БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет ЗВиДО

Специальность ПОИТ

Контрольная работа № 2

по дисциплине «Основы алгоритмизации и программирования»

часть 1

Вариант № 29

Выполнила: Марфель Т.В.

Минск 2018

1. **Условие задачи**

Для заданного преподавателем варианта составить программу на языке Паскаль. Программа должна быть откомментированной в соответствии с принятыми правилами. Предусмотреть простейший вывод на экран входных и выходных данных. Изобразить алгоритм графически методом Насси-Шнейдермана, используя средства текстового редактора Word.

**Индивидуальное задание №29**

Вычислить значение числа π, учитывая в формуле значения слагаемых, по абсолютной величине большие или равные введенной точности :

1. **Схема алгоритма методом Насси-Шнейдермана;**

Ввод аргументов m, n

sign := 1;

i := 1;

x := (16 / (i \* m) - 4 / (i \* n))

mstep := m;

nstep := n;

Чтение eps

(eps <= 0) or (eps > abs(x))

Да

Нет

Pi := Pi + x \* sign;

inc(j)

i := i + 2;

mstep := mstep \* m \* m

nstep := nstep \* n \* n

sign := sign \* (-1);

x :=(16/(i\*mstep)-4/(i\*nstep))

abs(x) <= eps

Str(eps, s);

Вывод eps, Pi, j

Вывод ошибки

Рис. 1 Диаграмма Насси-Шнейдермана

1. **Листинг**

**PROGRAM PiCalculation;**

**Uses sysutils;**

**Const**

**m = 5;** { Первый аргумент }

**n = 239;** { Второй аргумент }

**Var**

**s : String;** { Введенная точность конверсированная в строку }

**x, Pi, eps : Double;** { Слагаемое, сумма и введенная точность }

**i, j, sign, mstep, nstep : Integer;**{ Вспомогательные переменные }

**begin**

**Writeln('Please, enter the epsilon value ');**

**Readln(eps);** { Чтение введенной точности }

**if (eps <= 0) or (eps > abs(x)) then** { Обработка некорректных значений }

**begin**

**WriteLn('Error. The entered value should be less or equal to x.');**

**exit;**

**end;**

**sign := 1;** { Определение знака для первого слагаемого}

**i := 1;** { Начальное значение i }

**j := 0;** { Начальное значение j для подсчета количества суммируемых слагаемых }

**x := (16 / (i \* m) - 4 / (i \* n));** { Начальное значение x }

**Pi := 0;** { Начальное значение Pi}

**mstep := m;** { Начальное значение степени m }

**nstep := n;** { Начальное значение степени n }

**repeat** { Итерационный цикл для вычисления значения Pi }

**Pi := Pi + x \* sign;** { Вычисление первого слагаемого }

**inc(j);** { Увеличение количества просуммированных слагаемых }

**i := i + 2;** { Увеличение переменной-счетчика }

**mstep := mstep \* m \* m;** { Наращивание степени m }

**nstep := nstep \* n \* n;** { Наращивание степени n }

**sign := sign \* (-1);** { Изменение знака следующего слагаемого }

**x := (16 / (i \* mstep) - 4 / (i \* nstep));** { Вычисление следующего слагаемого }

**until abs(x) <= eps;** { Сравнение полученного слагаемого с введенной точностью }

**Str(eps, s);** { Конвертация введенной точности в строк }

**WriteLn('You assigned', eps: StrToInt(s[Length(s)]), ' value to epsilon.');** { Вывод исходной точности }

**WriteLn('As a result, Pi = ', Pi : 5 : StrToInt(s[Length(s)]), '.');** { Вывод полученной суммы с точностью, определенной пользователем }

**WriteLn('Overall, ', j, ' numbers were added.');** { Вывод количества просуммированных слагаемых }

**end.**

1. **Тестовый набор данных**

|  |  |
| --- | --- |
| **Входные данные (eps)** | **Результат работы программы** |
| 0.0001 | You assigned 1.0E-004 value to epsilon.  As a result, Pi = 3.1416.  Overall, 3 numbers were added. |
| 0.01 | You assigned 1.0E-002 value to epsilon.  As a result, Pi = 3.14.  Overall, 2 numbers were added. |
| 0 | Error. The entered value should be less or equal to x. |
| -0.01 | Error. The entered value should be less or equal to x. |
| 4 | Error. The entered value should be less or equal to x. |

1. **Описание схемы алгоритма**

Рисунок 1 на стр. 3 представляет собой диаграмму алгоритма вычисления Pi по методу Насси-Шнайдермана на основе ряда, определенного на стр.2 Структурограмма содержит блок решения и цикл «до», осуществляющий итерации до тех пор, пока очередное значение слагаемого не станет меньшим или равным введенной точности.

**Блок следования**

1. Определяются постоянные значения для m и n, являющимися неизменными для вычисления степенной функции.
2. Инициализируется переменная sign, которая позволит чередовать знаки при вычислении очередной степенной функции.
3. Инициализируется переменная i, являющаяся переменной-счетчиком и определяющая множитель и степень для общего члена ряда.
4. Присвоение переменной x формулы, по которой будет происходить вычисление очередного слагаемого.
5. Установление начального значения Pi равным 0.
6. Присвоение переменным mstep и nstep начального значения степени констант m и n. В дальнейшем эти переменные будут необходимы для определения шага для наращивания степени с каждой итерацией.
7. Ввод значения eps, определяющая точность для вычисления значения Pi и чтение введенного значения.

**Блок решения**

1. Производится проверка корректности введенного значения: если значение меньше или равно нулю или больше значения первого слагаемого для вычисления Pi, то выполнение программы завершается, и выводится ошибка. В противном случае происходит переход на итерационный цикл.

**Цикл «До»**

1. В первом блоке следования вычисляется значение сумме присваивается значение первого слагаемого в соответствии с начальными значениями переменной-счетчика и степени для констант m и n. Полученной значение для слагаемого умножается на переменную sign для чередования знаков на каждой итерации. Также прежде полученное значение суммы будет сохранятся в переменной Pi и дополняться вновь полученным значением для слагаемого.
2. Переменная j увеличивается на единицу с каждой новой итерацией.
3. Для вычисления нового слагаемого переменная i увеличивается на 2 с каждой новой итерацией, храня в себе значение, полученное в предыдущей.
4. В следующих двух блоках действия определяется шаг, с которым наращивается степень m и n. При этом в этих переменных хранится значение степени, полученное ранее.
5. Перед вычислением очередного слагаемого текущее значение переменной sign умножается на -1 и меняет знак данной и переменной и вследствие слагаемого на противоположный, что позволяет чередовать знаки для слагаемых.
6. Перед проверкой условия происходит вычисление очередного слагаемого в соответствии с формулой, определенной на шаге 4, а также значениям переменных, определенных в шагах 11-13.
7. Полученное значение x сравнивается с введенной точностью: если оно больше, то происходит переход на новую итерацию. Если значение меньше или равно введенной точности, то цикл завершается.
8. В последнем блоке действия осуществляется вывод результатов: значений введенной точности, полученной суммы и количества учтенных слагаемых.