



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
Брянский государственный технический университет

Д.И. Булатицкий, Е.В. Коптенок, Р.А. Исаев, А.О. Радченко

**ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ:
ВЕТВЛЕНИЯ, ЦИКЛЫ, МАССИВЫ, ФАЙЛЫ**

Утверждено редакционно-издательским советом университета
в качестве практикума

Брянск
ИЗДАТЕЛЬСТВО БГТУ
2018

Булатицкий, Д.И. Программирование на языке Си: Ветвления, циклы, массивы, файлы [Текст] + [Электронный ресурс]: практикум / Д.И. Булатицкий, Е.В. Коптенок, Р.А. Исаев, А.О. Радченко – Брянск: БГТУ, 2018. – 180 с.

ISBN 978-5-907111-61-5

Приводится описание семнадцати лабораторных работ и одной расчетно-графической работы по дисциплине «Программирование».

Практикум предназначен для студентов очной формы обучения по направлениям подготовки: 09.03.01 «Информатика и вычислительная техника», 09.03.04 – «Программная инженерия», 02.03.03 «Математическое обеспечение и администрирование информационных систем».

Ил. 81. Табл. 8. Библиогр. – 23 назв.

Научный редактор А.А.Азарченков

Рецензенты: кафедра математики, информационных технологий и информационного права Брянского филиала ФГБОУ ВО «Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации»;
кандидат технических наук Б.Н.Прусс

ISBN 978-5-907111-61-5

© Брянский государственный
технический университет, 2018

ПРЕДИСЛОВИЕ

Практикум предназначен для студентов очной формы обучения по направлениям подготовки: 09.03.01 «Информатика и вычислительная техника», 09.03.04 – «Программная инженерия», 02.03.03 «Математическое обеспечение и администрирование информационных систем» для изучения в первом семестре первой части дисциплины «Программирование». Оно также может использоваться студентами других форм обучения и родственных направлений подготовки.

Практикум способствует более глубокому изучению дисциплины «Программирование»: позволяет установить и закрепить связь между теорией программирования и практикой решения задач с помощью ЭВМ, развивает у студентов навыки разработки программного обеспечения.

В практикуме приводится описание семнадцати лабораторных работ и одной расчетно-графической работы. В каждом описании лабораторной работы содержатся краткие теоретические сведения, необходимые для выполнения лабораторной работы; подробное пошаговое иллюстрированное описание действий, общих для всех студентов; индивидуальные задания; задачи для самостоятельного решения и контрольные вопросы.

Лабораторные работы для удобства сгруппированы по темам в главы. В каждой следующей главе (и даже каждой лабораторной работе) используется понятийный аппарат и инструментарий, изученный в предыдущих лабораторных работах, поэтому студентам рекомендуется выполнять лабораторные работы в том порядке, в каком они приведены в практикуме. Расчетно-графическая работа выполняется студентами самостоятельно и позволяет обобщить и закрепить полученные навыки программирования.

В конце практикума приведен список литературы, который может оказать помощь студентам в изучении дисциплины «Программирование».

Рекомендуется следующий порядок выполнения лабораторной работы. Сначала рекомендуется изучить теоретические сведения по теме лабораторной работы по учебникам, конспекту лекций и параграфу «Краткие теоретические сведения», затем выполнить по шагам действия, описанные в параграфе «Общая часть

лабораторной работы». После этого необходимо решить задачи индивидуальной части лабораторной работы и решить задачи для самостоятельного решения по своему варианту. Если при решении задач своего варианта студент испытывает затруднения, то ему рекомендуется дополнительно прорешать задачи других вариантов. И наконец студентам предлагается ответить на контрольные вопросы. Если при ответе на какие-то контрольные вопросы студент испытывает затруднения, он может обратиться к рекомендуемой литературе. Если даже с помощью рекомендуемой литературы не удалось ответить на них, студенту рекомендуется обратиться за помощью к своим одноклассникам или к преподавателю. Для успешного завершения выполнения лабораторной работы студенту необходимо защитить её результаты. При защите лабораторной работы студент показывает все разработанные программы, выполняет их тестовые запуски, даёт обоснование правильности получаемых программой результатов, а также отвечает на вопросы преподавателя.

Основой для выбора варианта задачи для самостоятельного решения является номер студента в списке группы. Если другое не определено непосредственно в задании, то номер варианта выбирается следующим образом. Вычисляется остаток от целочисленного деления номера студента в списке группы на количество вариантов. Если этот остаток отличен от нуля, то он и принимается в качестве номера варианта. Иначе выбирается наибольший номер варианта.

Практикум подготовлен коллективом авторов. За общую координацию действий авторов, разработку тематики лабораторных работ и последовательность их подачи отвечал Д.И. Булатицкий. Подбором теоретического материала занимались совместно Д.И. Булатицкий и Е.В. Коптенок. Е.В. Коптенок отвечала за подбор большинства примеров задач, описание решения этих задач и компьютерный набор. Особой заслугой Р.А. Исаева является разработка вариантов задач для самостоятельного решения. А.О. Радченко отвечал за подготовку многих примеров программ, экранных снимков их работы, листингов.

ГЛАВА 1. ОСНОВЫ ПРОГРАММИРОВАНИЯ

Лабораторная работа №1. Разработка простых алгоритмов и их запись различными способами

1. Цель лабораторной работы

Цель лабораторной работы – изучить понятие и свойства алгоритма, основные способы записи алгоритмов, условные обозначения в блок-схемах и структурограммах.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Общие сведения об алгоритмах

Алгоритм – это последовательность чётко определенных действий, выполнение которых приводит к решению некоторой задачи.

Название «алгоритм» произошло от латинской формы имени среднеазиатского математика аль-Хорезми (Alhorithmi), жившего в 783-850 гг. [1].

Исполнитель алгоритма – это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом.

Часто исполнителем является компьютер, но понятие алгоритма необязательно относится к компьютерным программам. Так, чётко описанный рецепт приготовления блюда также является алгоритмом, в таком случае исполнителем является человек.

Рассмотрим *свойства алгоритма* [2].

1. **Дискретность** – это разбиение алгоритма на ряд отдельных законченных действий (шагов).

2. **Детерминированность** (от лат. determinate – определенность, точность) – любое действие алгоритма должно быть строго и недвусмысленно определено в каждом случае.

3. **Понятность** – алгоритм должен содержать только те команды, которые доступны исполнителю и входят в его систему команд.

4. **Конечность** – каждое действие в отдельности и алгоритм в целом должны иметь возможность завершения.

5. **Массовость** – алгоритм должен быть применим к разным наборам исходных данных.

6. **Результативность** – алгоритм должен приводить к достоверному решению.

Примерами алгоритмов могут являться:

- кулинарный рецепт;
- инструкция к прибору;
- описание маршрута и др.

Различают три основных вида алгоритмов:

- линейный алгоритм;
- разветвляющийся алгоритм;
- циклический алгоритм.

Линейный алгоритм – это алгоритм, в котором действия выполняются однократно и строго последовательно. Примером линейного алгоритма может являться последовательность действий при приготовлении бутерброда. Алгоритм в этом случае может выглядеть следующим образом [1]:

- 1) начало;
- 2) отрезать хлеб;
- 3) намазать хлеб маслом;
- 4) отрезать колбасу;
- 5) положить колбасу на хлеб;
- 6) конец.

Все действия выполняются последовательно друг за другом в определенном порядке и приводят к конечному результату.

Разветвляющийся алгоритм – это алгоритм, в котором некоторая последовательность действий либо выполняется, либо не выполняется в зависимости от условия. Простой пример реализации разветвляющегося алгоритма: *если* на улице идет дождь, *то* необходимо взять зонт, *иначе* не брать зонт с собой.

Циклический алгоритм – это алгоритм, предусматривающий многократное повторение одной и той же последовательности действий над разными исходными данными. Простой пример реализации циклического алгоритма – чтение книги, при котором будут повторяться одни и те же действия: прочитать страницу, перелистнуть на следующую, прочитать страницу и т.д.

Циклические алгоритмы бывают трех видов:

- **цикл со счетчиком** – действия внутри цикла выполняются определенное количество раз;
- **цикл с предусловием** – перед каждым новым проходом цикла проверяется определенное условие, если оно истинно, действия внутри цикла выполняются, в противном случае происходит выход из цикла;
- **цикл с постусловием** – вначале выполняются действия внутри цикла, а только затем проверка условия.

Формы записи алгоритма [3]:

- словесная, или вербальная (языковая, формульно-словесная);
- псевдокод;
- формальные алгоритмические языки;
- схематическая:
 - структурограммы (схемы Насси-Шнейдермана);
 - графическая (блок-схемы).

Обычно сначала, на уровне идеи, алгоритм описывается словами, но по мере приближения к реализации он обретает всё более формальные очертания и формулировку на языке, понятном исполнителю (например, машинный код).

Пример словесной записи алгоритма приведен при рассмотрении линейного алгоритма. Примером записи алгоритма с помощью формального языка является код любой программы на любом языке программирования (Си, Pascal, Basic и др.). Способы схематической записи алгоритмов подробно рассматриваются далее.



Блок-схемы

Правила выполнения блок-схем алгоритмов определяются документом ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

Этот стандарт, в частности, регулирует способы построения схем и внешний вид используемых символов. Основные символы схем алгоритмов представлены в таблице 1 [2].

Основные элементы схем алгоритма

Наименование	Обозначение	Функция
Блок начало-конец (пуск-остановка)		Символ отображает вход из внешней среды или выход из неё (чаще всего обозначает начало и конец программы). Внутри фигуры записывается соответствующее действие
Блок вычислений (вычислительный блок)		Выполнение одной или нескольких операций, обработка данных любого вида (изменение значения данных, формы представления, расположения). Внутри фигуры записывают непосредственно сами операции, например, операцию присваивания: $a = 10 * b + c$
Логический блок (блок условия)		Отображает решение или функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления выражения-условия внутри этого элемента. Вход в элемент обозначается линией, входящей обычно в верхнюю вершину элемента. Если выходов два или три, то обычно каждый выход обозначается линией, выходящей из оставшихся вершин (боковых и нижней). Примеры решения: в общем случае – сравнение (три выхода: $>$, $<$, $=$); в программировании – условные операторы <code>if</code> (два выхода: <code>true</code> , <code>false</code>) и <code>case</code> (множество выходов)
Предопределённый процесс		Символ отображает выполнение процесса, состоящего из одной или нескольких операций, который определен в другом месте программы (в подпрограмме, модуле). Внутри символа записывается название процесса и передаваемые в него данные, например, в программировании – вызов процедуры или функции
Данные (ввод-вывод)		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод). Этот символ не определяет носителя данных (для указания типа носителя данных используются специфические символы)
Граница цикла		Символ состоит из двух частей – соответственно начало и конец цикла – операции, выполняемые внутри цикла, размещаются между ними. Условия цикла и приращения записываются внутри символа начала или конца цикла – в зависимости от типа организации цикла. Часто для изображения на блок-схеме цикла вместо этого символа используют символ условия, указывая в нём условие цикла, а одну из линий выхода замыкают выше в блок-схеме (перед операциями цикла)
Соединитель		Символ отображает вход в часть схемы и выход из другой части этой схемы. Используется для обрыва линии и продолжения её в другом месте (для избежания излишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц). Соответствующие соединительные символы должны иметь одинаковое (притом уникальное) обозначение

Наименование	Обозначение	Функция
Комментарий		Символ позволяет привести более подробное описание шага, процесса или группы процессов. Описание помещается со стороны квадратной скобки и охватывается ей по всей высоте. Пунктирная линия идет к описываемому элементу, либо группе элементов. Также символ комментария следует использовать в тех случаях, когда объём текста, помещаемого внутри некоего символа, превышает размер самого этого символа.
Цикл счетчиком	so 	Условие цикла (начальное значение счетчика, конечное значение и шаг) записываются внутри шестиугольника. Ниже располагается тело цикла, возвращающее управление счетчику. Выход из цикла указывается с помощью боковой стрелки.

Рассмотрим пример построения блок-схемы алгоритма решения полного квадратного уравнения $Ax^2 + Bx + C = 0$. Для начала запишем алгоритм в вербальной форме:

1. Начало.
2. Ввод А.
3. Ввод В.
4. Ввод С.
5. Вывод уравнения на экран.
6. Вычисление дискриминанта по формуле $D = B^2 - 4AC$.
7. Если $D < 0$, вывести сообщение об отсутствии корней и перейти к шагу 10, иначе перейти к шагу 8.
8. Если $D = 0$, вычислить по формуле $x = -\frac{B}{2A}$ единственный корень уравнения, вывести его на экран и перейти к шагу 10. В противном случае перейти к шагу 9.
9. Вычислить корни уравнения по формулам $x_1 = \frac{-B + \sqrt{D}}{2A}$, $x_2 = \frac{-B - \sqrt{D}}{2A}$, вывести результаты на экран.
10. Конец.

Последовательность действий задается управляющими стрелками, связывающими блоки, внутри графических элементов записывается соответствующая команда. Блок-схема алгоритма приведена на рис. 1.

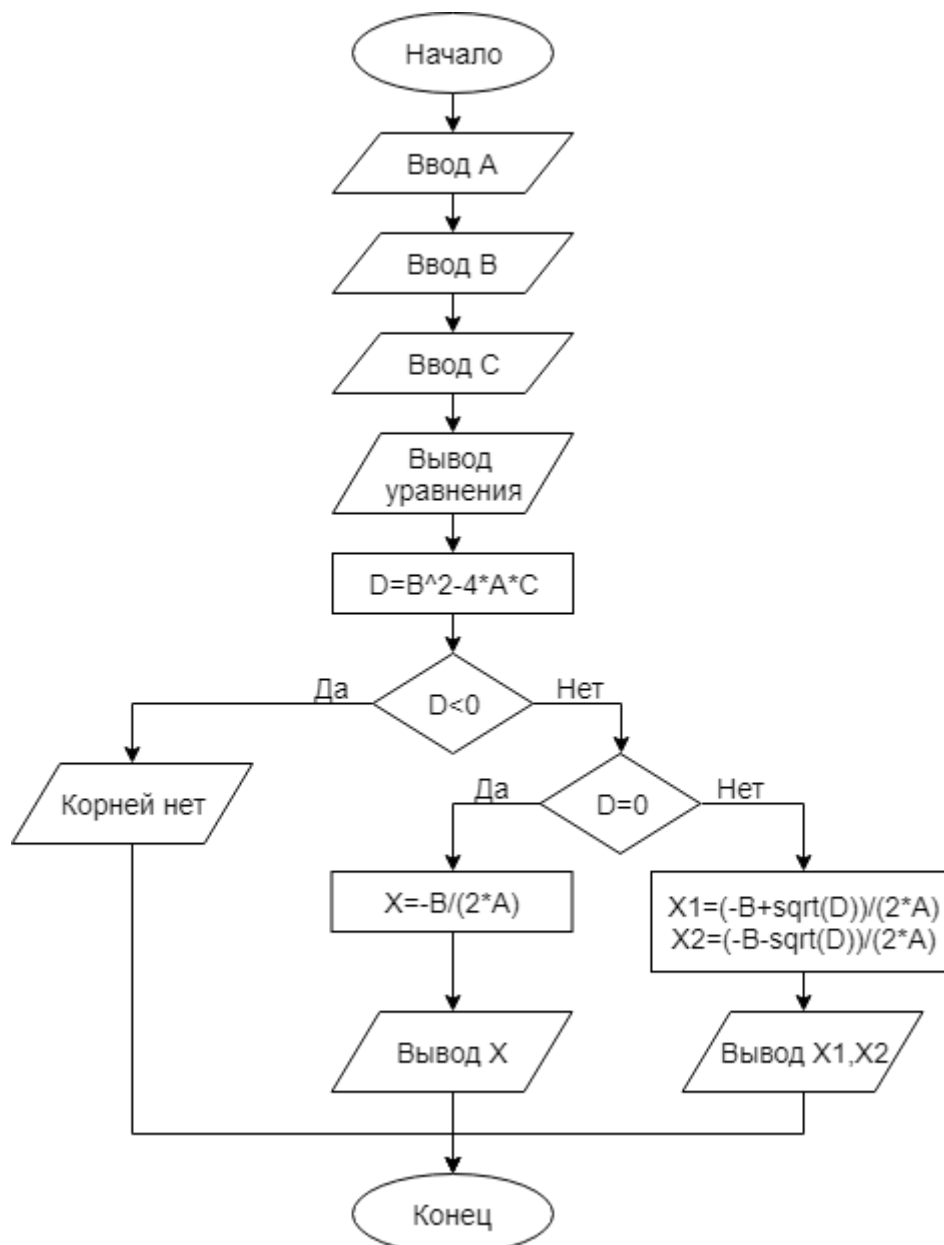


Рис. 1. Блок-схема алгоритма

Далее рассматривается другой способ графической записи алгоритма – с помощью диаграммы Насси-Шнейдермана.

Диаграмма Насси-Шнейдермана (структурограммы)

Диаграмма Насси-Шнейдермана – это графический способ представления структурированных алгоритмов и программ, разработанный в 1972 году американскими аспирантами Беном Шнейдерманом и Айзеком Насси.

Диаграммы Насси-Шнейдермана имеют следующие преимущества перед блок-схемами при разработке структурированных алгоритмов и программ.

- Запись является более компактной (в первую очередь из-за отсутствия стрелок между элементами).
- Гарантированное соблюдение принципов структурного программирования (при использовании блок-схем можно случайно получить неструктурированный алгоритм, если быть невнимательным).
- Диаграмму Насси-Шнейдермана более удобно использовать для пошаговой детализации задачи, так как они тоже строятся по принципу пошаговой детализации – изначально диаграмма представляет собой один прямоугольник (исходная задача), затем в нём рисуется некоторая структура управления, в которой имеется несколько прямоугольников (подзадач исходной задачи), и далее с каждым прямоугольником (подзадачей) может быть проделана та же операция.

При записи структуры последовательного выполнения элементы изображаются вертикально один за другим. При этом все элементы последовательности должны иметь одинаковую ширину – вследствие этого вся последовательность тоже имеет прямоугольную форму (рис.2)[3].

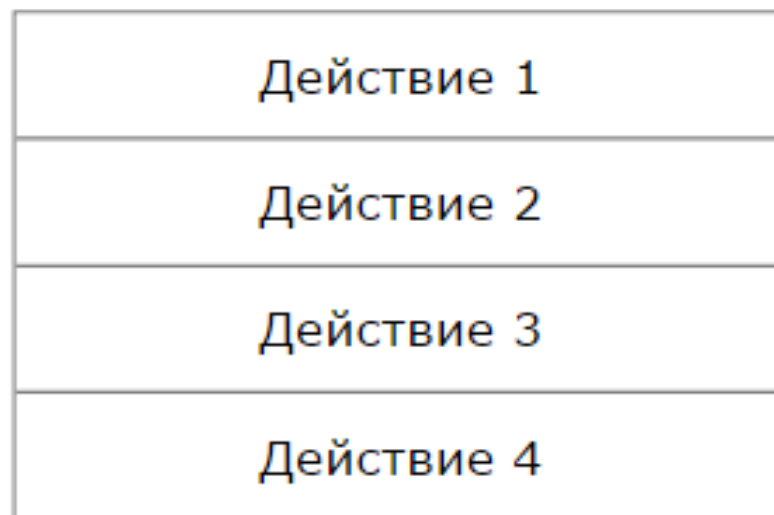


Рис.2. Последовательность

Структура простого ветвления изображается в виде прямоугольника, разделенного вертикальной чертой на две части (рис.3) [3]. В верхней части располагается заголовок ветвления, а в нижней – две ветки, разделенные вертикальной чертой. В заголовке рисуются две линии, ведущие от верхних углов к началу

линии, разделяющей ветви. В получившемся вверху треугольнике записывается условие ветвления, в двух нижних треугольниках над ветвями подписываются значения условия, соответствующие этим ветвям, например «истина» и «ложь», или «да» и «нет».

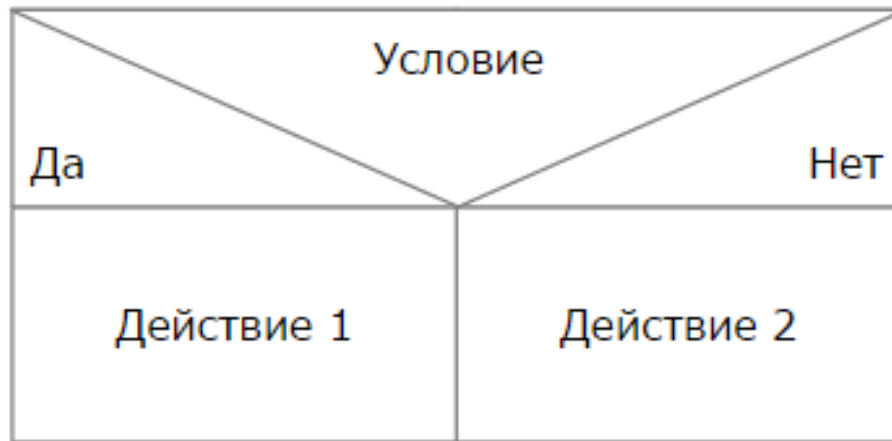


Рис.3. Простое ветвление

Структура многовариантного выбора похожа на структуру простого ветвления, только основная ветвь и треугольник над ней делятся на много частей вертикальными линиями (рис. 4) [3]. В верхнем треугольнике записывается выражение-переключатель, над ветвями записываются соответствующие значения переключателя.



Рис.4. Многовариантный выбор

Повтор с предусловием

Структура повтора с условием в начале (с предусловием) изображается как прямоугольник, внутри которого в правой нижней части нарисован ещё один прямоугольник (рис.5) [3]. Над

внутренним прямоугольником записывается заголовок цикла, а внутри него – тело цикла.

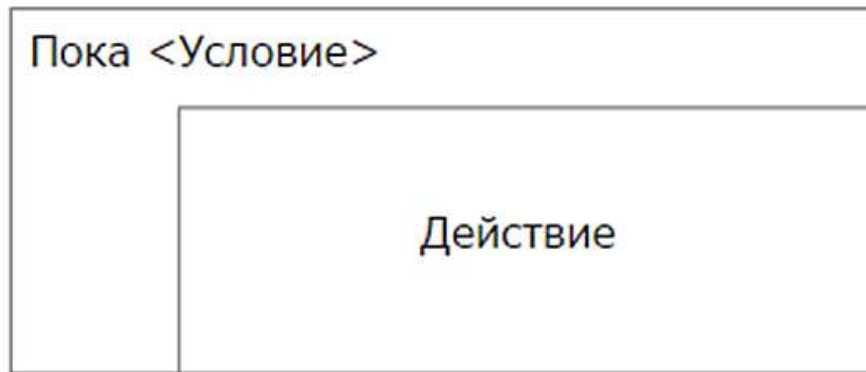


Рис. 5. Повтор с предусловием

Повтор с постусловием

Отличается от цикла с предусловием только тем, что внутренний прямоугольник рисуется в правой верхней части внешнего, а заголовок записывается снизу (рис. 6) [3].

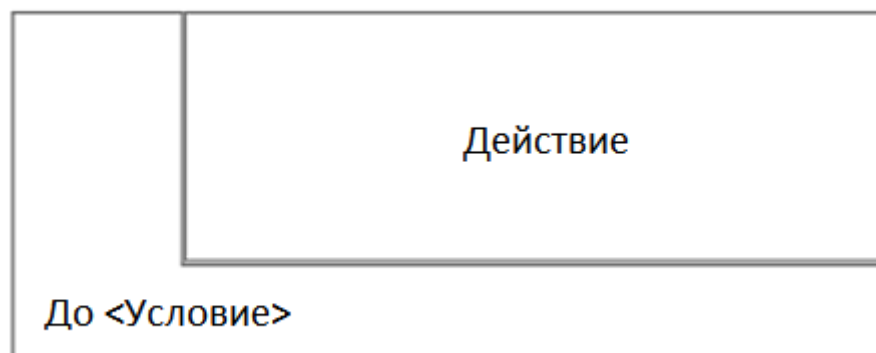


Рис. 6. Повтор с постусловием

Повтор со счётчиком

Внутренний прямоугольник рисуется в правой части и не касается верха и низа внешнего прямоугольника (рис. 7). Условие цикла записывается сверху.

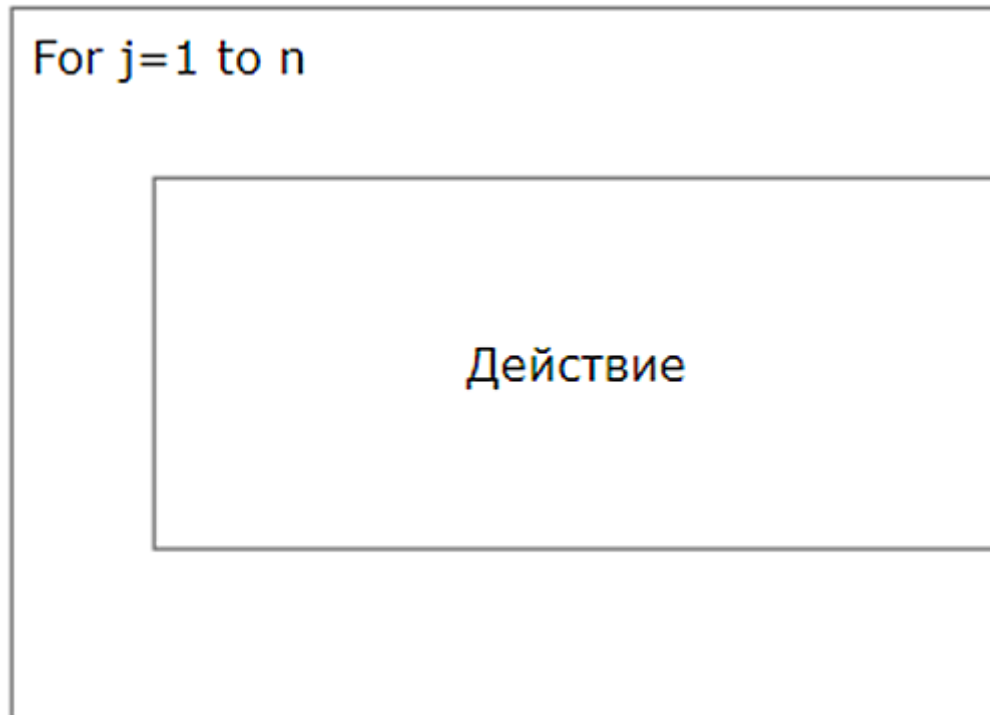


Рис. 7. Повтор со счётчиком

Пример записи цикла со счетчиком представлен на рис. 8.

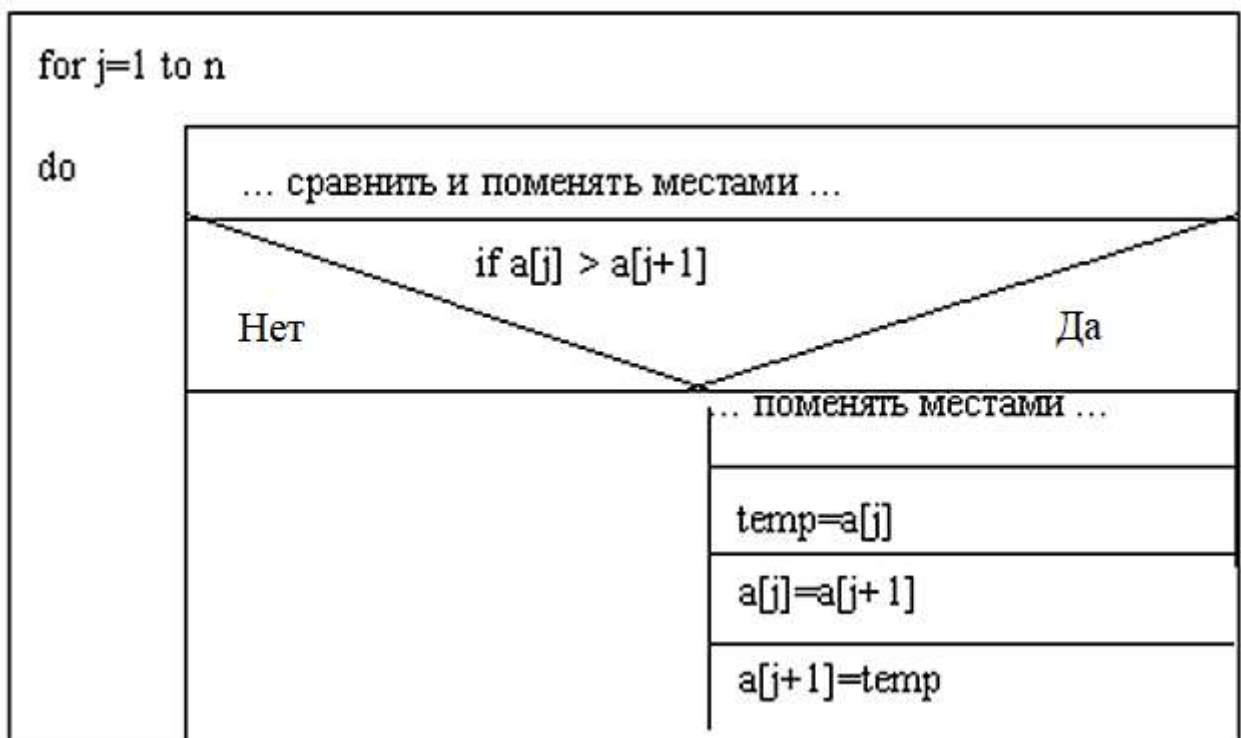


Рис. 8. Пример записи цикла со счётчиком

3. Общая часть лабораторной работы

В общей части лабораторной работы в рассмотренных ниже задачах выполнить описанные действия и при защите лабораторной работы предъявить преподавателю результат этих действий.

Задача 1. Разработка блок-схемы

Задача. Разработать блок-схему алгоритма вычисления факториала натурального числа ($n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$).

Решение. В алгоритме необходимо отразить ввод числа n , проверку на ввод отрицательного числа, проверку на равенство n нулю (в этом случае факториал равен единице) и последовательно перемножить все числа от одного до n (с помощью цикла со счетчиком), занося результат в переменную f , после чего ее следует вывести на экран.

Построение блок-схемы осуществим с помощью онлайн-сервиса <https://www.draw.io>. При переходе на страницу сервиса появляется диалог выбора дальнейшего режима работы – создание новой диаграммы (Create New Diagram) или открытие существующей диаграммы (Open Existing Diagram).

Выберем пункт создания новой диаграммы. Появится окно создания нового проекта (рис. 1).

В текстовом поле вверху необходимо ввести имя диаграммы. В списке слева представлены шаблоны диаграмм, из которых выберем пункт «Basic», позволяющий создать пустую блок-схему. Для продолжения работы требуется нажать на кнопку «Create» внизу окна.

После выбора параметров новой диаграммы появится рабочая область сервиса (рис. 2). Слева располагаются элементы блок-схемы, по центру – область построения диаграммы. Справа располагается окно настройки рабочего листа (размер страницы, настройка сетки и фоновое изображение).

Перейдем к созданию диаграммы. Перенесем на область построения диаграммы элементы «Начало алгоритма» и «Ввод данных» (рис. 3).

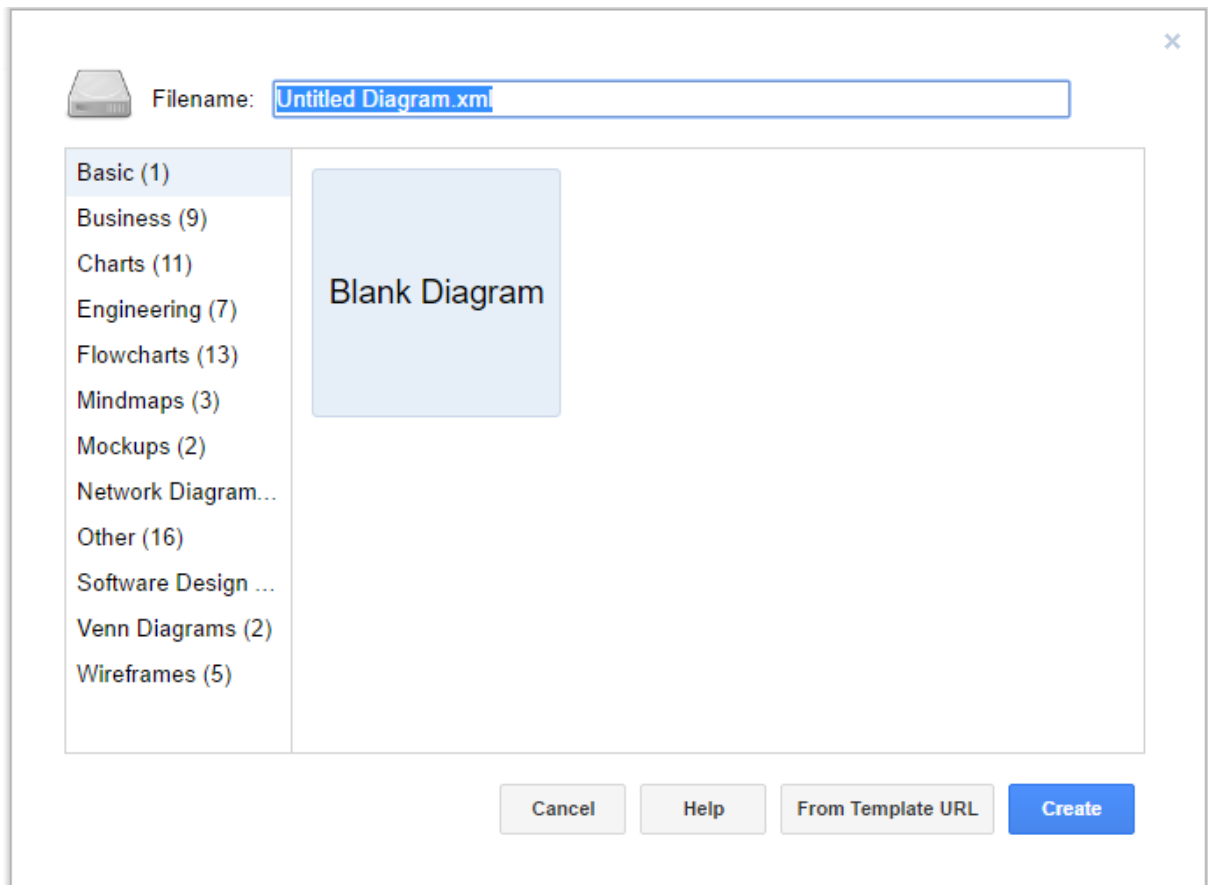


Рис.1. Окно создания нового проекта

Двойной щелчок по элементу позволяет перейти в режим ввода и редактирования текста.

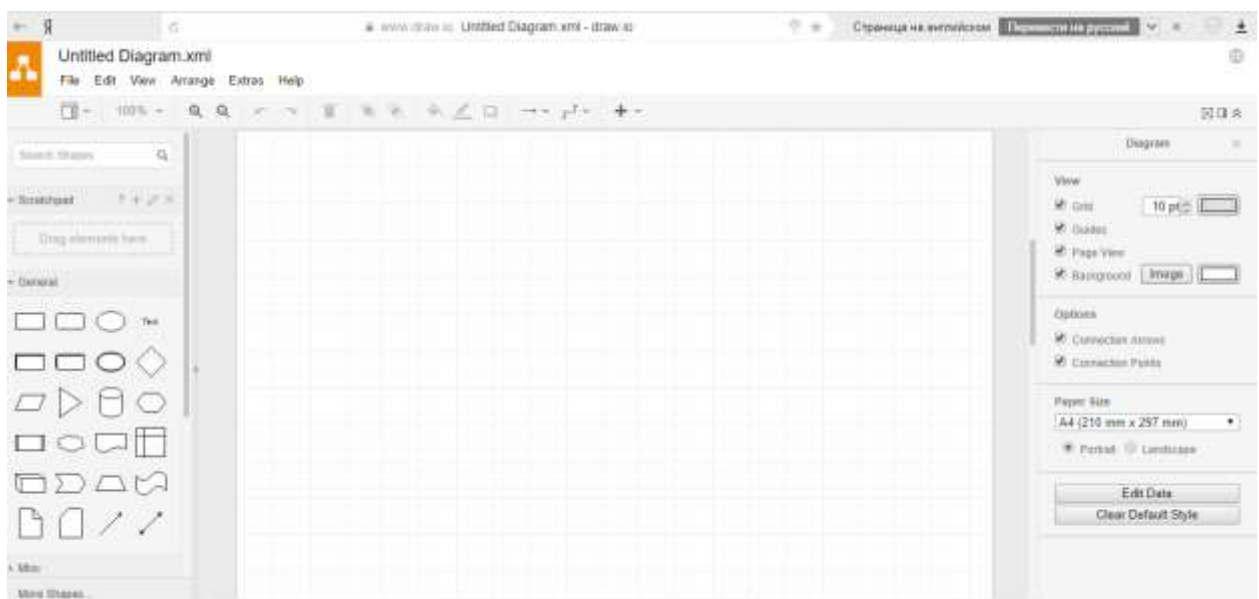


Рис. 2. Рабочая область сервиса

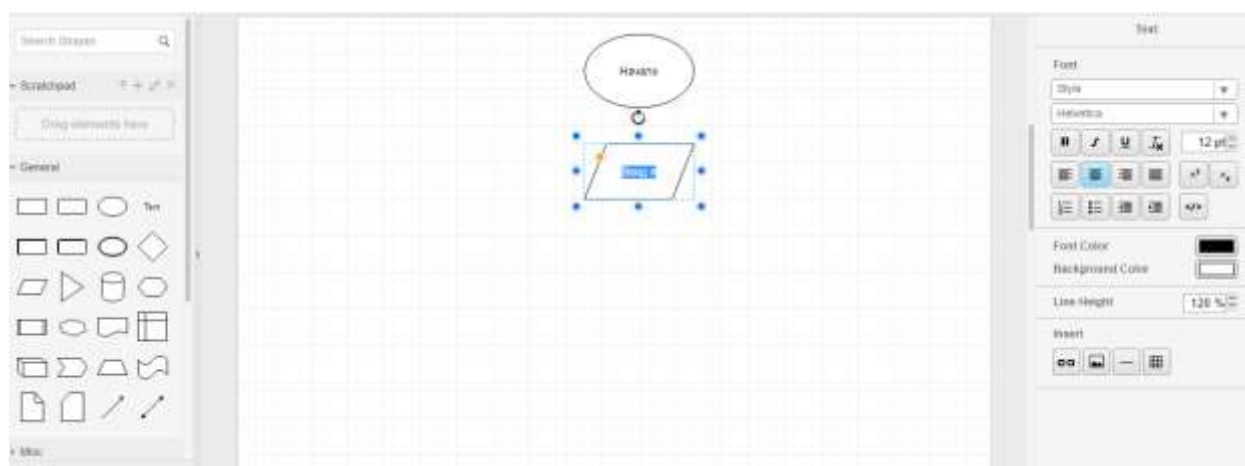


Рис.3. Создание диаграммы

Чтобы создать соединительные линии, достаточно однократно нажать на элемент блок-схемы, нажать на появившуюся стрелку на элементе и протащить ее до элемента, к которому должна быть направлена стрелка. Для размещения на стрелке надписи дважды кликнем на ней мышью. Аналогично разместим на рабочей области блок условия и добавим на блок-схему подписи.

Добавим на блок-схему элементы, отвечающие за действия при вводе отрицательного числа и проверку условия равенства числа нулю. Теперь добавим на блок-схему оставшиеся элементы в соответствии с алгоритмом вычисления факториала. Счетчик на блок-схеме изображен в виде шестиугольника. Кроме того, разместим элемент вывода результата и элемент конца алгоритма (рис. 4).

Готовую блок-схему необходимо сохранить в виде изображения в формате PNG. В горизонтальном меню выберем пункт «File» – «Export As» – «PNG». В открывшемся меню можно сохранить изображение в облачное хранилище, выбрав соответствующий пункт, либо сохранить на компьютер, нажав на кнопку «Download».

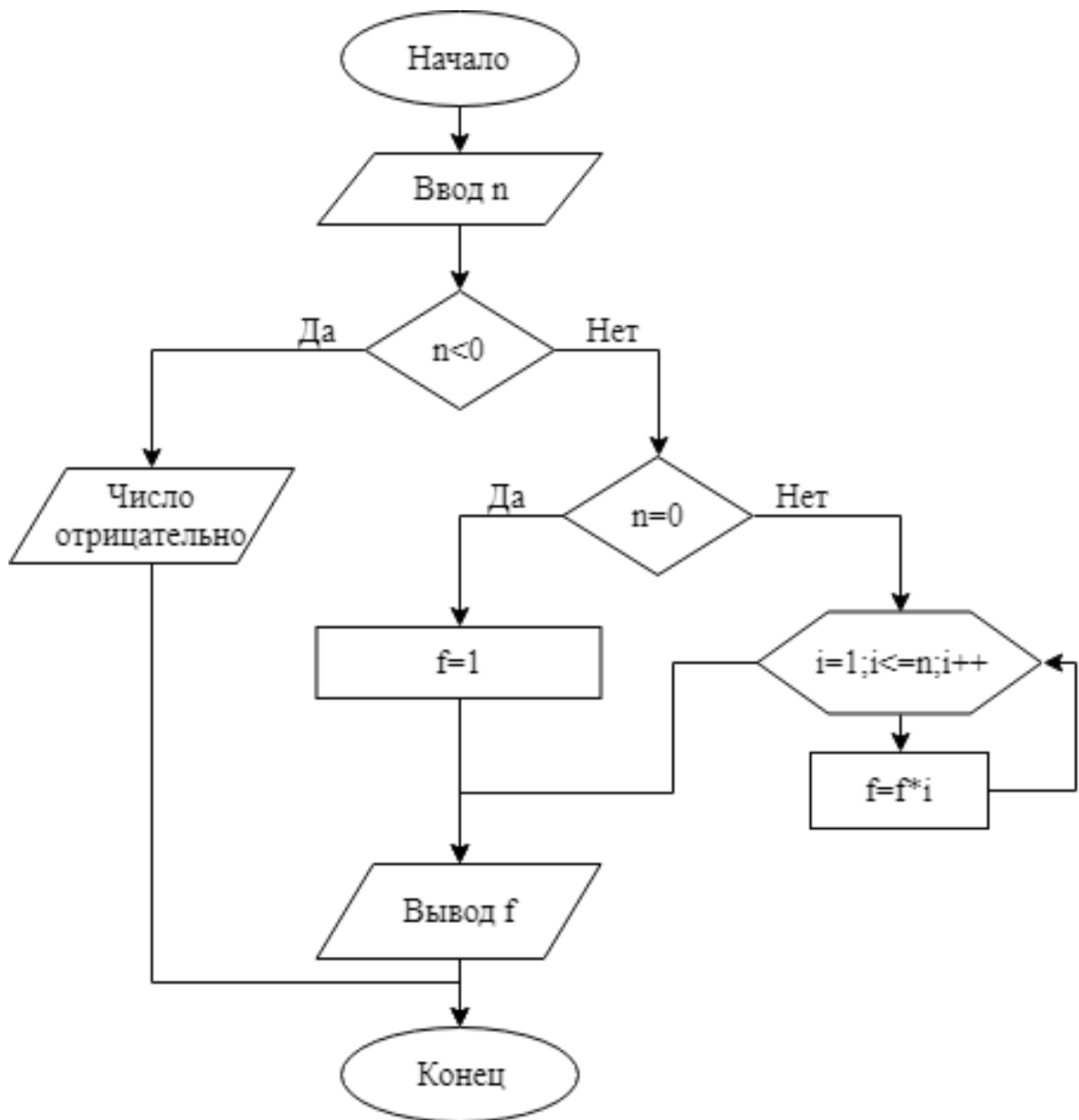


Рис. 4. Блок-схема алгоритма

Задача 2. Разработка структурограммы

Задача. Разработать структурограмму алгоритма вычисления факториала натурального числа ($n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$).

Решение. Для разработки структурограммы воспользуемся онлайн-сервисом <https://cloud.smartdraw.com> (сервис может потребовать регистрацию).

В окне браузера при загрузке страницы сервиса слева расположен список всех доступных для построения диаграмм. Выберем пункт «Software Design», в раскрывшемся списке выберем подпункт «Other Software Diagram». В центральной части окна выберем пункт «Nassi-Schneiderman» (рис. 5).

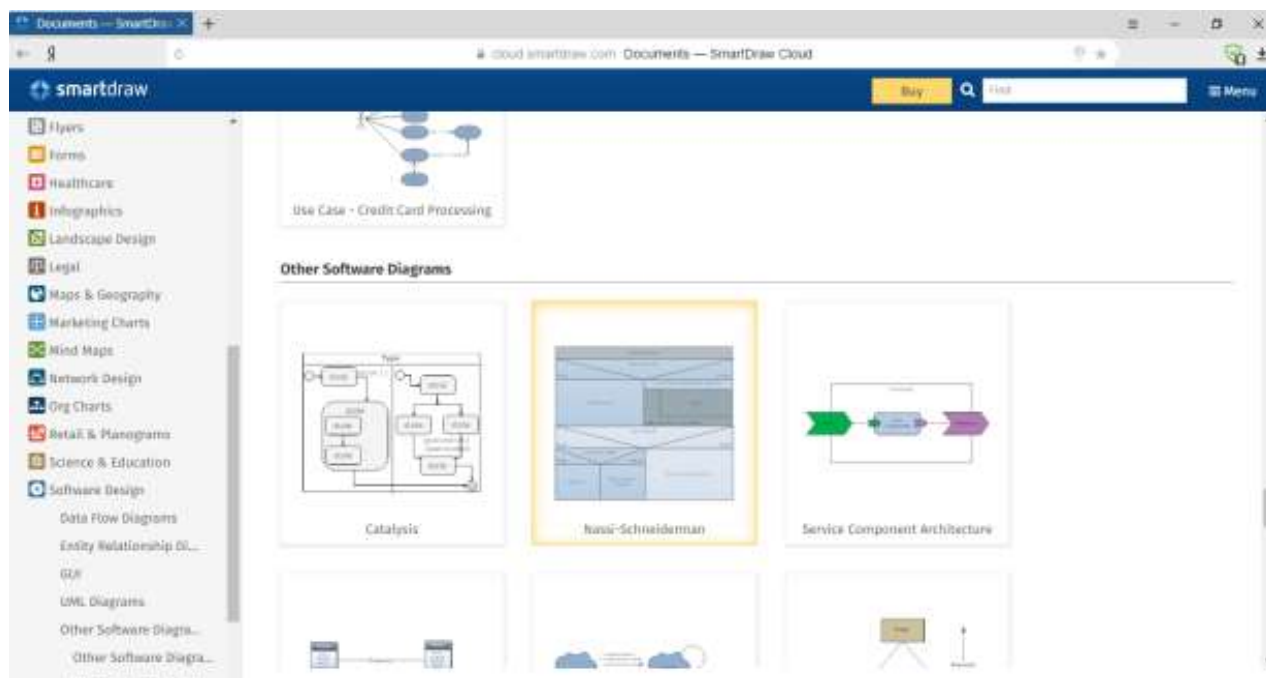


Рис. 5. Выбор типа диаграммы

В открывшемся новом окне браузера располагается редактор для построения структурограмм (рис. 6). Слева расположены основные элементы диаграммы, в центре окна размещается рабочая область, вверху – меню параметров текста и режимов работы с объектами.

Для начала добавим на диаграмму элементы ввода n и проверки условия отрицательности введенного значения (рис. 7).

Чтобы добавить на рабочую область элемент, выберем его из списка слева и щелкнем по рабочей области или перетащим элемент из списка в подходящее место. Чтобы заполнить элемент текстом, дважды кликнем по нему мышью. Стоит обратить внимание на то, что встроенная система проверки орфографии не поддерживает русский язык. Присутствует возможность изменения размера и перемещения элементов.

Внутри второго блока добавим проверку равенства n нулю и цикл со счетчиком (рис. 8).

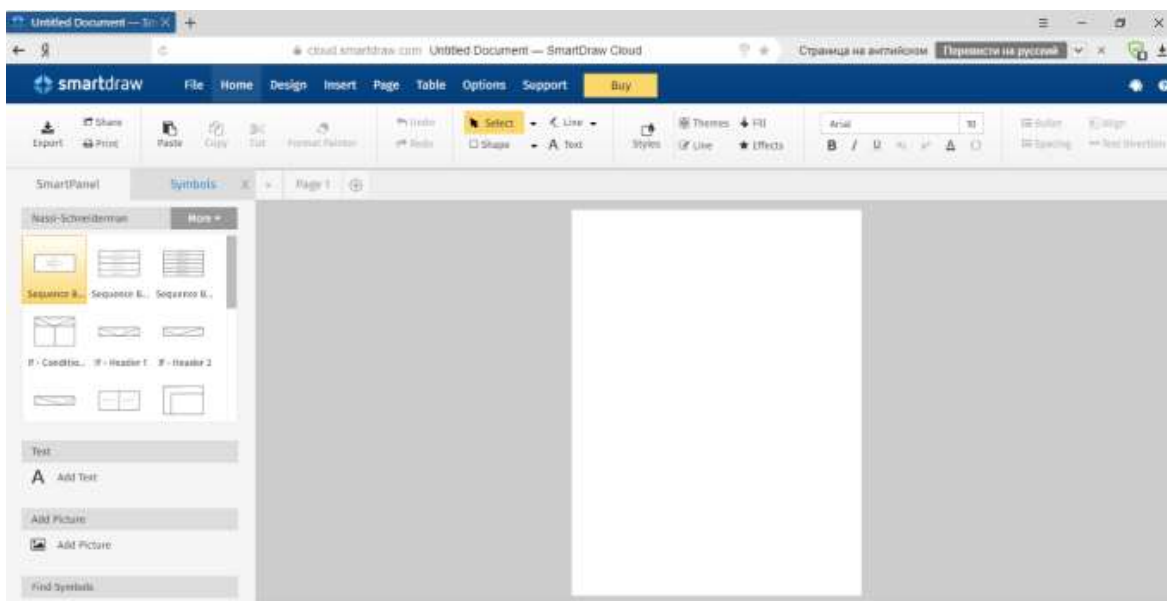


Рис. 6. Редактор для построения структурограмм



Рис. 7. Проверка условия отрицательности

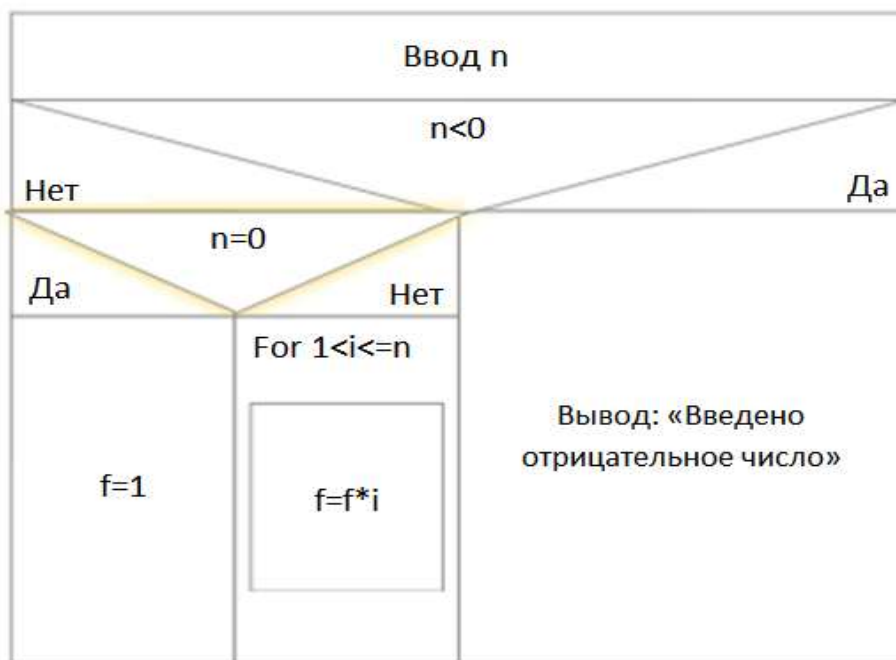


Рис. 8. Построение структурограммы. Шаг 2

Добавим на диаграмму блок вывода результата иотрегулируем длину блока, соответствующего выводу сообщения об отрицательном значении n (рис.9).

Готовую диаграмму экспортируем в виде изображения с помощью кнопки «Export» в разделе меню «Home», выбрав режим экспортирования «PNG». Отметим, что при сохранении в бесплатной версии онлайн-сервиса экспортированный файл содержит подложку с логотипом сайта. Сохранить созданную диаграмму без посторонних надписей позволит стандартная программа «Ножницы» или другие средства для экранных снимков.

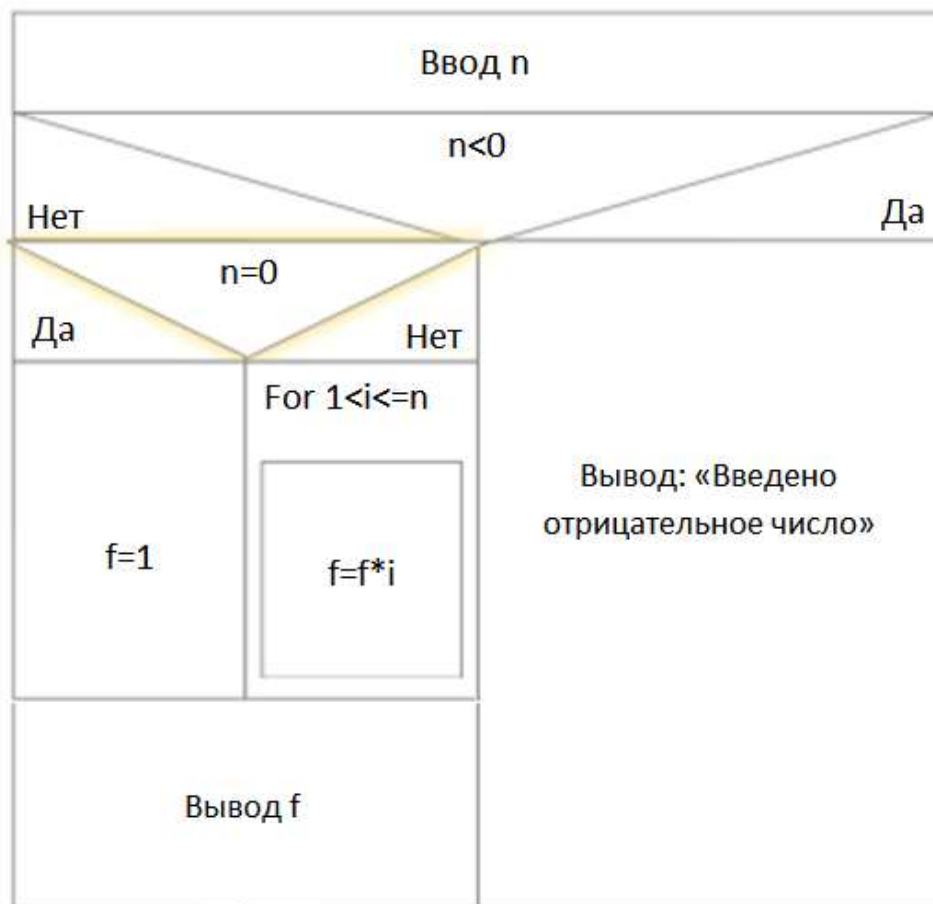


Рис. 9. Структурограмма алгоритма

4. Индивидуальная часть лабораторной работы

Каждый студент должен разработать алгоритм некоторой процедуры (в соответствии с вариантом задачи для самостоятельного решения). Сначала необходимо изобразить алгоритм в виде блок-схемы, затем в виде структурограммы.

При защите лабораторной работы разработанный и записанный двумя способами алгоритм необходимо предъявить преподавателю в электронном виде в сервисе разработки алгоритмов.

5. Задачи для самостоятельного решения

1. Разработать алгоритм вывода чисел, меньших заданного числа P , из последовательности квадратов натуральных чисел (1, 4, 9, 25 и т.д.).

2. Разработать алгоритм вывода чисел, меньших заданного числа P , из последовательности $\sqrt{2}$, $\sqrt{3}$, $\sqrt{4}$ и т.д.

3. Даны два целых числа A и B ($A < B$). Разработать алгоритм вывода всех целых чисел, расположенных между этими числами (не включая сами эти числа), в порядке их возрастания.

4. Даны два целых числа A и B ($A < B$). Разработать алгоритм вывода всех целых чисел, расположенных между этими числами (не включая сами эти числа), в порядке их убывания.

5. Дано целое число N ($N > 1$). Разработать алгоритм вывода наименьшего целого K , при котором выполняется неравенство $3K > N$ и самого значения $3K$.

6. Дано целое число N ($N > 1$). Разработать алгоритм вывода наибольшего целого K , при котором выполняется неравенство $3K < N$ и самого значения $3K$.

7. Дано число n . Разработать алгоритм поиска и вывода первого натурального числа, квадрат которого больше n .

8. Даны числа n и m . Разработать алгоритм вывода минимального числа, большего n , которое делится на m без остатка.

9. Даны числа n и m ($n > m$). Разработать алгоритм вывода максимального из натуральных чисел, не превышающих n , которое делится на m без остатка.

10. Начав тренировки, лыжник в первый день пробежал n км. Каждый следующий день он увеличивал пробег на 10% от пробега предыдущего дня. Разработать алгоритм определения, в какой день он пробежит больше m км.

11. Дано число n . Разработать алгоритм вывода всех натуральных чисел, кратных одиннадцати, меньше n .

12. Дано число n . Разработать алгоритм вывода всех натуральных чисел, кратных n и меньших 100.

13. Разработать алгоритм вычисления значения выражения y для значений x , равных $1, 2, \dots, 20$. $y = 2t^2 + 2t + 2, t = 1 + x$.

14. Разработать алгоритм вычисления значения выражения z для значений x , равных $2, 4, \dots, 20$. $z = 8f^3 - f, f = 2x$.

15. Дано целое число $m > 1$. Разработать алгоритм вычисления наименьшего целого k , при котором $4^k > m$.

16. Дано натуральное число n . Разработать алгоритм вычисления и вывода произведения $P = \left(1 + \frac{1}{1^2}\right) \cdot \left(1 + \frac{1}{2^2}\right) \cdot \dots \cdot \left(1 + \frac{1}{n^2}\right)$.

17. Дано целое число n . Разработать алгоритм вычисления и вывода наименьшего числа вида 2^r , превосходящего n (r – натуральное число).

18. Дано натуральное число n и действительное число a . Разработать алгоритм вычисления и вывода произведения $P = a \cdot (a + 1) \cdot (a + 2) \cdot \dots \cdot (a + n - 1)$.

19. Дано натуральное число n . Разработать алгоритм вычисления и вывода суммы $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$

20. Дано натуральное число n . Разработать алгоритм вычисления и вывода суммы n первых слагаемых $S = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} \dots$

21. Пусть последовательность чисел образована по следующему закону: $a_1 = 1$; $a_k = k * a_{k-1} + 1/k$; $k = 1, 2, \dots$ Дано целое число n . Разработать алгоритм вычисления и вывода a_n .

22. Дано вещественное число a . Разработать алгоритм вывода всех натуральных чисел n , для которых выполняется условие $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} < a$.

23. Дано вещественное число a . Разработать алгоритм вычисления и вывода всех натуральных чисел n , для которых выполняется условие $\frac{1}{2} + \frac{3}{4} + \dots + \frac{2n-1}{2n} < a$.

24. Дано натуральное число n . Разработать алгоритм вычисления и вывода произведения $\prod_{i=1}^n \left(1 - \frac{1}{i!}\right)^2$

25. Дано натуральное число n . Разработать алгоритм вычисления и вывода суммы $\sum_{k=1}^n \frac{k!}{(2k+1)k}$.

6. Контрольные вопросы

1. Что такое алгоритм?
2. Что такое исполнитель алгоритма?
3. Каковы основные свойства алгоритма?
4. В чём заключается свойство дискретности алгоритма?
5. В чём заключается свойство детерминированности алгоритма?
6. В чём заключается свойство понятности алгоритма?
7. В чём заключается свойство конечности алгоритма.
8. В чём заключается свойство массовости алгоритма?
9. В чём заключается свойство результативности алгоритма?
10. Каковы основные виды алгоритмов?
11. В чём состоит суть линейного алгоритма?
12. В чём состоит смысл разветвляющегося алгоритма?
13. Какова основная идея циклического алгоритма?
14. Какие бывают виды циклических алгоритмов?
15. В чём состоит суть цикла со счетчиком?
16. В чём состоит смысл цикла с предусловием?
17. Какова суть цикла с постусловием.
18. Какие бывают формы записи алгоритма?
19. Что такое блок-схема?
20. Какие бывают типы блоков?
21. Каким образом задается последовательность действий в блок-схемах?
22. Какие блоки используются при реализации линейного, разветвляющегося, циклического алгоритмов?
23. Можно ли составить разные варианты блок-схем для одной и той же задачи?
24. Каковы основные обозначения в блок-схемах?
25. Какие преимущества перед блок-схемами имеют диаграммы Насси-Шнейдермана?
26. Какие существуют структурные элементы в диаграммах Насси-Шнейдермана?
27. Каковы основные обозначения в структурограммах?

Лабораторная работа №2. Основы работы в интегрированной среде MS Visual Studio

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки создания простейшего приложения на языке Си в среде Microsoft Visual Studio и ознакомиться с простейшими инструментами отладки.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Microsoft Visual Studio – это набор инструментов разработки, основанных на использовании компонентов и других технологий для создания мощных, производительных приложений.

Программа – это реализация алгоритма для выполнения задачи компьютером (ЭВМ). В виде программы формулируется алгоритм на языке, понятном компьютеру. Таким языком является язык программирования.

Компиляция – преобразование программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком к машинному коду.

Компилятор – программа или техническое средство, выполняющее компиляцию.

Программа на языке Си состоит из функций, описаний и директив препроцессора. **Функция** – это законченный фрагмент кода, к которому можно обратиться по имени (вызвать функцию). Одна (и только одна) функция программы должна иметь имя `main()`. Выполнение программы начинается с первого оператора этой функции [2].

Описания представляют собой **определения** и **объявления** элементов программы: переменных, функций, классов, типов и т.д. Любой элемент программы должен быть определен только один раз. Это правило в стандарте называется правилом одного определения.

Программа на языке Си имеет определенную структуру. Существует определенная последовательность заранее определенных строк кода, которая приведена в таблице 1.

Структура программы на языке Си

#include <назв_заголов_файла>	Подключение заголовочных файлов
int main() {	Главная функция программы. Именно она начинает выполняться, когда запускается программа. Обязательная часть
Тело_функции_main	В теле функции main() записываются действия и операции, предусмотренные алгоритмом. Обязательная часть
return 0; }	Конец программы. Самый последний оператор. Обязательная часть

Проект – это набор взаимосвязанных исходных файлов, компиляция и компоновка которых позволяет создать исполняемую Windows-программу или библиотеку.

Исходные файлы проекта хранятся в отдельном каталоге, кроме того, проект часто зависит от внешних файлов, таких как подключаемых (include) и библиотечных файлов. В проекте Visual C++ взаимозависимости между его отдельными компонентами описаны в текстовом файле проекта с расширением VCPROJ. Специальный текстовый файл решения с расширением SLN содержит список всех проектов решения [2].

Решение (Solution) – набор проектов, объединенных вместе, которые решают одну задачу.

Любая программа работает с данными. В простейшем случае данные хранят в переменных.

Переменная – это именованная область памяти компьютера, у переменной есть тип, имя и значение. Имя служит для обращения к области памяти, в которой хранится значение, тип определяет диапазон значений и допустимые операции. Во время выполнения программы значение переменной можно изменять.

Перед использованием любая переменная должна быть определена. Пример описания целой переменной с именем *a* и вещественной переменной с именем *b*:

```
int a;
float b;
```

В общем случае описание переменной может задавать не только ее имя и тип, но и ее начальное значение:

```
int a = 5;
```

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, зачастую приходится узнавать текущие значения переменных и выяснять, по какому пути выполнялась программа.

3. Общая часть лабораторной работы

Запустите среду Microsoft Visual Studio. В появившемся окне приветствия можно перейти к открытию существующего проекта (Open Project), созданию нового (New Project), а также открыть проекты, работа с которыми происходила недавно (Recent).

Выберите пункт создания нового проекта. В левой части окна выберите пункт Visual C++, после чего в центральной части окна выберите режим создания консольного приложения Win32 (Windows Console Application) (рис. 1). В строке Имя (Name) внизу окна укажите имя проекта (для этой лабораторной работы, например lab2), а также в строке Расположение (Location) укажите путь к папке, в которой будут размещаться файлы проекта.

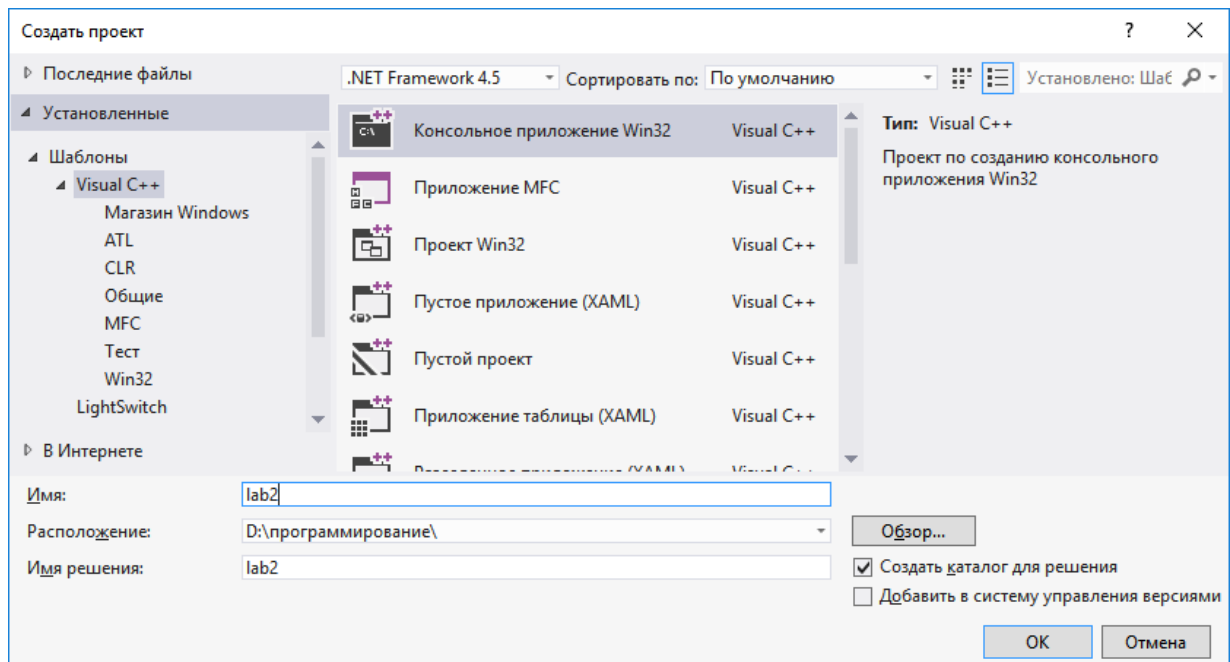


Рис. 1. Создание нового проекта

При нажатии на кнопку «ОК» откроется мастер создания нового проекта. Требуется нажать на кнопку Далее (Next). В открывшемся диалоговом окне сделать активным флаг «Пустой проект» («Empty

project») и нажать кнопку Готово (Finish). В этом случае в созданный проект не будут добавлены никакие файлы, и их необходимо будет создать самостоятельно.

После закрытия мастера откроется окно с пустым проектом. Слева располагается окно обозревателя решений. В нем представлены все файлы проекта, разделенные по категориям (рис.2).

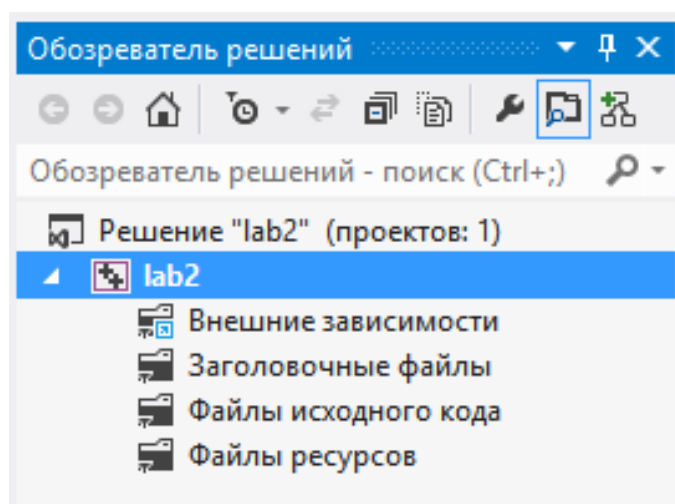


Рис. 2. Окно обозревателя решений

С понятиями заголовочных файлов и ресурсов проекта ознакомимся позднее. На этом этапе лабораторной работы добавим файл с кодом в созданный проект. Для этого нажмем правой кнопкой мыши на имени проекта, в появившемся контекстном меню выберем пункт Добавить (Add), затем нажмем пункт Создать элемент (New Item) (рис. 3).

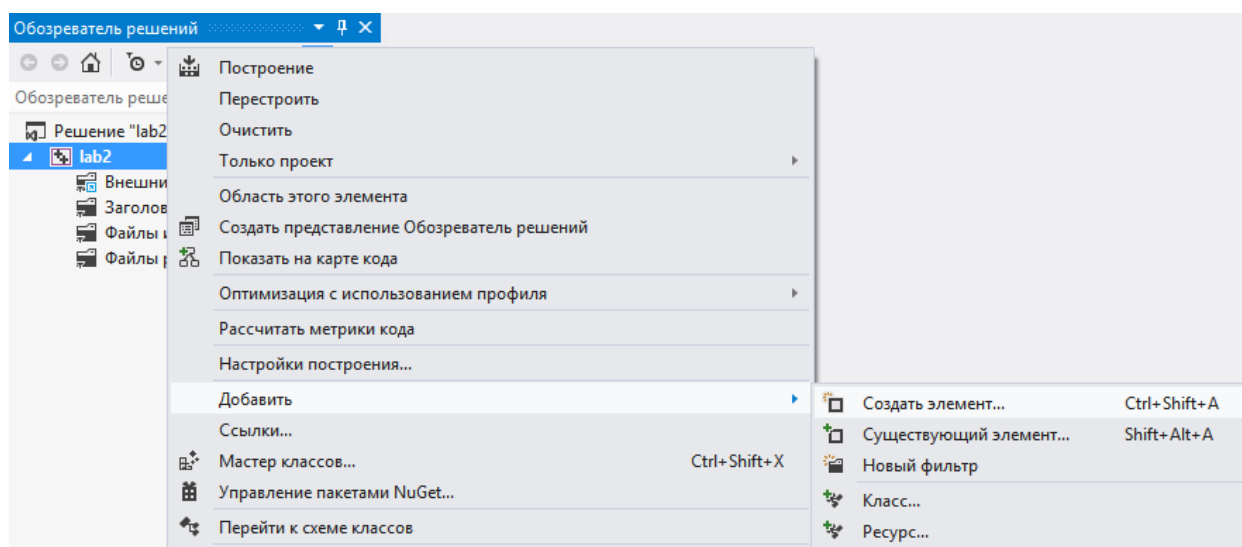


Рис. 3. Добавление нового элемента в проект

В открывшемся диалоговом окне требуется выбрать Файл C++ (C++ File) и указать имя файла (lab2.cpp), директорию хранения файла менять не рекомендуется, по умолчанию файл создается в папке проекта (рис. 4).

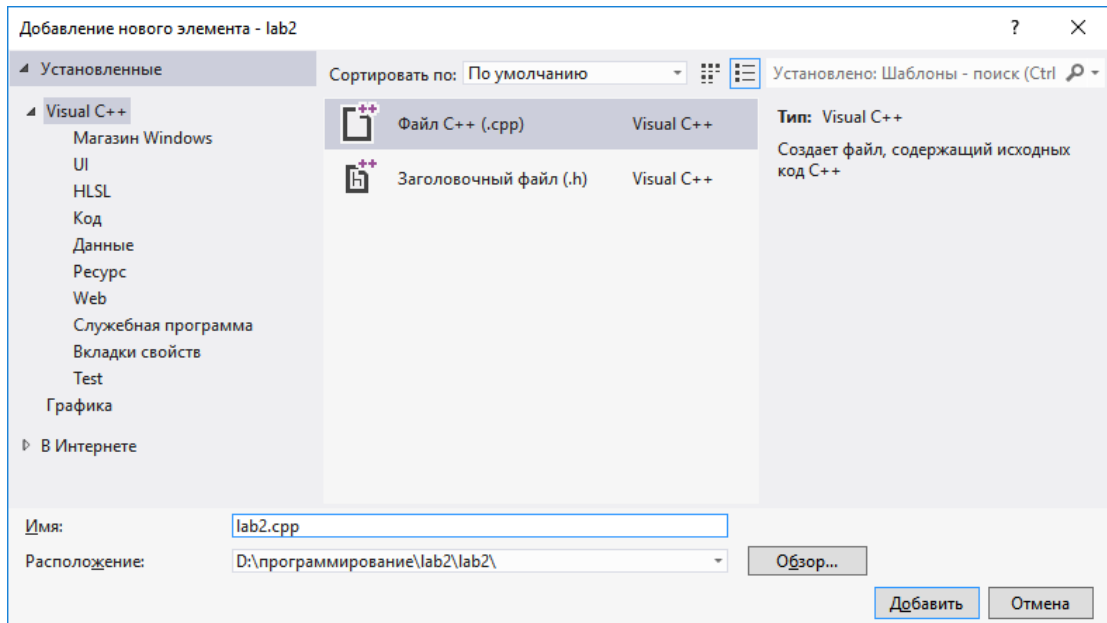


Рис. 4. Добавление в проект файла .cpp

Созданный файл отобразится в обозревателе решений. Содержимое файла доступно до редактирования в центральном окне.

В центральной части окна располагается рабочая область, в которую заносится текст программы. Простейшая программа на языке Си должна содержать функцию `main()`, которая является точкой входа в программу. Добавим в файл «lab2.cpp» следующий код:

```
int main()
{
    return 0;
}
```

Произведем отладку добавленного кода, выбрав в пункте меню «Отладка» (Debug) команду «Начать отладку» (Start Debugging) или нажав клавишу F5. В результате этого на экране на короткое время появляется консольное окно. Это означает, что процесс отладки прошел успешно. Никаких действий программы не видно, потому что функция `main()` пока не содержит никаких команд.

Теперь откроем папку, которую указали как место хранения файлов проекта. В ней обратим внимание на файл решения MVS Solution. В папке с именем проекта можно найти сам файл проекта и

файл с кодом программы. В папке «Debug» располагается скомпилированный файл консольного приложения (рис. 5). Это и есть готовая программа.

Локальный диск (D:) > программирование > lab2 > Debug			
	Имя	Тип	Размер
	lab2	Приложение	30 КБ
	lab2	Incremental Linker File	215 КБ
	lab2	Program Debug Database	411 КБ

Рис. 5. Папка «Debug» проекта

Вернемся в Microsoft Visual Studio и добавим в функцию main() следующий код:

```
int a = 5;
a = a + 30;
int b = a * 2;
b = a + b
```

Обратим внимание на то, что код должен располагаться выше команды return, которая является точкой выхода из функции main(), а значит, и точкой завершения программы.

Заметим, что редактор подчеркнул красным фрагмент кода (рис.6). Это означает, что текст программы содержит синтаксические ошибки.

```
int main()
{
    int a = 5;
    a = a + 30;
    int b = a * 2;
    b = a + b

    return 0;
}
```

Рис. 6. Выделение синтаксических ошибок в коде

Произведем отладку программы, нажав клавишу F5. Так как текст программы содержит синтаксические ошибки, отладчик прекратит компиляцию, выдав соответствующее сообщение и предложив запустить последнюю рабочую версию программы. После нажатия кнопки «Нет» в нижней части окна отобразится список ошибок в тексте программы (рис. 7). В окне указана суть ошибки и строка, в которой она возникла. Двойной щелчок по ошибке позволит перевести курсор в окне с текстом программы на строку, содержащую ошибку. В этом случае среда разработки указывает на отсутствие точки с запятой перед командой `return`, это означает, что «;» пропущена выше, в строке «`b = a + b`». Добавим в конце строки недостающий символ и нажмем клавишу F5. Теперь процесс отладки будет произведен успешно.

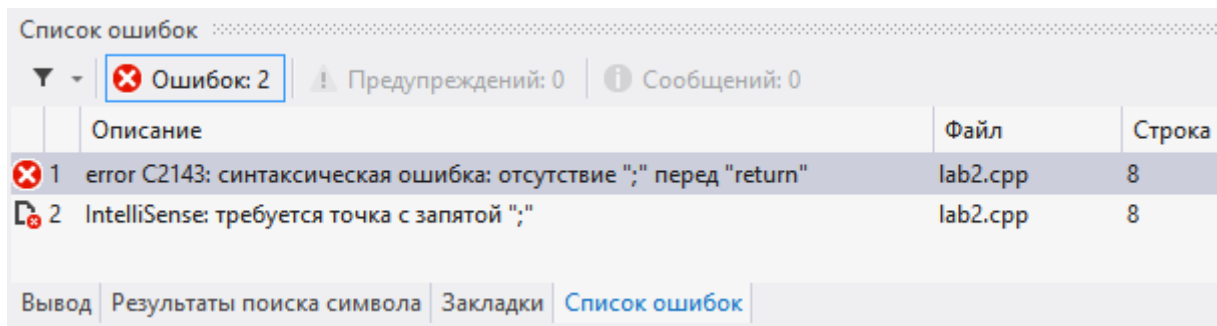


Рис. 7. Окно ошибок

Так как в программе не происходит ввод данных с клавиатуры и вывод результатов на экран, после запуска приложения снова увидим пустое консольное окно и только на короткое время.

Перейти к списку ошибок можно выбрав соответствующую вкладку в нижней части экрана. Стоит отметить, что эта вкладка содержит не только список ошибок, но и список предупреждений, которые могут повлиять на правильность работы программы, но при этом не делают невозможной компиляцию кода.

Зачастую программист допускает множество синтаксических ошибок (в основном по причине невнимательности), которые легко отслеживаются в редакторе кода и обнаруживаются при попытке компиляции. Однако намного сложнее выявлять логические ошибки, возникающие при написании алгоритма решения той или иной задачи. С этой целью Microsoft Visual Studio позволяет контролировать значения переменных во время выполнения программы.

Одним из удобных инструментов Microsoft Visual Studio является использование точек останова – намеренного прерывания выполнения программы. В режиме отладки программа будет выполняться до такой точки, после чего ее выполнение будет приостановлено и в дальнейшем может быть возобновлено повторным нажатием клавиши F5. Чтобы поставить точку останова, следует щелчком левой кнопкой мыши по серой области слева от кода программы (рис. 8).

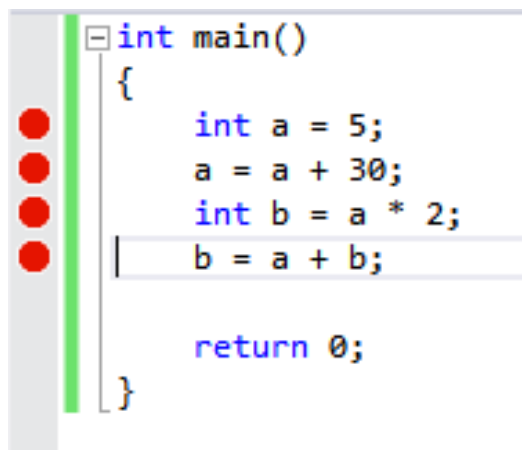


Рис. 8. Точки останова

В режиме отладки на строку, код которой будет выполнен следующим шагом, указывает желтая стрелка. При наведении на переменную появляется подсказка с ее текущим значением.

Таким образом, имеется возможность контролировать значения переменных на каждом шаге выполнения программы, а также при наличии ветвлений и циклов следить за тем, по какому пути (трассе) развивается процесс выполнения программы. Этот приём отладки называется *трассировкой* программы.

4. Индивидуальная часть лабораторной работы

Создайте новый проект. В функцию main() добавьте следующие команды:

```

int a=2,b=3,c=6;
a=номер варианта
b:=a*2;
c=a+b;
b=(a+b)*c
a=b+a+c;

```

Далее необходимо:

1. Найти и исправить синтаксические ошибки.

2. Выполнить программу в пошаговом режиме с помощью отладчика, наблюдая при этом за изменением значений переменных.
3. Результаты наблюдений представить в виде заполненной таблицы 2.

Таблица 2

Изменение значений переменных при выполнении программы

Номер строки	a	b	c
1			
...			

5. Контрольные вопросы

1. Что такое компиляция?
2. Что такое компилятор?
3. Что такое программа?
4. Для чего предназначен язык программирования?
5. Что такое функция?
6. Какая функция обязательно должна присутствовать в программе?
7. Что такое проект в Microsoft Visual Studio?
8. Что такое решение в Microsoft Visual Studio?
9. В чем заключаются отличия проекта и решения?
10. Что такое переменная?
11. Что такое отладка программы?
12. Какие инструменты отладки вы знаете?
13. В чем состоит смысл точки останова?
14. Как поставить точку останова в коде программы?
15. Как просмотреть значение переменной в ходе выполнения программы?
16. Где в Microsoft Visual Studio можно просмотреть список ошибок и предупреждений?

Лабораторная работа №3. Программирование линейного алгоритма

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки программирования линейных алгоритмов на языке Си.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Операторы, выражения и операции

Программа на языке Си состоит из *операторов*. Одним из видов операторов является *простой оператор*, представляющий собой выражение с точкой с запятой.

Выражение – это синтаксически корректное сочетание констант, переменных или вызовов функций и связывающих их операций.

Операция – конструкция в языках программирования, подразумевающая специальный способ записи некоторых действий над операндами для получения результата операции.

Арифметические операции:

- + сложение;
- вычитание;
- * умножение;
- / деление;
- % остаток от деления.

Операции отношения:

- == равно;
- != не равно;
- < меньше;
- <= меньше или равно;
- > больше;
- >= больше или равно.

Операция присваивания: = (например, $a = b$);).

Операции инкремента и декремента: вызывают соответственно увеличение и уменьшение значения выражения на 1.

- ++ операция инкремента;
- операция декремента.

В языке Си имеет значение регистр букв идентификатора. Так, идентификаторы st, ST, St и sT – это не одно и то же.

Основной задачей программирования является обработка информации, поэтому в любом языке программирования имеются средства для ввода и вывода информации. В языке Си ввод и вывод информации осуществляется через функции стандартной библиотеки. Прототипы рассматриваемых функций находятся в файле **stdio.h**. Эта библиотека, в числе прочих содержит следующие функции:

- printf() – для вывода информации;
- scanf() – для ввода информации.

Вывод информации

Функция printf() предназначена для форматированного вывода. Она переводит данные в символьное представление и выводит полученные изображения символов на экран. При этом у программиста имеется возможность форматировать данные, то есть влиять на их представление на экране.

Общая форма записи функции printf():

printf("СтрокаФорматов", объект 1, объект 2, ..., объект n);

СтрокаФорматов состоит из следующих элементов:

- управляющих символов;
- текста, представленного для непосредственного вывода;
- форматов, предназначенных для вывода значений переменных различных типов.

Объекты могут отсутствовать.

Управляющие символы не выводятся на экран, а управляют расположением выводимых символов. Отличительной чертой управляющего символа является наличие обратного слэша '\' перед ним.

Основные управляющие символы:

- '\n' – перевод строки;
- '\t' – горизонтальная табуляция;
- '\v' – вертикальная табуляция;
- '\b' – возврат на символ;
- '\r' – возврат на начало строки;
- '\a' – звуковой сигнал.

Форматы необходимы для того, чтобы указывать вид, в котором информация будет выведена на экран. Отличительной чертой формата является наличие символа процент '%' перед ним:

- %d – целое число типа int со знаком в десятичной системе счисления;
- %u – целое число типа unsigned int;
- %x – целое число типа int со знаком в шестнадцатеричной системе счисления;
- %o – целое число типа int со знаком в восьмеричной системе счисления;
- %hd – целое число типа short со знаком в десятичной системе счисления;
- %hu – целое число типа unsigned short;
- %hx – целое число типа short со знаком в шестнадцатеричной системе счисления;
- %ld – целое число типа long int со знаком в десятичной системе счисления;
- %lu – целое число типа unsigned long int;
- %lx – целое число типа long int со знаком в шестнадцатеричной системе счисления;
- %f – вещественный формат (числа с плавающей точкой типа float);
- %lf – вещественный формат двойной точности (числа с плавающей точкой типа double);
- %e – вещественный формат в экспоненциальной форме (числа с плавающей точкой типа float в экспоненциальной форме);
- %c – символьный формат;
- %s – строковый формат.

Ввод информации

Функция форматированного ввода данных с клавиатуры `scanf()` выполняет чтение данных, вводимых с клавиатуры, преобразует их во внутренний формат и передает вызывающей функции. При этом программист задает правила интерпретации входных данных с помощью спецификаций форматной строки. Общая форма записи функции `scanf()`:

`scanf ("СтрокаФорматов", адрес 1, адрес 2, ...);`

Строка форматов аналогична функции printf(). Для формирования адреса переменной используется символ амперсанд '&':

адрес = &объект

Строка форматов и список аргументов для функции обязательны.

Математические функции

В заголовочном файле <math.h> определены прототипы функции, выполняющие некоторые часто используемые математические задачи.

Перечислим некоторые математические функции:

abs(a) – модуль или абсолютное значение числа a ;

sqrt(a) – корень квадратный из a, причём a не отрицательно;

pow(a, b) – возведение a в степень b;

ceil(a) – округление с избытком;

floor(a) – округление a до наибольшего целого, не превосходящего a;

fmod(a, b) – остаток от деления a на b;

exp(a) – экспонента a, то есть e^a ;

sin(a) – синус a (a задаётся в радианах);

cos(a) – косинус a (a задаётся в радианах);

log(a) – натуральный логарифм a;

log10(a) – десятичный логарифм a;

asin(a) – арксинус a ($-1.0 < a < 1.0$).

3. Общая часть лабораторной работы

Задача. Написать программу для вычисления и вывода на экран значения выражения, заданного следующей формулой:

$$\frac{\sin(x) + a \cdot b}{c + 2x},$$

при заданных пользователем значениях x, a, b, c .

Решение. Проанализируем задачу. Для начала определимся с переменными, которые необходимо ввести. В нашем случае это x, a, b, c . Судя по использованию в выражении, значения a, b, c являются целочисленными, значение x является вещественным. Значит определим переменные a, b, c типа int, переменную x типа double.

Также потребуется переменная для хранения результата. Зададим ей имя *y*. Из вида формулы ясно, что результат будет вещественным (типа `double`).

Теперь запишем заданное в задаче выражение в линейном виде, используя арифметические операции языка Си и математические функции, а также скобки для определения порядка вычислений. Получим

$$(\sin(x) + a * b) / (c + 2 * x).$$

Далее создадим новый проект. Добавим в него файл исходного кода, запишем в нем функцию `main()` и объявим переменные:

```
int main()
{
    int a,b,c;
    double x,y;

    return 0;
}
```

После объявления переменных произведем чтение их значений с клавиатуры, используя функцию `scanf()`:

```
scanf("%lf", &x);
scanf("%d", &a);
scanf("%d", &b);
scanf("%d", &c);
```

Чтобы ее использовать, подключим заголовочный файл `stdio.h`. Для этого в самом начале кода (до функции `main`) добавим строку:

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

После ввода данных вычислим результат заданного выражения и занесём его в переменную *y* с помощью операции присваивания:

```
y = (sin(x)+a*b) / (c+2*x);
```

Заметим, что для использования функции `sin` потребуется подключить заголовочный файл `math.h`, содержащий прототипы математических функций:

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

Теперь осуществим вывод полученного значения *y*. В формате вывода вещественного числа укажем, что все число должно занимать 10 символов, а после запятой должно отображаться три знака. Добавим также управляющий символ перевода строки:

```
printf("%10.3lf\n", y);
```

При запуске программы в нашем случае консольное окно закрывается сразу после ввода исходных данных, и пользователь не

успевает увидеть выведенный результат. Чтобы предотвратить преждевременное закрытие окна, подключим заголовочный файл `stdlib.h` и добавим команду `system("pause")`:

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

int main()
{
    int a,b,c;
    double x,y;
    scanf("%lf", &x);
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &c);

    y=(sin(x)+a*b)/(c+2*x);
    printf("%10.3lf\n", y);
    system("pause");
    return 0;
}
```

Консольное окно с результатами выполнения программы представлено на рис. 1.

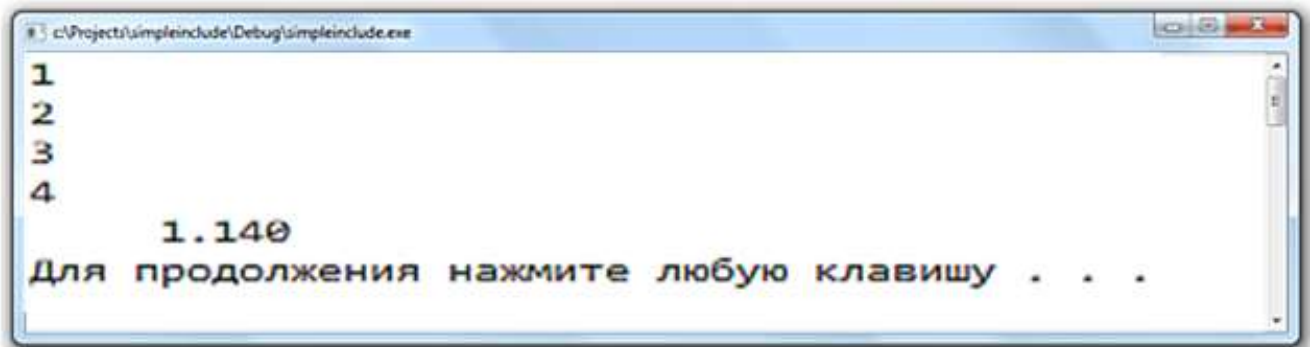


Рис. 1. Окно с результатами выполнения программы

Несмотря на то что алгоритм программы правильный, сама она неинформативна. Чтобы сделать ее более удобной в использовании, добавим так называемые приглашения ко вводу, то есть сообщения с просьбой ввести данные и подсказкой, какие именно данные ожидает программа:

```
printf(" x = "); scanf("%lf", &x);
printf(" a = "); scanf("%d", &a);
printf(" b = "); scanf("%d", &b);
```

```
printf(" c = "); scanf("%d", &c);
```

Вывод полученных результатов организуем следующим образом:

```
printf("Значение y при x = %.3lf, a = %d, b = %d, c = %d  
равно %10.3lf\n", x, a, b, c, y);
```

Консольное окно с результатами выполнения программы представлено на рис. 2.

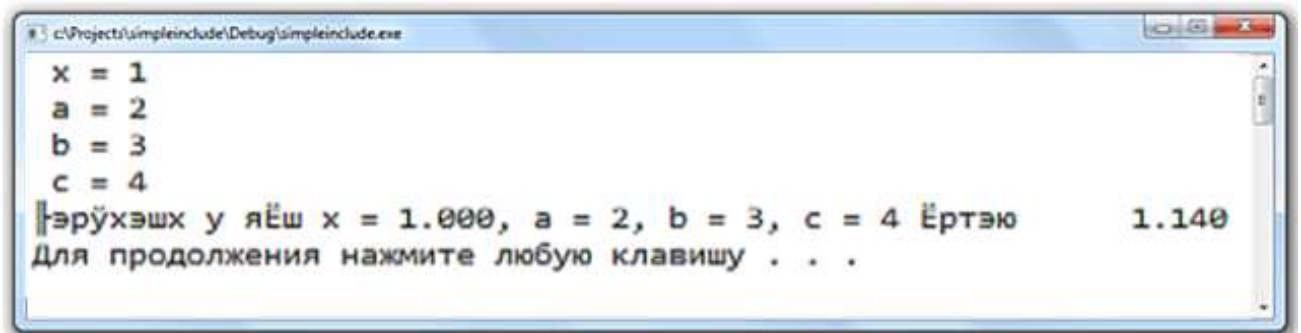


Рис. 2. Окно с результатами выполнения программы после добавления вывода пояснений

Как видно, кириллический шрифт отображается некорректно. Чтобы исправить это, добавим в начало функции `main` следующую строку:

```
setlocale(LC_ALL, "Russian");
```

Кроме того, необходимо подключить заголовочный файл `locale.h`.

Консольное окно с результатами выполнения программы представлено на рис. 3.

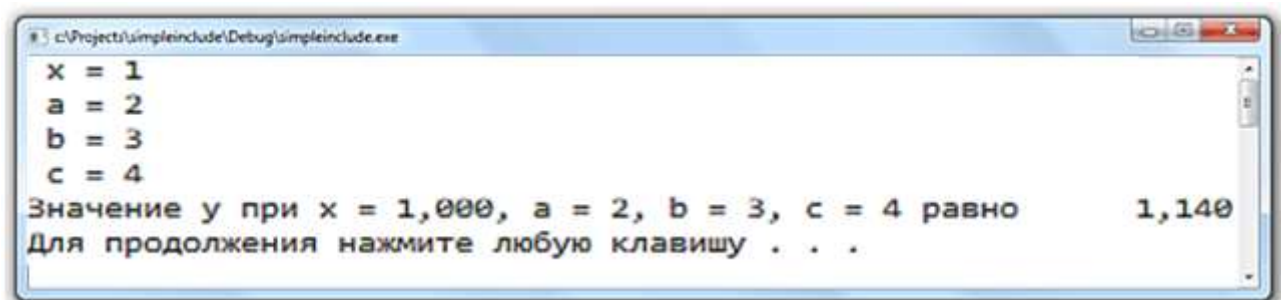


Рис. 3. Окно с результатами выполнения программы после установки русской локализации

Хорошим тоном считается добавление комментариев в код программы. Комментарии позволяют легко ориентироваться в коде, особенно если программа содержит много функций, или ее алгоритм достаточно сложен. Добавим комментарии в код программы. Кроме

того, для облегчения понимания кода отдельные его смысловые блоки выделим с помощью отступов:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <locale.h>

int main()
{
    setlocale(LC_ALL, "Russian"); // русский язык

    // объявление переменных
    int a,b,c;
    double x,y;

    // ввод данных
    printf(" x = "); scanf("%lf", &x);
    printf(" a = "); scanf("%d", &a);
    printf(" b = "); scanf("%d", &b);
    printf(" c = "); scanf("%d", &c);

    // вычисления
    y=(sin(x)+a*b)/(c+2*x);

    // вывод результата
    printf("Значение y при x = %.3lf, a = %d, b = %d, c =
        %d равно %10.3lf\n", x,a,b,c,y);

    system("pause"); // задержка экрана
    return 0;
}
```

На этом решение задачи завершено.

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить три программы на языке Си, каждая из которых должна решать задачу согласно варианту из списка задач для самостоятельного решения соответствующего типа. Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результаты на экран.

Программы должны быть снабжены комментариями и отформатированы.

Для получения дополнительных навыков студентам также предлагается по желанию решить задачи повышенной сложности.

5. Задачи для самостоятельного решения

Задачи первого типа

1. Автомобиль на каждом из пяти одинаковых по длине участках дороги шел с известной средней скоростью. Определить среднюю скорость на всем пути.

2. Определить стоимость мебельного гарнитура, содержащего 4 стула, 2 кресла и 1 стол. Цена изделий соответственно A , B и C .

3. Продуктовый магазин продает яблоки поштучно по 5 руб., груши по 7 руб., апельсины по 8 руб. В первые два дня недели продано: понедельник – A яблок, B груш, C апельсинов; вторник – X яблок, Y груш, Z апельсинов. Определить выручку за фрукты в каждый из этих дней и за оба дня вместе.

4. Определить стоимость мебельного гарнитура, содержащего A стульев, B кресел и C столов. Цена изделий мебельного гарнитура соответственно 1, 2 и 3 тыс. руб.

5. Стоимость мебельного гарнитура, содержащего A стульев, B кресел и C столов, равна D тыс. руб. Цена изделий мебельного гарнитура соответственно 1, 2 и X тыс. руб. Определить X .

6. Мальчик может бегать в 3 раза быстрее, чем ходить. Скорость его ходьбы равна 4 км/ч. Он принял участие в марафонском забеге, но сошел с дистанции, пробежав только X км. Определить время, за которое он преодолел это расстояние.

7. В магазине продается костюмная ткань. Ее цена B руб./м². Определить стоимость куска этой ткани длиной X м и шириной 80 см.

8. В магазине продается костюмная ткань. Стоимость куска этой ткани длиной X м и шириной 80 см равна B руб. Определить цену ткани за один квадратный метр.

9. Хозяин в своем доме хочет оклеить обоями длинную стену без окон и дверей. Длина этой стены равна A м, а высота B м. Обои он будет клеить целыми вертикальными полосками. Рулон обоев имеет длину 12 м и ширину 1 м. Определить стоимость обоев для всей стены, если цена одного рулона K руб.

10. Человек собирается съездить из Лондона в расположенный в 390 милях Эдинбург. Он может ехать на автомобиле марки Роллс-Ройс либо на автомобиле марки Форд. Роллс-Ройс расходует 1 галлон бензина на каждые 15 миль пути. Форд расходует 1 галлон бензина на каждые 36 миль пути. Определить стоимость топлива для поездки в Эдинбург на Ролс-Ройсе, если цена 1 галлона бензина составляет X фунтов? Также определить, сколько денег он сэкономит, если вместо этого он поедет на машине марки Форд.

11. В видеоигре игрок получает 50 очков за сбитый самолет, 100 очков за сбитую ракету и 200 очков за сбитый спутник. Определить количество очков игрока, который сбил A самолетов, B ракет и C спутников.

12. Первая бригада может выполнить задание за A дней, а вторая за B дней. За сколько дней обе бригады выполнят задание, работая вместе?

13. Первая бригада может выполнить задание за A дней, а обе бригады, работая вместе, за B дней. За сколько дней выполнит задание вторая бригада, работая одна?

14. Работник зарабатывает X руб. за каждые 38 ч своей работы. Ему платят в 1,5 раза больше за каждый час сверх 38 ч. Определить сумму заработка, если известно, что он отработал A часов и A заведомо больше 38.

15. Работник зарабатывает X руб. за каждые 38 ч своей работы. Ему платят в 1,5 раза больше за каждый час сверх 38 ч. Определить, сколько часов он должен отработать, чтобы получить Y руб (Y должно быть заведомо больше X).

16. Каждую неделю Юра получает деньги на мелкие расходы. Из них он тратит X руб. на сладости. Это составляет одну четверть того, что он получает еженедельно. Юра сберегает одну треть того, что остается после покупки сладостей. Определить сумму, накопленную Юрой за год.

17. Город A находится от города B на расстоянии S км. Между ними на расстоянии S_1 от города A находится город C . Велосипедист выехал из A в B . Определить, за какое время он доедет до города B , если до города C он ехал со скоростью V_1 км/ч, от C до B со скоростью V_2 км/ч, а в городе C он сделал остановку на 30 мин.

18. Город A находится от города B на расстоянии S км. Между ними на расстоянии S_1 от города A находится город C . Велосипедист

выехал из A в B . На какое время он сделал остановку в городе C , если до города C он ехал со скоростью V_1 км/ч, от C до B со скоростью V_2 км/ч, а всего на дорогу затратил t часов.

19. Определить стоимость путевки, складывающейся из цен на билет (отдельно туда и обратно), стоимости проживания за один день, стоимости питания за один день, количества дней и стоимости экскурсий.

20. В первой четверти в школе было X хорошистов, во второй – на Y больше, чем в первой, а в третьей четверти – на Z хорошистов меньше, чем во второй. Сколько учеников закончили школу без троек в третьей четверти?

21. В цехе работает X человек. Из них Y мужчин, а остальные – женщины. На сколько больше в цехе работает мужчин, чем женщин?

22. Заработок рабочих на фабрике составил C руб. Его необходимо разделить поровну между A рабочими. Определить заработок каждого рабочего.

23. В пяти тестовых опросах мальчик получил оценки. Определить среднее арифметическое оценок, полученных мальчиком в пяти опросах.

24. Работник зарабатывает X руб. за каждые 30 ч своей работы. Ему платят в 1,5 раза больше за каждый час сверх 30 ч. Определить, какую сумму он получит, если работает A часов (A должно быть заведомо больше 38).

25. Магазин продает B машин по цене A руб. за каждую. Определить общую выручку от продажи машин.

Задачи второго типа

1. Определить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b .

2. Заданы координаты трех вершин треугольника (x_1, y_1) , (x_2, y_2) , (x_3, y_3) . Определить его периметр.

3. Заданы координаты трех вершин треугольника (x_1, y_1) , (x_2, y_2) , (x_3, y_3) . Определить его площадь.

4. Определить длину окружности и площадь круга одного и того же заданного радиуса R .

5. Даны два числа. Определить среднее арифметическое кубов этих чисел.

6. Даны два числа. Определить среднее геометрическое модулей этих чисел.
7. Даны два числа. Определить среднее гармоническое этих чисел.
8. Дана длина ребра куба. Определить площадь грани, площадь полной поверхности и объем этого куба.
9. Дана сторона равностороннего треугольника. Определить площадь этого треугольника и длины его высот.
10. Дана сторона равностороннего треугольника. Определить радиусы вписанной в него и описанной около него окружностей.
11. Известна длина окружности. Определить площадь круга, ограниченного этой окружностью.
12. Определить площадь кольца, внутренний радиус которого равен r , а внешний – заданному числу R ($R > r$).
13. Определить площадь равнобедренной трапеции с основаниями a и b и углом X при большем основании a .
14. Определить площадь треугольника, две стороны которого равны a и b , а угол между этими сторонами равен g .
15. Определить путь, пройденный лодкой, если ее скорость в стоячей воде v_1 км/ч, скорость течения реки v_2 км/ч (известно, что $v_1 > v_2$), время движения по озеру t_1 ч, а против течения реки – t_2 ч.
16. Дана величина A , выражающая объем информации в байтах. Перевести A в более крупные единицы измерения информации.
17. Смешано V_1 литров воды температуры T_1 с V_2 литрами воды температуры T_2 . Определить объем и температуру образовавшейся смеси.
18. Заданы координаты четырех вершин прямоугольника. Определить его площадь.
19. Заданы координаты четырех вершин прямоугольника. Определить его периметр.
20. Дана величина A , выражающая объем информации в Гигабайтах. Перевести A в меньшие единицы измерения информации.
21. Определить площадь параллелограмма, две стороны которого равны a и b , а угол между этими сторонами равен g .
22. По введенным значениям радиусов кругов определить, на сколько площадь первого круга больше площади второго.
23. По введенным значениям площадей кругов определить, на сколько радиус первого круга больше радиуса второго.

24. По введенным значениям радиусов окружностей определить, на сколько длина первой окружности больше длины второй.

25. По введенным значениям длин окружностей определить, на сколько радиус первой окружности больше радиуса второй.

Задачи третьего типа

Вычислить значение выражения при заданных с клавиатуры значениях переменных согласно варианту:

$$1. \sqrt[3]{2(x-2)^2(8-x)-1}$$

$$2. \frac{2(x^2+3)}{x^2-2x+5}$$

$$3. 1 + \sqrt[3]{2(x-1)^2(x-7)}$$

$$4. \frac{10x}{x^2+1} - 3$$

$$5. -2 + \sqrt[3]{2(x+1)^2(5-x)}$$

$$6. \sqrt[3]{2x^2(x-3)}$$

$$7. \frac{2(-x^2+7x-7)}{x^2-2x+2} - 1$$

$$8. 1 - \sqrt[3]{2(x-2)^2(5-x)}$$

$$9. 1 + \sqrt[3]{2x^2(x-6)}$$

$$10. \sqrt{a^2 + b^2 - 2ab \cos c}$$

$$11. \frac{ad+bc}{abcd}$$

$$12. \sqrt{1 - \sin^2 x}$$

$$13. \frac{1}{\sqrt{ax^2 + bx + c}}$$

$$14. |2x| + |2x+1|$$

$$15. 3\sqrt{x^2} - 5\sqrt{x+\sqrt{y}}$$

$$16. 1 + \sqrt[3]{2x^2(x-6)}$$

$$17. \frac{ad+bc}{abcd}$$

$$18. 3\sqrt{x^2} - 5\sqrt{x+\sqrt{y}}$$

$$19. \frac{|3x-5|}{x^2y}$$

$$20. \frac{x^2y^3}{|x-5y|}$$

$$21. \frac{\sqrt[3]{x(y-2)}}{|2x-y|}$$

$$22. \frac{\sqrt{|y-x|}}{x^3y^2}$$

$$23. \frac{x^2y^3}{\sqrt[3]{x(y-5)}}$$

$$24. \frac{|2x-5|}{\sqrt[3]{y(x-2)}}$$

$$25. \frac{\sqrt{|x-y|}}{|2x-x^3y^2|}$$

Задачи повышенной сложности

Дано действительное число a . Не пользуясь никакими другими арифметическими операциями, кроме умножения, вычислите:

- a) a^4 за две операции;
- b) a^{13} за пять операций;
- c) a^6 за три операции;
- d) a^{15} за пять операций;
- e) a^3 и a^{10} за четыре операции;
- f) a^4 и a^{20} за пять операций.

6. Контрольные вопросы

1. Что называется выражением в языке Си?
2. Что называется операцией в языке Си?
3. Какие существуют типы операций в языке Си?
4. Какие существуют арифметические операции в языке Си?
5. Какие существуют операции отношения в языке Си?
6. Для чего предназначены операции инкремента и декремента?
7. В каком заголовочном файле находятся прототипы функций, предназначенных для ввода-вывода?
8. Какая функция используется для ввода информации в программах на языке Си?
9. Какая функция используется для вывода информации в программах на языке Си?
10. Из каких элементов состоит строка форматов?
11. Для чего предназначены форматы и какие они бывают?
12. Какой символ используется для получения адреса переменной?
13. Как осуществляется ввод и вывод информации в программах на языке Си?
14. В каком заголовочном файле находятся прототипы математических функций?
15. Привести примеры математических функций языка Си.
16. С помощью какой команды можно предотвратить завершение программы?
17. Как можно настроить корректное отображение кириллического шрифта в консоли?

Лабораторная работа №4. Решение задач на целые числа

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки решения задач, связанных с нахождением остатка от деления целых чисел, а также выделением цифр в десятичной записи числа.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Операция – конструкция в языках программирования, подразумевающая специальный способ записи некоторых действий.

Целая часть и остаток от деления целых чисел

При занесении в целочисленную переменную результата деления целых чисел, этой переменной присваивается только целая часть полученного числа. Так, в результате выполнения следующего кода переменной *c* будет присвоено значение 2:

```
int a,b,c;  
a=2; b=5; c=b/a;
```

Операцию получения остатка от деления рассмотрим на следующем примере:

```
int a, b, c;  
a=2;  
b=5;  
c=b%a;
```

Так как остаток от деления 5 на 2 равен 1, то в переменную *c* будет занесено значение 1.

Десятичная запись числа

В десятичной записи число можно представить в виде суммы произведений соответствующих степеней числа 10 на количество единиц соответствующих разрядов: $123 = 1 \cdot 100 + 2 \cdot 10 + 3 \cdot 1$. Таким образом, чтобы выделить цифру в записи числа, можно использовать операции деления и остатка от деления на соответствующую степень числа 10. Так, первая цифра в записи числа 123 может быть представлена как $123/100$ (целая часть

от деления 123 на 100 равна 1). Последняя цифра может быть представлена как $123 \% 10$ (остаток от деления числа на 10 равен трем).

Среднюю цифру можно выделить следующими действиями: $(123 \% 100) / 10$. В этом случае в начале, взяв остаток от деления, «отбросили» первую цифру и получили число 23, а после этого взяли целую часть от деления на 10 и получили 2.

Этот прием выделения цифр в записи числа подходит для целых чисел с любым количеством разрядов. При выделении цифр в отрицательном числе вначале избавляются от знака, вычисляя модуль числа.

Чтобы составить число из имеющихся цифр на основе его десятичного представления, вычисляют сумму из произведений количества единиц различных разрядов на соответствующие степени десятки. Так, число, в котором a единиц, b десятков и c сотен, может быть представлено выражением вида $c \cdot 100 + b \cdot 10 + a$.

3. Общая часть лабораторной работы

Задача 1. Количество предметов

Задача. С начала суток прошло A мин (A – целое число). Определить количество полных часов, прошедших с начала суток.

Решение. Один час содержит 60 мин, следовательно, целая часть от деления A на 60 и является количеством целых часов, прошедших с начала суток.

Определим, какие переменные потребуются. В переменную A по аналогии с условием занесем количество прошедших минут, то есть исходные данные. В переменную N занесем количество полных прошедших с начала суток часов, то есть результат. По смыслу задачи понятно, что обе переменные будут иметь целочисленный тип.

Функция *main()* программы должна содержать ввод исходных данных (с приглашением ко вводу и пояснением), блок вычислений и вывод результата с пояснением:

```
int main()
{
    int A, N;
    printf("Введите A – количество минут, прошедших
           с начала суток (A – натуральное число): ");
```

```
scanf("%d", &A);
N=A/60;
printf("Количество полных часов, прошедших с начала
      суток: %d", N);
return 0;
}
```

Упражнение. В приведённый код добавить задержку экрана после вывода результата и поддержку кириллицы.

Задача 2. Разложение числа на разряды

Задача. Дано трехзначное число. Вычислить сумму и произведение его цифр.

Решение. Один час содержит 60 мин, следовательно, целая часть от деления A на 60 и является количеством целых часов, прошедших с начала суток.

Решение. Определим следующие целочисленные переменные: a , $a1$, $a2$, $a3$, где a – само число, $a1$, $a2$ и $a3$ – цифры в его записи слева направо. Выделить цифры возможно с помощью операций остатка от деления и целочисленного деления.

Переменные для хранения суммы и произведения цифр заводить не будем, вычисления произведем непосредственно внутри оператора вывода. Получим следующий код:

```
int main()
{
    int a, a1, a2, a3;
    printf ("Введите трехзначное число: ");
    scanf ("%d", &a);
    a1=a/100;           // количество сотен
    a2=(a%100)/10;      // количество десятков
    a3=a%10;           // количество единиц
    printf ("Сумма цифр числа %d равна %d, а произведение
           %d", a, a1+a2+a3, a1*a2*a3);
    return 0;
}
```

Упражнение. В имеющийся код добавить команды, отвечающие за задержку экрана после вывода результата и вывод в консоли текста кириллицей.

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить две программы на языке Си, каждая из которых должна решать задачу соответствующего типа согласно варианту из списка задач для самостоятельного решения. Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результаты на экран.

Программы должны быть снабжены комментариями и отформатированы.

Для получения дополнительных навыков студентам также предлагается по желанию решить задачи повышенной сложности.

5. Задачи для самостоятельного решения

Задачи о количестве предметов

1. С начала суток прошло N секунд (N – целое). Выразить текущее время в формате чч:мм:сс.

2. Дана масса в граммах. Выразить эту массу а) в полных тоннах; б) в полных килограммах.

3. Дана масса в граммах. Выразить эту массу в формате X тонн Y центнеров Z килограмм W грамм.

4. Дана масса в килограммах. Выразить эту массу а) в полных тоннах; б) в полных центнерах.

5. Дана масса в килограммах. Выразить эту массу в формате X тонн Y центнеров Z килограмм.

6. Дана длина в миллиметрах. Выразить эту длину а) в полных метрах; б) в полных сантиметрах.

7. Дана длина в миллиметрах. Выразить эту длину в формате X метров Y сантиметров Z миллиметров.

8. Дана длина в сантиметрах. Выразить эту длину а) в полных метрах; б) в полных дециметрах.

9. Дана длина в сантиметрах. Выразить эту длину в формате X метров Y дециметров Z сантиметров.

10. Дана длина в дециметрах. Выразить эту длину а) в полных километрах; б) в полных метрах.

11. Дана длина в дециметрах. Выразить эту длину в формате X километров Y метров Z дециметров.

12. Дан размер файла в битах. Выразить размер этого файла в формате X килобайтов Y байтов Z бит.

13. Дан размер файла в байтах. Выразить размер этого файла в формате X мегабайтов Y килобайтов Z байтов.

14. Дан размер файла в килобайтах. Выразить размер этого файла в формате X гигабайтов Y мегабайтов Z килобайтов.

15. Даны длины двух отрезков AB и AC . Точка C принадлежит отрезку AB . Определить, сколько отрезков длины AC без наложений можно разместить внутри отрезка AB , а также определить длину отрезка, оставшегося незанятым.

16. Дан прямоугольник со сторонами a и b и квадрат со стороной c . Определить, сколько таких квадратов поместится в прямоугольник, и какая площадь останется незанятой?

17. Пусть дан квадрат со стороной a и прямоугольник со сторонами b и c . Определить, сколько таких прямоугольников поместится в квадрат, и какая площадь останется незанятой?

18. С начала суток прошло N секунд (N – целое). Определить количество секунд, прошедших с начала последней минуты.

19. С начала суток прошло N секунд (N – целое). Определить количество минут, прошедших с начала последнего часа.

20. С начала недели прошло N секунд (N – целое). Определить количество дней, прошедших с начала недели.

21. С начала недели прошло N секунд (N – целое). Определить количество часов, прошедших с начала последнего дня.

22. С начала месяца прошло N секунд (N – целое). Определить количество недель, прошедших с начала месяца.

23. С начала месяца прошло N секунд (N – целое). Определить количество дней, прошедших с начала последней недели.

24. С начала месяца прошло N секунд (N – целое). Определить количество часов, прошедших с начала последнего дня.

25. Дни недели пронумерованы следующим образом: 0 – понедельник, 1 – вторник, ..., 6 – воскресенье. Дано целое число K (от 0 до 365). Определить номер дня недели для K -го дня в году, если 1 января был а) понедельник б) четверг.

26. Дни недели пронумерованы следующим образом: 0 – понедельник, 1 – вторник, ..., 6 – воскресенье. Дано целое число K (от

0 до 365). Определить номер дня недели для K -го дня в году, если 15 января был а) вторник б) пятница.

27. Дни недели пронумерованы следующим образом: 0 – понедельник, 1 – вторник, ..., 6 – воскресенье. Дано целое число K (от 0 до 365). Определить номер дня недели для K -го дня в году, если 20 января был а) среда б) суббота.

28. Дни недели пронумерованы следующим образом: 0 – понедельник, 1 – вторник, ..., 6 – воскресенье. Дано целое число K (от 0 до 365). Определить номер дня недели для K -го дня в году, если 30 января был а) вторник б) воскресенье.

Задачи о разложении числа на разряды

В задачах этого типа недостаточно вывести цифры на экран в заданном порядке, необходимо получить новое число, записав его в целочисленную переменную, а затем вывести это число на экран.

Считать, что цифры в числе нумеруются слева направо начиная с 1, то есть левая цифра считается первой.

1. Дано четырехзначное число. Определить число, полученное записью цифр исходного числа в обратном порядке.

2. Дано пятизначное число. Определить число, полученное записью цифр исходного числа в обратном порядке.

3. Дано шестизначное число. Поменять местами первую цифру с предпоследней, вторую с последней.

4. Дано шестизначное число. Поменять местами первую цифру с последней, третью с предпоследней.

5. Дано шестизначное число. Определить сумму цифр, стоящих на нечетных позициях.

6. Дано шестизначное число. Определить произведение цифр, стоящих на четных позициях.

7. Дано пятизначное число. Первую справа цифру зачеркнули и приписали слева. Определить полученное число.

8. Дано пятизначное число. Первую слева цифру зачеркнули и приписали справа. Определить полученное число.

9. Дано четырехзначное число. Первые две справа цифры зачеркнули и приписали слева. Определить полученное число.

10. Дано четырехзначное число. Первые две слева цифры зачеркнули и приписали справа. Определить полученное число.

11. Дано четырехзначное число. Определить число, полученное при перестановке в нем десятков и единиц.

12. Дано четырехзначное число. Определить число, полученное при перестановке в нем десятков и сотен.

13. Дано пятизначное число. Определить число, полученное при перестановке в нем тысяч и сотен.

14. Дано пятизначное число. Определить число, полученное при перестановке в нем тысяч и десятков.

15. Дано пятизначное число. Определить число, полученное при перестановке в нем сотен и единиц.

16. Дано пятизначное число. Определить число, полученное при перестановке в нем тысяч и единиц.

17. Дано четырехзначное число. Определить число, полученное при перестановке в нем десятков и тысяч.

18. Дано семизначное число. Определить на сколько сумма цифр, стоящих на нечетных позициях, меньше самого числа.

19. Дано семизначное число. Определить на сколько произведение цифр, стоящих на четных позициях, меньше самого числа.

20. Дано шестизначное число. Определить на сколько сумма цифр, стоящих на четных позициях, меньше самого числа.

21. Дано шестизначное число. Определить на сколько произведение цифр, стоящих на нечетных позициях, меньше самого числа.

22. Дано шестизначное число. Определить разность между произведением цифр, стоящих на нечетных позициях, и суммой цифр, стоящих на четных позициях.

23. Дано семизначное число. Определить разность между произведением цифр, стоящих на четных позициях, и суммой цифр, стоящих на нечетных позициях.

Задачи повышенной сложности

1. Часовая стрелка образует угол a (в градусах, от 0 до 360) с лучом, проходящим через центр циферблата и точку, соответствующую 12 часам на циферблате. Ввести с клавиатуры значение a и определить значение угла для минутной стрелки, а

также количество часов и полных минут, соответствующих положению часовой стрелки.

2. Текущее показание электронных часов m часов ($0 \leq m \leq 23$), n минут ($0 \leq n \leq 59$), k секунд ($0 \leq k \leq 59$). Какое время будут показывать часы через p часов q минут r секунд ($p > 0$, $q > 0$, $r > 0$)?

6. Контрольные вопросы

1. Каким образом реализуется получение целой части от деления?

2. Каким образом реализуется получение остатка от деления?

3. Каким образом можно выделить старшую цифру в десятичной записи числа?

4. Каким образом можно выделить среднюю цифру в десятичной записи числа?

5. Каким образом можно выделить младшую цифру в десятичной записи числа?

6. Что необходимо сделать перед выделением цифр в отрицательном числе?

7. Каким образом можно составить число из набора цифр?

ГЛАВА 2. ВЕТВЛЕНИЯ И ЦИКЛЫ. РЕКУРСИЯ

Лабораторная работа №5. Программирование простых ветвлений

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки программирования простых разветвляющихся алгоритмов на языке Си.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Разветвляющимся называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется **ветвью алгоритма**. Признаком разветвляющегося алгоритма является наличие операций проверки условия. Чаще всего для проверки условия используется условный оператор *if*.

Условный оператор *if* может использоваться в форме полной или неполной развилки (табл. 1) [4].

В языке Си условием может быть любое выражение, в результате вычисления которого получается значение. Это значение может быть любого типа, при этом любое ненулевое значение считается истиной, а любое нулевое значение считается ложью. Часто для формулирования условий используются операции отношения и логические операции.



Операции отношения:

== равно;
!= не равно;
< меньше;
<= меньше равно;
> больше;
>= больше равно.

Логические операции:

&& логическое И (одновременное выполнение условий);
|| логическое ИЛИ (выполнение хотя бы одного из условий);
! логическое НЕ (требуется невыполнение условия).

Формы условного оператора *if*

Неполная развилка	Полная развилка
<pre> if(условие) { блок операций; } </pre>	<pre> if(условие) { блок операций 1; } else { блок операций 2; } </pre>
	

3. Общая часть лабораторной работы

Задача 1. Простые ветвления

Задача. Дано целое число. Если оно является положительным, прибавить к нему 1, в противном случае умножить его на 2.

Решение. Для решения задачи используется условный оператор *if* с полной развилкой. Вначале определим переменную целого типа (`int a`), затем организуем ввод ее значения с клавиатуры. Далее проверим условия положительности $a > 0$, при выполнении которого увеличим значение a на единицу, в противном случае умножим на 2. Результат вычислений выведем на экран.

Листинг основы программы для задачи имеет следующий вид (подключение библиотек и задержку экрана организовать самостоятельно):

```

int main()
{
    int a;
    printf("a=");
    scanf("%d", &a);
    if (a>0)
        a++;
    else
        a*=2;
    printf("a= %d", a);
}

```

Заметим, что в этой задаче используется сокращенная запись умножения числа на два и операция инкремента.

Задача 2. Вычисление кусочно-заданной функции

Задача 2. Определить значение кусочно-заданной функции по введенному аргументу:

$$f(x) = \begin{cases} 2x + 3, & x < 0 \\ x^2, & 0 \leq x < 5 \\ 0, & x = 5 \\ \sin(x), & x > 5 \end{cases}$$

Решение. Чтобы понять алгоритм решения задачи, схематично изобразим указанные интервалы для x , над каждым из которых укажем формулу для вычисления значения функции (Рис.1.).

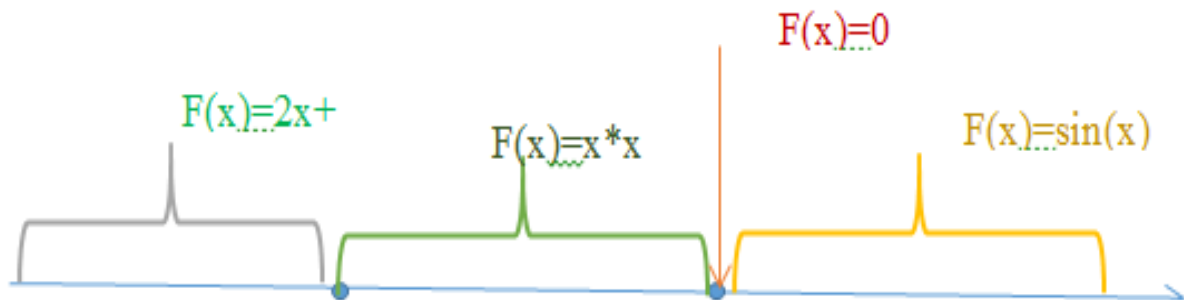


Рис.1. Разбиение числовой оси на множества, соответствующие различным значениям функции

Таким образом видно, что введенное значение переменной x может попасть в один из четырех областей. Для проверки каждого из условий может быть использован вложенный оператор `if`.

Вначале проверяется условие $x < 0$, в случае его невыполнения проверяется условие $x < 5$, затем равенство, а если ни одно из трех

условий не будет выполнено, x попадает в четвертую область, то есть $x > 5$.

Условный оператор будет записан следующим образом (ввод и вывод переменных организовать самостоятельно):

```
if (x<0)
    f=2*x+3;
else
    if (x<5)
        f=x*x;
    else
        if (x==5)
            f=0;
        else
            f=sin(x);
```

Полученную конструкцию можно несколько упростить, используя операцию логического И (&&), подразумевающую под собой одновременное выполнение условий. Так, принадлежность x второй области (0;5) аналогична тому, что x одновременно больше нуля и меньше пяти. Записано это условие будет как $(x>0)&&(x<5)$. Тогда условия выбора формулы для расчета функции на основе аргумента можно записать в следующем виде:

```
if (x<0) {f=2*x+3;}
if ((x>0) && (x<5)) {f=x*x;}
if (x==5) {f=0;}
if (x>5) {f=sin(x);}
```

В этом листинге проверяется принадлежность x всем четырем областям.

Далее проанализируем порядок выполнения алгоритма при различных значениях переменной x . Выясним, какие строки кода выполняются, а какие пропускаются после запуска программы, наблюдая за её выполнением с помощью трассировки в отладчике.

Выполним наблюдения при x , равном следующим значениям: -1, 2, 5, 6. Каждое из этих значений принадлежит одной из областей, выделенных при анализе задачи. Результаты наблюдений занесем в абл. 2.

Полный листинг программы представлен на рис.2.

Чтобы организовать пошаговое выполнение программы отметим точку останова на строке проверки условия $x < 0$. Нажмем клавишу F5. Программа выполняется до этого момента и приостанавливается.

Дальше программу будем выполнять по шагам, отдавая команду на выполнение очередного шага нажатием клавиши F11. Строка кода, которая будет выполнена следующей, отмечается желтой стрелкой (рис.3).

Таблица 2

Пути выполнения программы с ветвлениями

x	Путь выполнения программы	Значение f в результате выполнения программы
-1		
2		
3		
6		

```

#include <stdio.h>
#include "math.h"

int main(void) {
    int x; double f;

    scanf("%d", &x);

    if (x<0)
        {f=2*x+3;}
    else
    {
        if (x<5)
            {f=x*x;}
        else
        {
            if (x==5)
                {f=0;}
            else
                {f=sin(double(x));}
        }
    }

    printf("%lf",f);

    return 0;
}

```

Рис.2. Листинг программы

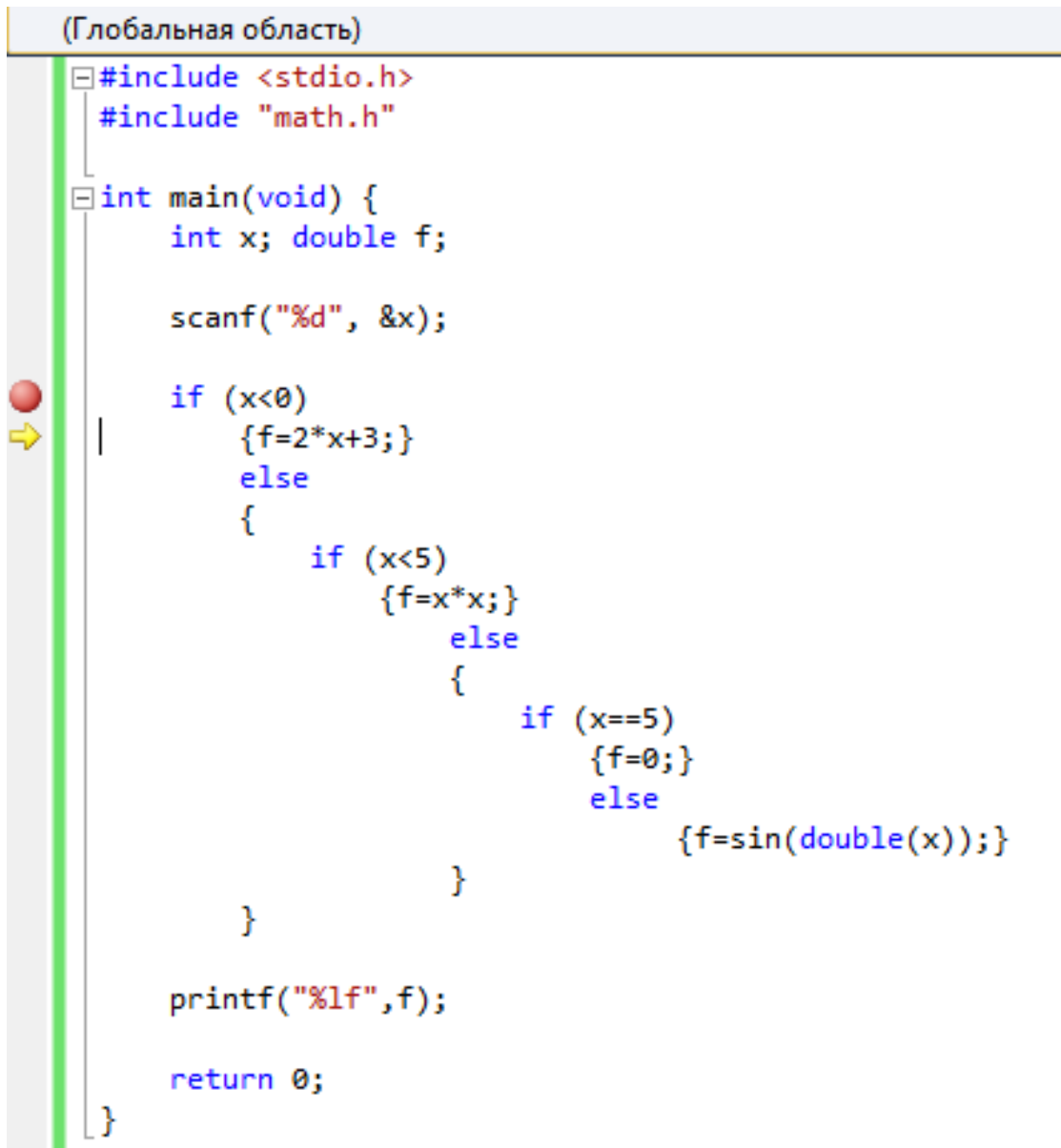


Рис. 3. Пошаговое выполнение программы

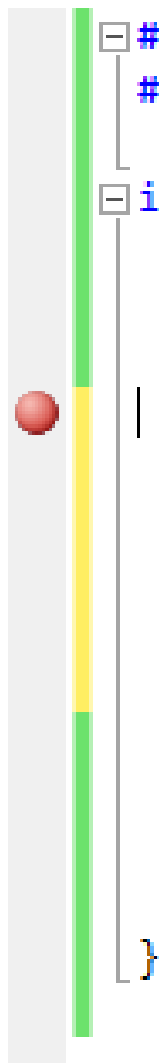
Номера строк кода, которые после ввода значения x выполнила программа, занесем в таблицу. Номер строки кода можно увидеть в строке состояния внизу окна справа. После установки указателя (желтой стрелки) на строку вывода значения функции вновь нажмем клавишу F5 и зафиксируем в таблице выведенное значение.

Наблюдая за пошаговым выполнением программы при различных введенных значениях x , получим табл. 3.

Аналогично трассировку и наблюдения произведем для программы, в которой использована форма записи ветвления с операцией логического И. Её листинг представлен на рис. 4. Выполнив наблюдения и зафиксировав их результаты, получим табл. 4.

Результаты пошагового выполнения программы

x	Путь выполнения программы	Значение f в результате выполнения программы
-1	9,10,24	1
2	9,11,13,14	4
3	9,11,13,15,17,18	0
6	9,11,13,15,17,19,20	-0,27945



```

#include <stdio.h>
#include "math.h"

int main(void) {
    int x; double f;

    scanf("%d", &x);
    if (x<0) {f=2*x+3;}
    if ((x>0)&&(x<5)) {f=x*x;}
    if (x==5) {f=0;}
    if (x>5) {f=sin(double(x));}

    printf("%lf",f);

    return 0;
}

```

Рис.4. Листинг программы, содержащей операции логического И

Таким образом, получены аналогичные результаты выполнения программы, при этом её код стал более компактным. Однако это не привело к увеличению быстродействия программы, так как количество операций сравнения во втором случае даже больше, чем в первом.

Результаты пошагового выполнения программы, содержащей операцию логического И

x	Путь выполнения программы	Значение f в результате выполнения программы
-1	8	1
2	8,9	4
3	8,9,10	0
6	8,9,10,11	-0,27945

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы необходимо выполнить два обязательных задания. Для получения дополнительных навыков студентам также предлагается по желанию решить задачи повышенной сложности.

Задание 1

Написать и отладить программу на языке Си, которая должна решать задачу согласно варианту из списка задач для самостоятельного решения первого типа.

Задание 2

Написать программу на языке Си для вычисления значений кусочно-заданной функции согласно варианту из списка задач для самостоятельного решения второго типа. **Решить задачу необходимо двумя способами:** с помощью вложенных условных операторов и с помощью составных логических условий. Для каждого из решений произвести пошаговую отладку и заполнить таблицы аналогично разобранному примеру (входные значения x выбрать таким образом, чтобы на их основе проверялось каждое из имеющихся условий).

Общие замечания

Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результаты решения задачи на экран.

Программы должны быть снабжены комментариями и отформатированы.

5. Задачи для самостоятельного решения

Задачи первого типа

1. Дано целое число. Проверить, является ли оно четным положительным.
2. Дано целое число. Проверить, является ли оно нечетным отрицательным.
3. Дано целое число. Проверить, является ли оно двузначным.
4. Дано целое число. Проверить, является ли оно трехзначным.
5. Дано целое число. Проверить, является ли оно кратным пяти и трем одновременно.
6. Дано целое число. Проверить, является ли оно кратным семи и девяти одновременно.
7. Дано целое число. Проверить, является ли оно отрицательным двузначным.
8. Дано целое число. Проверить, является ли оно отрицательным трехзначным.
9. Дано целое число. Проверить, является ли оно нечетным кратным 11.
10. Дано целое число. Проверить, является ли оно четным кратным 13.
11. Дано целое число. Если оно положительное, умножить его на два, в противном случае возвести в квадрат.
12. Дано целое число. Если оно положительное, извлечь из него корень, в противном случае умножить на 3.
13. Дано целое число. Если оно отрицательно, возвести его в квадрат, в противном случае извлечь из него корень.
14. Дано целое число. Если оно отрицательно, взять от него модуль, в противном случае возвести в квадрат.
15. Даны два целых числа. Если первое число больше второго, вычислить их сумму, в противном случае вычислить их произведение.
16. Даны два целых числа. Если первое число не меньше второго, вычислить их разность, в противном случае вычислить отношение первого ко второму.

17. Даны два целых числа. Если первое число меньше второго, вычислить их произведение, в противном случае вычислить их разность.
18. Даны два целых числа. Если первое число не больше второго, вычислить их сумму, в противном случае вычислить отношение первого ко второму.
19. Даны два целых числа. Если они равны, вычислить их произведение, в противном случае вычислить их сумму.
20. Даны два целых числа. Если они равны или первое больше второго, вычислить их разность, в противном случае вычислить отношение первого ко второму.
21. Даны координаты точки. Определить, какой координатной четверти принадлежит точка.
22. Даны координаты двух точек. Определить, принадлежат ли они обе одной и той же координатной четверти.
23. Даны координаты точки. Определить, принадлежит ли точка окружности с центром в начале координат и заданным радиусом.

Задачи второго типа

1. $f(x) = \begin{cases} 4x + 1, x < -1 \\ 5 \cos(x), -1 \leq x \leq 10 \\ \operatorname{tg}(x), x > 10 \end{cases}$
2. $f(x) = \begin{cases} x^4, x < 0 \\ 5 \sin(x), 0 \leq x \leq 10 \\ \operatorname{ctg}(x), x > 10 \end{cases}$
3. $f(x) = \begin{cases} 0, x < 0 \\ \frac{1}{x+4}, 0 \leq x < 1 \\ \exp(x), x \geq 1 \end{cases}$
4. $f(x) = \begin{cases} 2x + 5, x \leq -9 \\ 2 \cos(x), -9 < x \leq 9 \\ 0, x > 9 \end{cases}$
5. $f(x) = \begin{cases} x - 1, x \leq -1 \\ 5 \cos(2x), -1 < x < 10 \\ \operatorname{ctg}(x), x \geq 10 \end{cases}$

6. $f(x) = \begin{cases} 21, x \leq 0 \\ \frac{x^2+5}{x+4}, 0 < x \leq 1 \\ \ln(x), x > 1 \end{cases}$
7. $f(x) = \begin{cases} 4x + 10, x \leq -10 \\ 1, -10 < x \leq 10 \\ 2x^2, x > 10 \end{cases}$
8. $f(x) = \begin{cases} 3, x < -1 \\ 5 \operatorname{ctg}(x), -1 \leq x \leq 10 \\ \frac{1}{x}, x > 10 \end{cases}$
9. $f(x) = \begin{cases} 4x, x \leq 5 \\ 5 \log(x), 5 < x < 7 \\ \exp(x), x \geq 7 \end{cases}$
10. $f(x) = \begin{cases} 3x - \frac{1}{x}, x < -1 \\ 5 \cos(3x), -1 \leq x \leq 10 \\ 4, x > 10 \end{cases}$
11. $f(x) = \begin{cases} 3x + 1, x < -1 \\ 3 \sin(x), -1 \leq x \leq 10 \\ \operatorname{ctg}(x), x > 10 \end{cases}$
12. $f(x) = \begin{cases} x^3, x < 0 \\ 4 \cos(x), 0 \leq x \leq 10 \\ \operatorname{tg}(x), x > 10 \end{cases}$
13. $f(x) = \begin{cases} 1, x < 0 \\ \frac{2}{x+2}, 0 \leq x < 1 \\ 2\exp(x), x \geq 1 \end{cases}$
14. $f(x) = \begin{cases} 4x + 10, x \leq -9 \\ 2 \sin(x), -9 < x < 9 \\ 1, x \geq 9 \end{cases}$
15. $f(x) = \begin{cases} 2x - 2, x \leq -1 \\ 3 \cos(3x), -1 < x < 10 \\ \operatorname{tg}(x), x \geq 10 \end{cases}$
16. $f(x) = \begin{cases} 5, x \leq 0 \\ \frac{x^3+1}{2x+3}, 0 < x \leq 1 \\ 5\ln(x), x > 1 \end{cases}$

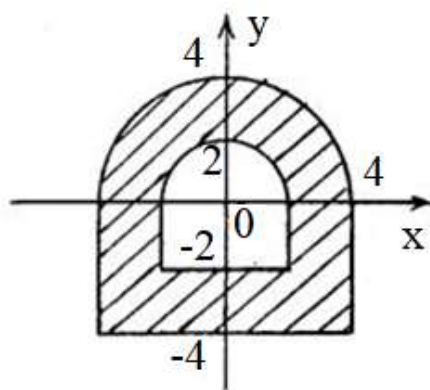
- $$\begin{aligned}
 17. \quad f(x) &= \begin{cases} 5x + 15, x \leq -10 \\ 1, -10 < x \leq 10 \\ 2x^5, x > 10 \end{cases} \\
 18. \quad f(x) &= \begin{cases} 4, x < -1 \\ 6 \operatorname{tg}(x), -1 \leq x \leq 10 \\ \frac{3}{2x}, x > 10 \end{cases} \\
 19. \quad f(x) &= \begin{cases} 2x, x \leq 5 \\ 3 \log(2x), 5 < x < 7 \\ \exp(x), x \geq 7 \end{cases} \\
 20. \quad f(x) &= \begin{cases} 2x - \frac{1}{3x}, x < -1 \\ 3 \cos(2x), -1 \leq x \leq 10 \\ 3, x > 10 \end{cases} \\
 21. \quad f(x) &= \begin{cases} 2x + 5, x < -1 \\ \sin(x), -1 \leq x \leq 10 \\ 2 \operatorname{ctg}(x), x > 10 \end{cases} \\
 22. \quad f(x) &= \begin{cases} 2, x \leq 0 \\ \frac{x^4 + 1}{3x}, 0 < x \leq 1 \\ 2 \ln(x3), x > 1 \end{cases} \\
 23. \quad f(x) &= \begin{cases} 0, x < 0 \\ \frac{1}{2x+2}, 0 \leq x < 1 \\ 3 \exp(x), x \geq 1 \end{cases}
 \end{aligned}$$

Задачи повышенной сложности

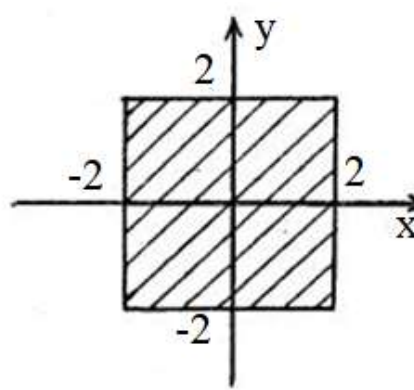
Даны координаты точки. Определить принадлежность точки заштрихованной области, представленной на рис. 5, а – 5, и.

6. Контрольные вопросы

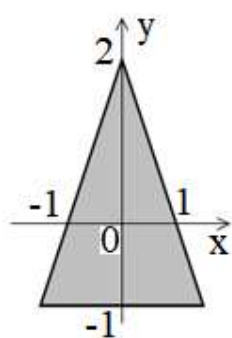
1. Что такое разветвляющийся алгоритм?
2. Что называют ветвью алгоритма?
3. Чем отличаются формы полной и неполной развилки в записи условного оператора?
4. Какие операции отношения используются в языке Си?
5. Какие логические операции используются в языке Си?



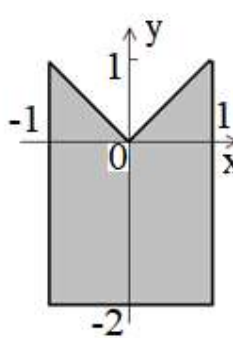
а)



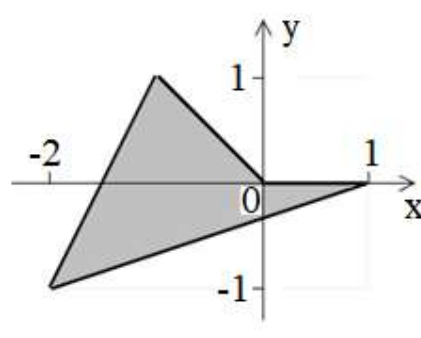
б)



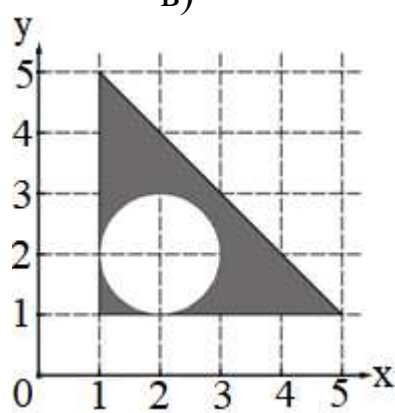
в)



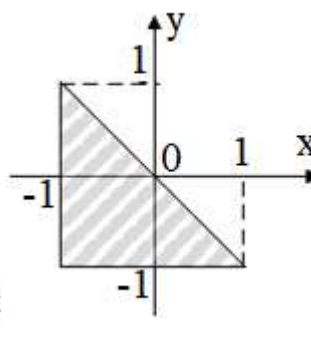
г)



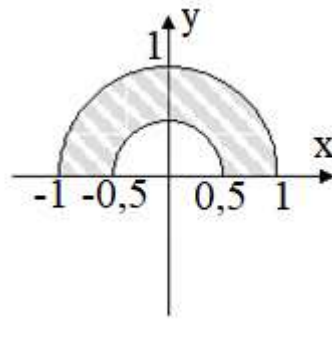
д)



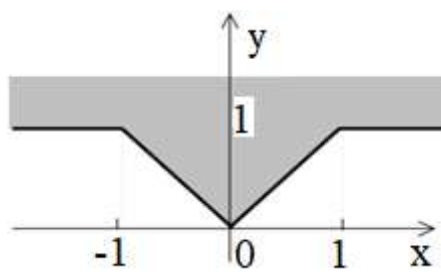
е)



ж)



з)



и)

Рис. 5. Варианты заштрихованных областей

Лабораторная работа №6. Программирование множественного ветвления. Пользовательские функции

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки использования оператора множественного выбора и создания пользовательских функций.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Пользовательские функции

Подпрограммой называется именованная группа операторов, решающая какую-то конкретную задачу, это подпрограмму можно запустить, вызвав по имени, любое количество раз из различных мест программы. Подпрограммы избавили разработчиков от рутинного повторения однотипного кода.

Вызовом подпрограммы называется обращение к ней по имени с целью её использования. Вызов – это запуск подпрограммы.

Функция – это определенная группа операций с уникальным именем, которая может:

- вызываться по имени в любом месте программы;
- получать определенный набор значений из внешней программы в момент вызова;
- возвращать в качестве значения некоторый результат заранее заданного типа.

Чтобы использовать функцию, ее необходимо описать. Для этого необходимо задать тип возвращаемого функцией результата, имя функции, передаваемые ей параметры, а затем записать выполняемый функцией алгоритм. Схема описания функции имеет следующий вид [3]:

```

Тип_результата  Имя_функции  (Тип_пар1  Имя_пар1,  Тип_пар2
Имя_пар2,  ...)
{
    Оператор1;
    ...
    ОператорN;
    return n;
}

```

Тип_результата – некоторый существующий (например, встроенный) тип данных или ключевое слово `void`, указывающее на то что функция никакого значения возвращать не будет.

Имя_функции – уникальный для пространства имён идентификатор.

Тип_парN – некоторый существующий (например, встроенный) тип данных для N-ого аргумента

Имя_парN – уникальное внутри функции имя N-го параметра. Имена параметров можно задавать и в прототипе функции, тогда в определении необходимо использовать те же имена.

ОператорN – некоторые операторы и выражения, содержащиеся внутри функции и выполняющиеся каждый раз при вызове функции. Внутри операторов можно обращаться к глобальным объектам программы; к локальным объектам, объявленным внутри функции; а также к аргументам функции.

`return n` – оператор, останавливающий работу функции и возвращающий `n` в качестве её значения (при этом тип `n` должен соответствовать типу результата в объявлении функции).

Наличие этого оператора обязательно для функции, возвращающей значение. Для функции, объявленной как `void`, можно вызывать оператор `return` без аргументов, это досрочно завершит функцию, иначе – будут выполнены все действия до конца блока описания функции.

Блок определения функции называется также её *телом*.

Оператор множественного выбора

Условный оператор *if*, рассмотренный в предыдущей лабораторной работе, позволяет разделить алгоритм на две ветви. Для разделения алгоритма на большее количество ветвей используется оператор множественного выбора *switch*. Его структура записи представлена ниже [3]:

```
switch (/*переменная или выражение*/)
{
    case /*константное выражение1*/:
    {
        /*группа операторов*/;
        break;
    }
    case /*константное выражение2*/:
```

```

{
/*группа операторов*/;
break;
}
//. . .
default:
    {
        /*группа операторов*/;
    }
}

```

Здесь в скобках после оператора *switch* указывается выражение или переменная, на основе которой производится ветвление алгоритма. Служебное слово *case* указывает возможные варианты значений для условия выбора.

Если фактическое значение выражения, задающего условие, совпадает со значением, указанным после *case*, выполняется группа операторов, записанных в фигурных скобках после него. Если значение не совпадает ни с одним из обозначенных вариантов, выполняется группа операторов в фигурных скобках после служебного слова *default*.

3. Общая часть лабораторной работы

Задача 1. Из пяти введенных значений выбрать наименьшее.

Решение. Чтобы решить задачу, составим программу, в которой объявим пять целочисленных переменных для входных данных a_1 , a_2 , a_3 , a_4 и a_5 , а также ещё одну для результата m .

Алгоритм выбора минимального среди пяти чисел можно словестно описать следующим образом:

1. Сравнить первые два числа и выбрать среди них минимальное.
2. Полученное в шаге 1 минимальное число сравнить с третьим, выбрать среди них минимальное.
3. Полученное в шаге 2 минимальное число сравнить с четвертым, выбрать среди них минимальное.
4. Полученное в шаге 3 минимальное число сравнить с пятым, выбрать среди них минимальное.

Из описания алгоритма видно, что операция сравнения производится 4 раза. Чтобы не описывать ее несколько раз

для каждой группы сравниваемых чисел, создадим функцию выбора минимального значения из двух. Функция возвращает целое число и в нее передаются два параметра (переменные), которые участвуют в равнении. Таким образом, функция выбора минимального из двух значений имеет следующий вид:

```
int min(int a, int b)
{
    if (b<m) return b
    else return m;
}
```

Заметим, что это наглядный, но не самый лучший способ реализации функции. Более эффективный вариант её реализации может иметь следующий вид:

```
int min(int a, int b)
{
    return b<m ? b : a;
}
```

В функции `main()` программы после ввода значений переменной организуем многократный вызов функции `min` для поиска минимального значения среди пяти чисел.

Использование описанной функции для решения задачи имеет следующий вид:

```
m = min(a1, a2);
m = min(m, a3);
m = min(m, a4);
m = min(m, a5);
```

Полный код программы представлен на рис. 1.

Задача 2. По введенному номеру дня в неделе вывести его название.

Решение. Для решения задачи используем оператор множественного выбора ***switch***. Определим целочисленную переменную *n*, введем ее значение с клавиатуры.

Оператор множественного выбора организуем следующим образом. В качестве условия выбора используем переменную *n*. Варианты выбора зададим в скобках после ***case***, указав числа от 1 до 7 (номера дней недели), для каждого из возможных значений выведем на экран название дня недели. Предусмотрим случай, когда

пользователь ввел число, не являющееся номером дня недели. Для этого используем *default* и выведем сообщение об ошибке.

Код программы представлен на рис. 2. (вывод кириллицы в консоли организовать самостоятельно).

```
#include "stdio.h"

int min(int a, int b)
{
    int m=a;
    if (b<m) m=b;
    return m;
}

int main()
{
    int a1,a2,a3,a4,a5,m;

    scanf("%d",&a1);
    scanf("%d",&a2);
    scanf("%d",&a3);
    scanf("%d",&a4);
    scanf("%d",&a5);

    m=min(a1,a2);
    m=min(m,a3);
    m=min(m,a4);
    m=min(m,a5);

    printf("min=%d ",m);
    return 1;
}
```

Рис.1. Программа выбора минимума

На этом решение задачи окончено.

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить две программы на языке Си, каждая из которых должна

решать задачу согласно варианту из списка задач для самостоятельного решения соответствующего типа. Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результаты решения задачи на экран.

Программы должны быть снабжены комментариями и отформатированы.

Для получения дополнительных навыков студентам также предлагается по желанию решить задачи повышенной сложности.

```
#include "stdio.h"

int main()
{
    int n;

    scanf("%d",&n);

    switch (n)
    {
        case 1: printf("понедельник:"); break;
        case 2: printf("вторник:");break;
        case 3: printf("среда:");break;
        case 4: printf("четверг:");break;
        case 5: printf("пятница:");break;
        case 6: printf("суббота:");break;
        case 7: printf("воскресенье:");break;

        default: printf("неверный ввод!");
    }

    return 1;
}
```

Рис. 2. Программа вывода названия дня недели

5. Задачи для самостоятельного решения

Задачи первого типа

1. По номеру введенного месяца вывести название месяца и название сезона.
2. Дано двузначное число. Вывести его название в текстовой форме.
3. Вводятся два числа и символ операции над ними (+; -; *; /). Вывести результаты вычислений.
4. Вводится сторона квадрата. Вывести его площадь, радиус вписанной окружности, площадь описанной окружности, длину диагонали (на выбор пользователя).
5. Вводится сторона квадрата. Вывести его периметр, радиус описанной окружности, площадь вписанной окружности, длину диагонали (на выбор пользователя).
6. Дано количество дней с начала года. Вывести название дня недели, если первое января был понедельник.
7. Введено пять оценок у студента по программированию. Вывести текстовое описание среднего балла по предмету («отлично», «хорошо», «удовлетворительно», «неудовлетворительно»).
8. Даны три числа. Определить их: среднее арифметическое, среднее гармоническое, сумму, минимальное из них (на выбор пользователя).
9. Даны три числа. Определить их: среднее геометрическое, среднее квадратическое, произведение, максимальное из них (на выбор пользователя).
10. Определить синус, тангенс, натуральный логарифм или экспоненту введенного числа (на выбор пользователя).
11. Определить косинус, котангенс, десятичный логарифм или квадрат введенного числа (на выбор пользователя).
12. Определить арксинус, арктангенс, натуральный логарифм или куб введенного числа (на выбор пользователя).
13. Определить арккосинус, арккотангенс, десятичный логарифм или квадратный корень введенного числа (на выбор пользователя).

Задачи второго типа

1. Заданы радиусы четырех окружностей. Определить и вывести минимальную из площадей этих окружностей.

2. Заданы радиусы пяти окружностей. Определить и вывести максимальную из длин этих окружностей.
3. Для каждого из трех отрезков заданы координаты его концов. Определить самый длинный отрезок и вывести его длину.
4. Даны стороны четырех квадратов. Определить среди периметров этих квадратов максимальный.
5. Даны стороны пяти квадратов. Определить среди площадей этих квадратов минимальную.
6. Даны стороны пяти квадратов. Определить квадрат с самой длинной диагональю и вывести ее значение.
7. Даны стороны шести квадратов. Определить количество квадратов, у которых площадь меньше 10.
8. Даны радиусы семи окружностей. Определить количество окружностей, у которых площадь больше 20.
9. Даны радиусы шести окружностей. Определить количество окружностей, у которых длина окружности меньше 15.
10. Эталонная масса одной пачки сортового чая составляет 1,2 кг. Последовательно взвешено 8 пачек чая. Допустимая погрешность массы составляет 10 г. Определить количество пачек, масса которых соответствует заявленной с необходимой точностью.
11. Согласно стандарту, масса одного слитка сплава должна составлять 800 г. Последовательно измерена масса 7 слитков. Допустимое отклонение массы составляет 20 г. Определить процент брака в этой партии.
12. Студенческая группа состоит из пяти человек. Каждый студент сдаёт по три экзамена, успеваемость в группе считается нормальной, если средний балл по группе не менее 3,7. Определить средний балл по группе и сделать вывод об успеваемости.
13. Подарочные комплекты формируются из трех предметов различной стоимости. Имеется четыре таких комплекта. Определить стоимость самого дорогого из них.

Задачи повышенной сложности

1. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, предшествующей указанной.

2. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, следующей за указанной.
3. Локатор ориентирован на одну из сторон света («С» – север, «З» – запад, «Ю» – Юг, «В» – Восток) и может принимать три цифровые команды поворота: 1 – поворот налево, -1 – поворот направо, 2 – поворот на 180° . Дан символ С – исходная ориентация локатора и целый числа №1 и №2 – две посланные команды. Вывести ориентацию локатора после выполнения этих команд.
4. Мастям игральных карт присвоены порядковые номера: 1 – пики, 2 – трефы, 3 – бубны, 4 – червы. Достоинству карт, старших десятки, присвоены номера: 11 – валет, 12 - дама, 13 – король, 14 – туз. Даны два целых числа: N – достоинство ($6 \leq N \leq 14$) и M – масть карты ($1 \leq M \leq 4$). Вывести название соответствующей карты вида «шестерка бубен», «дама червей», «туз треф» и т.п.

6. Контрольные вопросы

1. Что называют подпрограммой?
2. Приведите схему описания функции.
3. Что такое оператор множественного выбора, какова его структура?
4. В чем отличие оператора *if* от *switch*?

Лабораторная работа №7. Использование циклов с предусловием

1. Цель лабораторной работы

Цель лабораторной работы – познакомиться с понятием цикла; научиться применять оператор цикла с предусловием для решения задач.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Цикл – это управляющая конструкция языка программирования, предназначенная для организации многократного повторения фрагмента программы.

Можно выделить два типа циклов: типа «пока» и типа «n-раз».

Первый тип «пока» предназначен для повторения некоторых действий до тех пор, пока выполняется некоторое условие [1]. Пример: увеличивать число на 5 до тех пор, пока оно не станет трёхзначным. Этот тип подразделяется на два подтипа: с предусловием и постусловием.

В лабораторной работе рассматривается **цикл с предусловием**.

Второй тип «n-раз» предназначен для повторения некоторых действий заранее известное количество раз. Пример: умножить число само на себя 4 раза.

Оператор **while** повторяет указанные действия до тех пор, пока его параметр имеет истинное значение и является оператором цикла с предусловием. Например, такой цикл выполнит 4 витка (тело цикла выполнится 4 раза), а на экран будет выведено «1 2 3 4 »:

```
int i = 1;
while (i < 5)
{
    i++;
    printf("%d", i);
}
```

Такой цикл не выполнит ни одного витка (тело цикла не выполнится ни разу) и на экран ничего не выведется:

```
int i = 1;
while (i < 0)
{
    i++;
    printf("%d", i);
}
```

Такой цикл будет выполняться бесконечно, а на экран выведется
«1 2 3 4 5 6 7 ...»

```
int i = 1;
while (true) {
    i++;
    printf("%d", i);
}
```

Условие, определяющее будет ли цикл повторяться снова, проверяется перед каждым шагом цикла, в том числе перед самым первым. Говорят, что происходит предпроверка условия.

3. Общая часть лабораторной работы

Задача 1. Дано целое число А. Определить, является ли оно степенью числа В.

Решение. Вначале заведём целочисленные переменные А и В и введём их значения с клавиатуры. Далее опишем основной алгоритм программы. Если А является степенью числа В означает, то А можно представить в виде $A = B * B * \dots * B * 1$. Таким образом, для решения задачи опишем и проинициализируем единицей целочисленную переменную $C = 1$, затем будем циклически умножать ее на В.

Чтобы понять, до какого момента выполнять умножение, рассмотрим решение задачи на примерах. Пусть $A = 8$ и $B = 2$. Инициализируем переменную C значением 1, а затем будем последовательно умножать её на В, сравнивая с А.

Получим:

- 1) $C = C * B$; // $C = 1 * 2$, значит $C = 2$, то есть $C < A$
- 2) $C = C * B$; // $C = 2 * 2$, значит $C = 4$, то есть $C < A$
- 3) $C = C * B$; // $C = 4 * 2$, значит $C = 8$, то есть $C = A$.

Результат умножения стал равен А. Заметим, что 8 является третьей степенью числа 2.

Теперь рассмотрим случай, когда А – не степень В. Пусть $A = 9$, $B = 2$. Будем выполнять аналогичные операции.

Получим:

- 1) $C = C * B$; // $C = 1 * 2$, значит $C = 2$, то есть $C < A$
- 2) $C = C * B$; // $C = 2 * 2$, значит $C = 4$, то есть $C < A$
- 3) $C = C * B$; // $C = 4 * 2$, значит $C = 8$, то есть $C < A$
- 4) $C = C * B$; // $C = 8 * 2$, значит $C = 16$, то есть $C > A$.

Результат умножения стал больше, чем А.

Таким образом, умножение следует продолжать до тех пор, пока C не станет больше или равно A . То есть условием выполнения цикла будет выражение $C < A$.

Сам цикл будет иметь следующий вид:

```
while (C<A)
    C*=B;
```

После выполнения цикла сделаем вывод с помощью условного оператора о том, является ли A степенью B . Если накопленное произведение C равно A , то является, в противном случае не является.

Листинг программы будет иметь вид

```
int A,B,C=1;
printf("A="); scanf("%d",&A);
printf("B="); scanf("%d",&B);
while (C<A)
    C*=B;
if (A==C) printf("является");
else printf("не является");
```

Отметим, что разбор задачи на конкретных примерах «на бумаге» позволяет лучше сориентироваться в выборе способа решения.

Задача 2. Дано положительное число A . Определить сумму его цифр.

Решение. Эта задача похожа на условие Задачи 2 из лабораторной работы №4, но отличается тем, что не указано количество разрядов в числе.

Вначале определим целочисленную переменную и введём ее значение с клавиатуры. По условию задачи A должно быть больше нуля. Осуществим проверку правильности ввода, для этого в цикле будем предлагать ввести A до тех пор, пока пользователь не введет допустимое значение переменной. Фрагмент кода, осуществляющий проверку правильности ввода, будет иметь следующий вид:

```
int A=0;
while (A<=0)
{
    printf("A=");
    scanf("%d",&A);
}
```

Если пользователь введет отрицательное число или ноль, программа запросит значение снова и будет продолжать запрашивать

значение до тех пор, пока пользователь не введет положительное число. После этого условие $A \leq 0$ станет ложным и произойдет выход из цикла.

Чтобы определить сумму цифр, потребуется последовательно определять по одной все цифры в записи числа. Операция остатка от деления на 10 позволит получить последнюю цифру в записи числа, а операция целочисленного деления позволит получить число без последней цифры. Этот процесс повторяют до тех пор, пока не будут отделены все цифры в записи числа.

Для хранения суммы цифр будем использовать переменную C , предварительно ее обнулив.

Чтобы легче было записать код, рассмотрим алгоритм на примере числа 123 (рис. 1.).



Рис.1. Шаги алгоритма выделения цифр в записи числа

Код цикла, используемого для последовательного выделения цифр в записи числа, будет иметь вид

```

int C=0;
while (A>0)
{
    C+=A%10;
    A=A/10;
}
  
```

Полный листинг решения задачи имеет вид

```

int A=0, C=0;
while (A<=0) // чтение A с проверкой правильности ввода
  
```

```

{
    printf("A=");
    scanf("%d", &A);
}
while (A>0) // подсчет суммы цифр
{
    C+=A%10;
    A=A/10;
}
printf("C=%d", C);

```

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить две программы на языке Си, каждая из которых должна решать задачу соответствующего типа согласно варианту из списка задач для самостоятельного решения. Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результат решения задачи на экран.

Программы должны быть снабжены комментариями и отформатированы.

Для получения дополнительных навыков студентам также предлагается по желанию решить задачи повышенной сложности.

5. Задачи для самостоятельного решения

Задачи первого типа

1. Задано положительное вещественное число. Определить наименьшее целое положительное число, квадрат которого превосходит данное.
2. Даны целые числа A и B , $A < B$. Определить целое положительное число C , такое, что $A^C = B$, если B – степень числа A , или вывести сообщение, что такого числа не существует.
3. Дано целое положительное число A . Определить такое наименьшее целое положительное число B , чтобы выполнялось $B! \geq A$.
4. Дано целое положительное число A . Определить такое наименьшее целое положительное число B , для которого сумма $1+2+\dots+B > A$.

5. Дано целое положительное число A . Определить наибольшее целое положительное число, квадрат которого не превосходит данное.
6. Даны целые положительные числа A и B . Определить, является ли число A факториалом числа B .
7. Выяснить, является ли число A членом арифметической прогрессии с шагом B и первым членом C .
8. Выяснить, является ли число A членом геометрической прогрессии с шагом B и первым членом C .
9. Дано целое положительное число A . Определить такое наибольшее целое число B , для которого сумма $1+2+\dots+B < A$.
10. Даны целые числа A и B , $A < B$. Вывести все степени числа A , не превосходящие B .
11. Вывести все числа, меньшие заданного числа A , из последовательности квадратов натуральных чисел (1, 4, 9, 25 и т.д.).
12. Вывести все числа, меньшие заданного числа A , из последовательности $\sqrt{2}$, $\sqrt{3}$, $\sqrt{4}$ и т.д.
13. Дано целое число N ($N > 1$). Составить алгоритм вывода наименьшего целого K , при котором выполняется неравенство $3K > N$ и самого значения $3K$.
14. Дано целое число N ($N > 1$). Составить алгоритм вывода наибольшего целого K , при котором выполняется неравенство $3K < N$ и самого значения $3K$.
15. Даны числа n и m . Составить алгоритм вывода минимального числа, большего n , которое нацело делится на m .
16. Даны числа n и m ($n > m$). Составить алгоритм поиска максимального из натуральных чисел, не превышающих n , которое нацело делится на m .
17. Дано число n . Составить алгоритм вывода всех натуральных чисел, кратных одиннадцати, меньше n .
18. Дано число n . Составить алгоритм вывода всех натуральных чисел, кратных n , меньше 100.
19. Дано целое число $m > 1$. Получить наименьшее целое k , при котором $4^k > m$.
20. Дано целое число n . Получить наименьшее число вида 2^r , превосходящее n (r - натуральное).

21. Дано целое число $m > 1$. Получить наибольшее целое k , при котором $3^k < m$.
22. Дано целое число n . Получить наибольшее число вида 3^r , не превосходящее n (r - натуральное).

Задачи второго типа

1. Дано целое положительное число. Вывести его в обратном порядке следования цифр.
2. Дано целое положительное число. Определить среднее арифметическое его цифр.
3. Дано целое положительное число. Определить среднее геометрическое его цифр.
4. Дано целое положительное число. Определить среднее гармоническое его цифр.
5. Дано целое положительное число. Определить среднее квадратическое его цифр.
6. Дано целое положительное число. Определить минимальную цифру в его записи.
7. Дано целое положительное число. Определить максимальную цифру в его записи.
8. Дано целое положительное число. Определить минимальную четную цифру в его записи.
9. Дано целое положительное число. Определить максимальную нечетную цифру в его записи.
10. Дано целое положительное число. Определить минимальную цифру, стоящую на нечетной позиции в его записи.
11. Дано целое положительное число. Определить максимальную цифру, стоящую на четной позиции в его записи.
12. Дано целое положительное число. Определить количество четных цифр, стоящих на четной позиции.
13. Дано целое положительное число. Определить количество четных цифр, стоящих на нечетной позиции.
14. Дано целое положительное число. Определить количество нечетных цифр, стоящих на четной позиции.
15. Дано целое положительное число. Определить количество нечетных цифр, стоящих на нечетной позиции.
16. Дано целое положительное число. Определить сумму цифр, стоящих на четных позициях.

17. Дано целое положительное число. Определить сумму цифр, стоящих на нечетных позициях.
18. Дано целое положительное число. Определить произведение цифр, стоящих на четных позициях.
19. Дано целое положительное число. Определить произведение цифр, стоящих на нечетных позициях.
20. Дано целое положительное число. Определить на сколько сумма цифр, стоящих на четных позициях, меньше самого числа.
21. Дано целое положительное число. Определить на сколько сумма цифр, стоящих на нечетных позициях, меньше самого числа.
22. Дано целое положительное число. Определить на сколько произведение цифр, стоящих на четных позициях, меньше самого числа.
23. Дано целое положительное число. Определить на сколько произведение цифр, стоящих на нечетных позициях меньше самого числа.

Задачи повышенной сложности

1. Пусть задано число A в десятичной системе счисления и число B – основание новой системы счисления, $B < 9$. Перевести A из десятичной системы счисления в систему счисления с основанием B .
2. Пусть число A задано в системе счисления $B < 9$. Написать программу перевода A в десятичную систему счисления.
3. Написать программу для вычисления N -го члена последовательности, называемой числами Фибоначчи.

6. Контрольные вопросы

1. Что такое цикл?
2. Для чего используются циклы?
3. Перечислите виды циклов.
4. Какой вид имеет оператор цикла с предусловием на языке Си?

Лабораторная работа №8. Использование циклов с постусловием

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки использования оператора цикла с постусловием для решения задач.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Цикл – это управляющая конструкция языка программирования, предназначенная для организации многократного повторения фрагмента программы.

Цикл с постусловием отличается от цикла *while* тем, что условие в нём проверяется после выполнения цикла, то есть тело этого цикла будет выполнено как минимум один раз (в отличие от цикла *while*, тело которого может вообще не выполниться). Синтаксис цикла [1]:

```
do {
    тело цикла
}while (условие);
```

Следующий фрагмент кода реализует вывод чисел от 1 до 4:

```
int i=1;
do{
    printf("%d  ",i);
    i++;
} while(i<5);
```

Этот фрагмент кода выведет числа от 1 до 5:

```
int i=0;
do {
    i++;
    printf("%d  ",i);
} while(i<5);
```

В этом случае будет выведено только число 6, так как условие цикла не выполняется и тело цикла будет выполнено всего один раз:

```
int i=5;
do {
    i++;
    printf("%d  ",i);
} while(i<5);
```

3. Общая часть лабораторной работы

Задача 1. Написать программу, которая определяет, сколько из введенных чисел не кратны трем. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

Решение. Для решения задачи воспользуемся оператором цикла с постусловием. В теле цикла организуем ввод числа с клавиатуры, проверку условия некрatности трем и увеличение счетчика в случае выполнения условия.

Пусть значения вводимых чисел будут заноситься в переменную *a*, а счетчиком будет являться переменная *c*.

После завершения цикла выведем полученное значение переменной *c*.

Алгоритм программы запишем в виде

```
int c=0;
int a;
do
{
    scanf("%d ", &a);
    if (a%3!=0) c++;
}
while(a>0);
printf("%d ", c);
```

Отметим, если $a=0$, то $a\%3=0$, и влияние на счетчик *c* не оказывается. Если бы по условию требовалось вычислить количество положительных чисел, кратных трем, введенный ноль обеспечил бы выполнение условия кратности, но не являлся членом последовательности введенных положительных чисел, следовательно, счетчик *c* после ввода 0 был бы увеличен на единицу неправильно, и после выполнения цикла из *c* требовалось бы отнять 1, чтобы исключить один лишний инкремент счетчика.

Задача 2. В *Задаче 2* Лабораторной работы №7 организовать проверку корректности введенного значения при помощи цикла с постусловием.

Решение. Для проверки корректности ввода значений удобно использовать цикл с постусловием, поскольку до проверки ввода как минимум один раз производится ввод числового значения.

Для *Задачи 2* из предыдущей лабораторной работы в условии говорено, что вводимое A должно быть больше нуля. Используем цикл `do while` с условием $A > 0$, в теле цикла читая значение переменной с клавиатуры:

```
int A;
do{
    printf("A=");
    scanf("%d", &A);
}while(A<0);
```

Переменная будет прочитана с клавиатуры минимум один раз. Ввод будет продолжаться до тех пор, пока не будет введено положительное значение переменной.

Полный листинг решения задачи имеет вид

```
int A=0;
int C=0;
do{ //чтение переменной A
    printf("A=");
    scanf("%d", &A);
}while(A<0);
while (A>0) // подсчет суммы цифр
{
    C+=A%10;
    A=A/10;
}
printf("C=%d", C);
```

Консольное окно с результатами выполнения программы показано на рис. 1. По рисунку видно, что ввод числа A продолжается до тех пор, пока пользователь не введет положительное число.

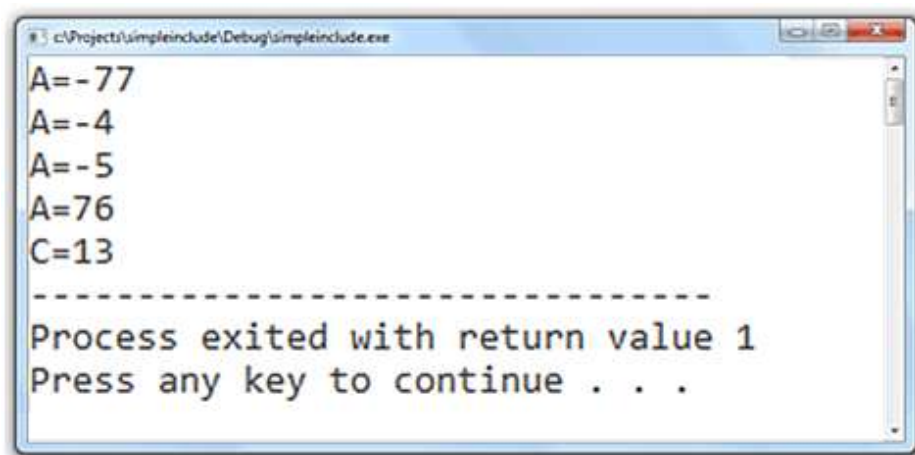


Рис.1. Окно с результатами выполнения программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы выполнить два обязательных задания

Задание 1

Написать и отладить программу на языке Си, которая должна решать задачу согласно варианту из списка задач для самостоятельного решения первого типа.

Задание 2

Взяв за основу свою программу, написанную для решения Задачи 2 лабораторной работы №7, доработать её, организовав проверку корректности введенного значения с помощью цикла с постусловием.

Общие замечания

Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результат решения задачи на экран.

Программы должны быть снабжены комментариями и отформатированы.

5. Задачи для самостоятельного решения

1. Написать программу ввода и суммирования произвольных чисел до тех пор, пока не будет введено число 0.
2. Написать программу перемножения введенных чисел до тех пор, пока пользователь не введет 0.
3. Написать программу определения знаков введенных чисел до тех пор, пока пользователь не введет 0.
4. Написать программу определения модулей введенных чисел до тех пор, пока пользователь не введет 0.
5. Написать программу вычисления квадратов введенных чисел до тех пор, пока пользователь не введет 0.
6. Написать программу вычисления квадратных корней введенных чисел до тех пор, пока пользователь не введет 0.

7. Написать программу для вычисления среднего арифметического последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

8. Написать программу для вычисления среднего геометрического последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

9. Написать программу для вычисления среднего гармонического последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

10. Написать программу для вычисления среднего квадратического последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

11. Написать программу определения минимального четного числа из последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

12. Написать программу определения минимального нечетного числа из последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

13. Написать программу определения максимального четного числа из последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

14. Написать программу определения максимального нечетного числа из последовательности положительных чисел, вводимых с клавиатуры. Ввод данных завершить после того, как пользователь введет 0 или любое отрицательное число.

15. Написать программу, которая определяет количество четных чисел среди введенных пользователем. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

16. Написать программу, которая определяет количество нечетных чисел среди введенных пользователем. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

17. Написать программу, которая определяет, количество чисел, кратных N (N вводится с клавиатуры в начале), среди введенных пользователем. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

18. Написать программу, которая определяет, количество чисел, больших N (N вводится с клавиатуры в начале), среди введенных пользователем. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

19. Написать программу, которая определяет сумму всех четных чисел из введенной последовательности. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

20. Написать программу, которая определяет сумму всех нечетных чисел из введенной последовательности. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

21. Написать программу, которая определяет произведение всех четных чисел из введенной последовательности. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

22. Написать программу, которая определяет произведение всех нечетных чисел из введенной последовательности. Ввод данных продолжать до тех пор, пока пользователь не введет 0.

6. Контрольные вопросы

1. Что такое цикл?
2. Для решения каких задач применяются циклы с предусловием и постусловием?
3. Какой вид имеет оператор цикла с постусловием на языке Си?
4. В чем отличие цикла с постусловием от цикла с предусловием?

Лабораторная работа №9. Использование циклов со счётчиком

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки использования оператора цикла со счётчиком для решения задач.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Общий вид цикла со счётчиком

Цикл – это управляющая конструкция языка программирования, предназначенная для организации многократного повторения фрагмента программы.

for – **цикл со счётчиком** (или **цикл с параметром**, **цикл с фиксированным числом повторений** [1]). Для организации такого цикла необходимо описать три действия:

- инициализация параметра – присваивание параметру цикла начального значения;
- проверка условия продолжения – сравнение значения параметра с некоторым граничным значением;
- коррекция параметра – изменение значения параметра после каждого витка, то есть очередного выполнения тела цикла.

Эти три действия определяются выражениями, записанными в скобках в заголовке цикла и разделёнными точками с запятыми (;). Как правило, параметром цикла является целочисленная переменная. Общая форма записи цикла со счётчиком в языке Си имеет следующий вид:

```
for (инициализация параметра; проверка условия  
    продолжения; коррекция параметра)  
{  
    тело цикла;  
}
```

Инициализация параметра осуществляется только один раз – когда цикл *for* начинает выполняться. Проверка условия окончания осуществляется перед каждым возможным выполнением тела цикла. Когда выражение становится ложным (равным нулю), цикл завершается. Коррекция параметра осуществляется в конце каждого

выполнения тела цикла. Параметр может как увеличиваться, так и уменьшаться.

Пример

```
#include <stdio.h>
int main()
{
    int num;
    for(num = 1; num < 5; num++)
        printf("num = %d\n", num);
    getchar();
    return 0;
}
```

Консольное окно с результатами выполнения программы представлено на рис. 1.



Рис.1. Окно с результатами выполнения программы

Рекомендации по выбору цикла

При выборе цикла принимают во внимание необходимость проверки условия при входе в цикл или по завершении прохождения цикла. Если цикл ориентирован на работу с параметром, который меняется в заданных пределах с заданным шагом, то более предпочтительным является параметрический цикл.

3. Общая часть лабораторной работы

Задача. Вычислить сумму квадратов целых чисел от А до В (А и В – целые числа).

Решение. Сначала запросим у пользователя и обеспечим ввод с клавиатуры значений переменных А и В. Далее организуем цикл с параметром, в котором начальное значение равно А, конечное В, а

счетчик за один ход увеличивается на единицу. Заголовок цикла будет иметь вид

```
for (int i=A; i<=B; i++)
```

При этом счетчик *i* будет принимать значения, равные числам из отрезка от *A* до *B*, включая его границы.

Для хранения значения суммы определим переменную *S*. Перед циклом обнулим её, а на каждом витке цикла к её текущему значению будем прибавлять квадрат счётчика цикла. Получим следующий фрагмент кода:

```
int S=0;
for (int i= A; i<=B; i++)
    S+=i*i;
```

После вычисления суммы в цикле, требуется вывести полученное значение на экран. Полный код программы представлен на рис.2.

```
int main(void) {
    int A,B,S=0;

    scanf("%d", &A);
    scanf("%d", &B);

    for (int i=A;i<=B;i++)
        S+=i*i;

    printf("%d",S);

    return 0;
}
```

Рис.2. Код программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы выполнить два обязательных задания.

Задание 1

Написать и отладить программу на языке Си, которая должна решать задачу согласно варианту из списка задач для самостоятельного решения.

Задание 2

Взяв за основу свою программу, написанную для решения Задачи 2 лабораторной работы №5 (вычисление значений кусочно-заданной функции). Доработать программу так, чтобы она выполняла табулирование функции, то есть вычисление и вывод ее значений при изменении аргумента от некоторого начального значения A до некоторого конечного значения B (включая A и B) с определенным шагом C . Параметры A , B и C вводятся с клавиатуры и могут быть дробными числами.

Общие замечания

Каждая программа должна запросить у пользователя все необходимые исходные данные (с проверкой их корректности) и вывести результаты на экран.

Программы должны быть снабжены комментариями и отформатированы.

5. Задачи для самостоятельного решения

1. Арифметико-геометрическая прогрессия — это последовательность чисел u_n , задаваемая соотношением $u_{n+1} = qu_n + d$, где q и d — константы. Пусть заданы первый член прогрессии и параметры q и d . Вычислить сумму первых n ее членов.

2. Вычислить значение факториала целого числа по формуле $a! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot a$.

3. Дано натуральное число n . Вычислить сумму

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

4. Не используя функцию возведения в степень, вычислить A^B .

5. Вычислить сумму всех четных чисел от A до B включительно.

6. Вычислить произведение всех нечетных чисел от A до B включительно.

7. Дано натуральное число n и вещественное число x .
Вычислить $1 + \frac{x}{2} + \frac{x^2}{4} + \dots + \frac{x^n}{2^n}$.

8. Вычислить произведение всех двузначных чисел, оканчивающихся на цифру a .

9. Вычислить сумму всех трехзначных чисел, оканчивающихся на цифру a .

10. Пусть задана последовательность Фибоначчи: $a_1 = 1, a_2 = 1, a_n = a_{n-1} + a_{n-2}$. Определить n -й член последовательности.

11. Пусть заданы первый член и шаг арифметической прогрессии. Вычислить сумму первых n ее членов.

12. Пусть заданы первый член и знаменатель геометрической прогрессии. Вычислить сумму первых n ее членов.

13. Дано натуральное число n . Вычислить произведение $P = \left(1 + \frac{1}{1^2}\right) \cdot \left(1 + \frac{1}{2^2}\right) \cdot \dots \cdot \left(1 + \frac{1}{n^2}\right)$

14. Дано натуральное число n и действительное число a .
Вычислить произведение $P = a \cdot (a+1) \cdot (a+2) \cdot \dots \cdot (a+n-1)$

15. Дано натуральное число n . Вычислить сумму $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$

16. Дано натуральное число n . Вычислить сумму n первых слагаемых $S = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} \dots$

17. Пусть последовательность чисел образована по следующему правилу: $a_1 = 1$; $a_k = k \cdot a_{k-1} + 1/k$; $k = 1, 2, \dots$. Дано целое число n .
Получить a_n .

18. Вычислить значения выражений y для значений x , равных 1, 2, ..., 20: $y = 2t^2 + 2t + 2, t = 1 + x$.

19. Вычислить значения выражений y для значений x , равных 3, 6, ..., 30: $y = t^3 + 3t^2 + 1, t = 2 - x$.

20. Вычислить значения выражений z для значений x , равных 2, 4, ..., 20: $z = 8f^3 - f, f = 2x$.

21. Вычислить значения выражений z для значений x , равных 5, 10, ..., 100: $z = 6f^2 - 2f, f = 2x - 1$.

22. Вычислить значения выражений z для значений x , равных 4, 8, ..., 80: $z = 5f^3 - 3f^2, f = 1 - 3x$.

6. Контрольные вопросы

1. Какой вид имеет оператор цикла со счетчиком на языке Си?
2. По каким критериям выбирают тип оператора цикла при решении задач?
3. Может ли цикл с параметром быть замененным циклом с условием?
4. Переменные каких типов могут быть использованы в качестве счетчика в цикле с параметром?

Лабораторная работа №10. Использование вложенных циклов

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки использования вложенных циклов для решения задач.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Схема применения вложенных циклов

Цикл – это управляющая конструкция языка программирования, предназначенная для организации многократного повторения фрагмента программы.

В Си допускаются **вложенные** циклы, то есть такие конструкции, когда один цикл находится внутри другого [2]:

```
for ( i=0; i<n; i++)      // цикл 1
{
    // тело цикла 1;
    for( j=0; j<n; j++)    // цикл 2
    {
        // тело цикла 2;
    }
}
```

Пример

Пусть необходимо вывести числа от 0 до 99 так, чтобы они были расположены по 10 в каждой строке. Это можно сделать с помощью вложенных циклов:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    unsigned int i,j;
    for(i=0; i<10; i++)
    {
        for(j=0; j<10; j++)
        {
            printf("%2d ",i*10+j);
        }
        printf("\n");    // перевод строки
    }
    getchar();
    return 0;
}
```

Операторы **break** и **continue**

В теле любого цикла можно использовать операторы **break** и **continue**. Оператор **break** позволяет выйти из цикла, не завершая его. Оператор **continue** позволяет пропустить часть операторов тела цикла и начать новую итерацию [2].

Пример

Пусть из матрицы размера 10*10, заполненной по порядку числами от 0 до 99, необходимо вывести только её часть, расположенную ниже главной диагонали. Чтобы не выводить «лишние» элементы очередной строки, будем досрочно завершать выполнение вложенного цикла оператором **break**, когда номер столбца сравняется или превысит номер строки.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    unsigned int i,j;
    for(i=0; i<10; i++)
    {
        for(j=0; j<10; j++)
        {
            if(j>=i)
                break;
            printf("%2d ",i*10+j);
        }
        printf("\n");
    }
    getchar();
    return 0;
}
```

3. Общая часть лабораторной работы

Задача. Составить программу для вычисления значения выражения при введенных параметрах: $\prod_{i=1}^n \sum_{j=0}^m (\sin(i) - 2j)$.

Решение. Для решения задачи будем использовать два цикла. Первый, внешний, будет содержать счетчик по переменной *i*, второй, внутренний, будет изменять значения переменной *j*.

Во внутреннем цикле будет вычисляться сумма выражений. Во внешнем будет вычисляться произведение результатов, полученных

после выполнения внутреннего цикла. Заметим, что результаты суммирования внутри внешнего цикла перед входом во вложенный цикл требуется обнулить.

Результаты вычислений требуется занести в переменную типа `double`.

Фрагмент программы, реализующий решение этой задачи, может иметь вид

```
double s=0, p=1;
int n, m;
scanf("%d", &n);
scanf("%d", &m);

for(int i=1; i<=n;i++)
{
    s=0;
    for (int j=1; j<=m; j++)
        s+= sin(i)-2*j;
    p*=s;
}
printf("p=%lf", p);
```

Окно с результатами выполнения программы представлен на рис.1.

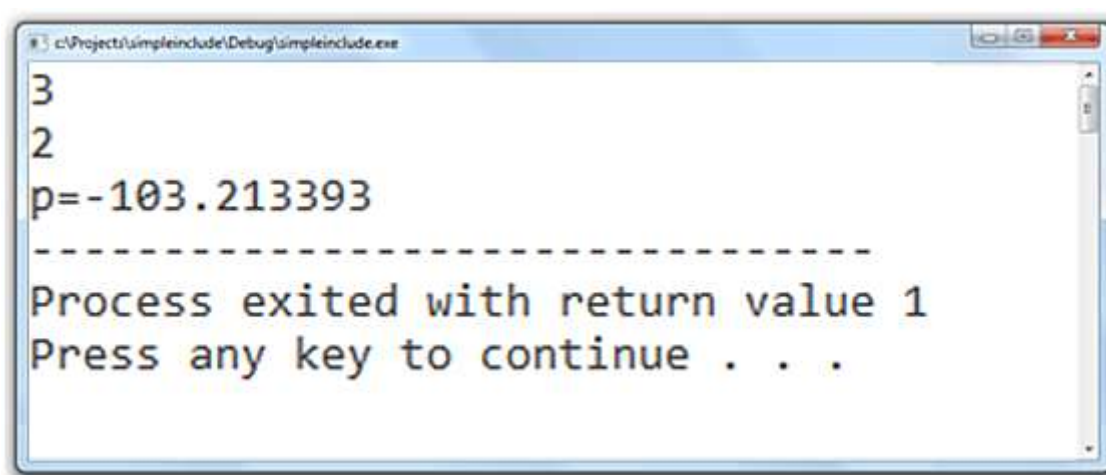


Рис.1. Окно с результатами выполнения программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить программу на языке Си, которая должна решать задачу

согласно варианту из списка задач для самостоятельного решения. Программа должна запросить у пользователя все необходимые исходные данные и вывести результат решения задачи на экран.

Программа должна быть снабжена комментариями и отформатирована.

5. Задачи для самостоятельного решения

1. Доказать гипотезу Сиракуз на диапазоне натуральных чисел от A до B . Гипотеза Сиракуз утверждает, что любое натуральное число сводится к единице в результате повторения следующих действий над самим числом и результатами этих действий.

а. Если число четное, то разделить его на 2.

б. Если нечетное, то умножить его на 3, прибавить 1 и разделить на 2.

2. Дано натуральное число n . Составить программу вычисления выражения $1^1+1^2+1^3+\dots+1^n+2^1+2^2+2^3+\dots+2^n+n^1+n^2+n^3+\dots+n^n$

3. Дано натуральное число n . Найти все натуральные числа a, b, c из интервала от 1 до n , для которых выполняется равенство $a^2+b^2=c^2$

4. Дано натуральное число n . Найти все простые дроби, заключённые между 0 и 1, знаменатели которых не превышают n (дробь задаётся двумя натуральными числами - числителем и знаменателем).

5. Определить размеры всех прямоугольников, площади которых равны заданному натуральному числу S , и стороны которых выражены натуральными числами.

6. Вывести таблицу умножения натуральных чисел $A \times B$.

7. Вводятся десять натуральных чисел больше 2. Посчитать, сколько среди них простых чисел.

8. Определить количество простых чисел в интервале от A до B .

9. Дано натуральное число n . Найти все совершенные числа до n . Совершенное число – это такое число, которое равно сумме всех своих делителей, кроме себя самого. Так, число 6 является совершенным, так как кроме себя самого делится на числа 1, 2 и 3, которые в сумме дают 6.

10. Даны натуральные числа A, B ($A < B$) и n . Найти такие натуральные числа от A до B , включая границы, количество делителей у которых не меньше значения n .

11. Дано натуральное число n . Можно его представить в виде суммы трёх квадратов натуральных чисел? Если можно, то указать все тройки x, y, z таких натуральных чисел, что $x^2 + y^2 + z^2 = n$.

12. Найти все целочисленные решения уравнения $2x + 3y = z$ на интервале от A до B .

13. Найти все целочисленные решения уравнения $3x - 4z = 2y$ на интервале от A до B .

14. Найти все натуральные решения уравнения $5y + z = 3x$ на интервале от 1 до n .

15. Дано натуральное число n . Найти натуральное число от 1 до n с максимальной суммой делителей.

16. Посчитать, сколько раз встречается определенная цифра n во всех натуральных числах из промежутка от A до B .

17. Среди натуральных чисел из промежутка от A до B найти число с наибольшей суммой цифр. Вывести на экран это число и сумму его цифр.

18. Среди натуральных чисел из промежутка от A до B найти число с наименьшим произведением цифр. Вывести на экран это число и произведение его цифр.

19. Найти все простые делители числа A .

20. Вычислить значение выражения при введенных параметрах:

$$\sum_{j=1}^m \prod_{i=1}^n (ai - bj)$$

21. Вычислить значение выражения при введенных параметрах:

$$\sum_{i=1}^n (i + \prod_{j=0}^m (a + bj))$$

22. Вычислить значение выражения при введенных параметрах:

$$\prod_{j=1}^m (a * j + \sum_{i=0}^n (i + bj))$$

6. Контрольные вопросы

1. Что такое цикл?
2. Что такое вложенные циклы?
3. Для решения каких задач используются вложенные циклы?

4. Существуют ли ограничения на количество вложенных циклов?
5. Какой оператор позволяет прекратить выполнение текущей итерации цикла и перейти к следующей?
6. Какой операор позволяет приндительно завершить выполнение цикла?
7. Циклы какого типа могут содержать вложенные циклы?

Лабораторная работа №11. Числовые ряды

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки использования циклов для работы с числовыми рядами.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Пусть имеется **числовая последовательность** $a_1, a_2, a_3, \dots, a_n, \dots$, где $a_k \in R, k=1,2,3,\dots$.

Примером числовой последовательности может послужить арифметическая прогрессия с шагом 3: 1, 4, 7, 11... или геометрическая прогрессия со знаменателем -0.5: 2, 1, 1/2, 1/4, 1/8...

Числовой ряд – это сумма членов числовой последовательности вида [13]

$$\sum_{k=1}^{\infty} a_k = a_1 + a_2 + \dots + a_n + \dots$$

Здесь a_k называют **общим членом** числового ряда или **k -ым членом ряда**.

Также вводят понятие **частичной суммы** ряда, это сумма вида

$$s_n = \sum_{k=1}^n a_k = a_1 + a_2 + \dots + a_n$$

Для работы с последовательностями и рядами удобно использовать цикл **for**.

3. Общая часть лабораторной работы

Задача 1. Вычисление частичной суммы числового ряда

Задача. Вычислить частичную сумму ряда $\sum_{i=1}^n (-1)^i * \frac{1}{x}$

Решение. Для решения задачи необходимо вести значения n и x , после чего организовать цикл, в котором в сумму будет прибавляться результат вычисления очередного члена ряда. Степень $(-1)^i$ можно хранить в отдельной переменной и домножать её на каждом витке цикла на -1. Это позволит избежать вычисления степени на каждом витке. Заметим, что возведение (-1) в степень обеспечивает

знакопеременность ряда, но не влияет на величину модуля членов ряда. Получим следующий код:

```
int n, st=1;
double s=0, x;
scanf("%d%lf", &n, &x);
for(int i=1; i<=n; i++)
{
    st*=-1;
    s+= st/x;
}
printf("%lf", s);
```

Самостоятельно добавьте в программу проверку ввода недопустимых значений и вывод текстовых подсказок.

Задача 2. Приближенное вычисление суммы числового ряда

Задача. Дан бесконечный знакопеременный ряд $\sum_{i=1}^{\infty} \frac{(-1)^i x^i}{i!}$. Вычислить его приближённую сумму с заданной точностью.

Решение. Введем параметр x и требуемую точность e . Затем организуем цикл с постусловием, на каждом витке которого будем вычислять очередной член ряда и добавлять его к накопленному значению суммы. В качестве условия продолжения цикла примем утверждение, что очередной член ряда по модулю превосходит заданную точность. Значение степеней и факториала будем формировать внутри тела цикла, умножая предыдущие их значения соответственно на i , x и i .

Получим код

```
#include <stdio.h>
#include <math.h>
int main()
{
    int st1, i;
    double s=0, x, stx, e, sl, f;
    scanf("%lf%lf", &x, &e);
    st1=-1;
    f=1;
    stx=x;
    i=1;
    do
```

```

{
    sl=st1*stx/f;
    s+=sl;
    i++;
    stx*=x;
    st1*=-1;
    f*=i;
}while(fabs(sl)>e);
printf("%lf",s);

return 1;
}

```

Заметим, что если $|x|$ значительно превосходит 1, то даже тип *double*, используемый для хранения степени x и факториала, переполнится до того, как очередной знаменатель станет настолько больше числителя, что член ряда станет по модулю меньше заданной точности.

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить две программы на языке Си, каждая из которых должна решать задачу соответствующего типа согласно варианту из списка задач для самостоятельного решения. Каждая программа должна запросить у пользователя все необходимые исходные данные и вывести результаты решения задачи на экран.

Программы должны быть снабжены комментариями и отформатированы.

Для получения дополнительных навыков студентам также предлагается по желанию решить задачи повышенной сложности.

5. Задачи для самостоятельного решения

Задачи на вычисление частичной суммы ряда

Вычислить частичную сумму ряда согласно варианту. В процессе суммирования выводить на экран также все текущие элементы ряда.

1. $\sum_{i=1}^n (-1)^i \frac{i+1}{x}$
2. $\sum_{i=1}^n (-1)^{i+1} 2x$

3. $\sum_{i=1}^n (-1)^{-i} \ln(x)$
4. $\sum_{i=1}^n (-1)^i (i + x)$
5. $\sum_{i=1}^n (-1)^{-i} \exp(ix)$
6. $\sum_{i=1}^n (-1)^i \sin(ix)$
7. $\sum_{i=1}^n (-1)^{i+1} 1/\cos(x)$
8. $\sum_{i=1}^n (-1)^i \frac{x}{i}$
9. $\sum_{i=1}^n (-1)^i \frac{2+i}{3x}$
10. $\sum_{i=1}^n (-1)^{2i+1} \frac{1}{x+i}$
11. $\sum_{i=1}^n (-1)^i \frac{i-1}{1-x}$
12. $\sum_{i=1}^n (-1)^{i-1} 3x$
13. $\sum_{i=1}^n (-1)^{1-i} 2\ln(x)$
14. $\sum_{i=1}^n (-1)^{i+1} (2i - x)$
15. $\sum_{i=1}^n (-1)^i \exp(-ix)$
16. $\sum_{i=1}^n (-1)^{i-1} \sin(-ix)$
17. $\sum_{i=1}^n (-1)^{i+1} \cos(-x)$
18. $\sum_{i=1}^n (-1)^{-i} \frac{1-x}{i+1}$
19. $\sum_{i=1}^n (-1)^{i-1} \frac{2-i}{1-3x}$
20. $\sum_{i=1}^n (-1)^{2i} \frac{2}{x-i}$
21. $\sum_{i=1}^n (-1)^i \frac{1+x}{i-1}$
22. $\sum_{i=1}^n (-1)^{i+1} \frac{2+i}{3x}$
23. $\sum_{i=1}^n (-1)^{3i} \frac{2}{x+i}$

Задачи на приближённое вычисление суммы ряда

Вычислить приближённую сумму ряда с заданной точностью согласно варианту. В процессе суммирования выводить на экран также все текущие элементы ряда.:

1. $\sum_{i=1}^{\infty} (-1)^i \frac{\exp(x)}{i!}$
2. $\sum_{i=1}^{\infty} (-1)^i \frac{(x+1)!}{i^i}$
3. $\sum_{i=1}^{\infty} (-1)^i \frac{|\sin(i^x)|}{i(x+1)!}$

4. $\sum_{i=1}^{\infty} (-1)^i \frac{(2x)^i}{i!(x+1)}$
5. $\sum_{i=1}^{\infty} (-1)^i \frac{2^{x^i}}{i!}$
6. $\sum_{i=1}^{\infty} (-1)^i \frac{x^i}{(x+4)!}$
7. $\sum_{i=1}^{\infty} (-1)^i \frac{\sin(x^i)}{i!}$
8. $\sum_{i=1}^{\infty} (-1)^i \frac{|\cos(i!)| * i}{2^{x^i}}$
9. $\sum_{i=1}^{\infty} (-1)^i \frac{5^{x^i}}{(i+1)!}$
10. $\sum_{i=1}^{\infty} (-1)^i \frac{|(x-9)|^i}{(i+4)!}$
11. $\sum_{i=1}^{\infty} (-1)^i \frac{\exp(i)}{i!}$
12. $\sum_{i=1}^{\infty} (-1)^i \frac{(x+i)!}{i^i}$
13. $\sum_{i=1}^{\infty} (-1)^i \frac{|\cos(i^i)|}{(x+i)!}$
14. $\sum_{i=1}^{\infty} (-1)^i \frac{|(3x)|^i}{i!(x+1)!}$
15. $\sum_{i=1}^{\infty} (-1)^i \frac{2^{i^x}}{(i+1)!}$
16. $\sum_{i=1}^{\infty} (-1)^i \frac{3x^i}{(x+2)!}$
17. $\sum_{i=1}^{\infty} (-1)^i \frac{|\cos(2x^i)|}{(i+2)!}$
18. $\sum_{i=1}^{\infty} (-1)^i \frac{|\cos(i!)| * i}{3^{x^i}}$
19. $\sum_{i=1}^{\infty} (-1)^i \frac{4^{x^i}}{(i+2)!}$
20. $\sum_{i=1}^{\infty} (-1)^i \frac{(x-5)^{i+1}}{(i+3)!}$
21. $\sum_{i=1}^{\infty} (-1)^i \frac{3^{x^i}}{(2i+3)!}$
22. $\sum_{i=1}^{\infty} (-1)^i \frac{|(x-2)|^{i+2}}{(i+x)!}$
23. $\sum_{i=1}^{\infty} (-1)^i \frac{(x-3)^{i+3}}{(2i+x)!}$

Задание повышенной сложности

Вычислить частичную сумму ряда:

$$\sum_{i=1}^n \frac{i^i + i! - x^i + (x+1)^i}{(x+i)! + (2i)!}$$

6. Контрольные вопросы

1. Что такое цикл?
2. Какой вид имеет оператор цикла со счетчиком на языке Си?
3. Может ли цикл со счётчиком быть заменён циклом с условием?
4. Как применяются циклы для вычисления суммы числовых рядов?

Лабораторная работа №12. Использование рекурсии

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки использования рекурсивных функций для решения задач.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Функция называется **рекурсивной**, если во время ее обработки возникает ее повторный вызов, либо непосредственно, либо косвенно, путем цепочки вызовов других функций [3].

При **прямой (простой, непосредственной)** рекурсии рекурсивный вызов делается непосредственно из тела самой рекурсивно вызываемой функции.

```
int a( )
{
    ... a( ) ... ;
}
```

Косвенной (сложной, опосредованной) рекурсией называется рекурсия, осуществляющая рекурсивный вызов функции посредством цепочки вызовов других функций. Все функции, входящие в цепочку, также считаются рекурсивными.

```
a( ) { .....b( ) ..... }
b( ) { .....c( ) ..... }
c( ) { .....a( ) ..... }
```

Все функции a, b, c являются рекурсивными, так как при вызове одной из них, осуществляется вызов других и самой себя.

Процесс рекурсивных вызовов называется **рекурсивным спуском**, а процесс возврата из них – **рекурсивным возвратом**.

Глубиной рекурсии называется максимальное число вложенных рекурсивных вызовов.

Количество вложенных рекурсивных вызовов в данный момент выполнения программы называется **текущим уровнем рекурсии** [3].

3. Общая часть лабораторной работы

Задача 1. Вычисление факториала числа

Задача. Написать программу для рекурсивного вычисления факториала.

Решение. Напишем рекурсивную функцию для вычисления факториала. Заметим, что $A! = (A-1)! * A$. При этом факториалы нуля и единицы равны одному.

Таким образом, рекурсивная функция будет возвращать значение 1 для нуля и единицы, а для всех остальных чисел будет вызывать функцию вычисления факториала для уменьшенного на 1 числа и возвращать полученное значение на это число.

Пример функции вычисления факториала:

```
int factorial(int n)
{
    if (n == 0 || n == 1) return 1;
    return n * factorial(n - 1);
}
```

В функции `main()` будет считываться значение числа, для которого требуется вычислить факториал, после чего вызывается ранее описанная функция и выводится результат.

```
int main ()
{
    int n;
    printf("n: ");
    scanf_s("%d", &n);
    if (n >= 0)
        printf("%d! = %d\n", n, factorial(n));
    else
        printf("Error. n must be >= 0\n");
    _getch();
    return 0;
}
```

Так как значение факториала не определено для отрицательных чисел, в программе производится проверка на корректность ввода числа `n`.

Полный листинг программы представлен на рис.1

```

#include <stdio.h>
int factorial(int n)
{
    if (n == 0 || n == 1) return 1;
    return n * factorial(n - 1);
}
int main ()
{
    int n;
    printf("n: ");
    scanf("%d", &n);
    if (n >= 0)
        printf("%d! = %d\n", n, factorial(n));
    else
        printf("Error. n must be >= 0\n");
    return 0;
}

```

Рис.1. Листинг программы

Консольное окно с результатами выполнения программы представлено на рис.2.

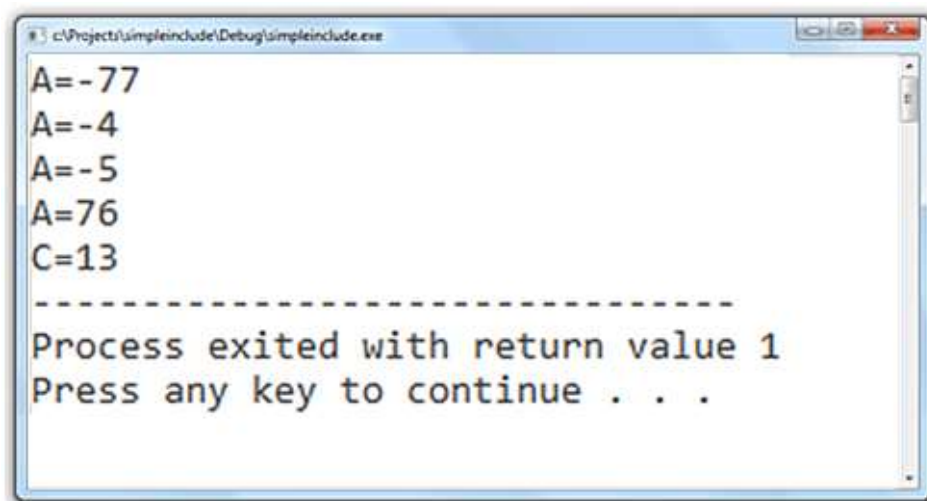


Рис.2. Окно с результатами выполнения программы

Задача 2. Текстовая задача на составление рекурсии

Задача. На первый день рождения мальчику подарили 1 доллар. Каждый следующий год сумма подарка будет удваиваться. Также будет добавляться еще несколько долларов (столько, сколько лет исполнилось). Определить, к какому дню рождения сумма подарка превысит заданную сумму.

Решение. Для решения задачи создадим рекурсивную функцию. Будем ей передавать сумму подарка и номер дня рождения текущего года, а также интересующую сумму (которую нужно превысить). Возвратить функция должна искомый номер дня рождения (когда сумма подарка превысит заданный порог).

После получения управления функция проверяет, превышен ли порог. Если да, то возвращает текущий возраст. В противном случае функция возвратит то значение, которое будет рассчитано ею самой при дополнительном вызове. Перед рекурсивным вызовом рассчитывается новая сумма подарка и новый возраст мальчика. Функция может быть реализована следующим кодом:

```
int money_count(int p, int s, int a)
{
    if (p > s) return a;
    a++;
    p = p * 2 + a;
    return money_count(p, s, a);
}
```

В главной функции программы с клавиатуры прочитаем сумму, которую необходимо превысить, а затем вызовем описанную функцию с аргументами, соответствующими первому дню рождения. Возвращённое функцией значение можно вывести для пользователя на экран:

```
printf_s("Введите интересующую сумму подарка: ");
scanf_s("%d", &s);
printf("Подарок превысит сумму в %d долларов на %d-й день
    рождения \n", s, money_count(1, s, 1));
```

На этом решение задачи закончено.

Полный листинг программы представлен на рис.3.

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить две программы на языке Си. Первая программа должна решать с помощью рекурсии задачу согласно варианту из списка задач для самостоятельного решения. ***В ней необходимо предусмотреть вывод на экран информации о текущем уровне рекурсии и текущих значениях переменных при каждом рекурсивном вызове.*** Вторая программа должна решать эту же задачу без использования рекурсии. Каждая программа должна запросить у

пользователя все необходимые исходные данные и вывести результат решения задачи на экран.

Программы должны быть снабжены комментариями и отформатированы.

```
#include <stdio.h>
int money_count(int p, int s, int a)
{
    if (p > s) return a;
    a++;
    p = p * 2 + a;
    return money_count(p, s, a);
}

int main ()
{
    int s;
    printf("Введите интересующую сумму подарка: ");
    scanf("%d", &s);
    printf("Подарок превысит сумму в %d долларов на %d-й день рождения \n", s,
    money_count(1, s, 1));
}
```

Рис.3. Листинг программы

5. Задачи для самостоятельного решения

1. Садовник в первый день посадил 2 розы, во второй день – 4 розы. В каждый следующий день количество посаженных роз должно быть равно сумме посаженных роз за два предыдущих дня минус 2 розы. Сколько роз должен посадить садовник на 13 день?

2. На каждом следующем дне рождения Винни-Пух съедает столько же пищи что и на двух предыдущих. На двух первых днях рождения у Пятачка и Кролика он съел по 100 г пищи. Определить, сколько килограммов пищи Винни-Пух съест на пятнадцатом дне рождения.

3. Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый день он увеличивал дневную норму на 10% от нормы предыдущего дня. Какой суммарный путь он пробежит за 7 дней?

4. Мачеха приказала Золушке перебрать мешок зерна (40 кг). Начале в 6 часов вечера, Золушка каждый час перебирала на 15% больше, чем в предыдущий час. Успеет ли она на бал (и сколько часов там пробудет, если успеет), если бал заканчивается в 2 часа ночи, а с 6 до 7 часов Золушка перебрала 6 кг зерна?

5. Ежедневно Незнайка учит половину от суммы выученных за два предыдущих дня иностранных слов и еще два слова. Знайка считает, что силы Незнайки иссякнут, когда нужно будет выучить 50 слов в день. Определить, через сколько дней иссякнут силы у Незнайки, если в первые два дня он выучил по одному слову.

6. Татьяна Ларина, читая очередной французский роман, подсчитала сумму номеров прочитанных страниц и получила значение 2006. Определить номер прочитанной страницы.

7. Царевна-лягушка съедает ежедневно на 20% комаров больше, чем в предыдущий день, и еще два комара. Определить, через сколько дней количество съеденных комаров превысит 100, если в первый день было съедено 12 комаров.

8. Одноклеточная амеба каждые три часа делится на две клетки. Определить, сколько клеток будет через 3, 6, 9, ... 24 часа.

9. С 25 лет штангист каждые полгода увеличивает вес штанги на 10% от предыдущего веса, начав со 120 кг. Определить, во сколько лет он может стать чемпионом мира, подняв свыше 210 кг.

10. Каждый раз, когда Буратино солжет, его нос вырастает на $\frac{1}{20}$ от предыдущей длины. Мудрый сверчок предупредил, что с носом длиной более 45 см его ждут большие неприятности. Определить, через сколько дней это может произойти, если Буратино лжет через день, а первоначально длина носа была 10 см.

11. Долг Саида хану составляет 900 золотых. За хорошую работу хан каждый год сокращает долг в 3 раза, но добавляет количество золотых, равное удвоенному числу пройденных лет. Последние 20 золотых хан готов простить. Определить, через сколько лет Саид отработает свой долг.

5. Контрольные вопросы

1. Что такое рекурсия?
2. Какие виды рекурсии существуют?
3. Что такое прямая рекурсия?
4. Что такое косвенная рекурсия?
5. Что такое глубина рекурсии?
6. В чем заключаются рекурсивный спуск и рекурсивный подъем?

ГЛАВА 3. МАССИВЫ, СТРУКТУРЫ, ФАЙЛЫ

Лабораторная работа №13. Массивы

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки работы с одномерными массивами.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Понятие массива

При решении задач с большим количеством данных одинакового типа использование переменных с различными именами, не упорядоченных по адресам памяти, затрудняет программирование. В подобных случаях в языке Си используют объекты, называемые массивами [4].

Массив – это совокупность элементов данных одинакового типа, лежащих в непрерывной области памяти и объединённых одним именем.

Элемент массива – значение, хранящееся в определенной ячейке массива.

Каждый элемент массива характеризуется тремя величинами:

- **адресом элемента** – адресом начальной ячейки памяти, в которой расположен этот элемент;
- **индексом элемента** – порядковым номером элемента в массиве;
- **значением элемента.**

Адрес массива – адрес начального элемента массива. В языке Си имя массива обозначает адрес начального элемента массива.

Для доступа к элементам массива используют **операцию индексации**. В языке Си индексация представляет собой бинарную операцию, записываемую с помощью квадратных скобок «[]». Первым операндом выступает адрес первого элемента, и обычно он представлен именем массива. Вторым операндом выступает индекс элемента, доступ к которому необходимо получить.

Размер массива – количество элементов массива.

Размер элемента – количество байт, занимаемых одним элементом массива.

Объявление и инициализация массивов

Для объявления массива в языке Си используется следующий синтаксис:

```
тип имя[размер_массива] = {инициализация};
```

Инициализация представляет собой набор начальных значений элементов массива, разделенных запятыми [1]. Если инициализирующих значений меньше, чем элементов в массиве, то остальным элементам присваивается значение 0.

```
// массив a из 10 целых чисел, инициализированный
// натуральными числами от 1 до 10
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
// массив a из 10 элементов, инициализированных нулями
int a[10] = {0};
```

Если массив проинициализирован, то количество элементов массива в квадратных скобках при его описании может быть не указано, и компилятор установит его равным количеству инициализирующих значений.

```
// Размер будет автоматически установлен равным 9
int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

При обращении к элементам массива индекс требуемого элемента указывается в квадратных скобках [].

Однако часто требуется задавать значения элементов массива при выполнении программы. При этом используется объявление массива без инициализации. В таком случае указание количества элементов в квадратных скобках обязательно.

```
int a[10];
```

Для задания начальных значений элементов массива очень часто используется цикл с параметром [2]. Аналогично обходят массив и для других целей:

```
// Ввод элементов массива
for(i=0; i<5; i++)
{
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
}
```

```
// Вывод элементов массива
for (i=0; i<5; i++)
{
    printf("%d ", a[i]);
}
```

Передача массива в функцию

Обработку массивов удобно организовывать с помощью специальных функций. Для обработки массива в качестве аргументов функции необходимо передать

- адрес массива,
- размер массива.

Исключение составляют функции обработки строк, в которые достаточно передать только адрес.

При передаче переменные в качестве аргументов функции данные передаются как копии. Это означает, что если внутри функции произойдет изменение значения параметра, то это никак не повлияет на его значение внутри вызывающей функции.

Если в функцию передается адрес переменной (или адрес массива), то все операции, выполняемые в функции с данными, находящимися в пределах видимости указанного адреса, производятся над оригиналом данных, поэтому исходный массив (или значение переменной) может быть изменено вызываемой функцией.

Пример 1. Функция вывода элементов массива.

```
// Выводит массив x из n элементов
void f(int x[], int n)
{
    for(int i=0; i<n; i++)
        printf("%d ", x[i]);
}
```

Заполнение массива случайными числами

Заполнить массив случайными числами в Си не трудно. В этом случае нам потребуются функции `rand()`; и `srand (m)`; из библиотеки `stdlib.h`.

Пусть имеется одномерный массив из десяти целочисленных элементов, тогда заполнение такого массива случайными числами будет выглядеть вот так:

```
for ( int i = 0; i < 10; i ++)  
{  
    Arr [i] = rand ()%100;  
}
```

Этот цикл заполняет массив *Arr* случайными числами от 0 до 100. При этом число 100 не включается в диапазон.

Использование функции *srand* с аргументом *time(NULL)* позволяет получить при каждом запуске программы различные числа, так как устанавливает текущее время в качестве начального значения для функции *rand*. В противном случае получим одинаковые значения элементов массива при каждом запуске программы.

Для использования функций в заголовок требуется подключить следующие библиотеки:

```
#include <stdlib.h>  
#include <ctime>
```

Пример 2. Функция генерации случайного числа, принадлежащего заданному диапазону.

```
// Возвращает случайное число из диапазона  
// от r_min до r_max включительно  
int rrand(int r_min, int r_max)  
{  
    return rand() % (r_max - r_min + 1) + r_min;  
}
```

3. Общая часть лабораторной работы

Задача 1. Найти максимальный элемент введенного с клавиатуры массива.

Решение. Вначале описать массив и запросить количество элементов в массиве (не больше длины массива, указанной при объявлении массива). Затем прочитать значения элементов массива с клавиатуры, после чего осуществить поиск максимального элемента массива.

Для начала переменной *max*, которая будет хранить искомый максимум, присваиваем значение первого элемента массива.

```
max = a[0];
```

После чего проходим по массиву, сравнивая каждый элемент с текущим значением переменной *max*. Если текущий элемент массива окажется больше *max*, то его значение присваиваем переменной *max*.

После завершения обхода массива переменная *max* будет содержать значение максимального элемента массива.

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

int main()
{
    int a[100];
    int n;
    int max;

    printf("n= ");
    scanf("%d", &n);

    printf("massiv:\n");
    for (int i = 0; i<n; i++)
    {
        printf("a[%d]= ", i);
        scanf("%d", &a[i]);
    }

    max = a[0];
    for (int i = 1; i<n; i++)
    {
        if (a[i] > max)
        {
            max = a[i];
        }
    }

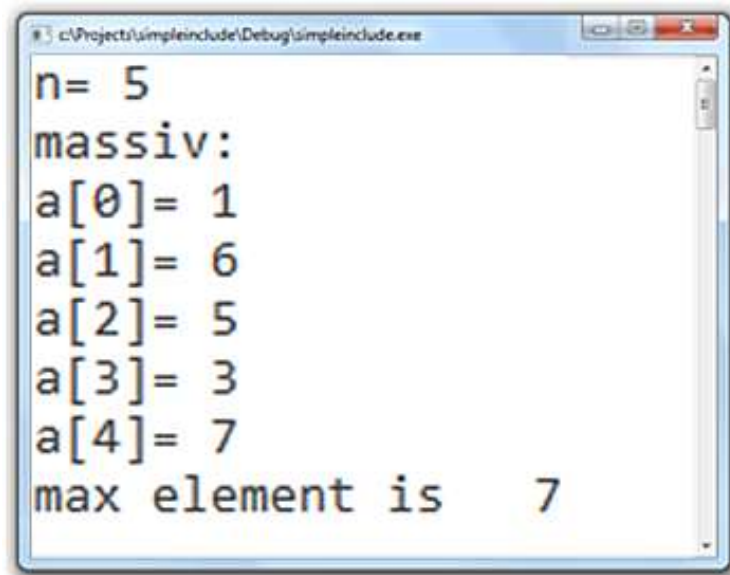
    printf("max element is   %d", max);

    return 1;
}
```

Окно с результатами выполнения программы представлено на рис.1.

Задача 2. Найти максимальный элемент массива, заполненного случайными числами из отрезка $[A, B]$, границы которого задаёт пользователь.

Решение. Воспользуемся функцией *rand*. Чтобы гарантировать выбор случайных чисел из заданного отрезка, используем выражение:
 $a[i] = \text{rand}() \% (B - A + 1) + A;$



```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
n= 5
massiv:
a[0]= 1
a[1]= 6
a[2]= 5
a[3]= 3
a[4]= 7
max element is 7

```

Рис.1. Результат выполнения программы

Чтобы при каждом запуске программы в массив заносились различные случайные числа, используем функцию *srand*, подключив заголовочные файлы `<stdlib.h>` и `<ctime>`.

Получим следующий код программы:

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctime>

int main()
{
    int a[100];
    int n;
    int max; int A,B;

    srand(time(NULL));

    printf("n= ");
    scanf("%d", &n);
    printf("A= ");
    scanf("%d", &A);
    printf("B= ");
    scanf("%d", &B);

    printf("massiv:\n");
    for (int i = 0; i<n; i++)
    {
        a[i]= rand() % (B - A + 1) + A;
    }
}

```

```
        printf("a[%d]= %d\n", i, a[i]);
    }

    max = a[0];
    for (int i = 1; i < n; i++)
    {
        if (a[i] > max)
        {
            max = a[i];
        }
    }
    printf("max element is %d", max);

    return 1;
}
```

Вывод элементов массива в приведённом примере происходит в том же цикле, где и заполняется значениями массив.

Окно с результатами выполнения программы представлено на рис.2.

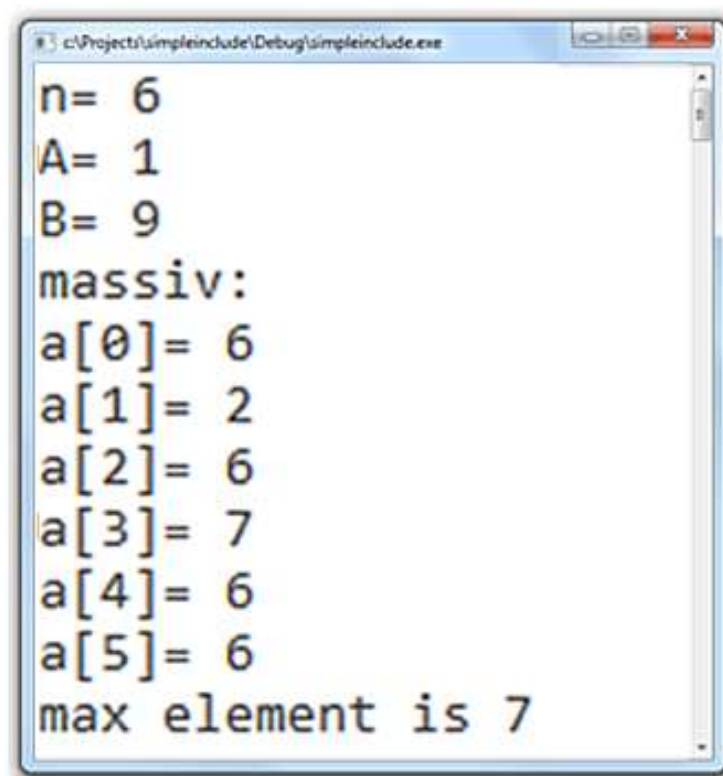


Рис.2. Результат выполнения программы

Задача 3. Заданы два массива. Вычислить сумму их минимальных элементов.

Решение. Опишем функцию поиска минимального элемента в массиве:

```
int min (int a[], int n)
{
    int min = a[0];
    for (int i = 1; i<n; i++)
        if (a[i] <min)
            min = a[i];
    return min;
}
```

В теле функции main() определим два массива, заполним их случайными числами и передадим в ранее описанную функцию. Суммируем возвращенные функцией значения.

```
int a[100];
int b[100];
int n1, n2;
int sum=0;

srand (time (NULL));

printf("n1= ");
scanf("%d", &n1);
printf("n2= ");
scanf("%d", &n2);

int A,B;
printf("A= ");
scanf("%d", &A);
printf("B= ");
scanf("%d", &B);

printf("massiv1:\n");
for (int i = 0; i<n1; i++)
{
    a[i]= rand() % (B - A + 1) + A;
    printf("a[%d]= %d\n", i,a[i]);
}

printf("massiv2:\n");
for (int i = 0; i<n2; i++)
{
```

```

    b[i]= rand() % (B - A + 1) + A;
    printf("b[%d]= %d\n", i,b[i]);
}

```

```

sum=min(a,n1)+(b,n2);
printf("sum=%d", sum);

```

Полный листинг программы представлен на рис.3.

```

#include <stdio.h>
#include <stdlib.h>
#include <ctime>
int min (int *a, int n)
{
    int min = a[0];
    for (int i = 1; i<n; i++)
        if (a[i] <min) min = a[i];
    return min;
}
int main() {
    int a[100]; int b[100];
    int n1,n2;
    int sum=0;
    srand (time (NULL));
    |
    printf("n1= "); scanf("%d", &n1);
    printf("n2= "); scanf("%d", &n2);

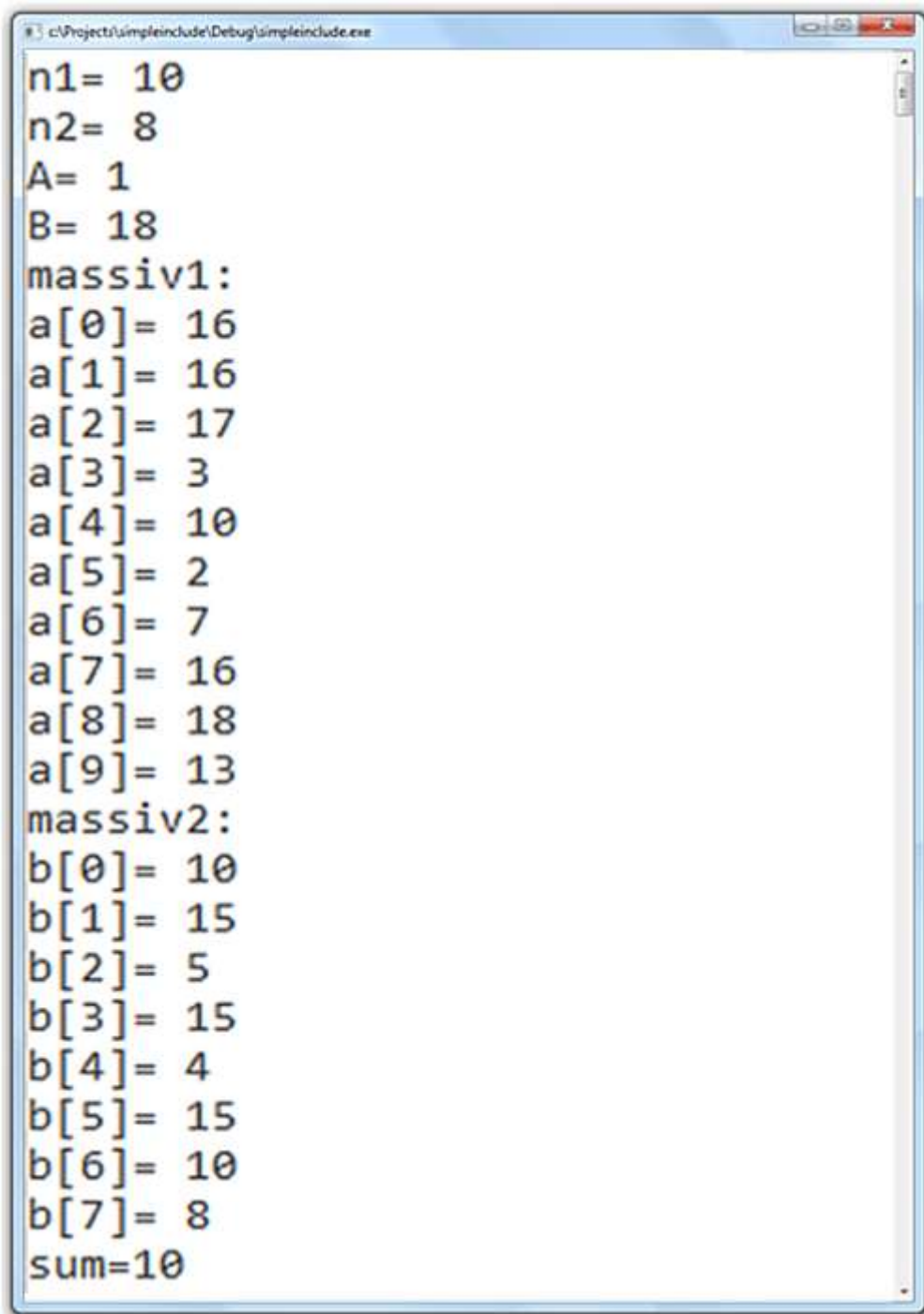
    int A,B;
    printf("A= "); scanf("%d", &A);
    printf("B= "); scanf("%d", &B);

    printf("massiv1:\n");
    for (int i = 0; i<n1; i++) {
        a[i]= rand() % (B - A + 1) + A;
        printf("a[%d]= %d\n", i,a[i]);
    }
    printf("massiv2:\n");
    for (int i = 0; i<n2; i++) {
        b[i]= rand() % (B - A + 1) + A;
        printf("b[%d]= %d\n", i,b[i]);
    }
    sum=min(a,n1)+(b,n2);
    printf("sum=%d", sum);
    return 1; }

```

Рис.3. Листинг программы

Результат выполнения программы представлен на рис.4.



```
c:\Projects\simpleinclude\Debug\simpleinclude.exe
n1= 10
n2= 8
A= 1
B= 18
massiv1:
a[0]= 16
a[1]= 16
a[2]= 17
a[3]= 3
a[4]= 10
a[5]= 2
a[6]= 7
a[7]= 16
a[8]= 18
a[9]= 13
massiv2:
b[0]= 10
b[1]= 15
b[2]= 5
b[3]= 15
b[4]= 4
b[5]= 15
b[6]= 10
b[7]= 8
sum=10
```

Рис.4. Результат выполнения программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить программу на языке Си.

Программа должна предоставить пользователю возможность выбора одного из двух способов заполнения массива: с клавиатуры или случайными числами из отрезка $[A, B]$, границы которого задаёт

пользователь. Заполнение массива каждым способом, вывод массива на экран и решение задач согласно варианту из списка задач для самостоятельного решения необходимо оформить в виде отдельных подпрограмм. Массивы и их длины передавать функциям с помощью списка аргументов, глобальные данные не использовать.

Программа должна запросить у пользователя все необходимые исходные данные, вывести массив и результат решения задачи на экран. Она должна быть отформатирована и снабжена комментариями.

5. Задачи для самостоятельного решения

1. Вычислить среднее арифметическое элементов массива. Поменять местами элементы, стоящие на четных и нечетных позициях.

2. Вычислить среднее геометрическое элементов массива. Поменять местами элементы, стоящие на четных и нечетных позициях.

3. Вычислить сумму отрицательных элементов массива. Все четные элементы массива разделить на 2.

4. Вычислить сумму положительных элементов массива. Все нечетные элементы массива умножить на 3.

5. Определить количество элементов массива, больших заданного числа. Найти минимальный элемент массива и вычесть его из всех остальных элементов кроме него самого.

6. Определить количество элементов массива, меньших заданного числа. Найти максимальный элемент массива и вычесть его из всех остальных элементов кроме него самого.

7. Вывести индексы всех положительных чисел. Умножить отрицательные элементы массива на 2.

8. Вывести индексы всех отрицательных чисел. Умножить положительные элементы массива на 3.

9. Определить, сколько раз меняется знак в массиве. Умножить все элементы массива на наименьший по модулю отрицательный элемент.

10. Определить, сколько раз меняется знак в массиве. Умножить все элементы массива на наименьший положительный элемент.

11. Определить, содержится ли в массиве ровно одно отрицательное число. Циклически сдвинуть все элементы массива на 1 элемент вправо.

12. Определить, содержится ли в массиве ровно одно положительное число. Циклически сдвинуть все элементы массива на 1 элемент влево.

13. Определить, содержатся ли в массиве хотя бы два отрицательных числа. Если да, прибавить сумму этих чисел ко всем элементам массива. В противном случае оставить массив неизменным.

14. Определить, содержатся ли в массиве хотя бы три положительных числа. Если да, отнять сумму этих чисел от всех элементов массива, в противном случае оставить массив неизменным.

15. Вывести индексы всех чисел, больших суммы соседних чисел. Умножить все элементы массива на наибольший элемент.

16. Вывести индексы всех чисел, меньших суммы соседних чисел. Умножить все элементы массива на наименьший элемент.

17. Определить, упорядочены ли числа в массиве по убыванию. Если нет, заполнить весь массив членами арифметической прогрессии с заданным начальным значением и разностью.

18. Определить, упорядочены ли числа в массиве по возрастанию. Если нет, заполнить весь массив членами геометрической прогрессии с заданным начальным значением и знаменателем.

19. Определить количество элементов массива, принадлежащих указанному пользователем диапазону $[X, Y]$. Вычесть из всех четных элементов массива сумму всех элементов массива.

20. Определить количество элементов массива, выходящих за границы указанного пользователем диапазона $[X, Y]$. Вычесть из всех нечетных элементов массива сумму всех элементов массива.

6. Контрольные вопросы

1. Что такое массив?
2. Каким образом производится доступ к элементу массива?
3. Как ввести массив с клавиатуры?
4. Как заполнить массив случайными числами?
5. Как вывести массив на экран?

Лабораторная работа №14. Текстовые файлы

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки работы с текстовыми файлами.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

В стандартной библиотеке языка Си предусмотрены функции для работы с файлами. Для работы с ними необходимо подключить заголовочный файл `<stdio.h>`.

Работа с файлами складывается из трех шагов [1]:

- открыть файл;
- работать с файлом;
- закрыть файл.

Рассмотрим эти шаги подробно.

1. Файл открывается. Это означает, что программа «захватывает» заданный по имени файл, сообщает операционной системе, что далее она будет с ним работать. Этот шаг необходим, чтобы не возникало конфликтов, когда несколько программ одновременно хотят записывать информацию в один и тот же файл. Однако считывать данные из файла, очевидно, допустимо одновременно множеством программ, поэтому в операции открытия файла обычно уточняется, что файл открывается «на чтение» (считывание информации, которая не меняется) либо «на запись» (данные в файле модифицируются).

Функция открытия файла имеет следующий прототип (здесь и далее пропущены некоторые платформенные особенности прототипов):

```
FILE *fopen (const char *filename, const char *opentype);
```

Эта функция передаёт запрос на открытие файла операционной системе. Операционная система при открытии файла для некоторой программы сообщает ей (программе) идентификатор (как правило, целое число), которое идентифицирует в программе в дальнейшем открытый файл. Этот идентификатор запоминается в переменной; обычно такая переменная называется файловой переменной. Файловая переменная в стандартной библиотеке языка Си представляет собой структуру типа `FILE` и кроме идентификатора

файла хранит также дополнительные данные о текущем состоянии работы с файлом. Функция *fopen* возвращает указатель на динамически созданную структуру типа *FILE*.

Параметров у функции *fopen* два. Первый – это путь к файлу (строка), второй – способ открытия файла. Способы открытия файлов представлены в табл.1 [1].

Таблица 1

Способы открытия файлов

Режим			Описание	Начинает с..
r	rb		Открывает для чтения	начала
w	wb		Открывает для записи (создаёт файл в случае его отсутствия). Удаляет содержимое и перезаписывает файл	начала
a	ab		Открывает для добавления (создаёт файл в случае его отсутствия)	конца
r+	rb+	r+b	Открывает для чтения и записи	начала
w+	wb+	w+b	Открывает для чтения и записи. Удаляет содержимое и перезаписывает файл	начала
a+	ab+	a+b	Открывает для чтения и записи (добавляет в случае существования файла)	конца

2. Ведется работа с файлом. Из него данные либо считываются, либо в него записываются. Приёмы работы с файлом зависят от типа контента файла и будут рассмотрены далее.

3. Файл закрывается. После этой операции он снова доступен другим программам для обработки.

Функция закрытия файла имеет следующий прототип:

```
int fclose (FILE *stream);
```

При вызове функции *fclose* необходимо передать указатель на структуру типа *FILE*, ранее возвращённый при открытии файла. Вызов функции *fclose* приводит к закрытию потока данных, на который указывает аргумент *stream*, сбросу остаточного содержимого буфера данных в файл (если в буфере оставались данные), освобождению памяти, выделенной под буфера чтения и записи (если буфера создавались), и закрытию файла, связанного с потоком данных. При успешной работе функция возвращает 0. Если

во время работы функции возникла ошибка, то она возвращает значение *EOF*, равное -1, при этом в соответствующую структуру типа *FILE* будет записан код ошибки, которую затем можно определить, используя функцию *ferror*.

Вот список некоторых дополнительных полезных функций для работы с файлами:

```
//проверка достижения конца файла
int feof (FILE *stream);
//позиционирование
int fseek(FILE *stream, long offset, int origin);
// возвращает текущий код ошибки
int ferror(FILE *stream);
// сбрасывает признак ошибки в 0
void clearerr (FILE* stream);
// записывает в pposition текущую позицию курсора в файле
int fgetpos(FILE* stream, fpos_t* pposition);
// устанавливает курсор в файле в позицию из pposition
int fsetpos (FILE*stream, const fpos_t* pposition);
```

Для работы с текстовыми файлами предназначены функции, аналогичные функциям для работы со стандартным устройством ввода/вывода, и отличаются от них наличием дополнительного аргумента: указателя на структуру типа *FILE*. Вот список основных функций для работы с текстовыми файлами:

```
int fprintf (FILE *stream, const char *template, ...);
int fscanf (FILE *stream, const char *template, ...);
int fgetc (FILE*);
char*fgets (char*, int, FILE*);
int fputc (int, FILE*);
int fputs (const char*, FILE*);
```

3. Общая часть лабораторной работы

Задача 1. Размер массива и его элементы заданы в текстовом файле. Организовать ввод массива из файла и запись его размера и элементов в другой файл.

Решение. Файл input.txt открывается в режиме чтения. В начале из него считывается количество элементов массива, затем сами значения элементов. После завершения работы с файлом, он закрывается.

```
int d[100];
```

```

int i,N;
FILE *ft;
ft=fopen("input.txt","rt");
fscanf(ft,"%d",&N);
for(i=0;i<N;i++)
fscanf(ft,"%d",&d[i]);
fclose(ft);

```

Чтобы записать размер и элементы массива в другой файл удобно использовать следующий код:

```

FILE* f=fopen("output.txt","w+");
fprintf(f,"%d\n",N);
for(int i=0; i<N; i++)
    fprintf(f,"%d ",d[i]);
fclose(f);

```

Задача 2. Размер (в пределах от 3 до 20) и элементы массива вводятся из файла или задаются пользователем. Найти минимальный элемент массива.

Решение. Позволим пользователю выбирать способ ввода элементов массива. Организуем работу с массивом с помощью пользовательских функций.

Функции ввода массива с клавиатуры и вывода массива на экран представлены на рис.1. На вход функции получают указатель на массив и количество элементов массива.

```

void input_key(int *a,int n) {

    printf("input array:\n");
    for(int i=0; i<n; i++) {
        printf("a[%d]=",i);
        scanf("%d",&a[i]);
    }
}

void print_arr(int *a,int n) {
    for(int i=0; i<n; i++) printf("%d ",a[i]);
}

```

Рис.1. Функции ввода массива с клавиатуры и вывода его на экран

Функция заполнения массива случайными числами показана на рис.2. Границы интервала случайных чисел запрашиваются у пользователя. Предусматривается обработка некорректного ввода. После заполнения массива случайными числами он выводится на экран.

```
void input_rand(int *a,int n) {

    srand (time (NULL));
    int c,b;
    do {

        printf("c=");
        scanf("%d",&c);
        printf("b=");
        scanf("%d",&b);
    } while (c>b);
    for(int i=0; i<n; i++) {
        a[i]= rand() % (b - c + 1) + c;
    }

    print_arr(a,n);
}
```

Рис.2. Функция заполнения массива случайными числами

На рис. 3 показана функция поиска минимального элемента в массиве.

Функции ввода массива из файла и записи его в файл представлены на рис.4.

```
int min_in_arr(int *a,int n) {
    int min=a[0];
    for(int i=0; i<n; i++) if (a[i]<min)min=a[i];
    return min;
}
```

Рис.3. Функция поиска минимального элемента в массиве

```

void input_file(int *a,int n) {
    FILE *ft;
    ft=fopen("f.txt","rt");
    fscanf(ft,"%d",&n);
    for(int i=0; i<n; i++)
        fscanf(ft,"%d",&a[i]);
    fclose(ft);
    print_arr(a,n);
}

void output_file(int *a,int n) {
    FILE* f=fopen("f.txt","w+");
    fprintf(f,"%d \n",n);
    for(int i=0; i<n; i++)
        fprintf(f,"%d\n ",a[i]);
    fclose(f);
}

```

Рис.4. Функции для работы с файлами

На рис. 5 представлена функция main(). В ней объявлен массив из ста элементов. Количество элементов, с которыми желает работать пользователь, запрашивается с осуществлением проверки корректности ввода.

Выбор варианта ввода массива производится с помощью оператора множественного выбора.

После ввода массива осуществляется поиск минимального элемента. Полученный массив записывается в файл.

Результаты выполнения программ представлены на рис. 6-8.

Как видно, при различных способах ввода программа работает корректно. Организация выбора пользователя является простейшим примером пользовательского меню.

```

int main() {
    int a[100];
    int n;
    int ch;
    do {
        printf("n=");
        scanf("%d",&n);
    } while ((n<0)|| (n>=100));

    printf("input array:\n 1-from keyboard, 2-random,3-from file");
    scanf("%d",&ch);
    switch (ch) {
        case 1:
            input_key(a,n);
            break;
        case 2:
            input_rand(a,n);
            break;
        case 3:
            input_file(a,n);
            break;
        default:
            printf("error");
            exit(0);
    };

    printf("\n min=%d",min_in_arr(a, n) );

    output_file(a, n);
    return 1;
}

```

Рис.5. Функция main()

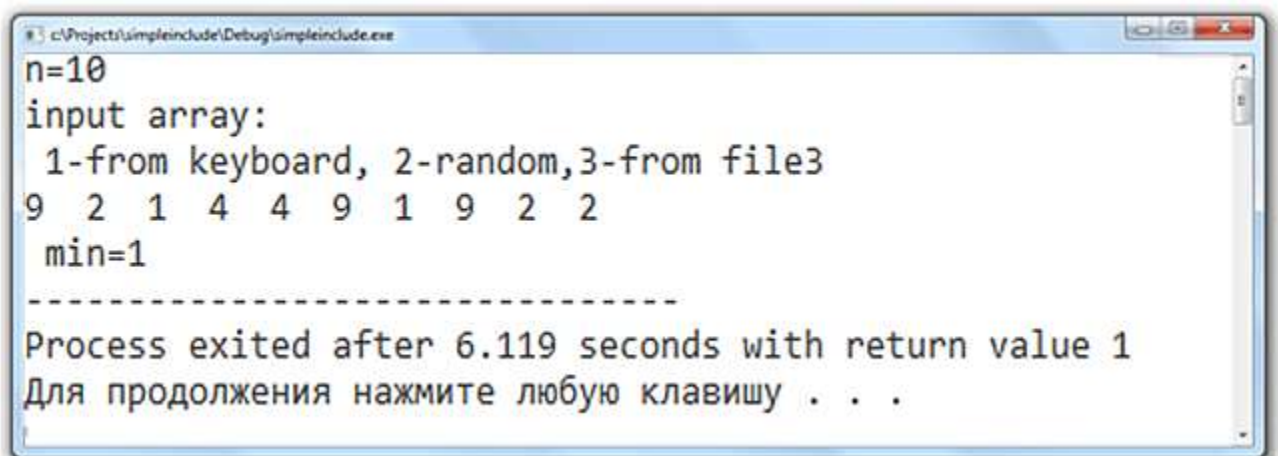
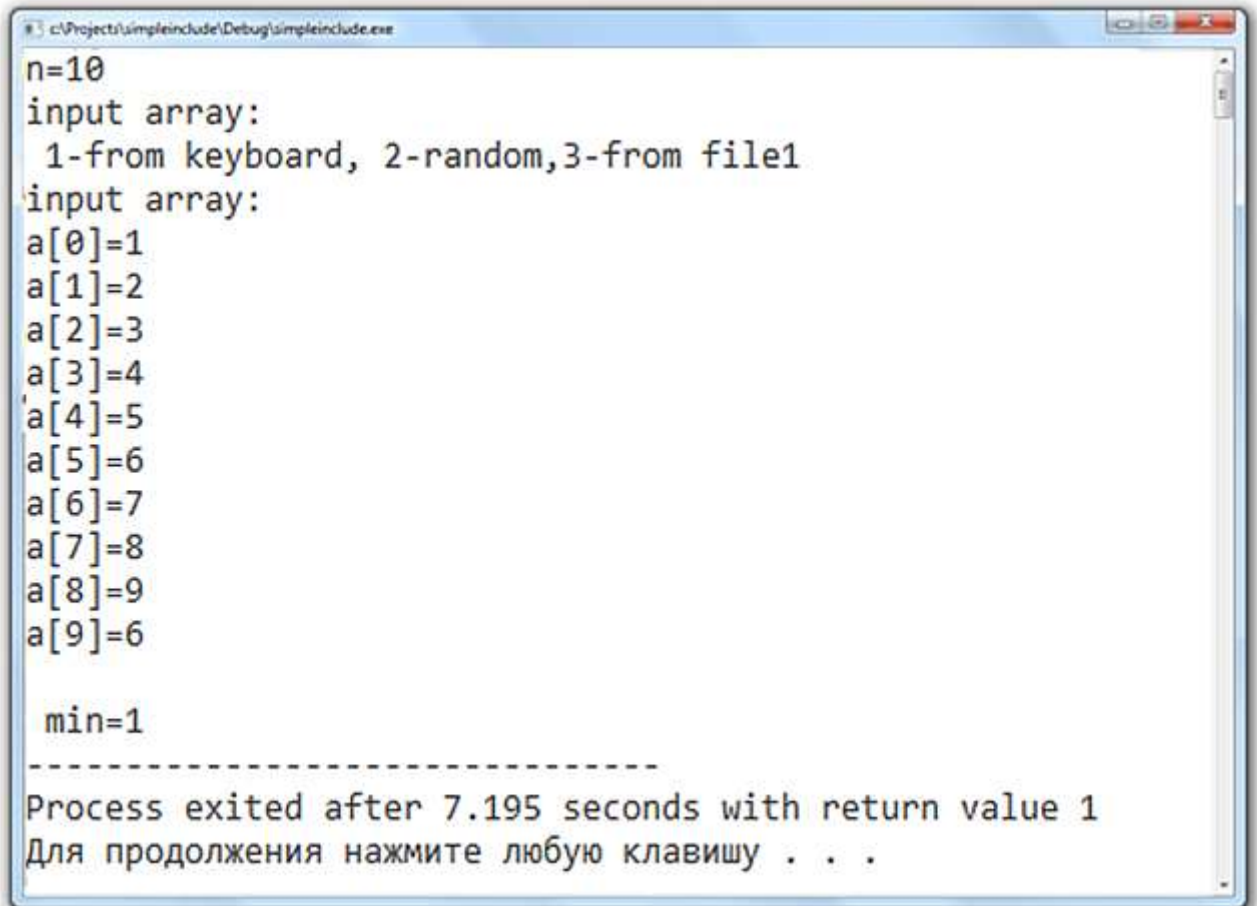


Рис.6. Результат выполнения программы (ввод данных из файла)



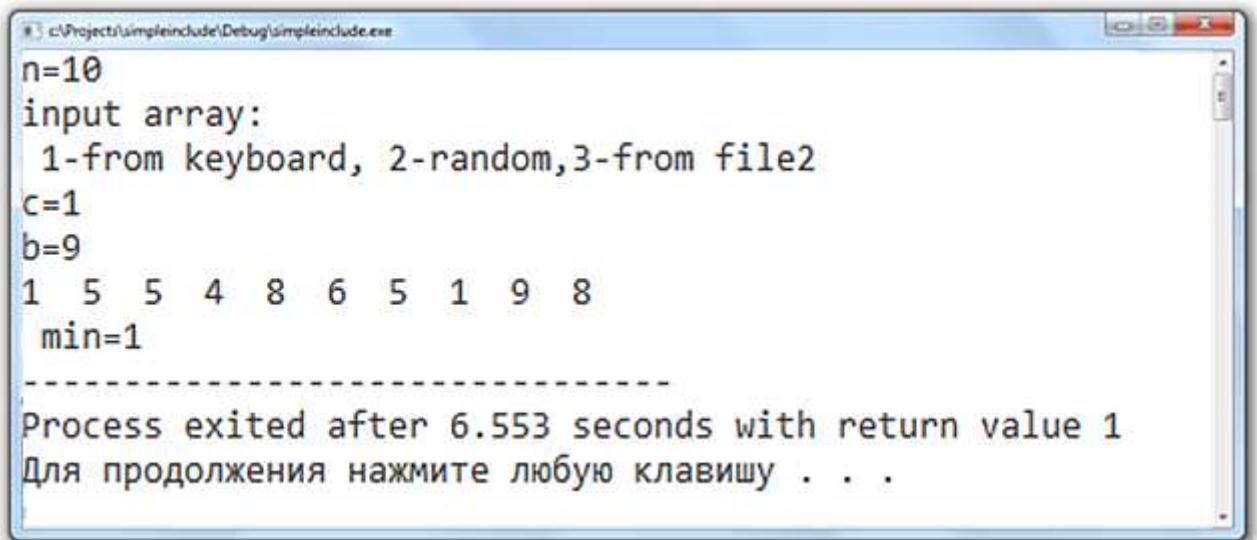
```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
n=10
input array:
 1-from keyboard, 2-random, 3-from file1
input array:
a[0]=1
a[1]=2
a[2]=3
a[3]=4
a[4]=5
a[5]=6
a[6]=7
a[7]=8
a[8]=9
a[9]=6

min=1
-----
Process exited after 7.195 seconds with return value 1
Для продолжения нажмите любую клавишу . . .

```

Рис.7. Результат выполнения программы (ввод данных с клавиатуры)



```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
n=10
input array:
 1-from keyboard, 2-random, 3-from file2
c=1
b=9
1 5 5 4 8 6 5 1 9 8
min=1
-----
Process exited after 6.553 seconds with return value 1
Для продолжения нажмите любую клавишу . . .

```

Рис.8. Результат выполнения программы
(заполнение массива случайными числами)

Полный код программы имеет вид

```

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctime>

```

```

void input_key(int *a,int n)
{
    printf("input array:\n");
    for(int i=0; i<n; i++)
    {
        printf("a[%d]=",i);
        scanf("%d",&a[i]);
    }
}

void print_arr(int *a,int n)
{
    for(int i=0; i<n; i++)
        printf("%d  ",a[i]);
}

void input_rand(int *a,int n)
{
    srand (time (NULL));
    int c,b;
    do
    {
        printf("c=");
        scanf("%d",&c);
        printf("b=");
        scanf("%d",&b);
    } while (c>b);
    for(int i=0; i<n; i++)
    {
        a[i]= rand() % (b - c + 1) + c;
    }

    print_arr(a,n);
}

void input_file(int *a,int n)
{
    FILE *ft;
    ft=fopen("f.txt","rt");
    fscanf(ft,"%d",&n);
    for(int i=0; i<n; i++)
        fscanf(ft,"%d",&a[i]);
}

```



```

        fclose(ft);
        print_arr(a,n);
    }

void output_file(int *a,int n)
{
    FILE* f=fopen("f.txt","w+");
    fprintf(f,"%d \n",n);
    for(int i=0; i<n; i++)
        fprintf(f,"%d\n ",a[i]);
    fclose(f);
}

int min_in_arr(int *a,int n)
{
    int min=a[0];
    for(int i=0; i<n; i++)
        if (a[i]<min)
            min=a[i];
    return min;
}

int main()
{
    int a[100];
    int n;
    int ch;
    do
    {
        printf("n=");
        scanf("%d",&n);
    } while ((n<0)|| (n>=100));
    printf("input array:\n 1-from keyboard, 2-random, 3-
from file");
    scanf("%d",&ch);
    switch (ch)
    {
    case 1:
        input_key(a,n);
        break;
    case 2:
        input_rand(a,n);
        break;
    case 3:

```

```

        input_file(a,n);
        break;
default:
    printf("error");
    exit(1);
}
printf("\n min=%d",min_in_arr(a, n) );
output_file(a, n);
return 0;
}

```

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить программу на языке Си.

За основу берется выполненное задание предыдущей лабораторной работы. К реализованному функционалу добавить следующие возможности: считывание массива из файла; сохранение массива в другой файл после выполненных с ним преобразований (если такой файл есть, его предыдущее содержимое перезаписывается).

Заполнение массива каждым способом, вывод массива на экран, сохранение в файл необходимо оформить в виде отдельных подпрограмм. Массивы и их длины передавать функциям необходимо через список аргументов, глобальные данные не использовать.

Программа должна запросить у пользователя все необходимые исходные данные и вывести массив и результат решения задачи на экран. Она должна быть отформатирована и снабжена комментариями.

5. Контрольные вопросы

1. Что такое файл?
2. Каков общий алгоритм работы с файлом?
3. Какие режимы открытия файла существуют?
4. Какие существуют функции для работы с файлами?
5. Чем отличаются способы открытия файла «w» и «w+»?

Лабораторная работа №15. Массивы структур

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки работы со структурами и массивами структур.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Структура – это совокупность элементов данных различного типа, лежащих в непрерывной области памяти и объединенных одним именем. Определение структуры приводит к образованию шаблона, используемого для создания **объектов (экземпляров)** структуры. Переменные, образующие структуру, называются **членами (элементами, полями)** структуры [2].

Обычно все члены структуры связаны друг с другом. Так, информация об имени и адресе, находящаяся в списке рассылки, обычно представляется в виде структуры. Следующий фрагмент кода объявляет шаблон структуры, определяющий имя и адрес. Ключевое слово ***struct*** сообщает компилятору об объявлении структуры [2]:

```
struct addr
{
    char name[30];
    char street [40]; char city[20];
    char state[3];
    unsigned long int zip;
};
```

Определение завершается точкой с запятой, так как определение структуры – это оператор.

Синтаксис определения структуры:

```
struct <имя> {
    <тип1> <поле1>;
    <тип2> <поле2>;
    ...
    <типN> <полеN>;
};
```

После того как определена структура, можно создавать переменную такого типа с использованием служебного слова ***struct***.

Доступ к полям структуры осуществляется с помощью операции «точка»:

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

struct point_t
{
    int x;
    int y;
};

void main()
{
    struct point_t A;
    float distance;

    A.x = 10;
    A.y = 20;

    distance = sqrt((float) (A.x*A.x + A.y*A.y));

    printf("x = %.3f", distance);
    getch();
}
```

Структуры часто образуют массивы. Чтобы объявить массив структур, вначале необходимо определить структуру (то есть определить агрегатный тип данных), а затем объявить переменную массива этого типа.

```
struct addr addr_list[100];
```

3. Общая часть лабораторной работы

Задача 1. Заданы наименования товаров и их цена. Вывести наименование и стоимость самого дешевого товара.

Решение. Для начала создадим структуру, содержащую информацию о названии товара и его цене (рис.1.).

```

struct tovar {
    char name[30];
    int price;
};

```

Рис.1. Описание структуры

Внутри функции main определим массив элементов структуры:

```
struct tovar tovar_list[100];
```

Запросим у пользователя количество товаров и введем их с клавиатуры внутри цикла (рис 2.).

```

int n;
printf("n="); scanf("%d",&n);

for (int i=0; i<n; i++)
{
    printf("vvedite %d-i tovar:",i);
    printf("name: "); scanf("%s",&tovar_list[i].name);
    printf("price: "); scanf("%d",&tovar_list[i].price);
}

```

Рис.2. Ввод данных

Чтобы найти товар с минимальной ценой, организуем цикл с параметром, в котором найдем индекс товара с наименьшей ценой. После цикла по индексу выведем название соответствующего ему товара (рис.3.).

```

int i_min=0; int min_price=tovar_list[0].price;

for (int i=1;i<n;i++)
    if(tovar_list[i].price<min_price)
    {
        min_price=tovar_list[i].price;
        i_min=i;
    }
printf("samiy desheviy tovar: %s s cenoi %d",
    tovar_list[i_min].name,tovar_list[i_min].price );

```

Рис.3. Поиск индекса товара с наименьшей ценой

Результат выполнения программы представлен на рис.4, полный код программы – на рис.5.

```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
n=5
vvedite 0-i tovar:name: телевизор
price: 27000
vvedite 1-i tovar:name: фен
price: 1200
vvedite 2-i tovar:name: мультиварка
price: 3500
vvedite 3-i tovar:name: чайник
price: 2900
vvedite 4-i tovar:name: утюг
price: 2800
samiy desheviy tovar: фен s cenoi 1200

```

Рис.4. Результат выполнения программы

```

#include <conio.h>
#include <stdio.h>
#include <math.h>

struct tovar {
    char name[30];
    int price;
};

void main() {
    struct tovar tovar_list[100];
    int n;
    printf("n="); scanf("%d",&n);
    for (int i=0; i<n; i++)
    {
        printf("vvedite %d-i tovar:",i);
        printf("name: "); scanf("%s",&tovar_list[i].name);
        printf("price: "); scanf("%d",&tovar_list[i].price);
    }
    int i_min=0; int min_price=tovar_list[0].price;
    for (int i=1;i<n;i++)
        if(tovar_list[i].price<min_price)
        {
            min_price=tovar_list[i].price;
            i_min=i;
        }
    printf("samiy desheviy tovar: %s s cenoi %d",
        tovar_list[i_min].name,tovar_list[i_min].price );
    getch();
}

```

Рис.5. Листинг программы

Задача 2. Задано 5 отрезков с помощью координат их концов. Определить длину наибольшего из них.

Решение. Для начала определим структуру, определяющую отрезок:

```
struct Segment
{
    int x1, y1, x2, y2;
};
```

Функции заполнения массива с клавиатуры, случайными числами и из файла представлены на рис.6, 7, 8.

```
void Klava(Segment *mas)
{
    for (int i = 0; i < 5; i++)
    {
        printf("%d-й отрезок\n", i + 1);
        printf("Введите координаты начала %d-ого отрезка:\n", i + 1);
        printf("x="); scanf_s("%d", &mas[i].x1);
        printf("y="); scanf_s("%d", &mas[i].y1);
        printf("Введите координаты конца %d-ого отрезка:\n", i + 1);
        printf("x="); scanf_s("%d", &mas[i].x2);
        printf("y="); scanf_s("%d", &mas[i].y2);
    }
}
```

Рис.6. Ввод с клавиатуры

```
void Random(Segment *mas, int a, int b)
{
    for (int i = 0; i < 5; i++)
    {
        mas[i].x1 = rand() % (a - b + 1) - a;
        mas[i].y1 = rand() % (a - b + 1) - a;
        mas[i].x2 = rand() % (a - b + 1) - a;
        mas[i].y2 = rand() % (a - b + 1) - a;
    }
}
```

Рис.7. Заполнение случайными числами

```

void CopyFile(Segment*mas)
{
    FILE*a;
    fopen_s(&a, "Enter.txt", "rt");
    for (int i = 0; i < 5; i++)
    {
        fscanf_s(a, "%d %d %d %d\n", &mas[i].x1, &mas[i].y1, &mas[i].x2, &mas[i].y2);
    }
    fclose(a);
}

```

Рис.8. Ввод данных из файла

Функция, определяющая длину отрезка и вычисляющая максимум среди длин, представлена на рис.9.

```

double Task(Segment *a, double max)
{
    double l;
    for (int i = 0; i < 5; i++)
    {
        l = sqrt((a[i].x2 - a[i].x1)*(a[i].x2 - a[i].x1) + (a[i].y2 - a[i].y1)*(a[i].y2 - a[i].y1));
        if (l > max) max = l;
    }
    return max;
}

```

Рис.9. Поиск максимального отрезка

Стоит обратить внимание на передачу массива структур в функцию. Структура является пользовательским типом данных, поэтому для передачи массива элементов структур в файл достаточно передать указатель на первый элемент и количество элементов в массиве.

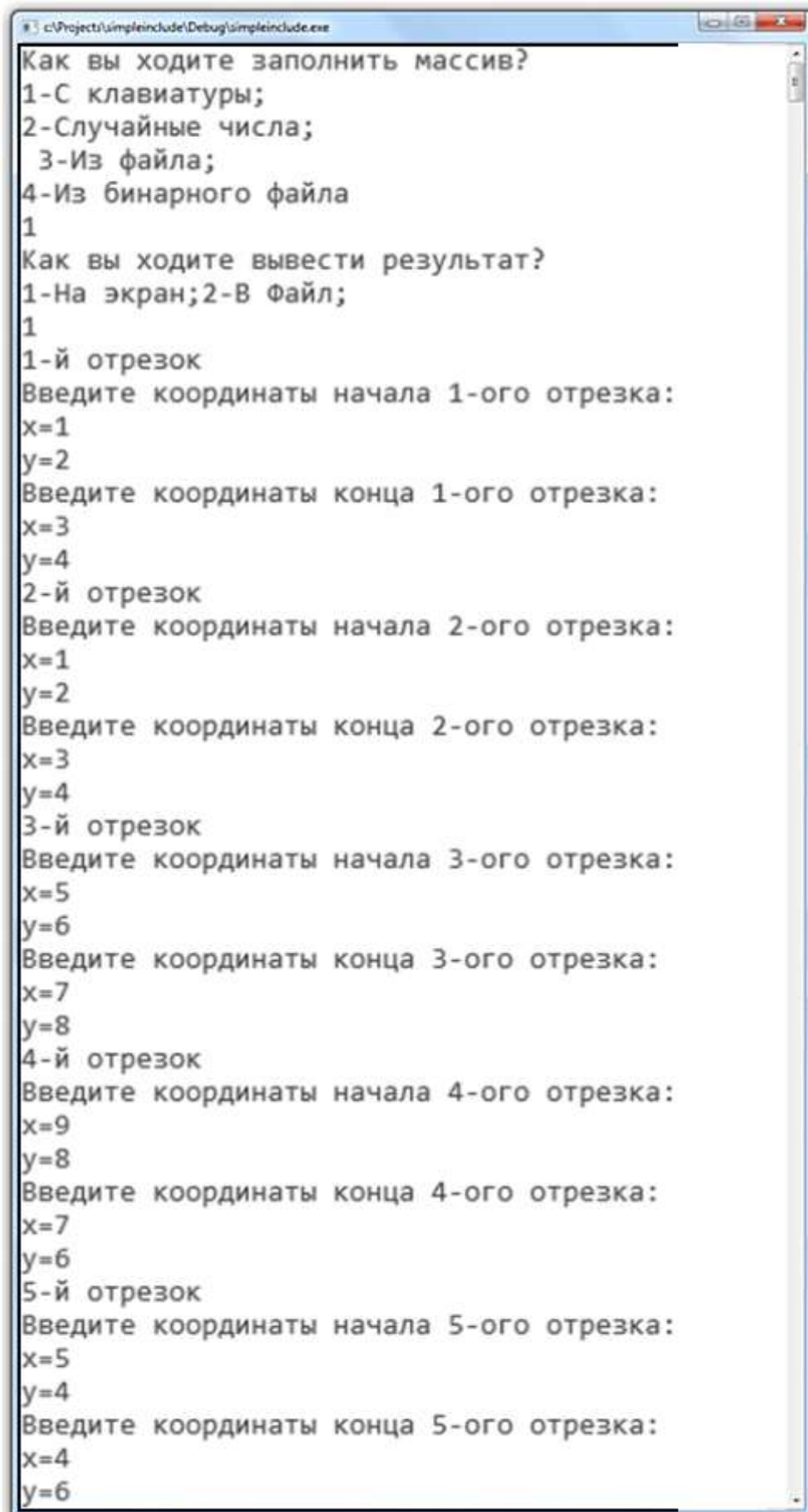
Реализация функции *main* представлена на рис.10. В функции описано создание массива, а также реализовано пользовательское меню.

На рисунках представлены только некоторые используемые функции. Реализацию остальных функций предлагается выполнить самостоятельно.

Результаты выполнения программы представлены на рис. 11, 12.

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Segment mas[5];
    double lengthMAX = 0;
    int A = -100, B = 100;
    srand(time(NULL));
    int Enter, OutPut;
    double s[5];
    printf("Как вы ходите заполнить массив?\n1-С клавиатуры;\n");
    printf("2-Случайные числа;\n 3-Из файла;\n4-Из бинарного файла\n");
    scanf_s("%d", &Enter);
    printf("Как вы ходите вывести результат?\n1-На экран;2-В Файл;\n");
    scanf_s("%d", &OutPut);
    switch (Enter)
    {
        case 1: Klava(mas); break;
        case 2: Random(mas, A, B); break;
        case 3: CopyFile(mas); break;
        case 4: EnterBinary(mas); break;
        default:
            break;
    }
    lengthMAX=Task(mas, lengthMAX);
    switch (OutPut)
    {
        case 1: Screen(lengthMAX); break;
        case 2: File(lengthMAX); break;
        default:
            break;
    }
    system("pause");
}
```

Рис.10. Функция main()



```
c:\Projects\simpleinclude\Debug\simpleinclude.exe
Как вы ходите заполнить массив?
1-С клавиатуры;
2-Случайные числа;
3-Из файла;
4-Из бинарного файла
1
Как вы ходите вывести результат?
1-На экран;2-В Файл;
1
1-й отрезок
Введите координаты начала 1-ого отрезка:
x=1
y=2
Введите координаты конца 1-ого отрезка:
x=3
y=4
2-й отрезок
Введите координаты начала 2-ого отрезка:
x=1
y=2
Введите координаты конца 2-ого отрезка:
x=3
y=4
3-й отрезок
Введите координаты начала 3-ого отрезка:
x=5
y=6
Введите координаты конца 3-ого отрезка:
x=7
y=8
4-й отрезок
Введите координаты начала 4-ого отрезка:
x=9
y=8
Введите координаты конца 4-ого отрезка:
x=7
y=6
5-й отрезок
Введите координаты начала 5-ого отрезка:
x=5
y=4
Введите координаты конца 5-ого отрезка:
x=4
y=6
```

Рис.11. Результат выполнения программы

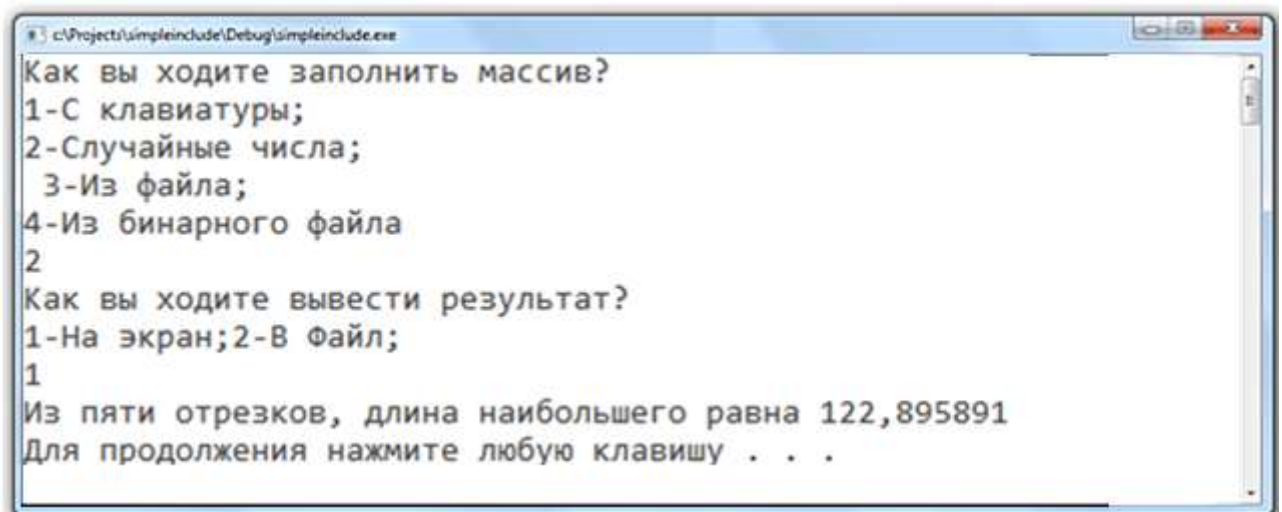


Рис.12. Результат выполнения программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить программу на языке Си, которая должна решать задачу согласно варианту из списка задач для самостоятельного решения.

Программа должна предоставить пользователю возможность выбора одного из двух способов заполнения массива: с клавиатуры или из текстового файла. Заполнение массива каждым способом, вывод массива на экран и решение задач согласно варианту из списка задач для самостоятельного решения необходимо оформить в виде отдельных подпрограмм. Массивы и их длины передавать функциям необходимо через список аргументов, глобальные данные не использовать. Ответ выводить на экран

Программа должна запросить у пользователя все необходимые исходные данные и вывести результаты на экран. Она должна быть отформатирована и снабжена комментариями.

5. Задачи для самостоятельного решения

1. Пусть для каждого студента в группе, состоящей из N человек, заданы оценки по трем экзаменам. Определить количество отличников и их процент от общего числа студентов.

2. Пусть для каждого студента в группе, состоящей из N человек, заданы оценки по трем экзаменам. Определить студента с наибольшим средним баллом по экзаменам и вывести его оценки.

3. Пусть для каждого студента в группе, состоящей из N человек, заданы оценки по трем экзаменам. Определить количество и процент студентов, сдавших экзамены без оценок «3» и «2».

4. Пусть для каждого студента в группе, состоящей из N человек, заданы оценки по трем экзаменам. Определить средний балл в группе по каждому из предметов.

5. Пусть задано N отрезков с помощью координат их концов. Определить длины наибольшего и наименьшего из них, также вывести их координаты.

6. Пусть задано N отрезков с помощью координат их концов. Определить среднюю длину отрезков, вывести номера отрезков, чьи длины больше средней.

7. Пусть задано N прямоугольников, каждый из них задан с помощью координат его левого верхнего угла и правого нижнего угла. Определить прямоугольник наибольшей площади, вывести его номер и площадь.

8. Пусть задано N прямоугольников, каждый из них задан с помощью координат его левого верхнего угла, ширины и высоты. Определить прямоугольник наименьшего периметра, вывести координаты его вершин и периметр.

9. Пусть задан список товаров (название и цена), количество которых задает пользователь. Вывести товары, которые дешевле заданной цены.

10. Пусть задан список товаров (название и цена), количество которых задает пользователь. Определить среднюю цену товаров.

11. Пусть задан список товаров (название, цена, дата поступления), количество которых задает пользователь. Вывести названия и цены товаров, поступивших позже указанной пользователем даты (день, месяц, год).

12. Пусть задан список товаров (название, цена, количество единиц товара на складе), количество которых задает пользователь. Определить общую стоимость товаров на складе.

13. Имеется массив данных о K работающих в фирме: фамилия и дата поступления на работу (день, месяц, год). Вывести на экран фамилии тех, кто поступил на работу в определенную дату. Дата (день, месяц и год) вводится с клавиатуры.

14. Имеется массив данных о K работающих в фирме: фамилия и дата поступления на работу (день, месяц, год). Вывести на экран фамилию того, кто работает дольше всех остальных.

15. Даны сведения о К пассажирах авиарейса: фамилия, место в самолете, вес багажа. Вывести на экран фамилии и места тех, кто имеет вес багажа, превосходящий средний среди всех пассажиров.

16. Даны сведения о К пассажирах авиарейса: фамилия, место в самолете, вес багажа. Вывести на экран фамилии и места пассажиров с самым тяжелым багажом и самым легким багажом.

17. Известны следующие данные о N учениках нескольких школ: фамилия, школа и класс. Вывести на экран фамилии тех учеников, которые учатся в определенной школе в старших классах (номер этой школы вводится с клавиатуры).

18. Известны следующие данные о N учениках нескольких школ: фамилия, школа и класс. Вывести на экран фамилии и школы тех учеников, которые учатся в определенном классе (номер класса вводится с клавиатуры).

19. Известны следующие данные о N учениках класса: фамилия, дата рождения (число, месяц и год). Вывести на экран фамилии тех учеников, у кого сегодня день рождения (сегодняшнюю дату вводить с клавиатуры).

20. Известны следующие данные о N учениках класса: фамилия, дата рождения (число, месяц и год). Вывести на экран фамилию самого старшего по возрасту ученика.

6. Контрольные вопросы

1. Что такое структура?
2. Как задается структура на языке Си?
3. Каким образом происходит обращение к полю структуры?
4. Как задается массив структур?
5. Может ли объект структуры в свою очередь являться элементом структуры?

Лабораторная работа №16. Бинарные файлы

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки работы с бинарными файлами.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

Текстовые файлы хранят данные в виде текста. Это значит, что если, например, записываем в файл целое число 12345678, то записывается 8 символов, а это 8 байт данных несмотря на то, что число помещается в целый тип. Кроме того, вывод и ввод данных является форматированным, то есть каждый раз, когда считывается число из файла или записывается в файл, происходит трансформация числа в строку или обратно. Это затратные операции, которых можно избежать.

Текстовые файлы позволяют хранить информацию в виде, понятном для человека. Можно, однако, хранить данные непосредственно в бинарном виде. Для этого используются бинарные файлы.

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

#define ERROR_FILE_OPEN -3

void main()
{
    FILE *output = NULL;
    int number;

    output = fopen("D:/c/output.bin", "wb");
    if (output == NULL)
    {
        printf("Error opening file");
        getch();
        exit(ERROR_FILE_OPEN);
    }

    scanf("%d", &number);
    fwrite(&number, sizeof(int), 1, output);
```

```

fclose(output);
_getch();
}

```

Выполните программу и посмотрите содержимое файла `output.bin`. Число, которое ввёл пользователь записывается в файл непосредственно в бинарном виде. Можете открыть файл в любом редакторе, поддерживающем представление в шестнадцатеричном виде (Total Commander, Far) и убедиться в этом.

Запись в файл осуществляется с помощью функции [2]

```

size_t fwrite ( const void * ptr, size_t size, size_t
count, FILE * stream );

```

Функция возвращает количество удачно записанных элементов. В качестве аргументов принимает указатель на массив, размер одного элемента, число элементов и указатель на файловый поток. Вместо массив, конечно, может быть передан любой объект.

Запись в бинарный файл объекта похожа на его отображение: берутся данные из оперативной памяти и пишутся как есть. Для считывания используется функция `fread`

```

size_t fread ( void * ptr, size_t size, size_t count, FILE * stream );

```

Функция возвращает число удачно прочитанных элементов, которые помещаются по адресу `ptr`. Всего считывается `count` элементов по `size` байт. Давайте теперь считаем наше число обратно в переменную.

Одной из значимых функций для работы с бинарными файлами является функция ***fseek***

```

int fseek ( FILE * stream, long int offset, int origin );

```

Эта функция устанавливает указатель позиции, ассоциированный с потоком, на новое положение. Индикатор позиции указывает, на каком месте в файле мы остановились. Когда открывается файл, позиция равна 0. Каждый раз при записи байта данных указатель позиции сдвигается на единицу вперёд.

fseek принимает в качестве аргументов указатель на поток и сдвиг в *offset* байт относительно *origin*. *origin* может принимать три значения:

SEEK_SET – начало файла;

SEEK_CUR – текущее положение файла;

SEEK_END - конец файла.

При удачной работе функция возвращает 0.

Также для установки указателя позиции в начало файла можно использовать функцию *rewind*.

В Си определён специальный тип `fpos_t`, который используется для хранения позиции индикатора позиции в файле.

Функция

```
int fgetpos (FILE * stream, fpos_t * pos)
```

используется, чтобы назначить переменной `pos` текущее положение.

Функция

```
int fsetpos (FILE * stream, const fpos_t * pos)
```

используется для перевода указателя в позицию, которая хранится в переменной `pos`. Обе функции при удачном завершении возвращают ноль.

```
long int ftell (FILE * stream);
```

возвращает текущее положение индикатора относительно начала файла. Для бинарных файлов – это число байт, для текстовых не определено (если текстовый файл состоит из однобайтовых символов, то также число байт).

3. Общая часть лабораторной работы

Задача 1. Записать число, затем сдвинуть указатель на начало файла и прочитать ранее записанное число.

Решение. Программа в представленном ниже листинге решает задачу.

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

void main() {
    FILE *iofile = NULL;
    int number;
    iofile = fopen("D:/c/output.bin", "w+b");
    if (iofile == NULL) {
        printf("Error opening file");
        getch();
        exit(ERROR_FILE_OPEN);
    }

    scanf("%d", &number);
    fwrite(&number, sizeof(int), 1, iofile);
```



```
fseek(iofile, 0, SEEK_SET);  
number = 0;  
fread(&number, sizeof(int), 1, iofile);  
printf("%d", number);  
  
fclose(iofile);  
_getch();  
}
```

Особый интерес представляет проверка существования файла:

```
if (iofile == NULL)  
{  
    printf("Error opening file");  
    getch();  
    exit(ERROR_FILE_OPEN);  
}
```

Результат выполнения программы при отсутствии файла представлен на рис.1.

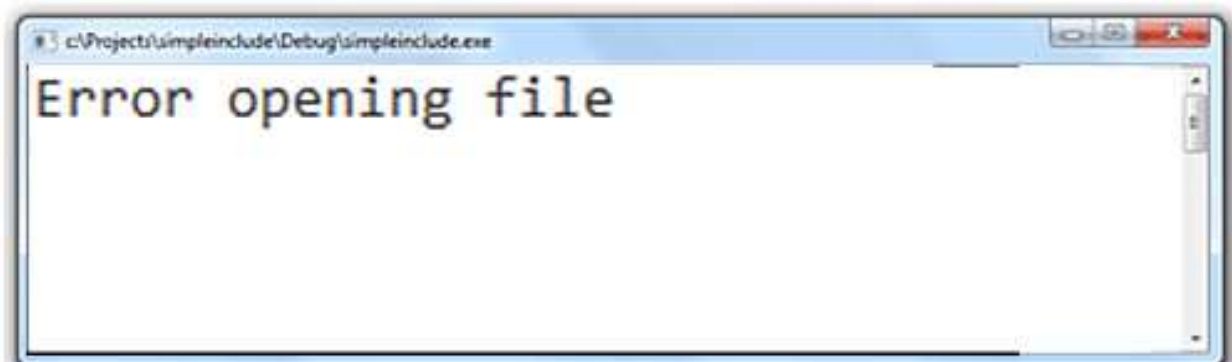


Рис.1. Результат выполнения программы
в случае отсутствия файла

Результат выполнения программы при наличии файла представлен на рис. 2.



Рис.2. Успешное выполнение программы

Задача 2. Организовать для пользователя последовательный ввод произвольного количества чисел. Записать эти числа в бинарный файл. В начале файла записать количество чисел в нём.

Решение. Откроем бинарный файл, организуем циклический ввод чисел. Каждое введённое число будем записывать в файл. После завершения ввода переместим указатель позиции в начало файла и запишем туда количество введённых элементов.

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

void main()
{
    FILE *iofile = NULL;
    unsigned counter = 0;
    int num;
    int yn;

    iofile = fopen("D:/c/numbers.bin", "w+b");
    if (iofile == NULL)
    {
        printf("Error opening file");
        getch();
        exit(ERROR_OPEN_FILE);
    }
    fwrite(&counter, sizeof(int), 1, iofile);

    do
    {
        printf("enter new number? [1 - yes, 2 - no]");
        scanf("%d", &yn);
        if (yn == 1)
        {
            scanf("%d", &num);
            fwrite(&num, sizeof(int), 1, iofile);
            counter++;
        }
        else
        {
            rewind(iofile);
            fwrite(&counter, sizeof(int), 1, iofile);
            break;
        }
    }
}
```

```

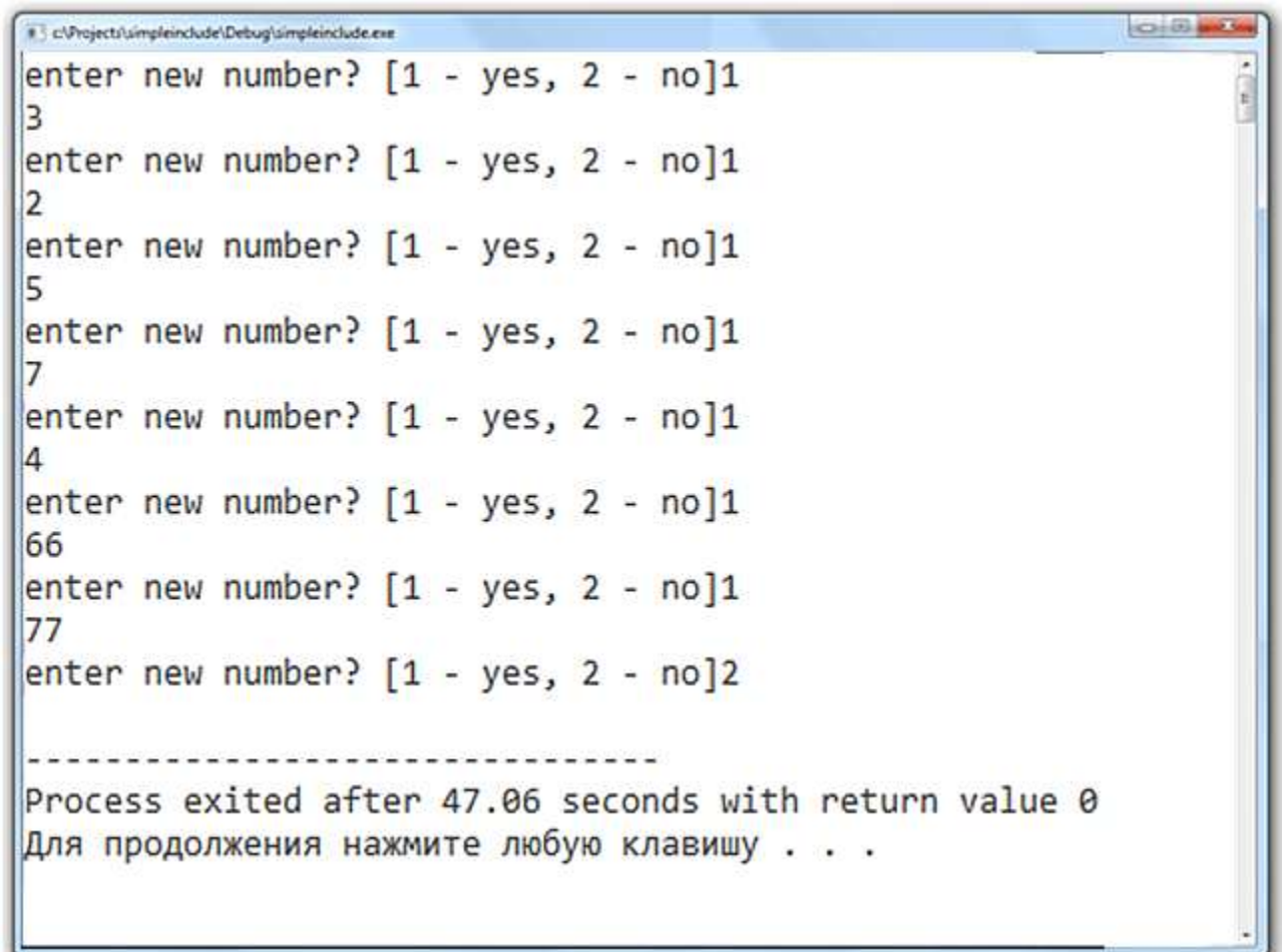
    }
}while(1);

fclose(iofile);
getch();
}

```

Результат выполнения программы представлен на рис.3.

Открытие полученного файла в блокноте не позволит прочесть его содержимое, однако используя специальный просмотрщик бинарных файлов, можно увидеть его содержимое.



```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
enter new number? [1 - yes, 2 - no]1
3
enter new number? [1 - yes, 2 - no]1
2
enter new number? [1 - yes, 2 - no]1
5
enter new number? [1 - yes, 2 - no]1
7
enter new number? [1 - yes, 2 - no]1
4
enter new number? [1 - yes, 2 - no]1
66
enter new number? [1 - yes, 2 - no]1
77
enter new number? [1 - yes, 2 - no]2

-----
Process exited after 47.06 seconds with return value 0
Для продолжения нажмите любую клавишу . . .

```

Рис.3. Результат выполнения программы

Задача 3. Прочитать из бинарного файла количество записанных чисел, а затем прочитать и вывести эти числа по порядку.

Решение. Решение задачи выполняется программой, листинг которой приведён ниже.

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

```

```
void main()
{
    FILE *iofile = NULL;
    unsigned counter;
    int i, num;

    iofile = fopen("D:/c/numbers.bin", "rb");
    if (iofile == NULL) {
        printf("Error opening file");
        getch();
        exit(ERROR_OPEN_FILE);
    }

    fread(&counter, sizeof(int), 1, iofile);
    for (i = 0; i < counter; i++)
    {
        fread(&num, sizeof(int), 1, iofile);
        printf("%d\n", num);
    }

    fclose(iofile);
    getch();
}
```

Результат выполнения программы представлен на рис.4. Как видно, на экран выведены числа, записанные в файл при решении предыдущей задачи.

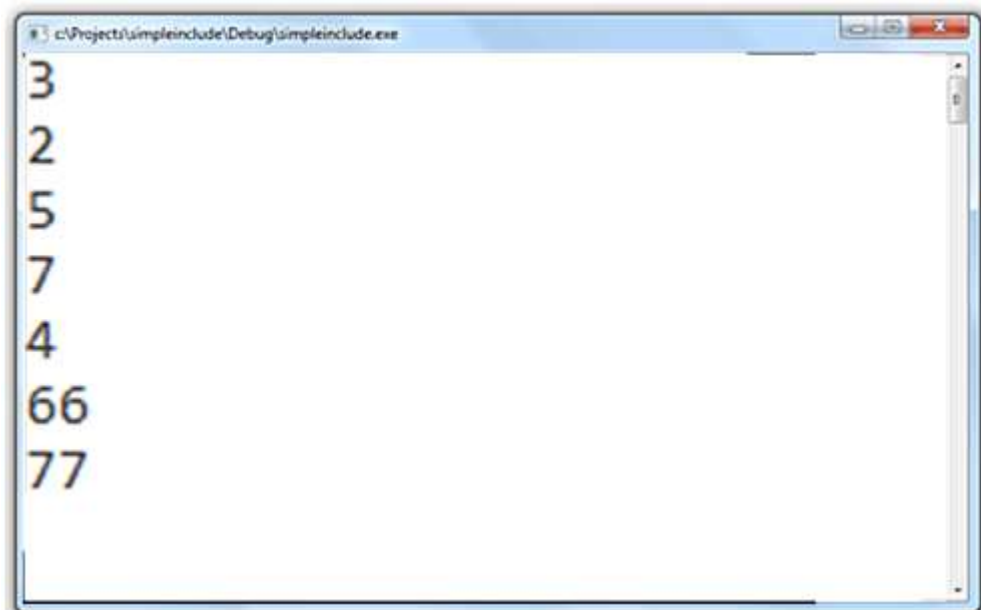


Рис.4. Результат выполнения программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить программу на языке Си.

За основу берется выполненное задание предыдущей лабораторной работы. К реализованному в ней функционалу добавить следующие возможности:

- сохранение в бинарный файл массива структур, введенного с клавиатуры;
- считывание массива структур из бинарного файла (количество элементов массива хранится в начале файла).

Сохранение массива в бинарный файл и считывание из бинарного файла необходимо оформить в виде отдельных подпрограмм. Массивы и их длины передавать функциям необходимо через список аргументов, глобальные данные не использовать.

Программа должна запросить у пользователя все необходимые исходные данные и вывести массивы и результат решения задачи на экран. Она должна быть отформатирована и снабжена комментариями.

5. Контрольные вопросы

1. Что такое бинарный файл?
2. Чем бинарный файл отличается от текстового?
3. Какие функции для работы с бинарными файлами существуют?
4. В каком виде хранится информация в бинарных файлах?
5. При решении каких задач более удобно использовать текстовые файлы, а при решении каких – бинарные?

Лабораторная работа №17. Двумерные массивы

1. Цель лабораторной работы

Цель лабораторной работы – получить навыки работы с двумерными массивами.

Продолжительность лабораторной работы – 2 часа.

2. Краткие теоретические сведения

До этого момента рассматривались только одномерные массивы, то есть, к элементу массива обращались через один индекс. Однако массивы могут быть и двумерными, и трехмерными, и даже n-мерными. Многомерные массивы – это массивы, у которых есть более одного индекса. Вместо одной строки элементов многомерные массивы можно рассматривать как совокупность элементов, которые распределены по двум или более измерениям [2].

Язык Си позволяет создавать многомерные массивы. Простейшим видом многомерного массива является двумерный массив. Двумерный массив – это массив одномерных массивов. Двумерный массив объявляется следующим образом:

```
тип имя_массива[размер_измерения2][размер_измерения1];
```

Следовательно, для объявления двумерного массива целых чисел размером 10 на 20 необходимо написать:

```
int d[10][20];
```

Объявление двумерного массива почти ничем не отличается от объявления одномерного, за исключением того, что при объявлении двумерного массива, указывают размер каждого измерения в квадратных скобках. Так, объявим двумерный массив размером 8 на 8, это размер поля для стандартных шашек – 8 строк и 8 столбцов:

```
int checkers[8][8]; // двумерный массив
```

Посмотрим внимательно на это объявление. В противоположность другим компьютерным языкам, где размеры массива отделяются запятой, язык Си помещает каждый размер в отдельные скобки.

Для доступа к элементу с индексами 3, 5 массива d необходимо использовать d[3][5].

Количество байт в памяти, требуемых для размещения двумерного массива, вычисляется следующим образом:

число байт = размер второго измерения * размер первого измерения * sizeof (базовый тип)

Если целые числа занимают 2 байта, то целочисленный массив размером 10 на 5 будет занимать $10 * 5 * 2$, то есть 100 байт.

Когда двумерный массив используется как аргумент функции, передается указатель на первый элемент. Функция, получающая двумерный массив, должна, как минимум, определять размер первого измерения, так как компилятору необходимо знать длину каждой строки для корректной индексации массива. Так, функция, получающая двумерный целочисленный массив размера 5 на 10, будет объявляться следующим образом:

```
void func1 (int x[][10])
{
...
}
```

Можно определить количество строк, но это не обязательно. Компилятору достаточно знать количество столбцов для правильного выполнения операторов типа.

3. Общая часть лабораторной работы

Задача 1. Заполнить массив 3*4 числами от 1 до 12, после чего массив вывести на экран.

Решение. Решение задачи выполняется программой, листинг которой приведён ниже.

```
#include <stdio.h>

////////////////////////////////////

int main(void)
{
    int t,i, num[3][4];

    /* загрузка чисел */
    for(t=0; t<3; ++t)
        for (i=0; i<4; ++i)
            num[t][i] = (t*4)+i+1;
```

```

/* ВЫВОД ЧИСЕЛ */
for (t=0; t<3; ++t)
{
    for (i=0; i<4; ++i)
    {
        printf("%d  ", num[t][i]);
    }
    printf ("\n");
}

return 0;
}

```

В примере num[0][0] имеет значение 1, num[0][1] – значение 2, num[0][2] – 3 и так далее. num[2][3] имеет значение 12.

Рассмотрим некоторые функции для работы с двумерным массивом. Функции для заполнения массива с клавиатуры, случайными числами, по формуле и из текстового файла представлены на рис.1, 2, 3.

```

void openTXT(double matrix[100][100], unsigned *string, unsigned *column){
    FILE *file;
    file=fopen("file.txt", "r");
    fscanf(file, "%u", string);
    fscanf(file, "%u", column);
    for(unsigned i=0; i<*string; i++)
    {
        for(unsigned j=0; j<*column; j++){
            fscanf(file, "%lf", &matrix[i][j]);
        }
    }
}

void byKeyboard(double matrix[100][100], unsigned string, unsigned column){
    for(unsigned i=0; i<string; i++)
    {
        for(unsigned j=0; j<column; j++){
            printf("Input element %u %u: ", i+1, j+1); scanf("%lf", &matrix[i][j]);
        }
    }
}

```

Рис.1. Функции ввода массива из файла и с клавиатуры


```

void byFormula(double matrix[100][100], unsigned string, unsigned column){
    for(unsigned i=1; i<string+1; i++){
        for(unsigned j=1; j<column+1; j++){
            if(i>j)
                matrix[i-1][j-1]=cos(i)/sin(i);
            else if(i==j)
                matrix[i-1][j-1]=tan(i+j);
            else matrix[i-1][j-1]=cos(j)/sin(j);
        }
    }
}

```

Рис.2. Заполнение массива по формуле

```

void byRandom(double matrix[100][100], unsigned string, unsigned column){
    int a,b;
    printf("Input left border: "); scanf("%d",&a);
    do{
        printf("Input right border:(more than a) "); scanf("%d",&b);
    } while(b<=a);
    for(unsigned i=0; i<string; i++){
        for(unsigned j=0; j<column; j++){
            matrix[i][j]=rand()%(b-a+1)+a;
        }
    }
}

```

Рис.3. Заполнение массива случайными числами

Функции вывода массива на экран и записи в текстовый файл представлены на рис. 4, 5.

```

void printMatrix(double matrix[100][100], unsigned string, unsigned column){
    for(unsigned i=0; i<string; i++){
        for(unsigned j=0; j<column; j++){
            if(j==column-1)
                printf("%3.2lf\n", matrix[i][j]);
            else
                printf("%3.2lf ", matrix[i][j]);
        }
    }
}

```

Рис.4. Вывод массива на экран

```

void writeToTXT(unsigned cout){
    FILE *file;
    file=fopen("answerFile.txt","w");
    fileOpenTest(file);
    fprintf(file, "%u",cout);
    fclose(file);
}

```

Рис.5. Запись массива в файл

Функция *main* представлена на рис.6.

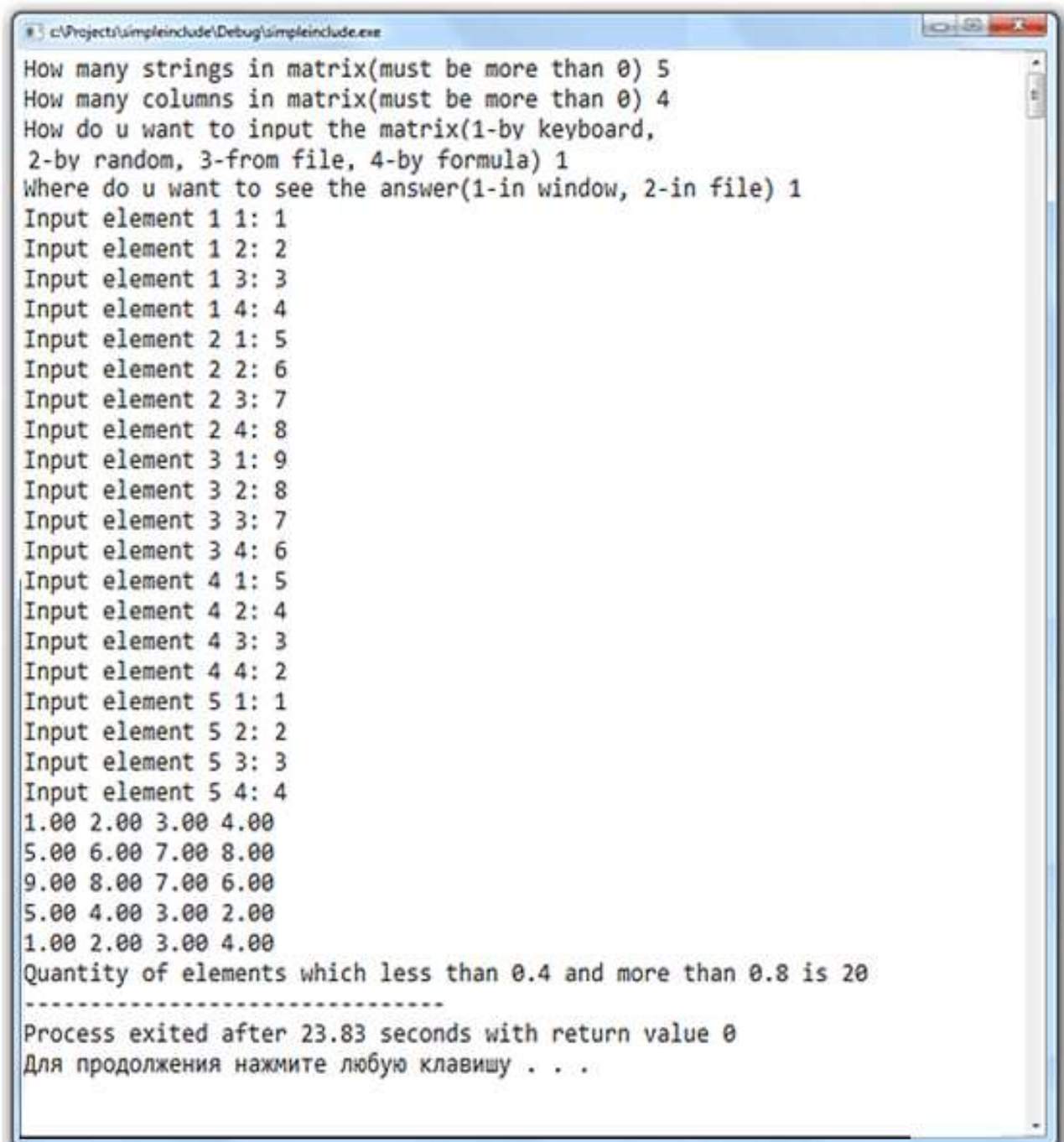
```

int main(){
    srand(time(NULL));
    unsigned short inputChoice, outPutChoice;
    unsigned string, column, cout=0;
    do{
        printf("How many strings in matrix(must be more than 0) "); scanf("%u",&string);
    } while (string==0);
    do{
        printf("How many columns in matrix(must be more than 0) "); scanf("%u",&column);
    } while (column==0);
    double matrix[100][100];
    do{
        printf("How do u want to input the matrix(1-by keyboard, 2-by random, 3-from file, 4-by formula) ");
        scanf("%hu",&inputChoice);
    } while (inputChoice==0 || inputChoice>4);
    do{
        printf("Where do u want to see the answer(1-in window, 2-in file) "); scanf("%hu",&outPutChoice);
    } while (outPutChoice==0 || outPutChoice>2);
    switch(inputChoice){
        case 1: byKeyboard(matrix,string, column); break;
        case 2: byRandom(matrix,string, column); break;
        case 3: fromFile(matrix,&string, &column); break;
        case 4: byFormula(matrix,string, column); break;
        default: break;
    }
    cout=findQuantityOfElements(matrix,string, column, cout);
    printMatrix(matrix,string,column);
    switch(outPutChoice){
        case 1: printToWindow(cout); break;
        case 2: printToFile(cout); break;
        default: break;
    }
    return 0;
}

```

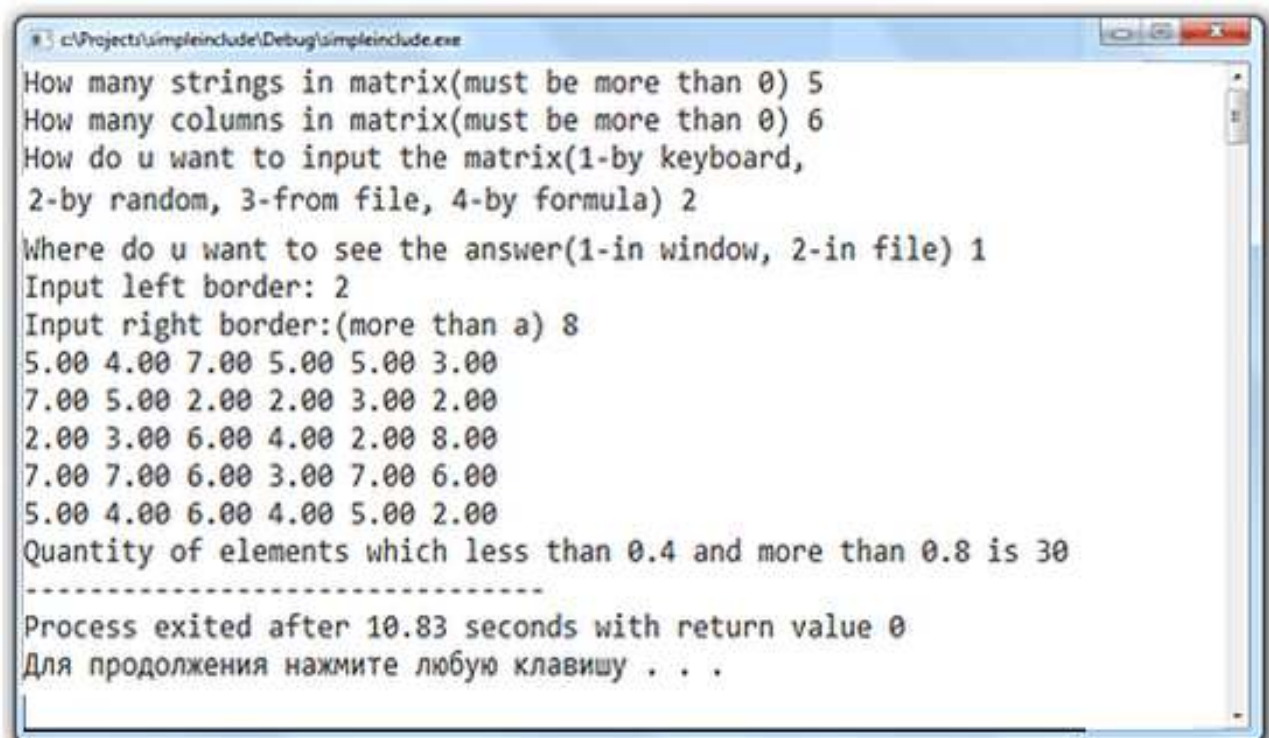
Рис.6. Функция *main*

Результаты выполнения программы представлены на рис.7, 8, 9.



```
c:\Project\simpleinclude\Debug\simpleinclude.exe
How many strings in matrix(must be more than 0) 5
How many columns in matrix(must be more than 0) 4
How do u want to input the matrix(1-by keyboard,
2-by random, 3-from file, 4-by formula) 1
Where do u want to see the answer(1-in window, 2-in file) 1
Input element 1 1: 1
Input element 1 2: 2
Input element 1 3: 3
Input element 1 4: 4
Input element 2 1: 5
Input element 2 2: 6
Input element 2 3: 7
Input element 2 4: 8
Input element 3 1: 9
Input element 3 2: 8
Input element 3 3: 7
Input element 3 4: 6
Input element 4 1: 5
Input element 4 2: 4
Input element 4 3: 3
Input element 4 4: 2
Input element 5 1: 1
Input element 5 2: 2
Input element 5 3: 3
Input element 5 4: 4
1.00 2.00 3.00 4.00
5.00 6.00 7.00 8.00
9.00 8.00 7.00 6.00
5.00 4.00 3.00 2.00
1.00 2.00 3.00 4.00
Quantity of elements which less than 0.4 and more than 0.8 is 20
-----
Process exited after 23.83 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Рис.7. Результат выполнения программы

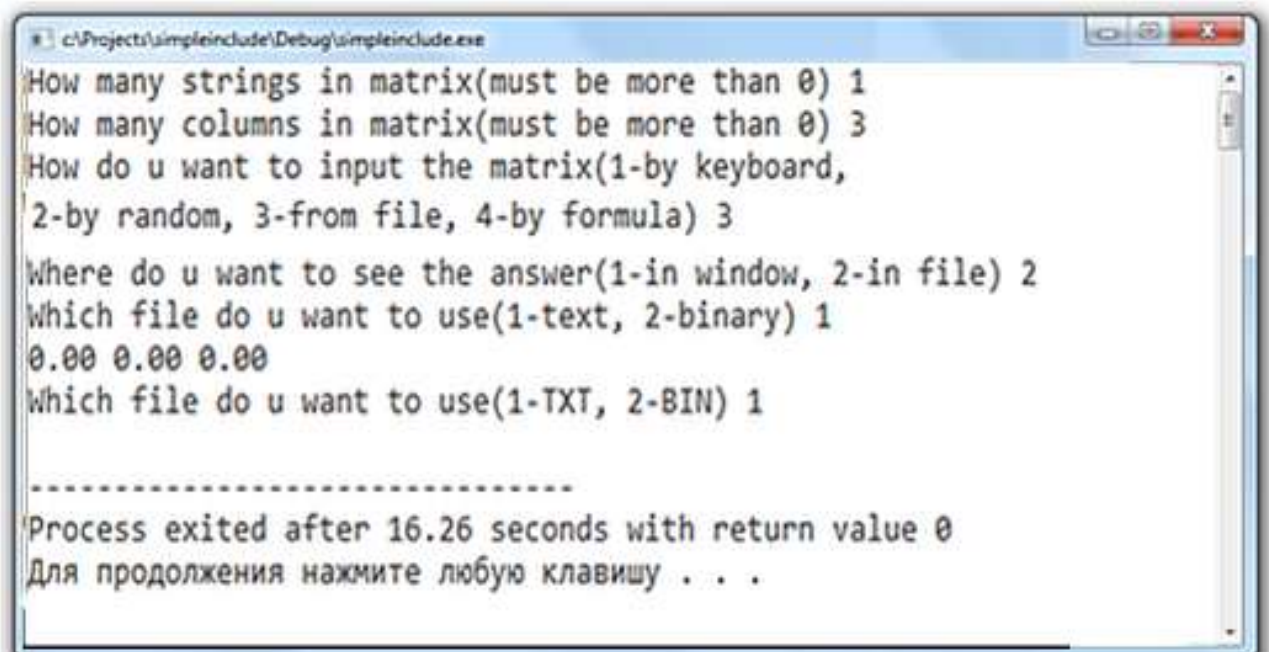


```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
How many strings in matrix(must be more than 0) 5
How many columns in matrix(must be more than 0) 6
How do u want to input the matrix(1-by keyboard,
2-by random, 3-from file, 4-by formula) 2
Where do u want to see the answer(1-in window, 2-in file) 1
Input left border: 2
Input right border:(more than a) 8
5.00 4.00 7.00 5.00 5.00 3.00
7.00 5.00 2.00 2.00 3.00 2.00
2.00 3.00 6.00 4.00 2.00 8.00
7.00 7.00 6.00 3.00 7.00 6.00
5.00 4.00 6.00 4.00 5.00 2.00
Quantity of elements which less than 0.4 and more than 0.8 is 30
-----
Process exited after 10.83 seconds with return value 0
Для продолжения нажмите любую клавишу . . .

```

Рис.8. Результат выполнения программы



```

c:\Projects\simpleinclude\Debug\simpleinclude.exe
How many strings in matrix(must be more than 0) 1
How many columns in matrix(must be more than 0) 3
How do u want to input the matrix(1-by keyboard,
2-by random, 3-from file, 4-by formula) 3
Where do u want to see the answer(1-in window, 2-in file) 2
Which file do u want to use(1-text, 2-binary) 1
0.00 0.00 0.00
Which file do u want to use(1-TXT, 2-BIN) 1
-----
Process exited after 16.26 seconds with return value 0
Для продолжения нажмите любую клавишу . . .

```

Рис.9. Результат выполнения программы

4. Индивидуальная часть лабораторной работы

В индивидуальной части лабораторной работы написать и отладить программу на языке Си, которая должна решать задачу согласно варианту из списка задач для самостоятельного решения.

Программа должна предоставить пользователю следующие возможности:

- ввод массива с клавиатуры;
- ввод массива с клавиатуры из файла;
- заполнение массива случайными числами из указанного диапазона;
- заполнение массива по заданной формуле согласно варианту;
- вывод массива на экран;
- вывод массива в текстовый файл;
- вывод массива в бинарный файл.

Заполнение массива каждым способом, вывод массива на экран и решение задач согласно варианту из списка задач для самостоятельного решения необходимо оформить в виде отдельных подпрограмм. Массивы и их размеры передавать функциям необходимо через список аргументов, глобальные данные не использовать. Ответ выводить на экран

Программа должна быть отформатирована и снабжена комментариями.

5. Задачи для самостоятельного решения

Формулы для заполнения массивов

$$1. \ b_{ij} = \begin{cases} e^{j-i}, & i \leq j \\ e^{\frac{i}{j}}, & i > j \end{cases}$$

$$2. \ l_{ij} = \begin{cases} i + j, & i > j \\ i \times j, & i \leq j \end{cases}$$

$$3. \quad a_{ij} = \begin{cases} \frac{1}{i^2 + 2}, & i \leq j \\ \frac{1}{i + j}, & i > j \end{cases}$$

$$4. \quad a_{ij} = \begin{cases} \sin(i + j), & i > j \\ 2, & i = j \\ \arcsin \frac{i + j}{2i + 3j}, & i < j \end{cases}$$

$$5. \quad b_{ij} = \begin{cases} \frac{1}{i + j - 1}, & i < j \\ 0, & i = j \\ -\frac{1}{i + j - 1}, & i > j \end{cases}$$

$$6. \quad a_{ij} = \begin{cases} i - 2j, & i > j \\ i \times j, & i = j \\ 3i + j, & i < j \end{cases}$$

$$7. \quad a_{ij} = \begin{cases} \operatorname{ctg} i, & i > j \\ \operatorname{tg}(i + j), & i = j \\ \operatorname{ctg} j, & i < j \end{cases}$$

$$8. \quad a_{ij} = \begin{cases} i + j, & i > j \\ \frac{i}{j}, & i = j \\ i - j, & i < j \end{cases}$$

$$9. \quad a_{ij} = \begin{cases} 0, & i > j \\ 1, & i = j \\ i + j, & i < j \end{cases}$$

$$10. \quad a_{ij} = \begin{cases} \sin(i * j), & i > j \\ \ln(2i + 3j), & i = j \\ \cos(i - j), & i < j \end{cases}$$

$$11. \quad a_{ij} = \begin{cases} \frac{i + j}{i^2 + 2}, & i \leq j \\ \frac{2}{i + j}, & i > j \end{cases}$$

$$12. \quad a_{ij} = \begin{cases} \sin(2i + j), & i > j \\ 3, & i = j \\ \cos(i) * \frac{i + j}{2i + 3j}, & i < j \end{cases}$$

$$13. \quad b_{ij} = \begin{cases} \frac{4}{i + j - 1}, & i < j \\ 0, & i = j \\ \frac{1}{2i + 7j - 1}, & i > j \end{cases}$$

$$14. \quad a_{ij} = \begin{cases} i - 2j, & i > j \\ i + j, & i = j \\ 3i + j, & i < j \end{cases}$$

$$15. \quad a_{ij} = \begin{cases} \operatorname{ctg} i, & i > j \\ \operatorname{tg}(i+j), & i = j \\ i+j, & i < j \end{cases}$$

$$16. \quad a_{ij} = \begin{cases} i+j, & i > j \\ 0, & i = j \\ i-2j, & i < j \end{cases}$$

$$17. \quad a_{ij} = \begin{cases} i, & i > j \\ 1, & i = j \\ i-j, & i < j \end{cases}$$

$$18. \quad a_{ij} = \begin{cases} \sin(i-j), & i > j \\ \ln(i), & i = j \\ \cos(i*j), & i < j \end{cases}$$

$$19. \quad a_{ij} = \begin{cases} \frac{1}{i^3 + 2j}, & i \leq j \\ \frac{2i}{i+j}, & i > j \end{cases}$$

$$20. \quad a_{ij} = \begin{cases} \sin(i+7j), & i > j \\ 2, & i = j \\ \arcsin \frac{i+j}{2i+3j}, & i < j \end{cases}$$

$$21. \quad b_{ij} = \begin{cases} \frac{1}{i+j}, & i < j \\ i, & i = j \\ -\frac{8}{i+j-1}, & i > j \end{cases}$$

$$22. \quad a_{ij} = \begin{cases} i-2j, & i > j \\ i+4j, & i = j \\ 3i+5j, & i < j \end{cases}$$

$$23. \quad a_{ij} = \begin{cases} \operatorname{ctg} 2i, & i > j \\ \operatorname{tg}(j), & i = j \\ \operatorname{ctg} 3j, & i < j \end{cases}$$

$$24. \quad a_{ij} = \begin{cases} i+j, & i > j \\ \frac{i-1}{j+2}, & i = j \\ i * j, & i < j \end{cases}$$

$$25. \quad a_{ij} = \begin{cases} i, & i > j \\ 1, & i = j \\ i / j, & i < j \end{cases}$$

$$26. \quad a_{ij} = \begin{cases} \sin(i+j), & i > j \\ \ln(i * j), & i = j \\ \cos(i / j), & i < j \end{cases}$$

Действия над массивом

1. Вычислить среднее арифметическое элементов строки, в которой находится максимальный элемент матрицы.
2. Вычислить среднее арифметическое элементов столбца, в котором находится минимальный элемент матрицы.
3. Вычислить среднее геометрическое элементов строки, в которой находится минимальный элемент матрицы.
4. Вычислить среднее геометрическое элементов столбца, в котором находится максимальный элемент матрицы.
5. Вычислить сумму каждого столбца. Среди полученных сумм найти максимальное значение.
6. Вычислить произведение каждой строки. Среди полученных произведений найти минимальное значение.
7. Вычислить сумму элементов, произведение индексов которых четно.
8. Вычислить произведение элементов, сумма индексов которых нечетна.
9. Найти максимальное среди минимальных значений по строкам.
10. Найти минимальное среди максимальных значений по столбцам.
11. Вычислить сумму элементов тех строк матрицы, в которых отрицателен элемент главной диагонали.
12. Вычислить произведение элементов тех столбцов матрицы, в которых положителен элемент побочной диагонали.
13. Вычислить среднее арифметическое элементов, лежащих выше главной диагонали матрицы.
14. Вычислить среднее геометрическое элементов, лежащих ниже побочной диагонали матрицы.
15. Найти минимальный элемент, лежащий выше побочной диагонали.
16. Найти максимальный элемент, лежащий ниже главной диагонали.
17. Вычислить сумму отрицательных чисел в каждой строке.
18. Вычислить произведение положительных чисел в каждом столбце.

19. Вычислить сумму наибольшего и наименьшего элементов матрицы.

20. Вычислить разность между суммой элементов главной диагонали и суммой элементов побочной диагонали.

21. Определить количество элементов матрицы, лежащих вне интервала $(0.4; 0.8)$.

22. Определить количество элементов матрицы, лежащих в интервале $(0.3; 0.7)$.

23. Вычислить среднее арифметическое неотрицательных элементов матрицы, а также подсчитать, сколько таких элементов в каждой строке матрицы.

6. Контрольные вопросы

1. Что такое двумерный массив?
2. Каким образом происходит адресация элементов в двумерном массиве?
3. Как передать двумерный массив в функцию?
4. Как организовать перебор всех элементов двумерного массива?

Расчётно-графическая работа

1. Цель расчётно-графической работы

Цель расчётно-графической работы – обобщить опыт использования структур и файлов.

2. Задание к расчётно-графической работе

При выполнении расчётно-графической работы студенты должны разработать следующие конструкции.

1. Структуру для хранения данных по заданной индивидуальным заданием теме.

2. Массив структур разработанного типа.

3. Набор средств по работе с массивом структур, включающий следующие функции:

- ввод с клавиатуры массива данных;
- вывод сведений на экран;
- вывод сведений в текстовый файл;
- вывод сведений в бинарный файл;
- загрузка массива данных из текстового файла;
- загрузка массива данных из бинарного файла;
- замена данных по номеру элемента в текстовом файле;
- замена данных по номеру элемента в бинарном файле;
- поиск элемента в массиве по заданным свойствам объекта.

4. Демонстрационную программу, которая позволяет убедиться в корректности работы описанных функций.

Пояснительная записка расчётно-графической работы должна содержать следующие элементы:

- титульный лист;
- описание создания функций и демонстрационного приложения;
- листинг функций и демонстрационного приложения;
- описание процесса проверки работоспособности и снимки экранов тестовых запусков.

Пояснительную записку необходимо оформить в соответствии с методическими указаниями к оформлению документов кафедры «ИиПО».

3. Задачи для самостоятельного решения

Заданчи определяются согласно списку из лабораторной работы №15.

4. Контрольные вопросы

1. Какие можно выделить преимущества сохранения структурированных данных в текстовый файл?
2. Какие можно выделить недостатки сохранения структурированных данных в текстовый файл?
3. Какие можно выделить преимущества сохранения структурированных данных в бинарный файл?
4. Какие можно выделить недостатки сохранения структурированных данных в бинарный файл?

СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная литература

1. Костюкова, Н.И. Программирование на языке Си [Электронный ресурс]: методические рекомендации и задачи по программированию / Н.И. Костюкова. – Электрон. текстовые данные. – Новосибирск: Сибирское университетское издательство, 2017. – 160 с. – 978-5-379-02016-3. – Режим доступа: <http://www.iprbookshop.ru/65289.html>. – ЭБС «IPRbooks»
2. Поляков, А.Ю. Программирование [Электронный ресурс] : практикум / А.Ю. Поляков, А.Ю. Полякова, Е.Н. Перышкова. – Электрон. текстовые данные. – Новосибирск: Сибирский государственный университет телекоммуникаций и информатики, 2015. – 55 с. – 2227-8397. – Режим доступа: <http://www.iprbookshop.ru/55494.html>. – ЭБС «IPRbooks»
3. Иванова, Г.С. Программирование: основы алгоритмизации и процедурное программирование, объектно-ориентированное программирование: учебник для вузов. – 2-е изд., стер. – М.: Кнорус, 2014. – 425 с. – (Бакалавриат). – 588 с.
4. Керниган Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М.: Вильямс, 2017. – 288 с.

Дополнительная литература

1. Шилдт, Г. Полный справочник по Си / Г. Шилдт. – М.: Вильямс, 2009. – 704 с.
2. Подбельский, В.В. Программирование на языке Си / В.В. Подбельский, С.С. Фомин. – М.: Финансы и статистика, 2009. – 600с.
3. Голицына, О.Л. Основы алгоритмизации и программирования: учеб. пособие/ О.Л. Голицына, И.И. Попов. – М.: Форум, 2010. – 432 с.
4. Дейтел, Х. Как программировать на С: [пер. с англ.]/ Х. Дейтел, П. Дейтел. – М.: БИНОМ-ПРЕСС, 2010. – 1456с.
5. Булатицкий, Д.И. Алгоритмические языки и программирование: учеб. пособие/ Д.И. Булатицкий. – Брянск: БГТУ, 2005.– 76 с.

6. Уэйт, М. Язык Си. Руководство для начинающих: [пер. с англ.]/М. Уэйт, С. Прайта, Д. Мартин. – М.: Мир, 1988. – 512с.
7. Давыдов, В.Г. Программирование и основы алгоритмизации: учеб. пособие для вузов/В.Г. Давыдов. – М.: Высш. шк., 2003. – 448 с.
8. Березин, Б.И. Начальный курс С и С++. /Б.И Березин, С.Б. Березин. – М.: Диалог-МИФИ, 2005. – 288с.
9. Павловская, Т.А. С/С++. Программирование на языке высокого уровня: учебник для вузов / Т.А Павловская. – СПб: Питер, 2007. – 460с.
10. Павловская, Т.А. С/С++. Структурное и объектно-ориентированное программирование: Практикум. / Т.А. Павловская, Ю. А. Щупак – СПб.: Питер, 2011. – 352 с.
11. Керниган, Б. Элементы стиля программирования: [пер. с англ.]/Б. Керниган, Ф. Плджер. – М.: Радио и связь 1984.
12. Ван Тассел, Д. Стил, разработка, разработка, эффективность, отладка и испытания программ: [пер. с англ.]/Д. Ван Тассел. – М.: Мир, 1981
13. Кутузов, А.С. Числовые ряды: учебное пособие / А.С. Кутузов, С.М. Серебрянский. – Челябинск: Челябинский государственный университет, 2010. – 51 с.

Ресурсы Интернета

1. <http://stackoverflow.com/> – форум программистов
2. <http://citforum.ru/programming/c/dir.shtml> – раздел сайта CITFORUM, посвященный курсу «Программирование на языке Си»
3. <http://www.intuit.ru/studies/courses/2193/67/info> – раздел сайта ИНТУИТ, посвященный курсу «Основы программирования»
4. <http://www.intuit.ru/studies/courses/43/43/info> – раздел сайта ИНТУИТ, посвященный курсу «Основы программирования на языке С»
5. <http://www.intuit.ru/studies/courses/40/40/info> – раздел сайта ИНТУИТ, посвященный курсу «Стили и методы программирования»
6. <http://kpolyakov.spb.ru/school/c.htm> – раздел сайта KPOLYAKOV, посвященный курсу «Язык программирования Си»

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	3
ГЛАВА 1. ОСНОВЫ ПРОГРАММИРОВАНИЯ	5
Лабораторная работа №1. Разработка простых алгоритмов	
и их запись различными способами	5
1. Цель лабораторной работы.....	5
2. Краткие теоретические сведения	5
3. Общая часть лабораторной работы	15
4. Индивидуальная часть лабораторной работы	21
5. Задачи для самостоятельного решения.....	22
6. Контрольные вопросы.....	24
Лабораторная работа №2. Основы работы в	
интегрированной среде MS Visual Studio	25
1. Цель лабораторной работы.....	25
2. Краткие теоретические сведения	25
3. Общая часть лабораторной работы	27
4. Индивидуальная часть лабораторной работы	32
5. Контрольные вопросы.....	33
Лабораторная работа №3. Программирование линейного	
алгоритма	34
1. Цель лабораторной работы.....	34
2. Краткие теоретические сведения	34
3. Общая часть лабораторной работы	37
4. Индивидуальная часть лабораторной работы	41
5. Задачи для самостоятельного решения.....	42
6. Контрольные вопросы.....	47
Лабораторная работа №4. Решение задач на целые числа	48
1. Цель лабораторной работы.....	48
2. Краткие теоретические сведения	48
3. Общая часть лабораторной работы	49
4. Индивидуальная часть лабораторной работы	51
5. Задачи для самостоятельного решения.....	51
6. Контрольные вопросы.....	55
ГЛАВА 2. ВЕТВЛЕНИЯ И ЦИКЛЫ. РЕКУРСИЯ.....	56
Лабораторная работа №5. Программирование простых	
ветвлений	56
1. Цель лабораторной работы.....	56

2. Краткие теоретические сведения	56
3. Общая часть лабораторной работы	57
4. Индивидуальная часть лабораторной работы	63
5. Задачи для самостоятельного решения	64
6. Контрольные вопросы.....	67
Лабораторная работа №6. Программирование	
множественного ветвления. Пользовательские функции	69
1. Цель лабораторной работы.....	69
2. Краткие теоретические сведения	69
3. Общая часть лабораторной работы	71
4. Индивидуальная часть лабораторной работы	73
5. Задачи для самостоятельного решения	75
6. Контрольные вопросы.....	77
Лабораторная работа №7. Использование циклов с	
предусловием	78
1. Цель лабораторной работы.....	78
2. Краткие теоретические сведения	78
3. Общая часть лабораторной работы	79
4. Индивидуальная часть лабораторной работы	82
5. Задачи для самостоятельного решения	82
6. Контрольные вопросы.....	85
Лабораторная работа №8. Использование циклов с	
постусловием	86
1. Цель лабораторной работы.....	86
2. Краткие теоретические сведения	86
3. Общая часть лабораторной работы	87
4. Индивидуальная часть лабораторной работы	89
5. Задачи для самостоятельного решения	89
6. Контрольные вопросы.....	91
Лабораторная работа №9. Использование циклов со	
счётчиком	92
1. Цель лабораторной работы.....	92
2. Краткие теоретические сведения	92
3. Общая часть лабораторной работы	93
4. Индивидуальная часть лабораторной работы	94
5. Задачи для самостоятельного решения	95
6. Контрольные вопросы.....	97

Лабораторная работа №10. Использование вложенных циклов	98
1. Цель лабораторной работы.....	98
2. Краткие теоретические сведения	98
3. Общая часть лабораторной работы	99
4. Индивидуальная часть лабораторной работы	100
5. Задачи для самостоятельного решения	101
6. Контрольные вопросы.....	102
Лабораторная работа №11. Числовые ряды	104
1. Цель лабораторной работы.....	104
2. Краткие теоретические сведения	104
3. Общая часть лабораторной работы	104
4. Индивидуальная часть лабораторной работы	106
5. Задачи для самостоятельного решения	106
6. Контрольные вопросы.....	109
Лабораторная работа №12. Использование рекурсии	110
1. Цель лабораторной работы.....	110
2. Краткие теоретические сведения	110
3. Общая часть лабораторной работы	111
4. Индивидуальная часть лабораторной работы	113
5. Задачи для самостоятельного решения	114
ГЛАВА 3. МАССИВЫ, СТРУКТУРЫ, ФАЙЛЫ	116
Лабораторная работа №13. Массивы	116
1. Цель лабораторной работы.....	116
2. Краткие теоретические сведения	116
3. Общая часть лабораторной работы	119
4. Индивидуальная часть лабораторной работы	125
5. Задачи для самостоятельного решения	126
6. Контрольные вопросы.....	127
Лабораторная работа №14. Текстовые файлы	128
1. Цель лабораторной работы.....	128
2. Краткие теоретические сведения	128
3. Общая часть лабораторной работы	130
4. Индивидуальная часть лабораторной работы	138
5. Контрольные вопросы.....	138
Лабораторная работа №15. Массивы структур.....	139
1. Цель лабораторной работы.....	139
2. Краткие теоретические сведения	139

3. Общая часть лабораторной работы	140
4. Индивидуальная часть лабораторной работы	147
5. Задачи для самостоятельного решения	147
6. Контрольные вопросы.....	149
Лабораторная работа №16. Бинарные файлы	150
1. Цель лабораторной работы.....	150
2. Краткие теоретические сведения	150
3. Общая часть лабораторной работы	152
4. Индивидуальная часть лабораторной работы	157
5. Контрольные вопросы.....	157
Лабораторная работа №17. Двумерные массивы	158
1. Цель лабораторной работы.....	158
2. Краткие теоретические сведения	158
3. Общая часть лабораторной работы	159
4. Индивидуальная часть лабораторной работы	165
5. Задачи для самостоятельного решения	165
6. Контрольные вопросы.....	171
Расчётно-графическая работа	172
1. Цель расчётно-графической работы.....	172
2. Задание к расчётно-графической работе.....	172
3. Задачи для самостоятельного решения	173
4. Контрольные вопросы.....	173
СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ.....	174

Учебное издание

**Дмитрий Иванович Булатицкий
Елизавета Викторовна Коптенок
Руслан Александрович Исаев
Алексей Олегович Радченко**

**ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ СИ:
ВЕТВЛЕНИЯ, ЦИКЛЫ, МАССИВЫ, ФАЙЛЫ**

Редактор издательства
Компьютерный набор

Л.Н. Мажугина
Е.В. Коптенок

Темплан 2018г., п. 19

Подписано в печать 20.02.19. Формат 60х84 1/16. Бумага офсетная. Офсетная
печать. Усл. печ.л. 10,46. Уч.-изд.л. 10,46. Тираж 300 экз. Заказ .

Издательство Брянского государственного технического университета
241035, г. Брянск, бульвар 50 лет Октября, 7, БГТУ, тел. 58-82-49
Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16