МИНИСТЕРСТВО ПУТЕЙ СООБЩЕНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ (МИИТ)

Кафедра вагонов и вагонного хозяйства

С.В. Беспалько, Г.Ф. Чугунов

Утверждено
редакционно-издательским
советом университета

РЕШЕНИЕ ЗАДАЧИ НА ЭВМ С ПРОГРАММИРОВАНИЕМ НА ЯЗЫКЕ СИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к курсовой работе
по дисциплине "Информатика"

для студентов специальности "Вагоны"

Москва - 2002

УДК 629.45/.46:681.3
Б - 53
Беспалько С.В., Чугунов Г.Ф. Решение задачи на ЭВМ с программированием на языке Си: Методические указания к курсовой работе по дисциплине "Информатика". - М.: МИИТ, 2002. - 22 с.

Методические указания предназначены для закрепления теоретических знаний при освоении программирования на языке Си.

Каждая из двух предлагаемых работ представляет собой полное решение задачи на ЭВМ и включает описание последовательности решения, пример и варианты заданий по данной работе.

Введение

При организации курсовой работы по дисциплине "Информатика" для студентов 1-го курса необходимо учитывать их теоретическую подготовку на момент выполнения работы. Предлагаемые темы ориентирована на уровень подготовки по математике в рамках программы вступительных экзаменов в технические вузы и на логическое мышление. Решение этой задачи закрепляет знания по следующим приемам программирования (перечисляются в порядке возрастания сложности):

- типы данных;
- операции: арифметические, присваивания, преобразования типа;
- использование стандартных математических функций;
- массивы;
- реализация разветвленных алгоритмов;
- организация циклов;
- разработка функций.

Кроме того, курсовая работа по "Информатике" способствует развитию у студентов логического мышления, навыков самостоятельной работы, умению применять знания по программированию для решения практических задач.

Настоящие методические указания содержат две темы курсовой работы, соответствующие двум уровням сложности: «Вычисление определенного интеграла заданной функции» (обычный уровень) и «Решение уравнения методом половинного деления» (повышенный уровень сложности). В главах 1 и 2 приводятся теоретические основы и последовательность выполнения соответствующей работы, списки вариантов заданий с ответами, а также пример решения одного варианта задания.

В главе 3 описываются правила выполнения и оформления работы.

Решение задачи на ЭВМ выполняется поэтапно. На каждом этапе программист (студент-вагонник) решает более простую задачу, на результатах которой основывается следующий этап:

- математическая формулировка;
- разработка алгоритма;
- составление программы;
- выполнение программы и анализ результатов.

Следует обратить особое внимание на то, что поэтапный подход к решению задач имеет много преимуществ, главное из которых - последовательное приближение к решению поставленной задачи и, как следствие, методически более правильное распределение затрат труда.

Кроме того, значительное время при программировании тратится на отладку программы, поиск и исправление ошибок. При поэтапном решении обычно не требуется для проверки возвращаться к начальным этапам решения. Поэтому в целом данный подход более эффективен, он позволяет быстрее добиться правильных результатов.

В качестве учебных пособий для выполнения курсовой работы можно рекомендовать следующую литературу: [1], [2], [3], [4].

1. Курсовая работа тему: «Вычисление определенного интеграла заданной функции»

Задана функция *f(x)* произвольного вида.

Требуется найти определенный интеграл этой функции: $\int_a^b f(x)dx$.

Очевидно, что функция на промежутке [a, b] должна быть непрерывна.

1.1 Математическая формулировка задачи

К решению поставленной задачи возможны различные подходы. Во-первых, это формула Ньютона-Лейбница. Во-вторых, это различные приближенные (численные) методы. Каждый подход имеет свои преимущества и недостатки.

Преимуществом применения формулы Ньютона-Лейбница является простота, так как решение можно получить при помощи одного выражения. Кроме того, формуда дает точное значение интеграла. Принципиальный недостаток этого подхода - привязка к определенному виду функции. Невозможно, основываясь на этом подходе создать универсальный алгоритм для вычисления интеграла функции произвольного вида. Наконец, не у всех функций вообще можно взять точный интеграл.

Численные методы, напротив, не "привязываются" к конкретному виду функции. Их приближенность не должна пугать, потому что практически все значения функций сами вычисляются приближенно. Поэтому обычно исследователи осознанно идут на применение приближенных методов, представляя себе погрешность каждого метода.

Для интегрирования функции *f(x)* применим метод трапеций. Заданный промежуток [a, b] разобьем на несколько малых участков длиной dx каждый, как показано на рис.1.1. На каждом участке функцию аппроксимируем прямой линией (хордой), проходящей через конечные точки данного участка, как показано на рисунке.

То есть площадь криволинейной трапеции приближенно заменим пощадью под ломаной линией.

Площадь криволинейной трапеции S_i на элементарном участке ("элементарную площадь") можно представить как площадь обычной трапеции по формуле:

$$S_i \approx \frac{f(x)+f(x+dx)}{2}dx,$$

где *f(x)*, *f(x+dx)* - основания трапеции;

$dx = \frac{b-a}{n}$ - ее высота;

n - количество участков разбиения.

Схема метода трапеций

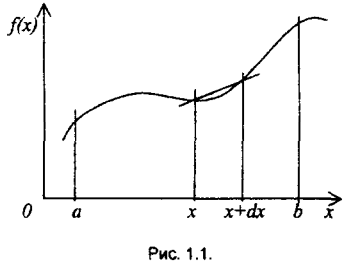

Рис. 1.1.

Если просуммировать элементарные площади S_i по всем участкам, то получим формулу для приближенного вычисления всей площади криволинейной трапеции:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} S_i = \left[\frac{f(a)}{2} + f(a+dx) + f(a+2dx) + \ldots + f(b-dx) + \frac{f(b)}{2}\right]dx$$

Таким образом, в окончательной формуле суммируются значения функции в точках сетки разбиения промежутка интегрирования, причем в граничных точках берутся половинные значения.

1.2. Разработка алгоритма

При составлении алгоритма математическая формулировка представляется в виде последовательности простых действий. Программист обдумывает общую схему решения, организацию ввода исходных данных и вывода результатов, взаимодействие отдельных этапов расчета. В результате формируется логическая схема программы, наглядно изображающая алгоритм, обычно в виде блок-схемы.

Алгоритм решения данной задачи удобно подразделить на два алгоритма:

- алгоритм главной функции (основной алгоритм), на который ложатся задачи ввода исходных данных, вызова алгоритма расчетной части и вывода результатов;
- алгоритм вычисления интеграла методом трапеций, содержащий вычислительную часть при решении задачи.

Выделение частей в обособленные алгоритмы очень полезно, так как упрощает решение общей задачи. Каждая "элементарная" задача прорабатывается отдельно, ее алгоритм, а затем и функцию, легче составлять и проверять.

1.2.1. Основной алгоритм

Блок-счема алгоритма приведена на рис. 1.2. Алгоритм является линейным, его простота обеспечена тем, что вычислительная часть вынесена в отдельный алгоритм.

Блок-схема основного алгоритма

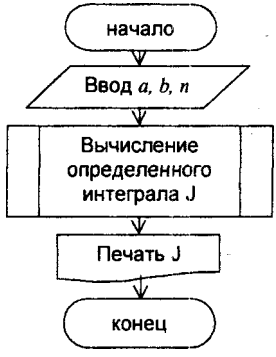

Рис. 1.2.

1.2.2. Алгоритм вычисления интеграла

При разработке алгоритма применения метода трапеций необходимо организовать суммирование значений функции в отдельных точках. Так как количество участков разбиения может быть задано любым, необходимо применить циклический алгоритм, показанный на рис.1.3.

В качестве счетчика, на первый взгляд, можно использовать переменную x. Однако следует отметить, что действительные переменные в качестве счетчика необходимо применять с осторожностью. Это связано с тем, что в памяти хранится ограниченное количество значащих цифр числа, и проверка условия выполнения цикла (например, $x<b$) иногда может привести к неправильным результатам.

В данном случае для решения этой проблемы введем дополнительный счетчик i - номер промежуточной точки в сетке разбиения оси x. Диапазон его изменения - от 1 до n-1, так как первая и последняя точки исключаются из суммирования в цикле.

Граничные точки суммируются особым образом (половинные значения функции), их удобно учесть в качестве начального значения суммы J.

Блоки, включающие вычисление значений функции, отмечены полосками.

Внутри цикла сумма J увеличивается на значение функции в текущей точке сетки, после чего осуществляется переход к следующему узлу (приращением его номера i и координаты x). После выхода из цикла сумма J будет представлять собой искомое значение интеграла.

Блок-схема вычисления интеграла методом трапеций

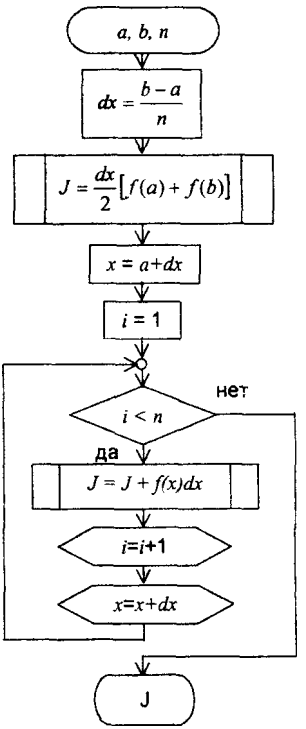

Рис. 1.3.

1.3. Составление программы

Составим программу расчета на основе разработанных алгоритмов. Для каждой части алгоритма разработаем отдельную функцию:

- для основного алгоритма - функцию main;
- для алгоритма вычисления интеграла - функцию Integral;
- для заданной подынтегральной функции - f.

Хотя для последней функции алгоритм не был приведен, при составлении алгоритма подразумевалось, что эта функция задана.

Когда разрабатывают функцию, то создают ее определение, включающее заголовок и тело функции (выполняемые в ней операторы). Кроме того, все функции, кроме main, перед их вызовом должны быть описаны. Описания функций целесообразно помещать в начало программы.

В качестве примера решим задачу определения энергоемкости поглощающего аппарата автосцепки на основе его силовой характеристики, заданной следующей функцией (вариант №30 задания):

$R = f(x) = 0.2 \cdot e^{30x}$,

где R - усилие в автосцепке, МН (мега-ньютоны);

x - ход аппарата, м, изменяющийся от 0 до 0,1.

Энергоемкость Э (в МДж) будем вычислять как интеграл функции изменения реакции на полном ходе аппарата, то есть по формуле: $Э = \int_0^{0,1} R dx$.

Количество участков сетки разбиения примем равным 100.

1.3.1. Главная функция

В этой функции реализуется алгоритм, показанный на рис. 1.2.

В начале необходимо описать все переменные, которые будут использоваться. При этом руководствуются "физическим смыслом" величин. Переменные a, b, J являются действительными величинами, для их описания применим тип double (тип - "с плавающей точкой двойной точности").

Опишем также целую переменную n, которая задает количество участков разбиения.

Ввод исходных данных организуем наиболее простым способом - при описании переменных.

```
void main() {
        /* Главная функция */
        double a=0., b=0.1, J;
        int n=100;
        J=Integral(a, b, n);
        printf("Интеграл равен %7.3lf\n", J);
}
```

Вывод результатов производится через вызов функции printf - стандартной функции вывода на экран. Для ее использования необходимо в начало программы включить описания стандартных функций ввода-вывода:

```
#include <stdio.h>
```

По этой команде препроцессор включит в исходный текст программы содержимое файла stdio.h, содержащего описания указанных функций.

1.3.2. Функция вычисления интеграла

В функции Integr реализуем алгоритм, приведенный на рис. 1.3.

Составим определение функции:

```
double Integral(double a, double b, double n) {
        /* Функция вычисления интеграла методом трапеций */
        double x, dx, J;
        int i;
        dx=(b-a)/(float)n;
        J=dx/2.*(f(a)+f(b));
        x=a+dx;
        for(i=1; i<n; i++) {
                J=J+f(x)*dx;
                x=x+dx;
        }
        return(J);
}
```

В начале тела функции описываются переменные двух типов:

- с плавающей точкой двойной точности (double) - x, dx и J;
- целая (int) - переменная i.

Кроме того, в заголовке функции описываются передаваемые ей параметры a, b, n. Вычисление dx содержит операцию преобразования типа переменной n, которая должна быть подставлена в выражение в виде double (как действительное значение).

Цикл перебора участков разбиения организуем через ключевое слово for. Тело цикла заключается в фигурные скобки. В заголовок цикла выносятся три главные операции, необходимые для организации цикла: задание начального значения счетчика (i=1), условие выполнения цикла (i<n) и операция увеличения счетчика на единицу (i++). Для понимания текста программы полезно сопоставлять его с блок-схемой алгоритма.

При выходе из цикла полученное значение интеграла J передается в качестве возвращаемого значения (оператор return). Так как тип J описан как double, этот же тип должен быть описан в качестве типа функции в ее заголовке (перед именем функции Integral).

Описание функции (помещаемое в начало программы):

```
double Integral(double a, double b, double n);
```

1.3.3. Заданная функция

Функция Integral несколько раз вызывает f(x) - подынтегральную функцию. Ее определение будет отличаться для каждого варианта задания. Составим определение функции для варианта: $f(x) = 0{,}2 \cdot e^{30x}$.

```
double f(double x) {
        /* Заданная функция */
        return(0.2*exp(30.*x));
}
```

Тело функции состоит всего лишь из одного оператора return, создающего возвращаемое значение функции. Сложность при составлении этой функции состоит в правильной записи арифметического выражения, часто включающего и математические функции. Обратите внимание на правильность обозначения констант с плавающей точкой (30.), для которых указывается десятичная точка.

Для использования стандартных математических функций в текст программы необходимо включить файл с их описаниями при помощи команды:

```
#include <math.h>
```

Кроме того, в начало программы следует поместить описание самой функции f.

```
double f(double x);
```

1.3.4. Структрура программы

Составим программу их разработанных выше частей, руководствуясь следующей последовательностью. В начале программы расположим команды

препроцессору, включающие в текст программы файлы stdio.h и math.h с описаниями стандартных функций. Затем поместим описания разработанных функций Integr и f.

Далее будут идти определения всех разработанных функций (main, Integr и f).

```
/* Программа вычисления определенного интеграла */
/* Описания стандартных функций: */
#include <stdio.h>
#include <math.h>
/* Описания разработанных функций: */
double Integral(double a, double b, double n);
double f(double x);
/* Определения разработанных функций: */
void main() {
        /* Главная функция */
        double a=0., b=0.1, J;
        int n=100;
        J=Integral(a, b, n);
        printf("Интеграл равен %7.3lf\n", J);
}
double Integral(double a, double b, double n) {
        /* Функция вычисления интеграла методом трапеций */
        double x, dx, J;
        int i;
        dx=(b-a)/(float)n;
        J=dx/2.*(f(a)+f(b));
        x=a+dx;
        for(i=1; i<n; i++) {
                J=J+f(x)*dx;
                x=x+dx;
        }
        return(J);
}
double f(double x) {
        /* Заданная функция */
        return(0.2*exp(30.*x));
}
```

1.4. Выполнение программы и анализ результатов

На этом этапе составленную программу вводят в память ЭВМ, записывают в виде исходного файла под именем, например, integral.c и выполняют обработку программы до получения результатов. Весь этот комплекс задач удобно решать в среде программирования Borland C++, как описано в главе 3.

В случае, если все этапы обработки и выполнения программы пройдут без ошибок, на экран будут выведен результат расчета. Для рассматриваемого варианта следующий:

```
Интеграл равен   0.127
```

Полученное значение можно сравнить с ответом, приведенным в задании по каждому варианту, и в случае совпадения переходить к оформлению работы.

1.5. Варианты заданий

Найти значение определенного интеграла заданной функции.

В таблице 1.1 приведены варианты заданий с ответами. Каждому варианту соответствует свой вид уравнения и промежуток.

Для всех вариантов количество участков разбиения n принять равным 50.

Таблица 1.1

Варианты заданий

№ вар.	Подынтегральная функция f(x)	Пределы интегрирования		Значения интеграла
		a	b	
1	$\sin 2x + 2 - e^x$	-1,57	1,57	1,68
2	$\cos 2x + 0,5 - x^3$	0	1	0,704
3	$10 \ln x - 2x^2 + 5$	0,1	2	1,321
4	$2^x + 1 = 2x^2$	-5	5	-110,605
5	$\operatorname{tg} x - 5 + 1 / x$	0,5	1,5	-0,377
6	$2x \sin x - e^x$	-3,14	0	5,324
7	$x^3 - \ln x - 2$	1	10	2468,528
8	$e^x + 2 - 3x^2$	-1,2	1,2	4,361
9	$2x^2 - \cos x$	-1,57	1,57	3,165
10	$\operatorname{tg} x - 2x^3$	0	1,57	20,674
11	$2^x - \sin 3x$	-3,14	3,14	12,562
12	$\lvert x^3 \rvert - \cos 2x$	-1,57	1,57	3,041
13	$\ln x - x^2 + 2$	1	1,5	0,317
14	$2^x - 1 - 3x^3$	-10	10	1466,77
15	$x \operatorname{tg} x - 3 + 1 / x$	1	1,5	1,623
16	$2 \sin x - e^x + 1$	-1,5	3,5	-25,907
17	$\lg x + 1 - x$	0,1	0,9	0,111
18	$e^{-x} \sin x + 0,1$	0	100	10,218
19	$e^{-x} \cos x + 0,2$	0	100	20,868
20	$e^{-x} \operatorname{tg} x + 0,3x$	0	10	14,186
21	$0,5 \sin 4x + 2x^2 + e^x$	-5	5	17,899
22	$\cos 2x - 0,1x^2$	-5	5	-8,877
23	$\ln 2x - (x-1)^2$	0,1	2,1	0,486
24	$2^x + 1 - \lvert 4x \rvert$	-5,5	5,5	-44,617
25	$\lvert \operatorname{tg} x \rvert - e^x$	-1,57	1,57	80,910
26	$x \sin x - e^{2x}$	-1,57	1,57	-9,547
27	$2 \lvert \lg x \rvert - 1 + x^3$	0,5	1,5	0,477
28	$e^x - \lvert x^3 \rvert$	-3	3	-20,505
29	$0,1 x^3 + 0,9 - \cos^2 x$	0	3	3,296
30	$0,2 \cdot e^{30x}$	0	0,1	0,127

2. Курсовая работа тему: «Решение уравнения методом половинного деления»

Задано уравнение с одной неизвестной x произвольного вида.
Требуется найти корни уравнения на заданном промежутке [A; B] с точностью ε.
Рассмотрим ход решения задачи по этапам.

2.1. Математическая формулировка задачи

На этом этапе математически описываются условия задачи, выбираются методы расчета и допущения, принимается расчетная схема, составляются расчетные формулы (то есть математическая модель).

Для решения уравнения используем метод половинного деления.

Первоначально заданное уравнение необходимо преобразовать к следующему виду: f(x) = 0. (Это можно сделать, например, вычитанием из левой части уравнения правой части.) Тогда задача сводится к определению таких значений аргумента x, при которых функция f(x) обращается в ноль.

Уравнение может иметь произвольный вид, однако для применения метода половинного деления функция f(x) должна быть непрерывна на заданном промежутке.

Метод половинного деления состоит в следующем. Пусть известен отрезок [a; b], на концах которого функция изменяет свой знак (см. рис. 2.1). Очевидно, что внутри данного отрезка непрерывная функция должна пройти через ноль.

Метод половинного деления

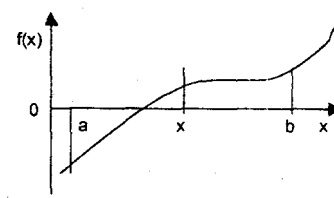

Рис. 2.1.

Отрезок делится пополам, то есть определяется середина x отрезка как среднее арифметическое пределов a и b: $x = (a + b) / 2$.

Сравниваются знаки значений функции на левой границе f(a) и в середине отрезка f(x). Сравнение знаков будем выполнять через перемножение значений функции. Если знаки разные (**$f(a)f(x) \le 0$**), то корень находится в левой половине [a; x] отрезка, и ее надо выбрать для дальнейшего деления, «передвинув» правую границу в точку x: b=x.

Если же знаки одинаковые (**$f(a)f(x) > 0$**), то корень лежит в правой половине [x; b], значит надо выбрать правую половину для дальнейшего деления, «передвинув» левую границу в середину: a=x.

Так деление отрезка пополам и выбор одной половины позволяют за один шаг вдвое сократить длину отрезка, содержащего корень. После этого выполняется следующий шаг, но уже над выбранной половиной, что снова сокращает отрезок вдвое. Процесс деления продолжается до тех пор, пока длина отрезка не станет меньше заданной погрешности ε: $b - a \le \varepsilon$.

Таким образом, в результате последовательных половинных делений удается локализовать корень с любой заданной точностью ε. Окончательно в качестве корня можно принять среднее арифметическое от пределов полученного отрезка: x=(a+b)/2.

Выбор участков, содержащих корни уравнения. Корни уравнения ищутся на промежутке от A до B. Чтобы найти все корни на промежутке [A,B], надо разбить его на малые участки и выбрать из них те, на которых функция меняет знак. Тогда для определения корня можно на каждом подобном участке применить метод половинного деления, как описано выше.

Разобьем заданный промежуток на k участков длиной dx (см. рис. 2.2):

$dx = (b-a) / k$.

Перебирая последовательно все участки, будем проверять условие

$f(x)\ f(x+dx) \le 0$,

показывающее, что данный участок [x, x+dx] содержит корень.

В случае выполнения этого условия необходимо обратиться к методу половинного деления, определить с заданной точностью корень и перейти к проверке следующего участка. При этом для начала половинного деления принимается: a=x, b=x+dx.

Выбор участков с корнями

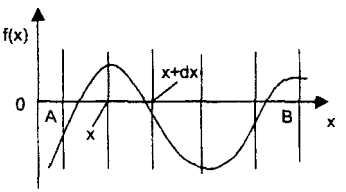

Рис. 2.2.

Если же условие $f(x)\ f(x+dx) \le 0$ не выполняется, то переход происходит сразу.

2.2. Алгоритм

При составлении алгоритма математическая формулировка представляется как последовательность простых действий (план работы программы). Программист обдумывает общую схему решения, организацию ввода исходных данных и вывода результатов, взаимодействие отдельных этапов расчета. В результате формируется логическая схема программы, наглядно изображающая алгоритм, обычно в виде блок-схемы.

Алгоритм решения задачи удобно подразделить на три алгоритма:

- главной функции (основной алгоритм);
- выбора участков с корнями;
- половинного деления.

Выделение частей в обособленные алгоритмы очень полезно, так как упрощает решение общей задачи. Каждая "элементарная" задача прорабатывается отдельно, ее алгоритм, а затем и функцию, легче составлять и проверять.

Составные части алгоритма можно, в принципе, разрабатывать в любой последовательности или разными программистами одновременно. Мы будем при разработке алгоритмов идти от общего к частному.

2.2.1. Основной алгоритм

На основной алгоритм обычно возлагаются задачи ввода исходных данных, вызова вычислительной части алгоритма и вывода результатов.

Сами по себе задачи организации ввода-вывода являются важными и трудоемкими, но логически эта часть представляет собой довольно простой алгоритм. Приведем его на рис. 2.3.

При определении корней уравнения они присваиваются элементам массива X, причем заранее не известно количество n вычисленных корней; оно определится одновременно с корнями.

При организации печати придется применить циклический алгоритм, чтобы напечатать массив корней X заранее не известной размерности. Однако на блок-схеме печать результатов представлена одним блоком.

Блок-схема основного алгоритма

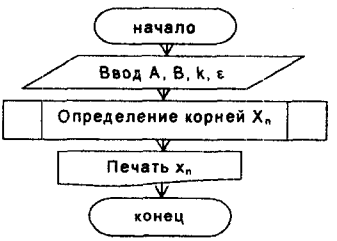

Рис. 2.3.

2.2.2. Алгоритм выбора участков с корнями

В этой части реализуется процесс перебора участков заданного промежутка [A, B] и поиск среди них тех участков, которые содержат корни уравнения.

Блок-схема данного алгоритма приведена на рис. 2.4.

Для перебора участков организован цикл, в качестве счетчика использована координата x начала каждого элементарного участка длиной dx; x должно изменяться от A до B-dx, то есть перебор выполняется до тех пор, пока x остается меньше B.

Для каждого такого участка проверяется условие наличия корня (то есть меняет ли функция знак на концах участка). В случае истинности условия вызывается алгоритм половинного деления, при помощи которого вычисляется значение корня X_i.

Для подсчета количества корней введен вспомогательный счетчик i, который увеличивается на единицу при вычислении каждого нового корня. После выхода из цикла этот счетчик будет содержать общее количество корней, которое возвращается в качестве результата.

Блок-схема алгоритма выбора участков с корнями

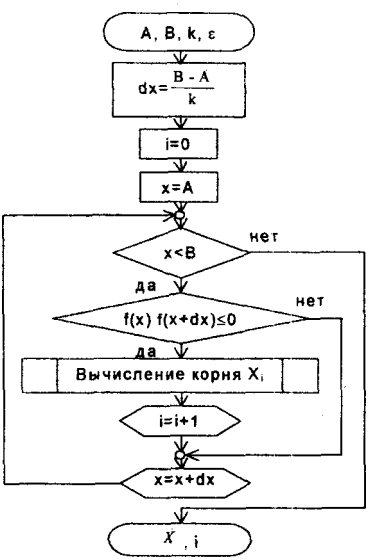

Рис. 2.4.

2.2.3. Алгоритм половинного деления

На рис. 2.5 оказана блок-схема алгоритма. В качестве входных параметров задаются границы a, b участка, на котором функция меняет знак, и точность вычисления корня ε.

Процесс половинного деления представляет собой цикл без специального счетчика, так как не выполняется фиксированное количество раз. Цикл выполняется до тех пор, пока длина отрезка [a; b] остается больше ε.

Тело цикла - это разветвленный алгоритм. В зависимости от выполнения условия f(a)f(x)≤0 либо правая (b), либо левая (a) граница передвигается в середину промежутка (x). Так как длина промежутка при каждом проходе цикла сокращается вдвое, выход из цикла должен рано или поздно произойти (программа не должна "зациклиться").

Блок-схема алгоритма половинного деления

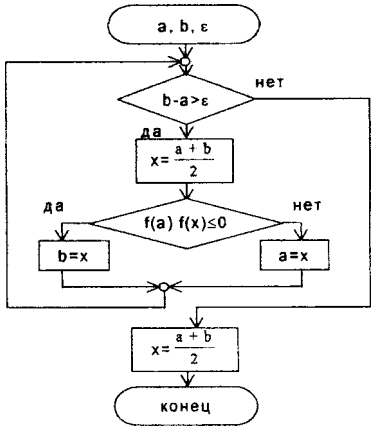

Рис. 2.5.

2.3. Составление программы

На этом этапе решения задачи разработанный алгоритм переводится на *язык программирования*, в данном случае - на язык Си. Программу будем составлять в той же последовательности, что и части алгоритма. Для каждой части разработаем отдельную функцию:

- для основного алгоритма - функцию main;
- для алгоритма поиска участков с корнями - функцию select;
- для алгоритма вычисления корня методом половинного деления - функцию divide;
- для заданной левой части уравнения - функцию f.

Хотя для последней функции не было составлено алгоритма, при составлении алгоритма половинного деления подразумевалось, что эта функция задана.

Когда разрабатывают функцию, то создают ее определение, включающее заголовок и тело функции (выполняемые в ней операторы). Кроме того, все функции, кроме main, перед их вызовом должны быть описаны. Описания функций целесообразно помещать в начало программы.

Составим программу для одного конкретного варианта - варианта №30 из приведенных в разделе 2.5. Различия вариантов касаются только функций f (вид уравнения) и main (границы заданного промежутка).

2.3.1. Главная функция

Составим функцию main, реализующую алгоритм, показанный на рис. 2.3.

В начале необходимо описать все переменные, которые будут использоваться. При этом руководствуются "физическим смыслом" величин. Большинство переменных является действительными величинами, для их описания применим тип float (тип - "с плавающей точкой обычной точности"). Обозначения величин постараемся приблизить к принятым в алгоритме, единственная трудность - обозначить точность ε английскими буквами (eps).

При описании массива X укажем максимально возможное количество корней, равное количеству участков разбиения (100).

Опишем также целые переменные, которые нужны для задания количества k участков разбиения и для организации печати корней уравнения в цикле (i - общее количество корней, j - счетчик корней).

Ввод исходных данных организуем наиболее простым способом - при описании переменных.

Определение корней состоит в вызове функции select, которая заполняет массив X корнями уравнения, а также возвращает количество i этих корней.

Вывод результатов необходимо организовать в цикле, так как количество корней заранее не известно. Цикл удобно реализовать при помощи конструкции for (наиболее компактно для данного случая).

```
void main() {
        /* Главная функция */
        float A=-0.99, B=1.5, eps=0.001, X[100];
        int k=100, i, j;
        i=select(A, B, eps, k, X);
        printf("Корни уравнения:\n");
        for(j=0; j<i; j++)
                printf("%d) %7.3f\n", j+1, X[j]);
}
```

Вывод результатов производится через вызов функции printf - стандартной функции вывода на экран. Для ее использования необходимо в начало программы включить описания стандартных функций ввода-вывода среди прочих описаний:

```
#include <stdio.h>
```

По этой команде препроцессор включит в исходный текст программы содержимое файла stdio.h, содержащего описания указанных функций.

2.3.2. Функция выбора участков с корнями

В функции select реализуем алгоритм, приведенный на рис. 2.4.

Определение функции имеет вид:

```
int select(float A, float B, float eps, int k, float* X) {
        /* Функция выбора участков с корнями */
        float x, dx;
        int i;
        dx=(B-A)/(float)k;
        i=0;
        for(x=A; x<B; x+=dx) {
                if(f(x)*f(x+dx)<=0.) {
                        X[i]=divide(x, x+dx, eps);
                        i++;
                }
        }
        return(i);
}
```

В начале тела функции описываются переменные двух типов:

- с плавающей точкой (float) - x и dx;
- целое (int) - переменная i.

Кроме того, передаваемые функции параметры A, B, k, X описываются в заголовке функции. Интересно отметить описание массива корней X как указателя (float*). Это сделано для того, чтобы "не привязываться" к определенной размарности массива. Работа же с элементами массива производится так же, как и в случае обычного описания массива.

Вычисление dx содержит операцию преобразования типа переменной k, которая должна быть подставлена в выражение в виде float (как действительное значение).

Цикл перебора участков разбиения организуем через ключевое слово for. В заголовке цикла изменение счетчика операция x+=dx представляет собой краткую запись, эквивалентную x=x+dx.

При вычислении каждого корня счетчик корней увеличивается на единицу (операция i++), тогда при выходе из цикла i будет содержать общее количество корней, которое передается в качестве возвращаемого значения (оператор return). Так как тип i описан как int, этот же тип должен быть описан в качестве типа функции в ее заголовке (перед именем функции select).

Описание функции (помещаемое в начало программы):

```
int select(float A, float B, float eps, int k, float* X);
```

2.3.3. Функция вычисления корня методом половинного деления

Данная функция реализует алгоритм, изображенный на рис. 2.5.

Определение функции divide:

```
float divide(float a, float b, float eps) {
        float x;
        while((b-a)>eps) {
                x=(a+b)/2.;
                if(f(a)*f(x)<=0.)
                        b=x;
                else
                        a=x;
        }
        x=(a+b)/2.;
        return(x);
}
```

Аргументы функции a, b, eps описываются в ее заголовке, а переменная x - в начале тела функции. Все переменные, а также возвращаемое значение функции, имеют тип float ("с плавающей точкой обычной точности").

Для реализации цикла (не имеющего счетчика) удобно применить конструкцию while. А для разветвленной части алгоритма (внутри тела цикла) - конструкцию if-else.

Описание функции (помещаемое в начало программы):

```
float divide(float a, float b, float eps);
```

2.3.4. Заданная функция

Функцию, задающую уравнение, создадим под именем f с единственным аргументом. И аргумент, и возвращаемое значение имеют тип float. В функции реализуется простейший алгоритм, поэтому в разделе 2.2 он не приводился.

Определение функции:

```
float f(float x) {
        /* Заданная функция */
        return( tan(x)-log10(x+1.)-0.2 );
}
```

Очевидно, что практически для всех вариантов функции требуется использование стандартных математических функций (в данном варианте, например, tan - тангенса и log10 - десятичного логарифма). Поэтому среди описаний в начало программы необходимо включить описания этих функций:

```
#include <math.h>
```

Это команда, по которой препроцессор включит в исходный текст программы содержимое файла math.h, содержащего описания математических функций.

Описание функции (помещаемое в начало программы):

```
float f(float x);
```

2.3.5. Структура программы

Разработаны все составные части программы расчета. Осталось их правильно расположить в тексте программы. Сделаем это в следующей последовательности:

- комментарий, показывающий назначение программы в целом;
- описания всех разработанных и стандариных функций;
- определение функции main;
- определения функций select, divide и f.

Таким образом, для варианта №30 получим следующий исходный текст:

```
/* Решение уравнения методом половинного деления.
   Вариант №30.*/
/* Описания стандартных функций */
#include <stdio.h>
#include <math.h>
/* Описания разработанных функций */
int select(float A, float B, float eps, int k, float* X);
float divide(float a, float b, float eps);
float f(float x);
/* Определения разработанных функций */
void main() {
        /* Главная функция */
        float A=-0.99, B=1.5, eps=0.001, X[100];
        int k=100, i, j;
        i=select(A, B, eps, k, X);
        printf("Корни уравнения:\n");
        for(j=0; j<i; j++)
                printf("%d) %7.3f\n", j+1, X[j]);
}
```

```
int select(float A, float B, float eps, int k, float* X) {
        /* Функция выбора участков с корнями */
        float x, dx;
        int i;
        dx=(B-A)/(float)k;
        i=0;
        for(x=A; x<B; x+=dx) {
                if(f(x)*f(x+dx)<=0.) {
                        X[i]=divide(x, x+dx, eps);
                        i++;
                }
        }
        return(i);
}
float divide(float a, float b, float eps) {
        float x;
        while((b-a)>eps) {
                x=(a+b)/2.;
                if(f(a)*f(x)<=0.)
                        b=x;
                else
                        a=x;
        }
        x=(a+b)/2.;
        return(x);
}
float f(float x) {
        /* Заданная функция */
        return( tan(x)-log10(x+1.)-0.2 );
}
```

2.4. Выполнение программы и анализ результатов

На этом этапе составленную программу вводят в память ЭВМ, записывают в виде исходного файла под именем, например, equation.c и выполняют обработку программы вплоть до получения результатов. Этот комплекс задач удобно решать в среде программирования Borland C++, как описано в главе 3.

В случае, если все этапы обработки и выполнения программы пройдут без ошибок, на экран будут выведены результаты расчета. Для варианта №30 следующие:

```
Корни уравнения:
1)  -0.979
2)   0.306
```

Полученные корни можно сравнить с ответами, приведенными в задании по каждому варианту, и переходить к оформлению работы.

2.5. Варианты заданий

Найти все корни уравнения на заданном промежутке.

В таблице 2.1 приведены варианты заданий с ответами. Каждому варианту соответствует свой вид уравнения и промежуток.

Для любого варианта точность ε следует принять равной 0,001. Количество k участков разбиения - 100.

Таблица 2.1

Варианты заданий

№	Уравнение	Промежуток	Корни уравнения
1	$\sin 2x + 1 = e^x$	[-1,5; 1,5]	{-1,186; 0; 0,683}
2	$\cos 2x - 0,5 = x^3$	[-5; 5]	{-0,896; -0,729; 0,464}
3	$10 \ln x = 2x^2 - 5$	[0,1; 5]	{0,662; 2,749}
4	$2^x + 1 = 2x^2$	[-5; 5]	{-0,879; 1,323}
5	$\operatorname{tg} x = 5 - 1 / x$	[0,1; 1,5]	{0,209; 1,340}
6	$2x \sin x = e^x$	[-5; 5]	{-3,135; -0,55}
7	$\ln x + 2 = x^3$	[0,1; 5]	{0,136; 1,315}
8	$e^x + 2 = 3x^2$	[-5; 5]	{-0,896; 1,440; 3,618}
9	$\cos x = 2x^2$	[-5; 5]	{-0,635; 0,635}
10	$\operatorname{tg} x = 2x^3$	[0; 1,5]	{0; 0,803; 1,384}
11	$\sin 3x = 2^x$	[-4; 4]	{-3,178; -2,011; -1,198}
12	$\cos 2x = \lvert x^3 \rvert$	[-5; 5]	{-0,648; 0,648}
13	$\ln x = x^2 - 2$	[0,1; 5]	{0,138; 1,564}
14	$2^x - 1 = x^3$	[-5; 5]	{-0,737; 0; 1}
15	$x \operatorname{tg} x = 3 - 1 / x$	[0,1; 1,5]	{0,348; 1,089}
16	$2 \sin x = e^x - 1$	[-5; 5]	{-2,658; 0; 0,978}
17	$\lg x + 1 = x$	[0,1; 5]	{0,137; 1}
18	$e^{-x} \sin x = 0,1$	[-5; 5]	{-3,146; 0,112; 2,135}
19	$e^{-x} \cos x = 1$	[-5; 5]	{-4,721; -1,293; 0}
20	$e^{-x} \operatorname{tg} x = 2x$	[-1,5; 1,5]	{-0,574; 0; 1,496}
21	$0,5 \sin 4x + 2x^2 = e^x$	[-3; 3]	{-0,636; 1,535; 2,512}
22	$\cos 2x = 0,1x^2$	[0; 5]	{0,757; 2,812; 3,161}
23	$\ln 2x = (x-1)^2$	[0,1; 5]	{0,591; 2,221}
24	$2^x + 1 = \lvert 4x \rvert$	[-5; 5]	{-0,435; 0,639; 3,846}
25	$\lvert \operatorname{tg} x \rvert = e^x$	[-1,5; 1,5]	{-0,531; 1,306}
26	$x \sin x = e^{2x}$	[-5; 5]	{-3,141; -0,578}
27	$2 \lvert \lg x \rvert = 1 - x^3$	[0,1; 5]	{0,329; 1}
28	$e^x = \lvert x^3 \rvert$	[-5; 5]	{-0,773; 1,857; 4,536}
29	$\cos^2 x = x^3 + 0,9$	[-5; 5]	{-0,616; -0,461; 0,282}
30	$\operatorname{tg} x = \lg(x+1)+0,2$	[-0,99; 1,5]	{-0,979; 0,306}

3. Выполнение и оформление курсовой работы

Выполнение курсовой работы состоит в изучении и освоении студентами *метода решения поставленной задачи* и *алгоритма*, в *составлении программы*, реализующей данный алгоритм применительно к конкретному варианту задания.

Составленная программа должна быть введена и выполнена в среде программирования, для чего можно воспользоваться, например, следующей краткой последовательностью команд:

1) запуск среды Borland C++

Пуск → Программы → Borland C++ 3.1 → Borland C++

2) установка рабочего каталога

Меню → File → Change dir → A:\ (если работаем с дискетой) или C:\STUDENT (если работаем с жестким диском) → OK

3) создание нового текста программы

Меню → File → New → *ввести текст составленной программы*

4) сохранение исходного файла на диске

Меню → File → Save → integral.c (для вычисления интеграла) или equation.c (для решения уравнения) → OK

5) запуск программы на обработку и выполнение:

Меню → Run → Run

6) просмотр результатов на экране:

Меню → Window → Output

7) выход из среды программирования:

Меню → File → Quit

Помимо собственно решения поставленной задачи при выполнении курсовой работы, студенты должны определенным образом оформить результаты своей работы. Это имеет целью ориентировать студентов на необходимость представления результатов проведенных исследований как в процессе обучения, так и в инженерной деятельности.

Курсовая работа оформляется в виде пояснительной записки.

Титульный лист работы можно оформить подобно оформлению обложки методических указаний, озаглавив его "Поянительная записка" (вместо слов "Методические указания") и указав ниже:

- тему работы;
- номер варианта задания;
- "Выполнил: студент гр. ..."
- "Проверил ...".

Содержание записки состоит из введения, условия задачи, описания основных этапов решения задачи и результатов. В целом при описании методики решения задачи можно придерживаться последовательности изложения материала в методических указаниях. Однако от каждого студента требуется самому разобраться в решении задачи и объяснить ход решения, не копируя текст методических указаний.

После изложения математической формулировки и алгоритма следует привести полный текст программы, соответствующий заданному варианту. Если в процессе запуска программы приходилось исправлять ошибки, то все необходимые изменения надо повторять и в тексте пояснительной записки.

В заключение пояснительной записки приводятся полученные результаты.

Общий объем записки может составить 4-7 листов, записка должна быть аккуратно оформлена, желательно, при помощи текстового редактора.

Для сдачи курсовой работы необходимо:

1) запустить программу на занятии и продемнострировать результаты преподавателю;

2) оформить пояснительную записку, включив в нее текст выполненной программы и полученные результаты;

3) защитить курсовую работу преподавателю.

Список литературы:

1. Чугунов Г.Ф., Меланин В.М., Беспалько С.В. Язык Си для студентов вагонной специальности. Часть 1: Учебное пособие. - М.: МИИТ, 2001. - 74 с.

2. Чугунов Г.Ф. и др. Язык Си для студентов вагонной специальности. Часть 2: Учебное пособие. - М.: МИИТ, 2002. - 80 с.

3. Керниган Б., Ритчи Д. Язык программирования Си. - М.: Финансы и статистика, 1992. - 272 с.

4. Уэйт М., Прата С., Мартин Д. Язык Си: Руководство для начинающих. - М.: Мир, 1988. - 512 с.

Содержание

Учебно-методическое издание

Беспалько Сергей Валерьевич.
Чугунов Геннадий Федосович

Решение задачи на ЭВМ с программированием на языке Си
Методические указания к курсовой работе по дисциплине "Информатика"

Подписано к печати 14.11.02 Изд.№ 221-02 Формат 60*84/16
Усл. п.л. 1,5 Заказ № 1156. Тираж 200 экз. Цена 9 руб. 50 коп.

127994, Москва, ул. Образцова, 15, Типография МИИТа