МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего
профессионального образования
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПИЩЕВЫХ ПРОИЗВОДСТВ»

Е.И. Конопленко, А.П. Лапусь

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по курсу: «Информатика»

(раздел: «компьютерные технологии вычисления в математическом моделировании»)

Москва, 2010г.

Оглавление

Введение 5

ЛАБОРАТОРНАЯ РАБОТА № 1 «Статистическая обработка результатов эксперимента» 5

Теоретические сведения 6
Математическая постановка задачи 6
Определение значимости коэффициента корреляции 8
Пример выполнения работы 8
Таблица значений критерия Стьюдента 9
БЛОК-СХЕМА 11
ПРОГРАММА НА ЯЗЫКЕ QBASIC 12
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 14
ПРИМЕР РАБОТЫ в EXCEL 14

Контрольные вопросы 15

ЛАБОРАТОРНАЯ РАБОТА № 2 «Численное интегрирование» 15

1. Цель работы. 15
2. Основные теоретические сведения. 16
1). Метод прямоугольников 16
2) Метод трапеций 18
3) Метод парабол 18
3. Порядок выполнения работы 19
Пример выполнения работы 19
БЛОК-СХЕМА 20
ВИД ПРОГРАММЫ НА ЯЗЫКЕ QBASIC 22
РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ В Qbasic 24
Результат расчета в ППП ЭВРИКА. 25

Методические указания к выполнению лабораторной работы на ПК 25

Контрольные вопросы 26

Варианты заданий для самостоятельного решения 26

Задание 26

ЛАБОРАТОРНАЯ РАБОТА № 3 «Уточнение корня уравнения» 29

1. Цель работы 29
2. Основные теоретические положения 29
1). Метод дихотомии 29
2). Метод касательных 31
3). Метод простой итерации 31
4). Метод хорд 33
3. Порядок выполнения работы 34
Пример выполнения лабораторной работы. 35
БЛОК-СХЕМА 36
ВИД ПРОГРАММЫ НА ЯЗЫКЕ QBASIC 38
РЕЗУЛЬТАТЫ РАБОТЫ В QBASIC 41
РЕЗУЛЬТАТЫ РАБОТЫ в Eureka. 42

Контрольные вопросы 42

Варианты заданий для самостоятельного решения 43
Задание. 43
ЛАБОРАТОРНАЯ РАБОТА № 4 «Методы численного решения дифференциальных уравнений. Уравнения 1-го порядка» 48
Цель работы 48
Метод Эйлера 51
Метод Эйлера - Коши 51
Метод Руге - Кутта 51
Правило Рунге - Ромберга 52
Пример решения поставленной задачи 52
БЛОК-СХЕМА АЛГОРИТМА РЕШЕНИЯ 53
ВИД ПРОГРАММЫ НА ЯЗЫКЕ QBASIC 55
Построение в Excel графика решений 58
Контрольные вопросы 60
Варианты заданий к лабораторной работе 62
ЛАБОРАТОРНАЯ РАБОТА № 5 Символьные переменные 64
Цель работы 64
Инструменты обработки текстовых величин 67
Базовые алгоритмы обработки текста 75
Сортировка текстовых массивов 81
Контрольные вопросы 93
Варианты заданий для самостоятельного решения 94
ЛАБОРАТОРНАЯ РАБОТА № 6 Оптимизация технологического процесса. 96
Методы оптимизации функции 1-ой переменной 96
Цель работы 96
Оптимизация функций одной переменной 96
Методы оптимизации функций одной переменной 101
Метод поразрядного приближения 101
Метод дихотомии 101
Метод Фибоначчи 102
Метод золотого сечения 103
Использование ППП Eureka и Excel при решении задач оптимизации 104
Содержание отчета 105
Пример выполнения лабораторной работы 106
БЛОК-СХЕМА 106
ПРОГРАММА НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ QBASIC 108
РЕЗУЛЬТАТ в Qbasic 110
Решение задачи с использованием ППП Eureka 110
Задания 111
Контрольные вопросы 112
ЛАБОРАТОРНАЯ РАБОТА № 7 Работа с файлами последовательного доступа 112
Цель работы 112
Работа с файлами 112
Требования к имени файла 113

Расширение файла....113
Операции над файлами....115
Порядок выполнения работы....120
Содержание отчета....121
Пример решения задачи....121
ПРОГРАММА НА ЯЗЫКЕ QBasic....122
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ....123
Контрольные вопросы....124
Варианты заданий к лабораторной работе....124
Список литературы....135

Введение

Широкое внедрение математических методов в самые разнообразные сферы деятельности сегодня уже никого не удивляет. Это не только технические и экономические науки, но и развивающиеся прикладные науки управления: менеджмент, логистика, социально-экономическое прогнозирование и т.д.

Математическое моделирование становится одним из главных направлений в технике, экономике, социологии, биологии и других областях. Поэтому специалистам различных направлений необходимо владеть концепциями и методами математического моделирования, иметь представление об инструментах, применяемых в моделировании.

При изучении курса информатики, студент знакомится с основами алгоритмизации и программирования, с пакетами прикладных программ общего назначения.

Данный лабораторный практикум по курсу «Информатики» включает следующие темы:

1. статистическая обработка результатов эксперимента
2. вычисление интегралов
3. решение нелинейных уравнений
4. решение дифференциальных уравнений
5. оптимизация технологического процессов
6. работа с файлами последовательного доступа
7. символьные переменные.

ЛАБОРАТОРНАЯ РАБОТА № 1

«Статистическая обработка результатов эксперимента»

Цель работы ознакомление с основными характеристиками случайных величин.

В результате эксперимента определились такие показатели, как рост и вес человека.

рост	160 x_1	170 x_2	156 x_3	.....	.....	.....	..... X_n
вес	62 Y_1	81 Y_2	60 Y_3	.....	.....	.....	..... Y_n

Обозначим x_i – рост, y_i – вес. Количество экспериментов- n, где n – размер выборки, i – текущий индекс.

Необходимо определить характеристики случайных величин (величин x и y).

Теоретические сведения

Математическая постановка задачи

(характеристики случайных величин)

1. Математическое ожидание (среднее значение)

$$\text{по x: } M_x = \bar{x} = \frac{x_1 + x_2 + x_3 + ... + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$\text{по y: } M_y = \bar{y} = \frac{y_1 + y_2 + y_3 + ... + y_n}{n} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Математическое ожидание характеризует положение случайной величины на числовой оси.

2. Дисперсия

$$\text{по x: } D_x = \frac{\sum_{i=1}^{n} (x_i - M_x)^2}{n-1} \qquad \text{по y: } D_y = \frac{\sum_{i=1}^{n} (y_i - M_y)^2}{n-1}$$

Где M_x, M_y – математическое ожидание.

Дисперсия характеризует разброс случайных величин. В данных формулах – разброс относительно математического ожидания.

3. Среднее квадратическое отклонение

$$\text{по x: } \delta_x = \sqrt{D_x} \qquad \text{по y: } \delta_y = \sqrt{D_y}$$

Эта величина называется также стандартным отклонением, выражается в тех же единицах, что и величины, полученные в результате эксперимента. И зачастую оказывается более удобной характеристикой, чем дисперсия. Чем слабее варьирует признак, тем меньше среднее квадратическое отклонение.

4. Коэффициент вариации

Коэффициент вариации необходим для сравнения изменчивости признаков, выраженных разными единицами. Дисперсия и среднее квадратическое отклонение – величины абсолютные, именованные, выражаемые в тех же единицах, что и характеризуемый ими признак.

Коэффициент вариации – относительный показатель, представляет процентное отношение среднего квадратического отклонения к математическому ожиданию

по х: $V_x = \frac{\delta_x}{M_x} \cdot 100\%$ по у: $V_y = \frac{\delta_y}{M_y} \cdot 100\%$

5. Нормированное отклонение

Нормированное отклонение – показатель, представленный отклонением той или иной величиной от математического ожидания, отнесённое к величине среднего квадратического отклонения:

по х: $t_{x_i} = \frac{x_i - M_x}{\delta_x}$ по у: $t_{y_i} = \frac{y_i - M_y}{\delta_y}$

6. Коэффициент корреляции

Коэффициент корреляции характеризует степень линейной зависимости (степень связи) между величинами х и у.

Вычисляется по формуле:

$$K_{xy} = \frac{\sum_{i=1}^{n}(x_i - M_x)(y_i - M_y)}{(n-1)\delta_x \cdot \delta_y}$$

Или

$$K_{xy} = \frac{\sum_{i=1}^{n}(x_i - M_x)(y_i - M_y)}{\sqrt{\sum_{i=1}^{n}(x_i - M_x)^2 \cdot \sum_{i=1}^{n}(y_i - M_y)^2}}$$

Значение $К_{ху}$ изменяется в пределах от -1 до +1. Если значение $К_{ху} > 0$, то корреляция положительная (с ростом х значение у увеличивается), если $К_{ху} < 0$, то корреляция отрицательная (с ростом х значение у уменьшается).

При значении $|К_{ху}|$ близком к 1 существует линейная зависимость между *x* и *y*, т.е. $y = a + bx$, знак корреляции совпадает со знаком коэффициента b.

Определение значимости коэффициента корреляции

Уровень значимости коэффициента корреляции может быть определён по критерию Стьюдента:

$$T_{\text{расч}} = \frac{\left|K_{xy}\right|\sqrt{n-2}}{\sqrt{1-(K_{xy})^2}}$$

Если $T_{\text{расч}} > T_{\text{табл}}(f,\alpha)$, где α - уровень значимости α=0.95, а f - число степеней свободы f=n-2, то можно утверждать, что между х и у существует линейная зависимость, в противном случае – линейная зависимость отсутствует. Значение $Т_{табл}$ выбирается по таблице значений критерия Стьюдента.

Отчет выполненной данной работы содержит:

1. Содержательную постановку задачи (выбор значений х и у, полученных в результате эксперимента)
2. Математическую постановку задачи (функции, по которым проводились расчеты)
3. Блок-схему алгоритма решения задач
4. Программу на алгоритмическом языке
5. Результат работы программы
6. Вывод по расчётам
7. Расчет, полученный в Excel

Содержательную постановку задачи каждый студент выполняет самостоятельно.

Пример выполнения работы

Провести расчет на Qbasic и в Excel.

Содержательная постановка задачи

Проведён эксперимент по определению зависимости коэффициента активности от ионной силы раствора.

Для этих величин необходимо рассчитать все характеристики случайных величин. Сделать вывод о наличии или отсутствии линейной зависимости

№	Ионная сила раствора X(i)	Коэффициент активности Y(i)
1	0,001	0,98
2	0,002	0,97
3	0,005	0,95
4	0,01	0,92
5	0,04	0,9
6	0,05	0,84
7	0,1	0,81
8	0,2	0,8
9	0,3	0,81
10	0,4	0,82
11	0,5	0,84

Табличные значения коэффициента Стьюдента = **2.201**

Таблица значений критерия Стьюдента

f	***q***							
	0.80	**0.90**	**0.95**	**0.98**	**0.99**	**0.995**	**0.998**	**0.999**
1	3.0770	6.3130	12.7060	31.820	63.656	127.656	318.306	636.619
2	1.8850	2.9200	4.3020	6.964	9.924	14.089	22.327	31.599
3	1.6377	2.35340	3.182	4.540	5.840	7.458	10.214	12.924
4	1.5332	2.13180	2.776	3.746	4.604	5.597	7.173	8.610
5	1.4759	2.01500	2.570	3.649	4.0321	4.773	5.893	6.863
6	1.4390	1.943	2.4460	3.1420	3.7070	4.316	5.2070	5.958

7	1.4149	1.8946	2.3646	2.998	3.4995	4.2293	4.785	5.4079
8	1.3968	1.8596	2.3060	2.8965	3.3554	3.832	4.5008	5.0413
9	1.3830	1.8331	***2.2622***	2.8214	3.2498	3.6897	4.2968	4.780
10	1.3720	1.8125	2.2281	2.7638	3.1693	3.5814	4.1437	4.5869
11	1.363	1.795	2.201	2.718	3.105	3.496	4.024	4.437
12	1.3562	1.7823	2.1788	2.6810	3.0845	3.4284	3.929	4.178
13	1.3502	1.7709	2.1604	2.6503	3.1123	3.3725	3.852	4.220
14	1.3450	1.7613	2.1448	2.6245	2.976	3.3257	3.787	4.140
15	1.3406	1.7530	2.1314	2.6025	2.9467	3.2860	3.732	4.072
16	1.3360	1.7450	2.1190	2.5830	2.9200	3.2520	3.6860	4.0150
17	1.3334	1.7396	2.1098	2.5668	2.8982	3.2224	3.6458	3.965
18	1.3304	1.7341	2.1009	2.5514	2.8784	3.1966	3.6105	3.9216

Значения критерия Стьюдента (t-критерия) для различного уровня значимости q и числа степеней свободы f (f = n-2 или n, n-число опытов).

1. Математическое ожидание: $M_x = \frac{\sum_{i=1}^{n} x_i}{n}$ и $M_y = \frac{\sum_{i=1}^{n} y_i}{n}$

2. Дисперсия: $D_x = \frac{\sum_{i=1}^{n} (x_i + M_x)^2}{n-1}$ и $D_y = \frac{\sum_{i=1}^{n} (y_i + M_y)^2}{n-1}$

3. Среднее квадратическое отклонение: $\delta_x = \sqrt{D_x}$ и $\delta_y = \sqrt{D_y}$

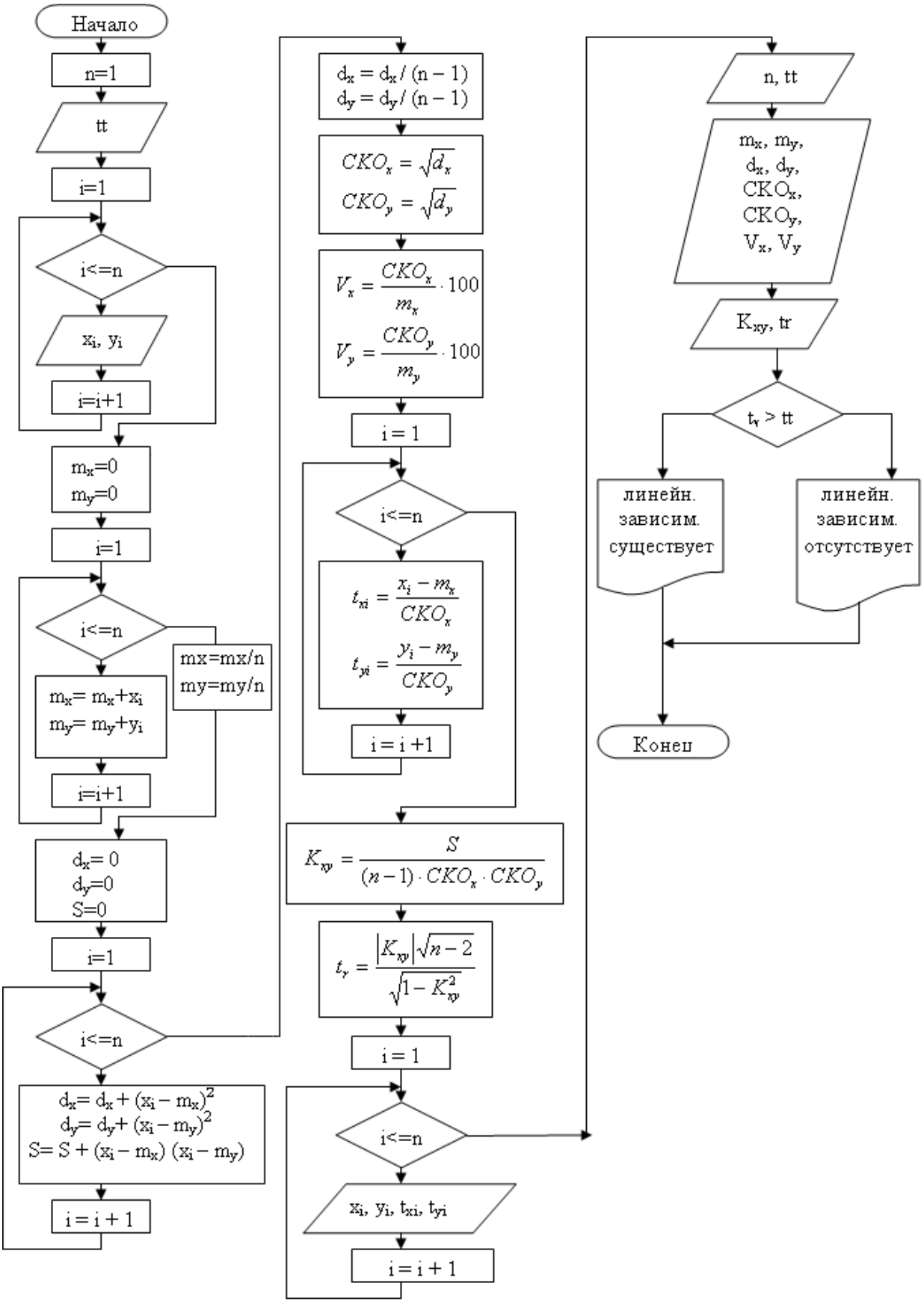
4 Коэффициент вариации: $V_x = \frac{\delta_x}{M_x} \cdot 100\%$ и $V_y = \frac{\delta_y}{M_y} \cdot 100\%$

5. Нормированное отклонение: $t_{x_i} = \frac{x_i - M_x}{\delta_x}$ и $t_{y_i} = \frac{y_i - M_y}{\delta_y}$

6. Коэффициент корреляции: $$K_{xy} = \frac{\sum_{i=1}^{n} (x_i - M_x)(y_i - M_y)}{(n-1)\delta_x \cdot \delta_y}$$

7. Критерий Стьюдента: $$T_r = \frac{|K_{xy}|\sqrt{n-2}}{\sqrt{1-(K_{xy})^2}}$$

БЛОК-СХЕМА

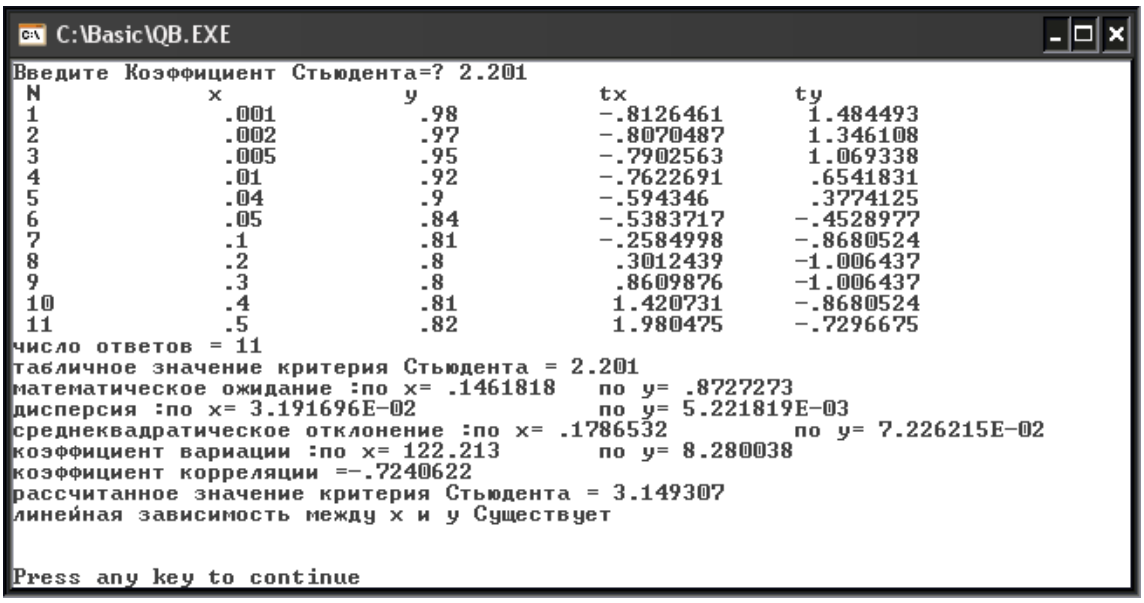


ПРОГРАММА НА ЯЗЫКЕ QBASIC

```
CLS
n = 11
INPUT "Введите Коэффициент Стьюдента="; tt
DIM x(1 TO n), y(1 TO n), tx(1 TO n), ty(1 TO n)
DATA 0.001, 0.002, 0.005, 0.01, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5
FOR i = 1 TO n
READ x(i)
NEXT i
DATA 0.98, 0.97, 0.95, 0.92, 0.9, 0.84, 0.81, 0.8, 0.8, 0.81, 0.82, 0.84
FOR i = 1 TO n
READ y(i)
NEXT i
mx = 0: my = 0
FOR i = 1 TO n
mx = mx + x(i)
my = my + y(i)
NEXT i
mx = mx / n: my = my / n
dx = 0: dy = 0: S = 0
FOR i = 1 TO n
dx = dx + (x(i) - mx) ^ 2
dy = dy + (y(i) - my) ^ 2
S = S + (x(i) - mx) * (y(i) - my)
NEXT i
dx = dx / (n - 1): dy = dy / (n - 1)
CKOx = SQR(dx): CKOy = SQR(dy)
Vx = CKOx * 100 / mx: Vy = CKOy * 100 / my
FOR i = 1 TO n
tx(i) = (x(i) - mx) / CKOx
```

```
 ty(i) = (y(i) - my) / CKOy
 NEXT i
 Kxy = S / ((n - 1) * CKOx * CKOy)
 tr = ABS(Kxy) * SQR(n - 2) / SQR(1 - Kxy ^ 2)
 PRINT " N", "x", "y", "tx", "ty"
 FOR i = 1 TO n
 PRINT i, x(i), y(i), tx(i), ty(i)
 NEXT i
PRINT "число ответов ="; n
PRINT "табличное значение критерия Стьюдента ="; tt
PRINT "математическое ожидание :"; "по x="; mx, "по y="; my
PRINT "дисперсия :"; "по x="; dx, "по y="; dy
PRINT "среднеквадратическое отклонение :"; "по x="; CKOx, "по y="; CKOy
PRINT "коэффициент вариации :"; "по x="; Vx, "по y="; Vy
PRINT "коэффициент корреляции ="; Kxy
PRINT "рассчитанное значение критерия Стьюдента ="; tr
IF tr >= tt THEN
PRINT "линейная зависимость между x и y Существует"
ELSE
PRINT "линейная зависимость между x и y Отсутствует"
END IF
END
```

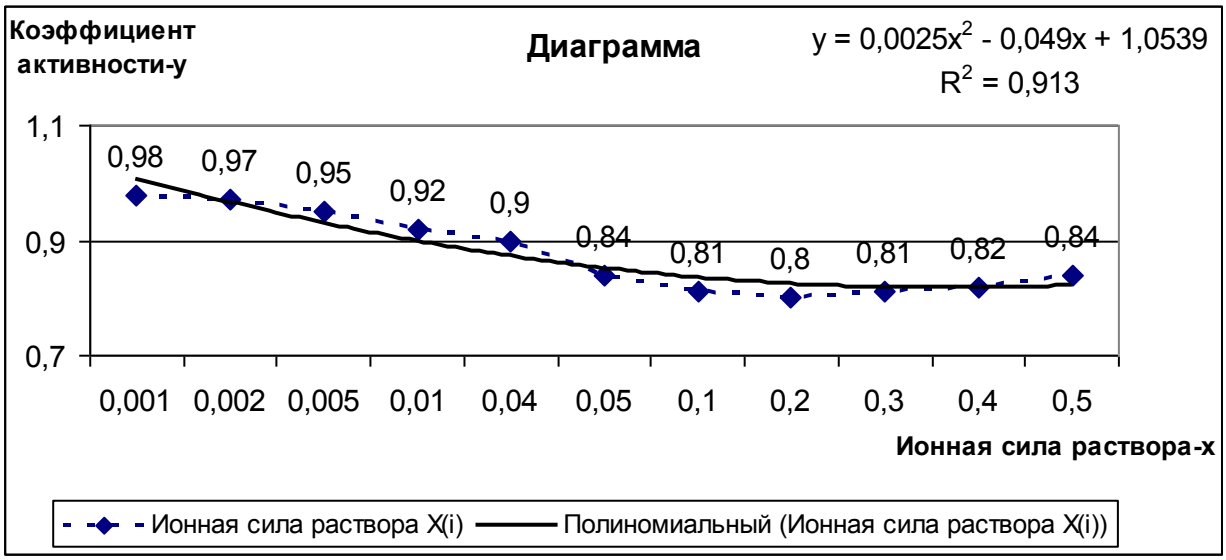
РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ



```
C:\Basic\QB.EXE
Введите Коэффициент Стьюдента=? 2.201
 N            x             y             tx            ty
 1            .001          .98           -.8126461      1.484493
 2            .002          .97           -.8070487      1.346108
 3            .005          .95           -.7902563      1.069338
 4            .01           .92           -.7622691       .6541831
 5            .04           .9            -.594346        .3774125
 6            .05           .84           -.5383717      -.4528977
 7            .1            .81           -.2584998      -.8680524
 8            .2            .8             .3012439      -1.006437
 9            .3            .8             .8609876      -1.006437
 10           .4            .81           1.420731       -.8680524
 11           .5            .82           1.980475       -.7296675
число ответов = 11
табличное значение критерия Стьюдента = 2.201
математическое ожидание :по x= .1461818    по y= .8727273
дисперсия :по x= 3.191696E-02               по y= 5.221819E-03
среднеквадратическое отклонение :по x= .1786532          по y= 7.226215E-02
коэффициент вариации :по x= 122.213         по y= 8.280038
коэффициент корреляции =-.7240622
рассчитанное значение критерия Стьюдента = 3.149307
линейная зависимость между x и y Существует


Press any key to continue
```

ПРИМЕР РАБОТЫ в EXCEL

	A	B	C	D	E
1	**№ п/п**	**Ионная сила раствора X(i)**	**Коэффициент активности Y(i)**	**нормир. откл. по x(i)**	**нормир. откл. по y(i)**
2	1	0,001	0,98	-0,81265	-0,81265
3	2	0,002	0,97	-0,80705	-0,80705
4	3	0,005	0,95	-0,79026	-0,79026
5	4	0,01	0,92	-0,76227	-0,76227
6	5	0,04	0,9	-0,59435	-0,59435
7	6	0,05	0,84	-0,53837	-0,53837
8	7	0,1	0,81	-0,2585	-0,2585
9	8	0,2	0,8	0,301244	0,301244
10	9	0,3	0,81	0,860988	0,860988
11	10	0,4	0,82	1,420731	1,420731
12	11	0,5	0,84	1,980475	1,980475
13	Матем ожидание по x =			0,146182	
14	Матем ожидание по y =			0,876363636	
15	Дисперсия по x =			0,031917	
16	Дисперсия по y =			0,004785455	
17	Среднеквадратичное отклонение по x=			0,178653	
18	Среднеквадратичное отклонение по y=			0,06917698	
19	Коэффициент корреляции =			-0,66611	
20	Критерий Стьюдента =			2,4735	
21					



Результаты эксперимента

Контрольные вопросы

«Статистическая обработка результатов эксперимента»

1. Что характеризует математическое ожидание?
2. Какие характеристики оценивают разброс случайных величин?
3. Как определяется дисперсия?
4. Для чего необходим коэффициент вариации?
5. Как определить нормированное отклонение?
6. Что характеризует коэффициент корреляции?
7. В каких пределах измеряется коэффициент корреляции?
8. Как определить уровень значимости коэффициента корреляции?
9. Как выбирается $T_{табл}$?
10. Какое соотношение должно быть между $T_{расч}$ и $T_{табл}$, для утверждения, что линейная зависимость существует?

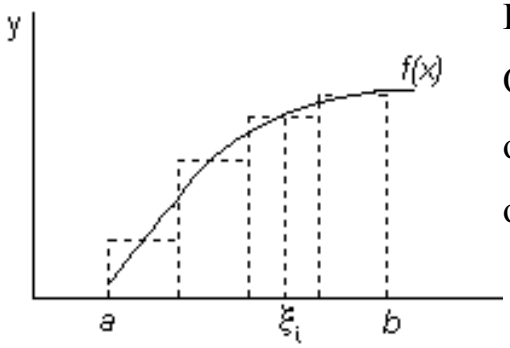
ЛАБОРАТОРНАЯ РАБОТА № 2

«Численное интегрирование»

1. Цель работы.

Ознакомится с принципом модульного программирования на примере задачи численного интегрирования. Использование оболочки QBASIC для построения процедур программ.

2. Основные теоретические сведения.



Пусть на отрезке [a,b] задана функция f(x). Определенный интеграл определяется как площадь, ограниченная подынтегральной функцией f(x), осью x и ординатами в точках «a» и «b»

Определенным интегралом от функции f(x) на отрезке [a, b] называется предел интегральной суммы при неограниченном увеличении числа точек разбиения.

$$\int_a^b f(x)dx = \lim_{\max \Delta x_i \to 0} \sum_{i=0}^{n} f(\xi_i)\Delta x_i$$

Во многих случаях, когда подынтегральная функция задана в аналитическом виде, определенный интеграл удается вычислить непосредственно по формуле Ньютона-Лейбница. Она состоит в том, что определенный интеграл равен приращению первообразной F(x) на отрезке интегрирования. На практике этой формулой часто нельзя воспользоваться по двум основным причинам:

Вид функции не допускает непосредственного интегрирования, т.е. первообразную нельзя выразить в элементарных функциях

Значения функций f(x) заданы таблично (множество x_i конечно)

В этих случаях используются методы численного интегрирования.

Частным случаем в методах численного интегрирования является тот, когда величина элементарного отрезка Δx,- величина постоянная и может быть вынесена за знак интегральной суммы. Эта величина называется шагом интегрирования и обозначается обычно Δx.

Рассмотрим методы численного интегрирования.

1). Метод прямоугольников

<u>В Методе прямоугольников</u> непосредственно используется замена определенного интеграла интегральной суммой. В качестве точек x_i; могут

выбираться левые (x_{i-1}) или правые (x_i) границы элементарных отрезков. Расчетные формулы можно записать так:

При выборе левых границ (см. рис.1)

$$\int_a^b f(x)dx \approx h(y(a)+y(a+h)+y(a+2h)+.....+y(b-h)=h\sum_{x=a+h}^{b} y(x)$$

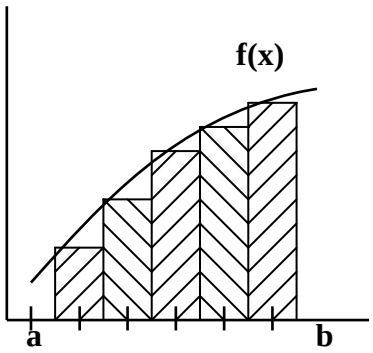


Рис.2

При выборе правых границ (см. рис.2)

$$\int_a^b f(x)dx \approx h(y(a+h)+y(a+2h)+.....+y(b)=h\sum_{x=a+h}^{b} y(x)$$

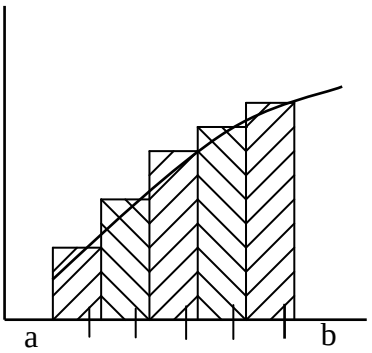


Рис.3

При выборе границ от a+ h/2 до b-h/2

$$\int_a^b f(x)dx \approx h(y(a+\frac{h}{2})+y(a+3\frac{h}{2})-y(b-\frac{h}{2})=h\sum_{x=a+\frac{h}{2}}^{b-\frac{h}{2}} y(x)$$

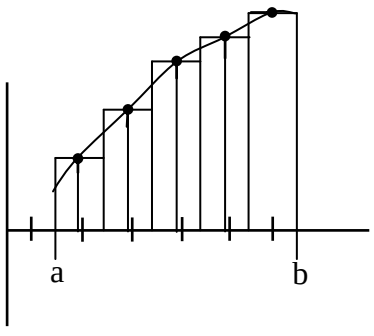
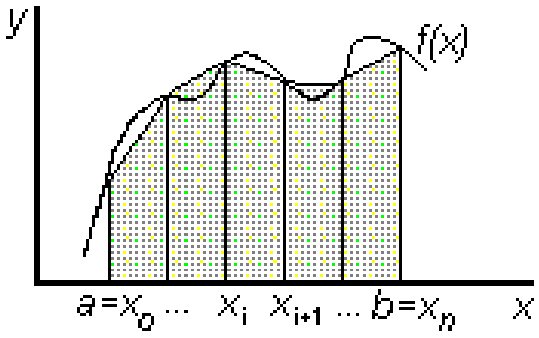


Рис.4

2) Метод трапеций



<u>В методе трапеций</u> график функции f(x) аппроксимируется ломаной, соединяющей точки с координатами (x_i, y)

Рис.5

Искомое значение определенного интеграла представляется в виде суммы площадей трапеций, построенных на каждом из элементарных отрезков:

$$\int_a^b f(x)dx \approx \frac{f(a)+f(a+h)}{2}\cdot h + \frac{f(a+h)+f(a+2h)}{2}\cdot h + \frac{f(b-h)+f(b)}{2}\cdot h =$$

$$= h\cdot\left(\frac{f(a)+f(b)}{2} + \sum_{x=a+h}^{b-h} f(x)\right)$$

3) Метод парабол

<u>В методе парабол</u> (формула Симпсона) на каждом из элементарных отрезков по трем известным значениям функции f(Xj) строится парабола, заданная уравнением ax^2+bx+c.

Формула для нахождения определенного интеграла может быть выведена из условия равенства значений: $y_i = ax_i^2 + bx_i + c$:

$$\int_a^b f(x)dx \approx \frac{h}{3}\cdot\left(y_0 + y_n + 4\cdot\left(y_1 + y_3 + \ldots + y_{n-2}\right) + 2\left(y_2 + y_4 + y_6 + \ldots\ldots + y_{n-1}\right)\right) =$$

$$= \frac{1}{3}\cdot\left(y(a) + y(b) + 4\cdot\sum_{\substack{x=a+h \\ шаг=2h}}^{b-2h} f(x) + \sum_{\substack{x=a+h \\ шаг=2h}}^{b-h} f(x)\right)$$

3. Порядок выполнения работы

3.1. Получить у преподавателя вариант задания, включающий в себя подынтегральную функцию (F(X)), отрезок интегрирования (a,b), точность вычисления значения интеграла (eps).

3.2. Исследовать подынтегральную функцию на непрерывность и существование на заданном отрезке.

3.3. Составить блок-схему для каждого метода и блок-схему головного модуля.

3.4. Написать подпрограмму для каждого метода (прямоугольников, трапеции, парабол).

3.5. Написать головной модуль.

3.6. Отладить программу и получить результаты .

3.7. Проанализировать полученные результаты и сделать выводы.

4. Содержание отчета.

4.1. Математическая постановка задачи.

4.2. Исходные данные.

4.3. Краткое описание методов. Блок-схема для каждого метода. Листинг подпрограмм.

4.4. Блок-схема головного (или управляющего) модуля. Листинг.

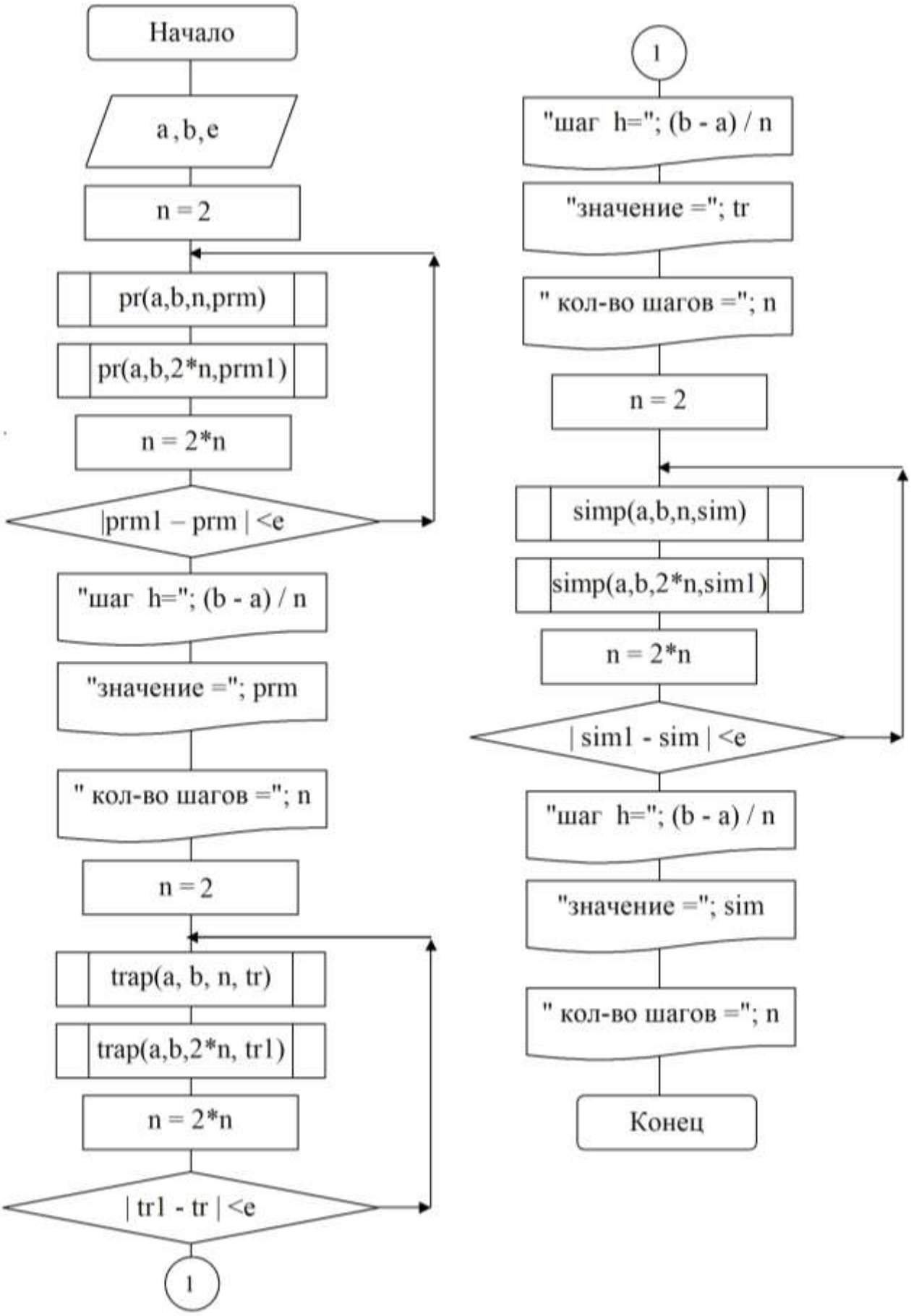
4.5. Распечатка полученных результатов.

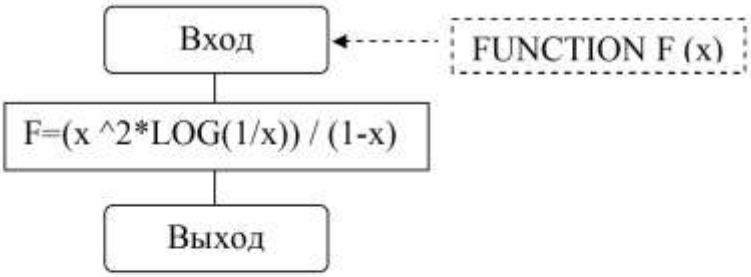
4.6. Сравнительный анализ полученных результатов разными методами.

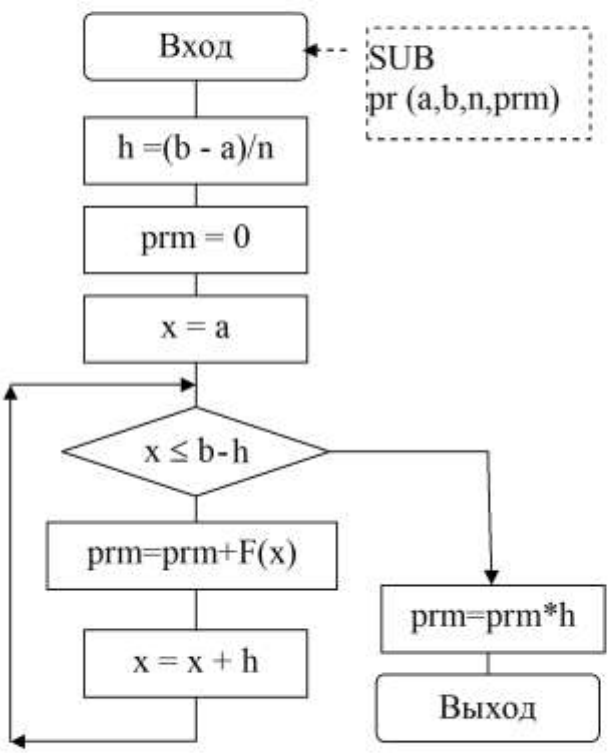
Пример выполнения работы

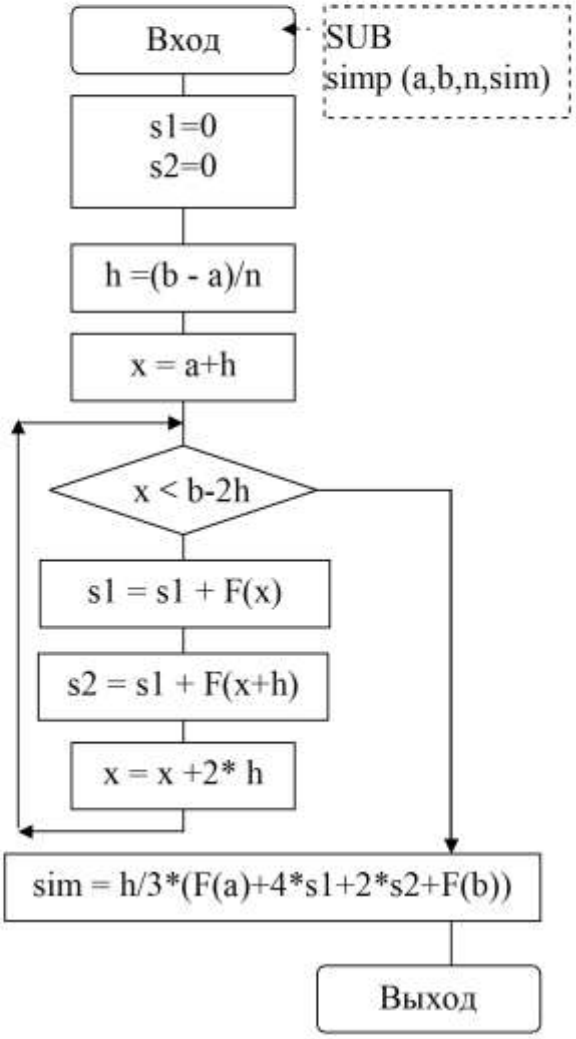
Вычислить интеграл $z = \int_3^4 x^2 \frac{\ln\left(\frac{1}{x}\right)}{1-x} dx$

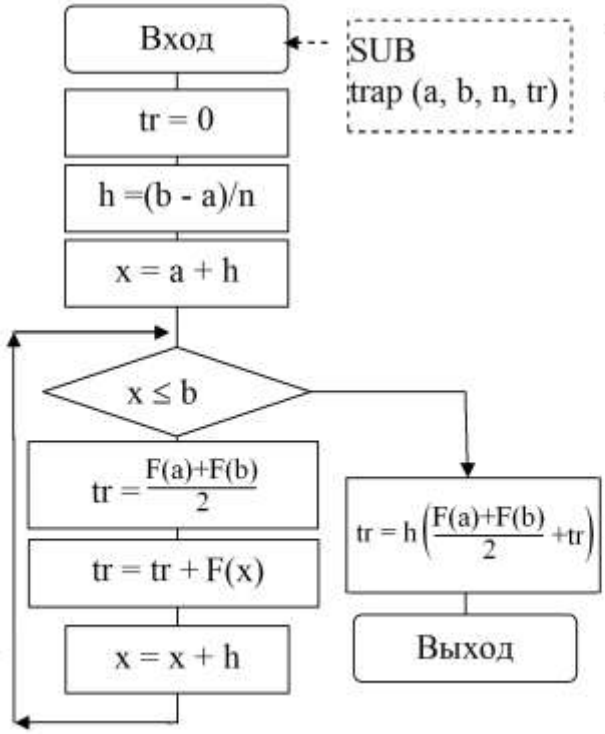
БЛОК-СХЕМА


Начало
a , b, e
n = 2
pr(a,b,n,prm)
pr(a,b,2*n,prm1)
n = 2*n
|prm1 – prm | <e
"шаг h="; (b - a) / n
"значение ="; prm
" кол-во шагов ="; n
n = 2
trap(a, b, n, tr)
trap(a,b,2*n, tr1)
n = 2*n
| tr1 - tr | <e
1
1
"шаг h="; (b - a) / n
"значение ="; tr
" кол-во шагов ="; n
n = 2
simp(a,b,n,sim)
simp(a,b,2*n,sim1)
n = 2*n
| sim1 - sim | <e
"шаг h="; (b - a) / n
"значение ="; sim
" кол-во шагов ="; n
Конец


Вход
FUNCTION F (x)
F=(x ^2*LOG(1/x)) / (1-x)
Выход


Вход
SUB
pr (a,b,n,prm)
h =(b - a)/n
prm = 0
x = a
x ≤ b-h
prm=prm+F(x)
x = x + h
prm=prm*h
Выход


Вход
SUB
simp (a,b,n,sim)
s1=0
s2=0
h =(b - a)/n
x = a+h
x < b-2h
s1 = s1 + F(x)
s2 = s1 + F(x+h)
x = x +2* h
sim = h/3*(F(a)+4*s1+2*s2+F(b))
Выход


Вход
SUB
trap (a, b, n, tr)
tr = 0
h =(b - a)/n
x = a + h
x ≤ b
tr = F(a)+F(b) / 2
tr = tr + F(x)
x = x + h
tr = h(F(a)+F(b) / 2 +tr)
Выход

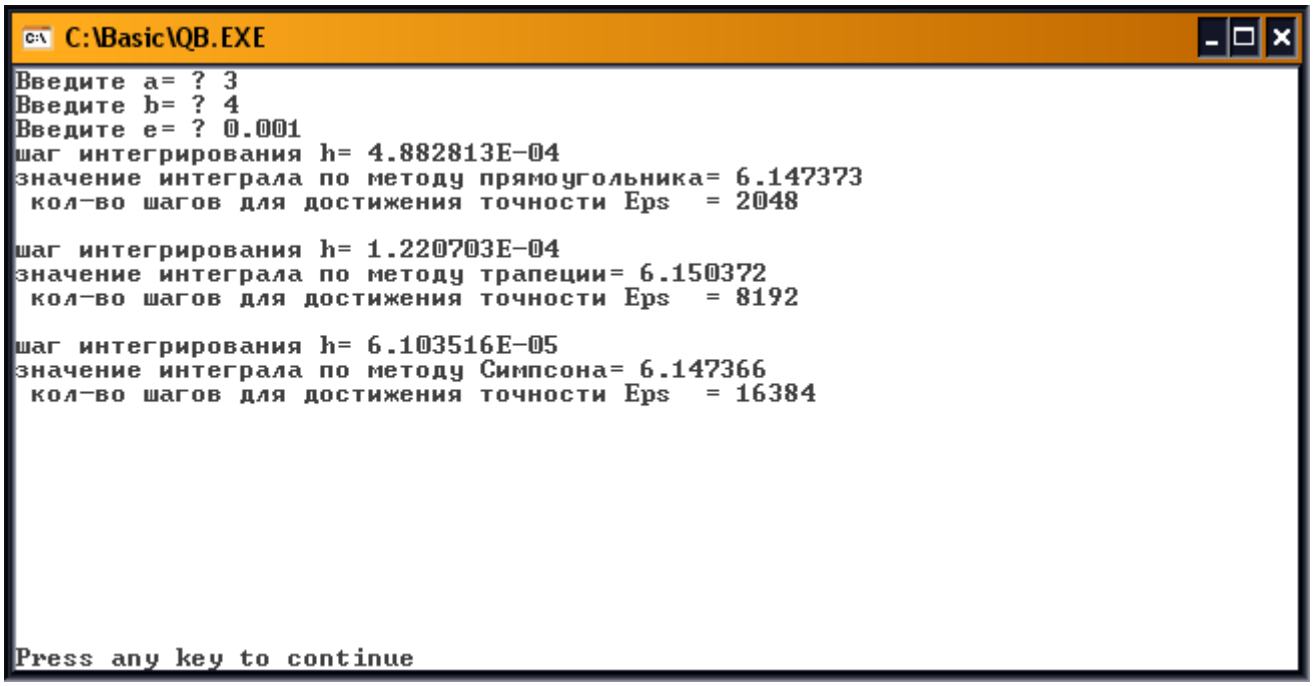
ВИД ПРОГРАММЫ НА ЯЗЫКЕ QBASIC

```
DECLARE SUB simp (a!, b!, n!, sim!)
DECLARE SUB trap (a!, b!, n!, tr!)
DECLARE SUB pr (a!, b!, n!, prm!)
DECLARE FUNCTION F! (x!)
CLS
INPUT "Введите a= "; a
INPUT "Введите b= "; b
INPUT "Введите e= "; e
n = 2
DO
CALL pr(a, b, n, prm)
CALL pr(a, b, 2 * n, prm1)
n = 2 * n
LOOP UNTIL ABS(prm1 - prm) < e
PRINT "шаг интегрирования h="; (b - a) / n
PRINT "значение интеграла по методу прямоугольника="; prm
PRINT " кол-во шагов для достижения точности Eps  ="; n
PRINT
n = 2
DO
CALL trap(a, b, n, tr)
CALL trap(a, b, 2 * n, tr1)
n = 2 * n
LOOP UNTIL ABS(tr1 - tr) < e
PRINT "шаг интегрирования h="; (b - a) / n
PRINT "значение интеграла по методу трапеции="; tr
PRINT " кол-во шагов для достижения точности Eps  ="; n
PRINT
n = 2
```

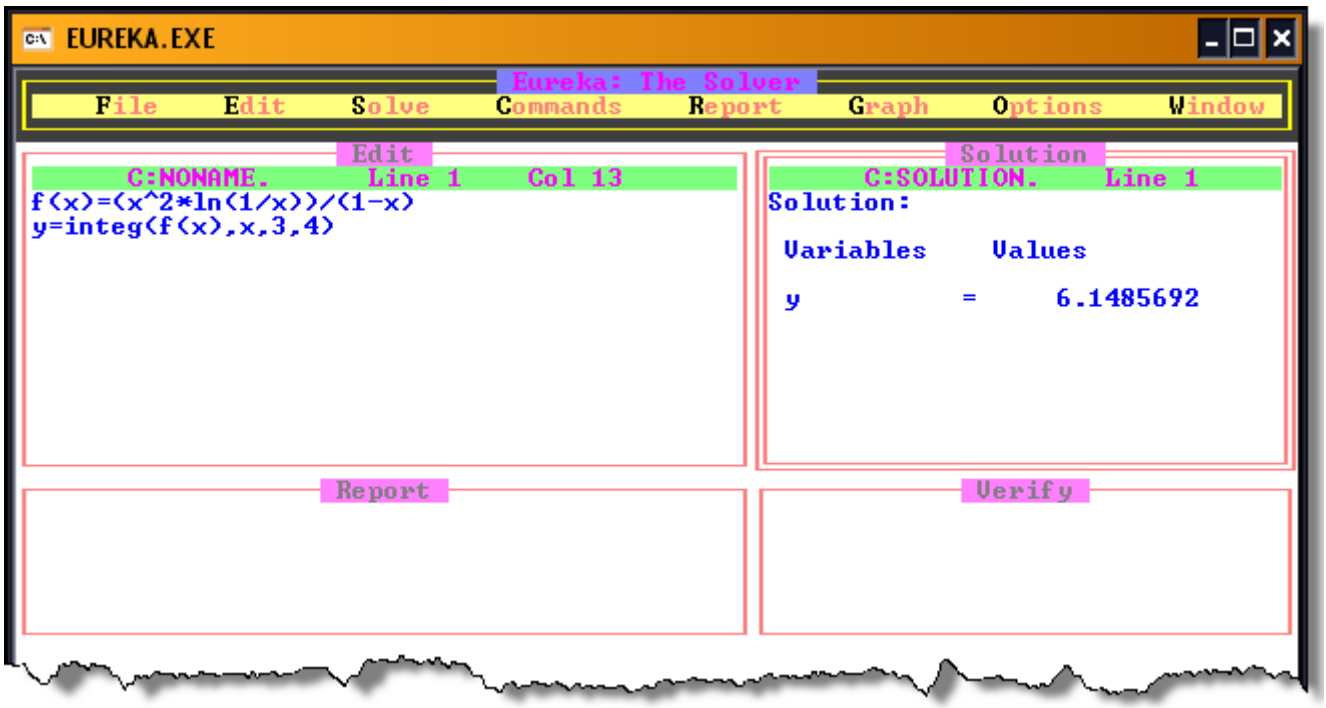
```
DO
CALL simp(a, b, n, sim)
CALL simp(a, b, 2 * n, sim1)
n = 2 * n
LOOP UNTIL ABS(sim1 - sim) < e
PRINT "шаг интегрирования h="; (b - a) / n
PRINT "значение интеграла по методу Симпсона="; sim
PRINT " кол-во шагов для достижения точности Eps  ="; n
END

FUNCTION F (x)
F = (x ^ 2 * LOG(1 / x)) / (1 - x)
END FUNCTION

SUB pr (a, b, n, prm)
h = (b - a) / n
prm = 0
FOR x = a TO b - h STEP h
prm = prm + F(x)
NEXT x
prm = prm * h
END SUB

SUB simp (a, b, n, sim)
s1 = 0: s2 = 0
h = (b - a) / n
FOR x = a + h TO b - 2 * h STEP 2 * h
s1 = s1 + F(x)
s2 = s2 + F(x + h)
NEXT x
```

```
sim = h * (F(a) + 4 * s1 + 2 * s2 + F(b)) / 3
END SUB

SUB trap (a, b, n, tr)
tr = 0
h = (b - a) / n
FOR x = a + h TO b STEP h
tr = tr + F(x)
NEXT x
tr = h * ((F(a) + F(b)) / 2 + tr)
END SUB
```

РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ В Qbasic



Результат расчета в ППП ЭВРИКА.



Вывод: Значение интеграла, вычисление всеми способами достаточно близки.

Методические указания к выполнению лабораторной работы на ПК

1. Подынтегральную функцию варианта задания оформляем как подпрограмму функцию, используя в меню оболочки QuickBasic режим:

FUNCTION F(X)

F= < функция соответствующего варианта >

END FUNCTION

Запись всех подпрограмм можно осуществить через меню оболочки QuickBasic:

1. Alt - вход в меню
2. EDIT - всплывающее меню редактирования
3. NEW SUB - создание новой подпрограммы

(NEW FUNCTION -создание новой подпрограммы функции)

4. Набираем в диалоговом окне новое имя подпрограммы, например: INTT
5. На экране появляется заготовка для создания подпрограммы:

SUB <имя подпрограммы>

END SUB

6. Приступаем к написанию подпрограммы между ключевыми словами SUB и END SUB

Все вспомогательные подпрограммы объединяются управляющей программой или головным модулем. Переход от текста управляющей программы к текстам подпрограмм происходит при нажатии клавиш ALT + F1, наоборот - ALT + F2

Контрольные вопросы

1. Когда используются процедуры?
2. Как создаются подпрограммы, процедуры?
3. Что такое управляющий модуль?
4. Как просмотреть все присоединённые модули?
5. Где объявляются присоединённые подпрограммы? Каким оператором?
6. Какие параметры называются фактические?
7. Какие параметры называются формальные?
8. Как передаются данные из подпрограммы в программу и наоборот?
9. Чем отличаются задачи на интегрирование с заданным числом разбиения отрезка от задачи с заданной точностью вычисления?
10. В чем заключаются численные методы интегрирования?
11. Как реализуется один из методов (по выбору) на QBasic?
12. Как определить погрешность метода?
13. Как осуществляется интегрирование с автоматическим методом выбором шага интегрирования?

Варианты заданий для самостоятельного решения

Задание

Вычислить интеграл тремя методами: прямоугольников, трапеций и методом парабол (Симпсона), используя автоматический выбор шага интегрирования.

Точность вычислений $\varepsilon = 10^{-4}$.

Таблица заданий № 1.

№ п/п	Уравнение	№ п/п	Уравнение	№ п/п	Уравнение
1	$\int_{-0,5}^{1,3} \frac{x^2 dx}{\sqrt{x^2+1}}$	**11**	$\int_{-0,8}^{1,4} \frac{x^2 dx}{\sqrt{x^2+4}}$	**21**	$\int_{-2,5}^{-1,3} \frac{xdx}{\sqrt{x^2+1,8}}$
2	$\int_{2}^{3,2} \frac{x+2}{\sqrt{x^2+1}} dx$	**12**	$\int_{2,6}^{3,4} \frac{x+0,5}{\sqrt{x^2+1,5}} dx$	**22**	$\int_{-0,4}^{1,8} \frac{x^2+2}{\sqrt{x^2+1}} dx$
3	$\int_{0,5}^{1,6} \frac{x^2+0,5}{\sqrt{x^2+1}} dx$	**13**	$\int_{0,8}^{2} \frac{xdx}{\sqrt{x^2+2}}$	**23**	$\int_{0,6}^{2} \frac{x^2 dx}{\sqrt{x^2+2}}$
4	$\int_{2,2}^{3,4} \frac{x^2 dx}{\sqrt{x+1}}$	**14**	$\int_{2,4}^{3,2} \frac{x^2 dx}{\sqrt{x+2}}$	**24**	$\int_{1,6}^{2,8} \frac{x^2 dx}{\sqrt{x^2+1,2}}$
5	$\int_{1,2}^{2} \frac{x-0,5}{\sqrt{x^2-1}} dx$	**15**	$\int_{0,2}^{2} \frac{x+0,5}{\sqrt{x^2+1}} dx$	**25**	$\int_{0,2}^{1,11} \frac{\sqrt{x^2+1}}{2x+2,5} dx$
6	$\int_{2,2}^{3,8} \frac{x+1}{\sqrt{x^2+2}} dx$	**16**	$\int_{0,7}^{1,5} \frac{x+2}{\sqrt{x^2+1}} dx$	**26**	$\int_{0,6}^{1,8} \frac{x^2 dx}{\sqrt{x+1,7}}$
7	$\int_{0,2}^{2,4} \frac{\sqrt{x^2+1}}{x+2} dx$	**17**	$\int_{0,2}^{2,5} \frac{\sqrt{x^2+2}}{x+2} dx$	**27**	$\int_{0,4}^{1,8} \frac{x^2+1,4}{\sqrt{x^2+0,2}} dx$
8	$\int_{1}^{2,6} \frac{xdx}{\sqrt{x^2+3}}$	**18**	$\int_{1,4}^{2,6} \frac{xdx}{\sqrt{x^2+2,5}}$	**28**	$\int_{2,2}^{2,8} \frac{(4-x)dx}{\sqrt{x^2+1}}$
9	$\int_{0,8}^{1,6} \frac{0,5x+2}{\sqrt{x^2+1}} dx$	**19**	$\int_{2,2}^{3,4} \frac{xdx}{\sqrt{x^2+1}}$	**29**	$\int_{1}^{2} \frac{dx}{x^2}$
10	$\int_{-0,4}^{1,6} \frac{x+1}{\sqrt{x^2+1}} dx$	**20**	$\int_{0,4}^{1,6} \frac{x+3}{\sqrt{x^2+1}} dx$	**30**	$\int_{0}^{1} \frac{dx}{1+x^2}$

Таблица заданий № 2.

№ п/п	Уравнение	№ п/п	Уравнение	№ п/п	Уравнение
1	$\int_{0,8}^{1,6} \frac{dx}{\sqrt{2x^2+1}}$	**11**	$\int_{2}^{3,5} \frac{dx}{\sqrt{x^2-1}}$	**21**	$\int_{0,8}^{1,7} \frac{dx}{\sqrt{2x^2+0,3}}$

2	$\int_{1,2}^{2,7} \frac{dx}{\sqrt{x^2+3,2}}$	**12**	$\int_{0,5}^{1,3} \frac{dx}{\sqrt{x^2+2}}$	**22**	$\int_{1,2}^{2} \frac{dx}{\sqrt{0,5x^2+1,5}}$
3	$\int_{1}^{2} \frac{dx}{\sqrt{2x^2+1,3}}$	**13**	$\int_{2,2}^{2,6} \frac{dx}{\sqrt{x^2+0,6}}$	**23**	$\int_{2,1}^{3,6} \frac{dx}{\sqrt{x^2-3}}$
4	$\int_{0,2}^{1,2} \frac{dx}{\sqrt{x^2+1}}$	**14**	$\int_{1,4}^{2,2} \frac{dx}{\sqrt{3x^2+1}}$	**24**	$\int_{1,3}^{2,5} \frac{dx}{\sqrt{0,2x^2+1}}$
5	$\int_{0,8}^{1,4} \frac{dx}{\sqrt{2x^2+3}}$	**15**	$\int_{0,8}^{1,8} \frac{dx}{\sqrt{x^2+4}}$	**25**	$\int_{0,6}^{1,4} \frac{dx}{\sqrt{1,2x^2+0,5}}$
6	$\int_{0,4}^{1,2} \frac{dx}{\sqrt{2+0,5x^2}}$	**16**	$\int_{1,6}^{2,2} \frac{dx}{\sqrt{x^2+2,5}}$	**26**	$\int_{1,3}^{2,1} \frac{dx}{\sqrt{3x^2-0,4}}$
7	$\int_{1,4}^{2,1} \frac{dx}{\sqrt{3x^2-1}}$	**17**	$\int_{0,6}^{1,6} \frac{dx}{\sqrt{x^2+0,8}}$	**27**	$\int_{1,4}^{2,6} \frac{dx}{\sqrt{1,5x^2+0,7}}$
8	$\int_{1,2}^{2,4} \frac{dx}{\sqrt{0,5+x^2}}$	**18**	$\int_{1,2}^{2} \frac{dx}{\sqrt{x^2+1,2}}$	**28**	$\int_{2}^{3} \frac{dx}{\sqrt{4x^2+0,7}}$
9	$\int_{0,4}^{1,2} \frac{dx}{\sqrt{3+x^2}}$	**19**	$\int_{1,4}^{2} \frac{dx}{\sqrt{2x^2+0,7}}$	**29**	$\int_{2}^{4} \frac{dx}{\sqrt{4x^2+6}}$
10	$\int_{0,6}^{1,5} \frac{dx}{\sqrt{1+2x^2}}$	**20**	$\int_{3,2}^{4} \frac{dx}{\sqrt{0,5x^2+1}}$	**30**	$\int_{1}^{2} \frac{dx}{\sqrt{3x^2+4}}$

ЛАБОРАТОРНАЯ РАБОТА № 3

«Уточнение корня уравнения»

1. Цель работы

Получить навыки модульного программирования на примере задачи численного решения нелинейных уравнений. Использование оболочки QBasic для построения программ и головного модуля.

2. Основные теоретические положения

Многие задачи исследования различных объектов с помощью математических моделей, применяемых их для прогноза или расчёта, приводят к необходимости решения нелинейных уравнений.

Уравнения могут быть алгебраическими и трансцендентными. Пример алгебраического уравнения: $y = a + bx + cx^2$, трансцендентного: $y = e^n + x$.

Решить уравнение – это найти такое значение переменной x, при котором заданная функция равна нулю (f (x) = 0).

Как правило, процесс решения нелинейного уравнения общего вида f(x)=0 осуществляется в два этапа. На первом этапе отделяют корни, т.е. находят такие отрезки, внутри которых находится строго один корень. На втором этапе уточняют корень, т.е. находят его значение x* с предварительно заданной точностью ε. В практических задачах решением называют любое значение x, отличающееся по модулю от точного значения x* не более чем на величину ε.

Рассмотрим следующие методы уточнения корня уравнения:

- метод дихотомии;
- метод касательных;
- метод простой итерации;
- метод хорд.

1). Метод дихотомии

Пусть функция f (x) отрицательна в точке a (f (a) < 0), положительна в точке b (f (b) > 0), и непрерывна на отрезке [a, b] график функции пересекает ось X, т. е. на этом отрезке имеется корень уравнения – точка, в которой f(x)= 0.

Тот же вывод следует, если f(a) > 0, f(b) < 0. В общем виде это формулируется так: в точках a и b функция f(x) принимает значения разных знаков. Если нам известен хотя бы один такой отрезок, пусть и большой длины, мы можем построить процедуру быстрого и сколь угодно точного поиска корня уравнения. Найдем значение функции в точке c, находящейся в середине отрезка: c =(a + b)/2. Знак f(c) совпадает со знаком функции на одном из концов отрезка и противоположен знаку функции на другом конце.

Пусть разные знаки f(x) в точках a и c. Значит на [a, c] наверняка есть искомый корень уравнения.

Таким образом, мы получили задачу, эквивалентную исходной, но теперь длина отрезка, на котором, как нам известно, находится корень, в два раза короче. Отрезок [a, c] опять можно разделить пополам и оставить в рассмотрении только один из двух получившихся (учитывая знаки значений f(x) на концах этих отрезков). Выбранный отрезок вновь разделить, и продолжать так до тех пор, пока отрезок, на котором находится корень, не станет достаточно мал.

Как написать программу на QuickBasic, соответствующую этому методу?

Прежде всего, функцию, корень которой мы ищем, лучше всего описать в виде отдельной function. Во время выполнения программа должна запросить начальные значения a и b.

Далее нужно проверить, что f(a) и f(b) действительно разных знаков.
Это условие можно записать двумя разными способами:

1. ((f (a) <= 0) and (f (b) > 0)) or ((f (b) <= 0) and (f (a) > 0))
2. f (a) * f(b) <= 0

Если это не так, нужно только напечатать сообщение об ошибке, иначе можно приступать к поиску корня. Поиск проще всего оформить в виде цикла:

Do until (b-a) < eps ... loop

Здесь eps - константа, равная, например 10^{-5}. На каждой итерации цикла нужно вычислить значение точки c и сравнить знак функции в этой точке со знаком f(a). Если знаки разные, о точке b можно "забыть": b:=c. Иначе – "забываем" о точке a. После окончания цикла остается лишь напечатать найденный

корень. Ближе всего к корню будет, середина окончательного отрезка [a,d], длина которого не превышает заданной точности eps.

2). Метод касательных

Метод Ньютона, или метод касательных основан на вычислении не только значений функции, но и значений ее производной. Разложим f(x) в ряд Тейлора и отбросим члены ряда выше второго порядка:

$f(x) = f(x0) + f'(x_0)(x - x_0)$

В этом приближении корень функции легко найти по формуле:

$f(x) = 0$, при $x_i = x_{i+1} - f(x_0)/f'(x_0)$

Для нелинейных функций это, конечно, не точное равенство, но способ получить значение, более близкое к корню. Таким образом, метод Ньютона состоит в построении последовательности итераций:

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

$$x_2 = x_1 - f(x_1)/f'(x_1)$$

$$x_{i+1} = x_i - f(x_i)/f'(x_i)$$

На какой итерации завершать процесс нахождения корня? Возможно два варианта: либо когда значение функции станет достаточно маленьким по абсолютной величине:

$|f(x)| < eps$, либо когда практически перестанет меняться значение x:

$$|f(x)/f'(x)| < eps$$

$$|X_{i+1} - X_i| <= eps$$

3). Метод простой итерации

Рассматриваемый метод реализует третий подход к решению задачи. Предварительно исходное уравнение $f(x) = 0$ преобразуют к виду $\varphi(x)=x$, что является частным случаем более общей структуры $g(x)=f(x)$.Затем выбирают начальное значение x_0 и подставляют его в левую часть уравнения, но $\varphi(x)\neq x_0$, поскольку x_0 взято произвольно и не является корнем уравнения.

Полученное $\varphi(x)=x_1$ рассматривают как очередное приближение к корню. Его снова подставляют в правую часть уравнения $\varphi(x_1)$ и получают следующее значение x_2 ($x_2 = \varphi(x_1)$ и т.д., в общем случае $x_{i+1} =\varphi(x_i)$. Получающаяся таким

образом последовательность $x_0, x_1, x_2, x_3, x_4,\ldots$ при определенных условиях может сходиться к корню x^* (рис. 1.а).

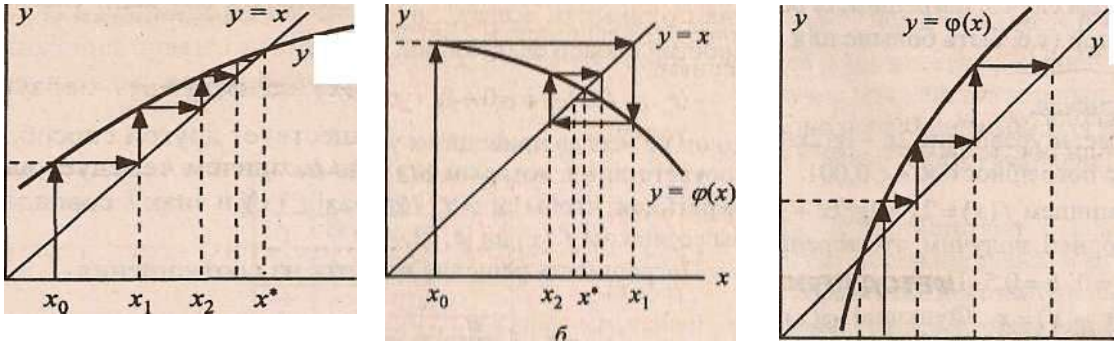
Условие сходимости $|\varphi'(x)| <= 1$ на [a,b], причем, чем ближе модуль к нулю, тем выше окажется скорость сходимости к решению. В противном случае последовательность расходится от искомого решения (“метод не сходится”).

На рисунке 2 приведён один из возможных случаев, когда итерационный процесс не сходится. Видно, что последовательность $x_0, x_1, x_2,\ldots$ удаляется от корня x^*. Это всегда будет иметь место в том случае, если тангенс угла наклона $\varphi(x)$ в окрестности корня по модулю больше единицы.

Существуют различные способы преобразования уравнения $f(x) = 0$ к виду $\varphi(x) = x$; одни могут привести к выполнению условия сходимости всегда, другие – в отдельных случаях.

Самый простой способ следующий:

$$f(x) + x = 0 + x,\ f(x) + x = \varphi(x)$$



(а) (б) (в)

а) $0 < \varphi'(x) < 1$; б) $-1 < \varphi'(x) < 0$

Рис.6

но он не всегда приводит к успеху. Существует другой способ, в соответствии с которым $\varphi(x)=x-f(x)/к$, причем k следует выбирать так, чтобы $|k| >Q/2$, где $Q=\max |f'(x)|$ и знак к совпадал бы со знаком $f'(x)$ на [a, b].

Погрешность решения можно оценить из соотношения

$|x^* - x_i| <= (q/(1 – q))*|X(i) – X(i+1)|$, где $q = \max \varphi(x)$, на отрезке [a,b].

Вследствие этого для окончания вычислений в методе итераций применят соотношение (q/(q – 1))*|X(i) – X(i + 1)|<= ε, где ε — заданная погрешность решения.

Часто используют упрощенное условие окончания поиска |X(i) – X(i + 1)|<=ε не вычисляя максимальное значение производной, но в этом случае погрешность решения может не соответствовать заданной (т.е. быть больше или меньше).

4). Метод хорд

В этом методе нелинейная функция f(x) на отделенном интервале [a, b] заменяется линейной, в качестве которой берется хорда — прямая, стягивающая концы нелинейной функции. Эта хорда определяется как прямая, проходящая через точки с координатами (a, f(a)) и (b,f(b)). Имея уравнение хорды y = cx + d, можно легко найти точку ее пересечения с горизонтальной осью, подставив в уравнение y = 0 и найдя из него x. Естественно, в полученной таким путем точке x1 не будет решения, ее принимают за новую границу отрезка, где содержится корень. Через эту точку с координатами (x1, f(x1)) и соответствующую границу предыдущего интервала опять проводят хорду, находят x_2 и т.д. несколько раз, получая последовательностьx3 , x4 ,x5,..., сходящуюся к корню.

Метод хорд применим только для монотонных функций.

Алгоритм метода зависит от свойств функции f(x). Если f(b)*f ”(b) > 0, то строящаяся на каждом этапе хорда имеет правый фиксированный ("закрепленный") конец, и алгоритм выглядит следующим образом:

xi+1=xi-(f(xi)/(f(b)-f(xi)))*(b-xi);

при этом последовательность x1 ,x2,,…будет приближаться к корню слева.

Если f(a)f"(a) > 0, то строящаяся на каждом этапе хорда имеет левый фиксированный ("закрепленный") конец, и алгоритм выглядит следующим образом: xi+1 = a +(f(a) / (f(a) – f(xi)))*(xi – a);

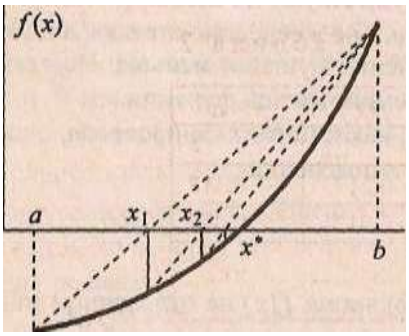


Рис.7

при этом последовательность $x_1, x_2, \ldots$ будет приближаться к корню справа.

На рисунке 7 приведен один из вариантов применения метода хорд. В рассматриваемом случае "закрепленным" является правый конец. Приведено пять шагов (пять хорд), при этом к решению приближаемся слева.

Теоретически доказано, что если первые производные на концах интервала при монотонной и выпуклой функции f(x) не различаются более чем в 2 раза, то справедливо соотношение $|x^* - x_i| < |x_i - x_{i-1}|$ и условием прекращения пополнения последовательности может быть $|x_{i+1} - x_i| <= \varepsilon$, а в качестве корня принято x_{i+1} (можно также окончить процесс и при достижении $f(x_i) <= \delta$, о чем указывалось в концепции методов). На практике указанные условия можно применять и без предварительной проверки производных, отклонение погрешности результата при пологих функциях не будет существенным.

3. Порядок выполнения работы

1. Получить у преподавателя вариант задания, включающий в себя трансцендентное или алгебраическое уравнение (F(x) = 0), отрезок для поиска решения (a, b), точность вычисления значения корня (eps) и задание, каким из методов (все методы) решить задачу.

2. Исследовать существование корня на заданном отрезке, требования к функции: разные знаки на концах отрезка, непрерывность).

3. Выяснить, с какой стороны отрезка строить приближение к решению уравнения: если знак второй производной совпадает со знаком функции (f(x)- f"(x) > 0), то приближение строим елевой стороны отрезка, в противном случае - с левой стороны.

4. Написать подпрограмму для вычисления функции F(x).

5. Написать подпрограмму, например для первого метода дихотомии.

6. Написать подпрограмму, например, для второго метода касательных или простой итерации.

7. Найти первую производную функции F(x) и оформить её вычисление процедурой функции (для метода касательных).

8. Найти приведённую функцию и оформить её вычисление процедурой функции.

9. Написать головной модуль.

10. Отладить программу и получить результат.

Пример выполнения лабораторной работы.

Уравнение	Отрезок	Точность
y = x - 2 + SIN(1 / x)	(1; 3)	10^{-3}

Решить уравнение методом дихотомии, простой итерации, методом касательных.

a=1

b=3

F(x) = x - 2 + SIN(1 / x) - исходная функция

G (x) = 2 - SIN(1 / x) - приведенная функция

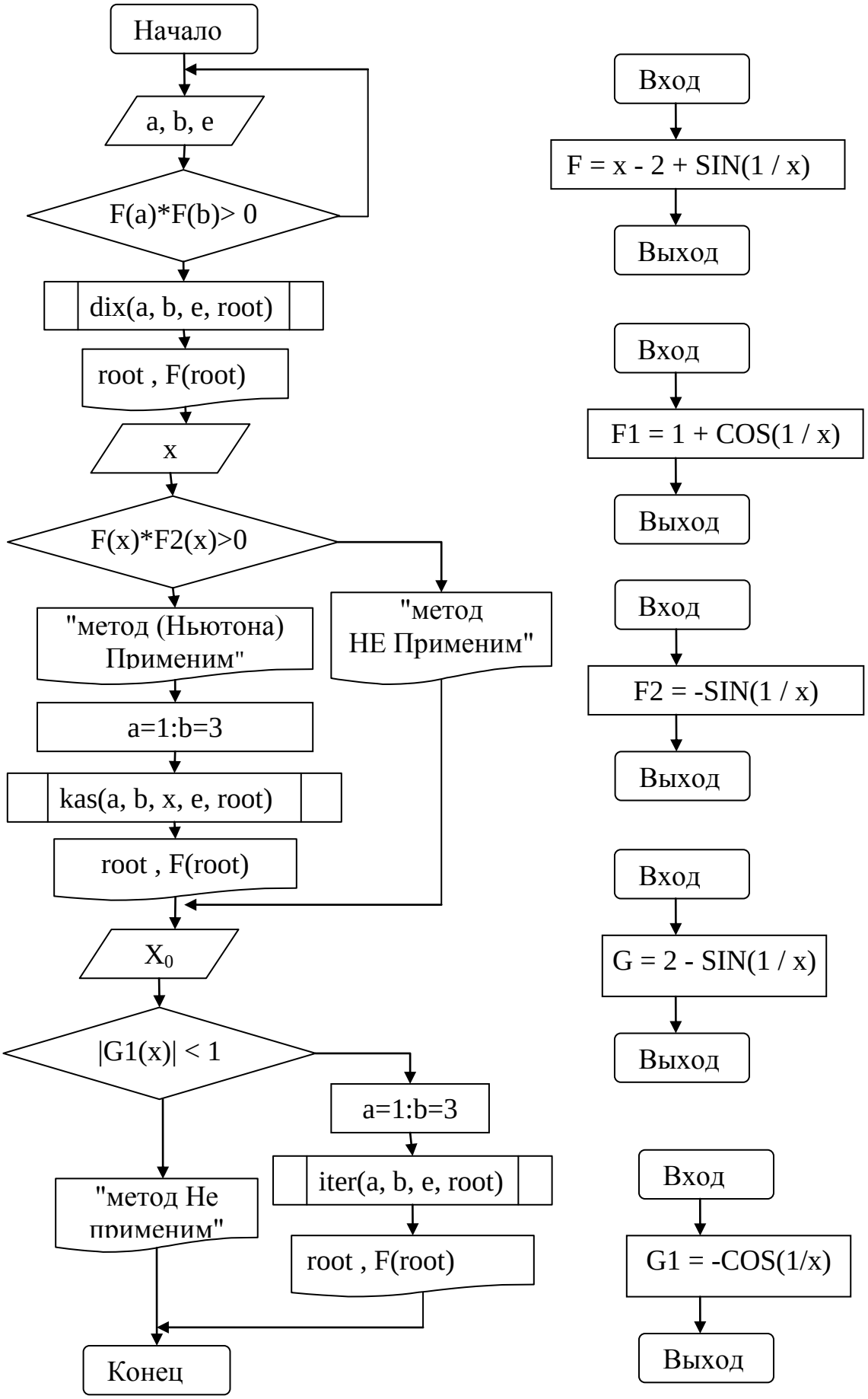
F1 (x) = 1+cos(1/x) 1-ая ПРОИЗВОДНАЯ ОТ F

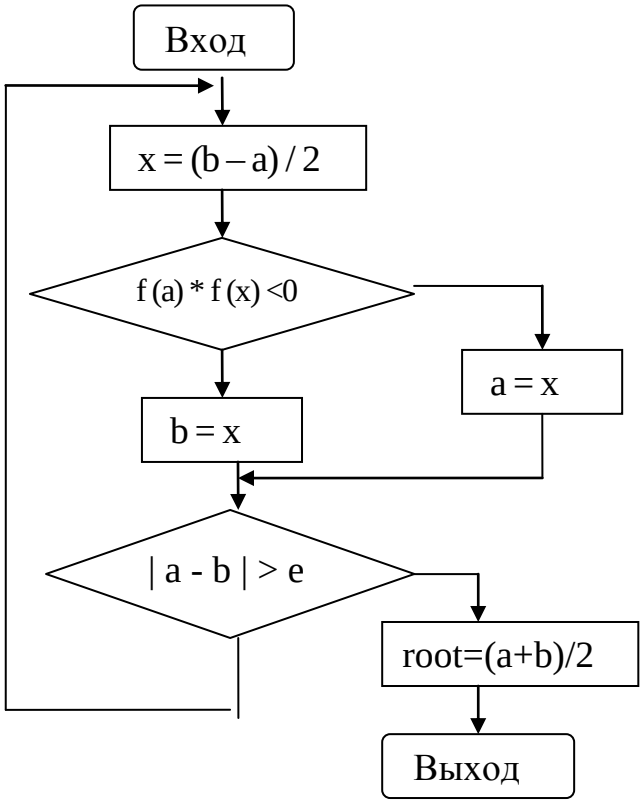
F2 (x) = -sin(1/x) 2-ая ПРОИЗВОДНАЯ ОТ F

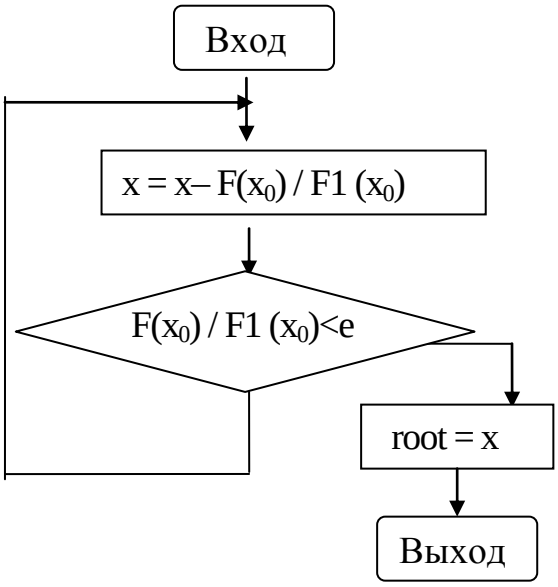
G1(x)=-cos(1/x) 1-ая ПРОИЗВОДНАЯ ОТ G

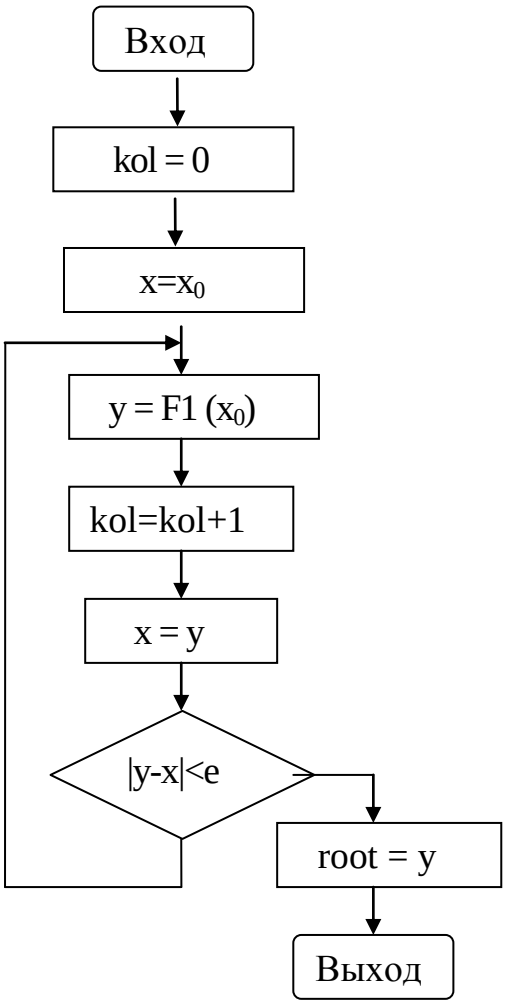
e = 0.001

БЛОК-СХЕМА


Начало
a, b, e
F(a)*F(b)> 0
dix(a, b, e, root)
root , F(root)
x
F(x)*F2(x)>0
"метод (Ньютона) Применим"
"метод НЕ Применим"
a=1:b=3
kas(a, b, x, e, root)
root , F(root)
X0
|G1(x)| < 1
"метод Не применим"
a=1:b=3
iter(a, b, e, root)
root , F(root)
Конец
Вход
F = x - 2 + SIN(1 / x)
Выход
Вход
F1 = 1 + COS(1 / x)
Выход
Вход
F2 = -SIN(1 / x)
Выход
Вход
G = 2 - SIN(1 / x)
Выход
Вход
G1 = -COS(1/x)
Выход


Вход
x = (b – a) / 2
f (a) * f (x) <0
a = x
b = x
| a - b | > e
root=(a+b)/2
Выход


Вход
x = x– F(x0) / F1 (x0)
F(x0) / F1 (x0)<e
root = x
Выход


Вход
kol = 0
x=x0
y = F1 (x0)
kol=kol+1
x = y
|y-x|<e
root = y
Выход

ВИД ПРОГРАММЫ НА ЯЗЫКЕ QBASIC

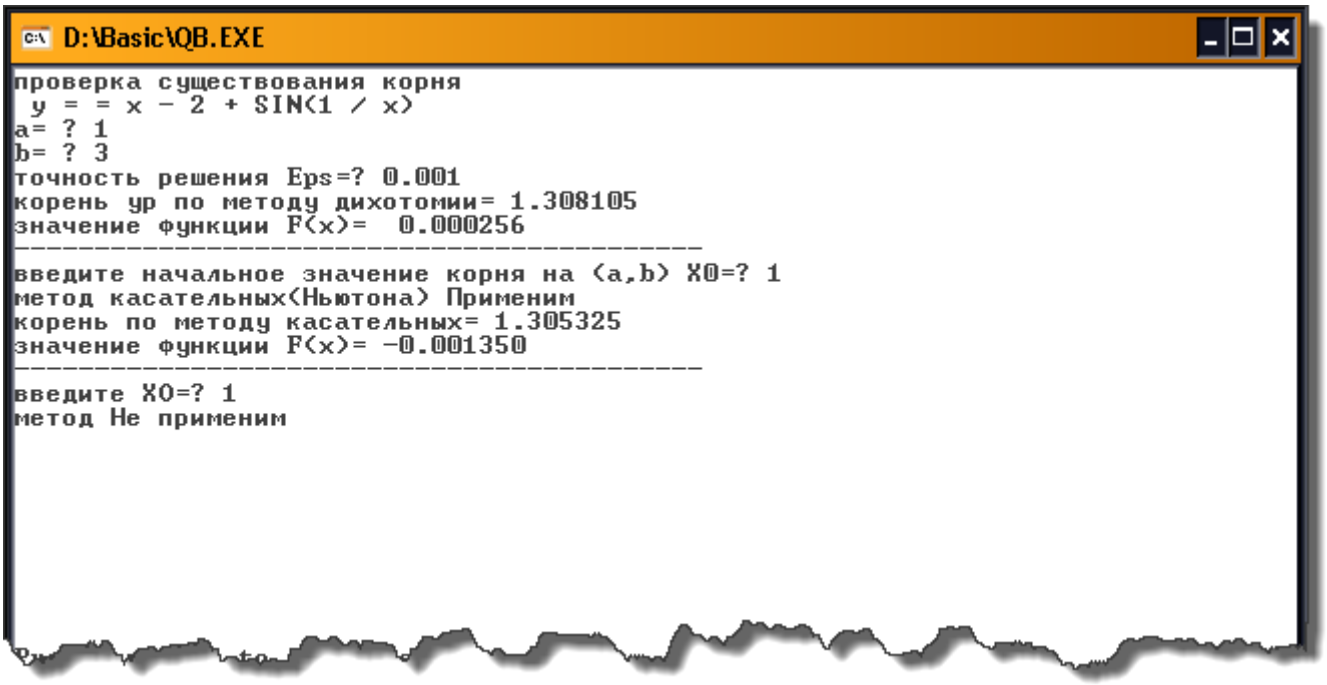
```
DECLARE SUB iter (x0!, e!, kol!, root!)
DECLARE SUB iter (x0!, e!, kol!, root!)
DECLARE SUB dix (a!, b!, e!, root!)
DECLARE SUB kas (a!, b!, x!, e!, root!)
DECLARE FUNCTION F! (x!)
DECLARE FUNCTION G! (x!)
DECLARE FUNCTION F1! (x!)
DECLARE FUNCTION F2! (x!)
DECLARE FUNCTION G2! (x!)
REM численное решение не линейных уравнений
'уточнение корня методами касательных/дихот/ньютона/итерации
CLS
PRINT "проверка существования корня"
PRINT " y = = x - 2 + SIN(1 / x)"
REM ввод отрезка с проверкой на сущ решения
DO
INPUT "a= "; a
INPUT "b= "; b
INPUT "точность решения Eps="; e
LOOP WHILE F(a) * F(b) > 0
REM мет дихотомии или метод деления отрезка пополам
CALL dix(a, b, e, root)
PRINT "корень ур по методу дихотомии="; root
PRINT "значение функции F(x)=";
PRINT USING " ##.######"; F(root)
PRINT "----------------------------------------"
REM метод касательных или метод Ньютона
INPUT "введите начальное значение корня на (a,b) X0="; x
IF F(x) * F2(x) > 0 THEN
```

```
PRINT "метод касательных(Ньютона) Применим"
a = 1: b = 3
CALL kas(a, b, x, e, root)
PRINT "корень по методу касательных="; root
PRINT "значение функции F(x)=";
PRINT USING " ##.######"; F(root)
ELSE
PRINT "метод касательных(Ньютона) НЕ Применим"
END IF
PRINT "----------------------------------------"
REM метод итерации
INPUT "введите XO="; x0
IF ABS(G1(x)) < 1 THEN
PRINT "метод Не применим"
ELSE
a = 1: b = 3
CALL iter(x0, e, kol, root)
PRINT "корень по методу итерации="; root
PRINT " количество итераций k="; kol
PRINT "значение функции F(x)=";
PRINT USING " ##.######"; F(root)
END IF
END

SUB dix (a, b, e, root)
x = (a + b) / 2
DO
IF F(x) * F(a) < 0 THEN
b = x
ELSE
```

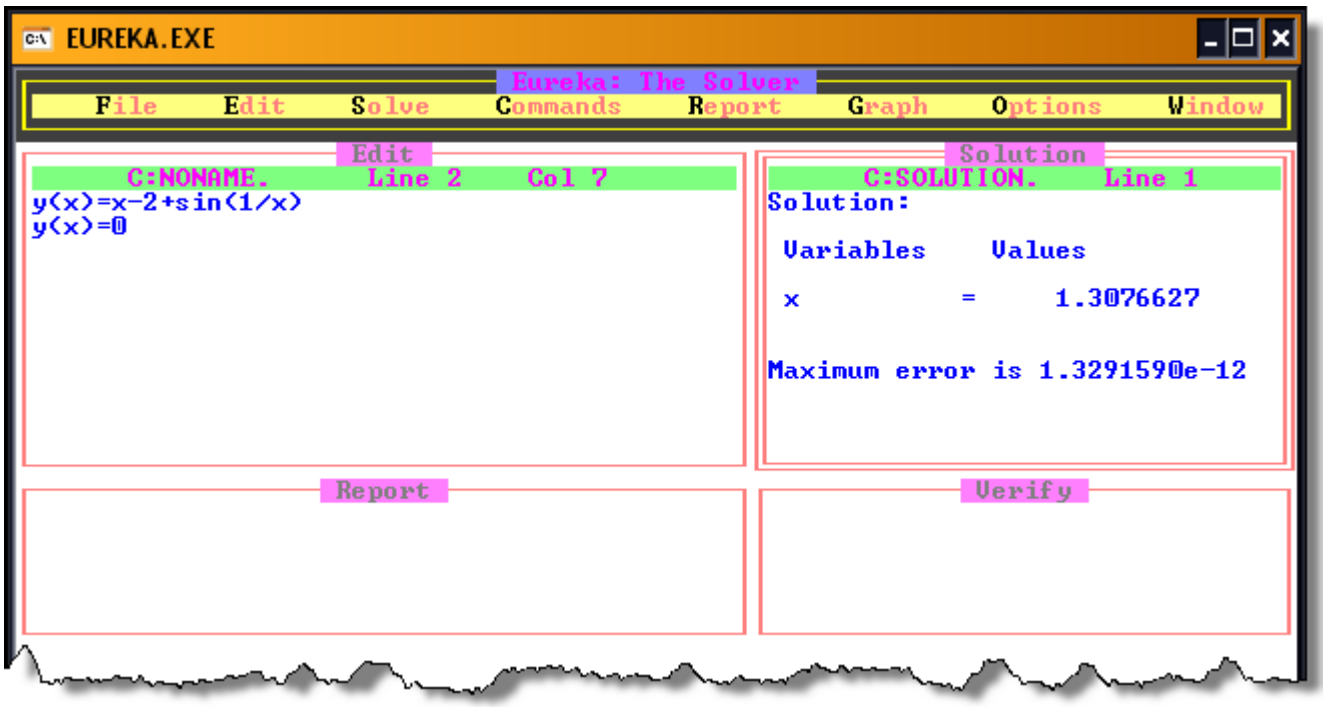
```
a = x
END IF
x = (b + a) / 2
LOOP UNTIL (b - a) < e
root = (b + a) / 2
END SUB

FUNCTION F (x)
F = x - 2 + SIN(1 / x)
END FUNCTION

FUNCTION F1 (x)
F1 = 1 + COS(1 / x)
END FUNCTION

FUNCTION F2 (x)
F2 = -SIN(1 / x)
END FUNCTION

FUNCTION G (x)
G = 2 - SIN(1 / x)
END FUNCTION

FUNCTION G1 (x)
G1 = -COS(1 / x)
END FUNCTION
SUB iter (x0, e, kol, root)
kol = 0
x = x0
DO
```

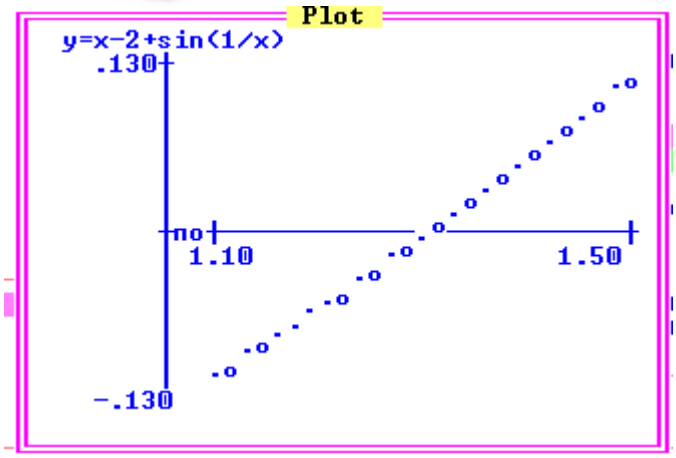
```
y = F1(x)
kol = kol + 1
x = y
LOOP UNTIL ABS(y - x) < e
root = y
END SUB

SUB kas (a, b, x, e, root)
DO
x = x - F(x) / F1(x)
LOOP UNTIL ABS(F(x) / F1(x)) < e
root = x
END SUB
```

РЕЗУЛЬТАТЫ РАБОТЫ В QBASIC



РЕЗУЛЬТАТЫ РАБОТЫ в Eureka.





Рекомендации по решению задачи:

1. Решить задачу, используя ППП Eureka.
2. Исходя из полученного решения, выбрать границы существования корня.
3. Составить блок – схему решения и программу на QBasic.

Контрольные вопросы

«Уточнение корня уравнения»

1. В чем заключается геометрический смысл метода половинного деления?
2. Какой оператор цикла используется в процедуре дихотомии?
3. Какими свойствами должна обладать функция F(x), чтобы методом половинного деления можно было гарантировать решение уравнения F(x)?

4. Что необходимо для нахождения хотя ы одного действительного корня уравнения F(x) методом половинного деления?
5. Какие процедуры функции используются в программе?
6. Какой функцией заменяется левая часть уравнения F(x)=0 в методе итерации?
7. Что называется сходимостью метода итерации?
8. Каково условие сходимости метода итерации и как это условие проверяется в программе?
9. В чем заключается геометрическая интерпретация метода Ньютона?
10. Исходя из чего выбирается в методе Ньютона первое приближение X_0
11. Для чего в программе предусмотрена процедура-функции для второй производной от исходной функции?

Варианты заданий для самостоятельного решения

Задание.

1. Уточнить корень уравнения, используя следующие методы:

- метод половинного деления;
- метод простой итерации;
- метод касательных (Ньютона).

2. Для вариантов заданий, представленных в таблице 4, выбрать точность вычисления.
3. Для вариантов заданий, представленных в таблице 6, вычислить корень с заданной точностью.
4. Для вариантов заданий, представленных в таблице 7,8,9, установить границы существования корня, точность вычисления, установить границы существования корня.

Таблица заданий № 4.

П\П	**Вид уравнения**	**Начальное приближение корня**
1.	x – sin 2x – 1 = 0	**0**
2.	2x ^ 3 + 4x – 1 = 0	**0.1**

3.	x ^ 3 + 12x – 2 = 0	**0.95**
4.	5 – x – 8lnx = 8	**4.32**
5.	x ^ 3 + x = 1000	**9.42**
6.	x – sin x = 0.25	**1.17**
7.	x ^ 3 – 6x ^ 2 + 20 = 0	**2.25**
8.	5x ^ 3 + 10x ^ 2 + 5x – 1 = 0	**0.6**
9.	$3\sin\sqrt{x}$ +0.34x-3.8 =0	**2**
10.	x – 3 + sin (3.6x) = 0	**0**
11.	arcos(x)-$\sqrt{1-0.3x\text{^}3}$ = 0	**0**
12.	√1- 0.4x ^ 2 – arcsin x = 0	**0**
13.	x – 2 + sin x = 0	**1.2**
14.	1 – x + sin x – ln (1 + x) = 0	**0**
15.	x ^ 2 – ln (1 + x) – 3 = 0	**2**
16.	x ^ 3 + x ^ 2 – 3 = 0	**0.6**
17.	x ^ 3 – x – 0.2 = 0	**0.9**
18.	5x ^ 3 – x – 1 = 0	**0.6**
19.	x ^ 3 – 2x – 5 = 0	**1.9**
20.	x ^ 3 + x = 1000	**9.1**
21.	x ^ 4 + 2x ^ 3 – x – 1 = 0	**0**
22.	x ^ 3 – x – 2 = 0	**0.9**
23.	x – sin x/2 – 1 = 0	**0**
24.	2 ^ 3 + 4x – 1 = 0	**0.1**
25.	x ^ 3 + 12x – 2 = 0	**0.95**

Таблица заданий № 5

П\П	**Вид уравнения**	**Отрезок**
1.	0.25x ^ 3 + x – 1 .2502 = 0	0, 2
2.	0.1x ^ 2 – xlnx = 0	1, 2
3.	3x – 4lnx – 5 = 0	2, 4

4.	e ^ x – e ^ -x – 2 = 0	0, 1
5.	e ^ x + lnx – 10x = 0	3, 4
6.	3x – 14 + e ^ x – e ^ -x = 0	1, 3
7.	3ln ^ 2x + 6lnx– 5 = 0	1, 3
8.	2xsinx – cosx = 0	0.4, 1
9.	xtgx – 1\3 = 0	0.2, 1
10.	√ 1 – x - cos√ 1 – x = 0	0, 1

Таблица заданий № 6

№ вар.	Уравнение	Интервал	Точность
1.	x – 1\ (2 + sin2x) = 0	[0; 1]	10^{-3}
2.	arcsin(x\3) - √ 1 – (x\3) ^ 2 = 0	[1,5; 3]	10^{-3}
3.	x - √ 9 –x+ x ^ 2 = 0	[1; 2]	10^{-3}
4.	√1 – x ^ 2 - arcsin x = 0	[0; 1]	10^{-3}
5.	tgx – (1/3)(tgx)^3 + (1/5)(tg x) ^ 5 – 1/3 = 0	[0; 0,8]	10^{-3}
6.	e ^ x – e (- x) – 2 = 0	[0; 1]	10^{-3}
7.	cosx – e(-(x ^ 2) / 2) + x – 1 = 0	[0; 2]	10^{-3}
8.	sin(x ^ 2) + cos(x ^ 2) – 10x = 0	[0; 1]	10^{-3}
9.	3sin√x + 0,35x – 3,8 = 0	[2; 3]	10^{-3}
10.	√1 – 0,04 (x ^ 2) – x = 0	[0; 1]	10^{-3}
11.	1/4(x ^ 3) + x – 1,25 = 0	[0; 1]	10^{-5}
12.	x – sin(x + 2) = 0	[0; 1]	10^{-5}
13.	√1 – x - cos√1 – x = 0	[0; 1]	10^{-3}
14.	0,1(x ^ 2) – x lnx = 0	[1; 2]	10^{-3}
15.	3x – 4 lnx – 5 = 0	[1;4]	10^{-3}
16.	e ^ x + lnx – 10 x = 0	[1; 4]	10^{-3}
17.	x tgx – 1/3 = 0	[0; 1]	10^{-3}
18.	0,25(x ^ 3) + x – 1,25 = 0	[0; 2]	10^{-3}
19.	3x – 14 + e ^ x + e (-x) = 0	[1; 3]	10^{-3}
20.	2x sinx – cosx = 0	[0,4; 1]	10^{-3}

21.	1/(1 + x ^ 2) – x = 0	[1; 2]	10^{-3}
22.	.(tg x) ^ 2 – x = 0	[1; 2]	10^{-3}
23.	x + ln(x + 0.5) - 0.5 = 0	[0;2]	10^{-3}
24.	x ^3 – x - 0.2 = 0	[1;1,1]	10^{-3}
25.	x^ 4 + 2x^ 3 – x – 1 = 0	[0; 1]	10^{-3}
26.	x ^ 3 – 0.2x^ 2 - 0.2x - 1.2 = 0	[1;1,5]	10^{-3}
27.	2sin^2x/3 – 3cos^2x/4 = 0	[0;П/2]	10^{-3}
28.	x^4+0.8x^3-0.4x^2-1.4x-1.2=0	[-1,2;-0,5]	10^{-3}
29.	x ^ 4 - 4.1x ^ 3 + x^ 2 - 5.1x + 4.1 = 0	[3,7;5]	10^{-3}
30.	x2 ^ x – 1 = 0	[0;1]	10^{-3}

Таблица заданий № 7

№ вар	Уравнение	№ вар	Уравнение
1.	x – sinx = 0,25 16.	**16.**	tg(0,3x + 0,4) = x ^ 2
2.	tg(0,58x + 0,1) = x ^ 217.	**17.**	x ^ 2 – 20sinx = 0
3.	√x – cos(0,387x) = 018.	**18.**	ctgx – x/4 = 0
4.	tg(0,4x + 0,4) = x ^ 2 19.	**19.**	tg(0,47x + 0,20 = 0
5.	lgx – 7/(2x + 6) = 020.	**20.**	x ^ 2 + 4sinx = 0
6.	tg(0,5x + 0,2) = x ^ 2 21.	**21.**	ctgx – x/2 = 0
7.	3x – cosx – 1 = 022.	**22.**	2x – lgx – 7 = 0
8.	x + lgx = 0,523.	**23.**	tg(0,44x + 0,30 = 0
9.	tg(0,5x + 0,1) = x ^ 2 24.	**24.**	3x – cosx – 1 = 0
10.	x ^ 2 + 4sinx = 025.	**25.**	ctgx – x/10 = 0
11.	ctg1,05x – x ^ 2 = 026.	**26.**	x ^ 2 + 4sinx = 0
12.	tg(0,4x + 0,3) = x ^ 2 27.	**27.**	tg(0,36x + 0,4) = 0
13.	xlgx – 1,2 = 028.	**28.**	x + lgx = 0,5
14.	1,8x ^ 2 – sin10x = 0 29.	**29.**	ctgx – x/5 = 0
15.	ctgx – x/4 = 030.	**30.**	2lgx – x/2 + 1 = 0

Таблица заданий № 8

№ вар	Уравнение	№ вар	Уравнение
1.	x ^ 3 – 3x ^ 2 + 9x – 8 = 0.	**16**	x ^ 3 + 4x – 6 = 0
2.	x ^ 3 – 6x – 8 = 0	**17**	x ^ 3 + 0,2x ^ 2 + 0,5x + 0,8 = 0
3.	x ^ 3 – 3x ^ 2 + 6x + 3 = 0	**18**	x ^ 3 – 3x 62 + 12x – 12 = 0
4.	x ^3 – 0,1x ^ 2 + 0,4x –1,5 = 0	**19**	x ^3 -0,2 x^2 + 0,3x + 1,2 = 0
5.	x ^ 3 – 3x ^ 2 + 9x + 2= 0	**20**	x ^ 3 – 2x + 4 = 0
6.	x ^ 2 + x – 5 = 0	**21**	x ^ 3 – 0,2x ^ 2 + 0,5x – 1,4 = 0
7.	x ^ 3+ 0,2 x ^2 +0,5x –1,2 = 0	**22**	x ^ 3 – 3x ^ 2 + 6x – 5 = 0
8.	x ^ 3 + 3x + 1 = 0	**23**	x ^ 3 – 0,1x ^ 2 + 0,4x + 1,2 = 0
9.	x ^ 3 + 0,2x ^ 2 + 0,5x – 2 = 0	**24**	x ^ 3 – 0,2x ^ 2 + 0,5x – 1 = 0
10.	x ^ 3 – 3x ^ 2 + 12x – 9 = 0	**25**	x ^ 3 + 3x ^ 2 + 12x + 3 = 0
11.	x ^3 –0,2x ^ 2 + 0,3x – 1,2 = 0	**26**	x ^ 3 – 0,1x ^ 2 + 0,4x + 2 = 0
12.	x ^ 3 – 3x ^ 2 + 6x – 2 = 0	**27**	x ^ 3 – 0,2 x ^ 2 + 0,4x – 1,4 = 0
13.	x ^ 3 –0,1x ^ 2 +0,4x –1,5 = 0	**28**	x ^ 3 + 0,4x ^ 2 + 0,6x – 1,6 = 0
14.	x ^ 3 + 3x ^ 2 + 6x – 1 = 0	**29**	x ^3 + x – 3 = 0
15.	x ^ 3 +0,1x ^ 2+0,4x –1,2 = 0	**30**	x ^ 3 – 0,2x ^ 2 + 0,5x + 1,4 = 0

Таблица заданий № 9

№ вар	Уравнение	№ вар	Уравнение
1).	2x ^ 3 – 3x ^ 2 – 12x – 5 = 0	**16**	2x ^ 3 – 3x ^ 2 – 12x + 1 = 0
2).	x ^ 3 – 3x ^ 2 – 24x – 3 = 0	**17**	x ^ 3 – 3x ^ 2 – 24x – 5 = 0
3).	x ^ 3 – 3x ^ 2 + 3 = 0	**18**	x ^ 3 – 4x ^ 2 + 2 = 0
4).	x ^ 3 – 12x + 6 = 0	**19**	x ^ 3 – 12x – 5 = 0
5).	x ^ 3 + 3x ^ 2 – 24x – 10 = 0	**20**	x ^ 3 + 3x ^ 2 – 24x + 1 = 0
6).	2x ^ 3 – 3x ^ 2 – 12x + 10 = 0	**21**	2x 6 3 – 3x^ 2 – 12x + 12 = 0
7).	2x ^ 3 + 9x ^ 2 – 21 = 0	**22**	2x ^ 3 + 9x ^ 2 – 6 = 0

8).	x ^ 3 – 3x ^ 2 + 2,5 = 0	**23**	x ^ 3 – 3x ^ 2 + 1,5 = 0
9).	x ^ 3 + 3x ^ 2 – 2 = 0	**24**	x ^ 3 – 3x ^ 2 – 24x + 10 = 0
10).	x ^ 3 + 3x ^ 2 – 3,5 = 0	**25**	x ^ 3 + 3x ^ 2 – 24x – 3 = 0
11).	x ^ 3 + 3x ^ 2 – 24x + 10 = 0	**26**	x ^ 3 – 12x – 10 = 0
12).	x ^ 3 – 3x ^ 2 – 24x – 8 = 0	**27**	2x ^ 3 + 9x ^ 2 – 4 = 0
13).	2x ^ 3 + 9x ^ 2 – 10 = 0	**28**	2x ^ 3 – 3x ^ 2 – 12x + 8 = 0
14).	x^ 3 – 12x + 10 = 0	**29**	X ^ 3 + 3x ^ 2 – 1 = 0
15).	x ^ 3 +3x ^ 2 – 3 = 0	**30**	x ^ 3 – 3x ^ 2 + 3,5 = 0

ЛАБОРАТОРНАЯ РАБОТА № 4

«Методы численного решения дифференциальных уравнений. Уравнения 1-го порядка»

Цель работы

Ознакомление с принципом модульного программирования на примере задачи решения дифференциальных уравнений и использование оболочки QBasic для построения подпрограмм и головного модуля.

Порядок выполнения работы

1. Получить у преподавателя вариант задания, включающий в себя
   - дифференциальное уравнение (F(x))
   - интервал (a,b)
   - шаг (h)
   - краевое значение функции (**y0)**
2. Написать подпрограмму для каждого метода (Эйлера, Эйлера-Коши, Рунге-Кутта)
3. Написать подпрограмму процедуры-функции
4. Написать головной модуль
5. Отладить программу и получить результаты
6. Построить график решений дифференциального уравнения для всех 3-х методов в Excel.

Содержание отчета

1. Содержательная постановка задачи
2. Исходные данные
3. Краткое описание методов
4. Блок схема подпрограмм и блок схема головного (или управляющего) модуля
5. Листинг подпрограмм и управляющего модуля
6. Распечатка полученных результатов
7. Распечатка результатов в Excel.

Теоретические сведения

Решение дифференциальных уравнений

Дифференциальные уравнения очень часто встречаются при построении моделей динамики объектов исследования. Они описывают, как правило, изменения параметров объекта во времени. Результатом **решения дифференциальных уравнений являются функции**, а не числа, как при решении конечных уравнений, вследствие чего методы решения их более трудоемки.

Дифференциальные уравнения описывают также процессом, тепло-массообмен, изменение концентрации вещества, процессы кристаллизации сахара и многие другие. При использовании численных методов решения дифференциальных уравнений:

$\frac{dx}{dy} = f(x, y)$ или $y' = f(x,y)$ представляется в табличном виде, т.е. получается dx совокупность значений Y_i и X_i.

Решение носит шаговый характер, т.е. по одной или нескольким начальным точкам (x, y) за один шаг находят следующую точку и т.д. Разница между двумя соседними значениями аргумента $h = x_{i+1} - x_i$ называется *шагом.*

Наибольшее распространение имеют задачи Коши, в которых заданы начальные условия: при $x = x_0$, $y(x_0) = y_0$. Имея их, легко начинать процесс решения, т.е. найти y_1 при x_1 , y_2 - при x_2 и т.д.

Основная идея получения простейших вычислительных алгоритмов в одношаговых методах сводится к разложению исходного решения *y(x)* в ряд Тейлора.

Количество оставленных членов ряда определяет порядок и, следовательно, точность метода. По полученному разложению, зная значения y в точке разложения y_i и производную $f(x_i, y_i)$, находят значения y через шаг h: $y_{i+1} = y_i + \Delta y_i$.

Если в разложении удерживается большее число членов, то необходимо рассчитывать $f(x_i, y_i)$ в несколько точках (таким способом избегают необходимости прямого вычисления высших производных, присутствующих в разложении в ряд Тейлора).

Расчётные алгоритмы многошаговых методов базируются на построении интерполяционных или аппроксимирующих функций, от которых берётся интеграл.

Численными методами решаются не только отдельные уравнения, но и системы уравнений (чаще всего первого порядка), причем большинство методов решения одного уравнения легко распространяются на решения систем.

К классу одношаговых методов относятся методы Эйлера, Рунге – Кутта и Эйлера-Коши.

Функциональное уравнение $y' = f(x,y)$, связывающее между собой независимую переменную, искомую функцию $y(x)$ и ее производную $y(x)$, называется **дифференциальным уравнением 1-го порядка.**

Решением (частным) решением уравнения на интервале (a, b) называется любая **функция** $y = \varphi(x)$, которая, будучи подставлена в это уравнение вместе со своей производной $\varphi'(x)$обращает его в тождество относительно $x \in (a,b)$. Уравнение $\Phi(x,y) = 0$, определяющее это решение как неявную функцию, называется интегралом дифференциального уравнения. На плоскости с фиксированной декартовой прямоугольной системой координат уравнение $\Phi(x,y) = 0$ определяет некоторую кривую, которая называется интегральной кривой дифференциального уравнения.

Если в дифференциальном уравнении $y' = f(x,y)$ функция $f(x,y)$ непрерывна в некоторой области D, плоскости Oxy и имеет в этой области ограниченную частную производную $f'_y(x,y)$, то для любой точки $(x_0,y_0) \in D$, в некотором

интервале $x_0 — h \le x \le x_0 + h$, существует и притом единственное решение *y (x)* этого уравнения, удовлетворяющее начальному условию

$$y\ (x_o) - y_o.$$

Это утверждение известно как теорема Коши о существовании и единственности решения дифференциального уравнения с заданным начальным условием.

Для задач подобного типа, выделенных в целый класс ***задач Коши,*** помимо аналитических методов решения разработаны методы численного решения.

Метод Эйлера

Значения искомой функции *y= y (x)* на отрезке $[x_0,X]$ находят по формуле:

$$y_{k+1} = y_k + h \cdot f(x_k, y_k), \qquad \textbf{(1)}$$

где $y_к = y\ (x_к)$, $x_{к+1} = x_k + h$, $(x_п = X)$, k = 0,1,2,...n -1 и $h = \frac{x - x_0}{n}$

По заданной предельной абсолютной погрешности е начальный шаг вычислений h устанавливают с помощью неравенства $h^2 < \varepsilon$.

Метод Эйлера - Коши

Для вычисления значений функции *y= y (x)* применяют формулу:

$$y_{k+1} = y_k + \frac{h}{2}(k_1 + k_2) \qquad \textbf{(2)}$$

где $x_{k+F} = x_k + h, k_1 = f(x_k, y_k), k_2 = f(x_{k+1}, y_{k+1}), y_{k+1} = y_k + k_1$

По заданной предельной погрешности ε начальный шаг вычислений h устанавливается с помощью неравенства $h^3 < \varepsilon$.

Метод Руге - Кутта

Значения искомой функции *y= y (x)* на отрезке $[x_0, X]$ последовательно находят по формулам:

$$y_{к+1} = y_k + \Delta y_k, \text{ k = 0, l, 2,...n – l} \qquad \textbf{(3)}$$

где $\Delta y_k = \frac{1}{6}(q_1^{(k)} + 2q_2^{(k)} + 2q_3^{(k)} + q_4^{(k)})$,

$$q_1^{(k)} = h \cdot f(x_k, y_k), \qquad q_2^{(k)} = h \cdot f(x_k + \frac{h}{2}, y_k + \frac{q_1^{(k)}}{2})$$

$$q_3^{(k)} = h \cdot f(x_k + \frac{h}{2}, y_k + \frac{q_2^{(k)}}{2}), \qquad q_4^{(k)} = h \cdot f(x_k, y_k + q_4^{(k)})$$

$$x_k = x_k + h,\ x_n = X, \qquad h = \frac{X - x_0}{n}$$

По заданной предельной абсолютной погрешности ε начальный шаг вычислений h устанавливают с помощью неравенства $h^4 < \varepsilon$.

Правило Рунге - Ромберга

Пусть y_{2k}^h и y_k^{2h} - значения искомой функции, полученные одним из указанных методов при шагах вычисления h и 2h соответственно, а ε - заданная абсолютная предельная погрешность. Тогда считается, что достигнута заданная точность вычислений, если выполняется неравенство:

$$\frac{1}{2^s - 1} \cdot \left| y_{2k}^{(h)} - y_k^{(2h)} \right| < \varepsilon \qquad \textbf{(4)}$$

при всех k и при s = 2,3,4 соответственно для методов Эйлера, Эйлера - Коши, Рунге - Кутта. Решением задачи является функция $\{y_k^h\}$.

Применяя указанное правило, последовательно вычисляют значения искомой функции с шагом 2h и с шагом h и сравнивают полученные результаты по формуле (4). Вычисления заканчивают, когда неравенство (4) выполняется при всех k.

Пример решения поставленной задачи

Функцию варианта задания оформляем как процедуру - функции, используя в меню оболочки QBasic.

- дифференциальное уравнение (Y′(x))

  FUNCTION f (x, y0)

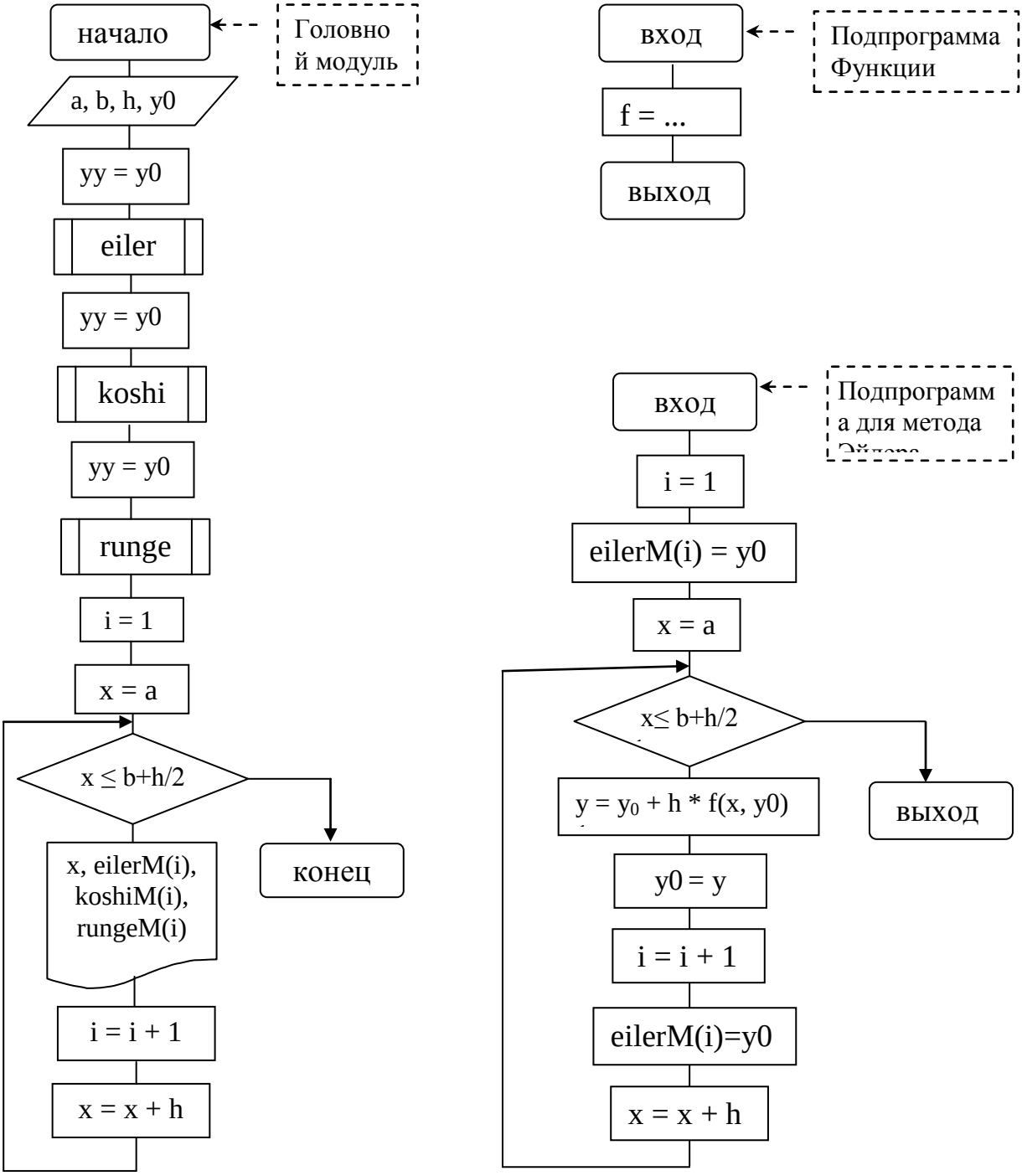
  f = < **функция соответствующего варианта** >

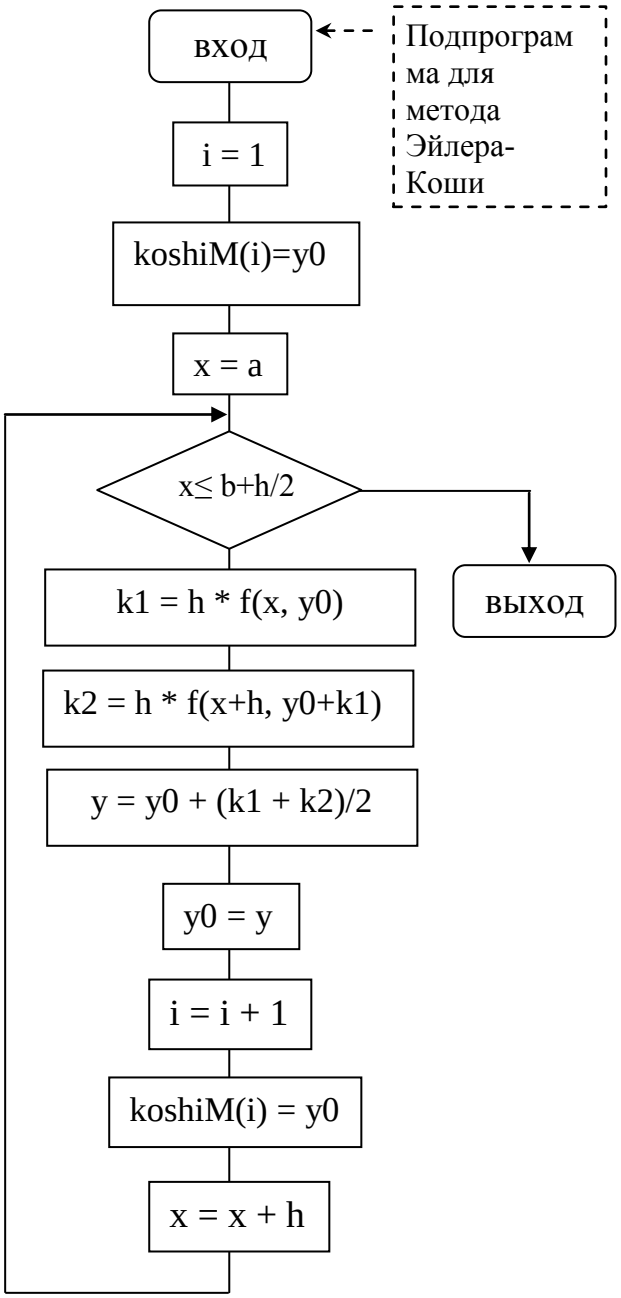
  END FUNCTION

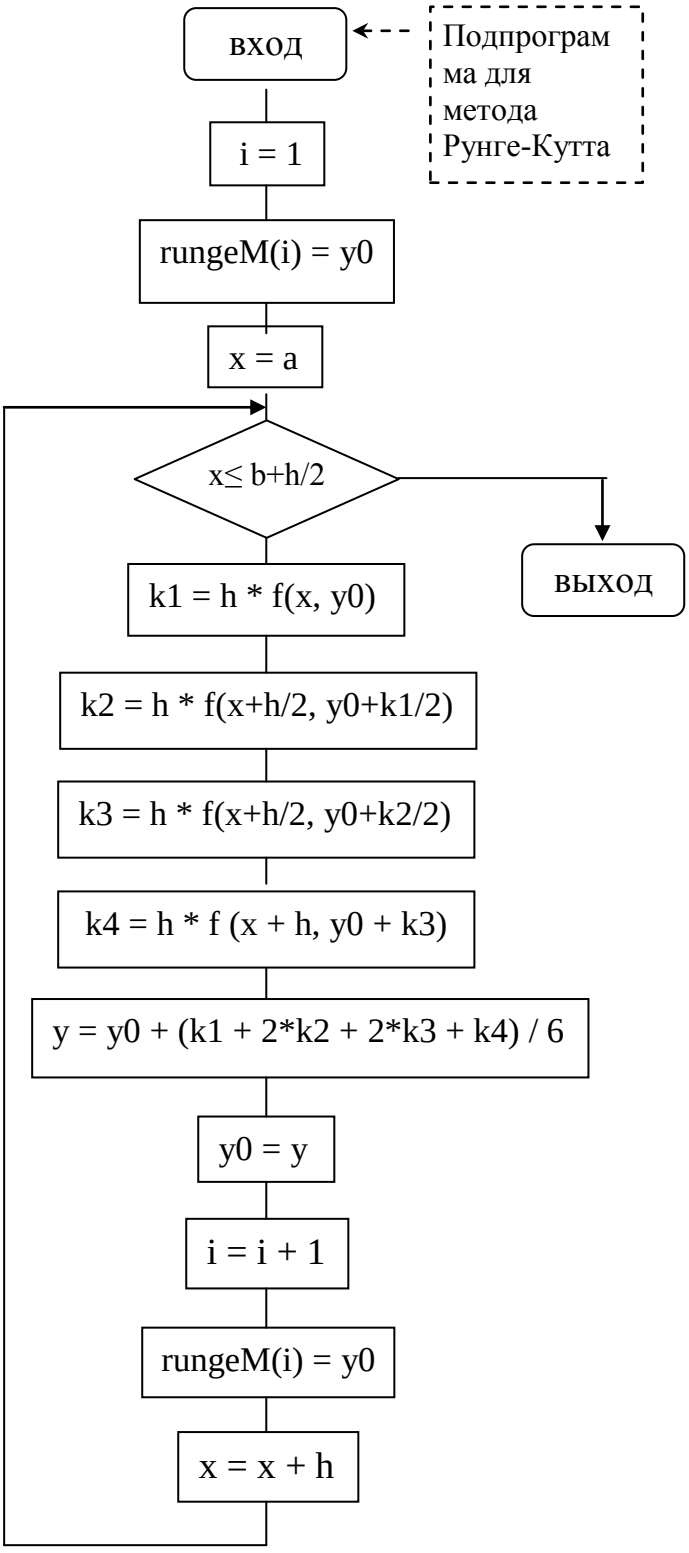
- интервал (a, b), шаг (h), краевое значение функции (y0)

Блок-схема для задачи решения дифференциального уравнения имеет вид.

БЛОК-СХЕМА АЛГОРИТМА РЕШЕНИЯ




вход
Подпрограм
ма для
метода
Эйлера-
Коши
i = 1
koshiM(i)=y0
x = a
x≤ b+h/2
выход
k1 = h * f(x, y0)
k2 = h * f(x+h, y0+k1)
y = y0 + (k1 + k2)/2
y0 = y
i = i + 1
koshiM(i) = y0
x = x + h


вход
Подпрограм
ма для
метода
Рунге-Кутта
i = 1
rungeM(i) = y0
x = a
x≤ b+h/2
выход
k1 = h * f(x, y0)
k2 = h * f(x+h/2, y0+k1/2)
k3 = h * f(x+h/2, y0+k2/2)
k4 = h * f (x + h, y0 + k3)
y = y0 + (k1 + 2*k2 + 2*k3 + k4) / 6
y0 = y
i = i + 1
rungeM(i) = y0
x = x + h

Запись всех подпрограмм можно осуществить через меню оболочки QBasic:

1. Alt - вход в меню
2. Edit→New SUB … - создание новой подпрограммы
3. Edit→New FUNCTION… - создание новой подпрограммы функции
4. Набираем в диалоговом окне новое имя подпрограммы (например: eiler)

На экране появляется заготовка для создания подпрограммы:

SUB <имя подпрограммы, параметры>

<текст подпрограммы>

END SUB

1. Приступаем к написанию подпрограммы между ключевыми словами SUB и END SUB
2. Все вспомогательные подпрограммы объединяются управляющей программой или головным модулем.
3. Переход от текста управляющей программы к текстам подпрограммам происходит при нажатии клавиш F2.

ВИД ПРОГРАММЫ НА ЯЗЫКЕ QBASIC

Головной модуль на языке QBasic

```
DECLARE SUB eiler (a!, b!, h!, y0!)
DECLARE SUB koshi (a!, b!, h!, y0!)
DECLARE SUB runge (a!, b!, h!, y0!)
DECLARE FUNCTION f! (x!, y0!)
CLS
DIM SHARED eilerM(1000), koshiM(1000), rungeM(1000)
INPUT "левый конец интервала a= "; a
INPUT "правый конец интервала b= "; b
INPUT "шаг "; h
INPUT "краевое значение функции Y0= "; y0
yy = y0                          'сохранение краевого значения функции
REM Вызов Метода Эйлера
CALL eiler(a, b, h, yy)
```

```
yy = y0

REM Вызов Метода Эйлера-Коши
CALL koshi(a, b, h, yy)
yy = y0
REM Вызов Метода Рунге-Кутта
CALL runge(a, b, h, yy)
PRINT "------------------------------------------------------------"
PRINT "│ x │ elier │ koshi │ runge │"
PRINT "------------------------------------------------------------"
L$ = "│ #.# │ ##.##### │ ##.##### │ ##.##### │"
i = 1
FOR x = a TO b + h / 2 STEP h
PRINT USING L$; x; eilerM(i); koshiM(i); rungeM(i)
i = i + 1
NEXT x
PRINT "------------------------------------------------------------"
END
```

Первые четыре строчки пишутся автоматически при присоединении подпрограмм к головному модулю в результате выполнения команды **Save All.**

Подпрограмма для решения дифференциального уравнения **методом Эйлера**:

```
SUB eiler (a, b, h, y0)
i = 1
eilerM(i) = y0
FOR x = a TO b + h / 2 STEP h
y = y0 + h * f(x, y0)
y0 = y
i = i + 1
eilerM(i) = y0
NEXT x
```

END SUB

Подпрограмма для решения дифференциального уравнения **методом Эйлера-Коши:**

```
SUB koshi (a, b, h, y0)
i = 1
koshiM(i) = y0
FOR x = a TO b + h / 2 STEP h
k1 = h * f(x, y0)
k2 = h * f(x + h, y0 + k1)
y = y0 + (k1 + k2) / 2
y0 = y
i = i + 1
koshiM(i) = y0
NEXT x
END SUB
```

Подпрограмма для решения дифференциального уравнения **методом Рунге-Кутта:**

```
SUB runge (a, b, h, y0)
i = 1
rungeM(i) = y0
FOR x = a TO b + h / 2 STEP h
k1 = h * f(x, y0)
k2 = h * f(x + h / 2, y0 + k1 / 2)
k3 = h * f(x + h / 2, y0 + k2 / 2)
k4 = h * f(x + h, y0 + k3)
y = y0 + (k1 + 2 * k2 + 2 * k3 + k4) / 6
y0 = y
i = i + 1
rungeM(i) = y0
NEXT x
END SUB
```

Процедура **функция**:

FUNCTION f (x, y0)

f = < **функция соответствующего варианта** >

END FUNCTION

Построение в Excel графика решений

1. Для построения графика в Excel следует:
2. Набить таблицу значений из Qbasic (в каждом столбце результат решения дифференциального уравнения соответствующим методом).

x	Метод Эйлера	Метод Эйлера-Коши	Метод Рунге-Кутта
0,5	0,672484	0,678814	0,678894
0,6	0,757629	0,770744	0,770909
0,7	0,855875	0,876242	0,876498
0,8	0,967663	0,995754	0,996108
0,9	1,093427	1,129714	1,130172
1	1,233588	1,278529	1,279097
1,1	1,388542	1,44257	1,443255
1,2	1,558648	1,622153	1,62296
1,3	1,74421	1,817519	1,818455
1,4	1,945463	2,028813	2,029882
1,5	2,162545	2,256061	2,257268

1. Выделить столбцы три столбца – Метод Эйлера, метод Эйлера-Коши, метод Рунге-Кутта.
2. Дальше в меню выбрать: Вставка → Диаграмма.
3. Появиться меню «Мастера диаграмм».
4. Выбрать: на вкладке «Стандартные» → График → График с маркерами, помечающими точки данных → Далее.
5. Откроется окно «Исходные данные».

6. В окне «Исходные данные», выбрать вкладку Ряд → Подписи оси X, нажать маркер . Рисунок 1.

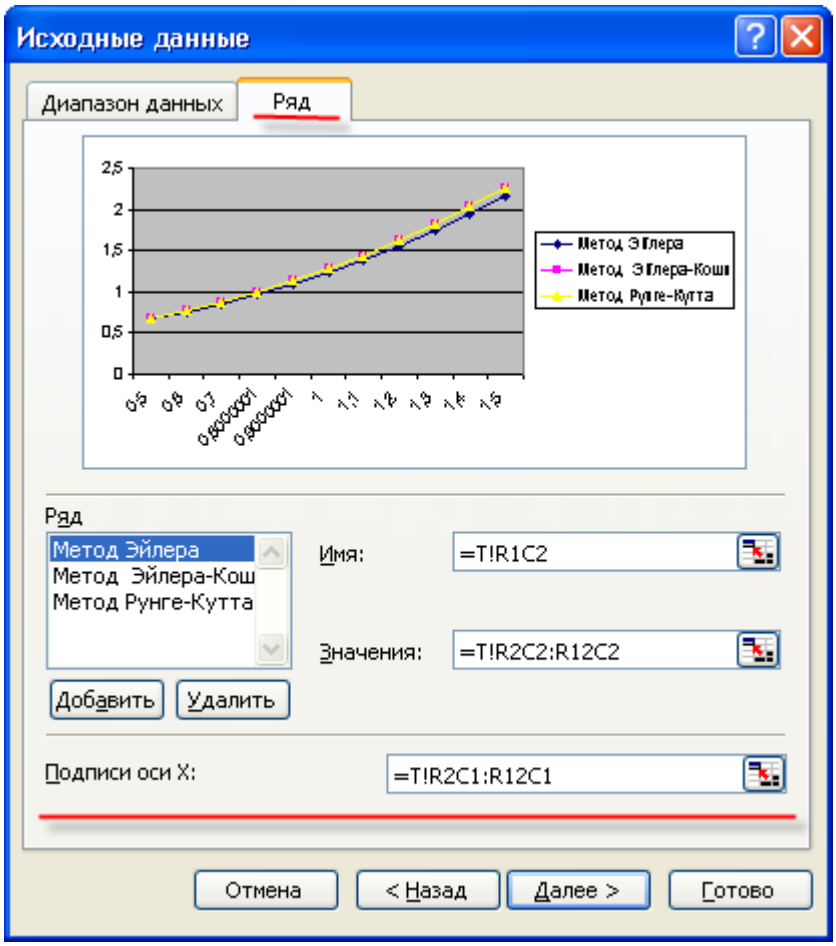


Рисунок 1.

1. Выделить столбец X, только цифры. Нажать→ Далее.
2. Легенду разместить в низу. Нажать →Далее → Готово (рис.1)
3. Добавить Линию тренда, уравнение и R^2. (рис.2)

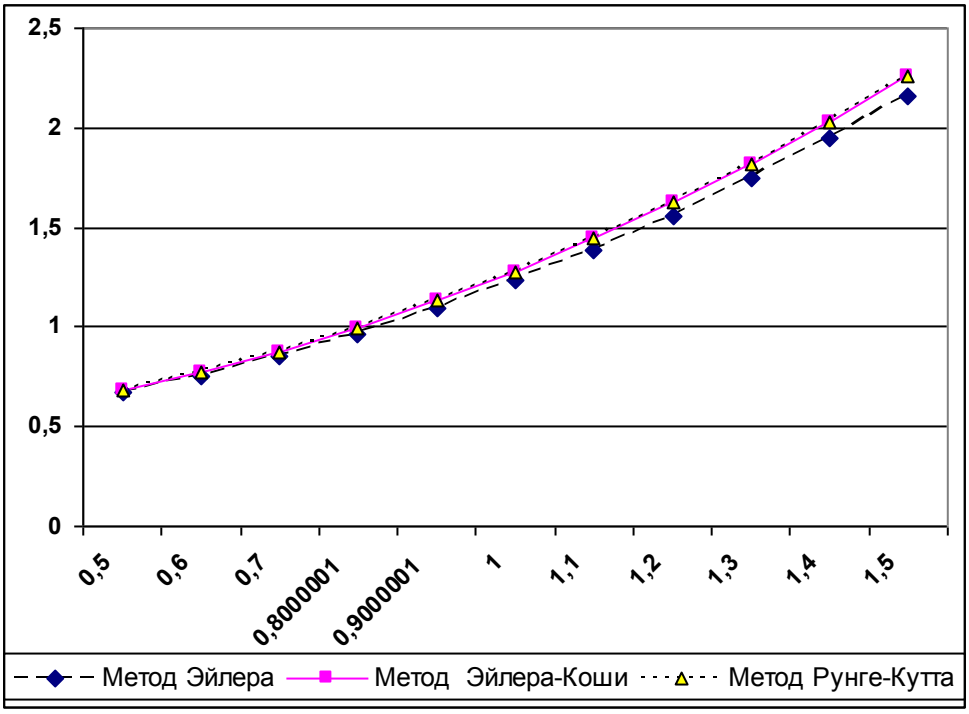


Рисунок 1

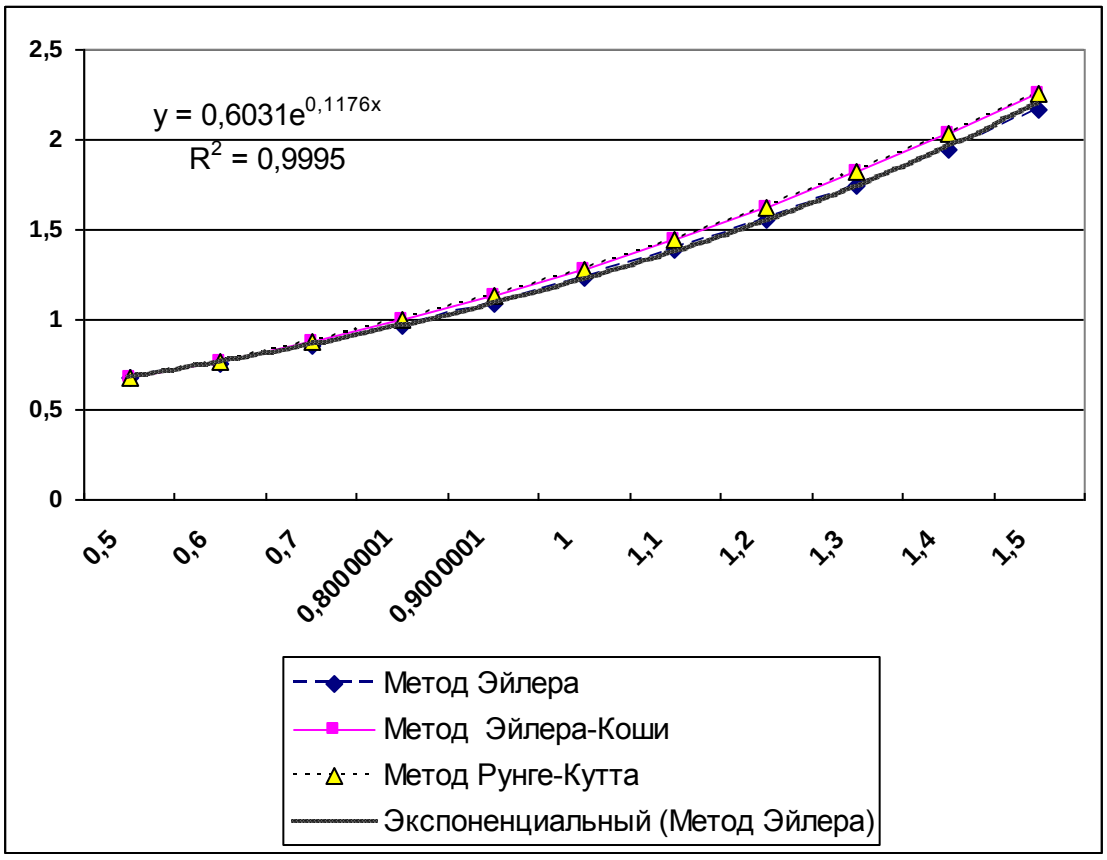


Рисунок 2

Контрольные вопросы

Метод Эйлера

1. Что является решением дифференциального уравнения?
2. Необходим ли поиск начальных условий в методе Эйлера?
3. К какой группе относится модифицированный метод Эйлера?

4. Почему точность метода Эйлера пропорциональна h, а модифицированного — h2?

5. Метод Эйлера относится к одно шаговым методам. В чем основное отличие одно- и многошаговых методов?

6. Можно ли методом Эйлера решать системы дифференциальных уравнений?

7. Можно ли использовать метод Эйлера для решения задач, не относящихся к задачам Коши?

8. Обязательно ли необходимо задание начальных условий при решении дифференциального уравнения методом Эйлера?

9. В чем заключается отличие явных и неявных вычислительных схем в модифицированном методе Эйлера?

10. Можно ли оценить погрешность решения дифференциального уравнения, не зная точного решения?

Метод Рунге — Кутта

1. Сколько раз необходимо на каждом шаге вычислять правую часть уравнения при использовании метода четвертого порядка?

2. Как можно оценить погрешность решения дифференциального уравнения при использовании метода Рунге — Кутта?

3. Можно ли задавать погрешность решения при автоматическом подборе шага в относительных величинах?

4. Сколько предыдущих значений функции нужно иметь, чтобы сосчитать одно следующее значение?

5. К какой группе методов (аналитические или численные) относится имеющий аналитическое выражение от искомого значения функции метод Рунге — Кутта?

6. Как записывается рекуррентная формула метода четвертого порядка?

7. Что можно отнести к недостаткам метода, например, самого распространенного четвертого порядка?

8. Как зависит погрешность метода от величины шага решения?

9. Возможно ли применение переменного шага в методе Рунге — Кутта?

Варианты заданий к лабораторной работе

№ п/п	Уравнение	Начальные значение	Конечное значение	Шаг	Начальное значение функции Y
1.	$Y' = y + e^{2x}$	0	1,5	0.16	Y(0)=3
2.	$Y' = \cos(x) - y$	0	2	0.2	Y(0)=1.5
3.	$Y' = -\frac{12x+5y-9}{5x+2y-3}$	0	2	0,2	Y(0)=0
4.	$Y' = x^2 - \frac{y}{x}$	1	3	0,2	Y(1)=1
5.	$Y' = e^{2x} - 3y$	0	2	0,2	Y(0) = 0
6.	$Y' = \frac{1}{\cos(x)} - \frac{1}{ytg(x)}$	0	2	0,2	Y (0) = 2
7.	$Y' = e^x - x + 2y$	0	2	0,2	Y (0) = 0
8.	$Y' = \frac{y}{1+x} - y^2$	0	2	0,2	Y (0) = 1
9.	$Y' = \frac{y - x^2}{x + x^2}$	1	3	0,2	Y (1)=1
10.	$Y' = -4y + \sin(2x)$	0	1	0,2	Y(0) = 1
11.	$Y' = -y + e^{-x}\cos(x)$	0	2	0,1	Y(0) = 0
12.	$Y' = -y + 1-e^x$	0	2	0,2	Y(0) = 2,5
13.	$Y' = -y + e^x\cos(x)$	0	1	0,2	Y(0) = 0
14.	$Y' = -y - \sin(xe^x)$	0	2	0,1	Y(0) = 1
15.	$Y' = xy$	0	1	0,2	Y(0) = 1
16.	$Y' = x + \cos\left(\frac{y}{п}\right)$	1,7	3	0,1	Y0(1,7) = 5,3
17.	$Y' = \frac{x}{2} + \frac{e^2}{x+y}$	1,8	2,5	0,15	Y0(1,3) = 4,5
18.	$Y' = \left(\frac{x}{2} + 3y\right)^{\frac{1}{3}}$	3,1	5,4	0,1	Y0(3) = 5
19.	$Y' = \frac{y}{x} \cdot \frac{y}{\ln x - 1}$	1	1,5	0,3	Y0(1) = O,5

№ п/п	Уравнение	Начальные значение	Конечное значение	Шаг	Начальное значение функции Y
20.	$Y' = 2x + \sin\frac{y}{x}$	0,1	1	0,05	Y0(0,1)=1
21.	$Y' = \sqrt{x} + \sqrt{y}$	0	1	0,1	Y0(0) = 0
22.	$Y' = x + \left(3 + y^2\right)^{\frac{1}{3}}$	0	1	0,1	Y0(0) = 0
23.	$Y' = \frac{y}{e^{-x} + y^2}$	0,1	1	0,1	Y0(0,1)=1
24.	Y' = x-y	0	1	0,1	Y0(0) = 0
25.	$Y' = \frac{xy}{1 - x^2}$	0	0,5	0,1	Y0(0,1)=1
26.	Y' = 2xy	0	6	0,05	Y0(0) = 1
27.	Y' = 2x – 3y	0	6	0,05	Y0(0) = 1

ЛАБОРАТОРНАЯ РАБОТА № 5
Символьные переменные

Цель работы

Ознакомление с принципами программирования задач с символьными переменным.

Алгоритмы обработки текстовых величин

Текстовыми (символьными, строковыми, литерными) называют переменные, которые содержат в качестве значения текстовую информацию. В отличие от числовых переменных, хранящих только число, текстовые содержат буквы, цифры и специальные знаки. В одной такой переменной можно разместить целый текст.

Переменные текстового типа широко используются в задачах обработки символов, создании шифров; баз данных и многих других. Логика программного решения, таких задач несколько отличается от рассмотренных ранее алгоритмов, поэтому знакомство с нею играет важную роль в подготовке программиста.

Текстовые переменные могут содержать только символы из таблицы ASCII (American Standard Code for Information Interchange). Таблица ASCII. У компьютера нет отдельного участка памяти для хранения текста. Все, что поступает в память машины, преобразуется в числовой формат, то есть в двоичное представление. Формат ASCII состоит в том, что каждый выводимый на экран символ имеет номер в интервале от 0 до 255. В это количество входят как видимые знаки, такие как буквы, цифры, знаки пунктуации, так и управляющие символы — табуляция, перевод строки и пр; Управляющие символы имеют номера от 1 до 32.

Различают **два типа строковых переменных** — переменной и фиксированной длины.

Присвоение значения текстовой переменной может быть осуществлено несколькими способами в зависимости от решаемой задачи.

1. Ввести текст с клавиатуры. INPUT A$

2. Присвоить текст в кавычках. A$ = "BASIC"

3. Присвоить значения других переменных или текста в кавычках.

B$ = A$ + "for ever! "

4. Присвоить результат расчета символьного выражения.

D$ = LEFT$(A$, 3)

5. Присвоить значение с помощью операторов DATA READ.

DATA "Терек": READ A$

Если строковой переменной не присвоено никакого значения, но она указана в тексте, то программа считает ее пустой — A$ = " ". Так можно «обнулять» текстовые переменные.

Таблица ASCII

128	А	144	Р	160	а	176	▒	192	└	208	╨	224	р	240	Ё
129	Б	145	С	161	б	177	▒	193	┴	209	╤	225	с	241	ё
130	В	146	Т	162	в	178	▓	194	┬	210	╥	226	т	242	Є
131	Г	147	У	163	г	179	│	195	├	211	╙	227	у	243	є
132	Д	148	Ф	164	д	180	┤	196	─	212	╘	228	ф	244	Ї
133	Е	149	Х	165	е	181	╡	197	┼	213	╒	229	х	245	ї
134	Ж	150	Ц	166	ж	182	╢	198	╞	214	╓	230	ц	246	Ў
135	З	151	Ч	167	з	183	╖	199	╟	215	╫	231	ч	247	ў
136	И	152	Ш	168	и	184	╕	200	╚	216	╪	232	ш	248	°
137	Й	153	Щ	169	й	185	╣	201	╔	217	┘	233	щ	249	·
138	К	154	Ъ	170	к	186	║	202	╩	218	┌	234	ъ	250	·
139	Л	155	Ы	171	л	187	╗	203	╦	219	█	235	ы	251	√
140	М	156	Ь	172	м	188	╝	204	╠	220	▄	236	ь	252	№
141	Н	157	Э	173	н	189	╜	205	═	221	▌	237	э	253	¤
142	О	158	Ю	174	о	190	╛	206	╬	222	▐	238	ю	254	■
143	П	159	Я	175	п	191	┐	207	╧	223	▀	239	я	255	

Сложение текстовых переменных

Текстовые переменные можно складывать и сравнивать между собой. Сложение текстовых переменных называется *конкатенацией.* Для сложения используют знак +. Фактически сложение является сцеплением текстовых строк — вторая записывается вплотную за первой.

Пример: Сложить слова «Мой» и «компьютер», поставив между ними пробел

```
A$ = "Мой"
B$ = "компьютер"
C$ = A$ + " " + B$
PRINT C$
```

Результат: Мой компьютер

Другие математические действия с текстовыми переменными проводить нельзя. Если указан знак, отличный от +, то будет выдано сообщение об ошибке Type mismatch (Несоответствие типа).

Сравнение текстовых переменных

Сравнение текстовых переменных проводится с помощью стандартных операторов сравнения: =, >, <, >=, <=, <>. В ходе сравнения компьютер анализирует три основных параметра: количество символов, последовательность ASCII кодов и номера ASCII - кодов. Программа анализирует только соответствующие знаки: первый знак первой переменной сравнивается с первым знаком второй переменной, второй знак со вторым и так далее.

Переменные равны, если они содержат равное количество одинаковых знаков, расположенных в одинаковой последовательности.

Сравнение текстовых переменных: равенство		
Равны	**Не равны**	**Причина**
"аа"="аа"	"аа"≠" аа"	Разное число знаков
"10"="10"	"10"≠"01	Нарушена последовательность знаков

"Аврора"="Аврора"	"Аврора"≠"аврора"	"А" и "а" имеют разный код
"QB" = "Q" + "B"	"norma"≠"почта"	На разных языках

Сравнивать переменные на равенство и выводить сообщение о результате можно в алгоритмах на основе оператора IF: INPUT A$: INPUT B$ IF A$ - B$ THEN PRINT "Равны" ELSE PRINT "Не равны"

Одна текстовая переменная больше другой, если в ходе опарного сравнения, соответствующих знаков очередной ее символ имеет ASCII-код больше, чем соответствующий знак сравниваемой переменной. Как только обнаруживается знак, ASCII-код которого больше или меньше, чем у соответствующего знака второй переменной, процесс сравнения заканчивается и программа выводит результат.

Этот же принцип используется при сортировке списков. Сначала машина сравнивает первые знаки. Если они равны, то сравниваются вторые, третьи и так далее. Таким образом, сортируют даже строки, имеющие в начале одинаковые наборы символов.

Сравнение текстовых переменных: неравенство	
Больше	**Причина**
"ww2" > "ww"	Три знака больше двух, если первые два знака равны
"ww2" > "wwO"	ASII-код 2 (50), больше ASII-кода 0 (48)
"f > "p"	ASII-код буквы f (102) больше F (70)
"A" > "A"	ASII-код русской буквы А (128) больше ASII-кода латинской буквы A (65)
"file.dat" > "file.bas"	ASII-код буквы d больше b
"21" > "2"+"00"	ASII-код цифры 1 больше, чем ASII-код цифры 0

Инструменты обработки текстовых величин

Существует ряд функций и операторов, предназначенных для анализа и обработки строк. Принципы их действия, допустимые аргументы, практика

использования очень похожи, поэтому рассмотрим большинство инструментов в одном параграфе, чтобы он стал справочной базой данной главы.

№	Оператор	Описание
1.	**Функция LEN**	Функция предназначена для определения количества символов в текстовой переменной
2.	**Функция LEFT$**	Функция применяется для выделения заданного количества левых (первых) символов текстового выражения.
3.	**Функция RIGHT$**	Функция аналогична функции LEFT$, только предназначена для выделения указанного количества правых (последних) символов переменной.
4.	**Функция MID$**	Функция предназначена для выделения n символов из символьного выражения, начиная с k-го.
5.	**Оператор MID$**	Оператор предназначен для *замены* n символов текстовой переменной, начиная с k-го, на знаки другого символьного выражения.
6.	**Функция INSTR**	Функция определяет, входит ли одна текстовая переменная в другую
7.	**Функции LTRIM$ и RTRIM$**	Эти функции используются для удаления первых (левых) и последних (правых) пробелов текстового выражения, если начальные или концевые пробелы есть.
8.	**Функция ASC**	Функция возвращает код ASCII, соответствующий указанному знаку.
9.	**Функция CHR$**	Выводит на экран символ, соответствующий определенному номеру в таблице ASCII.

Функция LEN

Функция предназначена для определения количества символов в текстовой переменной.

LEN *(символьное_выражение)*

Символьное_выражение — выражение, результат которого представлен величиной текстового типа. Символьным выражением может быть отдельная текстовая переменная, текст в кавычках, сумма переменных, символьная функция с соответствующим аргументом и пр.

Пример1. **Определить количество знаков в текстовой переменной и в сумме текстовой переменной и строки текста в кавычках.**

A$ = “Информатика”

X = LEN (A$): PRINT X

PRINT LEN(A$ + “BASIC”)

Результат пример 1:

11

16

Работа программы. Первая функция LEN вернула количество символов в переменной A$. Затем это значение было присвоено переменной X и выведено, на экран. Вторая функция LEN определила количество символов в сумме переменной A$ и строки текста в кавычках.

Функция LEFT$

Выделенные символы можно присвоить другой переменной или использовать как часть общего выражения. Содержание исходной строки не изменяется.

LEFT$ *(символьное_выражение. N)*

Здесь символьное_выражение — текстовая переменная, символы в кавычках или выражение, результатом вычисления которых является строковое значение;

N — количество левых знаков символьного выражения. N может быть числом, переменной или математическим выражением с целым положительным результатом.

Пример 2. Примеры работы функции LEFT$.

```
A$ = "Microsoft"
B$ = LEFT$(A$. 5): PRINT B$              '1
PRINT LEFT$(A$, LEN(A$) - 4)             '2
PRINT LEFT$("Computer". 4) + "IBM"       '3
```

Результат:

```
Micro
Micro
CompIBM
```

Работа программы. Первая строка иллюстрирует простейший случай. Функция LEFT$ выделяет первые 5 символов текстовой переменной и присваивает их переменной B$. В строке '2 число выделяемых символов задано выражением с участием текстовой функции. Запись *LEN(A$)* - 4 дает возможность выделить все первые знаки переменной A$, кроме последних четырех. Строка '3 показывает, что выделенные знаки могут использоваться как часть общего выражения, в примере они участвуют в сумме.

Функция RIGHT$

Функция аналогична функции LEFT$, только предназначена для выделения указанного количества правых (последних) символов переменной.

RIGHT$ *(сиивольное_выражение. N)*

Описание аргументов функции соответствует описанию аргументов функции LEFT$, приведенных выше.

Функция MID$

Функция предназначена для выделения n символов из символьного выражения, начиная с k-го.

MID$(*символьное_выражение. k [. n])*

Здесь символьное_выражение — строка, из которой выделяется указанное число символов, *k* — номер символа, с которого начинается выделение. Если значение k больше, чем количество символов в анализируемом тексте, то MID$ возвращает строку нулевой длины (""). Если

значение k равно нулю или отрицательно, то появляется сообщение Illegal function call (Неверный вызов функции). n — количество последовательных символов, которое нужно выделить. Если n не указано или n < k, то выделяются все символы, с k-го и до конца строки. При n = 0 функция возвращает, пустую строку.

В качестве k и n можно ставить числа, переменные или расчетные выражения с числовым результатом. Дробные значения автоматически округляются до целых чисел по правилам математики.

Пример 3. Примеры работы, функции MID$.

A$ = "ИНФОРМАТИКА"

M$ = MID$(A$, 3, 6)

PRINT M$

PRINT MID$(A$, 3)

Результат:

ФОРМАТ

ФОРМАТИКА

Работа программы. Первая функция MID$ выделяет 6 последовательных знаков, начиная с третьего, из текстовой переменной A$ и присваивает их переменной M$. Вторая функция MID$ возвращает все символы A$, начиная с третьего, потому что не указано количество выделяемых знаков.

Оператор MID$

Оператор MID$ предназначен для *замены* n символов текстовой переменной, начиная с k-го, на знаки другого символьного выражения. Оператор изменяет значение текстовой переменной. Согласно формату он замещает n знаков текстовой переменной, начиная с k-го, на первые n знаков текстового выражения, стоящего справа от знака равенства.

Пример 4:

A$ = "123456"

MID$(A$, 3, 2) = "abc"

PRINT A$

Результат:

12ab56

Два знака переменной A$, начиная с третьего, были замещены первыми двумя знаками выражения abc.

Формат оператора:

MID$*(текст._перем. ., k[,n]) = символьное_выражение*

k — номер знака в текстовой_переменной, с которого начинается замещение. Если номер k больше длины текстовой переменной, равен нулю или имеет отрицательное значение, то выводится сообщение об ошибке Illegal function call (Неверный вызов функции), n — количество замещаемых символов. Если оно не указано или k + n больше длины текстовой переменной, то замещаются все символы переменной, начиная с k-го, но длина полученной строки не может превышать исходной длины *текстовой_переиённой*. *символьное_выражение* - строка, первые n символов которой внедряются в текстовую переменную. Если символов в выражении меньше n, то используется столько знаков, сколько есть.

Пример 5. Примеры работы оператора MID$.

```
A$ = "123456": MID$(A$, 1, 3) = "abc"                    '1
 PRINT A$
A$ = "123456": MID$(A$, 5, 3) = "abc"                    '2
 PRINT A$
A$ = "123456": MID$(A$, 2) = "abcdefghijk"               '3
 PRINT A$
A$ = "123456": MID$(A$, 4!, 2) = MID$("abc", 2, 1)     '4
 PRINT A$
```

Результат:

abc456

1234ab

1abcde

123b56

Работа программы, В строке '1 три символа, начиная с первого, заменяются тремя символами выражения abc. В строке '2 размещена попытка заменить три символа, начиная с пятого, но в исходной переменной всего 6 знаков, поэтому замещены только пятый и шестой. В строке '3 не указан второй параметр оператора MIDS (количество замещаемых знаков). Программа заменяет все знаки от второго до конца строки. Итоговое значение содержит 6 символов, потому что именно столько знаков содержала переменная A$ в начале работы оператора. В строке '4 использованы оператор и функция MID$. Функция выделяет второй знак в abc, а оператор ставит его на место четвертого знака в переменной A$. Согласно аргументам оператора, замещаться должно два знака, но так как функция возвращает только один, то четвертый символ A$ замещен, а пятый остался без изменений.

Функция INSTR

Функция определяет, входит ли одна текстовая переменная в другую. Если результат положительный, то функция возвращает номер символа, с которого начинается вхождение. В противном случае функция возвращает нуль

INSTR*([k], текст_выраж_1. текст_выраж_2)*

Функция проверяет наличие текстового_выражения_2 в текстовом_выражении_1. Число k задает номер символа, с которого начинается поиск вхождения. Если число не указано, то поиск начинается с первого знака.

Пример 6 . Пример использования функции INSTR

```
A$ = "Кавказ"
B$ ="a"
PRINT INSTR(a$, b$)
PRINT INSTR(3, a$, b$)
```

Результат:

2

5

Работа программы. Первая функция INSTR начинает поиск вхождения, а в слово Кавказ с первого символа и возвращает номер первого вхождения (2). Вторая функция начинает поиск с третьей позиции и возвращает число 5.

Функции LTRIM$ и RTRIM$

Эти функции используются для удаления первых (левых) и последних (правых) пробелов текстового выражения, если начальные или концевые пробелы есть.

LTRIM$*(текстовое_выражение)*

RTRIM$*(текстовое_выражение)*

Функция ASC

Функция возвращает код ASCII, соответствующий указанному знаку.

ASC*(символьное_выражение)*

В качестве символьного выражения можно использовать текст в кавычках, строковую переменную или выражение с текстовым результатом. Функция возвращает ASCII-код только первого знака строки.

Пример 7. Пример использования функций ASC

```
PRINT ASC("A"):
PRINT ASC("AB")
```

Результат:

```
65 65
```

Функция CHR$

Выводит на экран символ, соответствующий определенному номеру в таблице ASCII.

CHR$(n)

Здесь n — код из таблицы ASCII. Он должен попасть в диапазон от 0 до 255, иначе будет выведено сообщение об ошибке *Illegal function call* (Неверный вызов функции).

Базовые алгоритмы обработки текста

С определенной долей условности можно выделить ряд действий, которые чаще других выполняются в программах, работающих с текстовой информацией. К таким действиям можно отнести:

1. **Определение общего количества символов в переменной.**
2. **Выделение символов.**
3. **Анализ символа на принадлежность к логической группе.**
4. **Уменьшение/увеличение текста путем удаления/ввода знаков.**
5. **Разделение текста на отрезки в соответствии с условием.**
6. **Выделение отдельного слова из текста.**
7. **Перестановка элементов текста.**

Каждому действию соответствует группа алгоритмов, которая имеет определенную специфику приложения к разным задачам

Определение количества символов в строке

Длину текстовой переменной вычисляет функция **LEN**. Она возвращает числовое значение целого типа (INTEGER), которое равно общему количеству знаков в анализируемой строке.

Пример 8. Определить количество символов в переменной «Москва»

A$ = “Москва” x = LEN (A$) PRINT X Результат: 6

Функция LEN может быть частью выражения или аргументом оператора. В частности, данную программу можно записать и в одну строку:

PRINT LEN("Москва”)

В качестве аргумента функции LEN можно ввести не только одну текстовую переменную, но и выражение с несколькими переменными и специальными функциями.

Выделение символов

Необходимость выделения определенного символа или символов текстовой переменной диктуется условиями, большого количества задач. Под выделением понимают нахождение знака, вывод его на экран или любой другой

вид обработки. Естественно, что выделяют знаки, удовлетворяющие определенному условию,

Алгоритмы, решающие подобные задачи, как правило, работают в два этапа: сначала выделяют очередной знак переменной, затем анализируют его на удовлетворение условию.

Поочередное выделение всех знаков переменной осуществляется в цикле с помощью функции **MID$.**

Пример 9. Вывести на экран буквы слова «Хорошо» по одной в каждой строке

```
A$ = "Хорошо"              'Задаем переменную
x = LEN(A$)                'и определяем ее длину
FOR i = 1 TO x
tmp$ = MID$(A$, i, 1) 'Присваиваем очередной значение переменной tmp$
PRINT tmp$                 'Выводим знак
NEXT i
PRINT LEN("Москва")
```

Результат

```
Х
о
р
о
ш
о
6
```

Работа программы. Переменной A$ присваиваем строку символов “Хорошо”. Определяем длину слова и сохраняем это значение в переменной x. Задаем цикл, счетчик которого изменяется от 1 до x. В ходе каждой итерации цикла функция MID$ будет присваивать tmp$ очередной (i-й) символ переменной A$. При этом значение самой A$ не изменяется, то есть MID$ не вырезает знак из нее, а просто копирует и присваивает его переменной tmp$. С

очередной итерацией цикла в tmp$ передается очередной знак. При i = 1 передается первый знак, при i - 2 — второй, и так далее. На то, что передастся один знак, указывает третий параметр в списке аргументов MID$.

Пример 10. Определить сколько раз встречается буква «а» в слове «абракадабра»

```
A$ ="абракадабра"
FOR i = 1 TO LEN(a$)
tmp$ = MID$(a$, i, 1)                    'Выделяем букву
IF tmp$ = "a" THEN k = k + 1             'Сравниваем букву с "a"
NEXT i
PRINT k                                  'Выводим результат
```

Результат: 5

Работа программы. В цикле функция MID$ поочередно выделяет буквы из слова абракадабра и присваивает их переменной tmp$. В строке с оператором IF проверяется, содержит ли tnp$ букву а. Если tmp$= "a", то значение счетчика k увеличивается на 1.

Функция MID$ — не единственный инструмент выделения знаков из текстовой переменной. В некоторых случаях

Пример 11.Удалить первый и последний символ текстовой переменной.

```
a$ = " В Россию можно только верить." a$ = LEFT$(a$, LEN(a$) - 1) a$ =
RIGHT$(a$, LEN(a$) - 1) PRINT a$
```

Выделение знаков, номера которых обладают определенными свойствами, происходит с использованием операторов логической передачи управления.

Пример 12. Сформировать B$, состоящую из знаков A$. порядковые номера, которых кратны N.

```
n = 3
a$ = "сиСтеМныЙ бЛок"
FOR i = 1 TO LEN(a$)
tmp$ = MID$(a$, i, 1)
IF i / n = i \ n THEN B$ = B$ + tmp$
```

```
NEXT i
PRINT B$
```

Результат: СМЙЛ!

Работа программы. Программа выделяет каждый третий знак строки A$ и прибавляет его к B$, в которой накапливается текстовая сумма выделенных знаков. Условие выбора описано в операторе IF.

Выделение логической группы

Логической группой называют совокупность знаков, которая объединена общими свойствами. К таким группам можно отнести гласные и согласные буквы русского алфавита, буквы латинского алфавита, цифры, строчные и прописные буквы и многие другие.

Каждый алгоритм анализа символа содержи три основных действия :

1. Выделение.
2. Анализ.
3. Обработка по условию.

Анализ заключается в проверке, соответствует ли знак условию принадлежности к логической группе. Как правило, анализ основан на сравнении выделенного знака с определенным ASCII-кодом или диапазоном номеров.

Обработка в соответствии с решаемой задачей определяет то, что нужно делать с выделенным символом или текстом в целом, и может подразумевать большой спектр действий.

Пример 13. Определить, сколько в слове «АВРОРА» русских букв А и Р.

```
A$ = "АВРОРА"
FOR i = 1 TO LEN(A$)
tmp$ = MID$(A$, i, 1)
IF tmp$ = "А" OR tmp$ = "Р" THEN k = k + 1
NEXT i
PRINT k
```

Результат: 4

Приведенная программа имеет недостаток — она определяет наличие только заглавных A и P. Если значение исходной переменной присваивается оператором INPUT, то есть вводится пользователем, то при вводе строчных букв результат будет неверным.

Пример 14. Определить к какому языку и регистру клавиатуры относится данная буква.

```
INPUT a$
SELECT CASE ASC(a$)
CASE 65 TO 90: PRINT " Лат.ЗАГЛАВНАЯ "
CASE 97 TO 122: PRINT "Лат. Строчная"
CASE 12 TO 159: PRINT "Русс. ЗАГЛАВНАЯ"
CASE 160 TO 175: PRINT "Русс. Строчная"
CASE 224 TO 239: PRINT "Русс. Строчная"
CASE ELSE: PRINT "НЕ является БУКВОЙ"
END SELECT
```

Работа программы. Алгоритм основан на использовании конструкции логической передачи управления SELECT CASE. Она анализирует ASCII-код введенного знака, который выдает функция ASC. В каждом CASE указан диапазон возможных значений, соответствующей группе букв.

Большой спектр задач выделения и анализа фрагментов строк решается с помощью функции **INSTR.**

Пример 15. Определить, входит ли строка B$="dow" в слово, A$ = "Windows".

```
INPUT A$
INPUT B$
k = INSTR(A$, B$)
IF k > 0 AND LEN(B$) > 0 THEN PRINT "Yes" ELSE PRINT "No"
END
```

Результат:

Yes

Работа программы. Алгоритм основан на работе функции INSTR. Результатом ее работы является целое число k равное номеру знака в A$, с которого начинается вхождение B$. Естественно, что если k больше нуля, то B$ входит в A$. Условие в операторе IF состоит из двух компонентов. В одном проверяется значение k, во втором (LEN(B$) > 0) — длина переменной B$.

Второе условие введено для того, чтобы выдать отрицательный ответ в случае ввода нулевой переменной. Иначе при B$="" значение k будет равно 1, и программа определит вхождение.

Пример 16. Определить, сколько раз B$ входит в A$.

```
A$ = "промышленное производство проводов"
B$="про"
i = 1
k = 0
DO
x = INSTR(i, A$, B$)
IF x > 0 THEN
i = x + LEN(B$)                ' Номер начала поиска
k = k + 1                      ' Счетчик вхождений
ELSE
EXIT DO                        ' Выход из цикла
END IF
LOOP
PRINT "всего вхождений="; k
```

Результат:

```
Всего вхождений= 3
```

Работа программы. Цикл DO …LOOP использован потому, что заранее неизвестно, сколько будет вхождений.

В строке i = x + LEN(B$) номер начала поиска передвигается направо. Если программа найдет первое вхождение, то нужно будет продолжить поиск с

позиции, номер которого равен номеру начала вхождения плюс длина искомого слова.

В примере первое вхождение «про» в «промышленное производство проводов» начинается с первой позиции, следователь но, продолжать поиск нужно с 4-й позиции (1 + LEN(B$)=1+3=4).

Если обнаруженное вхождение нужно заменить другой строкой, то используют оператор MID$.

Пример 17. Заменить все пробелы на тире в тексте A$.

```
A$= "Погиб поэт! невольник чести!"
FOR i = 1 TO LEN(A$)
IF ASC(MID$(A$, i, 1)) = 32 THEN MID$(A$, i, 1) = "-"
NEXT i
PRINT A$
```

Результат:

Погиб-поэт!-невольник-чести!

Работа программы. В цикле поочередно выделяются знаки переменной A$. Функция ASC определяет ASCII-код знака, но в том случае, если номер равен 32 (ASCII-код пробела), оператор MID$ заменяет текущий символ знаком «минус» (тире).

Сортировка текстовых массивов

Пример 18. Дан массив текстовых переменных. Отсортировать по всем знакам каждого слова в соответствии с алфавитом.

```
CLS
DATA "512","101","324","712","310","520","001","721" REM Формируем
исходный массив A$ n = 8
DIM a$(n)
PRINT "Исходный массив"
FOR i = 1 TO n
READ a$(i)
PRINT " "; a$(i);
```

```
NEXT i
PRINT
REM Сортировка
FOR i = 1 TO n
FOR j = 1 TO n k = 1
IF ASC(MID$(a$(i), k, 1)) < ASC(MID$(a$(j), k, 1)) THEN
 SWAP a$(i), a$(j) ELSE
 IF ASC(MID$(a$(i), k, 1)) = ASC(MID$(a$(j), k, 1)) THEN
 k = k + 1
 END IF
IF MID$(a$(j), k, 1) = "" THEN SWAP a$(i), a$(j)
NEXT j
NEXT i
REM вводим отсортированный массив на экран
PRINT "Отсортированный"
FOR i = 1 TO n
PRINT " "; a$(i);
NEXT i
```

Результат:

Исходный массив 512 101 324 712 310 520 001 721

Отсортированный 001 101 310 324 520 512 712 721

Для других исходных данных:

Исходный

Яковлев А. Андреев Д., Сланов Р., Ананко Р., Ваниёва Р., Антонов И.. Шейхов М, Гринберг Л., Галич Г., Сапин В.

Отсортированный

Ананко Р., Андреев Д., Антонов И.., Ваниева Р., Галич Г., Гринберг Л., Санин В., Сланов Р., Шейхов М., Яковлев А.

Работа программы. Алгоритм, как и многие программы сортировки, основан па сравнении слов во вложенных циклах. Внешний цикл со счетчиком i

задает номер слова для анализа. Внутренний цикл со счетчиком j сравнивает это слово со всеми остальными и при необходимости меняет его на другое. Логической группой может быть не только язык или тип символов, но и отдельная переменная.

Пример 19. Состоит ли B$ только из символов, входящих в A$.

По количеству знаков переменные не равны.

```
a$ = "123456"
b$ = "654"
DO
k = k + 1
tmp$ = MID$(b$, k, 1)
IF INSTR(a$, tmp$) = 0 THEN rez = 1: EXIT DO
LOOP WHILE k < LEN(b$)
IF rez = 1 THEN PRINT "НЕ состоит" ELSE PRINT " Состоит "
```

Результат: Состоит

Работа программы. Идея алгоритма состоит в том, что каждый элемент BS проверяется на вхождение в AS. Если обнаружено хотя бы одно несовпадение, проверка закапчивается и машина выводит ответ о том, что B$ не состоит только из символов A$. На основе подобного алгоритма можно решать любые задачи, связанные со сравнением одной строки символов с группой других. Например, поиск слова или числа с неповторяющимися знаками, поиск символа, который реже или чаще; других встречается в тексте, определение слова с наибольшим содержанием заданного знака.

Изменение текста путем вставки или удаления знаков

В Qbasic нет операторов, которые позволяют вставить новые элементы, раздвинув слово. Поэтому вставка символов осуществляется путем составления новой переменной, которая формируется из следующих частей исходного слова: первая_частъ + вставка + вторая часть.

Например, для того чтобы в переменную 0$="AC" вставить букву B. нужно создать новую переменную A$="A" I "B"+"C".

Такая же логика применяется в задачах **поиска и замены.** В них нужно не только выделить определенную последовательность символов, по и заменить ее другой, не изменив структуры исходной переменной. Причем таких замен для одной строки может быть несколько, и реализуются они в цикле.

Удаление части исходной строки проходит аналогично. Программа формирует новую переменную, в которую записываются все символы исходной строки, кроме удаляемых. Структуру новой переменной можно представить формулой:

символы_до_удаляемых + символы после_удаляемых.

Например, чтобы из переменной A$= "ABC" удалить B. Нужно создать новую переменную G$="A"+"C".

Рассмотрим ряд примеров вставки и удаления фрагментов текста.

Пример 20. Разделить цепочку литер «QuickBasic 4.5», вставив, робел между буквами «k» и «B». (Примем во внимание, что k - это 5-й по счету символ слева, а B - 9-й символ справа в A$).

```
a$ = "QuickBASIC 4.5"
b$ = LEFT$(a$, 5) + "_" + RIGHT$(a$, 9)
PRINT b$
```

Результат:

```
Quick_BASIC 4.5
```

Работа программы. В переменную B$ записывается сумма первой части A$, которая должна быть расположена слева от пробела, знак «пробел» и вторая часть исходной цепочку, которая должна стоять справа от пробела.

Особого внимания заслуживают задачи, в которых переменную необходимо изменять несколько раз в зависимости от условия.

Эти алгоритмы предполагают использование цикла и операторов логической передачи управления, хотя принцип действия остается неизменным — формирование новой переменной путем сложения частей

Пример 21. В тексте A$ после каждого пробела вставить B$.

```
a$= "Белеет парус одинокий"
```

```
PRINT a$
B$ = "11"
x = LEN(a$)
FOR i = 1 TO x
IF ASC(MID$(a$, i, 1)) = 32 THEN
a$ = MID$(a$, 1, i) + B$ + MID$(a$, i + 1, x)
END IF
NEXT i
PRINT
PRINT a$
```

Результат. Белеет 11парус 11одинокий

Работа программы. В цикле анализируется ASCП-код очередного выделенного символа на равенство 32 (код пробела). Если равенство выполняется в ходе 1-й итерации цикла, то переменной A$ присваивается сумма трех компонентов — левая часть исходной A$ (знаки с номерами от 1 до 1) + B$ + правая часть AS (знаки от 1 до x).

Удаление символов

Удаление символов проходит путем записи в новую текстовую переменную знаков исходной строки, расположенных до удаляемой части и после нее.

Пример 22. Удалить пятый символ из A$.

```
a$ = "123456789"
n = 5
x = LEN(a$)
a$ = MID$(a$, 1, n - 1) + MID$(a$, n + 1, x)
PRINT a$
```

Результат: 12346789

Работа программы. Первое значение A$ образуется путем сложения знаков 1-4 (1, N - 1) и 6-9 (N+1, x) исходной строки. Если удаляется не один

знак, а целый блок знаков BS, то во втором MID$ номер первого элемента правой части запишется N + LEN(B$) и команда в целом будет иметь вид

MID$(A$, N+LEN(B$)

Рассмотрим алгоритм многократного удаления знака по условию.

Такие задачи решаются с помощью цикла.

Пример 23. Удалить все запятые в строке A$.

```
a$ = ",,,1,2,3,4,5,,,,"
PRINT a$ x = LEN(a$)
FOR i = 1 TO x
IF MID$(a$, i, 1) = "," THEN
a$ = MID$(a$, 1, i - 1) + MID$(a$, i + 1, x)
i = i – 1                                    'Уменьшаем счетчик i
END IF
 NEXT i
 PRINT a$
```

Результат: ,,,1,2,3,4,5,,,, 12345 *Работа программы.* В значение A$ специально введено несколько блоков идущих друг за другом запятых. Это сделано для того, чтобы продемонстрировать частный случай задач на удаление. Если удаляются знаки, стоящие последовательно по одному, то необходимо уменьшать счетчик цикла па количество удаляемых знаков.

Выделение отдельного слова из текста

Текст состоит из отдельных слов. Слово – это последовательность букв, заключенных между двумя символами, не являющимися буквами.

Алгоритмы выделения слов, как правило, основаны на поиске знаков – разделителей. Ими могут быть пробелы, знаки пунктуации, специальные символы.

Пример 25. Определить количество слов в тексте.

```
a$ = " Отговорила роща золотая Березовым, веселым языком "
' выделяем i-й знак и проверяем буква ли это
FOR i = 1 TO LEN(a$)
```

```
tmp$ = MID$(a$, i, 1): isletter = 0
IF 128 <= ASC(tmp$) AND ASC(tmp$) <= 175 THEN isletter = 1
IF 224 <= ASC(tmp$) AND ASC(tmp$) <= 239 THEN isletter = 1
IF 65 <= ASC(tmp$) AND ASC(tmp$) <= 90 THEN isletter = 1
IF 97 <= ASC(tmp$) AND ASC(tmp$) <= 122 THEN isletter = 1
' Если это буква, то формируем слово
' Иначе переходим к формированию след слова
IF isletter = 1 AND i < LEN(a$) THEN
T$ = T$ + tmp$
ELSE
IF LEN(T$) > 0 OR isletter = 1 THEN
IF i = LEN(a$) AND isletter = 1 THEN
T$ = T$ + tmp$ ' Теперь можно проводить анализ очередного слова T$ PRINT T$
L = L + 1
T$ = ""                                    ' Обнулять T$ обязательно
END IF
END IF
NEXT i
PRINT " Всего слов "; L
```

Результат: Отговорила роща золотая Березовым веселым языком Всего слов 6

Работа программы. Приведенный алгоритм выполняет следующие операции: выделяет очередной символ; если это буква, то начинает накапливать очередное слово; если это не буква — переходит к анализу полученного слова и формированию следующего.

Четыре оператора IF в начале цикла предназначены для определения принадлежности выделенного знака к буквам русского или английского алфавита. Действие операторов основано на анализе ASCII-кода каждого знака. Если знак является буквой, то переменной isletter присваивается значение 1, в противном случае — 0.

Пример 26. Сформировать массив слов Word$, входящих в текст A$. Определить слова с максимальным и минимальным количеством букв.

```
CLS 'Очистка экрана
DIM word$(LEN(a$))
a$ = "Еще светло перед окном, В разрывы облак солнце блещет"
min = LEN(a$)
FOR i = 1 TO LEN(a$)
' выделяем i-й знак и проверяем буква ли это
…
' Если это буква, то формируем слово
…
'Теперь можно проводить анализ очередного слова T$
K = K + 1: word$(K) = T$ 'формируем массив слов
' находим слово с max кол-вом букв
IF LEN(T$) > max THEN max = LEN(T$): wmax$ = T$
' находим слово с min кол-вом букв
IF LEN(T$) < min THEN min = LEN(T$): wmin$ = T$
T$ = ""                          ' Обнуляем T$
END IF
END IF
NEXT i
PRINT "Слово с кол букв max "; wmax$
PRINT " Слово с кол букв min "; wmin$
PRINT "Массив слов"
FOR i = 1 TO K
PRINT word$(i)
NEXT i
END
```

Результат

Слово с кол букв max разрывы

Слово с кол букв min B

Массив слов

Еще

светло

перед

окном,

B

разрывы

облак

солнце

блещет

Работа программы. Алгоритм выделения слова из предыдущего примера приведен сокращенно. В первой сроке программы происходит объявление массива Word$. Рассчитывать, сколько нем будет слов не рационально. Для этого пришлось бы анализировать исходную строку и, получив количество слов, и объявить массив, а для его заполнения выполнить алгоритм заново. Поэтому предполагаем, что в массиве Word$ Len(a$) элементов.

Перестановка элемента в тексте

Перестановка элементов текста осуществляется путем формирования новой переменной на основе исходной. Можно переставлять не только отдельные слова, но и логические группы, блоки.

Алгоритмы перестановки текста состоят из двух основных этапов:

выделение элементов

формирование новой строки в соответствии с условиями. Например, для того чтобы поменять местами самое длинное и самое короткое слова, нужно сначала их найти, а затем в ходе формирования новою текста вместо первого записать второе, а вместо второго первое. Последовательность всех остальных знаков при этом не изменится.

Пример 27. Поменять местами самое длинное и самое короткое слова в A$.

CLS

```
a$ = "Белеет парус одинокий" PRINT a$ min = LEN(a$) 10 : FOR i = 1 TO
LEN(a$) ' выделяем i-й знак и проверяем буква ли это tmp$ = MID$(a$, i, 1):
isletter = 0
IF 128 <= ASC(tmp$) AND ASC(tmp$) <= 175 THEN isletter = 1 IF 224 <=
ASC(tmp$) AND ASC(tmp$) <= 239 THEN isletter = 1 IF 65 <= ASC(tmp$) AND
ASC(tmp$) <= 90 THEN isletter = 1 IF 97 <= ASC(tmp$) AND ASC(tmp$) <= 122
THEN isletter = 1 ' Если это буква, то формируем слово IF isletter = 1 AND i <
LEN(a$) THEN T$ = T$ + tmp$ ELSE
IF LEN(T$) > 0 OR isletter = 1 THEN
 IF i = LEN(a$) AND isletter = 1 THEN T$ = T$ + tmp$: tmp$ = "" 'Теперь можно
проводить анализ очередного слова T$ ' max
IF LEN(T$) > max THEN max = LEN(T$): wmax$ = T$ 'min IF LEN(T$) < min
THEN min = LEN(T$): wmin$ = T$
' Блок замены и прибавление слова rez$
IF switsch = 1 THEN
IF T$ = long$ AND c <> i THEN T$ = short$: c = i
IF T$ = short$ AND c <> i THEN T$ = long$: c = i
rez$ = rez$ + T$
END IF                    ' закрываем switsch
T$ = ""                   ' обнуляем T$
END IF                    ' закрываем Len(T$)>0 or....
IF switsch = 1 THEN
rez$ = rez$ + tmp$        ' прибавляем не букву
```

```
45
END IF                          ' закрываем switsch
END IF                           ' закрываем islitter
NEXT i
' возвращаемся в начало программы после 1-го прохода
IF short$ = "" THEN
switsch = 1: short$ = wmin$
long$ = wmax$: GOTO 10
ELSE
PRINT rez$                      ' вывод результата
END IF
END
Белеет парус одинокий
Белеет одинокий парус
```

Работа программы. Алгоритм работает в два прохода, В первом проходе определяет самое длинное (wmax$) и самое короткое (wmin$) слово. Затем управление оператором GOTO передается в начало программы и осуществляется второй проход да является формирование новой переменной rez$, в которой накапливается результат, — строка с переставленными максимальным и минимальным словами.

На основе подобных алгоритмов решаются задачи перестановки или вставки блоков, а также некоторых видов сортировки элементов строки.

Сортировка массива слов **Сортировка** — один из самых сложных видов обработки текстовых переменных, а необходимость перестановки элементов в рамках одной строки усложняет задачу еще и тем, что требует сохранения последовательности знаков, не участвующих в перемещении. К таким элементам относятся числа, знаки пунктуации, пробелы.

Пример 28. Переставить слова в A$ согласно алфавиту. Количество и порядок следования других знаков остается неизменным.

```
CLS
```

```
A$ = "август - астры,,,,, - звезды - грозди винограда -- рябины??????"
PRINT A$
DIM word$(LEN(A$))
FOR i = 1 TO LEN(A$)
' выделяем i-й знак и проверяем буква ли это
…
' Если это буква, то формируем слово
…
' Теперь можно проводить анализ очередного слова T$
k = k + 1
      word$(k) = T$ ' формируем массив слов
T$ = ""                        ' обнуляем T$
END IF END IF NEXT i
' сортировка массива word$ по росту ASCII-кодов
FOR i = 1 TO k
FOR j = 1 TO k
f = 1
10 : IF ASC(MID$(word$(i), f, 1)) < ASC(MID$(word$(j), f, 1)) THEN
  SWAP word$(i), word$(j) ELSE
IF ASC(MID$(word$(i), f, 1)) = ASC(MID$(word$(j), f, 1)) THEN
f = f + 1
 IF MID$(word$(i), f, 1) <> "" AND MID$(word$(j), f, 1) <> "" THEN GOTO 10
END IF
END IF
END IF
NEXT j
      NEXT i PRINT
PRINT "Сортировка массива слов"
FOR i = 1 TO k
PRINT " "; word$(i);
```

```
NEXT i
END
```

Результат: август - астры,,,,, - звезды - грозди винограда -- рябины??????

Сортировка массива слов август астры винограда грозди звезды рябины

Работа программы. Программа анализирует исходную строку A$. Формирует массив word$ только слов. Переставляет слова в массиве в порядке возрастания ASCII-кодов букв.

Контрольные вопросы

1. Какие переменные называют символьными?
2. Где используются переменные текстового типа?
3. Что могут содержать текстовые переменные?
4. В чем состоит формат ASCII ?
5. Какие типы строковых переменных Вы знаете?
6. Какими способами может быть осуществлении присвоение значения текстовой переменной?
7. Какой знак используется для сложения текстовых переменных?
8. С помощью, каких знаков производится сравнение текстовых переменных?
9. В каком случае текстовые переменные считаются равными?
10. Какой принцип используется при сортировке переменных?
11. Перечислите функции работы с символьными переменными?
12. Что выполняет оператор MID$?
13. Что выполняет функция LEN?
14. Что выполняют функции LEFT$ и RIGHT$?
15. Что выполняет функция INSTR$?
16. Для чего предназначается функция MID$?
17. Что называют логической группой?
18. В чем заключается анализ символа?
19. На чем основаны алгоритмы выделения слов?
20. Как осуществляется перестановка элементов текста?

Варианты заданий для самостоятельного решения

1. Введите строку A$ и определить, сколько в ней 1) букв русского алфавита; 2) заглавных латинских букв; 3) цифр.
2. Введите строки A$ и B$. Определите, сколько раз B$ входит в A$.
3. Введите строки A$ и B$. Удалите из A$ все вхождения B$.
4. Упорядочить по алфавиту вектор A$(10), состоящий из десяти любых.

1). латинских букв; 2). русских букв; 3). цифр.

5. Упорядочьте в обратном алфавитном порядке вектор A$(10). Каждый элемент анализировать по всем знакам.
6. Введите переменную A$. Замените вес сочетания C$ на B$.
7. Введите текст A$. Считая словом непрерывную последовательность букв, выделите все слова A$ в отдельный массив.
8. Введите текст A$. Удалите все слова, которые встречаются больше двух раз, оставив только первое вхождение каждого такого слова.
9. Введите название переменной как текстовую строку. Проверьте, соответствует ли ее название требованиям QBasic.
10. Дана строка, содержащая английский текст. Найти количество слов, начинающихся с буквы B.
11. .Дана строка. Подсчитать, сколько в ней букв g, k, t.
12. Дана строка, содержащая текст. Найти длину самого короткою слова и самого длинного слова.
13. Дана строка, содержащая текст, закапчивающийся точкой. Вывести на экран слова, содержащие три буквы.
14. Дана строка. Подсчитать самую длинную последовательность подряд идущих букв *а.*
15. Дана строка. Указать тс слова, которые содержат хотя бы одну букву *к.*
16. В строке между словами вставить вместо пробела запятую и пробел.
17. Определить, сколько раз в строке встречается заданное слово.
18. Дана строка. Преобразовать ее, заменив точками все двоеточия (:),

19. Строка содержит произвольный русский текст. Проверить, каких букв в нем больше: гласных или согласных.

20. Результаты вступительных экзаменов представлены в виде списка из N строк, в каждой строке которого записаны фамилия студента и отметки по каждому из M экзаменов. Определить количество абитуриентов, сдавших вступительные экзамены только на «отлично».

21. Для заданного текста определить длину содержащейся в нем максимальной серии символов, отличных от букв.

22. Отредактировать заданное предложение, удаляя из него все слова с нечетными номерами.

ЛАБОРАТОРНАЯ РАБОТА № 6

Оптимизация технологического процесса.

Методы оптимизации функции 1-ой переменной

Цель работы

Ознакомление с методами одномерной оптимизации (поиска максимума и минимума).

Оптимизация функций одной переменной

Постановка задачи

На практике часто возникает задача нахождения экстремума некоторой целевой функции F(x). Такая функция одного параметра x описывает некоторую кривую на плоскости.

На определённом интервале функция может иметь одно (рис. 1а) или несколько экстремальных значений (рис. 1б).

Функция, изображённая на рис. 1а называется унимодальной.

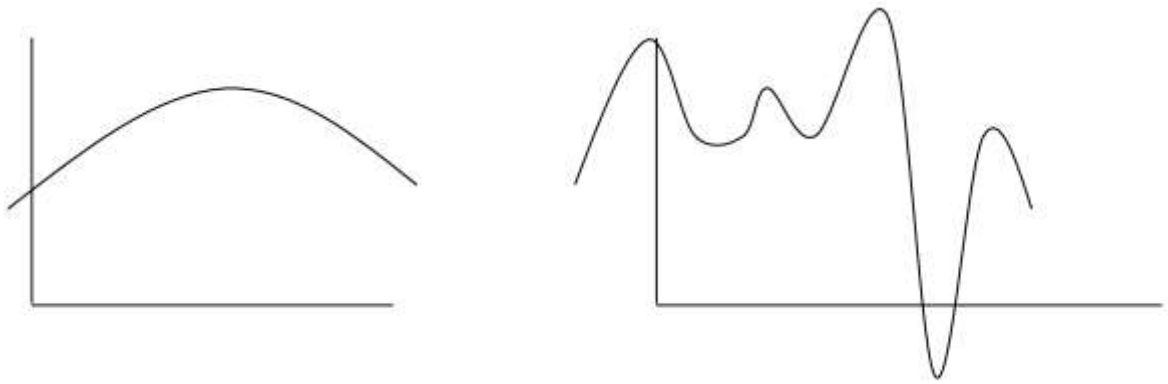


Рис. 1. Функции одного параметра x описывающие некоторую кривую на плоскости.

Функция на рис. 1б имеет несколько экстремумов (максимумов или минимумов). Из них главный (оптимальное решение для рассматриваемого интервала) называется глобальным.

Унимодальная функция не обязательно должна быть гладкой (рис. 2 а), она может быть ломаной (рис. 2б), разрывной (рис. 2в).

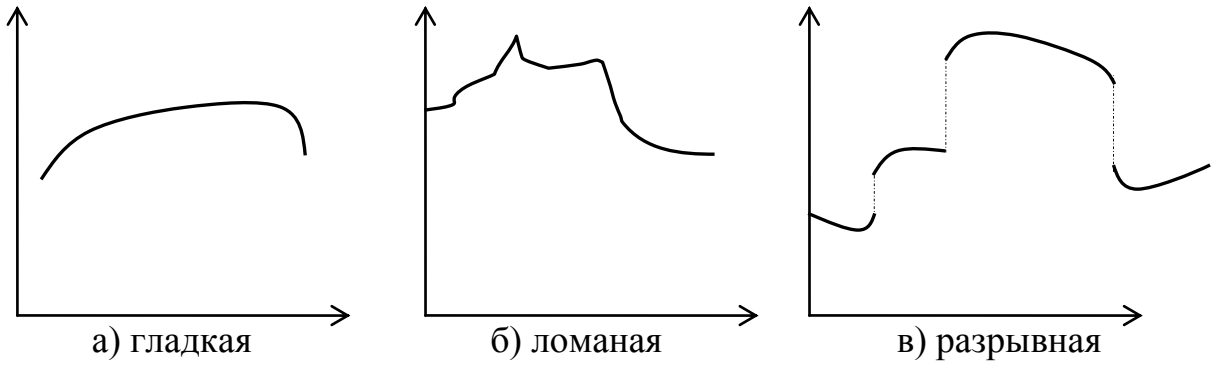


Рис. 2. Унимодальные функции

Рассматриваемые методы справедливы для функций на рисунке 2а, а для функций на рисунках 2б, 2в могут использоваться на отдельных интервалах.

Если целевая функция унимодальная, то можно сузить интервал исследования функции на оптимум путём определения значений целевой функции в двух точках интервала задания функций $F(x_1)$ и $F(x_2)$ и последующего поинтервального сравнения. При этом возможны три случая (рис. 3):

1) если $F(x_1) > F(x_2)$,то $x_{опт} < x_2$, т.е. оптимум не может находиться правее, интервал $[x_2, x]$ из дальнейшего рассмотрения исключается

2) если $F(x_1) < F(x_2)$, то $x_{опт} > x_1$

3) если $F(x_1) = F(x_2)$, то $x_1 < x_{опт} < x_2$

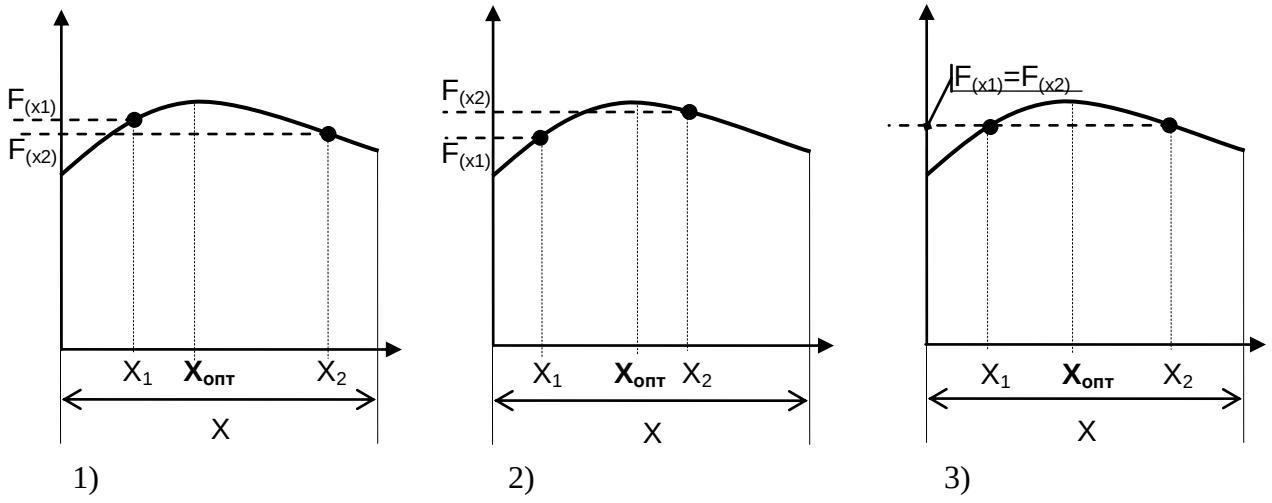


Рис. 3. Определение значений целевой функции в двух точках интервала задания функций

Последовательно сужая интервал исследования, в котором находится оптимальное значение функции, можно с достаточной степенью точности найти оптимальное значение переменной.

Задача поиска экстремумов сводится к их локализации и уточнению значений x и F(x) в точке экстремума. В дальнейшем для функций одной переменной под экстремумом будем подразумевать максимум F(x).

Поскольку максимуму функции F(x) соответствует минимум функции -F(x), то, сменив знак у F(x), программами поиска максимума можно пользоваться и для поиска минимума функций. Будем также полагать, что на изменения x (если это особо не оговорено) накладываются ограничения в виде неравенств a < x < b, где a и b – границы интервала поиска. В пределах отрезка [a , b] функцию считаем унимодальной, т.е. содержащей один максимум.

С помощью численных методов мы непосредственно находим максимум (минимум) функции F(x) в некотором интервале, в котором, как предполагается, лежит максимум (минимум). Иногда это единственно возможная стратегия поиска.

На пример, стоимость проведения химического процесса может зависеть от температуры процесса. Инженер знает, что стоимость является функцией от T, хотя может и не знать явного вида функции. Однако он может поставить эксперимент и провести эксперимент при различных температурах и, следовательно, найти стоимость для этих температур и определить минимальную стоимость и температуру проведения процесса, при которой она достигается.

Алгоритм нахождения максимума функции

Простейшая задача нахождения максимума функции решается по следующему алгоритму:

1. Задаются границы a и b, в пределах которых имеется максимум функции.
2. Интервал [a,b] разбивается на определенное количество шагов.

3. Функция табулируется в пределах заданного интервала, и каждое вычисленное значение функции сравнивается с максимальным (заданным до начала табулирования).

4. Находится максимальное значение функции на заданном интервале с определенным шагом и выводится на печать.

БЛОК-СХЕМА АЛГОРИТМА ИМЕЕТ ВИД:

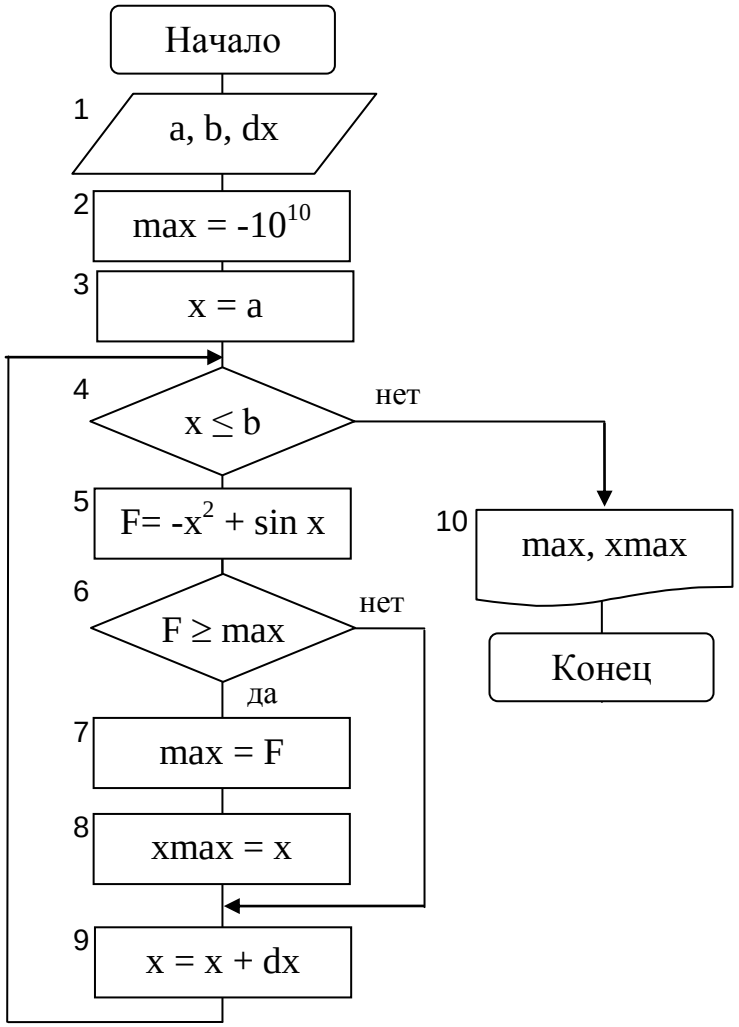


Рис. 4. Блок – схема алгоритма нахождения максимума функции

Естественно, что с уменьшением шага изменения аргумента точность вычисления максимума увеличивается.

Можно воспользоваться и следующим алгоритмом:

1. Найти значение максимума по алгоритму, представленному выше.
2. Для дальнейшего рассмотрения выбрать отрезок [xmax-dx, xmax+dx] и выполнить вычисление максимума с шагом, например, dx/10.

3. Сравнить два найденных значения.

max1 – значение максимума с шагом dx и max2 – значение максимума с шагом dx/10. Если |max2-max1| <=E (где E – заданная степень точности вычисления), то закончить решение задачи и max2 вывести на печать, если нет, то вычисление продолжить дальше, повторяя п.2.

Такую задачу удобнее решать, используя процедуру нахождения максимального значения функции.

Блок – схема решения задачи имеет вид:

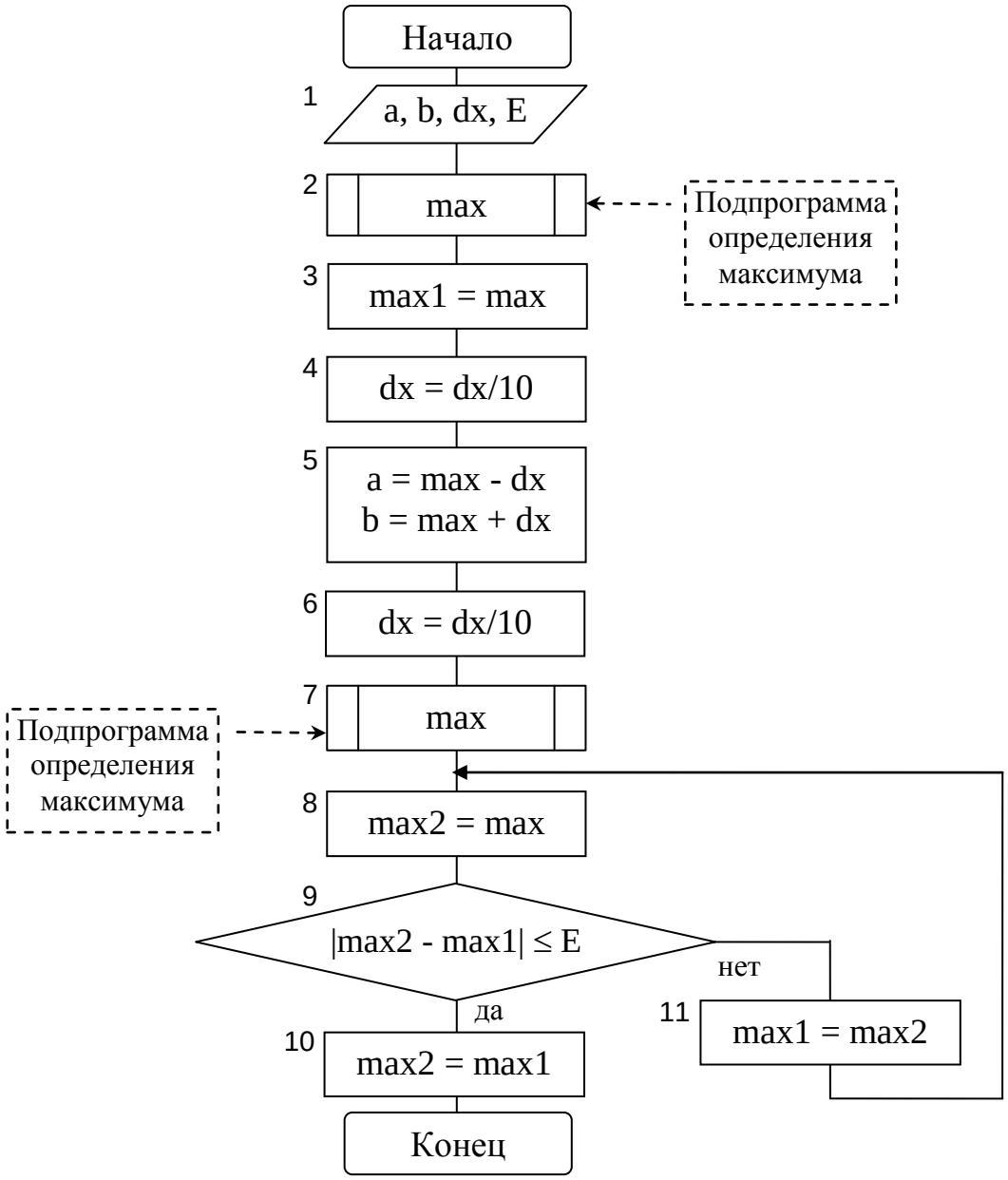


Рис. 5. Блок – схема алгоритма нахождения максимума функции

Методы оптимизации функций одной переменной

Метод равномерного поиска

Этот метод основан на том, что переменной x присваиваются значения x+Δx с шагом Δx = const и вычисляются значения F(x). Если $F(x_{n+1})>F(x_n)$, переменной x даётся новое приращение. Как только $F(x_{n+1})$станет меньше F(x) поиск останавливается. При малой заданной погрешности этот метод неэкономичен по затратам машинного времени.

Метод поразрядного приближения

Этот метод является разновидностью метода равномерного поиска и реализуется следующим алгоритмом:

1. Задаём начальное приближение $x=x_0$ слева от максимума F(x) и вычисляем $F(x_0)$. Задаём D=h, где h=Δx – начальный шаг поиска.
2. Полагаем, что $G=F(x_n)$, где вначале $F(x_n)=F(x_0)$, задаём x=x+D и вычисляем $F(x_{n+1})=F(x)$.
3. Проверяем условие $F(x_{n+1})>G$; если оно выполняется, идём к п.2, если нет, то к п.4.
4. Полагаем D=-D/4. Проверяем условие |D|>E/4, где E – заданная погрешность вычисления x_n в точке максимума. Если оно выполняется, идём к п.2, т.е. обеспечиваем поиск максимума в другом направлении с шагом в 4 раза меньше прежнего. Если данное условие выполняется, процесс вычисления заканчиваем.

Метод дихотомии

Метод дихотомии (деление интервала поиска [a, b] пополам) реализуется следующим алгоритмом:

1. Проверяем условие $|b-a|<2E$, где E – заданная погрешность вычисления x_n. Если это условие выполняется, идём к п.6; если не выполняется, идём к п.2.
2. Делим интервал поиска [a, b] пополам и вычисляем две абсциссы, симметрично расположенные относительно точки

$$x=(a + b)/2$$

$$x_1=(a + b - E)/2 \text{ и } x_2=(a + b + E)/2$$

3. Для этих значений x вычисляем $F(x_1)>F(x_2)$.

4. Проверяем условие $F(x_1)>F(x_2)$. Если оно выполняется, полагаем $b=x_2$ и идём к п.1. Если не выполняется, идём к п.5.
5. Полагаем $a=x_1$ и идём к п.1.
6. Выводим на печать $x_n=(a+b)/2$ и вычисляем $F(x_n)$.

Метод Фибоначчи

В методе Фибоначчи точка деления интервала исследования определяется с каждым новым расчётом (в методе дихотомии необходимо на каждом шаге выполнять два расчёта). В интервал исследования попадет предыдущий расчёт и для продолжения поиска достаточно произвести расчёт симметрично имеющемуся.

Допустим, задано число расчётов (шагов) N. Необходимо их произвести так, чтобы интервал, в котором лежит оптимум, был минимальным. Числа Фибоначчи, используемые в этом методе, определяются следующим образом:

$$F_N=F_{N-1}+F_{N-2}$$

$$F_0=F_1=1$$

Алгоритм метода Фибоначчи состоит из следующих этапов:

1) Изменяют масштаб исходного интервала, в котором лежит оптимум. В качестве единицы измерения принимают $1=X_o/F_N$, или если задана длина l, в котором лежит оптимум, находят его на исходном интервале длиной X_o. Для этого, разделив X_o на l, находят ближайшее большее число Фибоначчи F_N, а по нему определяют N – число необходимых расчётов для определения интервала.
2) Расставляют первые две точки $X_1^{'}$ и $X_1^{''}$ на интервале исследования X_0 на расстоянии F_{N-2} от конца b.
3) Вычисляют значение целевой функции в этих точках для сужения интервала исследования. Пусть $Y_1^{'} > Y_1^{''}$, тогда интервал $[X_1^{'}, F_N]$ исключается из рассмотрения.
4) На новом интервале исследования снова расставляют две точки $X_2^{'}$ и $X_2^{''}$, но в одной из них уже известно значение целевой функции $Y_2^{'} = Y_1^{'}$.

5) Переходят к этапу 3 и т.д., пока не достигают искомого интервала, в котором находится значение переменной, максимизирующее её целевую функцию.

На рис. 6 показан процесс сужения интервала исследования:

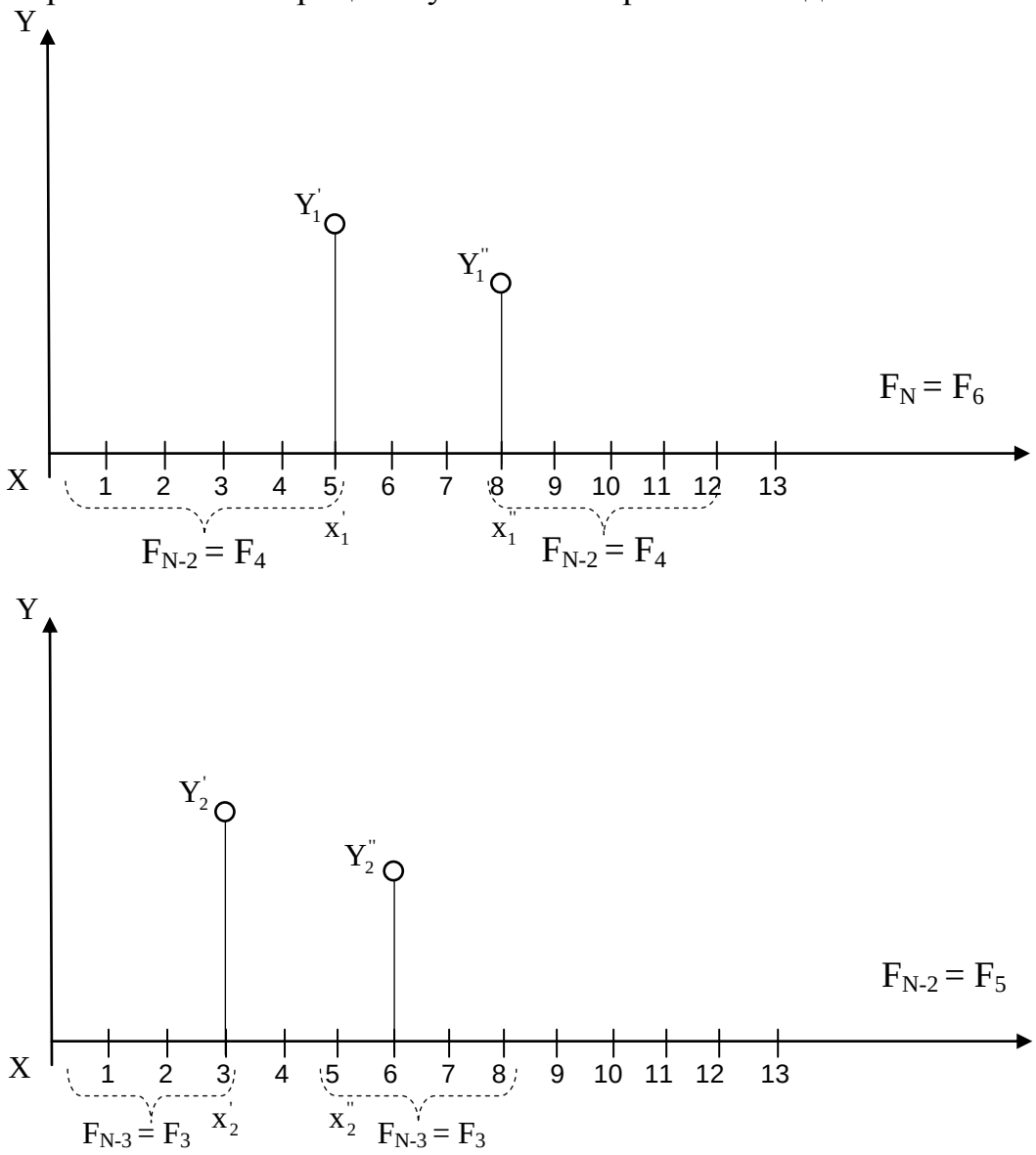
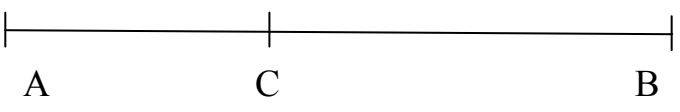


Рис. 6. Процесс сужения интервала исследования.

Последний N–й расчёт определяет интервал длиной l, в котором находится экстремум целевой функции.

Метод золотого сечения

Золотое сечение проводит деление отрезка AB на две неравные части так, чтобы было справедливо соотношение (рис. 7).



$$\frac{AB}{CB} = \frac{CB}{AC} \quad k = \frac{\sqrt{5}-1}{2} = 0{,}61803$$

Рис. 7

Метод золотого сечения позволяет сужать отрезок [a, b] каждый раз вычисляя лишь одно значение F(x), а не два, как в методе дихотомии.

Данный метод реализуется следующим алгоритмом:

1. Находим коэффициент дробления $k=(\sqrt{5}-1)/2$ отрезка [a, b].
2. Находим абсциссу $x_1=a + (1-k)*(b-a)$ и вычисляем $F(x_1)$.
3. Находим абсциссу $x_2=a + k*(b-a)$ и вычисляем $F(x_2)$.
4. Проверяем выполнение условия $|x_2-x_1|<E$, где E – заданная погрешность вычисления x_n. Если это условие выполняется, вычисляем $x_n = (x_1+ x_2)/2$ и $F(x_n)$, после чего останавливаем счёт с выдачей значений x_m и $F(x_n)$. Если данное условие не выполняется, идём к п.5.
5. Проверяем условие $F(x_1) < F(x_2)$. Если оно выполняется, полагаем, $a = x_1$, $x_1 = x_2$ и $F(x_1) = F(x_2)$, после чего идём к п.3. и п.4.

   Если $F(x_1) \geq F(x_2)$, полагаем $b = x_2$, $x_2 = x_1$, $f(x_1) = f(x_2)$, после чего выполняем п.2 и п.4

<u>Использование ППП Eureka и Excel при решении задач оптимизации</u>

Использование ППП **Eureka** для поиска экстремумов функций одной переменной.

Для поиска максимумов функций одной переменной необходимо в окне Edit набрать

$|__|max(F)

y(x)=

F=y(x)

В окне Solution будет выдано решение

F=

x=

Перед решением задачи весьма полезно оценить вид функции, экстремум которой необходимо найти и уточнить интервал x, в котором этот экстремум находится. Для этого достаточно воспользоваться командой Plot в позиции Graf основного меню. Из вида графика сделать вывод о правильности решения.

Использование ППП **Excel** для поиска экстремумов функций одной переменной.

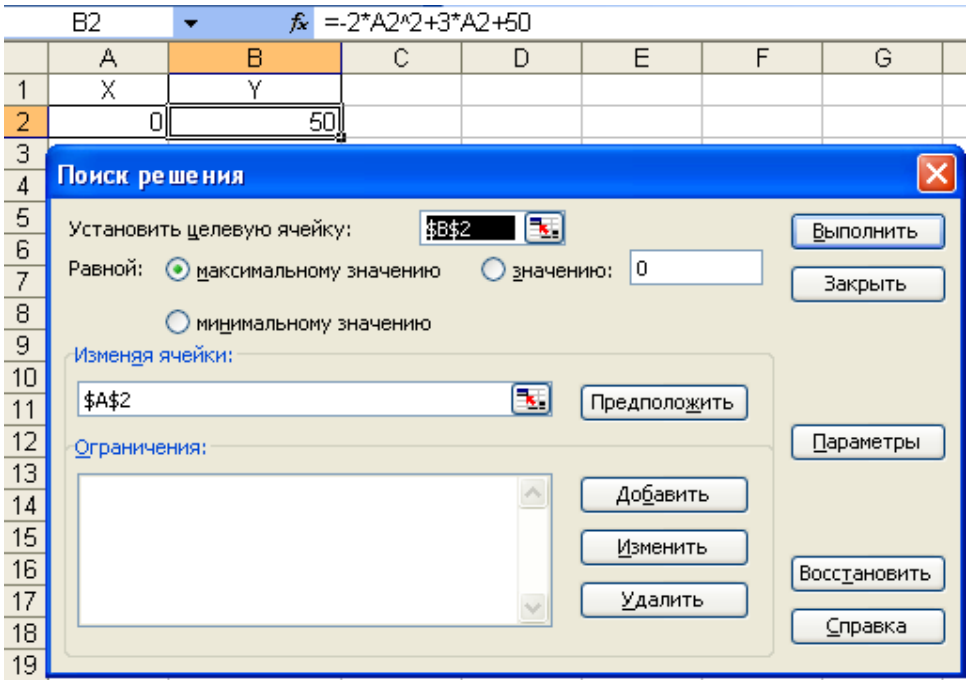
Для поиска максимумов функций одной переменной необходимо:

Вызвать **Подбор параметра**, с помощью команды в меню **Сервис.** Окно Подбор параметра состоит из трех полей:

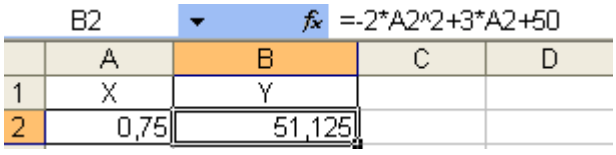
- *Установить целевую ячейку*, в котором ставится ссылка на ячейку с формулой (Y);

- *Равной* – выбираем максимальному значению;

- *Изменяя ячейки*, в которой ставится ссылка на ячейку с изменяемым параметром (первая граница, а интервала (а,в)).



После нажатия кнопки ОК, появляется окно, Результаты поиска решения, сохраняем найденное решение. Полученное решение:



Содержание отчета

1. Содержательная постановка задачи.
2. Исходные данные.
3. Краткое описание методов.

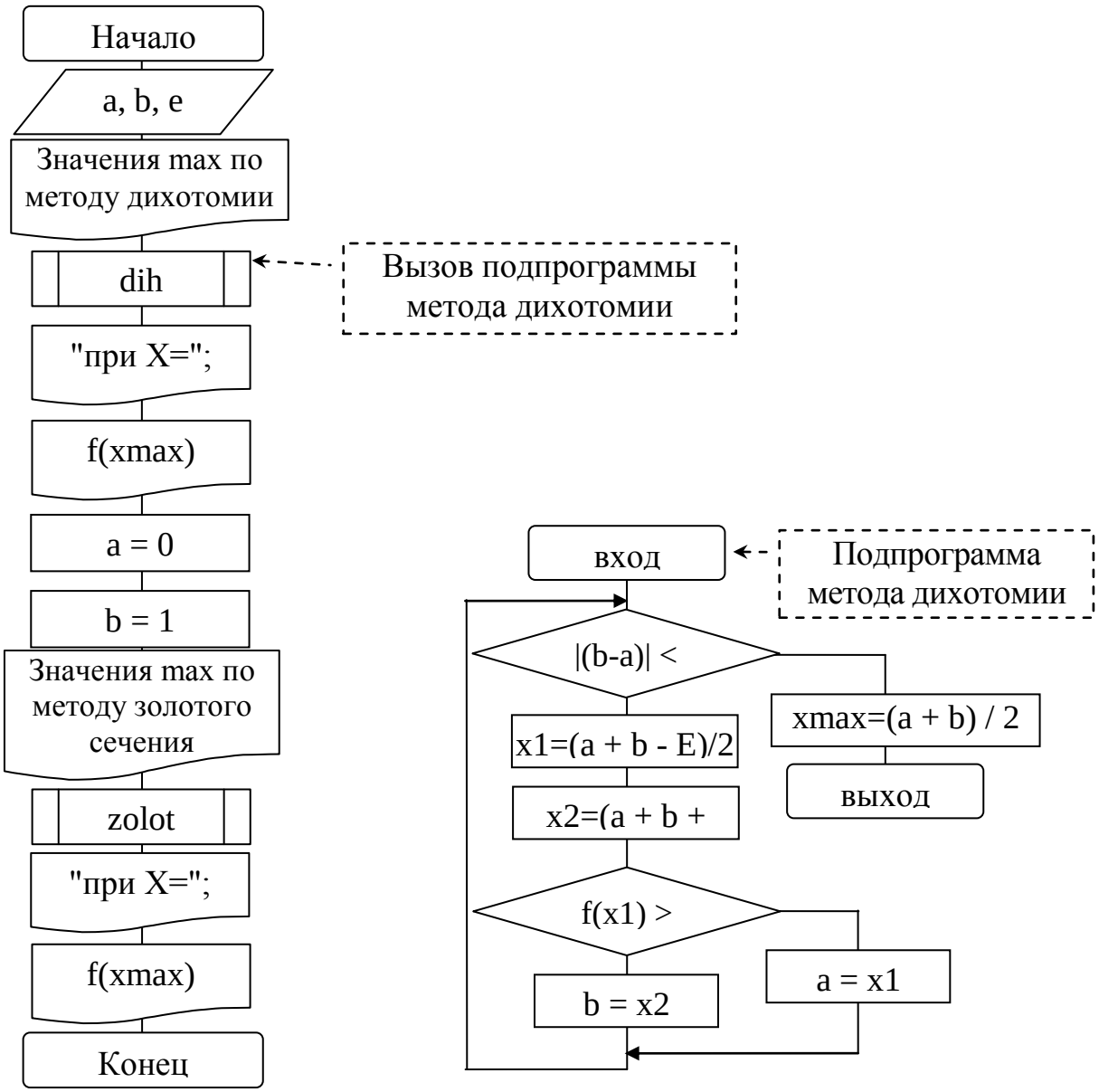
4. Блок схема подпрограмм и блок схема головного (или управляющего) модуля.
5. Листинг подпрограмм и управляющего модуля.
6. Распечатка полученных результатов.
7. Распечатка результатов в Excel и Эврика.

Пример выполнения лабораторной работы

Дана функция $y = -2x^2 + 3x + 50$.

Найти оптимальное значение функции у двумя способами: методом «золотого сечения» и методом «половинного деления». Заданный интервал измерения х (0;1), точность вычисления E = 0.001.

БЛОК-СХЕМА



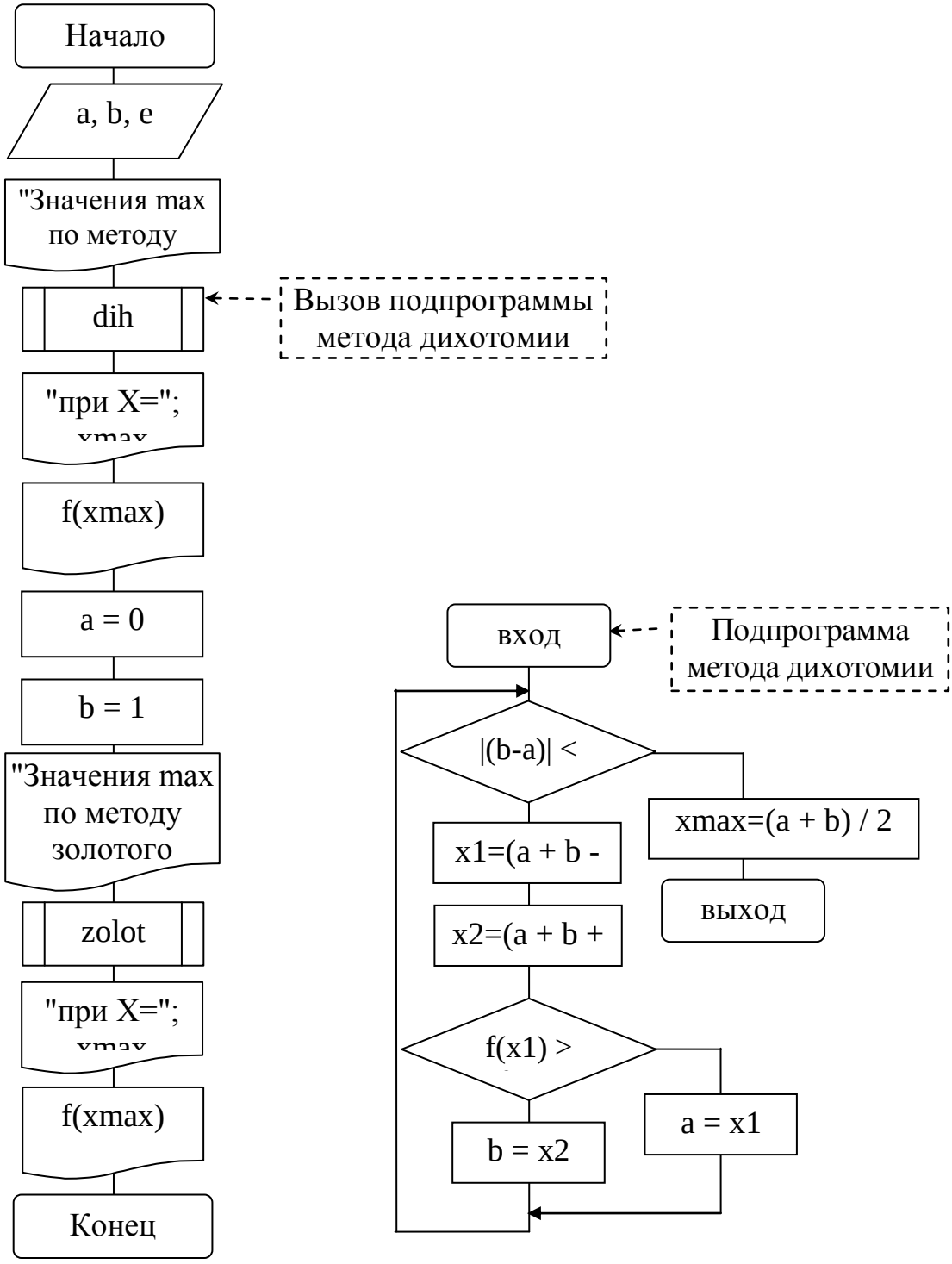


Рис. 8. Блок схема алгоритма (общая и процедура решения по методу половинного деления):

где a, b - нижняя и верхняя границы изменения x;

e - точность вычислений;

dih - процедура вычисления методом половинного деления;

zolot - процедура вычисления методом золотого сечения.

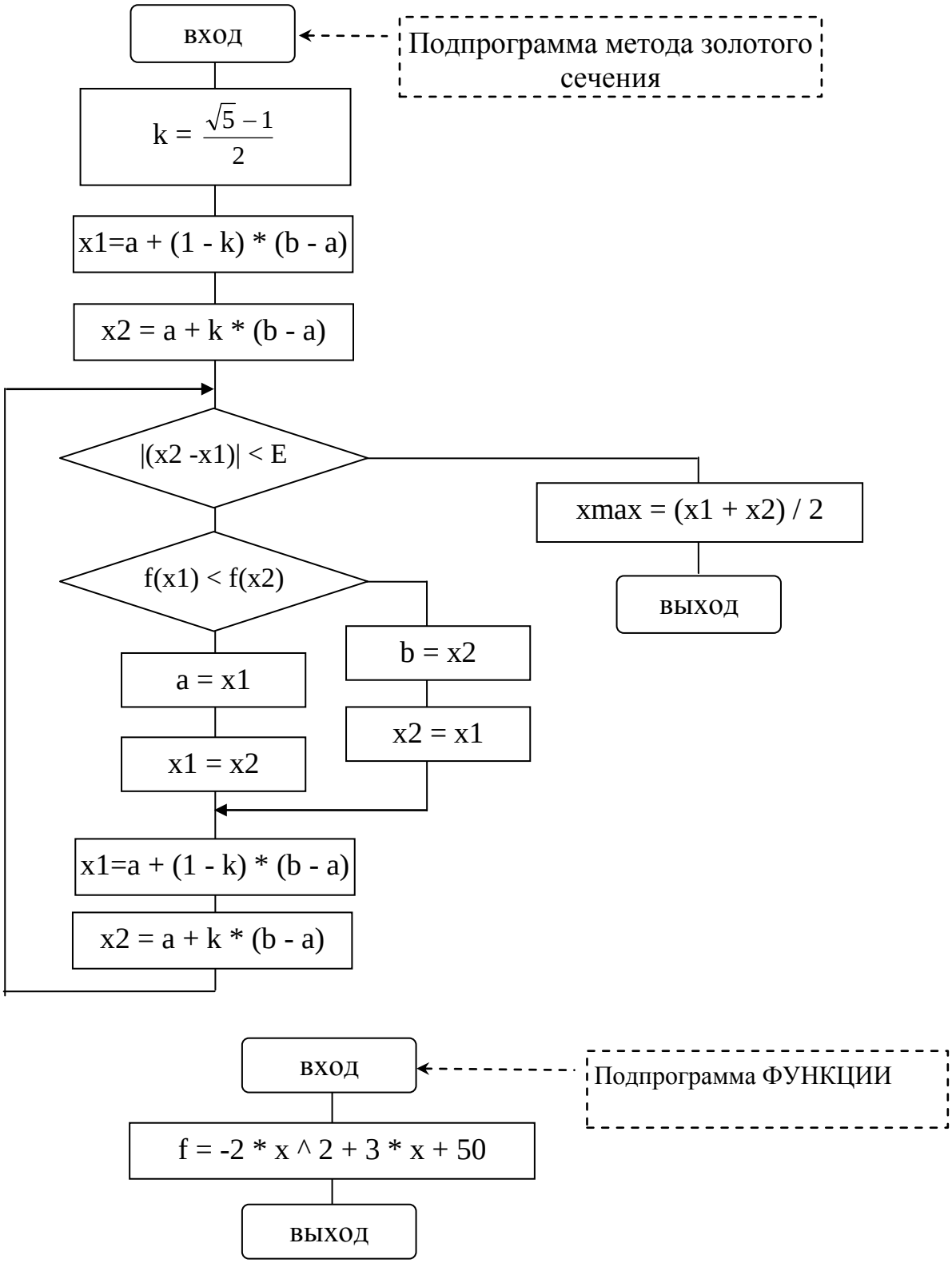


Рис. 9. Блок схема процедуры решения по методу золотого сечения, функция.

ПРОГРАММА НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ QBASIC

```
DECLARE SUB zolot (a!, b!, E!, xmax!)
DECLARE SUB dih (a!, b!, E!, xmax!)
DECLARE FUNCTION f! (x!)
```

```
CLS
INPUT "введите значения отрезка a="; a
INPUT "введите значения отрезка b="; b
INPUT "введите погрешность вычисления Eps="; E
REM метод дихотомии или половинного деления
CALL dih(a, b, E, xmax)
PRINT "Значения max по методу дихотомии"
PRINT "при X="; xmax
PRINT "значение функции Y(xmax)="; f(xmax)
a = 0
b = 1
PRINT "Значение max по методу золотого сечения"
CALL zolot(a, b, E, xmax)
PRINT "при X="; xmax
PRINT "значение функции Y(xmax)="; f(xmax)
END
SUB dih (a, b, E, xmax)
DO UNTIL ABS(b - a) < 2 * E
x1 = (a + b - E) / 2
x2 = (a + b + E) / 2
IF f(x1) > f(x2) THEN
b = x2
ELSE
a = x1
END IF
LOOP
xmax = (a + b) / 2
END SUB
FUNCTION f (x)
f = -2 * x ^ 2 + 3 * x + 50
```

```
END FUNCTION
SUB zolot (a, b, E, xmax)
k = (SQR(5) - 1) / 2
x1 = a + (1 - k) * (b - a)
x2 = a + k * (b - a)
DO UNTIL ABS(x2 - x1) < E
IF f(x1) < f(x2) THEN
a = x1
x1 = x2
ELSE
b = x2
x2 = x1
END IF
x1 = a + (1 - k) * (b - a)
x2 = a + k * (b - a)
LOOP
xmax = (x1 + x2) / 2
END SUB
```

РЕЗУЛЬТАТ в Qbasic

Значение max по методу дихотомии

при X= .7563525

значение функции Y(xmax)= 51.12492

Значение max по методу золотого сечения

при X= .748997

значение функции Y(xmax)= 51.125

Решение задачи с использованием ППП Eureka

```
$ max(F)
y(x)=-2*x^2+3*x+50
F=y(x)
```

Решение:

Переменные Значения

F = 51.125000

x = .75000000

Все ограничения удовлетв. = 98,6%

График функции имеет вид:

Y(x) = -2x^2 + 3*x + 50

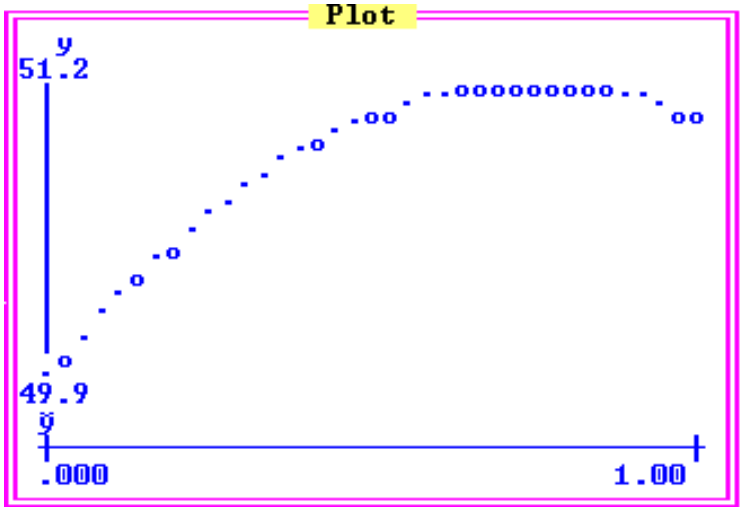


Рисунок 10. График в Эврике.

Задания

для выполнения лабораторной работы

«Оптимизация технологического процесса»

В соответствии с содержательной постановкой задачи (описанной ранее) студенты определяют диапазон нахождения оптимального значения функции $y = ax^2 + bx + c$. Коэффициенты квадратичной модели студенты выбирают самостоятельно.

По указанию преподавателя студенты составляют блок-схемы алгоритмов решения задачи, программы на алгоритмическом языке для решения задачи предлагаемыми методами.

Отладив программу на ЭВМ, получив решение задачи, студенты анализируют полученные решения и сравнивают их с решением, полученным с помощью ППП Eureka и Excel.

Контрольные вопросы

1. Какой экстремум называется глобальным?
2. Что такое унимодальная функция?
3. В чем состоит задача оптимизации?
4. Каким образом можно сузить интервал исследования?
5. перечислите этапы алгоритма решения задачи нахождения максимума функции?
6. Что происходит с уменьшением шага изменения аргумента?
7. Перечислите методы оптимизации функции?
8. На чем основан метод равномерного поиска?
9. Каким алгоритмом реализуется метод дихотомии?
10. Для каких функций пригоден метод половинного деления?
11. Какого основное достоинство метода половинного деления?
12. В чем заключается метод Фибоначчи?
13. На чем основан метод «золотого сечения»?
14. Приведите алгоритм метода «золотого сечения»?
15. Какой из рассмотренных в лабораторной работе методов приводит к более «быстрому» решению?
16. Какие подпрограммы Вы использовали в лабораторной работе?
17. Каким оператором осуществляется вызов процедуры?
18. Где используются формальные и фактические параметры?

ЛАБОРАТОРНАЯ РАБОТА № 7

Работа с файлами последовательного доступа

Цель работы

Ознакомление с основными принципами работы с файлами последовательного доступа.

Работа с файлами

Информация, вводимая с клавиатуры или обрабатываемая с помощью программных средств Бейсика, размещается в оперативной памяти компьютера.

Алгоритм, набранный в Бейсике, может быть сохранен на диске в виде файла.

Файл — это поименованная область на магнитном или лазерном диске. В файлах могут содержаться тексты, графические и видеоизображения, звуки и музыка, таблицы и базы, данные программы, данные для этих программ.

Требования к имени файла

- имя не должно быть больше чем 8 символов;
- имя может состоять из букв латинского алфавита, цифр и символов, например, _, -, (,), $ и некоторых других.
- в имени файла **запрещены** символы <Пробел>, *, точка, запятая, кавычки, двоеточие.

Впрочем, злоупотреблять специальными символами не стоит — букв и цифр вполне хватает.

Расширение файла

Файл имеет расширение.

Расширение имени файла (англ. *filename extension*, часто говорят просто расширение файла или расширение) — последовательность символов, добавляемых к имени файла и предназначенных для идентификации типа (формата) файла. Расширение имеет длину не более трех символов, указывается через точку после имени.

Расширение файла	Описание формата файла
*.aif, *.aifc, *.aiff	Файлы аудиоданных
*.asm	Исходный текст программы на Ассемблере.
*.avi	Основной формат видеоизображений
*.bas	Текст программы на языке алгоритмическом языке Basic и различных его вариантах (GWBasic, TurboBasic, QuickBasic)
*.bmp	Формат графических файлов (растровая графика).
*.com	Исполняемый файл в двоичном коде
*.cpp	Текст программы на языке C++

*.doc	Файл с документами или продукт работы текстового процессора Microsoft Word for Windows
*.dot	Шаблон документа текстового процессора MS Word
*.exe	Это всегда исполняемый бинарный файл
*.gif	(Graphics Interchange Format). Растровый графический формат фирмы CompuServe
*.htm, *.html	Специальный файл текстового типа, написанный на Hyper Text Markup Language
*.mdb	Файл баз данных Microsoft Access
*.mov -	Формат хранения видео и аудио
*.pas -	Текст программы на языке Pascal
*.ppt	Файл с презентацией Microsoft PowerPoint
*.sys -	Системные файлы ядра DOS IO.sys и MSDOS.sys.
*.txt	Текстовой файл, созданный в блокноте
*.xls	Файл работы табличного процессора Microsoft Excel
*.zip -	Файл архива сжатого архиваторами

В файлах вы можете хранить как исходные данные для обработки, так и результаты работы программы.

Для работы в Бейсике необходимы файлы, хранящие однородные по типу или структуре сведения, о каких-либо объектах. Набор данных о каком-либо одном объекте называется **записью.**

Файл может быть пустым, т. е. содержать 0 байт информации, но имя файла и символ конца файла будут присутствовать. (**Байт** - единица измерения количества информации, объема памяти и емкости запоминающего устройства. По умолчанию байт считается равным 8 битам).

Записи могут содержать данные разных типов, но должны быть обязательно одинаковы по структуре, например:

"Запорожец", "4067 ЛДЕ", "1972", "100$"

"ГАЗ-34", "6666 ЛАА", 1989, "3500$"

В соответствии со способом доступа к файлам они делятся на два вида.

1. Файл с последовательным доступом;

2. Файл с прямым доступом.

Файлы последовательного доступа наиболее просты как в организации, так и в работе с ними. Записи обрабатываются последовательно одна за другой.

Информация в таких файлах хранится в виде текста в кодах ASCII. Такие файлы легко просмотреть на экране, используя любой простейший редактор, или в самом Бейсике.

Простота — хорошо, а последовательность в данном случае — плохо. Если информация об интересующем объекте упорядочена в файле по алфавиту, то придется перебирать практически весь файл, чтобы добраться до нужной записи. Отсюда, при большом информационном объеме файла обработка его замедляется.

Файлы прямого доступа хранят информацию в специальном формате, в котором каждая запись занимает строго фиксированную одинаковую с остальными длину. Такие файлы занимают на диске больше места, чем файлы последовательного доступа, но скорость работы с ними значительно выше.

Операции над файлами

Независимо от того, какие действия происходят с информацией, хранящейся в файле, производятся следующие обязательные операции:

1. открытие файла;
2. чтение и запись обрабатываемых данных;
3. закрытие файла.

Открытие файла

Для открытия файла предназначен оператор OPEN, имеющий следующий формат:

OPEN *имя_ файла* FOR *режим* AS # *номер файла*

Режим определяет доступ к данным файла. Возможны следующие режимы:

- **INPUT**. Это режим чтения информации из файла. В случае если указывается несуществующее имя файла, возникнет сообщение об ошибке "Файл не найден".

- **OUTPUT.** Режим записи информации в файл. Обычно при этом создается новый файл. Если же открывается для записи уже существующий файл, то ранее хранимая в нем информация будет утеряна.
- **APPEND.** Режим добавления информации в файл. Новая информация будет размещена в конце файла, за последней записью.

Номер файла предваряется знаком #, после которого следует целое число от 1 до 255.

Запись в файл

Рассмотрим пример записи в файл.

```
OPEN "capitals.dat" FOR OUTPUT AS #1
FOR X=l TO 5
INPUT "ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ"; F$
WRITE #1, F$
NEXT X
CLOSE #1 :END
```

В результате работы программы мы получим (полужирным шрифтом выделены введенные с клавиатуры данные):

ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? **МОСКВА**
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? **САНКТ-ПЕТЕРБУРГ**
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? **ТАЛЛИН**
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? **РИГА**
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? **ВИЛЬНЮС**

При этом на диске в текущем каталоге образуется файл, содержащий пять строковых значений. Имя файла будет capitals.dat. Удобнее всего при работе с файлами сделать текущим каталог, где эти файлы содержатся или будут содержаться, а при обращении к ним указывать только их имена.

В качестве параметра *имя_файла* можно использовать переменную текстового типа. Это позволяет вводить имя файла с клавиатуры и является универсальным способом работы с файлами. На пример:

INPUT «Введите имя файла "; FileName$

OPEN FileName$ FOR OUTPUT AS #1

После ключевого слова AS указывается номер файла. Больше открыть файл с таким номером в данной программе нельзя. Для каждого последующего файла должен быть указан свой собственный неповторимый номер в пределах от 1 до 255. Сколько всего может быть открыто файлов, зависит от файла конфигурации вашего компьютера config.sys, в котором число одновременно открытых файлов определяется командой FILES.

Представленный далее пример демонстрирует программу создания небольшой базы данных автомобилей, предназначенных для продажи. О каждом автомобиле заносится информация о его марке, номере, цвете, годе производства и продажной цене.

```
REM Программа создания файла данных об автомобилях
CLS
INPUT «Введите имя файла"; F$
 Открытие файла
OPEN F$ FOR OUTPUT AS #1
DO
INPUT "Марка автомобиля? (Для окончания работы введите QWE.); M$
IF UCASE$(M$)="QWE" OR OCASE$(M$)="ЙЦУ" THEN 1
INPUT "Номер автомобиля?"; N$
INPUT "Цвет автомобиля?"; C$
INPUT "Год производства автомобиля?"; G$
INPUT "Продажная цена автомобиля?"; S$
WRITE #1, M$, N$, C$, G$, S$
LOOP
' Закрытие файла
1 : CLOSE #1 PRINT "Файл сформирован"
```

Программа действует следующим образом. Запрашивает имя файла, открывает его для записи, запрашивает информацию, записывает ее в файл до введения пользователем сочетания букв "QWE" или "ЙЦУ" (эти символы

расположены на одних клавишах, вследствие чего пользователь может случайно набрать как одну комбинацию, так и другую).

Запись в файл может производиться операторами:

PRINT # *номер_файла,* выражение

ИЛИ

WRITE # *номер_файла, выражение*

Результат работы этих операторов одинаков.

Для закрытия файлов применяется оператор

CLOSE # *номер_файла*

Если номер файла в операторе **CLOSE** указан, то будет закрыт именно этот, вполне определенный файл. Если же номер не указан, будут закрыты все открытые файлы.

Чтение из файла

Чтение из файла производится аналогично записи, но — вместо режима OUTPUT используется режим INPUT. Прочитаем занесенные нами данные из файла avto.dat.

```
REM Программа чтения файла данных об автомобилях
CLS
INPUT " Введите имя файла"; F$
' Открытие файла
OPEN F$ FOR INPUT AS #1
PRINT "База данных автомобилей на 17 декабря 2000 года"
I=1
DO
PRINT "Вывести данные об"; I; "автомобиле?"
INPUT "Для окончания введите QWE, для продолжения -<Enter>"; M$
IF UCASE$(M$)="QWE" OR UCASE$(M$)="ЙЦУ" THEN 1
INPUT #1, M$, N$, C$, G$, S$
PRINT M$, N$, C$, G$, S$
I=I+1
```

```
LOOP UNTIL EOF(1)
' Закрытие файла
1 : CLOSE #1
PRINT "Файл закрыт"
```

Оператор **LOOP UNTIL EOF(1)** . Означает, что считывание ведется до тех пор, пока не будет обнаружен символ конца файла (end of file), а в скобках указан номер открытого файла.

Изменения данных в файле

Для изменения какой-либо записи, удаления старых или добавления новых данных в последовательном файле необходимо открыть два файла: подлежащий изменению и новый, в котором создается обновленная версия исходного файла. Старый файл в дальнейшем можно удалить.

Приведенная ниже программа в файле avto.dat изменяет "МОСКВИЧ" на "МЕРСЕДЕС". В первых строках открываются исходный файл avto.dat и новый файл avto2.dat, сначала пустой. Очередная запись считывается из файла avto.dat и, при условии, что это не "МОСКВИЧ", переписывается без изменения в новый файл. Если же встречается значение "МОСКВИЧ", то оно заменяется на "МЕРСЕДЕС" путем присваивания нового значения переменной м$. В следующей. Строке данное значение попадает в выходной файл. После того как весь входной файл просмотрен, оба файла закрываются.

```
OPEN "avto.dat" FOR INPUT AS #1
OPEN "avto2.dat" FOR OUTPUT AS #2
FOR i=1 TO 5
INPUT #1, M$, N$, C$, G$, S$
IF UCASE$(M$)-"МОСКВИЧ" THEN M$-"МЕРСЕДЕС"
PRINT 12, M$, N$, C$, G$, S$
NEXT i
CLOSE 1, 2
KILL "avto.dat"
NAME "avto2.dat" AS "avto.dat"
```

```
END
```

Заключительный этап — удаление исходного и переименование нового файла, которому придается прежнее имя, что обеспечивает и в дальнейшем наличие на дискете файла avto.dat.

Добавление данных в файл

Указание FOR APPEND в операторе OPEN подготавливает файл для вывода данных и смещает указатель на конец файла. Последующие операторы приписывают новую информацию к уже имеющейся. В предложенной далее программе в файл данных об автомобилях добавляются сведения о двух новых поступлениях.

```
REM Программа создания файла данных об автомобилях
CLS
INPUT " Введите имя файла"; F$
REM Открытие файла
OPEN F$ FOR APPEND AS 1
DO
INPUT "Марка автомобиля? (Для окончания работы введите QWE.); M$
IF UCASE$(M$)="QWE" OR UCASE$(M$)="ЙЦУ" THEN 1
INPUT "Номер автомобиля?"; N$
INPUT "Цвет автомобиля?"; C$
INPUT "Год производства автомобиля?"; G$
INPUT "Продажная цена автомобиля?"; S$
WRITE #1, M$, N$, C$, G$, S$
LOOP
REM Закрытие файла 1
CLOSE #1
PRINT "Файл дополнен'"
```

Порядок выполнения работы

1. Получить у преподавателя вариант задания.
2. Написать программу на Qbasic.

3. Отладить программу.
4. Пполучить результат.
5. Проанализировать полученный результат.

Содержание отчета

1. Содержательная постановка задачи.
2. Исходные данные.
3. Краткие теоретические данные.
4. Блок схема программы.
5. Листинг программы.
6. Распечатка полученных результатов.

Задание

1. Составить программу создания файла данных **МАГАЗИН** (не менее 10 записей), каждая запись которого содержит следующие поля:

- Название товара
- Страна изготовитель
- Дата изготовления
- Срок годности (истек или нет, yes/no)
- Стоимость товара

2. Составить программу вывода на экран в табличной форме всех записей файла, и записей наименований товаров, срок годности которых не истек, найти самый дешевый товар, вывести его название и стоимость.

Пример решения задачи

- Зададим имя файла данных – magasin.txt
- Опишем переменные:

Название товара – name$

Страна изготовитель – ctrana$

Дата изготовления – den

Срок годности (истек или нет, yes/no) – god$

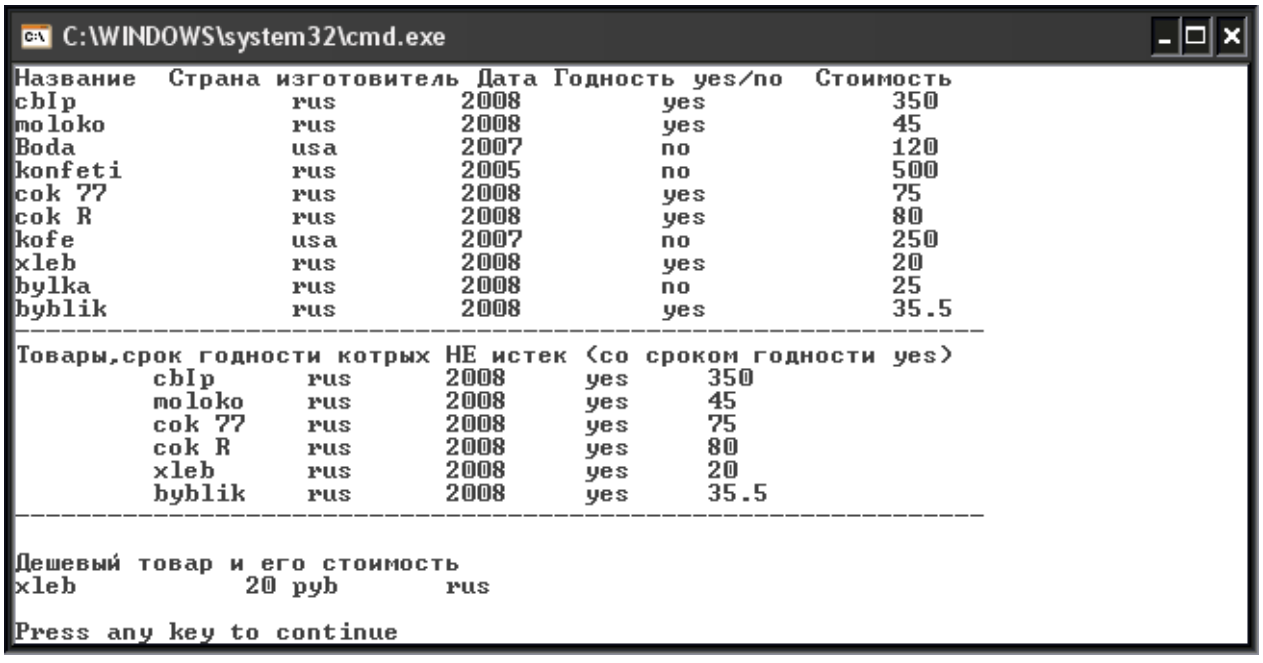
Стоимость товара – manu

ПРОГРАММА НА ЯЗЫКЕ QBasic

```
CLS
OPEN "magasin.txt" FOR OUTPUT AS #1
CLS
FOR i = 1 TO 10
INPUT " Название "; name$
INPUT " Страна изготовитель "; ctrana$
INPUT " год изготовления "; den
INPUT " годность yes/no: "; god$
INPUT " стоимость "; manu
WRITE #1, name$, ctrana$, den, god$, manu
NEXT i
CLOSE #1
PRINT "МАГАЗИН"
PRINT "--------------------------------------------------------------------------------"
PRINT "Название Страна изготовитель Дата Годность yes/no Стоимость"
OPEN "magasin.txt" FOR INPUT AS #1
DO WHILE NOT EOF(1)
INPUT #1, name$, ctrana$, den, god$, manu
PRINT name$, " "; ctrana$, den; " ", god$; " ", manu
LOOP
 CLOSE #1
PRINT "--------------------------------------------------------------------------------"
min = 10 ^ 10
OPEN "magasin.txt" FOR INPUT AS #1
PRINT "Товары со сроком годности yes"
DO WHILE NOT EOF(1)
INPUT #1, name$, ctrana$, den, god$, manu
 IF manu <= min THEN
```

```
 nas1$ = name$: ctrana1$ = ctrana$: god1$ = god$: min = manu
 END IF
 IF god$ = "yes" THEN
 PRINT TAB(10); name$;
 PRINT TAB(20); ctrana$; " "; den; " "; god$; " "; manu
 END IF
LOOP
 CLOSE #1
 PRINT
PRINT "-------------------------------------------------------------------------------"
PRINT "Дешевый товар и его стоимость "
 PRINT nas1$, min; "руб", ctrana1$
END
```

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ



Созданный файл данных magasin.txt из 10 записей рисунок 11.

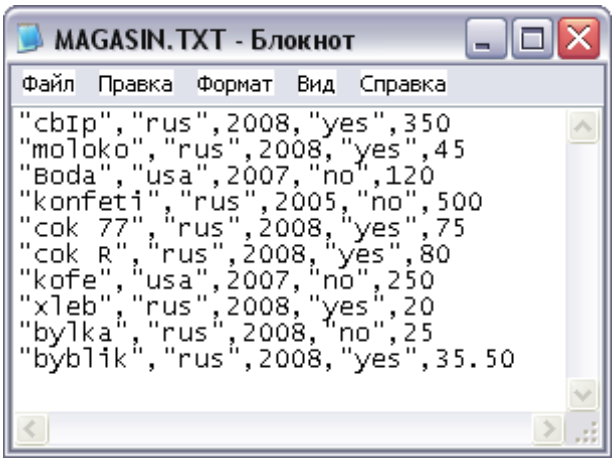


Рисунок 11.

Контрольные вопросы

1. Что такое файл?
2. Какие требования к имени файла?
3. Что такое расширение файла?
4. Что можно хранить в файлах?
5. Какие файлы в соответствии со способом доступа Вы знаете?
6. В чем отличие файла прямого доступа от последовательного?
7. Перечислите операции с файлами?
8. Как осуществляется открытие файла?
9. Перечислите режимы работы с файлами?
10. Как осуществляется запись в файл?
11. Как осуществляется чтение из файла?
12. Что используется в качестве параметра «имя файла»?
13. Какой оператор используется для закрытия файла?
14. Какими операторами производятся запись в файл?
15. Каком образом можно изменить данные в файле?
16. Как осуществляется добавление данных в файл?
17. В каком случае используется функция EOF(1)?

Варианты заданий к лабораторной работе

Вариант № 1.

1. Составить программу создания файла данных **СКЛАД** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *наименование товара;*

• *количество;*

• *стоимость за единицу;*

• *наличие на складе (да /нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла базы, и записей наименований товаров, количество которых превосходит 1000 и стоимость за единицы меньше 1000 руб.

Вариант № 2.

1. Составить программу создания файла данных **ЗООПАРК** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *вид животного;*

• *дата рождения;*

• *вес;*

• *является ли хищником (да / нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла базы данных, и записей видов животных, являющихся хищниками, вес которых превышает 150кг.

Вариант № 3.

1. Составить программу создания файла данных **АВТОПАРК** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *марка автомобиля;*

• *мощность двигателя;*

• *пробег;*

• *был ли в кап. ремонте (да / нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных, и записей марок автомобилей, которые не на ремонте и пробег которых более 100000 км.

Вариант № 4.

1. Составить программу создания файла данных **БИБЛИОТЕКА** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *автор;*

• *название;*

• *издательство;*

• *количество изданий;*

• *пользуется ли спросом (да / нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных, и записей, в которых содержится информация об авторах, количество изданий которых превышает 10000 и вышедших в издательстве Вариус.

Вариант № 5.

1. Составить программу создания файла данных **КОМПЬЮТЕРЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *тип процессора;*

• *объем RAM;*

• *объем HDD;*

• *цветной ли монитор (да / нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных, компьютеров имеющих цветной монитор и объем HDD превышающий 50 Gb.

Вариант № 6.

1. Составить программу создания файла данных **ПРИНТЕРЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *название фирмы;*

• *марка;*

• *характеристика (матричный, струйный, лазерный...);*

• *цветной (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных, и записей содержащих название фирм, выпускающих лазерные, цветные принтеры.

Вариант № 7.

1. Составить программу создания файла данных **СУПЕРМАРКЕТ** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *наименование товара;*
- *цена;*
- *количество на складе;*
- *отечественного производства (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных, отдельно записей, содержащий наименование товаров отечественного производства, не превышающих 200руб.

Вариант № 8.

1. Составить программу создания файла данных **АВТОЗАПЧАСТИ** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *наименование запасной части;*
- *цена;*
- *марка автомобиля;*
- *отечественного производства (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных, а так же записей, содержащих наименование запасных частей отечественного производства, ценно которых не превышает 500руб.

В следующих вариантах выполнить задание вывода на экран всех записей, а так же составить запрос на вывод (и вывести на экран) по двум условиям (см. вариант 1-8).

Вариант № 9.

1. Составить программу создания файла данных **ГРУППА** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *фамилия и инициалы;*

• *год рождения;*

• *телефон;*

• *изучал ли ранее английский язык (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 10.

1. Составить программу создания файла данных **МЕБЕЛЬ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *наименование;*

• *размеры;*

• *цвет;*

• *цена.*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 11.

1. Составить программу создания файла данных **КОСМЕТИКА** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *название фирмы или торговая марка;*

• *предназначение;*

• *цена;*

• *импортный товар (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант №12.

1. Составить программу создания файла данных **ДЕНДРАРИЙ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *вид растения;*

• *возраст;*

• *страна произрастания;*

• *холодостойкость (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант №13.

1. Составить программу создания файла данных **АТТРАКЦИОНЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название аттракциона;*
- *цена билета;*
- *количество мест;*
- *допуск детей до 16 лет (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 14.

1. Составить программу создания файла данных **ТЕАТРЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название;*
- *количество мест;*
- *средняя цена билета;*
- *есть ли утренние спектакли (да/нет).*

1. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 15.

1. Составить программу создания файла данных **ОДЕЖДА** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *наименование;*
- *размер;*
- *цена;*
- *импортного производства (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант №16.

1. Составить программу создания файла данных **КОМПЬЮТЕРНЫЕ ИГРЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название;*
- *фирма-создатель;*
- *год выпуска;*
- *требуется ли RAM свыше 1 Mb (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 17.

1. Составить программу создания файла данных **ХИТ-ПАРАД** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название группы;*
- *солист;*
- *месяц и год выпуска последнего альбома;*
- *имеется ли в свободной продаже последний альбом (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 18.

1. Составить программу создания файла данных **ВИДЕОТЕКА** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *название фильма;*

• *год выпуска;*

• *краткая характеристика;*

• *допускается ли просмотр детьми до 16 лет (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 19.

1. Составить программу создания файла данных **ГАЗЕТЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *название;*

• *тираж;*

• *цена подписки;*

• *выходит ежедневно (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 20.

1. Составить программу создания файла данных **ЖУРНАЛЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *название;*

• *цена подписки на год;*

• *количество листов;*

• *ежемесячный (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант №21.

1. Составить программу создания файла данных **АПТЕКА** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *наименование лекарства;*

• *цена;*

• *срок годности;*

• *импортное (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 22.

1. Составить программу создания файла данных **БИРЖА ТРУДА** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *профессия;*

• *стаж работы;*

• *ограничение по возрасту (не старше ...);*

• *необходимое владение компьютером (да/нет)*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 23.

1. Составить программу создания файла данных **РЕКИ МИРА** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название;*
- *континент;*
- *протяженность;*
- *впадает ли в океан (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 24.

1. Составить программу создания файла данных **САД И ОГОРОД** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название растения;*
- *сорт;*
- *возраст;*
- *многолетнее (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 25.

1. Составить программу создания файла данных **МЕНЮ** (не менее 10 записей), каждая запись которого содержит следующие поля:

• *название блюда:*

• *вес одной порции;*

• *цена;*

• *дежурное (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 26.

1. Составить программу создания файла данных **ПОЛИКЛИНИКА** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *фамилия и инициалы врача;*
- *специальность;*
- *дни приема;*
- *нужна ли предварительная запись (да/нет).*

1. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 27.

1. Составить программу создания файла данных **МУЛЬТФИЛЬМЫ** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название;*
- *год выпуска;*
- *продолжительность;*
- *рисованный или кукольный (да/нет).*

1. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 28.

1. Составить программу создания файла данных **УЧЕБНЫЙ ПЛАН** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *дисциплина;*
- *преподаватель;*
- *количество часов в неделю;*
- *есть ли лабораторные работы (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 29.

1. Составить программу создания файла данных **ГОРОДА** (не менее 10 записей), каждая запись которого содержит следующие поля:

- *название;*
- *страна;*
- *количество жителей;*
- *является ли столицей (да/нет).*

2. Составить программу вывода на экран в табличной форме всех записей файла данных.

Вариант № 30.

1. Составить программу создания файла данных **ТУРБЮРО** (не менее 10записей), каждая запись которого содержит следующие поля:

- *название курорта;*
- *количество дней;*
- *цена путёвки;*
- *входит ли питание в цену (да/нет).*

2**.** Составить программу вывода на экран в табличной форме всех записей файла данных.

Список литературы

1 Банди Б. Методы оптимизации. Вводный курс. – М.: Радио и связь, 1988.

2 Бахвалов Н.С. Численные методы : Численные методы/ Н.С.Бахвалов, — М.: Наука, 1973. — 630с.

3 Василькова Ю.В., Васильков Н.Н. Компьютерные технологии вычислений. – М.: Финансы и статистика, 2002 – 256 с.

4 Воробьёв Г.Н., Данилова А.Н. Практикум по численным методам: практикум по численным методам/ Воробьёв Г.Н., Данилова А.Н., — М.: Высшая школа, 1979. — 184с.

5 Гусева А.И. Учимся информатики: задачи и методы решения. – М.: Диалог-МИФИ, 1998 – 320 с.

6 Дьяконов В.П. Справочник по применению системы Eureka. – М.: Наука, 1993.

7 Конопленко Е.И Учебное пособие по курсу «Информатика»: учебное пособие по курсу «Информатика» — М.: Издательский комплекс МГУПП, 2005 — 102 с.

8 Сафонов И. Бейсик в задачах и примерах: бейсик в задачах и примерах/ Сафонов И., — СПБ. БХВ, 2001. — 224с.

9 Уайлд Д.Дж. Методы поиска экстремума. – М.: Наука, 1967.