БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет ЗВиДО

Специальность ПОИТ

Контрольная работа № 1

по дисциплине «Основы алгоритмизации и программирования»

часть 1

Вариант № 2

Выполнила: Марфель Т.В.

Минск 2018

1. **Условие задачи**

Циклический вычислительный процесс. Для заданного преподавателем варианта составить алгоритм решения задачи. Нарисовать три варианта графического представления алгоритма: а) в соответствии с ГОСТ 19.701-90; б) по методу Дамке; в) используя диаграмму Насси-Шнейдермана. Алгоритмы изобразить средствами Word.

**Индивидуальное задание №29**

Найти совершенные числа на отрезке [М, N]. (Число называется совершенным, если оно равно сумме своих делителей, включая 1, но не включая самое себя).

1. **Схема алгоритма.**

Ввод m, n

Цикл i

От i=m до n

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

Ввод аргументов m, n

i := 1;

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

mstep := m;

nstep := n;

Чтение eps

Pi := Pi + x \* sign;

inc(j)

i := i + 2;

mstep := mstep \* m \* m

nstep := nstep \* n \* n

sign := sign \* (-1);

sign := sign \* (-1);

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

Да

Нет

sign := 1;

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**

**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 }

**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;**

**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, пока очередное значение слагаемого не станет меньшим или равным введенной точности.

На входе в программу задаются значения констант m, n

Переменная **i** является переменной счетчиком с начальным значением 1 и определяет количество выполненных итераций для внешнего цикла.

В начале цикла для текущей итерации вычисляется значение ***f*1** в соответствии с заданным значением x.

Перед первым итерационным циклом для вычисления функции с точностью , инициализируются переменные k1, являющейся переменной счетчиком для данного ряда и определяющей количество слагаемых необходимых для достижения заданной точности, и *f2\_*1, а также определяется начальное значение степени общего члена ряда **x** – переменная **step**.

На входе в итерационный цикл значение **k1** увеличивается на 1, в соответствии с этим вычисляется новое значение **step**. В переменной *f2\_*1 накапливается значение функции для заданного аргумента **x**. При проверке постусловия достигнутое значение ***f2\_*1** сравнивается с заданной точностью: если значение больше, то начинается новая итерация, в ходе которой наращиваются значения переменных, сохраненных на этапе предыдущей. Если же значение меньшей или равно заданной точности, то происходит переход к следующему циклу для вычисления функции с точностью . Перед началом цикла переменной счетчику k2 присваивается значение **k1**, а ***f2\_*2** значение ***f2\_*1**, тем самым вычисления начинаются с уже достигнутых количества слагаемых и суммы. Внутри цикла осуществляются операции и проверки идентичные тем, что были описаны для ***f2\_*1**. По окончанию итерации, как и ранее, осуществляется сравнение значения ***f2\_*2** с заданнойточностью: если условие выполнено, то осуществляется переход к следующему циклу для вычисления функции с точностью . Перед началом цикла переменной счетчику k3 присваивается значение k2, а ***f2\_*3** значение ***f2\_*2**. Внутри цикла осуществляются операции и проверки идентичные тем, что были описаны для ***f2\_*1**. По окончанию итерации, как и ранее, осуществляется сравнение значения ***f2\_*3** с заданнойточностью: если условие выполнено, то осуществляется вывод результатов для данной итерации.

После вывода результатов значение **x** увеличивается в соответствии с заданным шагом **b**, а также значение переменной-счетчика **i**. Затем происходит переход в начало цикла, где осуществляется проверка числа повторений. Если значение становится выше 20, то выполнение программы завершается.