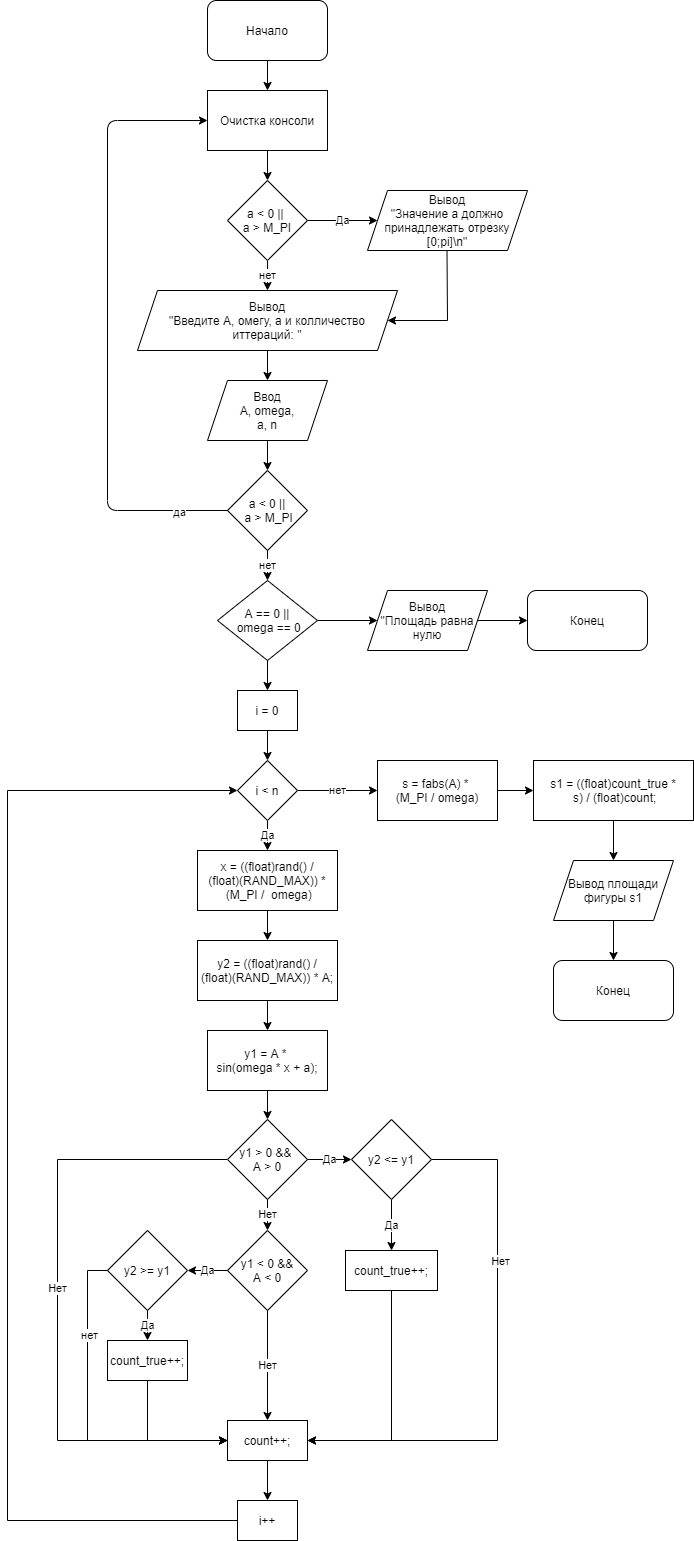
В качестве фигуры А мы будем брать прямоугольник, высота которого равна модулю значения А (т.к. синусоида по модулю не может быть больше одного, А увеличивает эту границу в А раз). А в качество длины мы будем брать значение  π / ω (Так как введенное значение омеги «расширяет» или «сжимает» синусоиду, то и точка полупериода увеличивается или уменьшается в расстоянии от начала коорд. плоскости)

Также стоит заметить, что При А = 0 или омега = 0 уравнение не имеет смысла, так как функция превращается в функцию вида y = 0 или y = Asin(a) соответственно. Как мы видим эти функции не зависят от каких-либо аргументов, значит это просто прямые, а значит площадь равна нулю.

Мы можем считать случайную точку R(x, y) расположенной внутри синусоиды, если y <= игрека, который был найден путем подставления x в функцию синусоиды, и А > 0. При A < 0: y должен быть больше и равен игреку, который мы находим путем подставления случайного икс в функцию синусоиды   
  
 **Алгоритм решения:**

1. Ввести исходные данные – А, омегу, а и n
2. Проверить введенные значения исходя из условий задачи
3. Если А = 0 или омега = 0 – площадь равна нулю, вывести площадь
4. В ином случае поставить n случайных точек
5. Посчитать сколько точек попало в фигуру, а сколько вне фигуры
6. Вычислить площадь фигуры
7. Вывести результат

**Блок схема**



**Листинг программы.**

1. #include <stdlib.h>
2. #include <math.h>
3. #include <stdio.h>
5. **int** main() {
7. **int** n, i, count\_true = 0, count = 0;
8. **double** A, omega, a = 1, x, y1, y2, s1, s;
9. system("chcp 1251 > nul");
10. srand(time(0)); // установка для генератора псевдо-случайных чисел
12. **do** {
13. system("cls");
14. **if**(a < 0 || a > M\_PI) {
15. printf("Значение а должно принадлежать отрезку [0;pi]\n");
16. }
17. printf("Введите А, омегу, а и колличество иттераций: ");
18. scanf("%lg%lg%lg%d", &A, &omega, &a, &n);
19. } **while** (a < 0 || a > M\_PI);
21. // При А = 0 или омега = 0 уравнение не имеет смысла, так как функция превращается в функцию вида y = 0 или y = Asin(a) соответственно
22. **if** (A == 0 || omega == 0) {
23. printf("Площадь равна нулю\n");
24. system("pause");
25. **return** 0;
26. }
28. **for**(i = 0; i < n; i++) {
29. x = ((**float**)rand()/(**float**)(RAND\_MAX)) \* (M\_PI /  omega); // Рандомный икс, границы которого обусловлены омегой
30. y2 = ((**float**)rand()/(**float**)(RAND\_MAX)) \* A; // Рандомный игрек, границы которого обусловлено значением А
31. y1 = A \* sin(omega \* x + a); // Коордианат y функции при значении x
32. **if** (y1 > 0 && A > 0) {
33. **if** (y2 <= y1) {
34. count\_true++;
35. }
36. } **else** **if** (y1 < 0 && A < 0) {
37. **if** (y2 >= y1) {
38. count\_true++;
39. }
40. }
41. count++;
42. }
44. s = fabs(A) \* (M\_PI / omega); // Вычисление площади
45. s1 = ((**float**)count\_true \* s) / (**float**)count;
46. printf("Площадь фигуры равна: %lg.\n", s1);
48. system("pause");
49. **return** 0;
50. }

**Пояснения к программе.**

srand(time(0)) – генератор псевдослучайных чисел будет работать в зависимости от времени устройства. Это гарантирует постоянную сменяемость «случайных» значений, которые будет генерировать функция rand()

В строке с 12 по 19 идет проверка на адекватность введенного значения альфа. По условии альфа не может быть меньше нуля или больше числа пи. При несоблюдение этого условия – программа стирает консоль, выводит предупреждение и просит вновь ввести значения.

При А = 0 или омега = 0 программа сразу завершает свою работу и выводит площадь равную 0. Это оптимизирует решение и позволяет не просчитывать ничего зря.

Генерирование случайного вещественного числа в диапазоне [0, х] выполняется с помощью команды ((float)rand()/(float)(RAND\_MAX)) \* X;. RAND\_MAX – это максимальное число, которое может дать функция rand(), самое минимальное число, которое может дать функция rand() – 0. С помощью деления и приведением к вещественному типу мы получаем случайно вещественное число в диапазоне [0, 1], умножив деление на число Х, мы увеличиваем максимальное допустимое число в Х раз, получив диапазон [0, X]

Для организации дружественного интерфейса пользователя перед вводом данных выводится сообщение для пользователя. Ввод и вывод организован при помощи стандартных функций scanf() и printf(), для этого подключен заголовочный файл stdio.h.

Заголовочный файл stdlib.h необходим для использования функции system(), чтобы приостановить выполнение программы перед завершением (выполняется команда «pause» операционной системы). А также для очистки консоли (выполняется команда «cls» операционной системы)

Заголовочный файл math.h необходим для использования функции sin(), fabs()

**Выводы:**

Для решения этой задачи я использовал цикл do {} while(); с постусловием. А также цикл for.

Выбор цикла for вместо цикла while было обусловлено тем, что мне нужен был цикл на определенное количество иттераций. При выборе цикла while пришлось бы увеличивать инициализировать и увеличивать переменную вне цикла, что не очень хорошо для восприятия.

Код для сравнения:

1. i = 0;
2. **while**(i < n) {
3. x = ((**float**)rand()/(**float**)(RAND\_MAX)) \* (M\_PI /  omega);
4. y2 = ((**float**)rand()/(**float**)(RAND\_MAX)) \* A;
5. y1 = A \* sin(omega \* x + a);
6. **if** (y1 > 0 && A > 0) {
7. **if** (y2 <= y1) {
8. count\_true++;
9. }
10. } **else** **if** (y1 < 0 && A < 0) {
11. **if** (y2 >= y1) {
12. count\_true++;
13. }
14. }
15. count++;
16. i++
17. }