МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра информатики и вычислительной техники (ИВТ)

Лабораторная работа №05

**Программирование с использованием одномерных массивов**

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.М. Губченко

Проверил: к.т.н,, доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Г. Скурыдин

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2019

1. **Формулировка задачи**

Выполнить арифметический сдвиг элементов 10-элементного массива на три позиции влево.

1. **Постановка задачи**

Арифметический сдвиг элементов массива на три позиции влево- это процесс, при котором первый элемент массива остаётся неизменным, а все последующие значения смещаются на 3 позиции влево. Последние три элемента массива заполняются нулями.

Инициализируется 10 элементный массив и программа выполняет арифметический сдвиг элементов на три позиции влево. На выход поступают 10 действительных чисел.

1. **Математическая модель**

Для построения математической модели рассмотрим пример. Инициализируется 10-элементный массив, с значениями вида a[i] a[1]=1, a[2]=2, a[3]=3, a[4]=4, a[5]=5, a[6]=6, a[7]=7, a[8]=8, a[9]=9, a[10]=10. После этого выполняется проверка условия, если i меньше 10 и не равно единице, то, если i меньше 8, то этому элементу массива присваивается значение i+3-го элемента массива, иначе присваивается значение равное нулю. В итоге элементу a[2] присваивается значение a[5], a[3] присваивается значение a[6] и так далее. Элементы a[8], a[9] и a[10] становятся равны нулю. Программа выводит конечный массив 1 5 6 7 8 9 10 0 0 0.

1. **Описание алгоритма**

**Начало алгоритма**

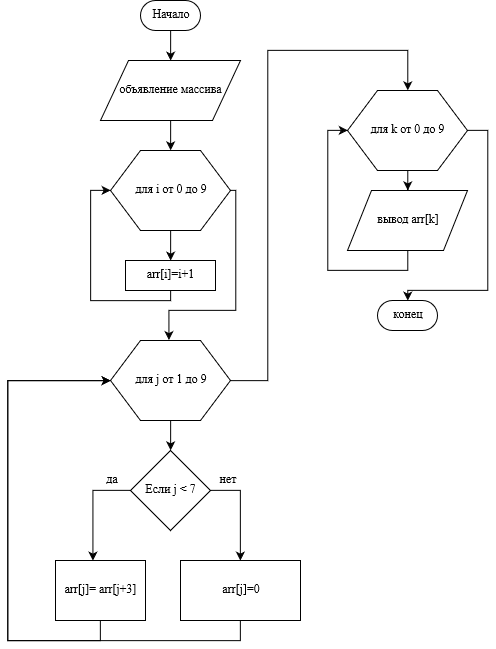
* 1. Инициализируется массив размерностью 10
  2. Присвоение переменной i значения 0
  3. Если i > 10, переход к пункту 4.6
  4. Присвоение i-ому элементу массива значения i+1
  5. Присвоение переменной i значения i+1, переход к пункту 4.3
  6. Присвоение переменной j значения 1
  7. Если j>9 переход к пункту 4.11
  8. Если j >=7 переход к пункту 4.10
  9. Присвоение j-ому элементу массива значения j+3-го элемента**,** переход к пункту 4.8
  10. Присвоение j-ому элементу значения 0, переход к пункту 4.7
  11. Присвоение переменной k значения 0
  12. Если k>=9, конец выполнения программы
  13. Вывод значения k-го элемента массива
  14. Присвоение переменной k значения k+1, переход к пункту 4.12

**Конец алгоритма**

1. **Проект программы с выполнением арифметического сдвига 10-элементного массива**

Проект программы с выполнением арифметического сдвига 10-элементного массива. В данном случае программа должна состоять из двенадцати частей: цикл с параметром (содержит блок определения переменной i, 1 блок арифметических действий и 1 блок вывода), цикл с параметром(содержит блок определения переменной j, 1 блок проверки условия, 2 блока арифметических действий), цикл с параметром(содержит блок определения переменной k, блок вывода).

1. **Опорный граф (блок-схема) алгоритма**



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int arr[9];

cout << "Input array" << '\n';

for (int i = 0; i < 10; i++) {

arr[i] = i+1;

cout << arr[i] << " ";

}

for (int j = 1; j<=9; j++)

{

if (j < 7) {

arr[j] = arr[j+3];

} else {

arr[j] = 0;

}

}

cout << " " << '\n';

for (int k = 0; k < 9; k++)

{

cout << arr[k] << " ";

}

cout << "0\n";

return 0;

}

**на языке Pascal**

program lab\_05;

var

i, j, k: integer;

arr: array [1..10] of integer;

begin

WriteLn('Input array');

for i := 1 to 10 do

begin

arr[i] := i;

Write(arr[i], ' ');

end;

for j := 2 to 10 do

begin

if j < 8 THEN

begin

arr[j] := arr[j + 3];

end ELSE

begin

if j >= 8 THEN begin

arr[j] := 0;

end;

end;

end;

WriteLn();

for k := 1 to 9 do

begin

Write(arr[k], ' ');

end;

WriteLn(0);

end.

1. **Проверка работоспособности (тестирование) программы**

Для проверки работоспособности программы необходимо просто посмотреть на то, что выведет программа в конце работы.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.

1. **Формулировка задачи**

По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла . Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций.

1. **Постановка задачи**

На вход поступает положительное вещественное число eps, определяющее точность вычисления, которое не превышает 0.001. После этого программа вычисляет интеграл по формуле Симпсона и методом прямоугольников. На выход поступают 2 действительных числа определяющих значения вычисленного интеграла разными способами и ещё два целых, положительных числа определяющих количество итераций в каждом методе вычислений.

1. **Математическая модель**

Для построения математической модели нам нужно рассмотреть алгоритм вычисления значения интеграла двумя способами. Программа получает на вход число, определяющее точность вычисления, eps=0.001. После этого программа вычисляет значение интеграла по формуле Симпсона в первом приближении, которое равно 0.0721471 и значение интеграла во втором приближении равное 0.236739. После этого мы находим модуль разности этих чисел и получаем число 0.164592. Т.к. это число больше степени точности, то программа находит значения интеграла в следующем приближении и находит модуль разности полученного значения с предыдущим. И так продолжается до тех пор, пока модуль не станет меньше степени точности. В итоге программа выведет значение интеграла, равное 0.283561. После этого программа вычисляет значение интеграла методом прямоугольников. Сначала она находит значение интеграла с использованием 2 прямоугольников равное -0.0659744. Потом количество прямоугольников увеличивается в два раза и находится значение интеграла равное 0.148694. Находится модуль разности между вторым значением и предыдущим. Численное значение которого равно 0.214669. Пока разность не будет меньше точности вычисления, количество прямоугольников будет увеличиваться в два раза. В итоге программа выведет число 0.283678 равное значению интеграла.

1. **Описание алгоритма**

**Начало алгоритма**

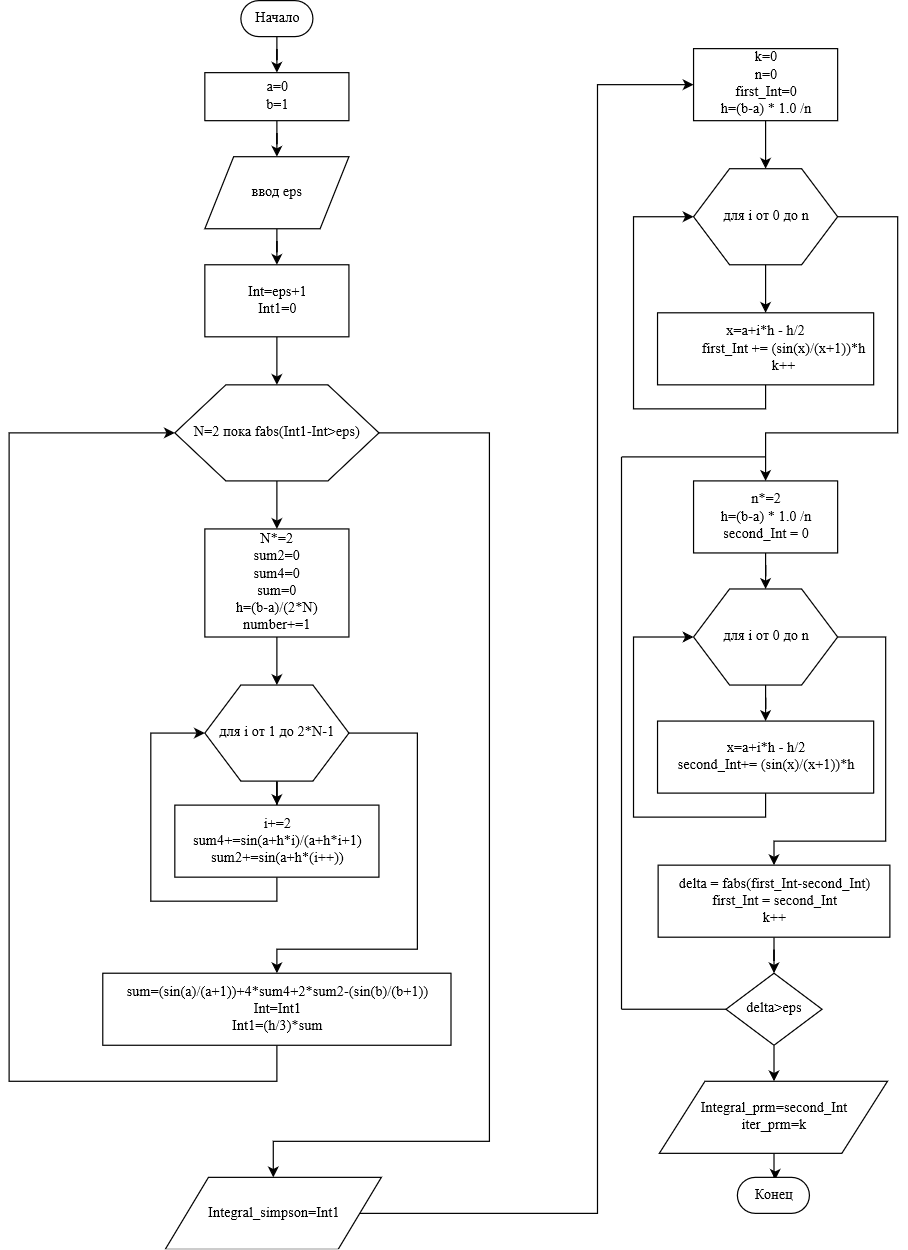
* 1. Присвоение переменной a значения 0
  2. Присвоение переменной b значения 1
  3. Ввод переменной eps
  4. Присвоение переменной Int значения eps+1
  5. Присвоение переменной Int1 значения 0
  6. Присвоение переменной N значения 2
  7. Если условие |Int1-Int|>eps не выполняется переход к пункту 4.20
  8. Присвоение переменной sum2 значения 0
  9. Присвоение переменной sum4 значения 0
  10. Присвоение переменной sum значения 0
  11. Присвоение переменной h значения (b-a)/(2\*N)
  12. Присвоение переменной I значения 1
  13. Если условие i<=2\*N-1 не выполняется переход к пункту 4.17
  14. Присвоение переменной sum4 значения sum4 + sin(a+h\*i)/(a+h\*i+1)
  15. Присвоение переменной sum2 значения sum2 + sin(a+h\*(i++))
  16. Присвоение переменной i значения i + 2
  17. Присвоение переменной sum значения (sin(a)/(a+1))+4\*sum4+2\*sum2-(sin(b)/(b+1))
  18. Присвоение переменной Int значения Int1
  19. Присвоение переменной Int1 значения (h/2.4)\*sum
  20. Присвоение переменной number значения number+1
  21. Вывод Integral\_simpson= Int1
  22. Вывод number
  23. Присвоение переменной k значения 0
  24. Присвоение переменной n значения 2
  25. Присвоение переменной first\_Int значения 0
  26. Присвоение переменной h значения (b-a) \* 1.0 /n
  27. Присвоение переменной i значения 0
  28. Если условие i < n не выполняется 4.31
  29. Присвоение переменной x значения a+i\*h - h/2
  30. Присвоение переменной first\_Int значения (sin(x)/(x+1))\*h
  31. Присвоение переменной k значения k+1
  32. Присвоение переменной n значения n\*2
  33. Присвоение переменной h значения (b-a) \* 1.0 /n
  34. Присвоение переменной second\_Int значения 0
  35. Присвоение переменной i значения 0
  36. Если условие i<n не выполняется переход 4.38
  37. Присвоение переменной x значения a+i\*h - h/2
  38. Присвоение переменной second\_Int значения (sin(x)/(x+1))\*h
  39. Присвоение переменной delta значения fabs(first\_Int-second\_Int)
  40. Присвоение переменной first\_Int значения second\_Int
  41. Присвоение переменной k значения k+1
  42. Если delta>eps переход к пункту 4.31
  43. Вывод Integral\_prm= second\_Int
  44. Вывод iter\_prm= k

**Конец алгоритма**

1. **Проект программы с нахождением значения интеграла по формуле Симпсона и методом прямоугольников**

Проект программы с определением замкнутых программных единиц и необходимых структур данных. В данном случае программа должна состоять из 23 частей: 1 блок ввода, цикла с параметром (содержит 5 блоков определения переменных, 1 блок проверки условия, 4 блока арифметических действий, 1 цикл с параметром[содержит блок проверки условия, два блока арифметических действий]) и 2 блока вывода, 4 блока арифметических действий, цикл с параметром(содержит блок определения переменной i, 2 блока арифметических действий). В качестве структуры данных могут быть введены переменные для хранения переменной определяющей точность вычислений, промежуточных переменных и результата.

1. **Опорный граф (блок-схема) алгоритма**



1. **Текст программы:**

**на языке С++**

#include<iostream>

#include<cmath>

#include<windows.h>

using namespace std;

int main()

{

double a, number, b, eps;//Нижний и верхний пределы интегрирования (a, b), погрешность (eps).

number=0;

a=0;

b=1;

cout << "eps=";

std::cin >> eps;

double Int, Int1;

Int=eps+1;

Int1=0;

//I-предыдущее вычисленное значение интеграла, I1-новое, с большим N.

for (int N=2; (fabs(Int1-Int)>eps); N\*=2)

{

double h, sum2=0, sum4=0, sum=0;

h=(b-a)/(2\*N);//Шаг интегрирования

for (int i=1; i<=2\*N-1; i+=2)

{

sum4+=sin(a+h\*i)/(a+h\*i+1);//Значения с нечётными индексами, которые нужно умножить на 4.

sum2+=sin(a+h\*(i++));//Значения с чётными индексами, которые нужно умножить на 2.

}

sum=(sin(a)/(a+1))+4\*sum4+2\*sum2-(sin(b)/(b+1));//Отнимаем значение f(b) так как ранее прибавили его дважды.

Int=Int1;

// std::cout << "Int=" << Int << '\n';

Int1=(h/2.4)\*sum;

// std::cout << "Int1=" << Int1 << '\n';

// std::cout << "delta=" << fabs(Int1-Int) << '\n';

number++;

}

cout << "Integral\_simpson=" << Int1 << endl;

cout << "iter\_simpson=" << number << endl;

int k=0;

int n=2;

double delta, h, x, first\_Int, second\_Int;

first\_Int = 0;

int i;

h=(b-a) \* 1.0 /n;

for(i=0;i<n;i++)

{

x=a+i\*h - h/2;

first\_Int += (sin(x)/(x+1))\*h;

//std::cout << "first\_Int= " << first\_Int << '\n';

k++;

}

do

{

n\*=2;

h=(b-a) \* 1.0 /n;

second\_Int = 0;

for (int i = 0; i < n; i++)

{

x=a+i\*h - h/2;

second\_Int+= (sin(x)/(x+1))\*h;

}

//std::cout << "second\_Int= " << second\_Int << '\n';

delta = fabs(first\_Int-second\_Int);

//std::cout << "delta= " << delta << '\n';

first\_Int = second\_Int;

k++;

} while (delta>eps);

std::cout << "Integral\_prm=" << second\_Int << '\n';

cout << "iter\_prm=" << k << endl;

return 0;}

1. **Проверка работоспособности (тестирование) программы**

Для проверки работоспособности программы, переменной eps присвоим значение 0.001. Тогда программа выведет четыре значения:

* + - 1. Integral\_simpson= 0.283678
      2. iter\_simpson=681
      3. Integral\_prm= 0.283678
      4. iter\_prm=1521

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.