МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»

ОСНОВЫ КОМПЬЮТЕРИЗАЦИИ ТЕХНОЛОГИЙ В СИСТЕМАХ АВТОМАТИКИ

Методические указания по выполнению расчетно-графических работ
для студентов специальности 1-53 01 01-05 «Автоматизация технологических процессов и производств (легкая промышленность)»

Витебск
2023

УДК 681.5:004.9

Составители:

А. С. Соколова, В. Е. Казаков, В. Н. Шут

Рекомендовано к изданию редакционно-издательским
советом УО «ВГТУ», протокол № 7 от 04.04.2023.

Основы компьютеризации технологий в системах автоматики : методические указания по выполнению расчетно-графических работ / УО «ВГТУ» ; сост. : А. С. Соколова, В. Е. Казаков, В. Н. Шут. – Витебск, 2023. – 58 с.

В методических указания изложены теоретические сведения и индивидуальные задания для выполнения двух расчетно-графических работ по дисциплине «Основы компьютеризации технологий в системах автоматики». Издание предназначено для студентов специальности 1-53 01 01-05 «Автоматизация технологических процессов и производств (легкая промышленность)».

Издание в электронном виде расположено в репозитории библиотеки УО «ВГТУ».

УДК 681.5:004.9

СОДЕРЖАНИЕ

ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ РАСЧЕТНО-ГРАФИЧЕСКИХ РАБОТ И ИХ ЗАЩИТЕ

Разработанный студентом программный продукт должен решать поставленную задачу и обладать всеми свойствами алгоритма (конечность, результативность и т. д.), не приводить к возникновению ошибок во время исполнения при вводе различных наборов входных данных.

Блок-схема разработанного алгоритма должна соответствовать исходному тесту программы.

Отчёт о выполнении каждой задачи должен содержать:

1. Полный текст задания.
2. Блок-схему алгоритма (если она указана в задании), выполненную в соответствии с требованиями ГОСТом 19.701-90.
3. Исходный текст программы.
4. Скриншоты консолей с результатами тестовых запусков.

На защите студент должен по заданию преподавателя:

- описать назначение каждой из переменных;
- описать назначение каждой части алгоритма;
- объяснить, как будет изменяться работа программы при вводе различных данных;
- обосновать выбор метода решения задачи.

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА 1. РЕШЕНИЕ ЗАДАЧ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ

Основные теоретические сведения

1.1 Идентификаторы

Идентификатор – это имя программного объекта (переменной, функции, класса или другого объекта). При выборе идентификатора следует иметь в виду следующее:

1. Идентификатор может состоять только из букв латинского алфавита, цифр или символов подчёркивания.
2. Идентификатор должен начинаться с буквы (нижнего или верхнего регистра). Он не может начинаться с цифры.
3. Не рекомендуется начинать идентификаторы с символа подчеркивания, т. к. в этом случае они могут совпасть с именами системных функций или переменных.
4. C++ различает нижний и верхних регистры. **`Min`**, **`MIN`**, **`min`** – три

разных идентификатора.

5. Идентификатор не может быть ключевым словом.

6. Длина идентификатора по стандарту не ограничена, но некоторые компиляторы и компоновщики налагают на нее ограничения.

1.2 Типы данных

К *основным* типам данных языка C++ относят:

– **char** – символьный;
– **int** – целый;
– **float** – с плавающей точкой;
– **double** – двойной точности;
– **bool** – логический;
void – пустой.

Типы данных, созданные на базе стандартных типов с использованием спецификаторов, называют *составными*. В C++ определены четыре спецификатора типов данных:

– **short** – короткий;
– **long** – длинный;
– **signed** – знаковый;
– **unsigned** – беззнаковый.

1.3 Переменные

Переменная – это поименованный участок памяти, в котором хранится значение определенного типа. Переменная имеет тип, имя и значение.

Перед использованием любую переменную надо определить:

```
тип список_переменных;
```

Например:

```
int a;
float g, u, m2;
```

Можно сразу при определении переменной дать ей некоторое начальное значение. Данный прием называется инициализацией. Например:

```
int age = 28;
```

1.4 Константы

Константы – это именованные ячейки памяти, значения которых фиксируются на начальном этапе выполнения программы и затем в процессе

выполнения программы не могут быть изменены.

В С ++ константы определяются следующим образом:

```
const тип имя = значение;
```

Например:

```
const double g = 9.81;
```

1.5 Операции, выражения, операторы

Оператор – законченное предложение на языке С++. Он указывает компьютеру выполнить некоторые действия. Оператор всегда завершается «;».

Выражение – конструкция, определяющая состав данных, операции и порядок выполнения операций над данными. Выражение состоит из операндов, знаков операций и круглых скобок.

Рисунок 1.1 – Выражение

Операнды – данные, над которыми выполняются действия.

Операции выполняют определенные действия над операндами.

Приоритет основных операций:

1. Инкремент, декремент.
2. Унарные плюс и минус, логическое и поразрядное НЕ, приведение к типу, взятие адреса.
3. Умножение, деление, остаток от деления.
4. Сложение, вычитание.
5. Больше, меньше, больше или равно, меньше или равно.
6. Равно, не равно.
7. Логическое И.
8. Логическое ИЛИ.
9. Операции присваивания.

Для изменения порядка вычисления используются круглые скобки.

Рассмотрим наиболее часто используемые операции.

Операции присваивания

Обычная операция присваивания имеет вид:

```
идентификатор = значение;
```

В C++ существует возможность множественного присваивания, то есть присваивания нескольким переменным одного и того же значения:

```
идентификатор1 = идентификатор2 = ... идентификаторN =
значение;
```

Так же в C++ имеются составные операции присваивания:

```
идентификатор операция= значение;
```

Их эквивалентом является следующая запись:

```
идентификатор = идентификатор операция значение;
```

операция – одна из операций **+**, **–**, *****, **/**, **%**, **&**, **|**, **^**, **<<**, **>>**.

Например, выражение `a += 2` эквивалентно выражению `a = a + 2`.

Арифметические операции

Операции **+**, **–**, *****, **/** относят к арифметическим операциям. Их назначение понятно и не требует дополнительных пояснений. Однако стоит отметить, если операция деления применяется к целочисленным операндам, то результатом является целая часть частного (дробная часть отбрасывается). Например:

```
11 / 4; //В результате будет 2
11 / 4.0; //В результате будет 2.75
```

Остаток от деления **%** применяется только к целочисленным аргументам.

Например:

```
11 % 4; //В результате будет 3
```

Операции инкремента **++** и декремента **--** так же причисляют к арифметическим, так как они выполняют увеличение и уменьшение на единицу значения переменной. Эти операции имеют две формы записи префиксную (операция записывается перед операндом) и постфиксную (операция записывается после операнда). Эти формы отличаются при использовании их в выражении. Если знак инкремента (декремента) предшествует операнду, то сначала выполняется увеличение (уменьшение) значения операнда, а затем операнд участвует в выражении.

Например:

```
int x = 12;
int y = ++x; //В результате y = 13
```

Если знак инкремента (декремента) следует после операнда, то сначала операнд участвует в выражении, а затем выполняется увеличение (уменьшение) значения операнда:

```
int x = 12;
int y = x++; //В результате y = 12
```

Логические операции

К логическим операциям относятся **&&**, **||**, **!**. Данные операции выполняются над логическими значениями. В таблице 1.1 приведены результаты логических операций.

Таблица 1.1 – Логические операции

A	B	!A	A&&B	A||B
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

Операции сравнения

Операции сравнения возвращают в качестве результат логическое значение `true` или `false`. К данной группе операций относят `>`, `>=`, `<`, `<=`, `==`, `!=`.

1.6 Структура программы

Программа на языке С++ состоит из директив препроцессора, указаний компилятору, объявлений глобальных переменных и/или констант, объявлений и определений функций.

Общая структура программы:

```
директивы препроцессора
описание типов пользователя;
прототипы функций;
описание глобальных переменных;
тип_результа main(параметры)
{
    операторы;
}
тип_результа имя1(параметры)
{
    операторы1;
}
тип_результа имя2(параметры)
{
    операторы2;
}
...
тип_результа имяN(параметры)
{
    операторыN;
}
```

Препроцессор – это программа, которая обрабатывает текст программы до компилятора.

Работа препроцессора управляется директивами.

Директива **#include** включает содержимое файла, путь к которому задан, в компилируемый файл вместо строки с директивой:

#include <путь> либо **#include "путь"**

Если путь заключен в угловые скобки, то поиск файла осуществляется в стандартных директориях. Если путь заключен в кавычки и задан полностью, то поиск файла осуществляется в заданной директории, а если путь полностью не задан – в текущей директории. С помощью этой директивы можно включать в текст программы как стандартные, так и свои файлы.

1.7 Стандартная математическая библиотека

В стандартную математическую библиотеку языка C++ входит множество специальных математических функций. Для того чтобы использовать эти функции, необходимо подключить заголовочный файл, содержащий описания этих функций командой:

#include <math.h> либо **#include <cmath>**

Таблица 1.2 – Основные математические функции

Функция	Описание
1	2

abs(x)	Модуль целого числа. Например:
	`abs(-3); // 3`
	`abs(-3.9); // 3`

acos(x)	$\arccos(x)$. Возвращается результат из диапазона -π… π
asin(x)	$\arcsin(x)$. Возвращается результат из диапазона -π… π
atan(x)	$arctg(x)$. Возвращается результат из диапазона -π/2… π/2
cbrt(x)	$\sqrt[3]{x}$

ceil(x)	Округление до большего целого. Например:
	`ceil(2.4); // 3`
	`ceil(-1.5); // -1`

cos(x)	$\cos(x)$
cosh(x)	$ch(x)$
exp(x)	e^x

fabs(x)	Модуль числа. Например:
	`fabs(-3); // 3`
	`fabs(-3.9); // 3.9`

floor(x)	Округление до меньшего целого. Например:
	`floor(2.4); // 2`
	`floor(-1.5); // -2`

Окончание таблицы 1.2

1	2

`fmod(x, y)`	Вычисление остатка от деления **x** на **y**. Например:
	`fmod(3,4); // 3`
	`fmod(6.4,3.1); // 0.2`

`log(x)`	$\ln(x)$
`log2(x)`	$\log_2(x)$
`log10(x)`	$\lg(x)$
`pow(x, y)`	x^y

`round(x)`	Округляет число по правилам арифметики. Например:
	`round(2.4); // 2`
	`round(-1.5); // -2`

`sqrt(x)`	$\sqrt{x}$
`sin(x)`	$\sin(x)$
`sinh(x)`	$sh(x)$
`tan(x)`	$tg(x)$
`tanh(x)`	$th(x)$

`trunc(x)`	Отбрасывание дробной части. Например:
	`trunc(2.4); // 2`
	`trunc(-1.5); // -1`

Стандартная математическая библиотека содержит также определения некоторых математических констант.

Таблица 1.3 – Некоторые математические константы

Символ	Описание	Значение
`M_E`	e	`2.71828182845904523536`
`M_PI`	π	`3.14159265358979323846`

1.8 Комментарии

Комментарий – это строка (или несколько строк) текста, которая служит для описания и документирования исходного кода. В C++ используются два типа комментариев: однострочные и многострочные.

Однострочные комментарии – это комментарии, которые пишутся после символов `//`. Они размещаются в отдельных строках и всё, что находится после этих символов до конца строки, игнорируется компилятором.

Многострочные комментарии – это комментарии, которые пишутся между символами `/*` и `*/`. Всё, что находится между звёздочками, игнорируется компилятором.

1.9 Ввод-вывод данных на консоль

Ввод-вывод данных в языке C++ осуществляется либо с помощью *функций ввода-вывода* в стиле C, либо с использованием *библиотеки классов* C++. Преимущество объектов C++ в том, что они легче в использовании, особенно если ввод-вывод достаточно простой. Функции ввода-вывода, унаследованные от C более громоздкие, но подходят для задач с форматированным выводом данных.

Форматированный ввод-вывод

Функции форматированного ввода и вывода данных расположены в библиотеке **stdio.h**.

Форматированный вывод переменных, указанных в списке, в соответствии со строкой форматирования выполняет функция:

```
printf(строка_форматирования, список_выводимых_переменных)
```

Ввод переменных, адреса которых указаны в списке, в соответствии со строкой форматирования выполняет функция:

```
scanf(строка_форматирования, список_адресов_вводимых_переменных)
```

Строка форматирования содержит символы, которые будут выводиться на экран или запрашиваться с клавиатуры и так называемые спецификации. *Спецификации* – это строки, которые начинаются символом **%** и выполняют управление форматированием:

```
%«флаг»«ширина».«точность»«модификатор»«тип»
```

Параметры флаг, ширина, точность и модификатор в спецификациях могут отсутствовать.

Таблица 1.4 – Символы управления вводом-выводом

Символ	Назначение
1	2
Флаг	
+	Перед числом выводится знак «**+**» или «**–**»
пробел	Перед положительным числом выводится пробел, перед отрицательным – минус
#	Выводится код системы счисления: **0** – перед восьмеричным числом, **0x** (**0X**) – перед шестнадцатеричным
Ширина	
n	Минимальное число выводимых символов. Незаполненные позиции заполняются пробелам
0n	Минимальное число выводимых символов. Незаполненные позиции заполняются нулями
Точность	
n	Для вещественных типов выводить n знаков после точки

Окончание таблицы 1.4

1	2
Модификатор	
`h`	используется в качестве префикса с целыми типами для определения, что аргумент является `short int`
`l`	используется в качестве префикса с целыми типами для обозначения, что аргумент является `long int` или `long long int`. Символ l используется также как префикс с вещественными типами для определения, что аргумент является скорее `double`, чем `float`
Тип	
`c`	Вывод одного символа
`s`	Вывод строки символов
`d`	Вывод целого десятичного числа
`i`	Вывод целого десятичного, восьмеричного или шестнадцатеричного числа
`E, e`	Вывод вещественного числа с плавающей точкой
`f`	Вывод вещественного числа с фиксированной точкой
`G, g`	Вывод более краткое из `e` и `f`
`o`	Вывод целого беззнакового восьмеричного числа
`X, x`	Вывод целого беззнакового шестнадцатеричного числа
`u`	Вывод целого беззнакового десятичного числа
`p`	Вывод значения указателя
`n`	Вывод числа прочитанных символов

Кроме того, строка форматирования может содержать Escape-последовательности.

Таблица 1.5 – Escape-последовательности

`\a`	Звуковой сигнал	`\\`	Обратная косая черта
`\b`	Возврат на шаг	`\'`	Одиночная кавычка
`\f`	Переход на новую страницу	`\"`	Двойная кавычка
`\n`	Переход на новую строку	`\?`	Вопросительный знак
`\r`	Возврат в начало строки	`\000`	Символ, заданный восьмеричным кодом ASCII 000
`\t`	Горизонтальная табуляция	`\xdd`	Символ, заданный шестнадцатеричным кодом ASCII dd
`\v`	Вертикальная табуляция	`\udddd`	Символ, заданный шестнадцатеричным кодом Unicode dddd

Потоковый ввод-вывод

Средства потокового ввода и вывода данных расположены в библиотеке **iostream**. Библиотека iostream определяет три стандартных потока:

- **cin** – стандартный входной поток;
- **cout** – стандартный выходной поток;
- **cerr** – стандартный поток вывода сообщений об ошибках.

Для их использования обязательно необходимо прописать строку:

```
using namespace std;
```

Для выполнения операций ввода-вывода переопределены две операции:

>> – получить из входного потока;

<< – поместить в выходной поток.

Вывод информации осуществляется командой:

```
cout << значение;
```

Здесь значение преобразуется в последовательность символов и выводится в выходной поток.

Возможно многократное назначение потоков:

```
cout << значение1 << значение2 << ... << значениеN;
```

Ввод информации осуществляется командой:

```
cin >> идентификатор;
```

При этом из входного потока читается последовательность символов до пробела, затем эта последовательность преобразуется к типу идентификатора, и получаемое значение помещается в идентификатор.

Возможно многократное назначение потоков:

```
cin >> идентификатор1 >> идентификатор2 >> ... >> идентификаторN;
```

Кириллица на консоли

В командной строке Windows кодировка символов соответствует стандарту cp866. Причём поменять кодировку в командной строке Windows нельзя. Во всех русскоязычных Windows кодировка cp1251 является стандартной. И при создании проекта этот стандарт кодирования символов наследуется проектом, то есть программой. В итоге, получается, что программа передаёт коды символов сообщения стандарта cp1251. Командная строка принимает эти коды и переводит их в символы, но уже по стандарту cp866, так как другого стандарта не знает. В итоге сообщение передано в консоль, но символы интерпретированы неправильно. Так появляются «иероглифы».

Существует несколько способов решения данной проблемы.

Самый простой – настройка локали.

Локаль – это набор параметров: набор символов, язык пользователя, страна, часовой пояс и др. В библиотеке **clocale** есть функция **setlocale()**, которая выполняет настройку локали.

Функция **setlocale()** имеет два параметра, первый параметр – тип категории локали (в данном случае **LC_TYPE** – набор символов), второй параметр – значение локали (в данном случае **"rus"**, **"Russian"** или пустые

двойные кавычки).

Функция **setlocale()** работает только для вывода данных. Если же осуществлять ввод, то будут все те же иероглифы.

В библиотеке **windows.h** содержатся функции, позволяющие решить и эту проблему. Функция **SetConsoleCP(1251)** устанавливает нужную кодовую таблицу для ввода, тогда как функция **SetConsoleOutputCP(1251)** устанавливает нужную кодовую таблицу для вывода.

1.10 Составной оператор

Составной оператор – это группа операторов, отделённых друг от друга точкой с запятой, начинающихся с открывающей фигурной скобки **{** и заканчивающихся закрывающейся фигурной скобкой **}**:

```
{
    оператор1;
    ...
    операторN;
}
```

Транслятор воспринимает составной оператор как одно целое.

1.11 Условный оператор

Для организации разветвляющихся алгоритмов в C++ предусмотрен условный оператор **if**, который в общем виде записывается следующим образом:

```
if (условие)
    оператор1;
else
    оператор2;
```

Работает условный оператор следующим образом. Сначала вычисляется значение **условие**. Если оно не равно нулю, т. е. имеет значение истина (**true**), выполняется **оператор1**. В противном случае, когда выражение равно нулю, т. е. имеет значение ложь (**false**), выполняется **оператор2**.

Оператор1 и **оператор2** могут быть составными.

Альтернативная ветвь **else** в условном операторе может отсутствовать, если в ней нет необходимости.

Условные операторы могут быть вложены друг в друга.

ПРИМЕР. Вычислить значение функции:

$$f(x)=\begin{cases}0, x<0\\ x^2, 0\le x\le 1\\ x, x>1\end{cases}$$

```
#include <iostream>
#include <math.h>
#include <windows.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    double x, f;
    cout << "Введите x: ";
    cin >> x;
    if (x < 0)
        f = 0;
    else if (x <= 1)
        f = pow(x, 2);
    else
        f = x;
    cout << "f(x)=" << f << endl;
    return 0;
}
```

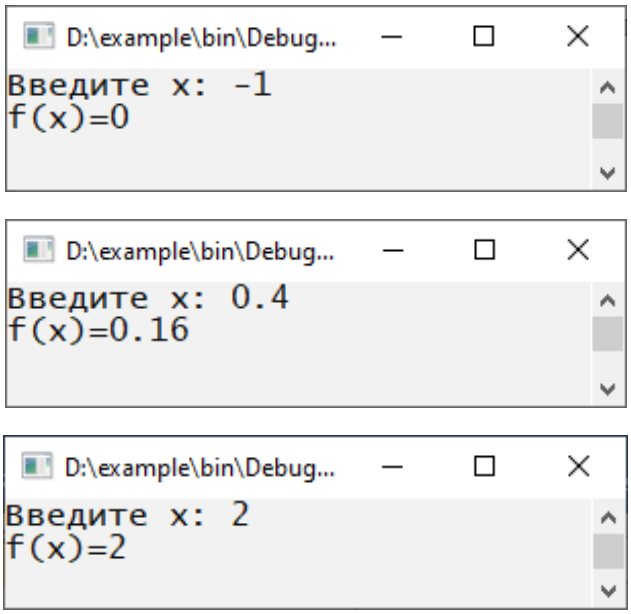

Рисунок 1.2 – Результат выполнения программы

1.12 Оператор выбора

Оператор варианта **switch** необходим в тех случаях, когда в зависимости от значений какой-либо переменной надо выполнить те или иные операторы:

```
switch (выражение)
{
    case значение1:
        операторы1;
        break;
    case значение2:
        операторы2;
        break;
    ...
    case значениеN:
        операторыN;
        break;
    default:
        операторы;
}
```

Оператор работает следующим образом. Вычисляется значение **выражения**. Если оно принимает **значение1**, то выполняются **операторы1**. Если выражение принимает **значение2**, то выполняется **операторы2** и так далее. Если **выражение** не принимает ни одно из значений, то выполняются операторы, расположенные после ключевого слова **default**.

Альтернативная ветвь **default** может отсутствовать.

Оператор **break** необходим для того, чтобы осуществить выход из оператора **switch**. Если оператор **break** не указан, то будут выполняться следующие операторы из списка, несмотря на то, что значение, которым они помечены, не совпадает со значением выражения.

Отсутствие оператора **break** используется, как правило, для того, чтобы выполнить ту или иную ветвь в зависимости не от одного, а от нескольких значений. Выглядеть такая ветвь будет следующим образом:

```
    ...
    case значение1:
    case значение2:
    case значение3:
        операторы;
        break;
    ...
```

ПРИМЕР. По заданному номеру месяца ***n*** вывести его название.

```
#include <iostream>
#include <windows.h>
```

```
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int n;
    cout << "Введите номер месяца: " << endl;
    cin >> n;
    switch (n)
    {
        case 12:
        case 1:
        case 2:
            cout << "Зима" << endl;
            break;
        case 3:
        case 4:
        case 5:
            cout << "Весна" << endl;
            break;
        case 6:
        case 7:
        case 8:
            cout << "Лето" << endl;
            break;
        case 9:
        case 10:
        case 11:
            cout << "Осень" << endl;
            break;
        default:
            cout << "В году только 12 месяцев!" << endl;
    }
    return 0;
}
```

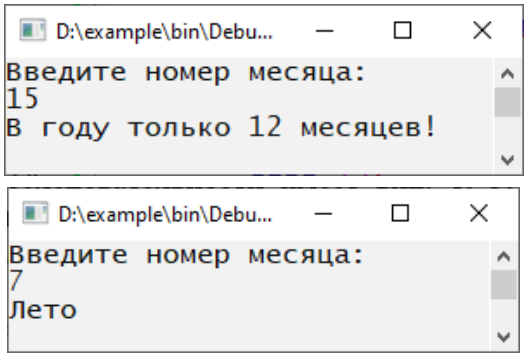

Рисунок 1.3 – Результат выполнения программы

1.13 Операторы цикла

Цикл – повторение одних и тех же действий. Последовательность действий, которые повторяются в цикле, называют *телом цикла*. Один проход цикла называют *шагом* или *итерацией*. Переменные, которые изменяются внутри цикла и влияют на его окончание, называются *параметрами* цикла.

В C++ для удобства пользователя предусмотрены три оператора, реализующих циклический процесс: **`while`**, **`do...while`** и **`for`**.

Оператор цикла с предусловием

Оператор цикла с предусловием имеет вид:

```
while (условие)
    оператор;
```

Работает цикл с предусловием следующим образом. Проверяется **условие**. Если оно истинно (не равно нулю), то выполняется **оператор**, и **условие** проверяется вновь. В противном случае цикл заканчивается, и управление передаётся оператору, следующему за телом цикла.

Оператор может быть составным.

ПРИМЕР. В последовательности чисел первое число – 1, а каждое следующее больше предыдущего на свой порядковый номер. То есть, начало последовательности имеет вид: 1, 3, 6, 10, 15, 21, … Написать программу, выписывающую все элементы последовательности, меньшие 50.

```
#include <iostream>
using namespace std;
int main()
{
    int number = 1, i = 1;
    while (number < 50)
    {
        cout << number << '\t';
        i++;
        number += i;
    }
    return 0;
}
```

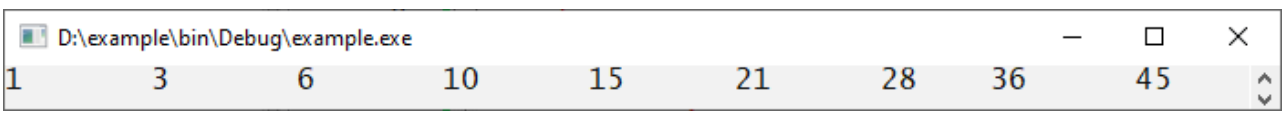

Рисунок 1.4 – Результат выполнения программы

Оператор цикла с постусловием

Оператор цикла с постусловием имеет вид:

```
do
    оператор;
while (условие);
```

Работает цикл следующим образом. В начале выполняется **оператор**. Затем проверяется **условие**. Если оно истинно (не равно нулю), **оператор** выполняется снова. В противном случае цикл завершается, и управление передаётся оператору, следующему за циклом.

Оператор может быть составным.

Тело цикла с постусловием будет выполнен хотя бы один раз, в отличие от цикла с предусловием, который может не выполниться ни разу.

ПРИМЕР. Написать программу, реализующую игру «Угадай число». Загадывать число будет компьютер, используя генератор случайных чисел в диапазоне от 1 до 10.

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int unknownNumber, enterNumber;
    srand(time(NULL));
    unknownNumber = 1 + rand() % 10;
    do
    {
        cout << "Введите число: " << endl;
        cin >> enterNumber;
    }
    while (enterNumber != unknownNumber);
    cout << "Победа!!!" << endl;
    return 0;
}
```

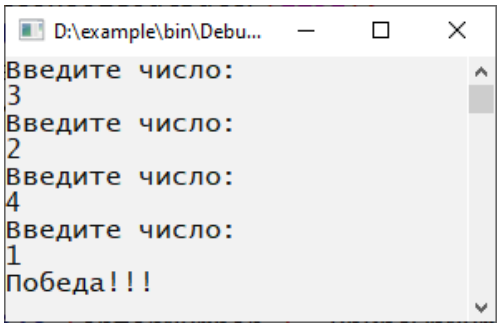

Рисунок 1.5 – Результат выполнения программы

Оператор цикла со счетчиком

Оператор цикла со счетчиком имеет вид:

```
for (начальные_присваивания; условие; последействие)
    оператор;
```

Здесь **начальные_присваивания** – оператор или группа операторов, разделённых запятой, которые применяются для присвоения начальных значений величинам, используемым в цикле, в том числе параметру цикла, и выполняются один раз в начале цикла. **Условие** – целое или логическое выражение, которое определяет условие выполнения тела цикла. Если **условие** истинно (не равно нулю), то тело цикла выполняется. **Последействие** – оператор или группа операторов, разделённых запятой, которые выполняются после каждой итерации и служат для изменения параметра цикла. **Оператор** – тело цикла. **Оператор** может быть составным.

Последействие или **оператор** должны влиять на **условие**, иначе цикл никогда не закончится. **Начальные_присваивания**, **условие** или **последействие** в записи оператора **for** могут отсутствовать, но при этом точки с запятой должны оставаться на своих местах.

ПРИМЕР. Написать программу, выводящую на экран таблицу значений функции $y = x^2$ на отрезке [0; 5] с шагом 0,5.

```
#include <iostream>
using namespace std;
int main()
{
    double x, y;
    cout << "x\ty" << endl;
    for (x = 0; x <= 5; x += 0.5)
    {
        y = x * x;
        cout << x << "\t" << y << endl;
    }
    return 0;
}
```

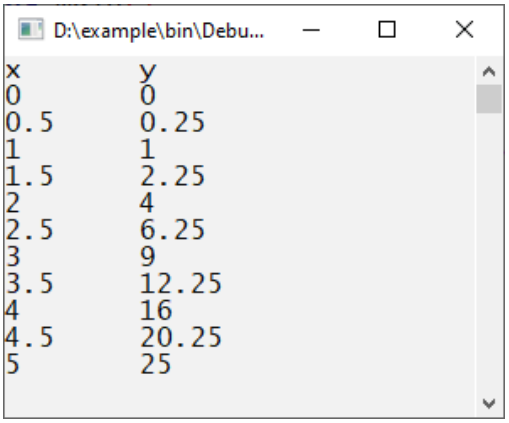

Рисунок 1.6 – Результат выполнения программы

1.14 Операторы передачи управления

Оператор **goto** передаёт управление оператору с меткой:

```
goto метка;
...
метка: оператор;
```

Оператор **break** осуществляет немедленный выход из циклов, а также из оператора выбора. Управление передаётся оператору, находящемуся непосредственно за циклом или оператором выбора.

Оператор **continue** начинает новую итерацию цикла, даже если предыдущая не была завершена.

Оператор **return** выражение завершает выполнение функции и передаёт управление в точку её вызова.

1.15 Рекуррентные вычисления сумм и произведений

Общая формулировка задач рекуррентного вычисления суммы или произведения элементов выглядит следующим образом:

$$S = \sum_{i=1}^{n} A_i \text{ или } P = \prod_{i=1}^{n} A_i ,$$

где $A_i = F\left(A_{i-1}\right)$. То есть последующее значение элемента A вычисляется на основе предыдущего его значения.

Для вычисления суммы используется прием накопления суммы, состоящий в том, что в цикле каждый раз к промежуточной сумме прибавляется очередное слагаемое. Выполнив описанные вычисления n раз, получают в переменной S требуемое значение суммы. При этом перед циклом, в котором вычисляется сумма, переменной S необходимо присвоить начальное значение, равное 0.

Произведение может быть найдено с использованием аналогичного приема накопления произведения, состоящего в том, что в цикле каждый раз произведение предшествующих множителей P умножается на очередной множитель. Начальное значение переменной P перед циклом, в котором оно вычисляется, должно быть задано равным 1.

В случае бесконечной суммы вычисление продолжается до тех пор, пока разница между предыдущим и последующим значением суммы не станет меньше, чем заданное пользователем значение точности ε:

$$\left|S_i - S_{i-1}\right| < \varepsilon \text{ или } \left|A_i\right| < \varepsilon .$$

ПРИМЕР. Написать программу вычисления суммы:

$$S=\sum_{i=2}^{n}(-1)^{i+2}\frac{(x+1)^{i+2}}{(2i-1)!}\sin\left(\frac{\pi}{i}\right).$$

В выражении для вычисления слагаемых выделим следующие части:

$$a=(-1)^{i+2}\ ;\ b=(x+1)^{i+2};\ c=(2i-1)!.$$

Таблица 1.6 – Нахождение первых значений a, b и c

i	a	b	c
2	1	$(x+1)^4$	$3!=1\cdot2\cdot3$
3	-1	$(x+1)^5$	$5!=1\cdot2\cdot3\cdot4\cdot5$
4	1	$(x+1)^6$	$7!=1\cdot2\cdot3\cdot4\cdot5\cdot6\cdot7$

Можно заметить, что текущие значения a, b и c связаны с предыдущими следующим образом:

$$a_i=a_{i-1}\cdot(-1);\ b_i=b_{i-1}\cdot(x+1);\ c_i=c_{i-1}\cdot(2i-1)\cdot(2i-2).$$

```
#include <iostream>
#include <windows.h>
#include <math.h>
using namespace std;
int main()
{
    int i, n;
    double a, b, c, x, s;
    SetConsoleOutputCP(1251);
    cout << "Введите x, n: ";
    cin >> x >> n;
    a = 1;
    b = pow(x + 1, 4);
    c = 2 * 3;
    s = a * b / c * sin(M_PI / 2);
    for(i = 3; i <= n; i++)
    {
        a = -a;
        b *= x + 1;
        c *= (2 * i - 1) * (2 * i - 2);
```

```
        s += a * b / c * sin(M_PI / i);
    }
    cout << "s=" << s << endl;
    return 0;
}
```

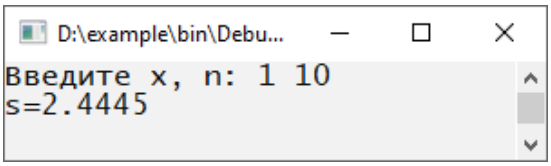

Рисунок 1.7 – Результат выполнения программы

1.16 Массивы

Массив – это структурированный тип данных, состоящий из фиксированного числа элементов одного типа, объединенных единым именем.

Одномерный массив описывается следующим образом:

```
тип имя[n];
```

n – количество элементов в массиве, которое может быть задано числом либо массивом.

Например:

```
int A[6];
const int n = 15;
double B[n];
```

Доступ к каждому элементу массива осуществляется с помощью *индекса* – порядкового номера элемента. Для обращения к элементу массива указывают его имя, а затем в квадратных скобках индекс.

Индексация элементов в массивах начинается с 0!

Можно инициализировать массив при объявлении. Например:

```
int a[6] = {1, 4, -8, 3, 0, 6};
```

или

```
int a[] = {1, 4, -8, 3, 0, 6};
```

Если при инициализации указать меньше значений, чем размер массива, последние элементы заполнятся нулями.

ПРИМЕР. Задан массив из 10 случайных чисел в диапазоне от 0 до 20. Написать программу поиска индекса и значения максимального элемента.

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
```

```
int main()
{
    SetConsoleOutputCP(1251);
    int i, a[10], i_max = 0;
    srand(time(NULL));
    for (i = 0; i < 10; i++)
        a[i] = rand() % 21;
    cout << "Массив A {";
    for (i = 0; i < 10; i++)
        cout << a[i] << ", ";
    cout << "\b\b}" << endl;
    for (i = 1; i < 10; i++)
        if (a[i] > a[i_max])
            i_max = i;
    cout << "Максимальный элемент a[" << i_max << "]=" <<
a[i_max] << endl;
    return 0;
}
```

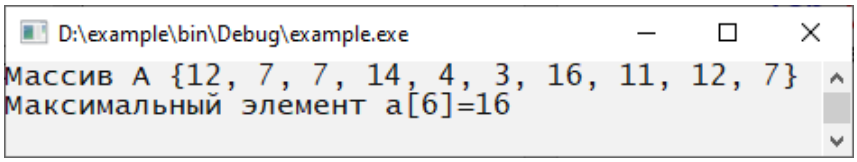

Рисунок 1.8 – Результат выполнения программы

Двумерный массив описывается следующим образом:

```
тип имя[m][n];
```

m – количество строк в массиве;

n – количество столбцов в массиве.

Например: `int a[2][3];`

Пример инициализации двумерного массива:

```
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

ПРИМЕР. Написать программу вычисления в двумерном массиве размерности 3 х 4 суммы элементов, кратных 5.

```
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int i, j, a[3][4] , s = 0;
```

```
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
        {
            cout << "a[" << i << "][" << j << "]=";
            cin >> a[i][j];
        }
    cout << "Матрица A:" << endl;
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 4; j++)
            cout << a[i][j] << "\t";
        cout << endl;
    }
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            if (a[i][j] % 5 == 0)
                s += a[i][j];
    cout << "Сумма элементов кратных 5 равна " << s << endl;
     return 0;
}
```

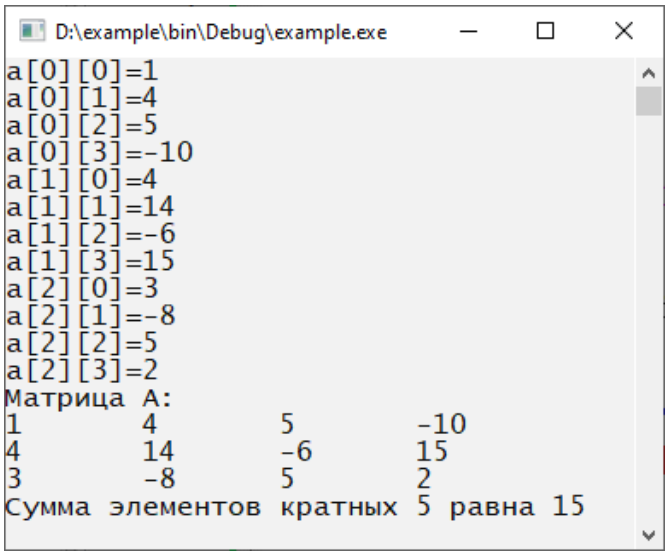

Рисунок 1.9 – Результат выполнения программы

1.17 Динамические массивы

Для создания динамического массива необходимо:

- описать указатель (`тип *имя_указателя;`);
- определить размер массива;

• выделить участок памяти для хранения массива и присвоить указателю адрес этого участка памяти.

Наиболее предпочтительным подходом к динамическому распределению памяти является использование операций языка C++ **new** и **delete**.

Операция **new** выделяет память из области свободной памяти, а операция **delete** высвобождает выделенную память.

Минимальный набор действий, необходимых для динамического размещения одномерного массива действительных чисел размером n:

```
int n;
double *a;
     . . .
a = new double[n]; // Захват памяти для n элементов
     . . .
delete []a;   // Освобождение памяти
```

ПРИМЕР. Написать программу сортировки одномерного динамического массива методом пузырька.

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int i, j, n, a_min, a_max, temp;
    cout << "Введите размер массива:" << endl;
    cin >> n;
    cout << "Введите диапазон значений:" << endl;
    cout << "a_min = ";
    cin >> a_min;
    cout << "a_max = ";
    cin >> a_max;
    int *a;
    a = new int[n];
    srand(time(NULL));
    for (i = 0; i < 10; i++)
        a[i] = a_min + rand() % (a_max - a_min + 1);
    cout << "Исходный массив: {";
    for (i = 0; i < 10; i++)
        cout << a[i] << ", ";
    cout << "\b\b}" << endl;
```

```
    for (i = 1; i < 10; i++)
        for (j = 0; j < 10 - i; j++)
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
    cout << "Результат сортировки: {";
    for(i = 0; i < 10; i++)
        cout << a[i] << ", ";
    cout << "\b\b}" << endl;
    delete []a;
    return 0;
}
```

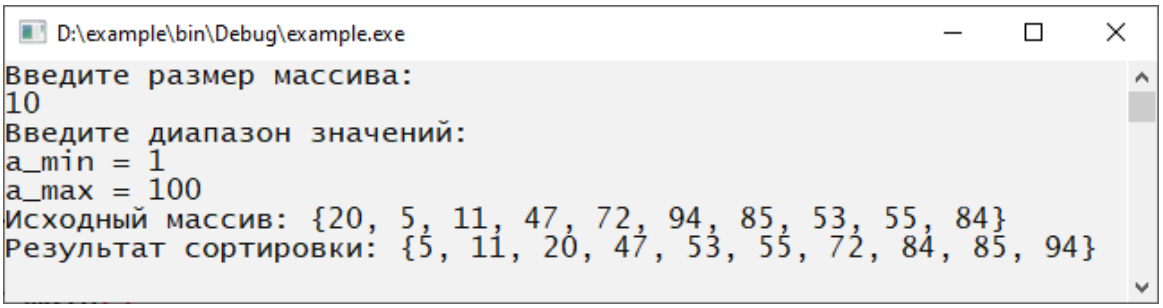

Рисунок 1.10 – Результат выполнения программы

Минимальный набор действий, необходимых для динамического размещения двухмерного массива действительных чисел размером $m \times n$:

```
int i, n, m;  // m, n – размеры массива
double **a;
     . . .
a = new double *[m];   // Захват памяти под указатели
for (i = 0; i < m; i++)
     a[i] = new double[n]; // и под элементы
     . . .
for (i = 0; i < m; i++)
    delete []a[i]; // Освобождение памяти
delete []a;
```

ПРИМЕР. Задана целочисленная матрица A размером nxn. Упорядочить строки матрицы по неубыванию их максимальных элементов.

```
#include <iostream>
```

```
#include <windows.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int i, j, k, n, t, **A, *B;
    cout << "Введите n: ";
    cin >> n;
    A = new int *[n];
    for (i = 0; i < n; i++)
        A[i] = new int[n];
    srand(time(NULL));
    cout << "Матрица A:" << endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            A[i][j] = rand() % 21;
            cout << A[i][j] << "\t";
        }
        cout << endl;
    }
    B = new int[n];
    for (i = 0; i < n; i++)
    {
        B[i] = A[i][0];
        for (j = 1; j < n; j++)
            if (A[i][j] > B[i])
                B[i] = A[i][j];
    }
    for (i = 0; i < n - 1; i++)
        for (j = i + 1; j < n; j++)
            if (B[i] > B[j])
            {
                t = B[i];
                B[i] = B[j];
                B[j] = t;
                for (k = 0; k < n; k++)
                {
                    t = A[i][k];
                    A[i][k] = A[j][k];
```

```
                    A[j][k] = t;
                }
            }
    cout << "Упорядоченная матрица A:" << endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            cout << A[i][j] << "\t";
        cout << endl;
    }
    for (i = 0; i < n; i++)
        delete []A[i];
    delete []A;
    delete []B;
    return 0;
}
```

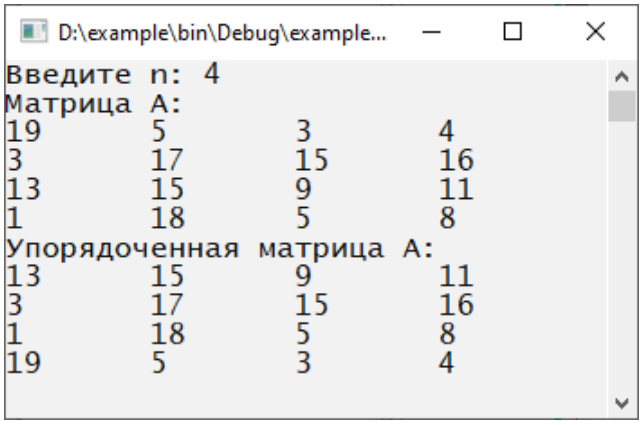

Рисунок 1.11 – Результат выполнения программы

1.18 Функции

Подпрограмма – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке C++ подпрограммы реализованы в виде функций.

Как известно, программа на C++ состоит из одной или нескольких функций. Функция должна быть описана перед своим использованием.

Описание функции состоит из заголовка и тела функции.

```
тип имя_функции([список_параметров]) // заголовок функции
{
    операторы; // тело функции
}
```

Тип функции показывает, какое значение возвращает функция: целое, вещественное, строковое и так далее.

список_параметров содержит перечень типов и имен величин, передаваемых в функцию и разделенных запятыми. Если функция не имеет параметров, то скобки все равно необходимы, хотя в них ничего не указывается.

В случае, если вызываемые функции определены после функции **main** *или даже в другом файле*, необходимо до первого вызова функции (перед функцией **main**) сообщить программе тип возвращаемого функцией значения, а также количество и типы ее параметров. Подобное сообщение называется *прототипом функции*.

Прототип может полностью совпадать с заголовком функции, хотя при объявлении функции компилятору необходимо знать только имя функции, количество аргументов, их типы и тип возвращаемого функцией значения. Поэтому имена параметров необязательно указывать в прототипах функций.

Если возврата значения функцией не требуется, то тип функции – **void**.

Возврат значений функцией выполняется с помощью оператора **return**:

```
return значение;
```

Если тип функции – **void**, то оператор возврата просто – **return;** или его можно даже опустить в теле функции.

Return вызывает немедленный выход из функции и возврат в вызвавшую ее подпрограмму.

Функция возвращает единственное скалярное значение. Это может быть число, адрес, структура.

Для вызова функции в программе пишется имя функции, за которым в скобках следует список аргументов, которые являются переменными и константами, определенными в программе. При этом происходит выполнение тела функции, в котором параметры заменяются значениями аргументов. Тип аргумента должен соответствовать типу соответствующего параметра или допускать приведение к типу параметра.

ПРИМЕР. Дано целое число. Проверить, есть ли среди цифр трехзначного числа повторяющиеся. Привести блок-схему алгоритма реализации функции проверки числа.

Сначала составим блок-схему алгоритма функции (рис.1.12).

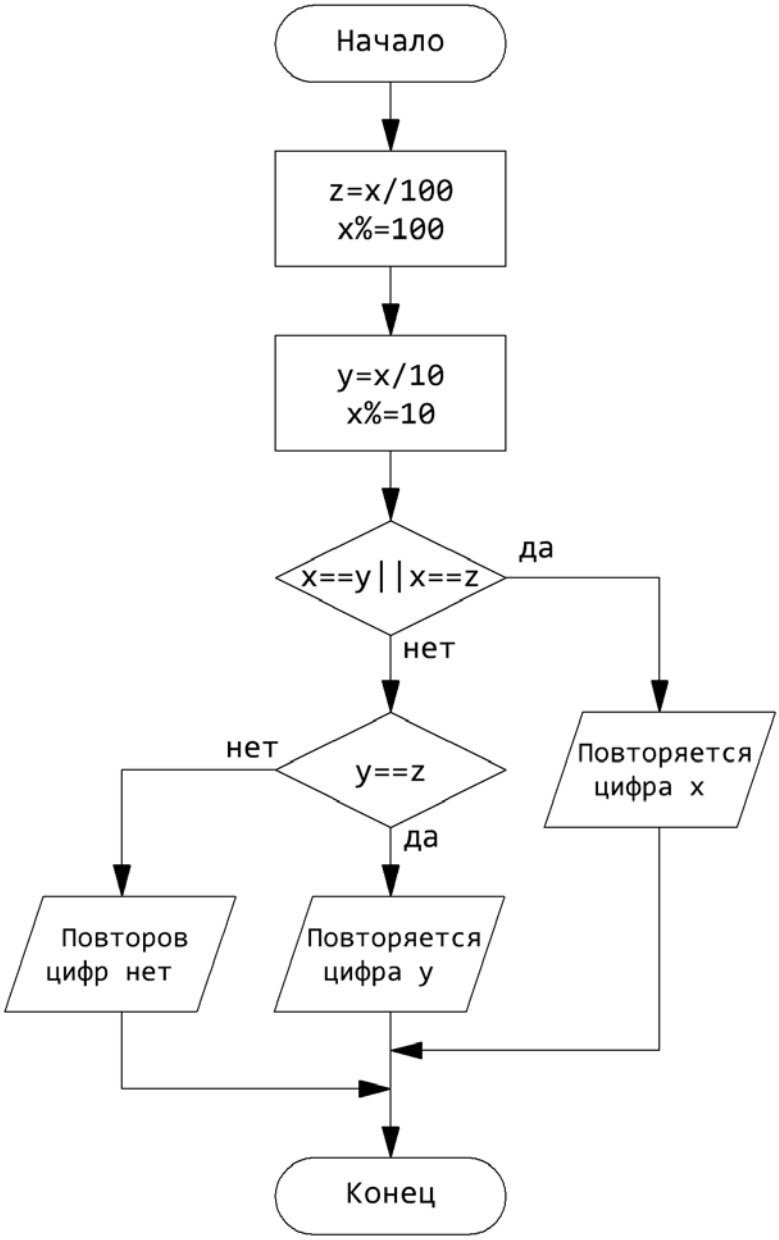

Рисунок 1.12 – Блок-схема алгоритма функции

Теперь приведем реализацию функции и основную программу:

```
#include <iostream>
#include <windows.h>
using namespace std;
void checkNumber(int x)
{
    int y, z;
    z = x / 100;
    x %= 100;
    y = x / 10;
    x %= 10;
```

```
    if ((x == y) || (x == z))
        cout << "Повторяется цифра " << x << endl;
    else if (y == z)
        cout << "Повторяется цифра " << y << endl;
    else
        cout << "Повторов цифр нет" << endl;
}
int main()
{
    SetConsoleOutputCP(1251);
    int n;
    cout << ">";
    cin >> n;
    while (n)
    {
        if ((n > 99) && (n < 1000))
            checkNumber(n);
        else
            cout << "Число не трехзначное!" << endl;
        cout << ">";
        cin >> n;
    }
    return 0;
}
```

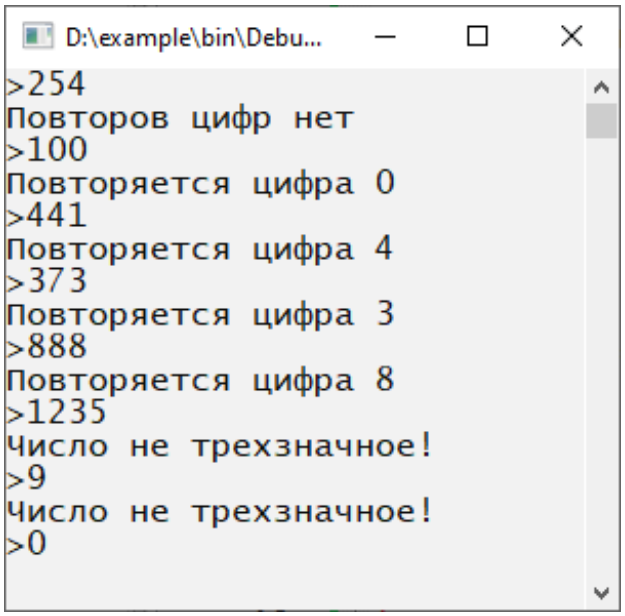

Рисунок 1.13 – Результат запуска программы

Индивидуальные задания

Задание 1. «Операторы языка С++».

Общие требования:

1. Для каждого x, изменяющегося от a до b с шагом h, найти значения функции Y(x), суммы S(x) и |Y(x) – S(x)| и вывести в виде таблицы.

2. Значения a, b, h и n вводятся с консоли.

3. Не использовать функции.

4. Работу программы проверить для a = 0,1; b = 1,0; h = 0,1; n = 7.

Примечание. Так как значение S(x) является рядом разложения функции Y(x), при правильном решении значения S и Y для заданного аргумента x (для тестовых значений исходных данных) должны совпадать в целой части и в первых двух-четырех позициях после десятичной точки.

Таблица 1.7 – Варианты задания

№ варианта	Y(x)	S(x)
1	2	3
1.	e^{-x}	$\sum_{k=0}^{n}(-1)^k \frac{x^k}{k!}$
2.	$\sin(x)$	$\sum_{k=0}^{n}(-1)^k \frac{x^{2k+1}}{(2k+1)!}$
3.	$\frac{e^x - e^{-x}}{2}$	$\sum_{k=1}^{n}\frac{x^{2k-1}}{(2k-1)!}$
4.	e^{x^3+2}	$\sum_{k=0}^{n}\frac{e^2 x^{3k}}{k!}$
5.	$\cos(x)$	$\sum_{k=0}^{n}(-1)^k \frac{x^{2k}}{(2k)!}$
6.	$\frac{1-e^{-x}}{x}$	$\sum_{k=0}^{n}(-1)^k \frac{x^k}{(k+1)!}$
7.	$\cos(3x^2)$	$\sum_{k=0}^{n}(-1)^n \frac{3^{2k} x^{4k}}{(2k)!}$,
8.	e^{-x^2}	$\sum_{k=0}^{n}(-1)^k \frac{x^{2k}}{k!}$
9.	$x\sin(x)$	$\sum_{k=0}^{n}(-1)^k \frac{x^{2k+2}}{(2k+1)!}$

Продолжение таблицы 1.7

1	2	3
10.	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$	$\sum_{k=0}^{n}\frac{k^2+1}{k!}\left(\frac{x}{2}\right)^k$
11.	$\sin\left(x^3\right)$	$\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{3(2k-1)}}{(2k-1)!}$
12.	e^{2x^4}	$\sum_{k=0}^{n}\frac{2^k x^{4k}}{k!}$
13.	$sh\left(x^2\right)$	$\sum_{k=1}^{n}\frac{x^{4k-2}}{(2k-1)!}$
14.	$\left(1+2x^2\right)e^{x^2}$	$\sum_{k=0}^{n}\frac{2k+1}{k!}x^{2k}$
15.	$\left(x^2+1\right)\cos(x)$	$\sum_{k=0}^{n}(-1)^{k+1}\frac{x^{2k}\left(4k^2-2k-1\right)}{(2k)!}$
16.	$e^{-\frac{x}{2}}$	$\sum_{k=0}^{n}(-1)^k\frac{\left(\frac{x}{2}\right)^k}{k!}$
17.	$\cos(2x)-1$	$\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}$
18.	$\frac{e^x+e^{-x}}{2}$	$\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!}$
19.	$\sin^2(x)$	$\sum_{k=1}^{n}(-1)^{k+1}\frac{2^{2k-1}x^{2k}}{(2k)!}$
20.	$2\left(\cos^2(x)-1\right)$	$\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}$
21.	e^{2x}	$\sum_{k=0}^{n}\frac{(2x)^k}{k!}$
22.	$ch(5x)$	$\sum_{k=0}^{n}\frac{(5x)^{2k}}{(2k)!}$
23.	3^{x^3}	$\sum_{k=0}^{n}\frac{\ln^k(3)x^{3k}}{(2k)!}$
24.	$\frac{\sin(2x)}{x}$	$\sum_{k=0}^{n}(-1)^k\frac{2^{2k+1}x^{2k}}{(2k+1)!}$

Окончание таблицы 1.7

1	2	3
25.	$\cos\left(\sqrt{x}\right)$	$\sum_{k=0}^{n}(-1)^k \frac{x^k}{(2k)!}$
26.	e^{x^2}	$\sum_{k=0}^{n}\frac{x^{2k}}{k!}$
27.	$\sin^2(x)$	$\sum_{k=1}^{n}\frac{\cos\left((n+1)\pi\right)}{2}\frac{(2x)^{2k}}{(2k)!}$
28.	$\frac{1}{\sqrt{e^x}}$	$\sum_{k=0}^{n}(-1)^k \frac{x^k}{2^k k!}$
29.	$e^{\cos(x)}\cos\left(\sin(x)\right)$	$S(x)=\sum_{k=0}^{n}\frac{\cos(kx)}{k!}$
30.	e^{-x}	$\sum_{k=0}^{n}\frac{\cos(k\pi)x^k}{k!}$

Задание 2. «Программирование на языке С++ с использованием функций».

Общие требования:

1. Разработать функцию согласно заданию.

2. В основной программе разработать пользовательский интерфейс, обеспечивающий доступ к разработанной функции.

3. В основной программе использовать циклические операторы. Не использовать массивы.

4. В отчёте привести блок-схему алгоритма реализации функции.

Вариант № 1. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти количество его делителей (функцией оформить определение количества делителей числа).

Вариант № 2. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти сумму его простых делителей (функцией оформить определение суммы простых делителей числа).

Вариант № 3. Вводится последовательность целых чисел, 0 – конец последовательности. Найти наименьшую по значению цифру в каждом числе последовательности (функцией оформить определение наименьшей цифры числа).

Вариант № 4. Вводится последовательность целых чисел, 0 – конец последовательности. Найти наибольшую по значению четную цифру в каждом числе последовательности (функцией оформить определение

наибольшей четной цифры числа).

Вариант № 5. Вводится последовательность целых чисел, 0 – конец последовательности. Найти в каждом числе последовательности сумму четных цифр (функцией оформить определение суммы четных цифр числа).

Вариант № 6. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти наибольшее целое число K, для которого 2^K не превосходит исходное число (функцией оформить определение нового числа для заданного). Функцию логарифма и возведения в степень не использовать.

Вариант № 7. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности вывести новое число, которое состоит только из четных цифр исходного числа (функцией оформить определение нового числа для заданного).

Вариант № 8. Вводится последовательность целых чисел, 0 – конец последовательности. Найти в каждом числе последовательности количество четных и нечетных цифр (функциями оформить определение количества четных и нечетных цифр числа).

Вариант № 9. Вводится последовательность целых чисел, 0 – конец последовательности. Найти среднее арифметическое цифр каждого числа последовательности (функцией оформить определение среднего арифметического цифр числа).

Вариант № 10. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности функцией проверить, представляют ли его цифры строго убывающую последовательность, например, 6543 (результатом функции будет 1 – Да, 0 – Нет).

Вариант № 11. Вводится последовательность целых чисел, 0 – конец последовательности. Найти количество двух- и количество трехразрядных чисел в последовательности (функцией оформить определение количества разрядов числа).

Вариант № 12. Вводится последовательность натуральных чисел, 0 – конец последовательности. Для каждого числа последовательности найти наибольшее целое положительное число, квадрат которого не превосходит исходное число (функцией оформить определение нового числа для заданного). Функцию извлечения квадратного корня не использовать.

Вариант № 13. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности вывести новое число, которое получится после записи цифр числа в обратном порядке (функцией оформить определение нового числа для заданного).

Вариант № 14. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти количество цифр 5 (функцией оформить определение количество цифр 5).

Вариант № 15. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности функцией проверить, является ли оно совершенным (результатом функции будет 1 – Да, 0 – Нет).

Вариант № 16. Вводится последовательность целых чисел, 0 – конец последовательности. Уменьшить каждое число в 2 раза (деление нацело). Проверить, изменилось ли в числе после уменьшения количество разрядов (функцией оформить определение количества разрядов числа).

Вариант № 17. Вводится последовательность целых чисел, 0 – конец последовательности. Проверить являются ли числа простыми. Простые числа увеличить на натуральное число M. Проверить, остались ли они простыми (функцией оформить проверку числа: функция возвращает 1, если число простое, 0 – в противном случае).

Вариант № 18. Вводится последовательность целых чисел, 0 – конец последовательности. Каждое число последовательности перевести в двоичную систему счисления (функцией оформить перевод числа).

Вариант № 19. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности функцией проверить, является ли оно избыточным (результатом функции будет 1 – Да, 0 – Нет).

Вариант № 20. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности функцией проверить, является ли оно числом Фибоначчи (результатом функции будет 1 – Да, 0 – Нет).

Вариант № 21. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности вывести новое число, которое состоит только из нечетных цифр исходного числа (функцией оформить определение нового числа для заданного).

Вариант № 22. Вводится последовательность натуральных чисел, 0 – конец последовательности. Для каждого числа последовательности найти наименьшее натуральное число, квадрат которого превосходит исходное число (функцией оформить определение нового числа для заданного). Функцию извлечения квадратного корня не использовать.

Вариант № 23. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности функцией проверить, является ли оно палиндромом (результатом функции будет 1 – Да, 0 – Нет).

Вариант № 24. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности функцией проверить, представляют ли его цифры строго возрастающую последовательность, например, 1234 (результатом функции будет 1 – Да, 0 – Нет).

Вариант № 25. Вводится последовательность натуральных чисел, 0 – конец последовательности. Для каждого числа последовательности найти

наибольшее натуральное число, факториал которого не превосходит исходное число (функцией оформить определение нового числа для заданного).

Вариант № 26. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти наименьшее целое число К, для которого 2^K превосходит исходное число (функцией оформить определение нового числа для заданного).

Вариант № 27. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти количество цифр 0 (функцией оформить определение количество цифр 0).

Вариант № 28. Вводится последовательность целых чисел, 0 – конец последовательности. Найти для каждого числа последовательности произведение нечетных цифр (функцией оформить определение произведения нечетных цифр числа).

Вариант № 29. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности найти наибольшее число Фибоначчи, не превосходящее исходное число (функцией оформить определение нового числа для заданного).

Вариант № 30. Вводится последовательность целых чисел, 0 – конец последовательности. Для каждого числа последовательности определить максимальную цифру (функция определяет максимальную цифру в числе).

Задание 3. «Одномерные массивы».

Общие требования:

1. Использовать динамическое размещение данных.
2. Массив заполнять случайными значениями.

Вариант № 1. Имеется m результатов измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от 100 до 120. Определить количество измерений, с отклоняющимися от нормы значениями.

Вариант № 2. Имеется последовательность значений (целые числа в диапазоне от 0 до 150), отражающие изменение во времени некого определённого параметра. Параметр замерялся с интервалом t секунд. Определить, через сколько секунд после начала измерений будет достигнуто максимальное значение параметра.

Вариант № 3. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Подсчитать количество точек находящихся в третьей четверти.

Вариант № 4. Имеется m результатов измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от

100 до 120. Определить среднее значение нормальных результатов.

Вариант № 5. Имеется последовательность значений (целые числа в диапазоне от 0 до 150), отражающие изменение во времени определённого параметра. Параметр замерялся с интервалом t секунд. Определить, через сколько секунд от начала измерений значение параметра превысило 100 во второй раз.

Вариант № 6. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Определить, сколько процентов точек лежит внутри окружности диаметром 5 с центром в начале координат.

Вариант № 7. Имеется m результатов измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от 100 до 120. Определить среднее значение превышающих норму результатов.

Вариант № 8. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Выяснить, сколько точек находиться вне первой четверти.

Вариант № 9. Имеется последовательность значений (целые числа в диапазоне от 0 до 150), отражающие изменение во времени некого параметра. Параметр замерялся с интервалом t секунд. Определить, сколько секунд, с момента начала измерений, значение параметра возрастало.

Вариант № 10. Имеется последовательность значений (целые числа в диапазоне от 0 до 150), отражающие изменение во времени определённого параметра. Параметр замерялся с интервалом t секунд. Определить, среднее значение параметра в промежутке времени от t_1 до t_2 с момента начала измерений.

Вариант № 11. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от 100 до 120. Определить, сколько процентов от общего количества значений составляют значения меньше нормы.

Вариант № 12. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OX. Подсчитать среднее расстояние от точки до начала координат.

Вариант № 13. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Определить, сколько процентов от общего количества значений составляют значения лежащие в диапазоне от 30 до 50.

Вариант № 14. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Вывести координаты точки, расположенной дальше всех от начала координат

Вариант № 15. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от 100 до 120. Определить, каких значений больше, нормальных, или не соответствующих норме. Результатом работы программы должно быть текстовое сообщение.

Вариант № 16. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от 100 до 120. Определить, сколько процентов от общего количества значений составляют значения превышающие норму.

Вариант № 17. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Подсчитать, какой процент точек находится во второй четверти.

Вариант № 18. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Определить, какой процент результатов превышает значение 100.

Вариант № 19. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Подсчитать процент от общего количества точек находящихся в первой или третьей четверти.

Вариант № 20. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Определить количество измерений с результатом выше среднего значения.

Вариант № 21. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Подсчитать, в какой из четвертей лежит больше точек.

Вариант № 22. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Подсчитать, сколько точек лежит на осях координат.

Вариант № 23. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения

координат по оси OY. Подсчитать, во второй или четвертой четверти лежит больше точек. Результатом должно быть текстовое сообщение.

Вариант № 24. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Проверить, все ли точки лежат в первой и третьей четверти. Результатом должно быть текстовое сообщение.

Вариант № 25. Имеется последовательность значений (целые числа в диапазоне от 0 до 150), отражающие изменение во времени определённого параметра. Параметр замерялся с интервалом t секунд. Определить, через сколько секунд от начала измерений значение параметра в первый раз превысило среднее значение.

Вариант № 26. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Определить процент точек, лежащих внутри квадрата со стороной 4 и центром в начале координат.

Вариант № 27. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Вывести координаты точек, лежащих за пределами окружности радиусом 3 и центром в начале координат.

Вариант № 28. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Вывести координаты точки, лежащей ближе всего к заданной.

Вариант № 29. Имеется последовательность точек на координатной плоскости, заданная с помощью двух массивов, в одном из которых содержаться значения координат точек по оси OX, в другом – значения координат по оси OY. Подсчитать, где больше точек: на осях координат или вне их. Результатом должно быть текстовое сообщение.

Вариант № 30. Имеется m значений, являющихся результатом измерения некого параметра (целые значения в диапазоне от 0 до 150). Нормальными считаются значения от 100 до 120. Определить, лежит ли среднее значение в пределах нормы. Результатом должно быть текстовое сообщение.

Задание 4. «Двумерные массивы».

Общие требования:

1. Использовать динамическое размещение данных.
2. Массив заполнить случайными значениями.

Вариант № 1. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию первую строку.

Вариант № 2. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию последнюю строку.

Вариант № 3. Дан двумерный массив A, размером nxn (n > 5). Упорядочить по возрастанию пятую строку матрицы.

Вариант № 4. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию первый столбец.

Вариант № 5. Дан двумерный массив A, размером nxm. Затем упорядочить по убыванию вторую строку.

Вариант № 6. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию главную диагональ.

Вариант № 7. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию параллель побочной диагонали, расположенную под ней.

Вариант № 8. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию параллель главной диагонали, расположенную над ней.

Вариант № 9. Дан двумерный массив A, размером nxn. Упорядочить по Дан двумерный массив A, размером nxn. Упорядочить по возрастанию последний столбец.

Вариант № 10. Дан двумерный массив A, размером nxn (n > 3). Упорядочить по возрастанию третий столбец.

Вариант № 11. Дан двумерный массив A, размером nxn. Упорядочить по убыванию первую строку.

Вариант № 12. Дан двумерный массив A, размером nxn. Упорядочить по убыванию последнюю строку.

Вариант № 13. Дан двумерный массив A, размером nxm. Найти сумму положительных элементов в каждой строке. Затем упорядочить по убыванию созданный массив.

Вариант № 14. Дан двумерный массив A, размером nxm. Найти сумму положительных элементов в каждом столбце. Затем упорядочить по убыванию созданный массив.

Вариант № 15. Дан двумерный массив A, размером nxm. Найти среднее арифметическое положительных элементов в каждом столбце. Затем упорядочить по убыванию созданный массив.

Вариант № 16. Дан двумерный массив A, размером nxm. Найти среднее геометрическое положительных элементов в каждой строке. Затем упорядочить по убыванию созданный массив.

Вариант № 17. Дан двумерный массив A, размером nxm. Упорядочить по убыванию последний столбец.

Вариант № 18. Дан двумерный массив A, размером nxm (m > 3). Упорядочить по убыванию третий столбец.

Вариант № 19. убыванию главную диагональ.

Вариант № 20. Дан двумерный массив A, размером nxn. Упорядочить по убыванию параллель побочной диагонали, расположенную под ней.

Вариант № 21. Дан двумерный массив A, размером nxn. Упорядочить по убыванию параллель главной диагонали, расположенную над ней

Вариант № 22. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию параллель главной диагонали, расположенную под ней.

Вариант № 23. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию параллель побочной диагонали, расположенную над ней.

Вариант № 24. Дан двумерный массив A, размером nxn. Упорядочить по убыванию побочную диагональ.

Вариант № 25. Дан двумерный массив A, размером nxn. Упорядочить по убыванию параллель главной диагонали, расположенную под ней.

Вариант № 26. Дан двумерный массив A, размером nxn. Упорядочить по убыванию параллель побочной диагонали, расположенную над ней.

Вариант № 27. Дан двумерный массив A, размером nxn. Упорядочить по возрастанию побочную диагональ.

Вариант № 28. Дан двумерный массив A, размером nxn (n > 2). Упорядочить по убыванию второй столбец.

Вариант № 29. Дан двумерный массив A, размером mxn. Упорядочить по убыванию последнюю строку.

Вариант № 30. Дан двумерный массив A, размером nxn (n > 4). Упорядочить по убыванию четвертую строку.

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА 2. РЕШЕНИЕ ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Основные теоретические сведения

2.1 Графический метод решения ЗЛП

Если в задаче линейного программирования имеется только две переменные, то ее можно решить графическим методом.

Рассмотрим задачу линейного программирования с двумя переменными x_1 и x_2:

$$C = c_1 x_1 + c_2 x_2 \to \min(\max) \tag{2.1}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 \le b_1 \\ a_{21}x_1 + a_{22}x_2 \le b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 \ge b_m \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 \ge b_n \end{cases} \tag{2.2}$$

Здесь c_i, a_{jk} есть произвольные числа. Задача может быть как на нахождение максимума, так и на нахождение минимума. В системе ограничений могут присутствовать как знаки ≤, так и знаки ≥.

Графический метод решения задачи (2.1) следующий.

Вначале мы проводим оси координат x_1 и x_2 и выбираем масштаб. Каждое из неравенств системы ограничений (2.2) определяет полуплоскость, ограниченную соответствующей прямой.

Так, первое неравенство $a_{11}x_1 + a_{12}x_2 \le b_1$ определяет полуплоскость, ограниченную прямой $a_{11}x_1 + a_{12}x_2 = b_1$. С одной стороны от этой прямой $a_{11}x_1 + a_{12}x_2 < b_1$, а с другой стороны $a_{11}x_1 + a_{12}x_2 < b_1$. На самой прямой $a_{11}x_1 + a_{12}x_2 = b_1$. Чтобы узнать, с какой стороны выполняется неравенство $a_{11}x_1 + a_{12}x_2 \le b_1$, мы выбираем произвольную точку, не лежащую на прямой. Далее подставляем координаты этой точки в неравенство. Если оно выполняется, то полуплоскость решения содержит выбранную точку. Если неравенство не выполняется, то полуплоскость решения не содержит выбранную точку. Заштриховываем полуплоскость, для которой выполняется неравенство $a_{11}x_1 + a_{12}x_2 \le b_1$.

То же выполняем для остальных неравенств системы (2.2). Так мы получим n заштрихованных полуплоскостей. Точки области допустимых решений удовлетворяют всем неравенствам (2.2). Поэтому графически область допустимых решений (ОДР) является пересечением всех построенных полуплоскостей. Заштриховываем ОДР. Она представляет собой выпуклый многоугольник, грани которого принадлежат построенным прямым. Также ОДР может быть неограниченной выпуклой фигурой, отрезком, лучом или прямой.

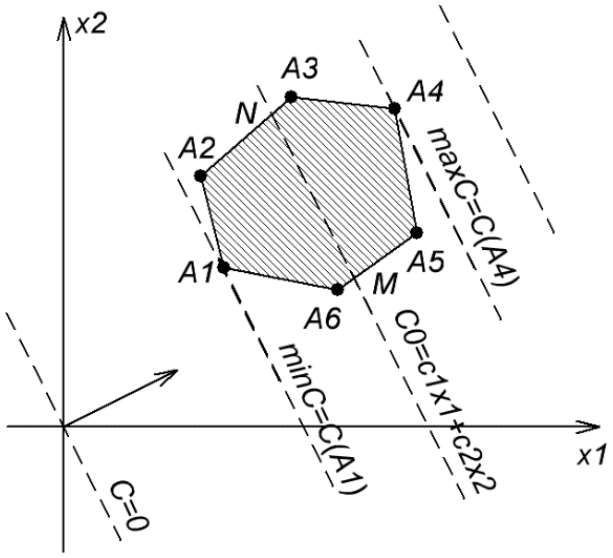

Рисунок 2.1 – Графическое решение ЗЛП

Может возникнуть и такой случай, что полуплоскости не содержат общих точек. Тогда областью допустимых решений является пустое множество. Такая задача решений не имеет.

Нахождение экстремума целевой функции

Итак, мы имеем заштрихованную область допустимых решений (ОДР). Она ограничена ломаной, состоящей из отрезков и лучей, принадлежащих построенным прямым. ОДР всегда является выпуклым множеством. Оно может быть как ограниченным множеством, так и не ограниченным вдоль некоторых направлений.

Теперь мы можем искать экстремум целевой функции (2.1).

Для этого выбираем любое число и строим прямую

$$C_0 = c_1 x_1 + c_2 x_2 . \qquad (2.3)$$

Для удобства дальнейшего изложения считаем, что эта прямая проходит через ОДР. На этой прямой целевая функция постоянна и равна C_0. такая прямая называется линией уровня функции C. Эта прямая разбивает плоскость на две полуплоскости. На одной полуплоскости $C = c_1 x_1 + c_2 x_2 > C_0$. На другой полуплоскости $C = c_1 x_1 + c_2 x_2 < C_0$. То есть с одной стороны от прямой (2.3) целевая функция возрастает. И чем дальше мы отодвинем точку от прямой (2.3), тем больше будет значение C. С другой стороны от прямой (2.3) целевая функция убывает. И чем дальше мы отодвинем точку от прямой (2.3) в другую сторону, тем меньше будет значение C. Если мы проведем прямую, параллельную прямой (2.3), то новая прямая также будет линией уровня целевой функции, но с другим значением C.

Таким образом, чтобы найти максимальное значение целевой функции, надо провести прямую, параллельную прямой (2.3), максимально удаленную от нее в сторону возрастания значений C, и проходящую хотя бы через одну точку ОДР. Чтобы найти минимальное значение целевой функции, надо провести прямую, параллельную прямой (2.3) и максимально удаленную от нее в сторону убывания значений C, и проходящую хотя бы через одну точку ОДР.

Если ОДР неограниченна, то может возникнуть случай, когда такую прямую провести нельзя. То есть, как бы мы ни удаляли прямую от линии уровня (2.3) в сторону возрастания (убывания), прямая всегда будет проходить через ОДР. В этом случае может быть сколь угодно большим (малым). Поэтому максимального (минимального) значения нет. Задача решений не имеет.

Рассмотрим случай, когда крайняя прямая, параллельная произвольной прямой вида (3), проходит через одну вершину многоугольника ОДР. Из графика определяем координаты этой вершины. Тогда максимальное (минимальное) значение целевой функции определяется по формуле

$$C_{\max} = c_1 x_{1A4} + c_2 x_{2A4}; \; C_{\min} = c_1 x_{1A1} + c_2 x_{2A1} .$$

Также может встретиться случай, когда прямая параллельна одной из граней ОДР. Тогда прямая проходит через две вершины многоугольника ОДР. Определяем координаты и этих вершин. Для определения максимального (минимального) значения целевой функции, можно использовать координаты любой из этих вершин. Задача имеет бесконечно много решений. Решением является любая точка, расположенная на отрезке прямой, принадлежащем ОДР.

2.2 Пример реализации графического метода в Matlab

Реализацию графического метода рассмотрим на следующем примере:

$$C = x_1 + 3x_2 \to \min(\max)$$

$$\begin{cases} 3x_1 + 4x_2 \geq 10 \\ 5x_1 - x_2 \geq 13 \\ 4x_1 + x_2 \leq 20 \\ x_1 \geq 0;\ x_2 \geq 0 \end{cases}$$

Задаем символьные переменные:

```
syms x1 x2
```

Проводим прямые, ограничивающие решения неравенств системы ограничений:

```
a=-1;
b=8;
eq1=3*x1+4*x2-10;
p1=ezplot(eq1,[a b a b]);
set(p1,'color','r')
hold on
eq2=5*x1-x2-13;
p2=ezplot(eq2,[a b a b]);
set(p2,'color','g')
eq3=4*x1+x2-20;
p3=ezplot(eq3,[a b a b]);
set(p3,'color','b')
plot([a b],[0 0],'k')
plot([0 0],[a b],'k')
legend('3*x1+4*x2=10','5*x1-x2=13','4*x1+x2=20','x2=0','x1=0')
grid on
```

Результат выполнения данного фрагмента представлен на рисунке 2.2.

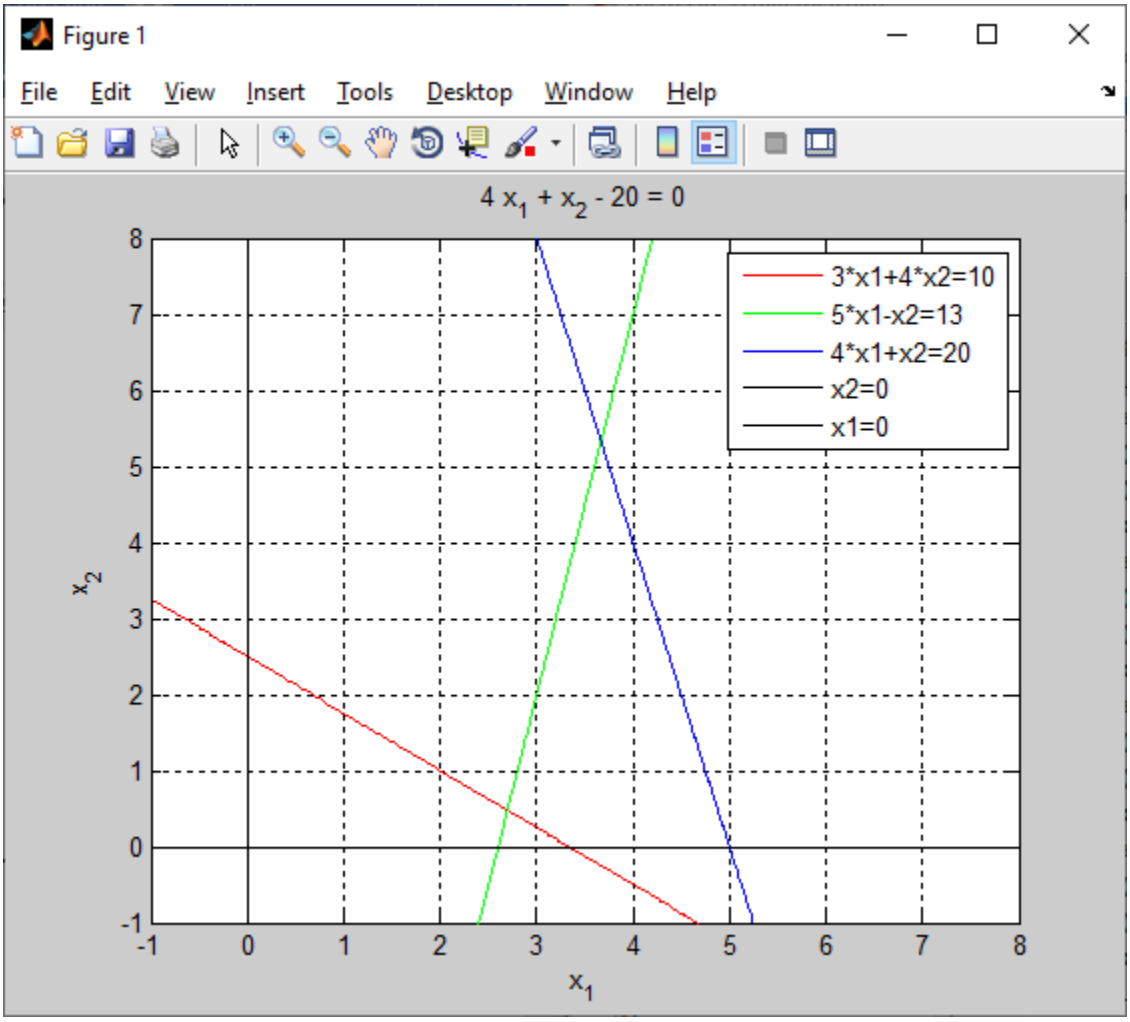

Рисунок 2.2 – Прямые, ограничивающие область допустимых решений

Определяем узловые точки области допустимых решений и закрашиваем ее:

```
s=solve(eq1,eq2);
X1(1)=s.x1;
X2(1)=s.x2;
text(double(X1(1)-0.5),double(X2(1)),' 1','FontSize',18)
s=solve(eq3,eq2);
X1(2)=s.x1;
X2(2)=s.x2;
text(double(X1(2)),double(X2(2)+0.3),' 2','FontSize',18)
s=solve(subs(eq3,x2,0));
X1(3)=s;
X2(3)=0;
text(double(X1(3)),double(X2(3)),' 3','FontSize',18)
s=solve(subs(eq1,x2,0));
X1(4)=s;
X2(4)=0;
text(double(X1(4))-0.3,double(X2(4)-0.5),' 4','FontSize',18)
fill(X1,X2,'y')
```

Результат построения представлен на рисунке 2.3.

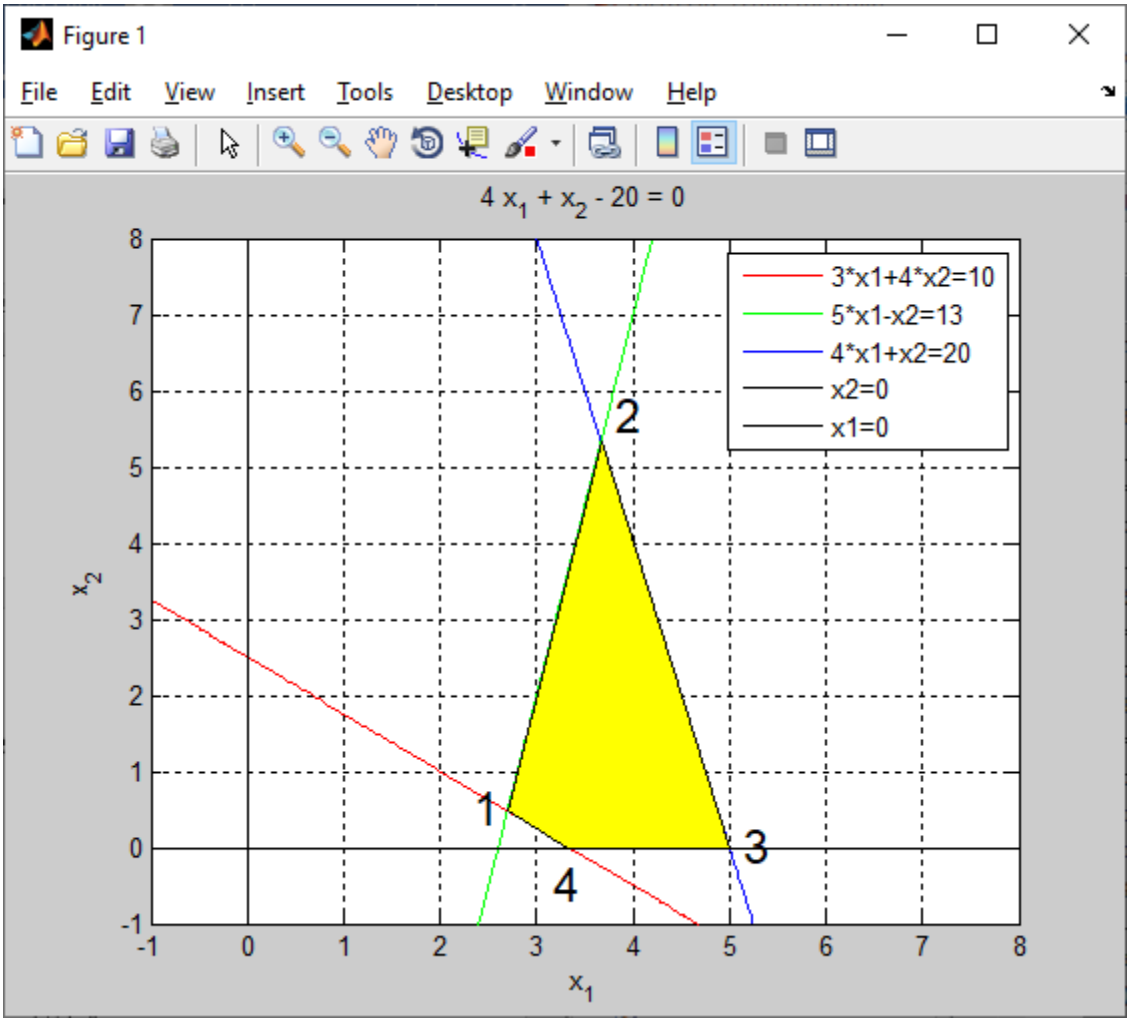

Рисунок 2.3 – Область допустимых решений

Выбираем произвольное значение целевой функции (например, 10) и строим соответствующую линию уровня:

```
C=x1+3*x2;
C0=C-10;
c0=ezplot(C0,[a b a b]);
set(c0,'color','c')
```

Результат построения представлен на рисунке 2.4.

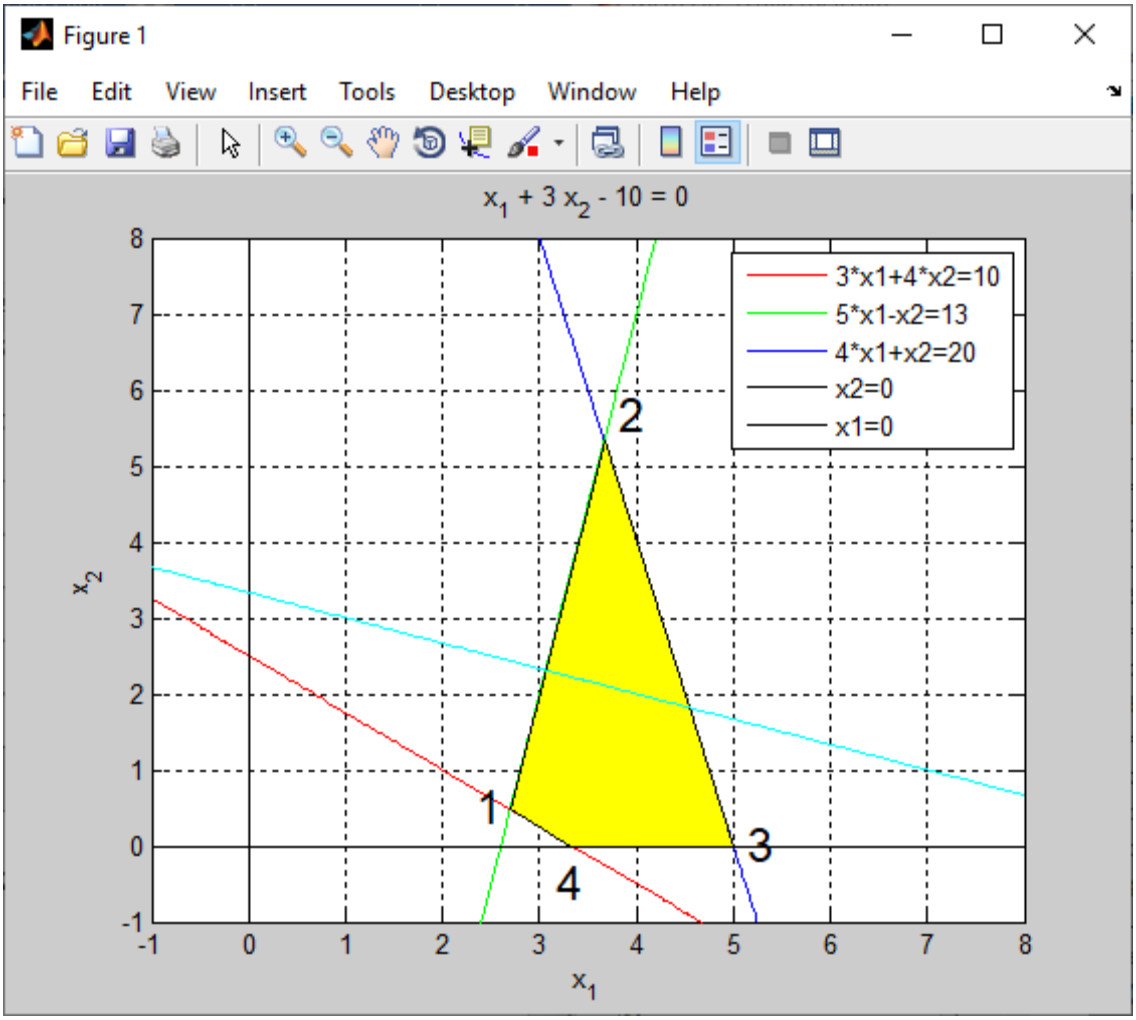

Рисунок 2.4 – Произвольная линия уровня

Видно, что минимум целевой функции будет в точке 4, а максимум – в точке 2. Выведем координаты точек 4 и 2 и определим минимум и максимум целевой функции:

```
xmin=[X1(4) X2(4)]
Cmin=subs(C,[x1 x2],[X1(4) X2(4)])
C1=C-Cmin;
c1=ezplot(C1,[a b a b]);
set(c1,'color','m')
xmax=[X1(2) X2(2)]
Cmax=subs(C,[x1 x2],[X1(2) X2(2)])
C2=C-Cmax;
c2=ezplot(C2,[a b a b]);
set(c2,'color',[0.5 0.5 0.5])
```

Получаем:

```
xmin =

[ 10/3, 0]
```

```
 Cmin =

10/3

xmax =

[ 11/3, 16/3]

Cmax =

59/3
```

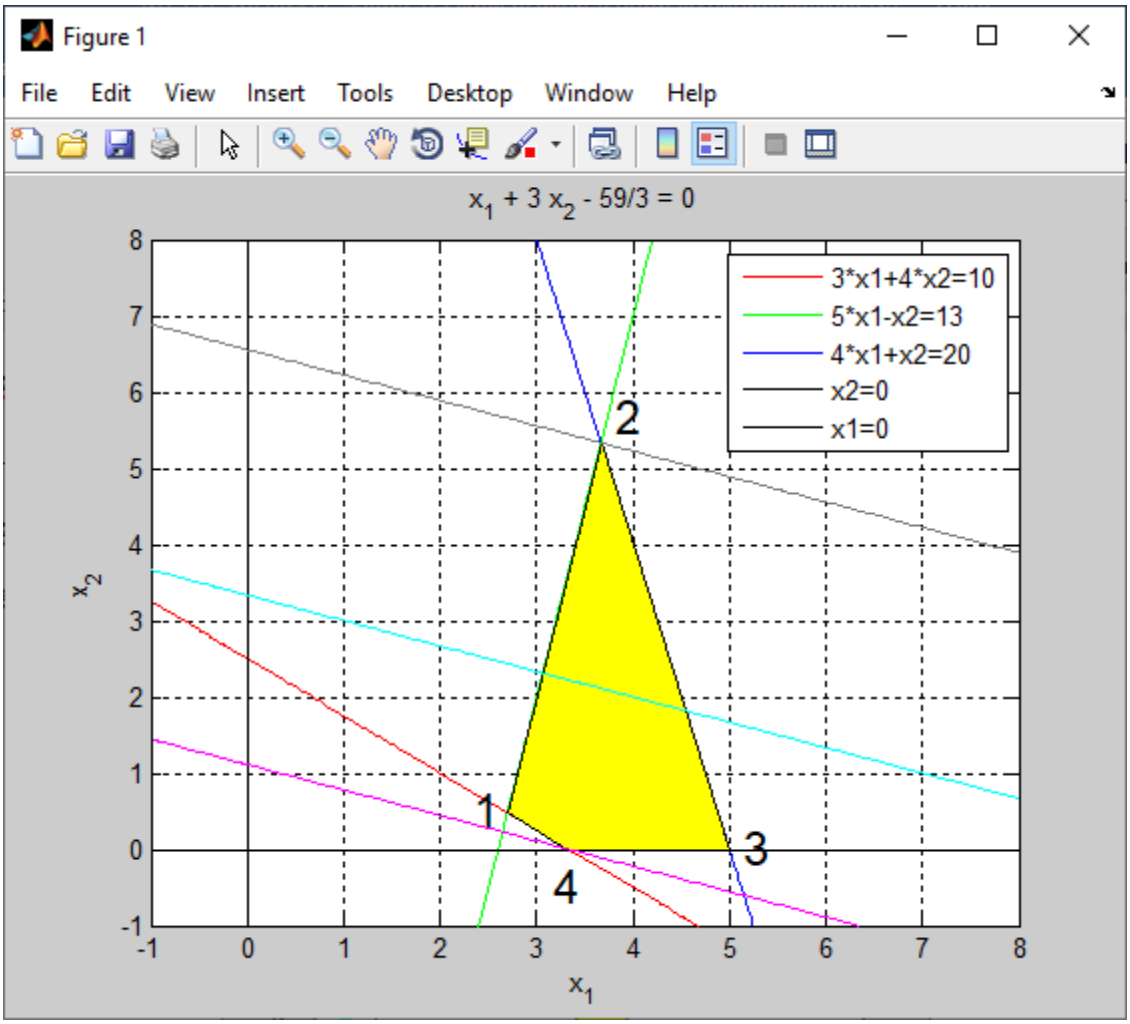

Рисунок 2.5 – График решения ЗЛП

2.3 Решение задачи линейного программирования средствами Matlab

Пусть задача линейного программирования сформулирована следующим образом. Требуется найти минимум целевой функции

$$C = c_1x_1 + c_2x_2 + ... + c_nx_n \to \min \tag{2.4}$$

при наложенных ограничениях

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \le b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \le b_2 \\ ... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \le b_m \\ aeq_{11}x_1 + aeq_{12}x_2 + ... + aeq_{1n}x_n = beq_1 \\ aeq_{21}x_1 + aeq_{22}x_2 + ... + aeq_{2n}x_n = beq_2 \\ ... \\ aeq_{k1}x_1 + aeq_{k2}x_2 + ... + aeq_{kn}x_n = beq_k \\ lb_1 \le x_1 \le ub_1 \\ lb_2 \le x_2 \le ub_2 \\ ... \\ lb_n \le x_n \le ub_n \end{cases} \tag{2.5}$$

или в матричной форме

$$\begin{cases} AX \le B \\ AeqX = Beq \\ Lb \le X \le Ub. \end{cases} \tag{2.6}$$

Для приведения имеющейся системы ограничений к виду (2.5) необходимо в неравенствах, содержащих знак «≥», умножить левые и правые части на -1. При этом знак неравенства станет «≤».

Поставленная задача решается с помощью функции:

```
[X,Cval,exi,out,La]=linprog(C,A,B,Aeq,Beq,Lb,Ub,X0,ops).
```

Аргументы функции:

`C` – вектор коэффициентов целевой функции;

`A` – матрица коэффициентов при неизвестных в ограничениях-неравенствах;

`B` – столбец свободных членов ограничений-неравенств;

`Aeq` – матрица коэффициентов при неизвестных в ограничениях-равенствах;

`Beq` – столбец свободных членов ограничений-равенств;

Lb – вектор, компоненты которого задают нижние граничные значения соответствующих переменных;

Ub – вектор, компоненты которого задают верхние граничные значения соответствующих переменных;

X0 – стартовый вектор – начальное приближение к решению;

ops – структура с параметрами оптимизации.

Выходные параметры:

X – возвращаемое значение, которое при exi=1 минимизирует целевую функцию;

Cval – значение целевой функции в точке X;

exi – индикатор характера завершения вычислений;

out – структура с полученными и используемыми параметрами оптимизации;

La – структура, содержащая в полях множители Лагранжа, вычисленные в точке X.

Для решения задачи представленной в расчетно-графической работе, достаточно использовать упрощенный вариант синтаксиса данной функции:

```
[X,Cval]=linprog(C,A,B,Aeq,Beq,Lb),
```

где Aeq=[], Beq=[] – пустые массивы, т. к. ограничения-равенства не заданы.

Для решения задачи нахождения максимума необходимо найти точку минимума функции –C и полученное значение Cval умножить на -1.

Решим с использованием функции linprog пример из п. 2.2:

Задаем вектор коэффициентов целевой функции:

```
C=[1 3];
```

В неравенствах, содержащих знак «≥», умножаем левые и правые части на -1. Затем задаем матрицу с коэффициентами при неизвестных в ограничениях и вектор свободных членов ограничений:

```
A=[-3 -4;-5 1;4 1];
B=[-10;-13;20];
```

Нижние границы значений переменных:

```
lb=[0 0];
```

Решаем задачу на минимум:

```
[Xmin, Cmin] = linprog(C, A, B, [], [], lb)
```

Задача на максимум:

```
C=-1*C;
[Xmax, Cmax] = linprog(C, A, B, [], [], lb);
Xmax,Cmax=-Cmax
```

Получаем:

```
Optimization terminated.

Xmin =

    3.3333
    0.0000

Cmin =

    3.3333

Optimization terminated.

Xmax =

    3.6667
    5.3333

Cmax =

   19.6667
```

Индивидуальные задания

1. Решить заданную задачу линейного программирования графическим методом в Matlab.
2. Проверить правильность решения с помощью команды linprog.

Таблица 2.1 – Варианты задания

Вариант 1	Вариант 2	Вариант 3
$C = 3x_1 + 10x_2 \to \min(\max)$	$C = 7x_1 + 10x_2 \to \min(\max)$	$C = 4x_1 + x_2 \to \min(\max)$
$\begin{cases} -x_1 + 8x_2 \ge 2 \\ -x_1 + 2x_2 \le 7 \\ 2x_1 - x_2 \le 7 \\ 7x_1 + 3x_2 \ge 3 \\ x_1 + 2x_2 \le 19 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$\begin{cases} x_1 - 7x_2 \le 7 \\ -5x_1 + 3x_2 \le 1 \\ x_1 + 4x_2 \le 15 \\ 9x_1 + 4x_2 \ge 12 \\ 2x_1 + 4x_2 \le 21 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$\begin{cases} -2x_1 + 3x_2 \le 12 \\ x_1 - 4x_2 \le 1 \\ x_1 + 2x_2 \le 15 \\ 3x_1 + 2x_2 \ge 5 \\ 2x_1 + x_2 \le 20 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$

Продолжение таблицы 2.1

Вариант 4	Вариант 5	Вариант 6
$C = 2x_1 + 3x_2 \to \min(\max)$ $\begin{cases} 2x_1 - 7x_2 \le 6 \\ -x_1 + 2x_2 \le 3 \\ 4x_1 - x_2 \ge 6 \\ 5x_1 + 2x_2 \le 17 \\ 3x_1 + 10x_2 \ge 8 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = 7x_1 - 3x_2 \to \min(\max)$ $\begin{cases} x_1 - 6x_2 \le 2 \\ -3x_1 + 6x_2 \le 4 \\ 4x_1 - x_2 \ge 1 \\ 3x_1 + 4x_2 \ge 3 \\ x_1 + 2x_2 \le 5 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = x_1 + 5x_2 \to \min(\max)$ $\begin{cases} x_1 + 2x_2 \le 14 \\ -5x_1 + 3x_2 \le 8 \\ -x_1 + 24x_2 \ge 24 \\ 2x_1 + x_2 \le 16 \\ 2x_1 - x_2 \le 12 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$
Вариант 7	Вариант 8	Вариант 9
$C = 2x_1 + x_2 \to \min(\max)$ $\begin{cases} 4x_1 - 10x_2 \le 5 \\ -5x_1 + 10x_2 \le 13 \\ 9x_1 - x_2 \ge 3 \\ 5x_1 + 7x_2 \ge 4 \\ 3x_1 + 3x_2 \le 10 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = x_1 + 4x_2 \to \min(\max)$ $\begin{cases} x_1 - 2x_2 \le 8 \\ -2x_1 + 3x_2 \le 3 \\ 9x_1 - 5x_2 \ge 4 \\ 3x_1 + 4x_2 \ge 7 \\ x_1 + 2x_2 \le 11 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = x_1 + x_2 \to \min(\max)$ $\begin{cases} 2x_1 - x_2 \le 6 \\ -x_1 + 2x_2 \le 11 \\ 4x_1 + x_2 \le 20 \\ -2x_1 + x_2 \le 5 \\ 2x_1 + 3x_2 \ge 10 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$
Вариант 10	Вариант 11	Вариант 12
$C = 6x_1 + 7x_2 \to \min(\max)$ $\begin{cases} x_1 - 6x_2 \le 3 \\ -x_1 + 2x_2 \le 2 \\ 5x_1 - x_2 \ge 1 \\ 6x_1 + 4x_2 \ge 5 \\ 2x_1 + 2x_2 \le 7 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = 7x_1 + 6x_2 \to \min(\max)$ $\begin{cases} 4x_1 - 5x_2 \le 9 \\ -2x_1 + 3x_2 \le 6 \\ 8x_1 - x_2 \ge 2 \\ 3x_1 + 4x_2 \ge 5 \\ x_1 + x_2 \le 4 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = 5x_1 + 2x_2 \to \min(\max)$ $\begin{cases} 7x_1 + 5x_2 \ge 8 \\ -x_1 + 9x_2 \ge 2 \\ 2x_1 + 3x_2 \le 15 \\ -3x_1 + x_2 \le 4 \\ 5x_1 + x_2 \le 17 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$
Вариант 13	Вариант 14	Вариант 15
$C = 7x_1 + 2x_2 \to \min(\max)$ $\begin{cases} 4x_1 + 3x_2 \le 10 \\ 8x_1 - 5x_2 \ge 3 \\ -2x_1 + 3x_2 \le 1 \\ 4x_1 + 9x_2 \ge 5 \\ 3x_1 + x_2 \le 7 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = 6x_1 - x_2 \to \min(\max)$ $\begin{cases} x_1 - 6x_2 \le 1 \\ -8x_1 + 5x_2 \le 5 \\ 10x_1 - x_2 \ge 1 \\ 4x_1 + 1x_2 \le 8 \\ x_1 + 2x_2 \le 5 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$	$C = -2x_1 + 3x_2 \to \min(\max)$ $\begin{cases} x_1 - 2x_2 \le 2 \\ -9x_1 + 2x_2 \le 1 \\ 22x_1 - x_2 \ge 8 \\ 8x_1 + 5x_2 \ge 5 \\ x_1 + 2x_2 \le 9 \\ x_1 \ge 0, x_2 \ge 0 \end{cases}$

Продолжение таблицы 2.1

Вариант 16	Вариант 17	Вариант 18
$C = 3x_1 + x_2 \to \min(\max)$ $\begin{cases} -x_1 + 11x_2 \geq 2 \\ -x_1 + 2x_2 \leq 4 \\ -7x_1 + x_2 \leq 1 \\ 5x_1 + 6x_2 \geq 2 \\ x_1 + 2x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 10x_1 + x_2 \to \min(\max)$ $\begin{cases} x_1 - 5x_2 \leq 5 \\ -x_1 + 2x_2 \leq 2 \\ 5x_1 - 3x_2 \geq 12 \\ 3x_1 + x_2 \geq 11 \\ x_1 + x_2 \leq 10 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = x_1 + 9x_2 \to \min(\max)$ $\begin{cases} 9x_1 + 3x_2 \geq 8 \\ -4x_1 + x_2 \leq 4 \\ 3x_1 + 9x_2 \geq 13 \\ 4x_1 - 2x_2 \leq 1 \\ x_1 + x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$
Вариант 19	Вариант 20	Вариант 21
$C = 2x_1 + 5x_2 \to \min(\max)$ $\begin{cases} 4x_1 - 9x_2 \leq 7 \\ -x_1 + 2x_2 \leq 3 \\ 11x_1 - 2x_2 \geq 1 \\ 7x_1 + 3x_2 \geq 8 \\ 6x_1 + 7x_2 \leq 27 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 5x_1 + 4x_2 \to \min(\max)$ $\begin{cases} x_1 - x_2 \leq 2 \\ -2x_1 + 4x_2 \leq 5 \\ 8x_1 - 2x_2 \geq 7 \\ 10x_1 + 4x_2 \geq 14 \\ x_1 + 4x_2 \leq 13 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 2x_1 + x_2 \to \min(\max)$ $\begin{cases} 8x_1 + 8x_2 \geq 9 \\ 6x_1 - 5x_2 \geq 3 \\ 5x_1 - 4x_2 \leq 7 \\ -5x_1 + 13x_2 \leq 2 \\ x_1 + 4x_2 \leq 4 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$
Вариант 22	Вариант 23	Вариант 24
$C = x_1 + x_2 \to \min(\max)$ $\begin{cases} x_1 - 3x_2 \leq 1 \\ -3x_1 + 2x_2 \leq 1 \\ 9x_1 - 2x_2 \geq 1 \\ 7x_1 + 5x_2 \geq 4 \\ 4x_1 + 5x_2 \leq 9 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 3x_1 + 6x_2 \to \min(\max)$ $\begin{cases} x_1 + x_2 \geq 1 \\ x_1 + 5x_2 \leq 9 \\ 12x_1 - 2x_2 \geq 6 \\ 8x_1 + 3x_2 \leq 15 \\ 4x_1 + 3x_2 \leq 9 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 5x_1 + 7x_2 \to \min(\max)$ $\begin{cases} 4x_1 + 4x_2 \leq 3 \\ -3x_1 + 9x_2 \leq 5 \\ -5x_1 + 12x_2 \geq 1 \\ 7x_1 + 5x_2 \geq 1 \\ 4x_1 + 8x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$
Вариант 25	Вариант 26	Вариант 27
$C = 5x_1 + 3x_2 \to \min(\max)$ $\begin{cases} 2x_1 + 7x_2 \geq 2 \\ -3x_1 + 3x_2 \leq 2 \\ x_1 + 6x_2 \leq 5 \\ 9x_1 - x_2 \leq 7 \\ 4x_1 + 6x_2 \leq 7 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 7x_1 + 5x_2 \to \min(\max)$ $\begin{cases} 9x_1 + 5x_2 \geq 4 \\ 4x_1 - 3x_2 \geq 1 \\ 7x_1 + 3x_2 \leq 8 \\ 7x_1 - 9x_2 \leq 6 \\ 3x_1 + 9x_2 \leq 8 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 5x_1 - x_2 \to \min(\max)$ $\begin{cases} 2x_1 + 3x_2 \geq 1 \\ x_1 + 7x_2 \leq 2 \\ 6x_1 - 7x_2 \geq 1 \\ x_1 + 4x_2 \leq 3 \\ x_1 + 9x_2 \leq 4 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$

Окончание таблицы 2.1

Вариант 28	Вариант 29	Вариант 30
$C = 2x_1 + x_2 \to \min(\max)$ $\begin{cases} 8x_1 + 8x_2 \geq 9 \\ 6x_1 - 5x_2 \leq 11 \\ 5x_1 - 4x_2 \geq 4 \\ 5x_1 - 11x_2 \geq 1 \\ x_1 + 4x_2 \leq 4 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 6x_1 + 5x_2 \to \min(\max)$ $\begin{cases} 3x_1 + 2x_2 \leq 8 \\ 2x_1 - 5x_2 \leq 3 \\ 4x_1 + 5x_2 \leq 15 \\ -7x_1 + 4x_2 \leq 1 \\ 3x_1 + 5x_2 \geq 2 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$	$C = 3x_1 + 2x_2 \to \min(\max)$ $\begin{cases} 8x_1 + 3x_2 \geq 5 \\ -x_1 + 7x_2 \leq 7 \\ 6x_1 - 9x_2 \leq 8 \\ 7x_1 - 7x_2 \leq 11 \\ 7x_1 + 9x_2 \geq 7 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$

ЛИТЕРАТУРА

1. Васильев, А. Н. Программирование на C++ в примерах и задачах / А. Н. Васильев. – Москва : Э, 2017. – 364 с.

2. Гилат, Амос. Теория и практика / Амос Гилат ; пер. с англ. – 5-е изд. – Москва : ДМК Пресс, 2016. – 416 с.

3. Орленко, П. А. C++ на примерах. Практика, практика и только практика / П. А.Орленко, П. В. Евдокимов. – Санкт-Петербург : Наука и техника, 2019. – 288 с.

4. Прата, Стивен. Язык программирования C++ (C++11). Лекции и упражнения / Стивен Прата ; пер. с англ. – 6-е изд. – Москва : Вильямс, 2012. – 1248 с.

5. Страуструп, Бьярне. Язык программирования C++ / Бьярне Страуструп ; пер. с англ. – Москва : Бином, 2011. – 1136 с.

6. Тимохин, А. Н. Моделирование систем управления с применением MatLab : учебное пособие для студентов высших учебных заведений, обучающихся по направлению подготовки 15.03.04 «Автоматизация технологических процессов и производств» (квалификация (степень) «бакалавр») / А. Н. Тимохин, Ю. Д. Румянцев ; Московский государственный университет дизайна и технологии ; под ред. А. Н. Тимохина. – Москва : ИНФРА-М, 2019. – 255 с.

7. Шилдт, Герберт. C++. Базовый курс / Герберт Шилдт ; пер. с англ. – 3-е изд. – Москва : Вильямс, 2019. – 624 с.

Учебное издание

ОСНОВЫ КОМПЬЮТЕРИЗАЦИИ ТЕХНОЛОГИЙ В СИСТЕМАХ АВТОМАТИКИ.

Методические указания по выполнению расчетно-графических работ

Составители:
Соколова Анна Сергеевна
Казаков Вадим Евгеньевич
Шут Виктор Николаевич

Редактор *А.В. Пухальская*
Корректор *А.В. Пухальская*
Компьютерная верстка *А.С. Соколова*

Подписано к печати 17.04.2023. Формат 60x90 $^{1}/_{16}$. Усл. печ. листов 3,6.
Уч.-изд. листов 3,9. Тираж 2 экз. Заказ № 111.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.

Учебное издание

ОСНОВЫ КОМПЬЮТЕРИЗАЦИИ ТЕХНОЛОГИЙ В СИСТЕМАХ АВТОМАТИКИ.

Методические указания по выполнению расчетно-графических работ

Составители:
Соколова Анна Сергеевна
Казаков Вадим Евгеньевич
Шут Виктор Николаевич

Редактор *А.В. Пухальская*
Корректор *А.В. Пухальская*
Компьютерная верстка *А.С. Соколова*

Подписано к печати 17.04.2023. Усл. печ. листов 3,6.
Уч.-изд. листов 3,9. Заказ № 112.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 72.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.