СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной курсовой работы является изучение структуры подпрограммы, механизмов передачи параметров в подпрограмму, возврата результатов из подпрограммы и вызова подпрограммы на выполнение с использованием языка программирования С++.

С++ - универсальный язык программирования, задуманный так, чтобы сделать программирование более приятным для серьезного программиста. За исключением второстепенных деталей С++ является надмножеством языка программирования С. Помимо возможностей, которые дает С, С++ предоставляет гибкие и эффективные средства определения новых типов. Используя определения новых типов, точно отвечающих концепциям приложения, программист может разделять разрабатываемую программу на легко поддающиеся контролю части. Такой метод построения программ часто называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных [1].

Программы на языке СИ обычно состоят из большого числа отдельных *функций* (подпрограмм). В математике понятие *функции* хорошо известно – с помощью функций задаются самые различные зависимости одних значений (являющихся значением функции) от других значений, называемых *аргументами* функции. Эти зависимости могут задаваться различными способами: таблично, графически, аналитически и т.д. В отличие от общематематического понятия функции, в алгоритмических языках, в том числе и в Си, рассматриваются только такие функции, для которых можно задать алгоритм определения их значений [2].

При этом в Си допустимы только такие функции, значения которых относятся к простым типам, так что значением функции не может быть, например, массив.

Если необходимость использования какого-либо фрагмента программы встречается в нескольких местах программы, то было бы неразумно каждый раз выписывать соответствующий алгоритм. Удобнее однажды определить требуемый фрагмент программы, дав ему некоторое имя, а в случае необходимости использовать этот фрагмент путем указания ее имени и задания конкретных значений аргументов. Как правило, такие фрагменты имеют небольшие размеры и могут находиться как в одном, так и в нескольких файлах.

Связь между функциями осуществляется через аргументы, возвращаемые значения и внешние переменные. Передача значения из вызванной функции в вызвавшую происходит с помощью оператора возврата.

1 ИСПОЛЬЗОВАНИЕ СРЕДСТВ ЯЗЫКА ПРОГРАММИРОВАНИЯ И ЧИСЛЕННЫХ МЕТОДОВ В ПРОГРАММИРОВАНИИ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ

1.1 Структура, объявление и вызов функции

Мощность языка СИ во многом определяется легкостью и гибкостью в определении и использовании функций в СИ-программах. В отличие от других языков программирования высокого уровня в языке СИ нет деления на процедуры, подпрограммы и функции, здесь вся программа строится только из функций.

Функция - это совокупность объявлений и операторов, обычно предназначенная для решения определенной задачи. Каждая функция должна иметь имя, которое используется для ее объявления, определения и вызова. В любой программе на СИ должна быть функция с именем main (главная функция), именно с этой функции, в каком бы месте программы она не находилась, начинается выполнение программы [3].

При вызове функции ей при помощи аргументов (формальных параметров) могут быть переданы некоторые значения (фактические параметры), используемые во время выполнения функции. Функция может возвращать некоторое (одно !) значение. Это возвращаемое значение и есть результат выполнения функции, который при выполнении программы подставляется в точку вызова функции, где бы этот вызов ни встретился. Допускается также использовать функции не имеющие аргументов и функции не возвращающие никаких значений. Действие таких функций может состоять, например, в изменении значений некоторых переменных, выводе на печать некоторых текстов и т.п..

С использованием функций в языке СИ связаны три понятия - определение функции (описание действий, выполняемых функцией), объявление функции (задание формы обращения к функции) и вызов функции.

Определение функции задает тип возвращаемого значения, имя функции, типы и число формальных параметров, а также объявления переменных и операторы, называемые телом функции, и определяющие действие функции. В определении функции также может быть задан класс памяти.

В языке СИ нет требования, чтобы определение функции обязательно предшествовало ее вызову. Определения используемых функций могут следовать за определением функции main, перед ним, или находится в другом файле.

Однако для того, чтобы компилятор мог осуществить проверку соответствия типов передаваемых фактических параметров типам формальных параметров до вызова функции нужно поместить объявление (прототип) функции.

Объявление функции имеет такой же вид, что и определение функции, с той лишь разницей, что тело функции отсутствует, и имена формальных параметров тоже могут быть опущены. Для функции, определенной в последнем примере, прототип может иметь вид

int rus (unsigned char r); или rus (unsigned char);

В программах на языке СИ широко используются, так называемые, библиотечные функции, т.е. функции предварительно разработанные и записанные в библиотеки. Прототипы библиотечных функций находятся в специальных заголовочных файлах, поставляемых вместе с библиотеками в составе систем программирования, и включаются в программу с помощью директивы #include.

Если объявление функции не задано, то по умолчанию строится прототип функции на основе анализа первой ссылки на функцию, будь то вызов функции или определение. Однако такой прототип не всегда согласуется с последующим определением или вызовом функции. Рекомендуется всегда задавать прототип функции. Это позволит компилятору либо выдавать диагностические сообщения, при неправильном использовании функции, либо корректным образом регулировать несоответствие аргументов устанавливаемое при выполнении программы [3].

Объявление параметров функции при ее определении может быть выполнено в так называемом "старом стиле", при котором в скобках после имени функции следуют только имена параметров, а после скобок объявления типов параметров.

Необязательный спецификатор-класса-памяти задает класс памяти функции, который может быть static или extern. Подробно классы памяти будут рассмотрены в следующем разделе.

Спецификатор-типа функции задает тип возвращаемого значения и может задавать любой тип. Если спецификатор-типа не задан, то предполагается, что функция возвращает значение типа int.

Функция не может возвращать массив или функцию, но может возвращать указатель на любой тип, в том числе и на массив и на функцию. Тип возвращаемого значения, задаваемый в определении функции, должен соответствовать типу в объявлении этой функции.

Функция возвращает значение если ее выполнение заканчивается оператором return, содержащим некоторое выражение. Указанное выражение вычисляется, преобразуется, если необходимо, к типу возвращаемого значения и возвращается в точку вызова функции в качестве результата. Если оператор return не содержит выражения или выполнение функции завершается после выполнения последнего ее оператора (без выполнения оператора return), то возвращаемое значение не определено. Для функций, не использующих возвращаемое значение, должен быть использован тип void,

указывающий на отсутствие возвращаемого значения. Если функция определена как функция, возвращающая некоторое значение, а в операторе return при выходе из нее отсутствует выражение, то поведение вызывающей функции после передачи ей управления может быть непредсказуемым [3].

Список-формальных-параметров - это последовательность объявлений формальных параметров, разделенная запятыми. Формальные параметры - это переменные, используемые внутри тела функции и получающие значение при вызове функции путем копирования в них значений соответствующих фактических параметров. Список-формальных-параметров может заканчиваться запятой (,) или запятой с многоточием (,...), это означает, что число аргументов функции переменно. Однако предполагается, что функция имеет, по крайней мере, столько обязательных аргументов, сколько формальных параметров задано перед последней запятой в списке параметров. Такой функции может быть передано большее число аргументов, но над дополнительными аргументами не проводится контроль типов.

Любая функция в программе на языке СИ может быть вызвана рекурсивно, т.е. она может вызывать саму себя. Компилятор допускает любое число рекурсивных вызовов. При каждом вызове для формальных параметров и переменных с классом памяти auto и register выделяется новая область памяти, так что их значения из предыдущих вызовов не теряются, но в каждый момент времени доступны только значения текущего вызова.

Переменные, объявленные с классом памяти static, не требуют выделения новой области памяти при каждом рекурсивном вызове функции и их значения доступны в течение всего времени выполнения программы.

1.2 Численное интегрирование. Метод трапеций

Пусть для функции f(x) требуется вычислить $\int_a^b f(x)dx$. Для этого отрезок [a,b] разбивается с помощью равноотстоящих точек (1.1)

$$x0 = a, xi = x0 + i \cdot h, (i = 1, \dots, n-1), xn = b \qquad (1.1)$$

на n равных частей и выбирается шаг интегрирования (1.2)

$$h = \frac{b-a}{n} \qquad (1.2)$$

Функция f(x) на каждом из элементарных отрезков[xi,xi+1] заменяется прямой (рисунок 1.1). Тогда для отрезка $[x_i,x_{i+1}]$ (1.3)

$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h}{2}\big(f(x_i) + f(x_{i+1})\big), i = 0, \dots, n-1 \qquad (1.3)$$

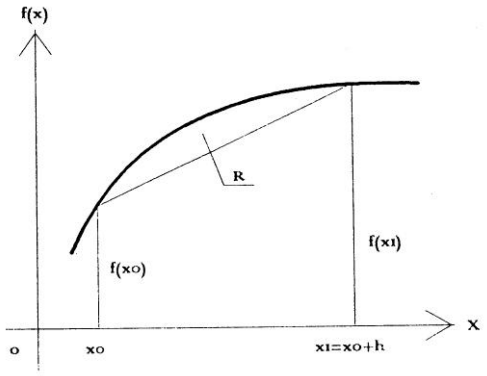


Рисунок 1.1 – График функции и разбиение ее на трапеции

И если просуммировать все элементарные площади по всему интервалу интегрирования, то получится формула, которая называется *обобщенной формулой трапеций* (1.4):

$$\int_a^b f(x)dx = \frac{h}{2}(y_0 - y_1) + \frac{h}{2}(y_1 - y_2) + \cdots + \frac{h}{2}(y_{n-1} - y_n) = \\ = \frac{h}{2}(y_0 + 2y_1 + \cdots + 2y_{n-1} + y_n) \qquad (1.4)$$

y_i - значения подынтегральной функции f(x) в узлах x_i;

h – шаг интегрирования;

I - значение интеграла.

Из рисунка 1.1 видно что на каждом элементарном отрезке $[x_i,x_{i+1}]$ функция f(x) заменяется интерполяционным многочленом Лагранжа 1-й степени.

1.2АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ЗАДАЧИ

2.1 Постановка задачи

2.1.1 Задача 1

В задаче №1 необходимо вычислить расстояние l между точками с координатами x1,y1 и x2,y2 по формуле 2:

$$l = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad (2.1)$$

Задачу можно разбить на такие подзадачи:

1. Ввод координат точек;
2. Вычисление расстояния между точками;
3. Вывод результатов.

2.1.2 Задача 2

В задаче №2 требуется вычислить значение функции 3:

$$L = tg^3 c - 0{,}58 \cos d^2 + f, \quad (2.2)$$

где c,d и f – количества значений, попадающих в интервале (1;3), стоящих на нечётных местах массивов X,Y и T, - соответственно.

Задачу можно разбить на такие подзадачи:

1. Ввод одномерного массива;
2. Вывод одномерного массива;
3. Определение количества элементов массива, попадающих в интервале (1;3), стоящих на нечетных местах;
4. Вычисление значения выражения;
5. Вывод результатов.

2.1.3 Задача 3

Сформировать векторы B и C, элементами которых являются минимальные значения строк матриц X и T соответственно.

Задачу можно разбить на такие подзадачи:

1. Ввод матрицы;
2. Вывод матрицы;
3. Формирование вектора;
4. Вывод сформированного вектора.

2.1.4 Задача 4

Необходимо разработать и оформить в виде графической схемы алгоритм вычисления определенного интеграла $\int_a^b f(x)$ с точностью ξ методом трапеций. Написать подпрограмму вычисления определенного интеграла с передачей имени подынтегральной функции и пределов интегрирования в качестве параметра. Написать подпрограммы для вычисления подынтегральных функций $f1(x)$, $f2(x)$, $f3(x)$.

$$f1(x) = \frac{sin^4 x}{\cos x} \tag{2.3}$$

$$f2(x) = 3^{0,1x} + \sin(0,2x) \tag{2.4}$$

$$f3(x) = (x - 2,2)(x - 1,1)(x + 6,1) \tag{2.5}$$

Разработать алгоритм вычисления определенного интеграла выбранной пользователем функции. Выбор функции осуществлять с помощью меню простого выбора.

2.1.5 Основная задача

После решения четырех поставленных задач, можно составить алгоритм решения основной задачи, заключающейся в выводе меню и решении выбранной задачи. Её алгоритм изображен на рисунке 2.24. Текст программы находится в приложении Д.

2.2 Графические схемы алгоритмов

Задача 1

На рисунке 2.1 изображена схема основного алгоритма для задачи №1.

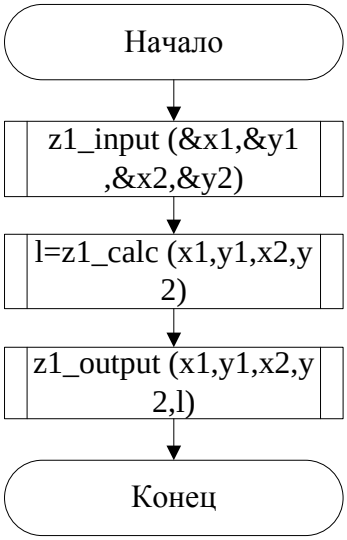


Рисунок 2.1 – Графическая схема основного алгоритма задачи №1

На рисунке 2.2 изображена схема алгоритма ввода координат точек. В подпрограмму передаются указатели на координаты точек.

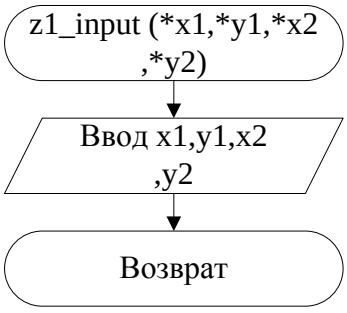


Рисунок 2.2 – Графическая схема алгоритма ввода координат точек

На рисунке 2.3 изображена схема алгоритма вычисления расстояния между точками. В подпрограмму передаются координаты точек. Подпрограмма возвращает расстояние между точками.

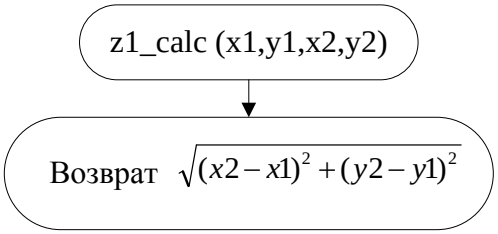


Рисунок 2.3 – Графическая схема алгоритма расчета расстояния между точками

На рисунке 2.4 изображена схема алгоритма вывода координат точек и расстояния между ними. В подпрограмму передаются координаты точек и расстояние.

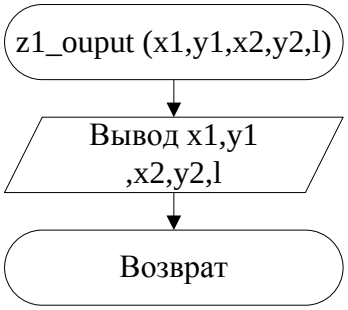
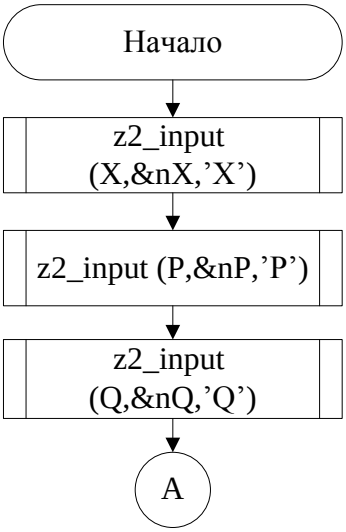


Рисунок 2.4 – Графическая схема алгоритма вывода координат точек и расстояния

Задача 2

На рисунке 2.5 изображена схема основного алгоритма задачи №2.



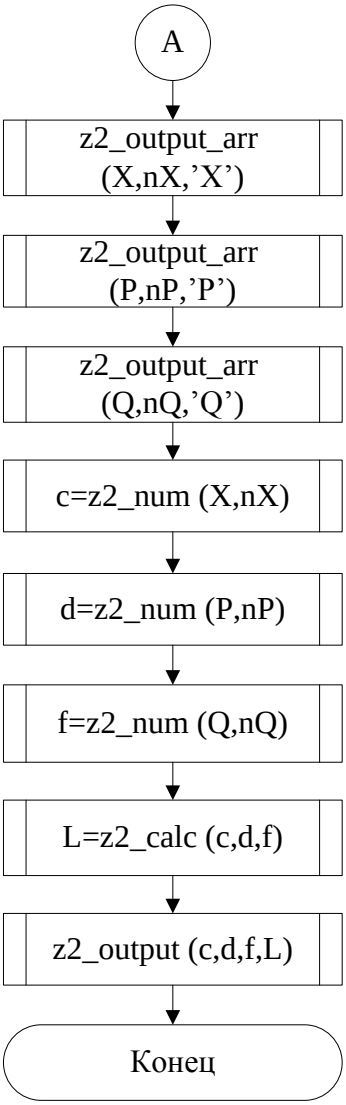


Рисунок 2.5 – Графическая схема основного алгоритма задачи №2

На рисунке 2.6 изображёна схема алгоритма ввода одномерного массива. В подпрограмму передаются заголовок массива и указатели на количество элементов и массив.

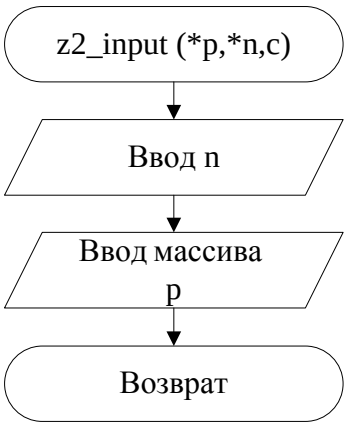


Рисунок 2.6 – Графическая схема алгоритма ввода одномерного массива

На рисунке 2.7 изображена схема вывода одномерного массива. В подпрограмму передаются размерность массива, массив и его заголовок.

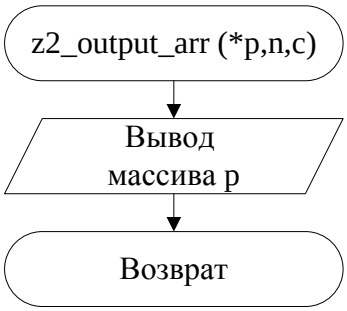
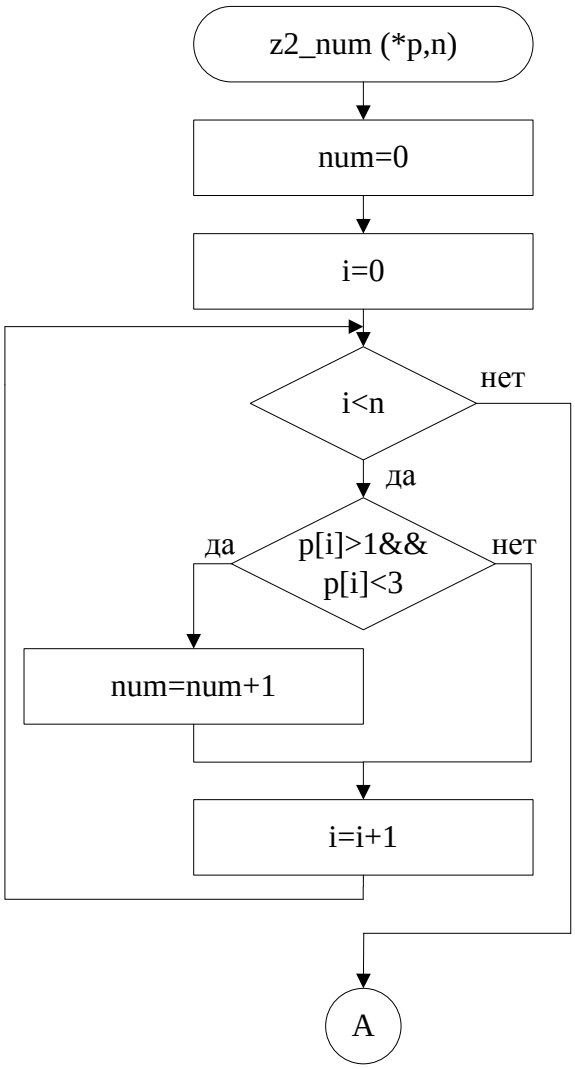


Рисунок 2.7 – Графическая схема алгоритма вывода одномерного массива

На рисунке 2.8 изображена схема алгоритма нахождения количества элементов, расположенных на нечетных местах, попадающих в интервале (1;3). В подпрограмму передаются количество элементов и указатель на массив. Подпрограмма возвращает количество элементов.



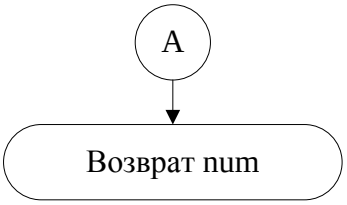


Рисунок 2.8 – Графическая схема алгоритма нахождения количества элементов, попадающих в интервале (1;3), среди элементов, расположенных на нечетных местах.

На рисунке 2.9 изображена схема алгоритма вычисления значения выражения (2.1):

$$L=tg^3\ c-\ 0{,}58\ cos\ d^2\ +\ f, \qquad (2.1)$$

где c,d и f – количества значений, попадающих в интервале (1;3), стоящих на нечётных местах массивов X, P и Q, - соответственно. В подпрограмму передаются c,d и f. Подпрограмма возвращает значение выражения.

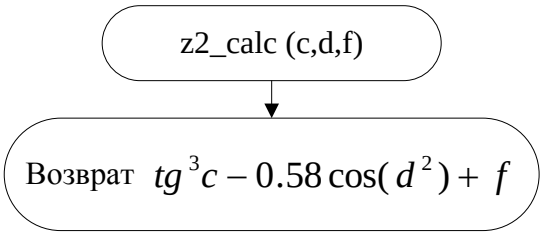


Рисунок 2.9 – Графическая схема алгоритма вычисления значения выражения (2.1)

На рисунке 2.10 изображена схема алгоритма вывода результатов вычислений. В подпрограмму передаются количества значений, попадающих в интервале (1;3), стоящих на нечётных местах массивов X, P и Q, - соответственно, и результат вычисления выражения (2.1).

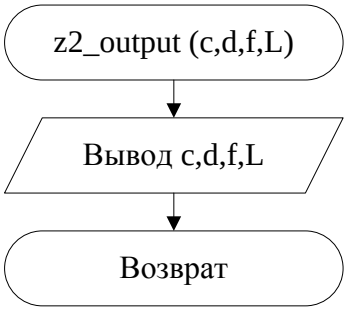


Рисунок 2.10 – Графическая схема алгоритма вывода результатов вычислений

Задача 3

На рисунке 2.11 изображена схема основного алгоритма задачи №3.

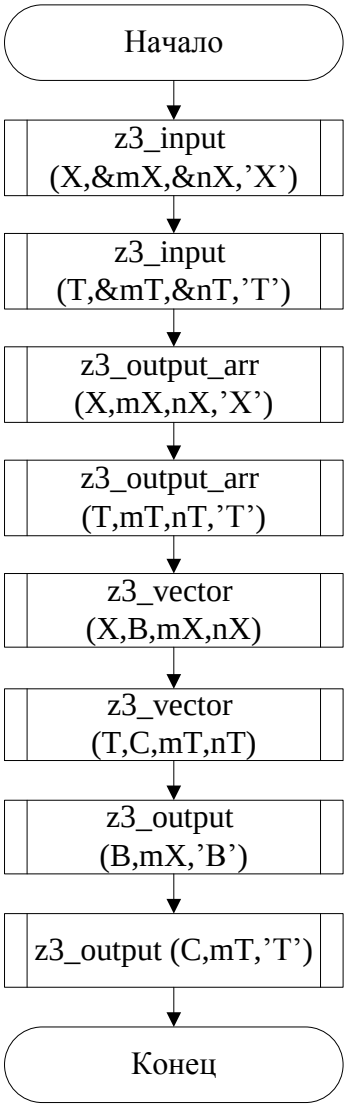
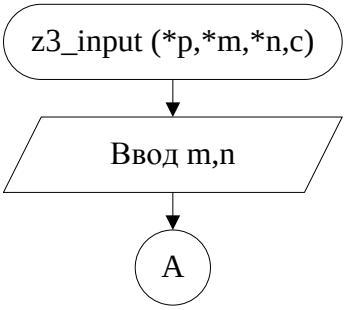


Рисунок 2.11 – Графическая схема основного алгоритма задачи №3

На рисунке 2.12 изображена схема алгоритма ввода матрицы. В подпрограмму передаются указатели на матрицу и количества строк и столбцов, а также заголовок матрицы.



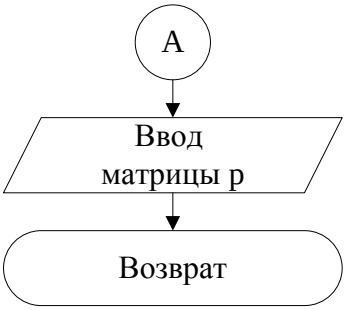


Рисунок 2.12 – Графическая схема алгоритма ввода матрицы

На рисунке 2.13 изображена схема алгоритма вывода матрицы. В подпрограмму передаются количество строк и столбцов матрицы, заголовок матрицы и указатель на нее.

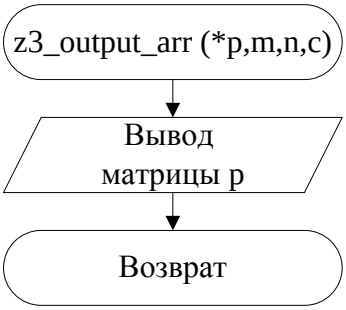


Рисунок 2.13 – Графическая схема алгоритма вывода матрицы

На рисунке 2.14 изображена схема алгоритма формирования нового вектора из минимальных элементов строк матрицы. В подпрограмму входят количества строк и столбцов матрицы, указатели на массив и вектор.

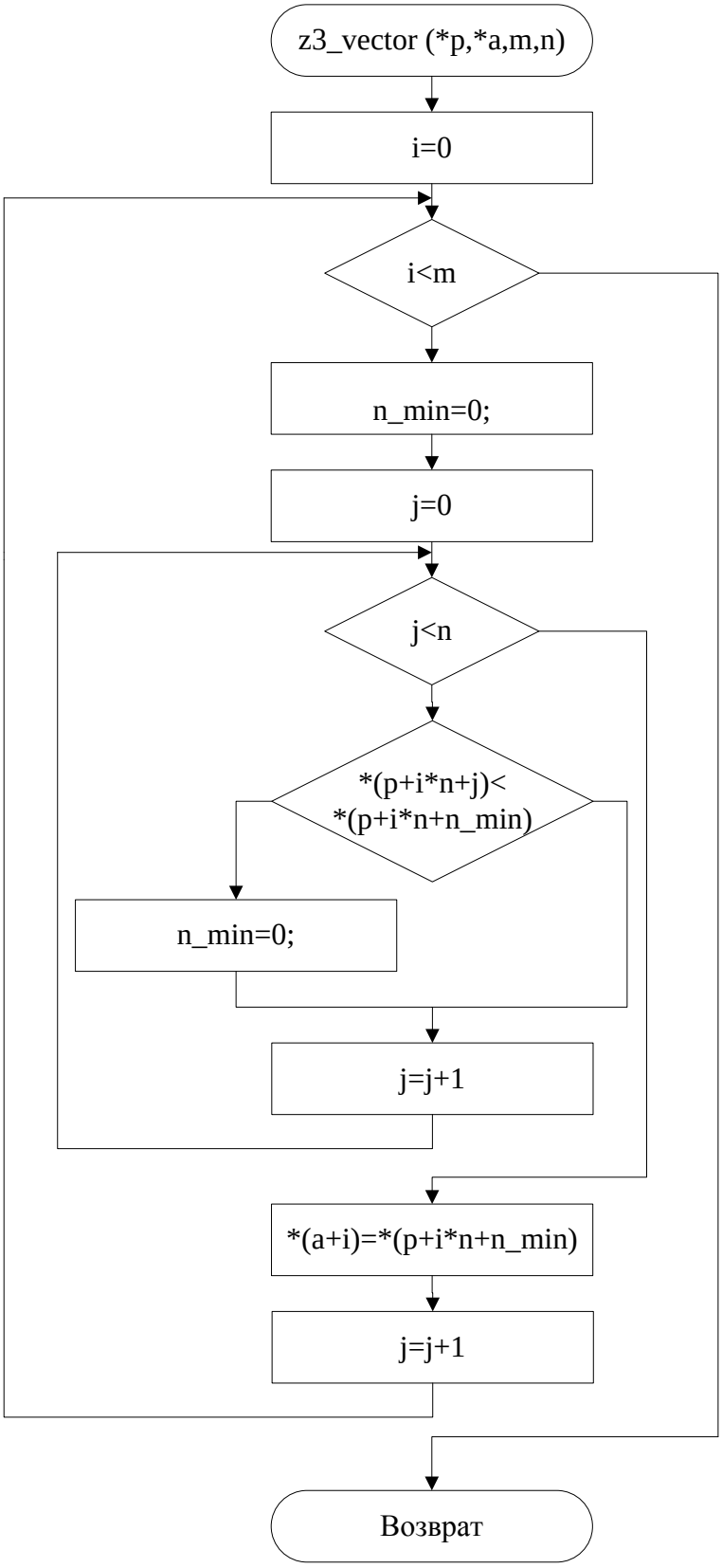


Рисунок 2.14 – Графическая схема алгоритма формирования нового вектора из минимальных элементов строк матрицы

На рисунке 2.15 изображена графическая схема алгоритма вывода сформированного вектора. В подпрограмму передаются количество элементов массива, его заголовок и указатель на массив.

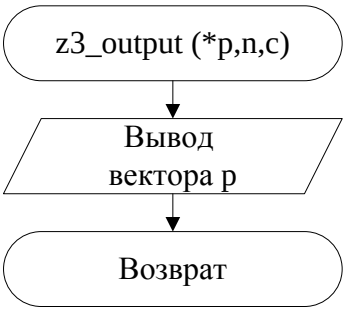


Рисунок 2.15 – Графическая схема алгоритма вывода сформированного вектора

Задача 4

На рисунке 2.16 изображена схема основного алгоритма задачи №4.

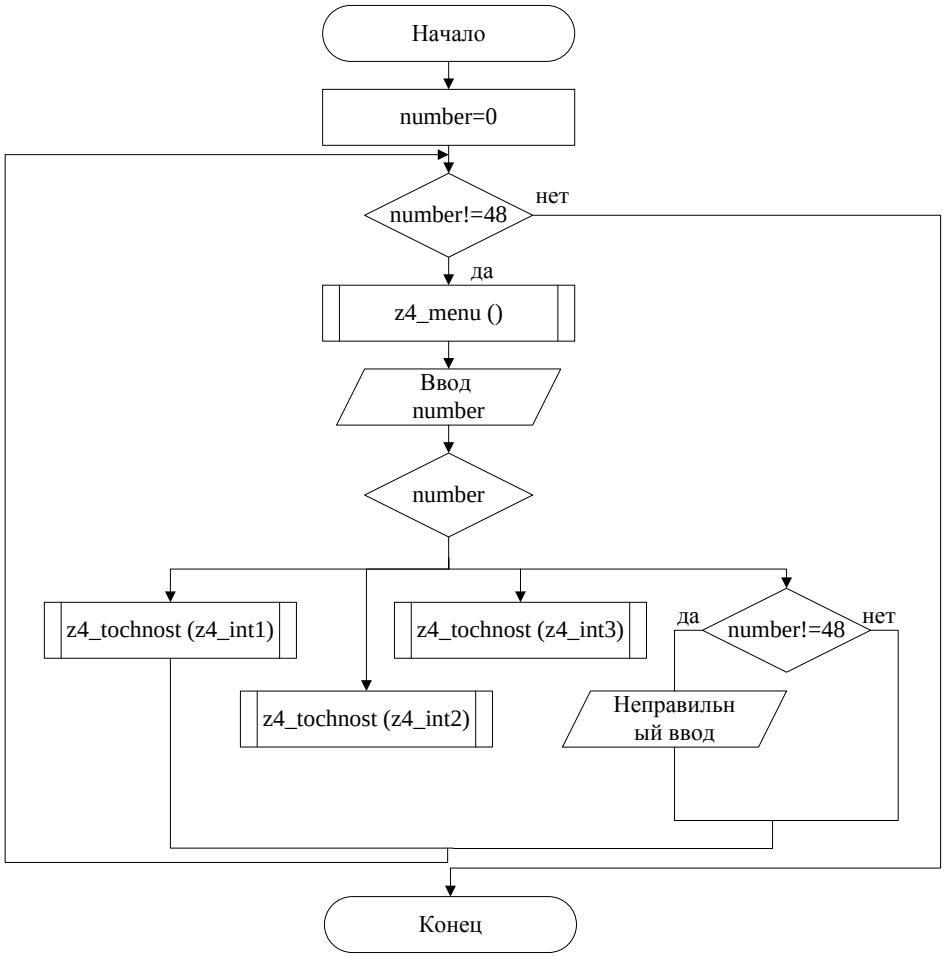


Рисунок 2.16 – Графическая схема основного алгоритма задачи №4

На рисунке 2.17 изображена схема алгоритма вычисления значения интеграла методом трапеций с заданной точностью. В подпрограмму входит указатель на подынтегральную функцию.

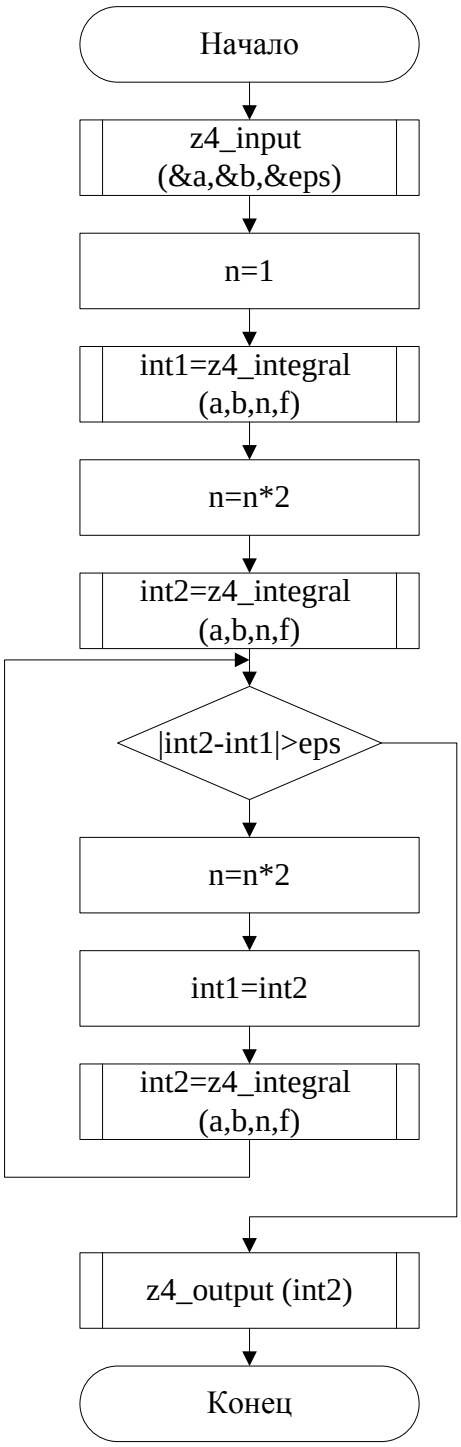


Рисунок 2.17 – Графическая схема алгоритма вычисления значения интеграла методом трапеций с заданной точностью

На рисунке 2.18 изображена схема алгоритма ввода пределов интегрирования и точности. В подпрограмму передаются указатели на точность и пределы интегрирования.

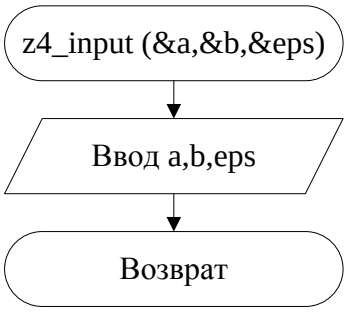


Рисунок 2.18 – Графическая схема алгоритма ввода пределов интегрирования и точности

На рисунке 2.19 изображена схема алгоритма вычисления значения подынтегральной функции №1 (2.2):

$$f1(x) = \frac{sin^4 x}{\cos x} \tag{2.2}$$

В подпрограмму передается координата x в определенной точке. Подпрограмма возвращает значение подынтегральной функции в этой точке.

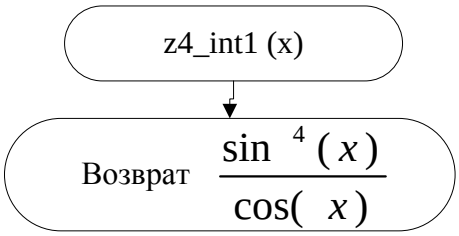


Рисунок 2.19 – Графическая схема алгоритма вычисления значения подынтегральной функции №1 (2.2)

На рисунке 2.20 изображена схема алгоритма вычисления значения подынтегральной функции №2 (2.3):

$$f2(x) = 3^{0,1x} + \sin(0,2x) \tag{2.3}$$

В подпрограмму передается координата x в определенной точке. Подпрограмма возвращает значение подынтегральной функции в этой точке.

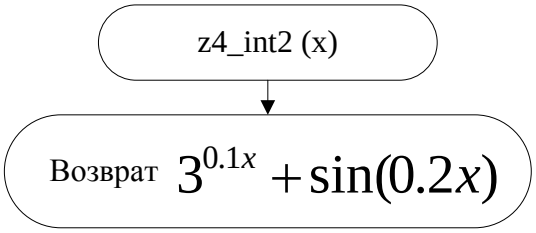


Рисунок 2.20 - Графическая схема алгоритма вычисления значения подынтегральной функции №2 (2.3)

На рисунке 2.21 изображена схема алгоритма вычисления значения подынтегральной функции №3 (2.4):

$$f3(x) = (x - 2{,}2)(x - 1{,}1)(x + 6{,}1) \quad (2.4)$$

В подпрограмму передается координата x в определенной точке. Подпрограмма возвращает значение подынтегральной функции в этой точке.

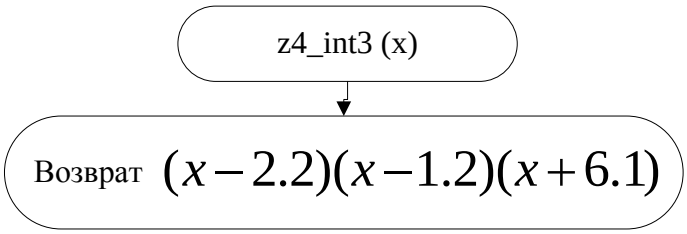
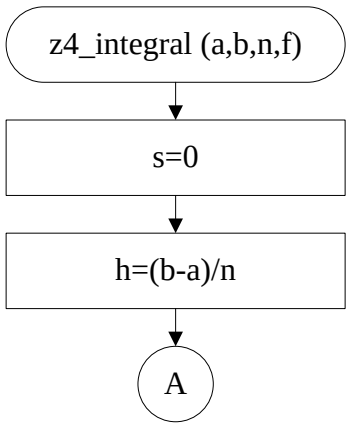


Рисунок 2.21 - Графическая схема алгоритма вычисления значения подынтегральной функции №3 (2.4)

На рисунке 2.22 изображена схема алгоритма вычисления значения интеграла методом трапеций при заданном количестве разбиений. В подпрограмму передаются пределы интегрирования, количество трапеций и указатель на функцию. Подпрограмма возвращает значение интеграла.



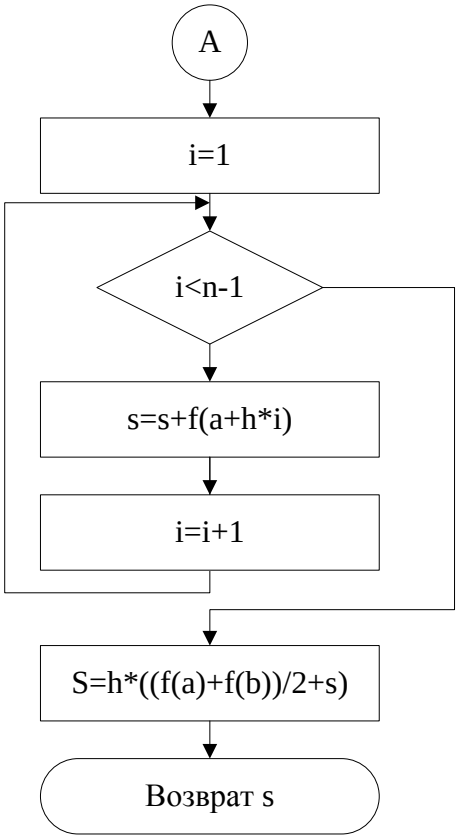


Рисунок 2.22 – Графическая схема алгоритма вычисления значения интеграла методом трапеций с заданным количеством разбиений

На рисунке 2.23 изображена схема алгоритма вывода значения интеграла. В подпрограмму передается значение интеграла.

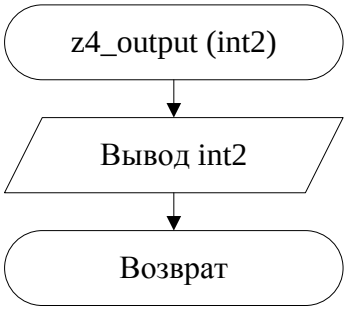


Рисунок 2.23 – Графическая схема алгоритма вывода значения интеграла

Основная задача

На рисунке 2.24 изображена схема основного алгоритма программы.

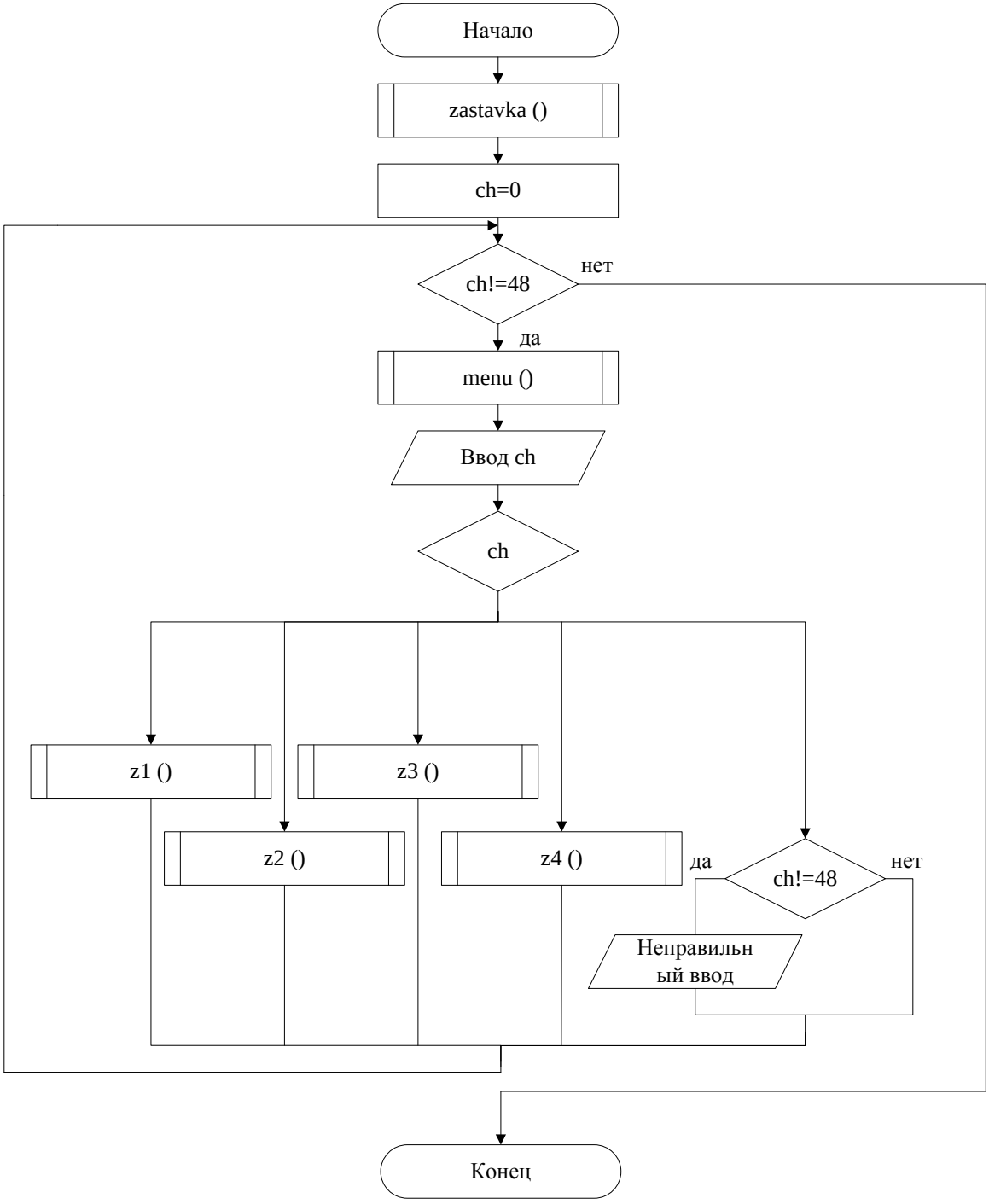


Рисунок 2.24 – Графическая схема основного алгоритма программы

На рисунке 2.25 изображена схема алгоритма вывода заставки.

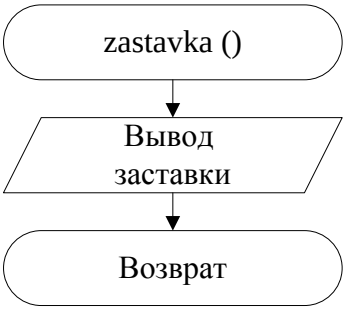


Рисунок 2.25 – Графическая схема алгоритма вывода заставки

На рисунке 2.26 изображена схема алгоритма вывода меню.

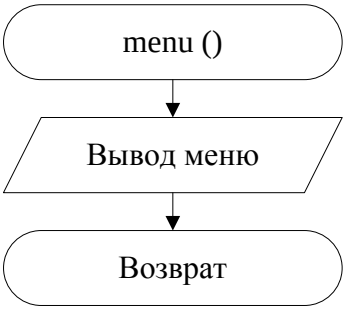


Рисунок 2.26 – Графическая схема алгоритма вывода меню

3 ОПИСАНИЕ РАЗРАБОТАННЫХ ПРИЛОЖЕНИЙ

3.1 Описание структуры программ

Задача 1

Программа состоит из главной функции (main) и 3-х вспомогательных (z1_input, z1_calc и z1_output). Подпрограмма z1_input предназначена для ввода координат точек. Значение не возвращает. Подпрограмма z1_calc вычисляет расстояние между точками и возвращает это значение в основную программу посредством оператора return. Подпрограмма z1_output выводит на экран исходные данные и расстояние. Значение не возвращает. Листинг программы приведён в Приложение А. Листинг и результаты выполнения программы №1

Переменные, использованные в программе №1 представлены в Таблице 1.

Таблица 1 – Таблица соответствия переменных задачи 1

Переменные в задаче	Переменные в программе	Тип	Комментарии
x1	x1	float	координата x 1-й точки
y1	y1	float	координата y 1-й точки
$x2$	$x2$	float	координата x 2-й точки
y2	y2	float	координата y 2-й точки
l	l	float	расстояние между точками

Задача 2

Программа состоит из главной функции (main) и 5-и вспомогательных (z2_input, z2_output_arr, z2_num, z2_calc и z2_output). Подпрограмма z2_input предназначена для ввода одномерного массива. Значение не возвращает. Подпрограмма z2_output_arr также не возвращает значение и предназначена для вывода одномерного массива на экран. В подпрограмме z2_num вычисляется количество элементов массива, попадающих в интервале (1;3), стоящих на нечетных местах. Результат вычисления передаётся в основную программу по оператору return. Подпрограмма z2_calc вычисляет значение функции (3.1):

$$L = tg^3 c - 0{,}58 \cos d^2 + f, \quad (3.1)$$

Результат вычисления передаётся в основную программу по оператору return. Подпрограмма z2_output выводит на экран значение функции (3.1). Не возвращает знчение.

Листинг программы приведён в Приложении Б. Листинг и результаты выполнения программы №2

Переменные, использованные в программе №2 представлены в Таблице 2.

Таблица 2 – Таблица соответствия переменных задачи 2

Переменные в задаче	Переменные в программе	Тип	Комментарии
X	X	float	Одномерный массив данных
P	P	float	Одномерный массив данных
Q	Q	float	Одномерный массив данных
-	nX	int	Количество элементов массива
-	nP	int	Количество элементов массива
-	nQ	int	Количество элементов массива
c	c	int	Количество элементов массива X, попадающих в интервале (1;3), стоящих на нечетных местах
d	d	int	Количество элементов массива P, попадающих в интервале (1;3), стоящих на нечетных местах
c	f	int	Количество элементов массива Q, попадающих в интервале (1;3), стоящих на нечетных местах
L	L	float	Функция (3.1)

Задача 3

Программа состоит из главной функции (main) и 4-х вспомогательных (z3_input, z3_output_arr, z3_vector и z3_output). Подпрограмма z3_input предназначена для ввода двумерного массива. Не возвращает значения. Подпрограмма z3_output_arr не возвращает значение и предназначена для вывода двумерного массива на экран. В подпрограмме z3_vector формируется одномерный массив из минимальных элементов строк матрицы. Значение не возвращает. Подпрограмма z3_output предназначена для вывода одномерного массива и значения не возврщет. Листинг программы приведён в Приложении В.

Переменные, использованные в программе №3 представлены в Таблице 3.

Таблица 3 – Таблица соответствия переменных задачи 3

Переменные в задаче	Переменные в программе	Тип	Комментарии
X	X	float	Матрица
T	T	float	Матрица
B	B	float	Вектор
C	C	float	Вектор
-	mX	int	Количество строк матрицы
-	nX	int	Количество столбцов матрицы
-	mT	int	Количество строкn матрицы
-	nT	int	Количество столбцов матрицы

Задача 4

Программа состоит из главной функции (main) и 8-и вспомогательных (z4_menu, z4_input, z4_int1, z4_int2, z4_int3, z4_integral, z4_output, z4_tochnost).Подпрограмма z4_menu не возвращает значения и выводит меню выбор интеграла на экран. Основной задачей подпрограммы z4_tochnost является достижение определенной точности вычисления. Значения он не возвращает и в процессе выполнения вызывает подпрограммы z4_input, z4_integral, z4_output. Подпрограмма z4_input предназначена для ввода границ интегрирования и точности. Значения не возвращает. В подпрограммах z4_int1, z4_int2, z4_int3 вычисляется значение функции f(x). Возвращают значения при помощи return. В подпрограмме z4_integral вычисляется интеграл выбранной пользователем функции методом трапеций. Выбор функции производится в основной программе. Результат из

подпрограммы z4_integral передаётся в подпрограмму z4_output, которая в свою очередь значения не возвращает и выводит значение интеграла на экран. Листинг программы приведён в Приложении Г

Переменные, использованные в программе №4 представлены в Таблице 4.

Таблица 4 – Таблица соответствия переменных задачи 4

Переменные в задаче	Переменные в программе	Тип	Комментарии
a	a	double	Предел интегрирования
b	b	double	Предел интегрирования
ε	eps	double	Точность
-	number	int	Номер элемента меню
-	s	double	Значение вычисления

3.2 Описание результатов

Результатом выполнения программы №1 является расстояние между двумя точками. Для проверки работы программы №1 ниже приведены тесты:

Тест 1
Исходные данные:
x1=1 y1=1
x2=2 y2=2
Результат:
l=1,414

Тест 2
Исходные данные:
x1=0 y1=0
x2=3 y2=7
Результат:
l=7.616

В программе № 2 результатом является значение выражения $L = = tg^3\, c - 0{,}58\, cos\, d^2 + f$. Для проверки работы программы №2 ниже приведены тесты:

Тест 1
Исходные данные:
X={1, 3, 5, 4, 2}
Y={2, 3, 4, 3, 2}
T={2, 2, 2, 2, 2}
Результат:
L=7.157

Тест 2
Исходные данные:
X={2, 0, -4, 9, 11}
Y={4, -23, 43, 8, 0}
T={1, 6, 2, 4, 4}
Результат:
L=4.198

В программе № 3 находятся минимальные значения строк матриц X и T. Результатом выполнения являются векторы B и C, сформированные из этих элементов. Для проверки работы программы №3 ниже приведены тесты:

Тест 1
Исходные данные:

Матрица X:	Матрица T:
1 2 3 4	2 3 4 5
5 6 7 8	1 -3 0 4
9 0 2 8	-4 3 2 1

Результат:

Массив B:	Массив C:
1 5 0	2 -3 -4

Результатом выполнения программы №4 является значение интеграла функции. Для проверки работы программы №4 ниже приведены тесты:

Тест 1
Исходные данные:
number=1, A=0, B=1, e=0.001
Результат:
S=0.18521
Тест 2
Исходные данные:
number =2, A=2, B=4, e=0.001
Результат:
S=3.907593
Тест 3
Исходные данные:
number =3, A=1, B=2, e=0.001
Результат:
S=-0.996366

ЗАКЛЮЧЕНИЕ

Я улучшил навыки алгоритмизации и программирования, изучил структуру подпрограмм, механизмы передачи параметров в подпрограмму, возврата результатов из подпрограммы и вызова подпрограммы на выполнение. Разработал алгоритмы решения четырех задач с использованием вспомогательных алгоритмов, написал комментированные программы на языке С. Оформил в виде блок-схем алгоритмы основной и вспомогательных программ. Подготовил тесты. Создал библиотеку на базе разработанных программ. Выполненная задача полностью соответствует заданию на курсовую работу.

Литература

1 Юлин В. А., Булатова И. Р. Приглашение к Си. – Мн.: Выш. шк., 1990

2 Джехани Н. Программирование на языке Си: Пер. с англ. – М.: Радио и связь, 1998

3 Котлинская Г. П., Галиновский О. И. Программирование на языке Си: Справ. пособие. – Мн.: Выш. шк., 1991

4 Шипачев В. С. Высшая математика. Учеб. для вузов/В. С. Шипачев. – 5-е изд., стер. – М.: Высш. шк. 2002

ПРИЛОЖЕНИЕ А

Текст программы №1

```
// подключение библиотек
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <math.h>
#include <windows.h>
char bufRus [256];
char *Rus (const char*text) // вывод русских символов
{
   CharToOem (text,bufRus);
   return bufRus;
}
void z1_input (float *x1,float *y1,float *x2,float *y2); // ввод координат точек A и B
float z1_calc (float x1,float y1,float x2,float y2); // вычисление расстояния между точками
void z1_output (float x1,float y1,float x2,float y2,float l); // вывод исходных данных и результата
main ()
{
   float x1,y1, // координаты точки A
      x2,y2, // координаты точки B
      l; // расстояние между точками
   system ("cls"); // очистка экрана
   puts (Rus("\n\t\t\t\t    Задача №1"));
   puts (Rus("\n\t\tНайти расстояние между точками A(x1;y1) и B(x2;y2)"));
   z1_input (&x1,&y1,&x2,&y2); // ввод координат точек
   l=z1_calc (x1,y1,x2,y2); // рассчет расстояния между точками
   z1_output (x1,y1,x2,y2,l); // вывод исходных данных и результата
   getch ();
}
void z1_input (float *x1,float *y1,float *x2,float *y2) // ввод координат точек A и B
{
   puts (Rus("\n\t\t\t     Введите координаты точки A"));
   printf ("\n\t\t\t\t   ");
   fflush (stdin); // очистка буфера клавиатуры
   scanf ("%f%f",x1,y1); // ввод координат точки A
   puts (Rus("\n\t\t\t     Введите координаты точки B"));
   printf ("\n\t\t\t\t   ");
   fflush (stdin); // очистка буфера клавиатуры
   scanf ("%f%f",x2,y2); // ввод координат точки B
}
float z1_calc (float x1,float y1,float x2,float y2) // вычисление расстояния между точками
{
   return (pow((pow((x2-x1),2)+pow((y2-y1),2)),0.5)); // возврат расстояния
}
void z1_output (float x1,float y1,float x2,float y2,float l) // вывод исходных данных и результата
{
```

```
   printf (Rus("\n Расстояние между точками A(%.3f;%.3f) и B(%.3f;%.3f) равно
%.3f"),x1,y1,x2,y2,l);
}
```

ПРИЛОЖЕНИЕ Б

Текст программы №2

```
// подключение библиотек
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <math.h>
#include <windows.h>
char bufRus [256];
char *Rus (const char*text) // вывод русских символов
{
   CharToOem (text,bufRus);
   return bufRus;
}
void z2_input (float *p,int *n,char c); // ввод одномерного массива
void z2_output_arr (float *p,int n,char c); // вывод одномерного мссива
int z2_num (float *p,int n); // вычисление количества эл-ов на нечетных местах,
попадающих в интервале (1;3)
float z2_calc (int c,int d,int f); // вычисление значения выражения
void z2_output (int c,int d,int f,float L); // вывод результатов
main ()
{
   float X[100],P[100],Q[100], // одномерные массивы X, P и Q
      L; // результт вычисления выражения
   int nX,nP,nQ, // кол-ва эл-ов в массивах X, P и Q соответственно
      c,d,f,  // кол-ва эл-ов на нечетных местах, попадающих в интервале (1;3) в массивах X,
P и Q соответственно
      i;
   system ("cls"); // очистка экрана
   puts (Rus("\n\t\t\t\t    Задача №2"));
   puts (Rus("\n\t     Вычислить значение функции L=tg^3(c)-0.58*cos(d^2)+f,"));
   puts (Rus("\tгде c,d и f - количество значений попадающих в интервале (1;3)"));
   puts (Rus("\tсреди элементов, стоящих на четных местах в массивах X, P и Q"));
   puts (Rus("\t\t\t\tсоответственно"));
   z2_input (X,&nX,'X'); // ввод массива X
   z2_input (P,&nP,'P'); // ввод массива P
   z2_input (Q,&nQ,'Q'); // ввод массива Q
   z2_output_arr (X,nX,'X'); // вывод массива X
   z2_output_arr (P,nP,'P'); // вывод массива P
   z2_output_arr (Q,nQ,'Q'); // вывод массива Q
```

```
   c=z2_num (X,nX);  // определение количества элементов на нечетных местах
попадающих в интервале (1;3) в массиве X
   d=z2_num (P,nP);  // определение количества элементов на нечетных местах
попадающих в интервале (1;3) в массиве P
   f=z2_num (Q,nQ);  // определение количества элементов на нечетных местах
попадающих в интервале (1;3) в массиве Q
   L=z2_calc (c,d,f); // вычисление значения выражения
   z2_output (c,d,f,L); // вывод результатов
   getch ();
}
void z2_input (float *p,int *n,char c) // ввод одномерного массива
{
   int i;
   printf (Rus("\n\t\t   Введите количество элементов массива %c\n"),c);
   printf ("\t\t\t\t\t");
   scanf ("%d",n); // ввод количества эл-ов
   printf (Rus("\n\t\t\t\tВведите массив %c\n"),c);
   printf ("\t\t\t\t");
   for (i=0;i<*n;i++)
      scanf ("%f",(p+i)); // ввод эл-а массива
}
void z2_output_arr (float *p,int n,char c) // вывод одномерного мссива
{
   int i;
   printf (Rus("\n\t\t\t       Введенный массив %c\n"),c);
   printf ("\t\t");
   for (i=0;i<n;i++)
      printf ("%.2f ",*(p+i)); // вывод эл-а массива
   printf ("\n");
}
int z2_num (float *p,int n) // вычисление количества эл-ов на нечетных местах, попадающих
в интервале (1;3)
{
   int i,
      num; // количество эл-ов на нечетных местах, попадающих в интервале (1;3)
   num=0;
   for (i=0;i<n;i=i+2) // цикл просмотра эл-ов на нечетных местах
      if (*(p+i)>1&&*(p+i)<3) // условие принадлежности к интервалу
         num=num+1;
   return num; // возврат количества
}
float z2_calc (int c,int d,int f) // вычисление значения выражения
```

```
{
   return (pow(tan(c),3)-0.58*cos(d*d)+f); // возврт значения выражения
}
void z2_output (int c,int d,int f,float L) // вывод результатов
{
   printf ("\n\t\t\t\tc=%d   d=%d   f=%d\n",c,d,f); // вывод количеств нужных эл-ов в массивах
X, P и Q соответственно
   printf (Rus("\n\t\t\tРезультат выражения равен %.3f\n"),L); // вывод значения выражения
}
```

ПРИЛОЖЕНИЕ В

Текст программы №3

```
// подключение библиотек
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <math.h>
#include <windows.h>
char bufRus [256];
char *Rus (const char*text) // вывод русских символов
{
   CharToOem (text,bufRus);
   return bufRus;
}
void z3_input (float *p,int *m,int *n,char c); // ввод двумерного массива
void z3_output_arr (float *p,int m,int n,char c); // вывод двумерного массива
void z3_vector (float *p,float *a,int m,int n); // формирование вектора из
минимальных эл-ов строк
void z3_output (float *p,int m,char c); // вывод сформированного вектора
main ()
{
   float X[400],T[400], // массивы X и T
      B[400],C[400]; // векторы B и C
   int mX,nX, // кол-ва строк и столбцов в матрице X
      mT,nT, // кол-ва строк и столбцов в матрице T
      i;
   system ("cls"); // очистка экрана
   puts (Rus("\n\t\t\t\t    Задача №3"));
   puts (Rus("\n\t     Сформировать векторы B и C, элементами которых являются"));
   puts (Rus("\t     минимальные значения строк матриц X и T соответственно"));
   z3_input (X,&mX,&nX,'X'); // ввод матрицы X
   z3_input (T,&mT,&nT,'T'); // ввод матрицы T
   z3_output_arr (X,mX,nX,'X'); // вывод матрицы X
   z3_output_arr (T,mT,nT,'T'); // вывод матрицы T
   z3_vector (X,B,mX,nX); // формирование вектора B
   z3_vector (T,C,mT,nT); // формирование вектора C
   z3_output (B,mX,'B'); // вывод вектора B
   z3_output (C,mT,'C'); // вывод вектора C
   getch ();
}
void z3_input (float *p,int *m,int *n,char c) // ввод двумерного массива через
одномерный
{
   int i,j;
   printf (Rus("\n\t\tВведите количество строк и столбцов в матрице %c\n"),c);
   printf ("\t\t\t\t      ");
   scanf ("%d%d",m,n); // ввод кол-в строк и столбцов
   for (i=0;i<*m;i++)
```

```
            for (j=0;j<*n;j++)
            {
               printf ("%\t\t\t\t   %c[%d][%d]=",c,i,j);
               scanf ("%f",(p+i**n+j)); // ввод эл-а массива
            }
         }
         void z3_output_arr (float *p,int m,int n,char c) // вывод одномерного массива в виде
двумерного
         {
            int i,j;
            printf (Rus("\n\t\t\t    Введенная матрица %c\n"),c);
            for (i=0;i<m;i++)
            {
               printf ("\t\t\t");
               for (j=0;j<n;j++)
                  printf ("%.2f ",*(p+i*n+j)); // вывод эл-а мссива
               printf ("\n");
            }
         }
         void z3_vector (float *p,float *a,int m,int n) // формирование вектора из минимальных
элементов строк
         {
            int i,j,
               n_min; // номер минимального элемента строки
            for (i=0;i<m;i++)
            {
               n_min=0;
               for (j=1;j<n;j++)
                  if (*(p+i*n+j)<*(p+i*n+n_min)) // условие минимальности
                     n_min=j;
               *(a+i)=*(p+i*n+n_min); // запись значения минимального эл-а в вектор
            }
         }
         void z3_output (float *p,int m,char c) // вывод сформированного вектора
         {
            int i;
            printf (Rus("\n\t\t\t    Сформированный вектор %c\n"),c);
            printf ("\t\t\t");
            for (i=0;i<m;i++)
               printf ("%.2f ",*(p+i)); // вывод эл-а вектора
            printf ("\n");
         }
```

ПРИЛОЖЕНИЕ Г

Текст программы №4

```
// подключение библиотек
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <math.h>
#include <windows.h>
char bufRus [256];
char *Rus (const char*text) // вывод русских символов
{
   CharToOem (text,bufRus);
   return bufRus;
}
typedef double (*func) (double); // указатель на подинтегральную функцию
void z4_menu (); // вывод меню выбора интегралов
void z4_input (double *a,double *b,double *eps); // ввод границ интегрирования и точности
double z4_int1 (double x); // подинтегральная функция №1
double z4_int2 (double x); // подинтегральная функция №2
double z4_int3 (double x); // подинтегральная функция №3
double z4_integral (double a,double b,int n,func f); // вычисление значения интеграла
void z4_output (double int2); // вывод значения интеграла
void z4_tochnost (func f); // вычисление значения интеграла с введенной точностью
main ()
{
   int number; // переменная для выбора эл-а меню
   system ("cls"); // очистка экрана
   number=0;
   while (number!=48)
   {
      z4_menu (); // вывод меню выбора интеграла
      number=getch (); // ввод номера эл-а меню
      switch (number)
      {
         case 49: z4_tochnost (z4_int1); // вычисление значения интеграла №1
         break;
         case 50: z4_tochnost (z4_int2); // вычисление значения интеграла №2
         break;
         case 51: z4_tochnost (z4_int3); // вычисление значения интеграла №3
         break;
         default:
            if (number!=48) // вывод сообщения при неверном вводе
            {
               puts (Rus("\n\t\t\t\t Неверный ввод"));
               getch ();
            }
      }
   }
}
```

```
void z4_menu () // меню выбора интегралов
{
   system ("cls");
   puts (Rus("\n\n\n\t\t\t\t *** Меню ***\n\n"));
   puts (Rus("\t\t\t    1 - рассчет интеграла №1\n"));
   puts (Rus("\t\t\t    2 - рассчет интеграла №2\n"));
   puts (Rus("\t\t\t    3 - рассчет интеграла №3\n"));
   puts (Rus("\t\t\t    0 - выход из программы"));
   puts (Rus("\n\n\n\t\t\t   Введите номер пункта меню"));
   printf ("\n\t\t\t\t     ");
}
void z4_input (double *a,double *b,double *eps) // ввод пределов интегрирования и точности
{
   system ("cls");
   puts (Rus("\n\t\tВведите верхний и нижний пределы интегрирования"));
   printf ("\n\t\t\t\t   ");
   scanf ("%lf%lf",a,b); // ввод пределов интегрирования
   puts (Rus("\n\t\t\t     Введите точность"));
   printf ("\n\t\t\t\t   ");
   scanf ("%lf",eps); // ввод точности
}
double z4_int1 (double x) // подинтегральная функция №1
{
   return (pow(sin(x),4)/cos(x)); // возврат значения подинтегральной функции
}
double z4_int2 (double x) // подинтегральная функция №2
{
   return (pow(3,0.1*x)+sin(0.2*x)); // возврат значения подинтегральной функции
}
double z4_int3 (double x) // подинтегральная функция №3
{
   return ((x-2.2)*(x-1.2)*(x+6.1)); // возврат значения подинтегральной функции
}
double z4_integral (double a,double b,int n,func f) // вычисление значения интеграла методом
трапеций
{
   double h,s;
   int i;
   s=0;
   h=(b-a)/n;
   for (i=1;i<n-1;i++)
      s=s+f(a+h*i);
   s=h*((f(a)+f(b))/2+s);
   return s; // возврат значения интеграла
}
void z4_output (double int2) // вывод значения интеграла
{
   printf (Rus("\n\t\t\tИнтеграл равен %lf"),int2); // вывод значения
}
void z4_tochnost (func f) // вычисление значения интеграла с введенной точностью
{
   int n;
```

```
    double a,b, // пределы интегрирования
       eps, // точность
       int1,int2; // значения интеграла
    z4_input (&a,&b,&eps); // ввод пределов интегрирования и точности
    n=1; // начльное кол-во трапеций
    int1=z4_integral (a,b,n,f); // вычисление значения интеграла
    n=n*2; // удвоение кол-ва трапеций
    int2=z4_integral (a,b,n,f); // вычисление значения интеграла
    while (fabs(int2-int1)>eps) // пока модуль разности больше точности
    {
       n=n*2; // удвоение кол-ва трапеций
       int1=int2;
       int2=z4_integral (a,b,n,f); // вычисление значения интегрла
    }
    z4_output (int2); // вывод значения интеграла
    getch ();
}
```

ПРИЛОЖЕНИЕ Д

Текст основной программы

```
// подключение библиотек
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <math.h>
#include <windows.h>
char bufRus [256];
char *Rus (const char*text) // вывод русских символов
{
   CharToOem (text,bufRus);
   return bufRus;
}
void zastavka (); // вывод заставки
void menu (); // вывод меню выбора задачи
main ()
{
   int ch; // переменная для выбора эл-а меню
   system ("color 1e"); // установка цвета фона и текста
   zastavka (); // вывод заставки
   ch=0;
   while (ch!=48)
   {
      menu (); // вывод меню выбора здачи
      ch=getch (); // считывание номера эл-а меню
      switch (ch)
      {
         case 49: z1 (); // решение задачи №1
         break;
         case 50: z2 (); // решение задачи №2
         break;
         case 51: z3 (); // решение задачи №3
         break;
         case 52: z4 (); // решение задачи №4
         break;
         default: // вывод сообщения при неверном вводе
            if (ch!=48)
            {
               puts (Rus("\n\t\t\t\tНеверный ввод"));
               getch ();
            }
      }
   }
   return 0;
}
void zastavka () // вывод заставки
{
   puts (Rus("\n\t\t\t\tКурсовая работа"));
```

```
    puts (Rus("\t\t\t\t    по ОАиП"));
    puts (Rus("\t\t\t     студента группы ИТ-11"));
    puts (Rus("\t\t\t\tШестюка Василия"));
    puts (Rus("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t   Гомель 2011"));
    puts (Rus("\n\t\t      Нажмите любую клавишу для продолжения"));
    getch ();
}
void menu () // вывод меню выбора задачи
{
    system ("cls");
    puts (Rus("\n\n\n\t\t\t\t *** Меню ***"));
    puts (Rus("\n\t\t\t    1 - решение задачи №1"));
    puts (Rus("\n\t\t\t    2 - решение задачи №2"));
    puts (Rus("\n\t\t\t    3 - решение задачи №3"));
    puts (Rus("\n\t\t\t    4 - решение задачи №4"));
    puts (Rus("\n\t\t\t    0 - выход из программы"));
    puts (Rus("\n\n\n\t\t\t  Введите номер пункта меню"));
    printf ("\n\t\t\t\t     ");
}
```