Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра O7 «Информационные системы и программная инженерия»

**Практическая работа №2**по дисциплине «Информатика: Основы программирования»  
на тему «Ветвления и циклы»  
  
Вариант 17

Выполнил:  
Студент Усов Д.А.  
Группа Е123Б  
  
Преподаватель:  
Лестенко Н.А.

Санкт-Петербург  
2022 г.

1. Вычислить значение функции , используя условную операцию ?:

*Исходные данные:*

Аргументы функции m и n. Так как значения m и n могут быть любыми, объявим соответствующие переменные типа double.

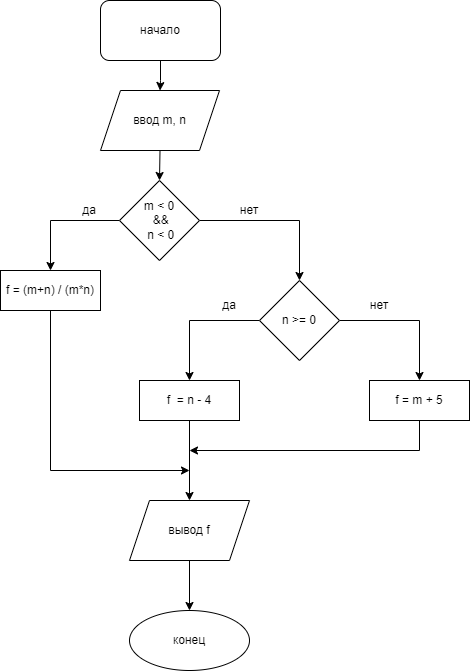
*Результирующие данные:*

Значение функции f , соответствующая переменная тоже будет типа double.

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| m = -1, n = -2 | -1.5 | -1.500000 |
| m = 5, n = 4 | 0 | 0.000000 |
| m = 4, n = -3 | 9 | 9.000000 |
| M = 0, n = -6 | 5 | 5.000000 |

Схема программы



Текст программы

#include <stdio.h>  
  
int main(){  
 double m,n,f; /\* объявление переменных \*/  
 printf("m = ");  
 scanf("%lf",&m); /\* ввод аргументов функции m,n \*/  
 printf("n = ");  
 scanf("%lf",&n);  
 f = m < 0 && n < 0 ? (m+n)/(m\*n) : n >= 0 ? n- 4 : m + 5;  
 /\* вложенное условие: при ложности условия первого усл. оператора,  
 программа переходит во второй усл. оператор\*/  
 printf("f = %lf",f); /\* вывод результата \*/  
 return 0;  
}

1. Вычислить значение функции

*Исходные данные:*

Аргументы x и y — действительные числа, тип double*.*

*Результирующие данные:*

A – результат функции, тип double.

*Предварительные вычисления:*

Чтобы можно было вычислить значение функции, должны быть выполнены следующие условия: знаменатель не должен быть равен 0, y ≠ 0.

*Вспомогательные переменные:*

d – знаменатель дроби, переменная типа double.

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| x = 1, y = 2 | -0.095158958 | -0.095158958 |
| x = 5, y = 0 | Функция не определена | Функция не определена |
| x = -6, y = -8 | -51.443123889 | -51.443123889 |
| x = 3, y = -90 | -0.000099959 | -0.000099959 |

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
#include <math.h>  
  
int main() {  
 double x, y, a, d; // объявление переменных  
 printf("x = ");  
 scanf("%lf", &x);  
 printf("y = ");  
 scanf("%lf", &y); // ввод аргументов функции  
 if (y == 0) { // проверка на соответствие ООФ  
 printf("Функция не определена"); // вывод сообщения об ошибке  
 } else {  
 d = log(fabs(4 \* pow(y, 3))); // знаменатель  
 if (d == 0) { // проверка на соответствие ООФ  
 printf("Функция не определена"); // вывод сообщения об ошибке  
 } else {  
 a = (sin(x) \* cos(y) - tan(x / y)) / d \* exp(-x);  
  
 printf("a = %.9lf", a); // вычисление и вывод результата  
 }  
 }  
 return 0;  
}

1. Определить, лежат ли три точки с координатами (x1, y1), (x2, y2), (x3, y3) на одной прямой. Если не лежат, вычислить расстояния до каждой точки из начала координат. Если лежат, получить общее уравнение этой прямой вида Ax+By+C = 0, где A ≥ 0.

*Исходные данные:*

x\_1, y\_1 – координаты первой точки, тип double;

x\_2, y\_2 – координаты первой точки, тип double;

x\_3, y\_3 – координаты первой точки, тип double;

*Результирующие данные:*

*описываем выходные данные, их обозначение в программе и тип*.

*Вспомогательные переменные:*

*если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.*

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| *наборы данных должны соответствовать максимально большему количеству возможных вариантов* |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
#include <math.h>  
  
int main() {  
 double x\_1, x\_2, x\_3, y\_1, y\_2, y\_3, A, B, C, d\_1, d\_2, d\_3;  
 printf("Enter x\_1 y\_1 x\_2 y\_2 x\_3 y\_3:");  
 scanf("%lf %lf %lf %lf %lf %lf", &x\_1, &y\_1, &x\_2, &y\_2, &x\_3, &y\_3);  
 if (!((x\_1 - x\_3) \* (y\_2 - y\_3) - (x\_2 - x\_3) \* (y\_1 - y\_3))) {  
 printf("Yes\n");  
 A = y\_2 - y\_1;  
 B = (x\_1 - x\_2) \* (A >= 0 ? 1 : -1);  
 C = (y\_1 \* (x\_2 - x\_1) + x\_1 \* (y\_1 - y\_2))\* (A >= 0 ? 1 : -1);  
 printf("%gx %c %gy %c %g = 0", fabs(A), B < 0 ? '-' : '+', fabs(B), C < 0 ? '-' : '+', fabs(C));  
 } else {  
 d\_1 = sqrt(x\_1 \* x\_1 + y\_1 \* y\_1);  
 d\_2 = sqrt(x\_2 \* x\_2 + y\_2 \* y\_2);  
 d\_3 = sqrt(x\_3 \* x\_3 + y\_3 \* y\_3);  
 printf("No\n");  
 printf("Distance for x\_1: %g, x\_2: %g, x\_3: %g", d\_1, d\_2, d\_3);  
 }  
 return 0;  
}

1. Получить таблицу температур по Цельсию от 0° до 100° и их эквивалентов по шкале Фаренгейта, используя для перевода формулу . Использовать управляющую инструкцию *for*.

*Исходные данные:*

*Все двузначные десятичные числа, которые будут анализироваться по очереди, поэтому для их представления достаточно одной переменной n типа char.*

*Результирующие данные:*

*Вывод сообщения. Отдельной переменной не требуется.*

*Вспомогательные переменные:*

*если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.*

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
|  |  |  |
|  |  |  |
|  |  |  |

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
  
int main(){  
 int t\_c;  
 double t\_f;  
  
 for (t\_c = 0; t\_c<=100; t\_c++){  
 t\_f = 9./5\*t\_c+32;  
 printf("%d\t%g\n", t\_c, t\_f);  
 }  
  
 return 0;  
}

1. Вычислить сумму кубов всех четных чисел, лежащих в диапазоне [X, Y], где X и Y – вводимые с клавиатуры натуральные числа.

*Исходные данные:*

*описываем входные данные, их обозначение в программе и тип.*

*Результирующие данные:*

*описываем выходные данные, их обозначение в программе и тип*.

*Вспомогательные переменные:*

*если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.*

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
|  |  |  |
|  |  |  |
|  |  |  |

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
#include <limits.h>  
  
int main(){  
 unsigned long long x,y, s = 0;  
 printf("Enter x y:\n");  
 scanf("%lld %lld", &x,&y);  
 if (x>=y){ // Проверка на то что первое число меньше второго  
 printf("The first number must be less than the second");  
 return 1;  
 }  
  
 for (; x<=y; x++){  
 if (LLONG\_MAX - x\*x >= s) { // Проверка на переполнение  
 if (!(x%2)) { // Проверка на четность  
 s += x \* x;  
 }  
 } else{  
 printf("You have entered too large a range or too large numbers");  
 return 2;  
 }  
 }  
 printf("%lld", s);  
 return 0;  
}

1. С клавиатуры вводится число N. Определить, может ли оно быть двоичным (т.е. состоять только из 0 и 1).

*Исходные данные:*

*описываем входные данные, их обозначение в программе и тип.*

*Результирующие данные:*

*описываем выходные данные, их обозначение в программе и тип*.

*Вспомогательные переменные:*

*если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.*

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
|  |  |  |
|  |  |  |
|  |  |  |

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
  
int main ()  
{  
 int c;  
 while ((c = getchar()) != '\n')  
 if (c != '0' && c != '1'){  
 printf("No");  
 return 1;  
 }  
 printf("Yes");  
 return 0;  
}

1. Вычислить значение суммы бесконечного ряда с заданной точностью ε=10-6 и значение функции (для проверки) . Использовать рекуррентные зависимости для вычисления значений слагаемых.

*Исходные данные:*

Аргумент функции. Тип аргумента в задании не указан, поэтому переменная х будет типа double.

*Результирующие данные:*

Значение суммы s тоже будет типа double.

*Вспомогательные переменные:*

n – индекс слагаемого – целое число типа int, a – значение текущего слагаемого – вещественное число типа double.

*Предварительные вычисления:*

n-ное слагаемое ,

предшествующее слагаемое

коэффициент пропорциональности

индекс первого слагаемого в рекуррентной последовательности n=0, первое слагаемое

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
#include <stdlib.h>  
#include <math.h>  
  
#define EPS 1e-6  
  
int main(void) {  
 double x, a, s;  
 int n = 0; /\*индекс первого слагаемого 0\*/  
 printf("x=");  
 scanf("%lf", &x);  
 a = 0.5;  
 s = 0.5 + a;  
/\*пока слагаемое по модулю больше заданной точности\*/  
 while (fabs(a) > EPS) {  
 n++; /\*увеличиваем индекс слагаемого\*/  
 a \*= -4 \* x \* x / (2 \* n \* (2 \* n - 1));  
/\*вычисляем очередное слагаемое домножением  
предыдущего на коэффициент пропорциональности\*/  
 s += a; /\*накапливаем сумму\*/  
 }  
 printf("s=%lf\n cos(%lf)^2=%lf\n",  
 s, x, cos(x) \* cos(x));  
 system("pause");  
 return 0;  
}

*Результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| x = 0 |  | s = 1.000000 |
| x = 5 |  | s = 0.080464 |

1. Вычислить значение суммы бесконечного ряда с заданной точностью ε=10-6 и значение функции (для проверки) учесть, что . Использовать рекуррентные зависимости для вычисления значений слагаемых по частям.

*Исходные данные:*

*описываем входные данные, их обозначение в программе и тип.*

*Результирующие данные:*

*описываем выходные данные, их обозначение в программе и тип*.

*Вспомогательные переменные:*

*если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.*

*Предварительные вычисления:*

n-ое слагаемое

предшествующее слагаемое

коэффициент пропорциональности

индекс первого слагаемого в рекуррентной последовательности

n=1,

первое слагаемое

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

#include <stdio.h>  
#include <math.h>  
  
#define EPS 1e-6  
  
int main(void) {  
 double x, a, s;  
 int n = 1; /\*индекс первого слагаемого 0\*/  
 printf("x=");  
 scanf("%lf", &x);  
 if (!(0.1 <=x && x <=0.8)) /\*если x не принадлежит отрезку [0.1;0.8]\*/  
 { /\*выводим сообщение об этом\*/  
 printf("Invalid value x\n");  
  
 return 1; /\*и прерываем программу\*/  
 }  
 a = 0.5 \* x;  
 s = a;  
 while (fabs(a) > EPS) {/\*пока слагаемое по модулю больше заданной точности\*/  
 n++; /\*увеличиваем индекс слагаемого\*/  
 a \*= (x \* cos(M\_PI \* n / 3) \* (n - 1)) / (n \* cos((M\_PI \* n - M\_PI) / 3));  
 /\*вычисляем очередное слагаемое домножением  
 предыдущего на коэффициент пропорциональности\*/  
 s += a; /\*накапливаем сумму\*/  
 }  
 printf("s=%.6lf\n y(%lf)=%.6lf\n",  
 s, x, -0.5 \* log(1 - 2 \* x \* cos(M\_PI / 3) + pow(x, 2)));  
  
 return 0;  
}

*Результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
|  | *значение функции, указанной в задании для проверки* |  |
|  |  |  |
|  |  |  |

1. Дано целое число а. Найти все такие простые числа p, чтобы дробь была сократима на p. Простым называется натуральное число N, не имеющее других делителей, кроме 1 и самого N.

*Исходные данные:*

*описываем входные данные, их обозначение в программе и тип.*

*Результирующие данные:*

*описываем выходные данные, их обозначение в программе и тип*.

*Вспомогательные переменные:*

*если нужны вспомогательные переменные, описываем, для чего они нужны, их обозначение и тип.*

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
|  |  |  |
|  |  |  |
|  |  |  |

Схема программы:

*Здесь должна быть схема программы в соответствии с ГОСТ 19.701-90. Схема может быть построена любым способом, в том числе начерчена вручную на листе бумаги с помощью карандаша и линейки и сфотографирована или отсканирована.*

Текст программы

*Сюда добавляем текст программы* ***с комментариями****. Шрифт Courier New или FreeMono 10 пт, междустрочный интервал одинарный.*