МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет  
по практическим работам

по дисциплине «Основы алгоритмизации и программирования»

Выполнила Проверил

студентка группы ГИ-11 ст. преп. каф. «ИУСиТ»

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Оглавление

[Практическая работа № 19 Тема: «Разработка алгоритма, составление, отладка и выполнение программы с использванием рекурсивных функций» 3](#_Toc36920850)

[1.1 Задания 3](#_Toc36920851)

[1.2 Контрольные вопросы 7](#_Toc36920852)

# Практическая работа № 19 Тема: «Разработка алгоритма, составление, отладка и выполнение программы с использванием рекурсивных функций»

### **Цель**

Разработать алгоритмы, составить и выполнить программы с использованием рекурсивных функций.

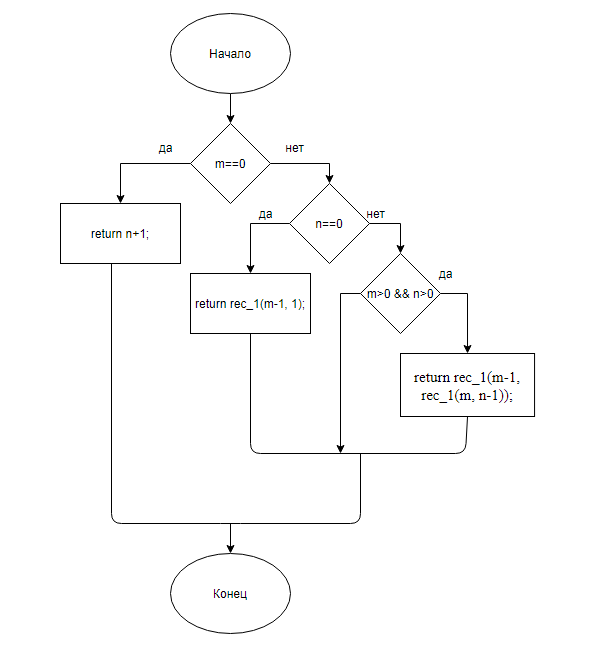
## Задания

### Задание 1

### Условие

Решить задачу двумя способами – с применением рекурсии и без нее.Найти значение функции Аккермана A(m, n), которая определяется для всех неотрицательных целых аргументов m и n следующим образом: A(0, n)= n + 1,если m = 0; A(m,0) = A(m – 1, 1), если n = 0; A(m, n) = A(m – 1, A(m, n – 1)), если и m > 0, и n > 0.

**Блок-схема**



### Программный код

void task\_1(int m, int n)

{int key;

printf("Рекурсия - %d\n", key=rec\_1(m, n));

printf("Проверка через цикл - %d\n", key);

}

int rec\_1(int m, int n)

{

if(m==0)

return n+1;

else if(n==0)

return rec\_1(m-1, 1);

else if(m>0 && n>0)

return rec\_1(m-1, rec\_1(m, n-1));

}

### Результат

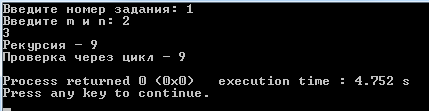


Рисунок – Решение задания 1

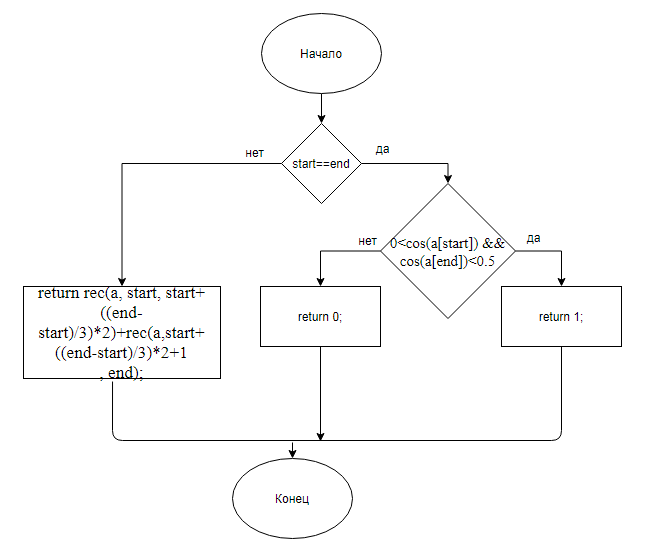
### Задание 2

### Условие

### Ввести с клавиатуры (сгенерировать) одномерный массив целых чисел. Решить задачу путем рекурсивного разбиения массива на две части. Для контроля решить задачу с использованием циклического алгоритма.

Определить количество элементов массива, для которых выполняется условие 0 < cos(a[i]) < 0.5. При рекурсивном разбиении массив делить на первые 2/3 и остальную треть массива.

### Блок-схема



### Программный код

void task\_2(int \*a)

{int key=0, start=0,end=9, i;

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

{

if(0<cos(a[i]) && cos(a[i])<0.5)

{

key++;

}

}

printf("\nРекурсия - %d\n", rec(a, start, end));

printf("Проверка через цикл - %d\n", key);

}

int rec(int \*a, int start, int end)

{

if(start==end)

{

if(0<cos(a[start]) && cos(a[end])<0.5)

{

return 1;

}

else {return 0;}

}

else

{

return rec(a, start, start+((end-start)/3)\*2)+rec(a,start+((end-start)/3)\*2+1 , end);

}

}

### Результат

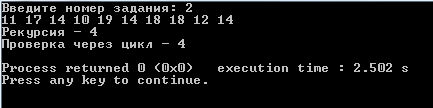


Рисунок 2 – Решение задания 2

### Задание 3

### Условие

Вычислить сумму ряда из задания 3 практической работы № 9 с применением рекурсии.

### Программный код

double rec\_3(int n,double x, double eps, double Sn)

{

if(fabs(Sn)<eps)

return 0;

Sn\*=(-1)\*2\*x/n;

++n;

return Sn + rec\_3(n, x, eps, Sn);

}

void task\_3()

{

int n=1;

double x, eps, S, Sn;

printf("x=");

scanf("%lf", &x);

while(!(x>=-2 && x<=-1))

{printf("x=");

scanf("%lf",&x);

}

printf("eps=");

scanf("%lf",&eps);

S=Sn=1;

S += rec\_3(n,x,eps,Sn);

printf("S= %.15lf\n", S);

printf("e^(-2x)= %lf",exp(-2\*x));

}

### Результат

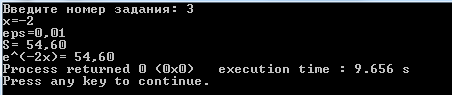


Рисунок 3 – Решение задания 3

## Контрольные вопросы

1. Понятие рекурсии.

Рекурсия – это функция, которая сама вызывает себя.

1. Структура рекурсивной функции.

Рекурсивная функция состоит из двух частей: базовой и рекурсивной.

1. Условие окончания рекурсивного алгоритма.

В тексте рекурсивной функции обязательно должен быть выполнен условный оператор, например if, который при определенных условиях вызовет завершение функции, т.е. возврат, а не выполнит очередной рекурсивный вызов. Если такого оператора нет, то после вызова функция никогда не сможет завершить работы. Распространенной ошибкой при написании рекурсивных функций как раз и является отсутствие в них условного оператора.

1. Достоинство рекурсии.

Естественность (натуральность) представления сложных, на первый взгляд, алгоритмов;

рекурсивный алгоритм более читабелен в сравнении с итерационным;

для многих распространенных задач рекурсию более легко реализовать чем

итерацию. Рекурсия хорошо подходит для реализации алгоритмов обхода

списков, деревьев, анализаторов выражений, комбинаторных задач и т.д.

1. Недостатки рекурсии.

По сравнению с итерацией многоразовый вызов рекурсивной функции требует больше времени. Это связано с тем, что при вызове рекурсивного метода его параметры копируются в стек. При завершении вызова рекурсивной функции предыдущие значения параметров вытягиваются из стека, что приводит к лишним операциям. Итерационный алгоритм для той же задачи работает быстрее;

Если рекурсивная функция содержит большие объемы локальных

внутренних переменных и большое количество параметров, то использование рекурсии не является эффективным. Это связано с тем, что для каждого рекурсивного вызова нужно делать копии этих переменных и параметров. При большом количестве рекурсивных вызовов это приведет к чрезмерному

использованию памяти.

### Вывод по практической работе

Разработал алгоритмы, составил и выполнил программы с использованием рекурсивных функций.