Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Л. А. АРТЮШИНА Т. В. СПИРИНА
Е. А. ТРОИЦКАЯ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Учебно-практическое пособие

Владимир 2019

УДК 004
ББК 32.81
А86

Рецензенты:
Кандидат технических наук
доцент кафедры информационно-математических технологий
и информационного права Российского университета транспорта (МИИТ)
Л. М. Груздева

Доктор технических наук, профессор
зав. кафедрой информационных систем и программной инженерии
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
И. Е. Жигалов

Артюшина, Л. А. Информационные технологии и основы
А86 объектно-ориентированного программирования : учеб.-практ. пособие / Л. А. Артюшина, Т. В. Спирина, Е. А. Троицкая ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2019. – 203 с.
ISBN 978-5-9984-0977-6

Рассматриваются основы объектно-ориентированного программирования, представлены примеры и иллюстрации, поясняющие теоретическое изложение материала. Более 200 упражнений позволяют проверить знания по всем рассматриваемым темам.

Предназначено для проведения практических и лабораторных работ студентами-бакалаврами технических направлений 15.03.04 – Автоматизация технологических процессов и производств (прикладной бакалавриат), 22.03.01 – Материаловедение и технологии материалов, 27.03.02 – Управление качеством (прикладной бакалавриат) в рамках дисциплин информационного блока, а также при изучении курсов, связанных с объектно-ориентированным программированием.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Табл. 17. Ил. 105. Библиогр.: 15 назв.

УДК 004
ББК 32.81

ISBN 978-5-9984-0977-6

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 5

Глава 1. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ 6
1.1. Специальные возможности текстового редактора MS Word 6
1.1.1. Вычисления в таблицах. Встроенные функции MS Word 6
Практическая работа № 1 «Вычисления в таблицах» 8
1.1.2. Работа с шаблонами 10
Практическая работа № 2 «Создание документов на основе шаблонов MS Word 2010 – 2013» 13
1.1.3. Вставка диаграмм и рисунков 14
Практическая работа № 3 «Работа с графическими объектами MS Word 2010 – 2013» 22
Вопросы для самоконтроля 22
Список литературы 23
1.2. Возможности MS Excel для моделирования различных задач 23
1.2.1. Математические функции MS Excel 23
1.2.2. Логические функции MS Excel 30
1.2.3. Встроенные функции MS Excel 31
Практическая работа № 4 «Составление таблиц истинности с помощью табличного процессора MS Excel» 32
1.2.4. Статистические функции MS Excel 33
Практическая работа № 5 «Статистические функции MS Excel» 35
1.2.5. Условное форматирование в MS Excel 35
Практическая работа № 6 «Использование условного форматирования в MS Excel» 46
1.2.6. Календарные функции MS Excel 47
Практическая работа № 7 «Календарные функции MS Excel» 48
1.2.7. Контроль ввода данных в MS Excel 48
Практическая работа № 8 «Контроль ввода данных в MS Excel» 52
Вопросы для самоконтроля 53
Список литературы 53

1.3. Информационные системы и базы данных 54
1.3.1. Основные понятия СУБД 54
1.3.2. Способы оптимизации работы таблиц 56
1.3.3. Способы обработки информации в БД 61
Практическая работа № 9 «Разработка базы данных средствами MS Microsoft Access» 80
Вопросы для самоконтроля 83
Список литературы 83

Глава 2. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ 84
2.1. Введение в практический курс 84
2.2. Практическая работа № 1 «Создание консольного приложения» 88
2.3. Практическая работа № 2 «Работа со встроенным отладчиком Microsoft Visual C++ 2015» 95
2.4. Практическая работа № 3 «Объявление и инициализация переменных. Стандартные типы данных» 97
2.5. Практическая работа № 4 «Функции и операторы ввода-вывода» 102
2.6. Практическая работа № 5 «Сокращенные варианты записи» 108
2.7. Практическая работа № 6 «Оператор условия IF-ELSE» 111
2.8. Практическая работа № 7 «Циклы» 117
2.9. Практическая работа № 8 «Структуры» 125
2.10. Практическая работа № 9 «Функции» 133
2.11. Практическая работа № 10 «Одномерные массивы» 147
2.12. Практическая работа № 11 «Двумерные массивы» 162
2.13. Практическая работа № 12 «Объекты и классы» 173

СПИСОК ЛИТЕРАТУРЫ 182

ЗАКЛЮЧЕНИЕ 183

ПРИЛОЖЕНИЯ 184

ГЛОССАРИЙ 188

ПРЕДИСЛОВИЕ

Данное пособие предназначено преподавателям дисциплин информационного цикла для бакалавров высшего образования технического профиля.

Содержит материалы к занятиям по изучению специальных возможностей программ пакета Microsoft Office, а также одного из наиболее популярных и перспективных языков программирования – C++. Однако, только этим мы не ограничились. В последнее десятилетие в сфере программного обеспечения произошел ряд значительных изменений. Поэтому еще одна задача данного учебного пособия состоит в изложении концепции языка C++ в контексте развития программного обеспечения. Так как пособие представляет собой только часть курса программирования на языке C++, в нем излагаются основы концепции.

Организация занятий с использованием предложенного материала предполагает знание студентами основ информатики в объеме базового уровня школьной программы, а также наличие опыта работы с персональным компьютером в операционной системе Windows и с программами пакета Microsoft Office.

Пособие ориентировано на использование программных пакетов Microsoft Office 2013 и Visual Studio 2015 или выше и в качестве введения содержит описание полного цикла создания консольного приложения с использованием этой системы. В то же время представленный материал может быть легко адаптирован для использования вместе с другими версиями компиляторов, необходимыми авторам для написания программ.

Пособие организовано в виде набора тем. К каждой теме предлагаются вопросы для самоконтроля и индивидуальные варианты заданий для самостоятельной работы, выполняя которые, студенты закрепят теоретические знания на практике. Пособие также содержит необходимый справочный материал и тексты примеров программ в помощь студентам.

Глава 1. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1.1. Специальные возможности текстового редактора MS Word

1.1.1. Вычисления в таблицах. Встроенные функции MS Word

Таблица – форма организации данных по столбцам и строкам, на пересечении которых находятся ячейки, в которых могут содержаться различные данные. Ячейки таблицы имеют адреса, образованные именем столбца (A, B, C, D…) и номером строки (1, 2, 3…). Движение курсора между ячейками таблицы можно осуществлять по нажатию клавиши *Tab*.

Word позволяет выполнить вычисления, записывая в отдельные ячейки таблицы формулы с помощью команды *Работа с таблицами, Макет, Формула* задается как выражение, в котором могут быть использованы:

1. *Абсолютные ссылки* на ячейки таблицы в виде списка (разделенные знаком «;» - A1; B5; E10 и т.д.) или блока (начало и конец блока ячеек – A1:A10);

2. *Ключевые слова* для ссылки на блок ячеек:

- LEFT – ячейки, расположенные в строке левее ячейки с формулами;
- RIGHT - ячейки, расположенные в строке правее ячейки с формулами;
- ABOVE - ячейки, расположенные в столбце выше ячейки с формулами;
- BELOW - ячейки, расположенные в столбце ниже ячейки с формулами;

3. *Константы* – числа, текст в двойных кавычках;

4. *Закладки*, которым соответствует определенный текст документа (например, числа), созданный с помощью команды *Вставка\Ссылки\Закладка*;

5. *Встроенные функции* MS Word, например, SUM(), AVERAGE();

6. *Знаки операций* (+, -, *, /, %, =, <, >,<=, >=).

Таблица 1. Виды встроенных функций

Категория	**Функция**	**Назначение**
Статистические	AVERAGE ()	Вычисление сред. значение для диапазона ячеек, например: =AVERAGE (A1:C20)
	COUNT ()	Подсчёт числа значений в указанном диапазоне ячеек, например: = COUNT (A1:C20; B25; A30)
	MAX ()	Нахождение максимального значения в указанном блоке ячеек, например: =MAX (A1:C20; B25; A30)
	MIN()	Нахождение мин-ого знач. в указанном блоке ячеек, например: =MIN (A1:C20; B25; A30)
	SUM()	Нахождение суммы чисел в указанном блоке ячеек, например: =SUM (A1:C20; B25; A30)
Математические	ABS(x)	Абсолютное значение вычисляемого выражения, например: = ABS (A1*B12-C25+100)
	MOD(x, y)	Остаток от деления первого числа на второе, например: = MOD (A1,C12)
	INT(x)	Целая часть числа, например: = INT(234.45)
	PRODUCT()	Произведение чисел в указанном диапазоне ячеек, например: = PRODUCT(A1:C20; B25; A30)
	ROUND(x, y)	Округление значения до указанного числа знаков, например, округлить до сотен: = ROUND(2345.45.-2)
	SIGN(x)	Определение знака числа, например (-1 для отрицательных и 1 для положительных): = SIGN(-2345.45)
Логические	IF(x,y,z)	Проверка заданного условия и присвоения значения ячейке: если условие истинно - *значение 1*, иначе *значение 2:*= IF (E12>G12; *значение 1;значение 2)*
	AND(x,y)	Вычисляет значение 1, если заданы истинные значения логических аргументов, иначе – 0, например: = AND (A4>3; B3<3)
	OR(x,y)	Вычисляет значение 0, если заданы истинные значения любого логического аргумента, иначе – 1, например: = OR (A2>3; D3<=4)
	NOT(x)	Вычисляет значение 0, если заданы истинное значение логического аргумента, иначе – 1, например: = NOT (D4>2)
	FALSE	Логическая константа ***ложь***, которой соответствует число 0.
	TRUE	Логическая константа ***истина***, которой соответствует число 1.
	DEFINED(x)	Определяет значение в ячейке.

Практическая работа № 1 «Вычисления в таблицах»

Задание 1. Создайте таблицу и произведите вычисления успеваемости студентов:

№ ПП	Учебная дисциплина	Группа	Средний балл	Всего сдавало	Отл.	Хор.	Удов.	Неуд.	Неявки
1		51			12	10	6	3	1
2		52			7	9	6	3	2
3		53			9	8	3	5	3
4		54			8	8	8	3	2
Итого:									

Задание 2. Добавьте в созданную таблицу столбец *Качество знаний студентов*.

№ ПП	Учебная дисциплина	Группа	Средний балл	Всего сдавало	Отл.	Хор.	Удов.	Неуд.	Неявки	Качество знаний
1		51			12	10	6	3	1	
2		52			7	9	6	3	2	
3		53			9	8	3	5	3	
4		54			8	8	8	3	2	
Итого:										

Установите с помощью встроенных логических функций Word значение этого столбца 1, если количество студентов, успевающих на хорошо и отлично, больше 50% от общего количества сдававших, и 0 – в противном случае.

Задание 3. Как-то раз во время каникул, несколько студентов отправились путешествовать на разных видах транспорта. О том кто, на чём путешествовал и сколько километров проехал, показывает таблица. Необходимо создать следующую таблицу и вычислить:

1. Сколько километров студенты проплыли на катере?

2. Какое расстояние преодолел Николай на всех видах транспорта?

3. Какое расстояние студенты преодолели с помощью поезда и автомобиля?

4. Какое расстояние студенты преодолели все вместе во время путешествия?

5. Сколько километров в среднем пропутешествовал, каждый из студентов?

6. Найти минимальное расстояние, преодолённое на самолёте.

7. Найти максимальное расстояние, преодолённое Евгением.

№	Студенты	Транспорт					Расстояние, которое преодолел Николай на всех видах транспорта
		катер	самолёт	Поезд	Автомобиль	Велосипед	
1	Витя	80	25	15	45	4	
2	Михаил	10	73	54	39	1	
3	Николай	60	24	37	53	7	
4	Павел	30	18	10	85	2	Максимальное расстояние, преодолённое Евгением на одном виде транспорта
5	Петр	15	58	14	12	8	
6	Анатолий	40	32	63	77	3	
7	Евгений	70	61	8	126	5	
		Расстояние, которое студенты проплыли на катере	Минимальное расстояние, преодолённое на самолёте	Расстояние, которое студенты проехали на поезде и автомобиле		Среднее расстояние, которое проехал каждый из студентов	Общее расстояние, преодолённое всеми студентами во время путешествия

1.1.2. Работа с шаблонами

В Word есть множество шаблонов документов, таких как, например, шаблоны открыток, конвертов, различных бланков, календарей и визиток. Они очень полезны, если вы не умеете / не хотите пользоваться графическими редакторами, или, если у вас нет достаточного количества времени для составления уникального документа с оформлением.

Несомненный плюс использования данной функции Word состоит в том, что документ уже оформлен, и остается только вписать недостающие данные.

Список некоторых из рекомендуемых шаблонов:

1. важное письмо;
2. важное резюме;
3. важный отчет;
4. резюме;
5. современное письмо;
6. современное резюме;
7. студенческий отчет.

Шаблоны создаются с помощью команды *Файл, Создать* (рис.1).

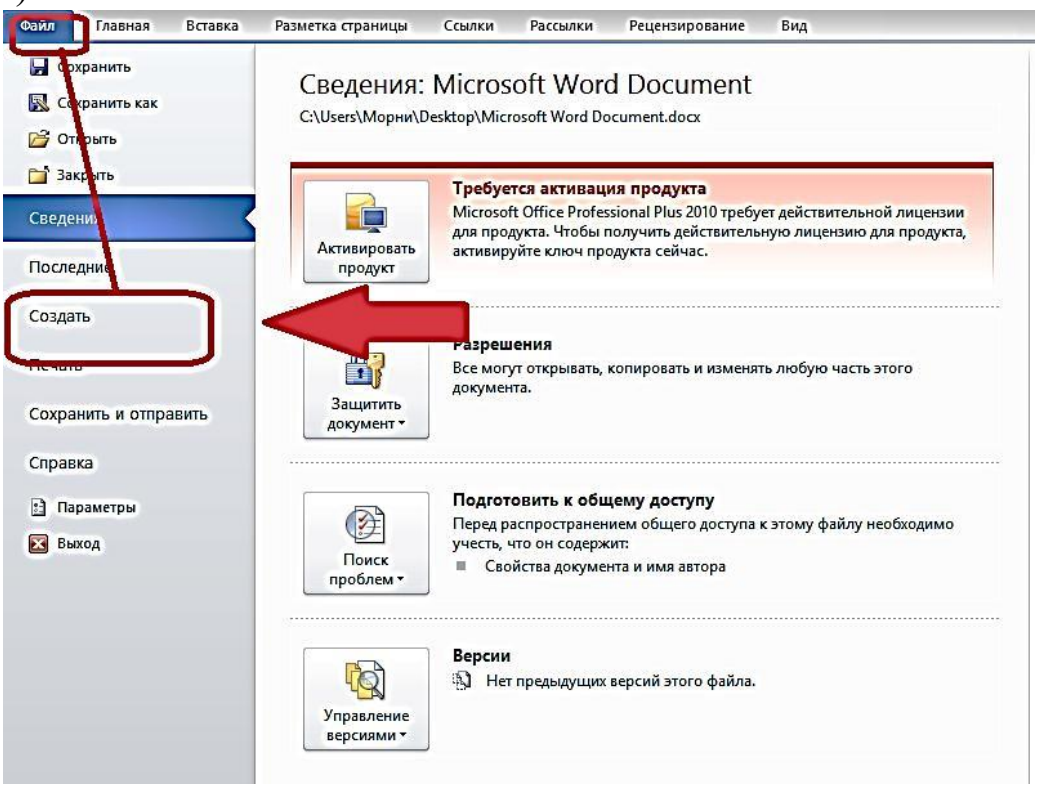

Рис. 1. Создание шаблона документа

Откроется окно с шаблонами, готовыми к использованию (рис.2).

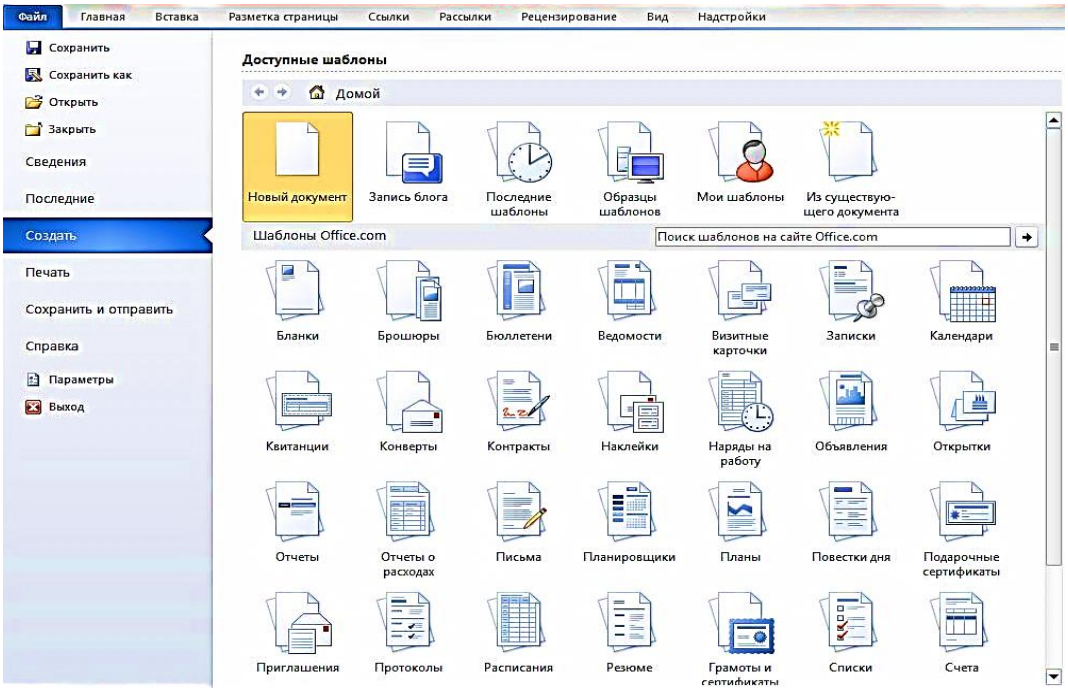

Рис. 2. Стандартные шаблоны

Для того чтобы открыть шаблон, достаточно выбрать соответствующую иконку и выгрузить шаблон в Word (рис. 3,4). Например, *Создать/ Приглашения/ Приглашения на мероприятия/ Приглашение на праздник/ Загрузить.*

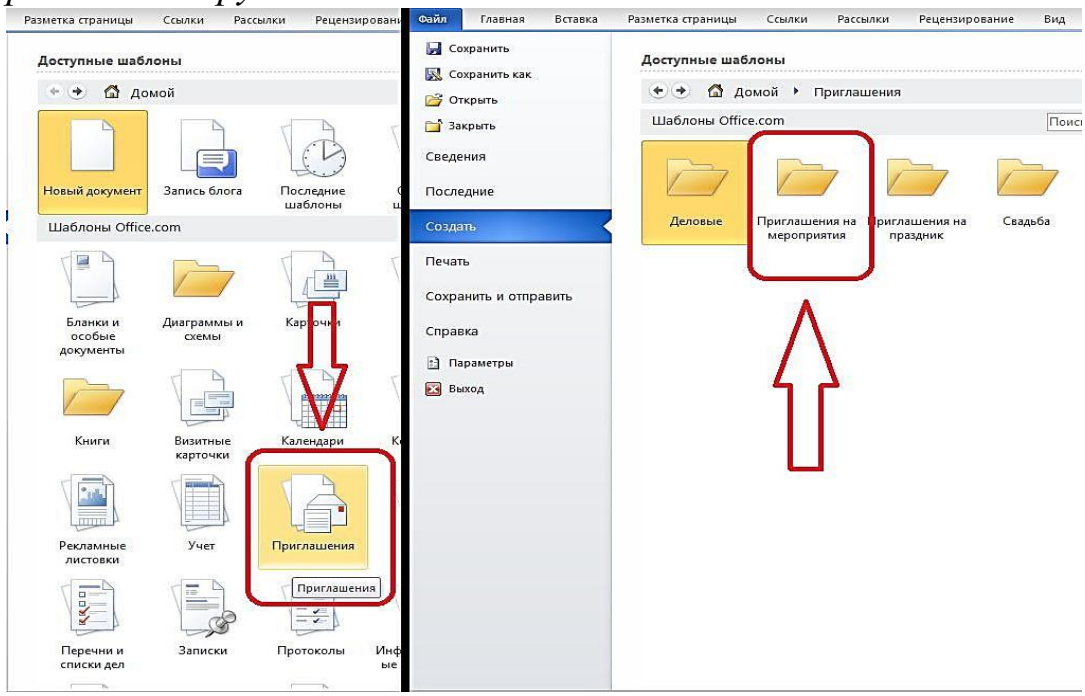

Рис. 3. Открытие шаблона

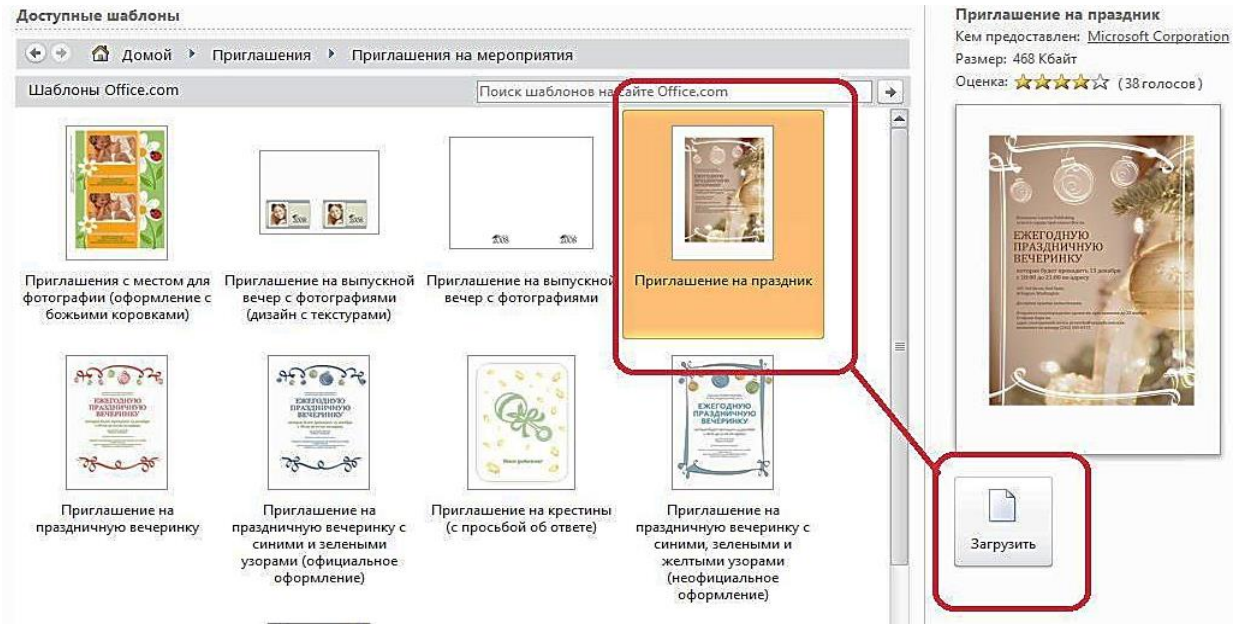

Рис. 4. Выгрузка шаблона в Word

Создание собственных шаблонов документов

Форма – это защищенный документ особого рода, который содержит поля для ввода информации. Любой документ, который содержит поля формы, считается формой.

Поле формы – это особое поле в документе, которое позволяет выполнить одно из трех действий: ввести текст, сбросить/установить флажок, выбрать значение из раскрывающегося списка.

Структура многих форм задается с помощью таблиц, потому что ячейки таблицы отлично подходят для размещения, как меток полей, так и информационных полей. Кроме того, таблицы позволяют ввести в форму затенение и обрамление.

Построение формы происходит в три этапа:

- создать новый шаблон и построить структуру формы, то есть задать текст, заливку, обрамление, форматирование, т.е. все то, что не меняется в процессе заполнения формы.
- вставить поля формы в тех местах, где должен быть предусмотрен ввод данных при заполнении формы.
- защитить и сохранить форму.

Создать форму можно выбрав последовательно вкладки *Разработчик, Элементы управления* (рис.5).

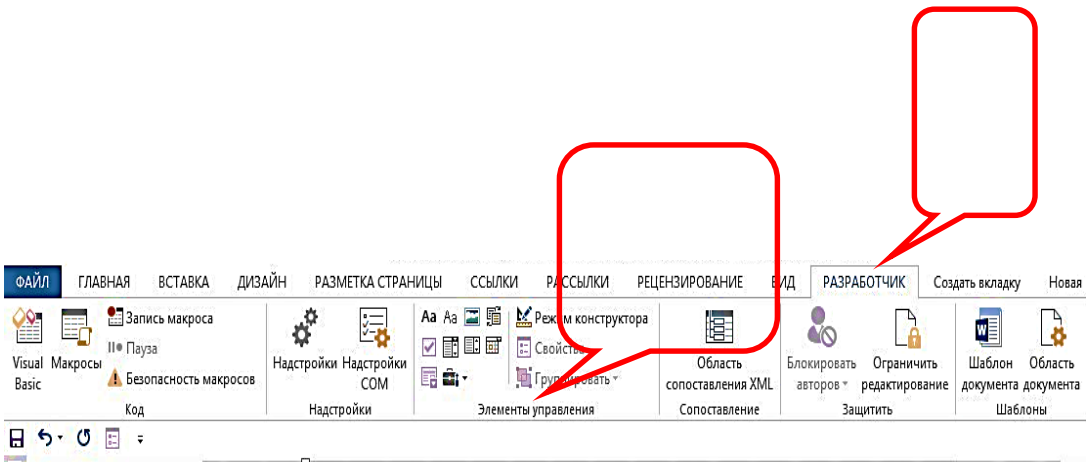

Рис. 5. Создание формы

Практическая работа № 2
«Создание документов на основе шаблонов MS Word 2010 – 2013»

Цель: научиться использовать стандартные и создавать собственные шаблоны, приобрести навыки в профессиональном оформлении документа.

Задание 1. Создайте документ на основе стандартного шаблона «Резюме», встроенного в MS Word.

Задание 2. Создайте собственный шаблон документа – контрольного теста по физике по образцу:

Контрольный тест по теме:
«Удельная теплоемкость»

1. *Напишите определение удельной теплоемкости вещества* [redacted]
2. *Чему равна удельная теплоемкость воды* $\frac{дж}{кг \cdot град}$ *4200*

(выберите правильный ответ).

3. *Чему равна удельная теплоемкость льда в* $\frac{дж}{кг \cdot град}$ *4100* [redacted]
4. Напишите, какая существует связь между удельной теплоемкостью вещества и изменением внутренней энергии [redacted]
5. Напишите, почему близость водоемов влияет на температуру воздуха? [redacted]

Порядок выполнения

1. Создайте файл **test**.
2. Откройте созданный файл.
3. Введите заголовок: «*Контрольный тест по теме:* «*Удельная теплоемкость*»
4. Создайте в поле документа нумерованный список вопросов теста.
5. В первом, четвертом и пятом вопросах теста для написания ответов установите вид формы **Текстовое поле**.
6. Во втором и третьем вопросах необходимо выбрать правильный ответ из формы **Поле со списком**

1.1.3. Вставка диаграмм и рисунков

Диаграммы строятся на основе данных, содержащихся в таблице, также внедряемой в документ Word. Созданная диаграмма связывается с таблицей данных, поэтому при изменении исходных данных диаграмма автоматически обновляется (рис.7,8). Можно создавать диаграммы четырнадцати основных и двадцати дополнительных типов. Кроме того, внутри каждого из основных типов можно выбрать конкретный формат (подтип).

Для построения диаграммы необходимо последовательно выбрать *"Диаграмма"* на вкладке "Иллюстрации" ленты "Вставка" (рис.6).

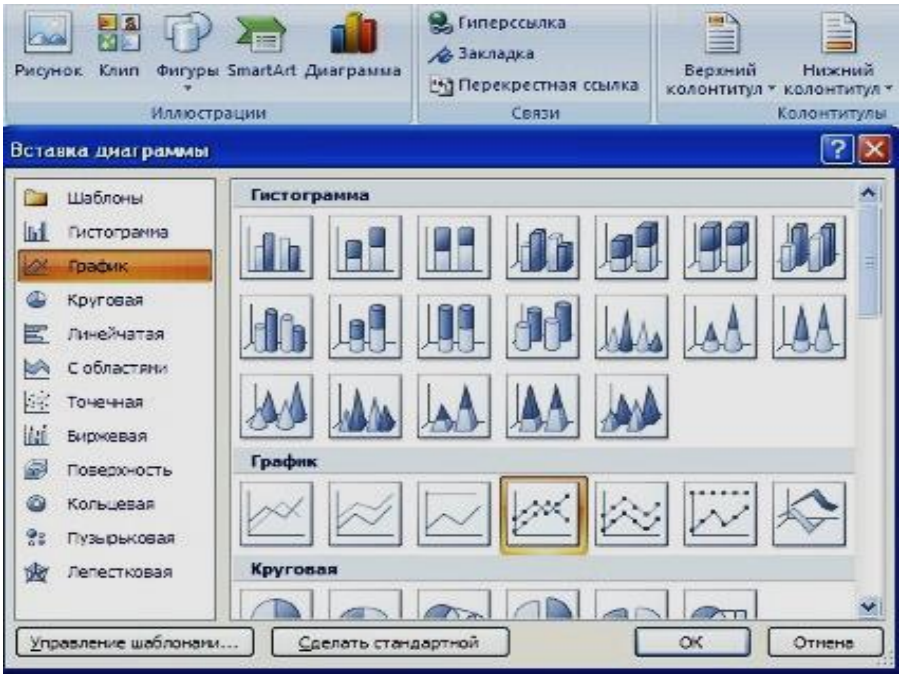

Рис. 6. Виды диаграмм вкладки Иллюстрация

В появившемся окне надо выбрать тип диаграммы и ее вид. После этого, автоматически открывается окно программы Excel с набором некоторых стандартных значений для построения графика (рис.7,8). Необходимо ввести данные для построения графиков. При необходимости можно удалить, или добавить диаграмму.

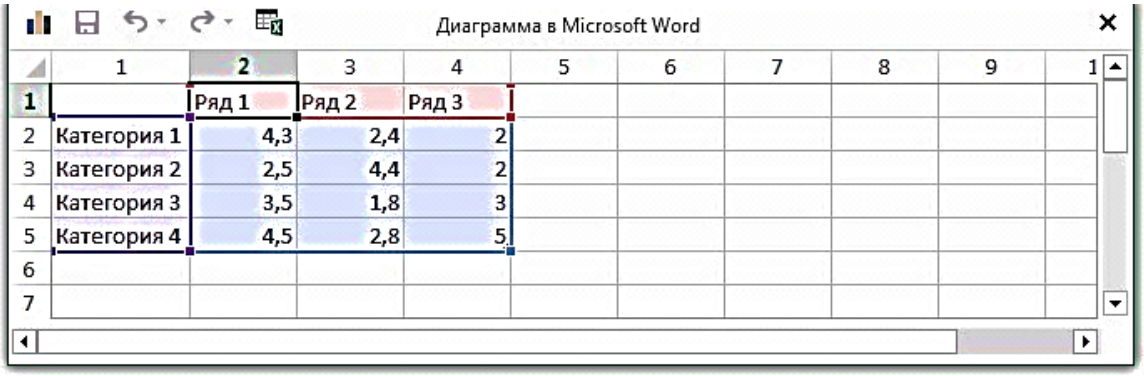

Рис. 7. Окно стандартных значений для построения графика

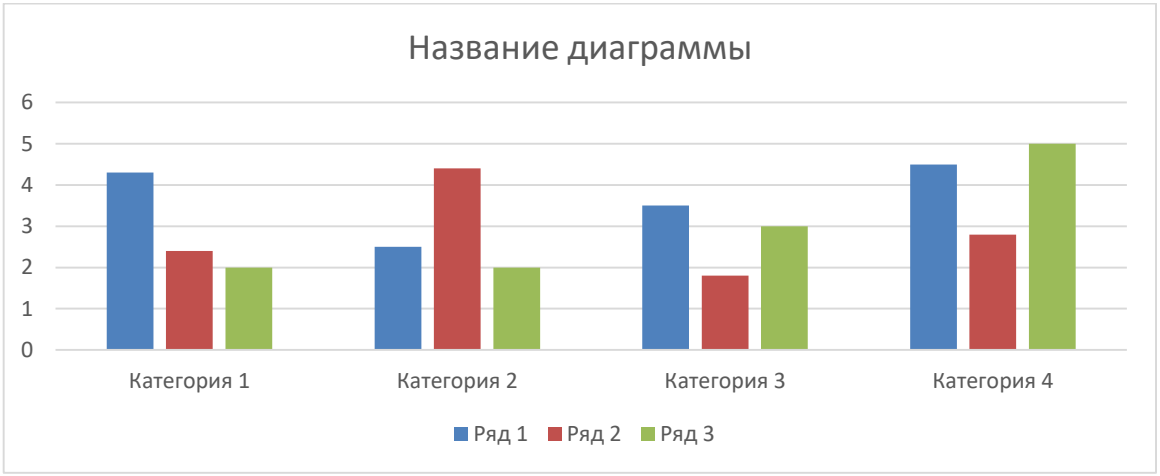

Рис. 8. Стандартная диаграмма

В документе Word 2010-2013 появится построенная диаграмма. При этом в окне редактора появляется контекстный инструмент «Работа с диаграммами», содержащий три ленты: «Конструктор», «Макет», «Формат» (рис.9).

Рис. 9. Контекстный инструмент «Работа с диаграммами»

Кратко охарактеризуем контекстные инструменты для работы с диаграммами.

Лента «Конструктор» состоит из четырех панелей: *«Тип»*, *«Данные»*, *«Макеты диаграмм"*, *"Стили диаграмм"* (рис.10,11,12). Основные операции, выполняемые этими инструментами: изменение вида диаграммы, её данных и стиля.

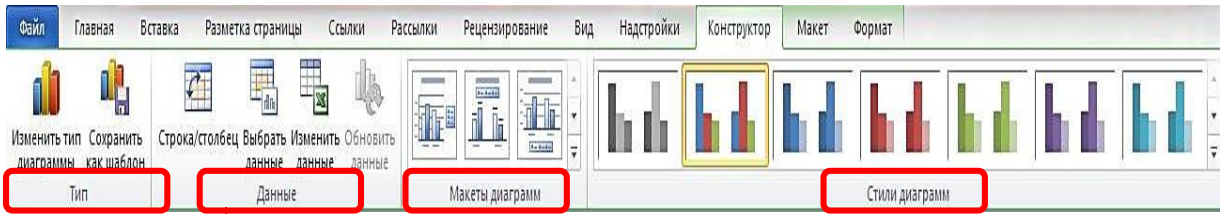

Рис. 10. Панель «Макеты диаграмм»

Лента "Макеты" содержит шесть панелей: *"Текущий фрагмент"*, *"Вставить"*, *"Подписи"*, *"Оси"*, *"Фон"*, *"Анализ"*. Эти инструменты предназначены для непосредственного оформления графиков диаграмм и отдельных элементов диаграммы. Для выбора элемента диаграммы служит выпадающий список *"Текущий фрагмент"* (рис.11).

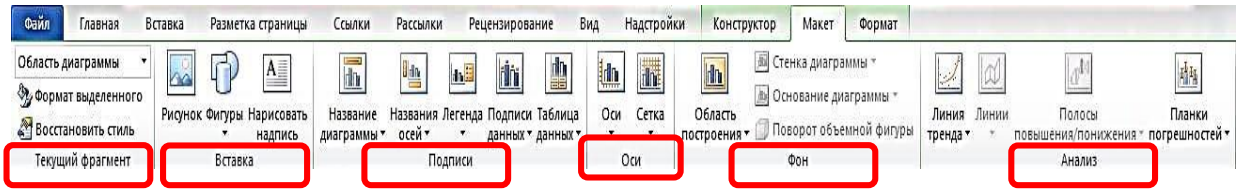

Рис. 11. Выбор элемента диаграммы

Лента "Формат" содержит инструменты для придания диаграмме окончательного вида (рис.12).

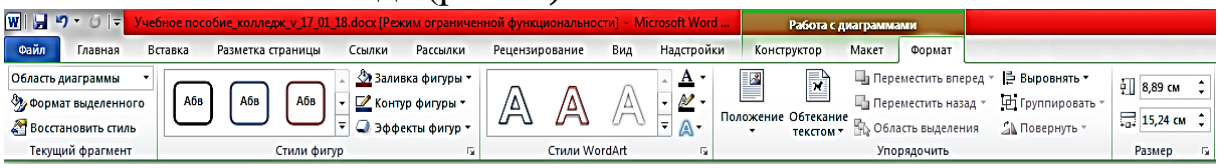

Рис. 12. Лента Формат вкладки Диаграммы

Графика - это один из важнейших элементов документа Word. Графика бывает двух видов - растровая и векторная. **Растровая графика** в Word может быть загружена из графического файла (с расширением BMP, TIFF, PNG, JPG или GIF) или из другой программы

(например, графического редактора Adobe Photoshop). **Векторная графика** может быть создана в документе Word или вставлена в документ с помощью встроенных графических средств Word.

В документ Word 2010-2013 можно вставить следующие типы **графики** (рисунок, клип, графические объекты, рисунок SmartArt, диаграмма) с помощью кнопок Рисунок, Клип, Фигуры, SmartArt и Диаграмма, расположенных на вкладке "Вставка" в группе "Иллюстрации" (рис.13).

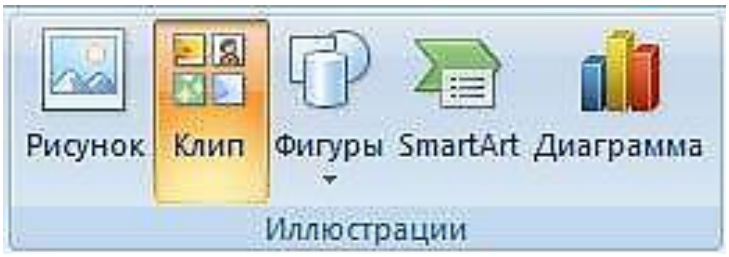

Рис. 13. Вкладка Иллюстрации

Кроме того, графические объекты или векторную графику Надпись и WordArt можно вставить из группы "Текст" на вкладке Вставка (рис.14).

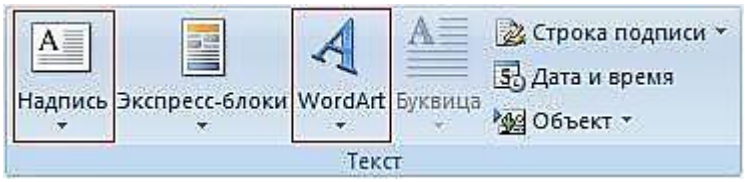

Рис. 14. Вкладка Текст

После вставки графики в документ Word, на Ленте появятся контекстно-зависимые инструменты под общим названием, которое отображается в строке заголовка окна приложения. Контекстные инструменты, разделенные на контекстные вкладки, появляются только тогда, когда в документе выделен объект определенного типа.

- Формат в группе "Работа с рисунками" (вставка растровых рисунков из файла и клипа);
- Формат в группе "Средства рисования" (вставка в документ готовых фигур);
- Конструктор, Формат в группе "Работа с рисунками SmartArt" (вставка рисунка SmartArt для визуального представления информации);

- Конструктор, Макет, Формат в группе "Работа с диаграммами" (вставка диаграммы для представления и сравнения данных);
- Формат в группе "Работа с надписями" (вставка предварительно отформатированных надписей);
- Формат в группе "Работа с объектами WordArt" (вставка декоративного текста в документ).

Растровые рисунки (растровую графику) и клипы можно вставлять или копировать в документ из множества различных источников. Растровые рисунки создаются различными графическими приложениями или техническими средствами (сканерами, фотоаппаратами и т.д.) и вставляются в документ Word из файла или прикладной программы. Вставку графики в Word осуществляют в то место документа, где установлен курсор.

Вставка растрового рисунка из файла в документ Word

Вставка рисунка осуществляется следующим образом: в документе надо определить место вставки рисунка, установив там курсор, затем щелкнуть на кнопке Рисунок на вкладке Вставка в группе Иллюстрации. В открывшемся окне диалога выбрать требуемый файл и дважды щелкнуть на нем, рисунок будет вставлен в документ. На Ленте окна приложения Word появятся контекстные инструменты с названием "Работа с рисунками", которые помещены на контекстной вкладке Формат (рис.15).

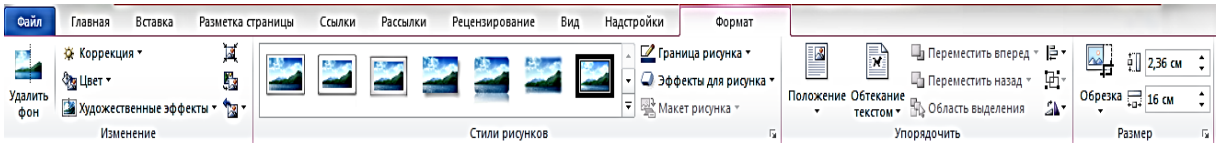

Рис. 15. Вкладка Работа с рисунками

Используя контекстные инструменты, имеющиеся в группах (Изменение, Стили рисунков, Упорядочить, Размер) на вкладке Формат можно выполнять различные действия над рисунками. Например, редактировать (изменять яркость, контрастность и т.д.), форматировать (применять различные стили), упорядочивать (определять положение рисунка и обтекание текстом), изменять размеры (изменять размеры, выполнять обрезку рисунка и замещение текста).

Если необходимо восстановить измененный рисунок в исходное состояние, надо щелкнуть на команде "Сброс параметров рисунка". Для выполнения любых операций над рисунком его надо предварительно выделить.

Вставка клипа в документ Word

Клип вставляется щелчком на кнопке *Клип* из группы Иллюстрации, в результате активизируется область задач. В области задач можно найти требуемый Клип в текстовом поле *Искать* или выбрать его из списка коллекций, щелкнув на команде "*Упорядочить клипы"*. Действия над клипами выполняются контекстными инструментами "*Работа с рисунками*", расположенными на контекстной вкладке Формат.

Вставка в документ готовых фигур

Вставку готовых фигур в документ Word 2010-2013 выполняют кнопкой Фигуры. Контекстные инструменты "Средства рисования", помещенные на вкладке Формат, которые появляются после вставки Фигуры, обеспечивают редактирование и форматирование готовых фигур, а также создание векторных рисунков из графических объектов (рис.16). Векторный рисунок, созданный из графических объектов, является графическим объектом.

Рис. 16. Инструмент «Средства рисования»

Необходимо отметить, что при создании векторного рисунка из графических объектов сначала следует вставить в документ полотно (Вставка/Фигура, затем выбрать "Новое полотно"), а затем размещать в нем фигуры и линии. Полотно способствует упорядочиванию рисунка и создает границу (рамку) между рисунком из графических объектов и остальной частью документа (рис.17). Для изменения размера полотна можно использовать контекстное меню.

Рис. 17. Вид графического полотна

Вставка рисунка SmartArt в документ Word

Рисунок SmartArt применяется для визуального представления информации (рис.18). Контекстные инструменты под названием "Работа с рисунками SmartArt" разделены на две контекстные вкладки Конструктор и Формат (рис.19), появившиеся после вставки объекта SmartArt, предназначены для редактирования и форматирования объектов визуальной информации.

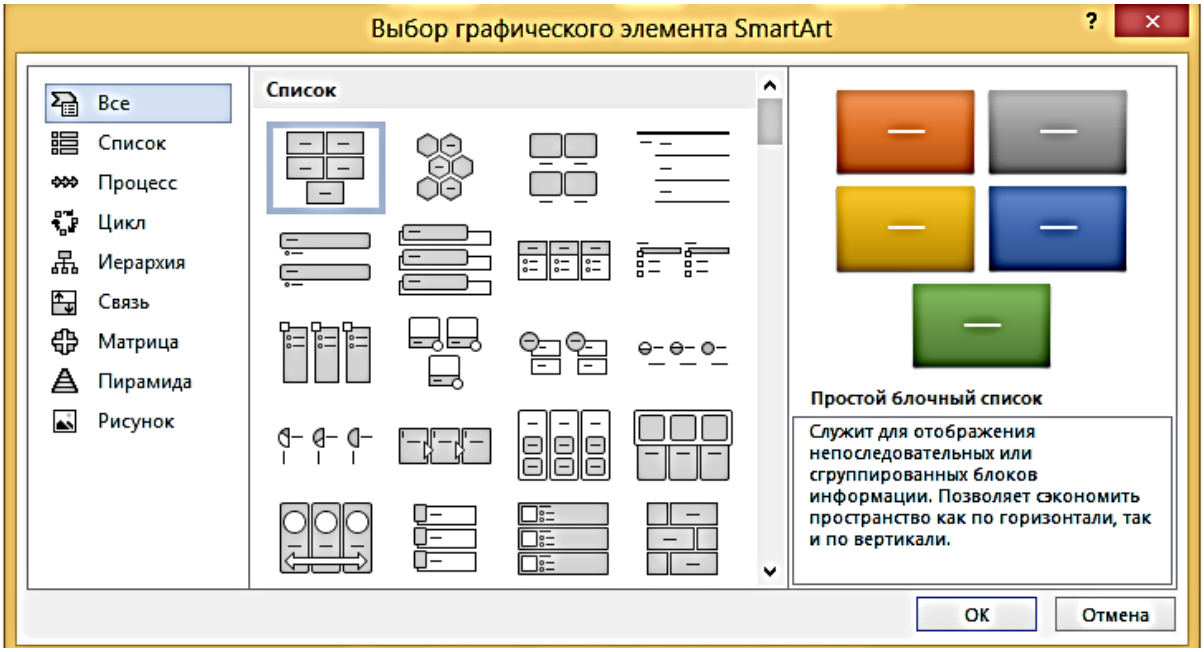

Рис. 18. Вкладка Виды элементов SmartArt

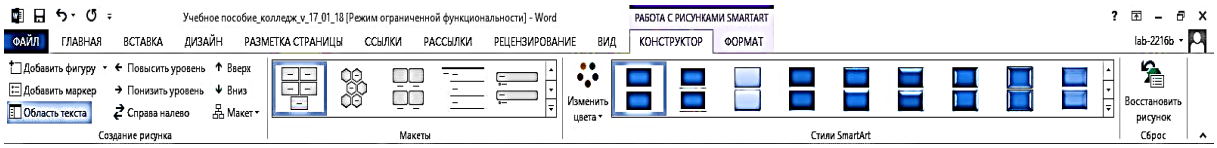

Рис. 19. Работа с элементами SmartArt

Вставка объекта Надпись в документ Word

Вставка предварительно отформатированных объектов Надпись (векторная графика) применяется для нестандартной вставки неболь-

ших текстов. Контекстные инструменты "Работа с надписями" вкладки Вставка используются для изменения размера и форматирования объекта, создания связи между несколькими объектами Надпись и для применения других эффектов (ри.20).

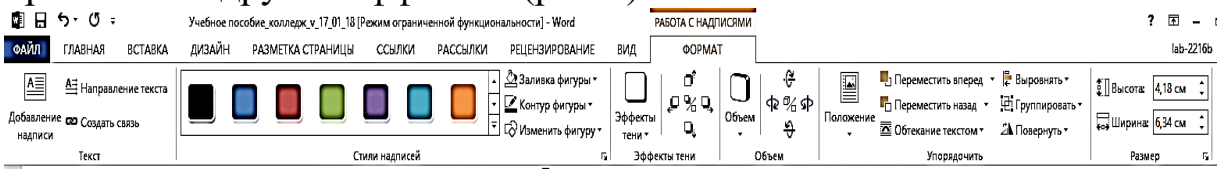

Рис. 20. Инструменты панели «Работа с надписями»

Вставка WordArt в документ Word

WordArt (векторная графика) вставляется из коллекции декоративных текстов для создания фигурного текста в документе. Виды объектов WordArt представлены на рис.21. Контекстные инструменты "Работа с объектами WordArt" на вкладке Текст предназначены для редактирования, форматирования и упорядочивания фигурного текста (рис.22).

Рис. 21. Виды объектов WordArt

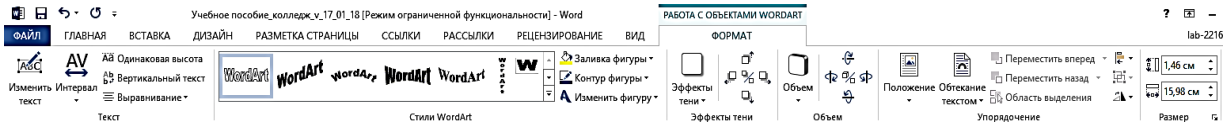

Рис. 22. Инструменты «Работа с объектами WordArt»

Для преобразования встроенного в текст рисунка (вставленного как символ текста) или другого графического объекта в перемещаемый (находящийся в графическом слое) необходимо выбрать один из стилей обтекания в окне "Обтекание текста" на контекстной вкладке *Положение*.

Практическая работа № 3
«Работа с графическими объектами MS Word 2010 – 2013»

Задание 1. Создайте таблицу согласно указанному образцу и постройте на её основе диаграмму, содержащую следующие элементы: название таблицы, легенда, метки данных.

Высочайшие и наиболее известные водопады мира

Местоположение	Название	Высота, м
Юж. Америка	Анхель	1054
Африка	Тугела	933
Сев. Америка	Йосемитский	727
Евразия	Утигард	610
Океания	Сатерленд	580
Африка	Виктория	120
Юж. Америка	Игуасу	72
Сев. Америка	Ниагарский	51
Африка	Бойома	40

Задание 2. Создайте рисунок, согласно образцу, представленному ниже на рис.23.

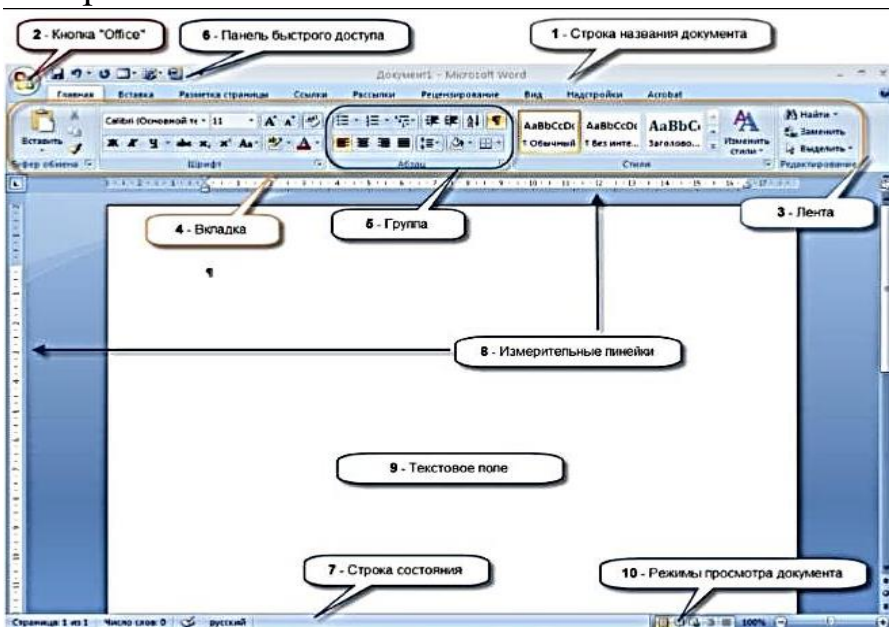

Рис. 23. Образец рисунка

Вопросы для самоконтроля

1. Назовите средство текстового редактора Word, предназначенное для создания художественных заголовков. Перечислите: возможности доступа к средству, способы ввода заголовка текста, основные

элементы управления панелью средства и заголовком в составе документа, основные свойства редактирования объекта в составе документа.

2. Назовите средство текстового редактора Word, предназначенное для ввода и редактирования формул. Перечислите: способы запуска средства, порядок работы со средством, основные элементы панели инструментов средства.
3. Назовите основные средства создания таблиц в текстовом редакторе Word.Перечислите основные операции редактирования структуры таблицы.
4. Перечислите основные операции форматирования структуры таблицы.
5. Перечислите основные операции форматирования содержимого ячеек таблицы.
6. Назовите основные методы создания диаграмм в Word.
7. Расскажите, как осуществляется настройка внешнего вида диаграммы.

Список литературы

1. Как делать вычисления в таблицах Word - http://prowebwriting.ru/kak-delat-vyichisleniya-v-tablitsah-word-2016/
2. Работа с шаблонами в Word https://www.youtube.com/watch?v=4T1yIN63F8M
3. Microsoft Word 2010: от новичка к профессионалу [Электронный ресурс] / Несен А.В. - М. : ДМК Пресс, 2011. - (Серия "Библиотека профессионала")." - http:// www.studentlibrary.ru/ book/ ISBN9785940747130.html

1.2. Возможности MS Excel для моделировании различных задач

1.2.1. Математические функции MS Excel

Математические функции используются для произведения различных арифметических и алгебраических действий. Их часто используют при планировании и в научных вычислениях. Рассмотрим наиболее популярные математические функции.

СУММ (). Этот оператор предназначен для сложения данных в нескольких ячейках. Хотя его можно использовать и для обычного суммирования чисел. Синтаксис, который можно применять при ручном вводе, выглядит следующим образом: =СУММ (число1; число2;…)

В окне аргументов в поля следует вводить ссылки на ячейки с данными или на диапазоны (рис.24). Оператор складывает содержимое и выводит общую сумму в отдельную ячейку.

СУММЕСЛИ (). Оператор также подсчитывает общую сумму чисел в ячейках. Но, в отличие от предыдущей функции, в данном операторе можно задать условие, которое будет определять, какие именно значения участвуют в расчете, а какие нет. При указании условия можно использовать знаки «>» («больше»), «<» («меньше»), «< >» («не равно»). То есть, число, которое не соответствует заданному во втором аргументе условию, при подсчете суммы не будет браться в расчёт.

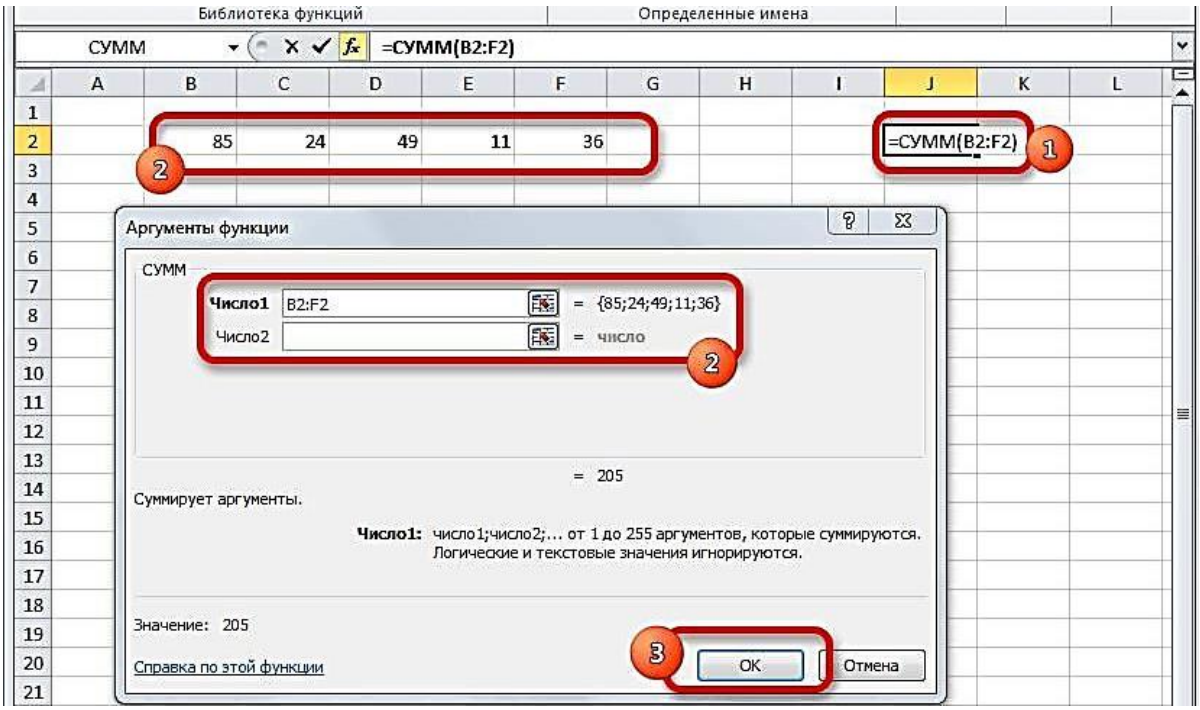

Рис. 24. Пример использования функции СУММ ()

Кроме того, существует дополнительный аргумент *«Диапазон суммирования»*, не являющийся обязательным. Данная операция имеет следующий синтаксис: =СУММЕСЛИ (Диапазон; Критерий; Диапазон_суммирования) (см. рис.25)

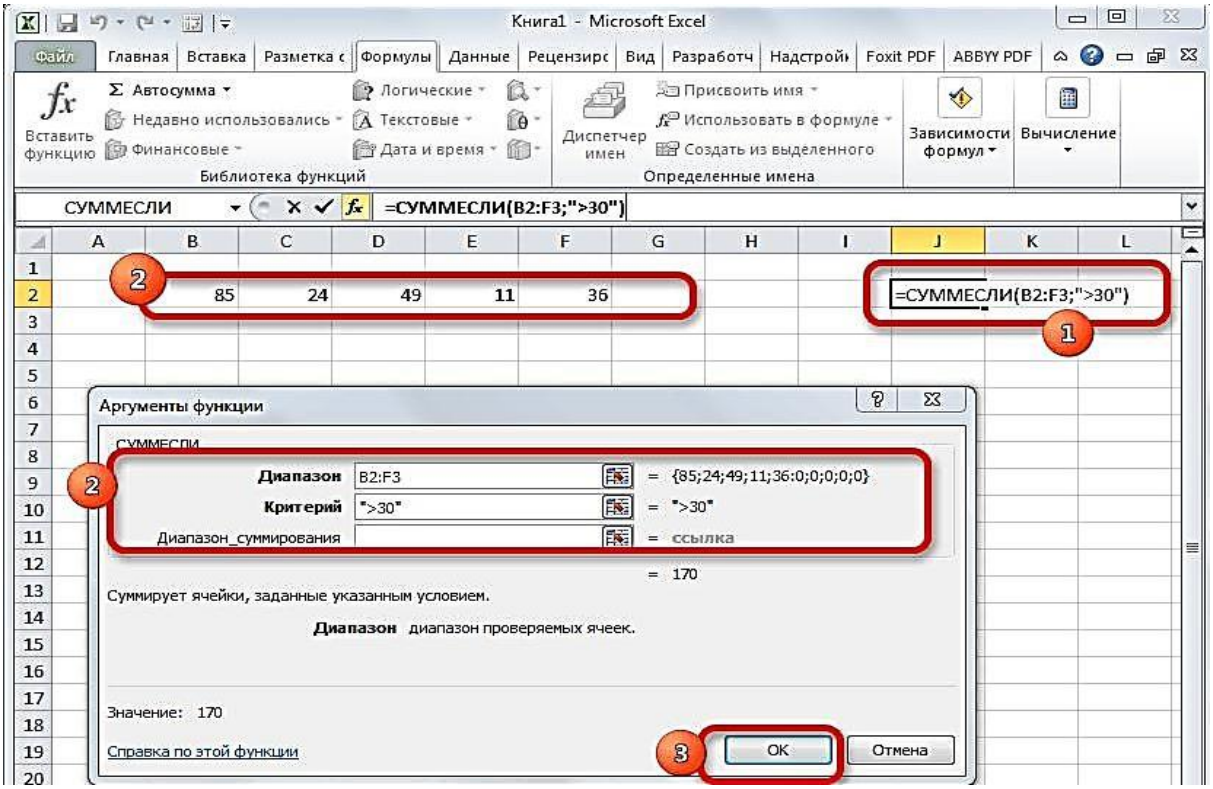

Рис. 25. Пример использования функции СУММЕСЛИ ()

ОКРУГЛ () служит для округления чисел до заданного количества разрядов. Первым аргументом данного оператора является число или ссылка на ячейку, в которой содержится числовой элемент. В отличие от большинства других функций, диапазон значением выступать не может. Вторым аргументом является количество десятичных знаков, до которых нужно произвести округление. Округления проводится по общематематическим правилам, то есть, к ближайшему по модулю числу. Синтаксис формулы: =ОКРУГЛ(число; число_разрядов). Пример использования этой функции представлен на рис.26.

Кроме того, в Excel существуют такие функции, как *ОКРУГЛВВЕРХ* () и *ОКРУГЛВНИЗ ()*, которые соответственно округляют числа до ближайшего большего и меньшего по модулю числа.

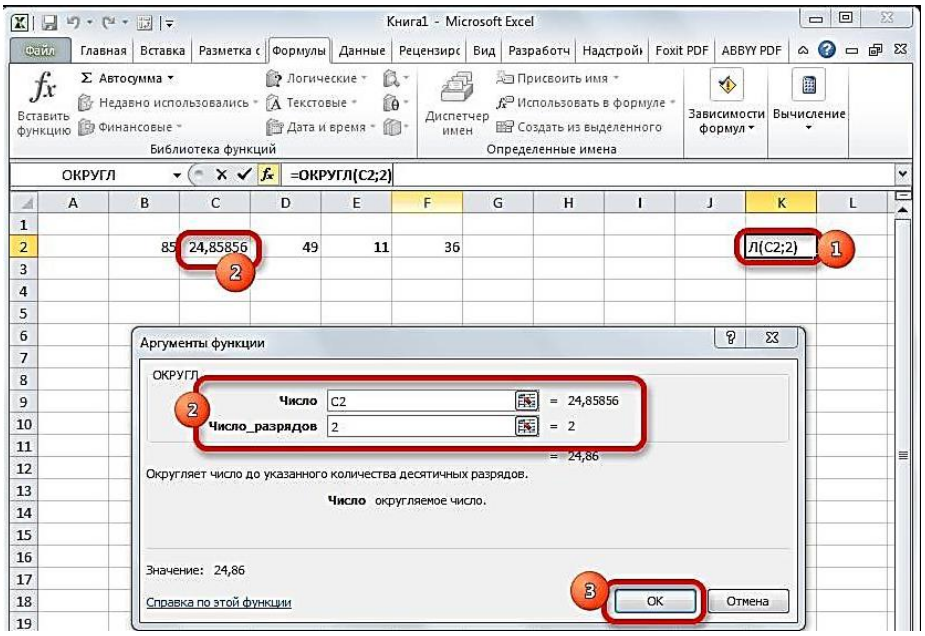

Рис. 26. Пример использования функции ОКРУГЛ ()

ПРОИЗВЕД (). Аргументами этой функции являются ссылки на ячейки, в которых содержатся данные для операции умножения. Всего может быть использовано до 255 таких ссылок. Результат умножения выводится в отдельную ячейку (рис.27). Синтаксис оператора: =ПРОИЗВЕД (число; число;…число)

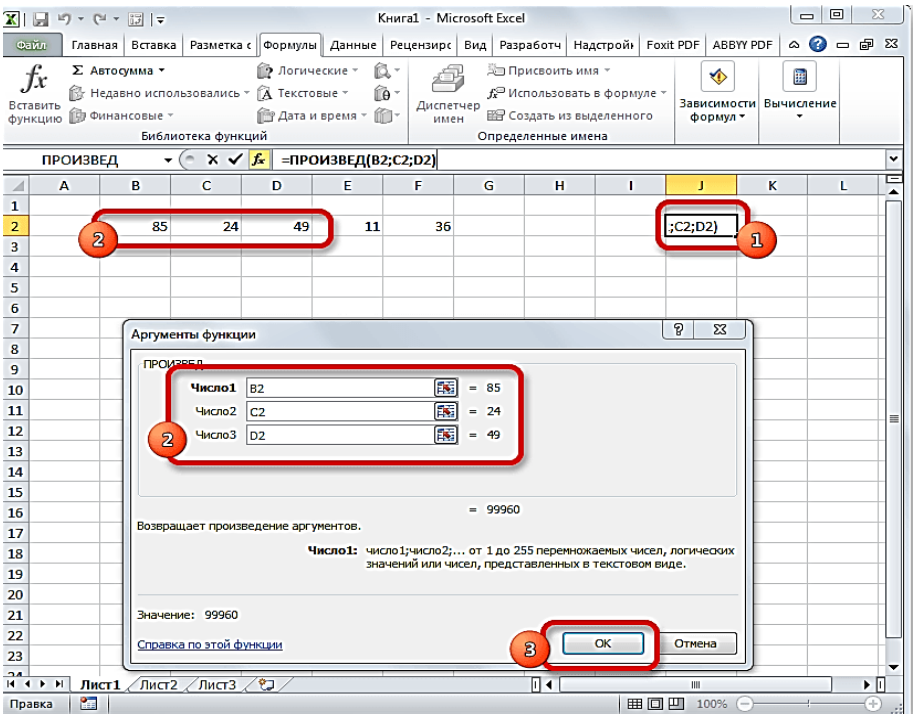

Рис. 27. Пример использования функции ПРОИЗВЕД ()

ABS () – производит расчет модуля числа (рис.28). Аргументом может быть ч*исло* или ссылка на ячейку, содержащую числовые данные. Диапазон в роли аргумента выступать не может. Синтаксис: =ABS (число).

СТЕПЕНЬ () используется для возведения числа в заданную степень. У функции два аргумента: *«Число»* и *«Степень»* (рис.29). Первый из них может быть указан в виде ссылки на ячейку, содержащую числовую величину. Второй аргумент указывается степень возведения. Синтаксис оператора: =СТЕПЕНЬ (число; степень)

КОРЕНЬ () служит для извлечения квадратного корня из аргумента (рис.30). В его роли может выступать число или ссылка на ячейку, содержащую данные. Синтаксис: =КОРЕНЬ (число)

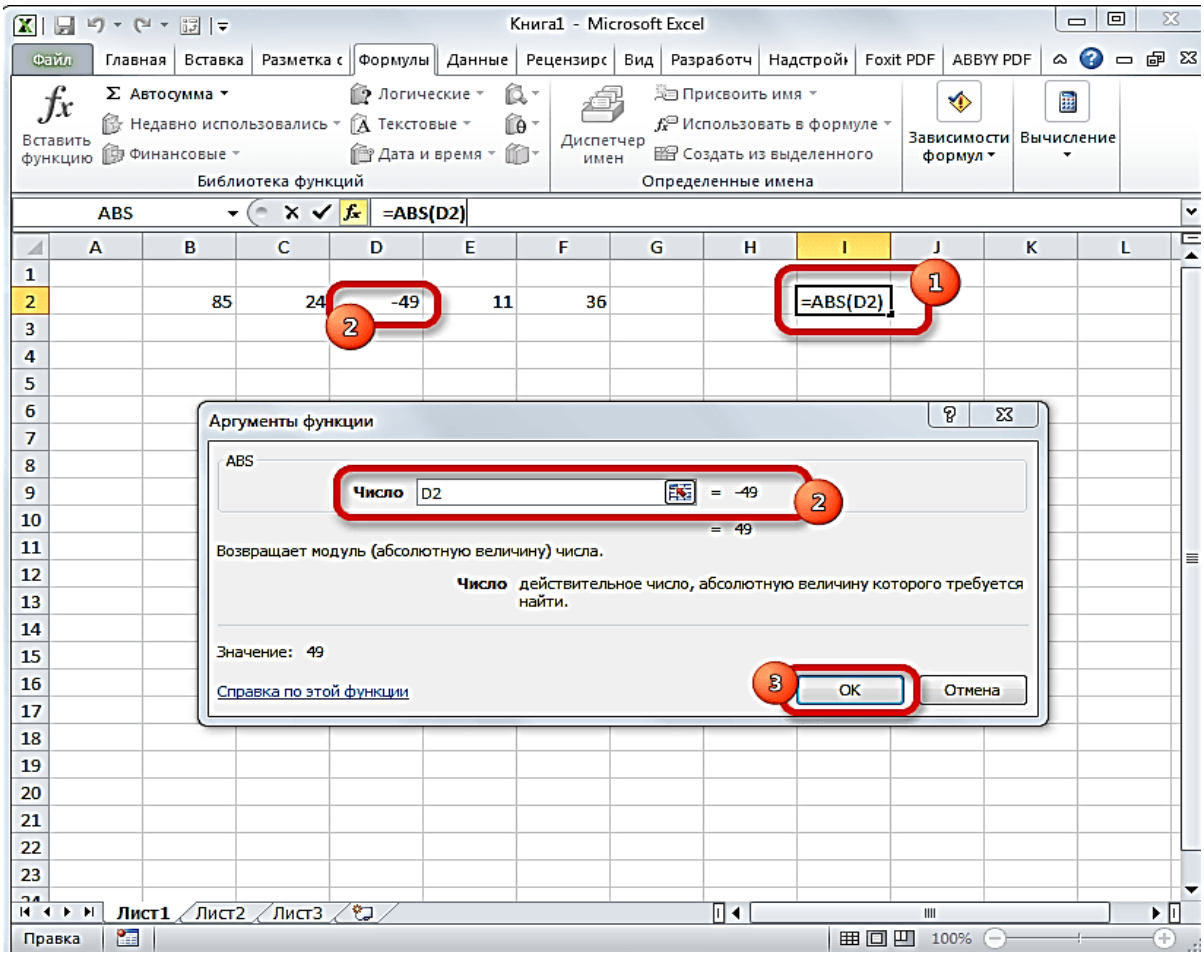

Рис. 28. Пример использования функции ABS ()

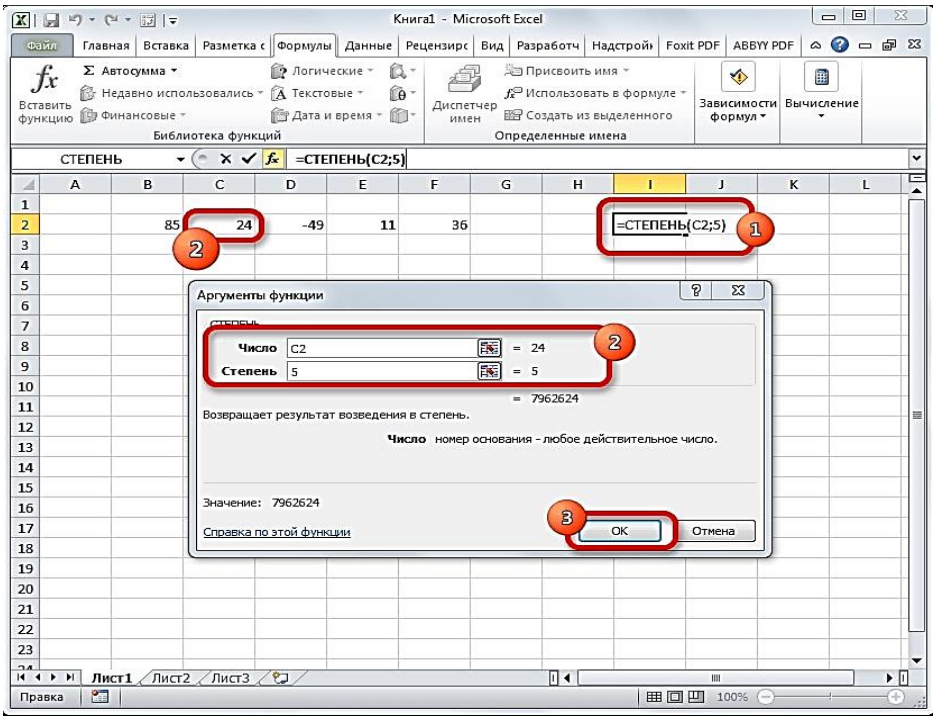

Рис. 29. Пример использования функции СТЕПЕНЬ ()

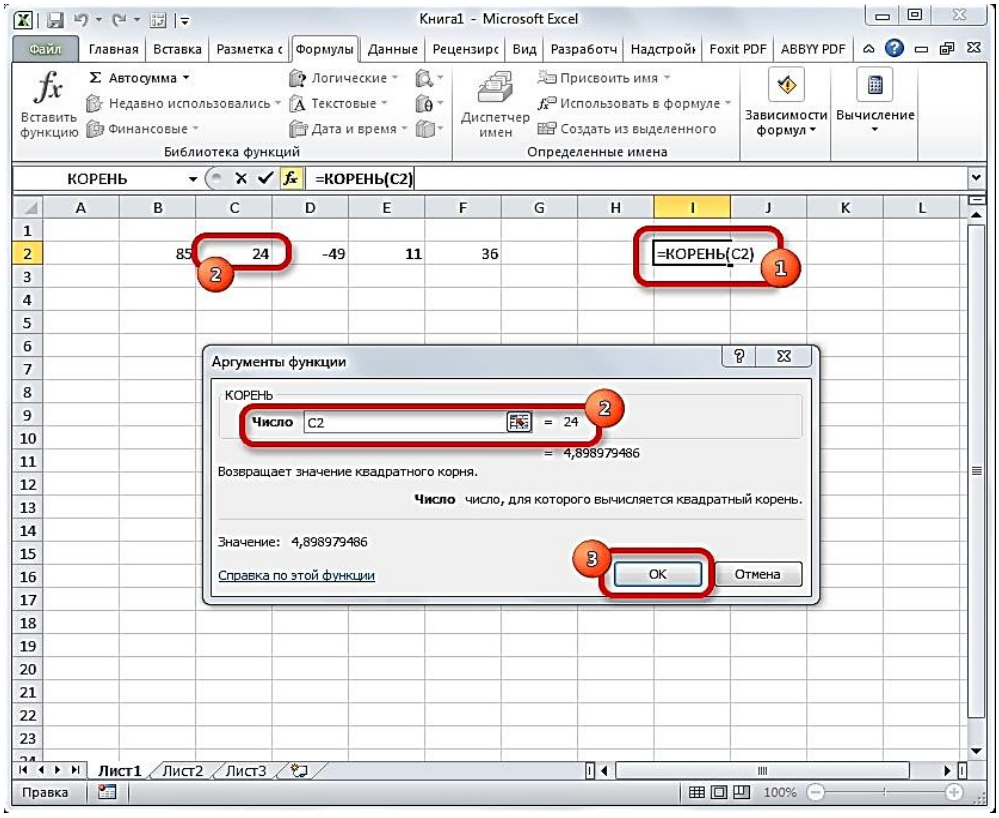

Рис. 30. Пример использования функции КОРЕНЬ ()

СЛУЧМЕЖДУ () выводит в указанную ячейку любое случайное число, находящееся между двумя заданными числами (рис. 31). Её аргументами является верхняя и нижняя границы интервала. Синтаксис: =СЛУЧМЕЖДУ (Нижн_граница; Верхн_граница).

РИМСКОЕ (). Данная функция позволяет преобразовывать арабские числа, которыми по умолчанию оперирует Excel, в римские (рис.32). У этого оператора два аргумента: ссылка на ячейку с преобразуемым числом и форма. Второй аргумент не является обязательным. Синтаксис: =РИМСКОЕ (Число; Форма).

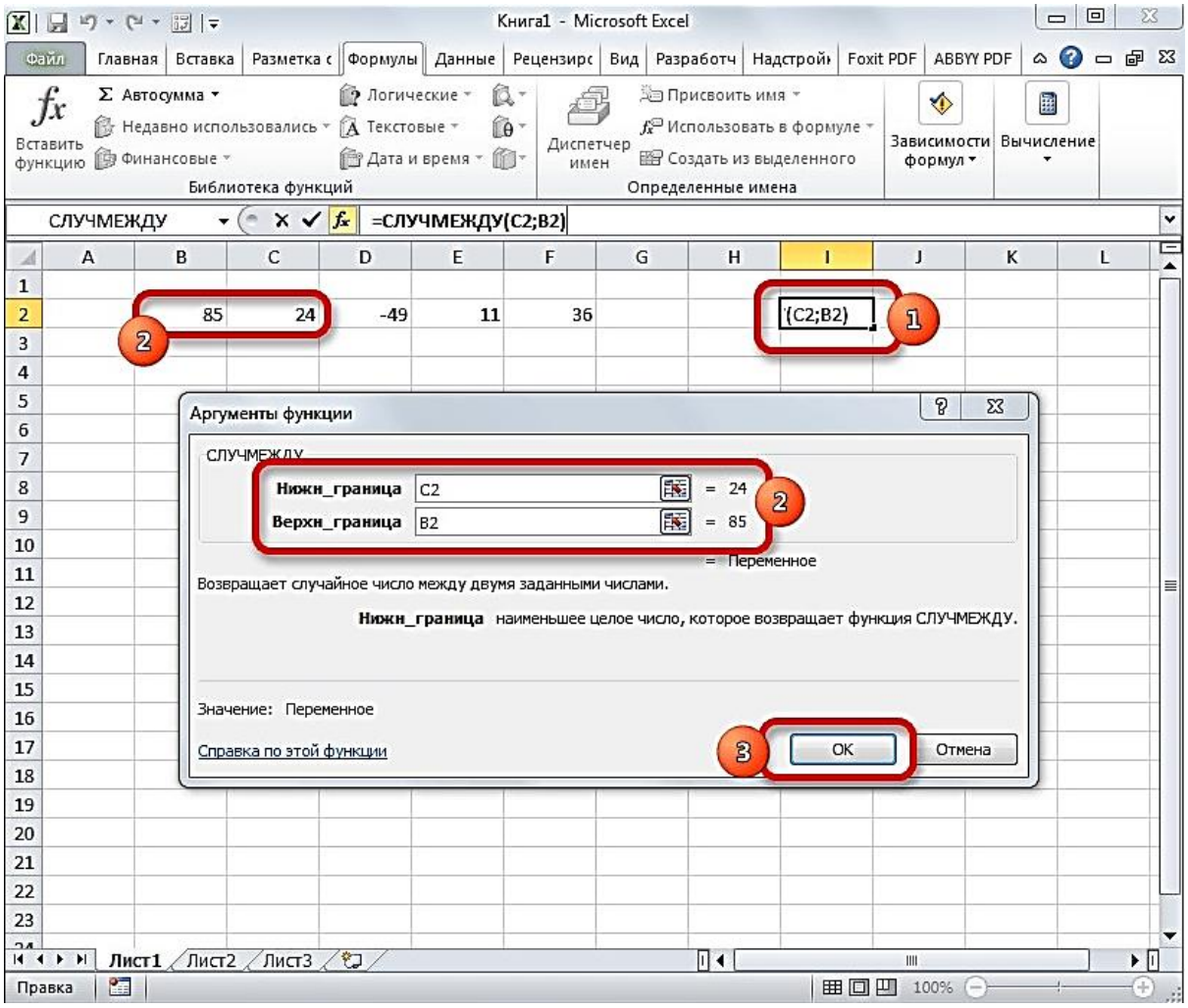

Рис. 31. Пример использования функции СЛУЧМЕЖДУ ()

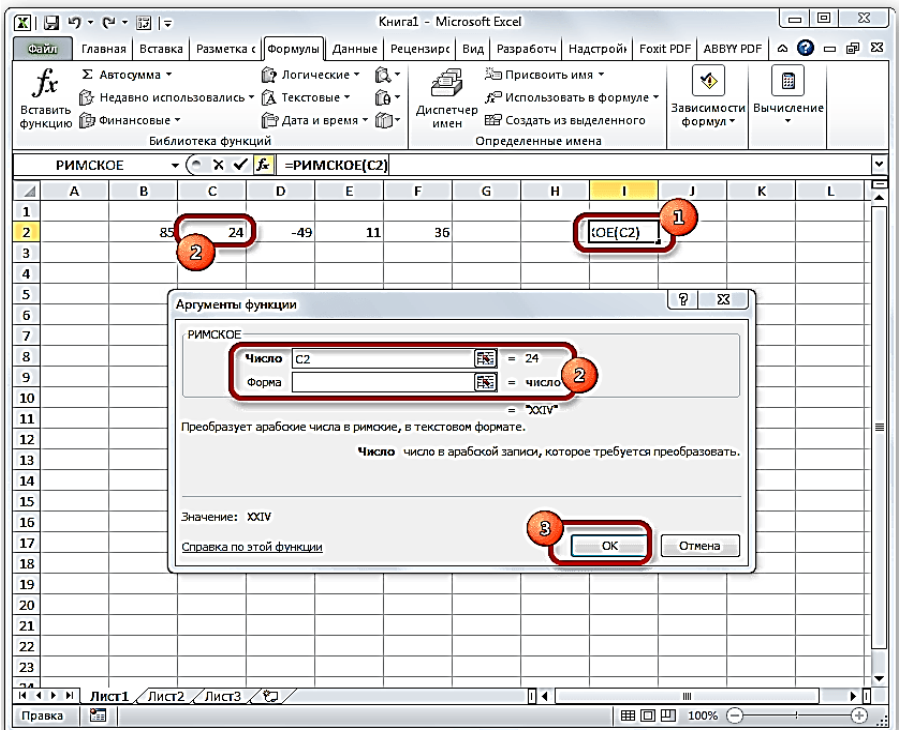

Рис. 32. Пример использования функции РИМСКОЕ ()

1.2.2. Логические функции MS Excel

Логические функции в Excel проверяют данные и возвращают результат «ИСТИНА», если условие выполняется, и «ЛОЖЬ», если нет. Значения логических переменных можно вводить с клавиатуры вручную или использовать Мастер функций (кнопка f_x на панели инструментов), последовательный выбор: f_x , *Категория, Логические, Функция, Истина (или Ложь)* (см. рис. 33).

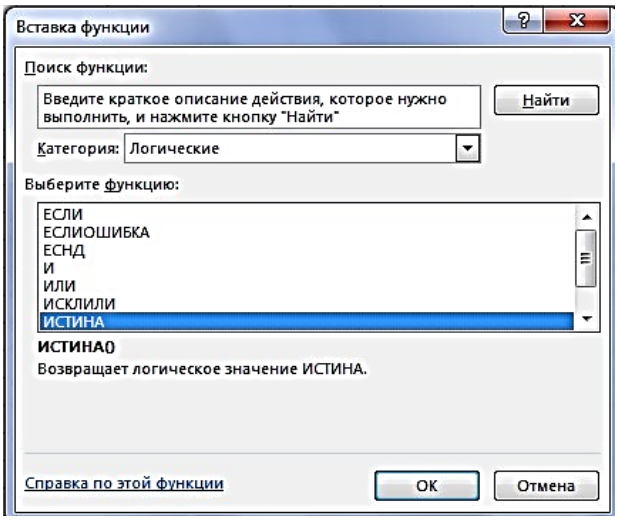

Рис. 33. Перечень логических функций

Рассмотрим синтаксис логических функций и примеры применения их в процессе работы с программой Excel.

1.2.3. Встроенные функции MS Excel

В Excel насчитывается более 400 встроенных функций. С полным перечнем функций можно ознакомиться, нажав кнопку f_x или выбрав вкладку Формулы (см. рис.34)

Рис. 34. Библиотека функций Excel

Подробно рассмотрим характеристики некоторых из них, а также их применение для решения инженерных и экономических задач.

Таблица 2. Использование логических функций в EXCEL

Название функции	Значение	Синтаксис	Примечание
ИСТИНА	Не имеет аргументов, возвращает логическое значение «ИСТИНА».	=ИСТИНА ()	
ЛОЖЬ	Не имеет аргументов, возвращает логическое выражение «ЛОЖЬ».	=ЛОЖЬ ()	
И	Если все заданные аргументы возвращают истинный результат, то функция выдает логическое выражение «ИСТИНА». В случае хотя бы одного ложного логического значения вся функция выдает результат «ЛОЖЬ».	=И (Лог_знач. 1; Лог_знач. 2;)	Принимает до 255 аргументов в виде условий или ссылок. Обязательным является первый аргумент.
ИЛИ	Показывает результат «ИСТИНА», если хотя бы один из аргументов является истинным.	=ИЛИ (Лог_знач.1; Лог_знач. 2;)	Принимает до 255 аргументов в виде условий или ссылок. Обязательным является первый аргумент.

Название функции	Значение	Синтаксис	Примечание
НЕ	Меняет логическое значение «ИСТИНА» на противоположное – «ЛОЖЬ» и наоборот.	НЕ ()	Обычно сочетается с другими операторами.
ЕСЛИ	Проверяет истинность логического выражения и возвращает соответствующий результат.	ЕСЛИ (логическое выражение; значение если истинна, значение если ложь)	Значение если ложь не обязательно
ЕСЛИОШИБКА	Если значение первого аргумента истинно, то функция возвращает сам аргумент. В противном случае – значение второго аргумента.	ЕСЛИОШИБКА (значение; значение если ошибка)	Оба аргумента обязательны.

Практическая работа № 4 «Составление таблиц истинности с помощью табличного процессора MS Excel»

Задание 1. С помощью MS Excel составьте таблицы истинности основных логических функций: инверсии, конъюнкции, дизъюнкции, импликации и эквивалентности согласно образцу на рис.35

	A	B	C	D	E	F	G
1	A	B	не A	A ∨ B	A ∧ B	A ⇒ B	A ⇔ B
2	1	1	0	1	1	1	1
3	1	0	0	1	0	0	0
4	0	1	1	1	0	1	0
5	0	0	1	0	0	1	1
6							

Рис. 35. Таблица истинности основных логических операций

Задание 2. Используя таблицу основных логических функций из задания 1, постройте таблицу истинности для формулы $(A \wedge \overline{B} \to C) \leftrightarrow A$ согласно образцу, представленному на рисунке 36.

	A	B	C	D	E	F	G
1	A	B	C	не B	A ∧ (не B)	A ∧ (не B) ⇒ C	(A ∧ (не B) ⇒ C) ⇔ A
2	0	0	0	1	0	1	0
3	0	0	1	1	0	1	0
4	0	1	0	0	0	1	0
5	0	1	1	0	0	1	0
6	1	0	0	1	1	0	0
7	1	0	1	1	1	1	1
8	1	1	0	0	0	1	1
9	1	1	1	0	0	1	1

Рис. 36. Результат выполнения задания 2

Задание 3. Определите с помощью таблиц истинности равносильность следующих формул:

$$(\overline{A} \to B) \wedge (A \to \overline{B}) \equiv (\overline{B} \to A) \wedge (B \to \overline{A})$$
$$\overline{A} \wedge B \vee \overline{C} \wedge D \equiv B \wedge \overline{A \wedge C}$$
$$(A \vee B) \wedge (\overline{A} \vee \overline{B}) \equiv (A \wedge B) \vee \overline{(A \vee B)}$$
$$\overline{(A \leftrightarrow B)} \equiv (A \wedge \overline{B}) \vee (\overline{A} \wedge B)$$
$$A \leftrightarrow B \equiv (\overline{A} \vee B) \wedge (A \vee \overline{B})$$
$$A \leftrightarrow B \equiv (\overline{A} \wedge \overline{B}) \vee (A \vee B)$$

1.2.4. Статистические функции MS Excel

Статистические функции используются для автоматизации статистической обработки данных.

Рассмотрим часто используемые статистические функции.

=МИН (число1; число2; …; число n)

Функция возвращает наименьшее значение из списка аргументов. Логические и текстовые значение игнорируются.

Например:

Ячейки A1:A5 содержать значения 10,7,9,27 и 2

Чему будет равно значение функции?

=МИН(A1:A5) $\Rightarrow$ *2*

=МАКС (число1; число2;…; число $_n$)

Функция возвращает наибольшие значение из списка аргументов. Логические и текстовые значение игнорируются.

Например:

Ячейки A1:A5 содержать значения 10,7,9,27 и 2

Чему будет равно значение функции?

=МАКС(A1:A5) $\Rightarrow$ *27*

=СРЗНАЧ (число1; число2;…; число $_n$)

Функция возвращает среднее арифметическое своих аргументов, которые могут быть числами, именами, массивами или ссылками на

ячейки с числами. Если аргумент текстовый, логический или пустая ячейка, то значения игнорируются.

Например:

Ячейки A1:A5 содержат значения 10,7,9,27 и 2

Чему будет равно значение функции?

=СРЗНАЧ(A1:A5) ⇒ *11*

Если требуется не только вычислить наибольшее или наименьшее число из списка значений, но и расположить числа в порядке возрастания или убывания применяется функция ранжирования, которая записывается следующим образом:

=РАНГ(число; ссылка на список; порядок)

Где:

Число – это число, для которого определяется ранг (порядок);

Ссылка на список – которому принадлежит число, нечисловые значения в ссылке игнорируются (ссылка на список должна быть абсолютной);

Порядок – способ упорядочения значений списка:

- 0 или ничего – определяет ранг числа, так как, если бы список сортировался в порядке убывания (т.е. максимальному значению присваивается ранг равный 1, чуть меньшему числу ранг 2 и т.д.);
- Число не равное 0 – определяет ранг числа, так как если бы список сортировался в порядке возрастания (т.е. минимальному числу присваивается ранг равный 1, чуть большему числу ранг 2 и т.д.).

Например:
Ячейки A1:A5 содержат числа 7 3,5 4 1 2
Чему будет равно значение функции для ячейки A2: =РАНГ(A1;$A$1:$A$5;1)
Обратите внимание, что порядок (последняя цифра) в этой функции равен 1, число не равное 0. Значит, иными словами, мы хотим узнать, какой порядок занимает число A2 равное 3,5 в списке чисел, отсортированном в порядке возрастания. Функция присвоит наименьшему числу ранг 1.

Практическая работа № 5 «Статистические функции MS Excel»

Задание 1. Используя статистические функции Excel, по заданному списку учащихся группы определить, кто:

а) является отличником;

б) самым старшим в группе;

в) самым младшим в группе;

г) средний балл девочек (мальчиков);

д) доля отличниц среди девочек (мальчиков);

е) разницу среднего балла учащихся разных возрастов.

Задание 2. Используя функцию РАНГ (), расставьте учащихся согласно их возрасту.

Список учащихся гр. Тсп113

№ п.п.	Фамилия	Имя	Ср. балл	Дата рождения	Пол	Возраст
1	Иванов	Сергей	3	12.01.1993	м	17
2	Петрова	Елена	3,7	14.05.1992	ж	18
3	Сидорова	Елизавета	4,4	30.03.1993	ж	16
4	Семенов	Роман	4,2	04.01.1993	м	16
5	Аникина	Инга	3,9	20.11.1992	ж	16
6	Сидоренко	Петр	4	06.06.1992	м	20
7	Прокошева	Оксана	4,9	22.05.1993	ж	19
8	Ошуркова	Ирина	4,3	21.04.1993	ж	18
9	Золотых	Игорь	5	05.07.1992	м	18
10	Дорошенко	Денис	5	04.08.1992	м	18
11	Светлаков	Михаил	3,1	01.03.1993	м	18
12	Серова	Наталья	5	15.02.1993	ж	18

1.2.5. Условное форматирование в MS Excel

Условное форматирование позволяет автоматически изменять форматирование ячеек в зависимости от их содержимого. Для настройки условного формата следует воспользоваться соответствующей командой на вкладке *Главная* (рис. 37).

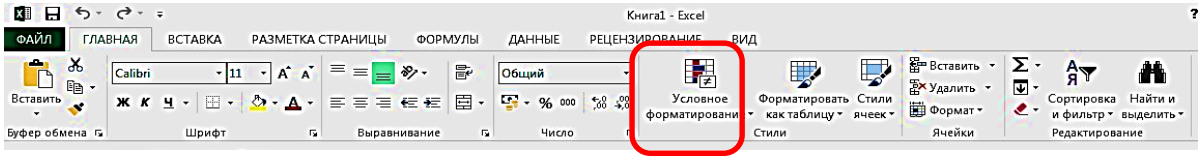

Рис. 37. Вкладка Условное форматирование

При её нажатии открывается меню (рис.38). Первые 5 команд – это готовые сценарии для быстрого условного форматирования. Чтобы ими воспользоваться достаточно выбрать нужный вариант и сделать минимальные настройки.

Оставшиеся три команды предназначены для ручного создания, удаления и управления правилами условного форматирования.

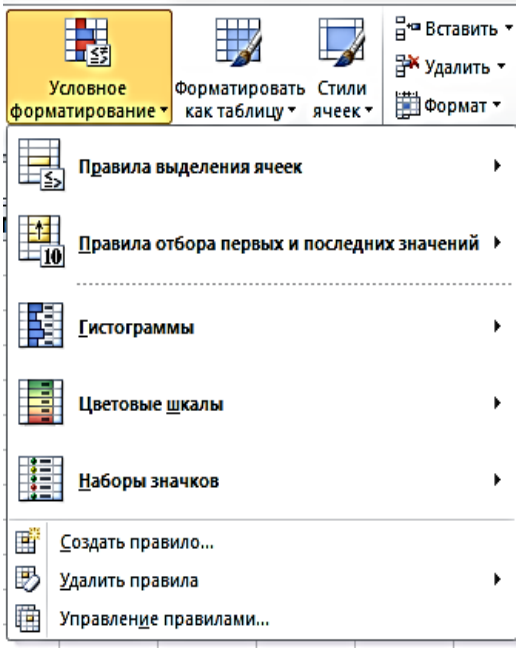

Рис. 38. Меню вкладки Условное форматирование

Рассмотрим сценарии быстрого условного форматирования.

Правила выделения ячеек применяют для ячеек, которые сравниваются с определенным значением. Возможны различные варианты, которые показаны на рисунке 39.

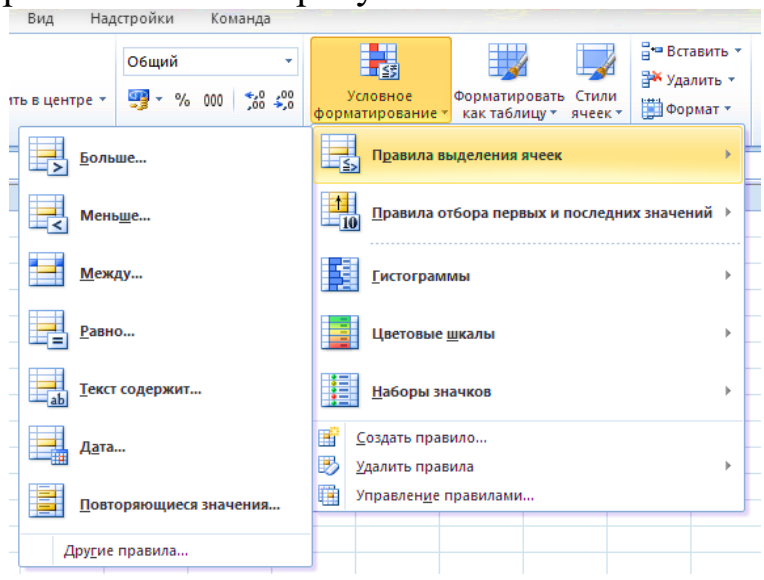

Рис. 39. Меню вариантов сравнения

- *Больше…* Если значение ячейки, к которой применяется правило выделения, больше указанного значения, то в силу вступает заданный формат.
- *Меньше…* Форматируются ячейки, у которых значение меньше заданного порога.
- *Между…* Форматирование наступает, если содержимое ячейки находится внутри заданных границ.
- *Равно…* если значение или текст в ячейке совпадает с условием.
- *Текст содержит…* Если совпадает только часть текста (слово, код, комбинация символов и т.д.). Результат действия представлен на рис. 40.
- *Дата…* Возможность форматировать периоды, отстоящие от текущей даты (рис.41). Например, сегодня, вчера, последние 7 дней, следующий месяц и др. Условное форматирование даты полезно при контроле платежей, отгрузок и т.п.
- *Повторяющиеся значения…* выделяются ячейки с одинаковым содержимым (рис.42). В настройках можно выбрать и обратный вариант – выделить только уникальные значения.

Текст содержит "ра"		
вап	апр	шщз
енг	вап	дж
фыва	акмра	зхъ
чсм	олб	ывп.дл
вап	рпт	ол
рак	вало	ьрь
про	ываолр	ывара.пл
гшщ	мваж	еккепи

Рис. 40. Результат действия правила Текст содержит…

Сегодня 21.04.2016		
21.04.2016	29.04.2016	07.05.2016
22.04.2016	30.04.2016	08.05.2016
23.04.2016	01.05.2016	09.05.2016
24.04.2016	02.05.2016	10.05.2016
25.04.2016	03.05.2016	11.05.2016
26.04.2016	04.05.2016	12.05.2016
27.04.2016	05.05.2016	13.05.2016
28.04.2016	06.05.2016	14.05.2016

Рис. 41. Результат действия правила Дата…

Дубликаты		
75	69	9
14	35	92
36	14	7
86	33	40
56	50	55
14	77	72
70	16	81
13	22	32

Рис. 42. Результат действия правила Повторяющиеся значения

Правила отбора первых и последних значений выделяют наибольшие или наименьшие значения. Помогают анализировать данные, показывая приоритеты и «слабые места» (рис.43 а,б).

Первые 10 элементов… Выделяются первые топ–10 ячеек. Количество регулируется в диалоговом окне (можно сделать топ-5, топ-20 и др.) (рис.43а,б).

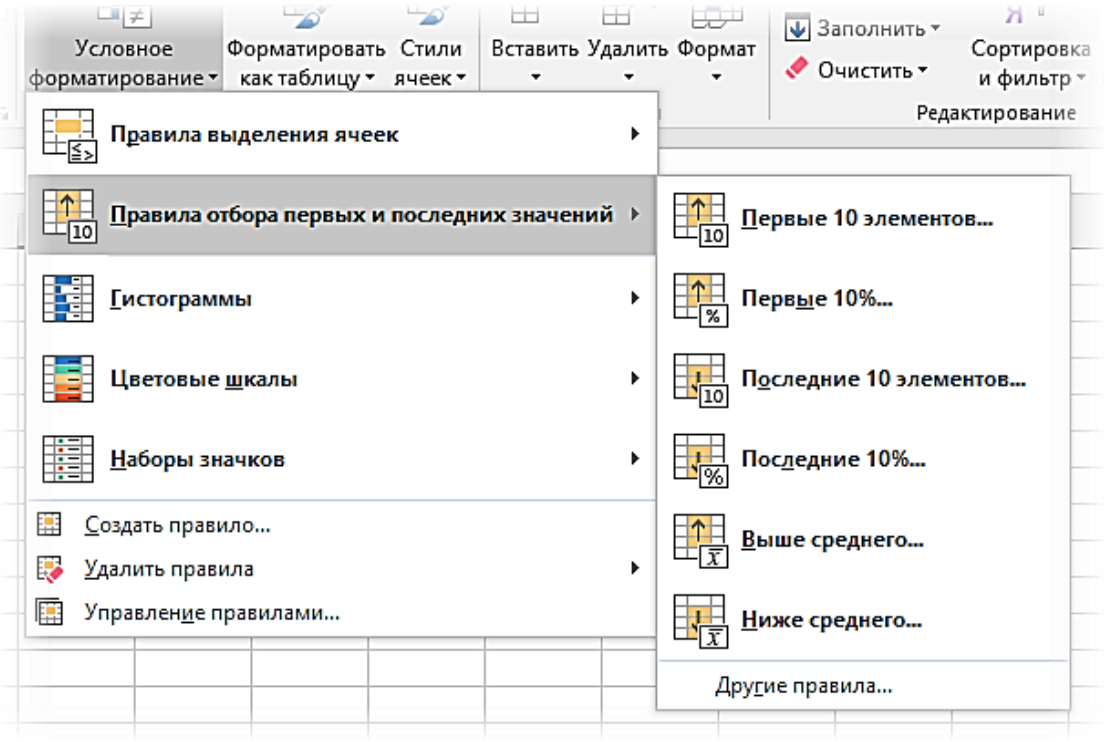

Рис. 43а. Варианты сценария Правила отбора первых и последних значений

Первые 5 элементов		
96	18	77
83	64	99
82	35	93
44	37	93
43	90	60
87	75	74
18	48	70
95	20	38

Рис. 43б. Результат применения Правила отбора первых и последних значений

Первые 10%... Выделяются 10% наибольших значений. Долю также можно изменить.

Последние 10 элементов… Аналогично с первым пунктом, только форматируются 10 (или другое количество) наименьших значений (рис.43в).

30			
31	Последние 5 элементов		
32	21	64	72
33	63	34	79
34	78	50	67
35	77	72	81
36	73	11	44
37	18	62	77
38	48	54	21
39	0	72	63
40			

Рис. 43в. Результат применения Правила Последние …элементов

Последние 10%... Наименьшие 10% или другая доля от всех элементов (рис.43г).

	Последние 20% элементов		
	8	26	75
	50	39	5
	88	42	91
	81	46	17
	86	34	40
	30	98	2
	10	28	99
	61	30	77

Рис.43г. Результат применения правила Последние …%

Выше среднего… Форматируются все значения, большие среднего арифметического (ри.43д).

Выше среднего		
16	81	83
29	19	8
31	15	71
58	18	91
30	2	95
19	11	29
4	48	31
90	9	31

Рис.43д. Результат применения правила Выше среднего…

Ниже среднего… Форматируются все значения, меньшие среднего арифметического (см. рис.43е).

Ниже среднего		
86	12	39
36	56	72
86	59	23
27	43	41
93	48	82
61	66	1
53	60	20
25	59	49

Рис. 43е. Результат применения правила Ниже среднего…

Гистограммы (рис.44) позволяют в каждую ячейку с числом добавить столбец линейной гистограммы, размер которой определяется относительно максимального значения в выделенном диапазоне. Помогает визуализировать небольшой набор данных без использования отдельных диаграмм (рис.45).

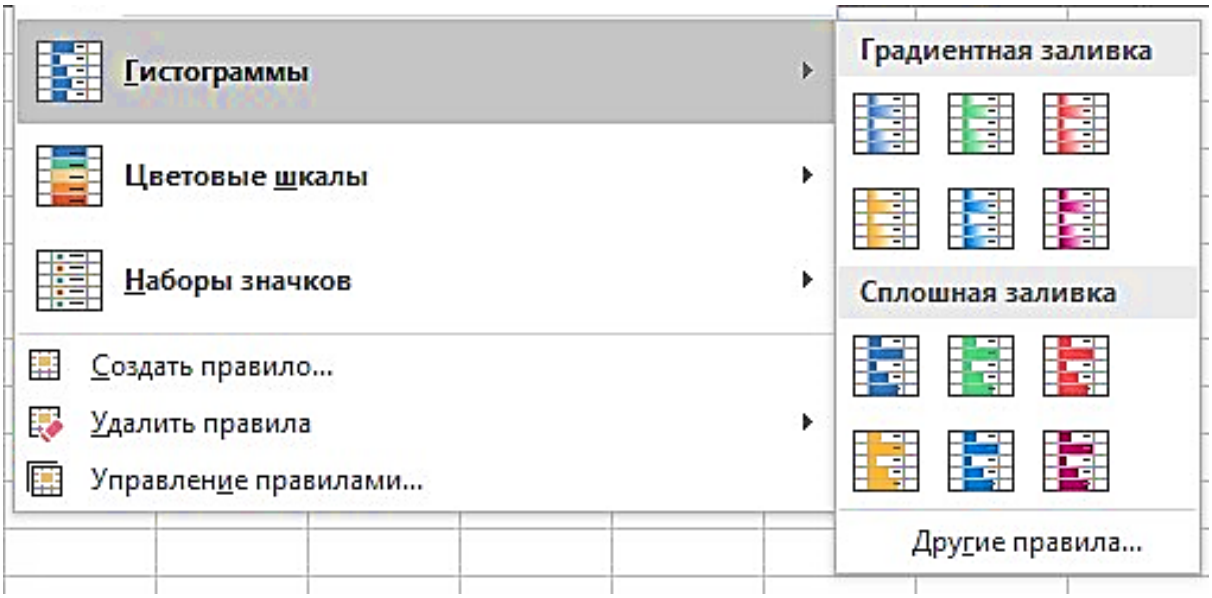

Рис. 44. Вкладка Гистограммы

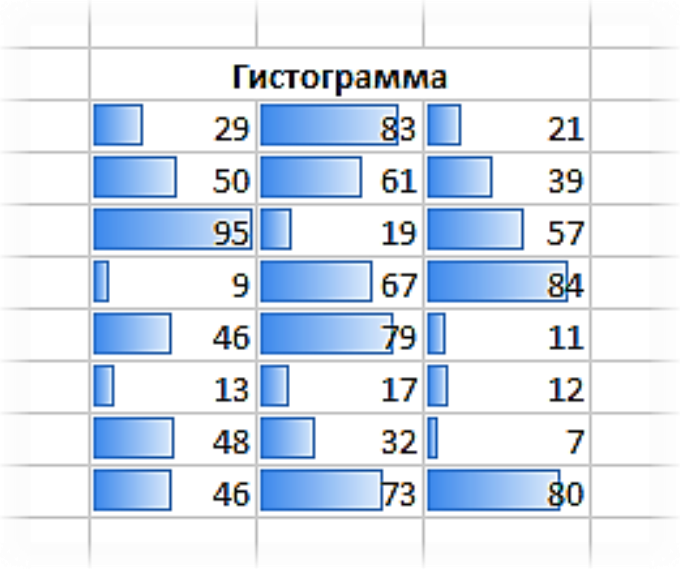

Рис. 45. Результат применения инструмента Гистограмма

Цветные шкалы (рис.46) автоматически определяют максимальное и минимальное значение в диапазоне и форматирует каждую ячейку по цвету, который соответствует значению, изображая что-то вроде тепловой карты. Например, наибольшее значения – это красное, наименьшее – зеленое, а остальные ячейки – это плавный переход от одного цвета к другому через промежуточный белый (рис.47).

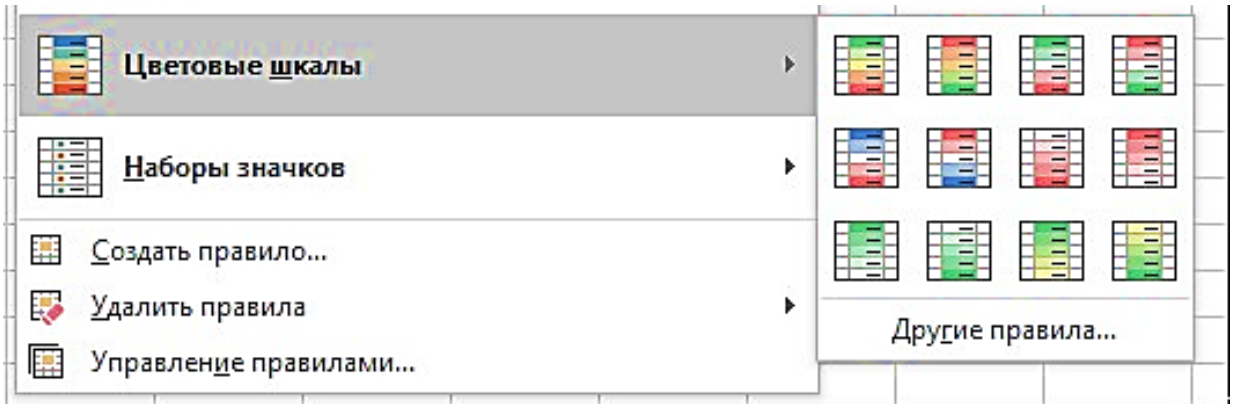

Рис. 46. Вкладка Цветовые шкалы

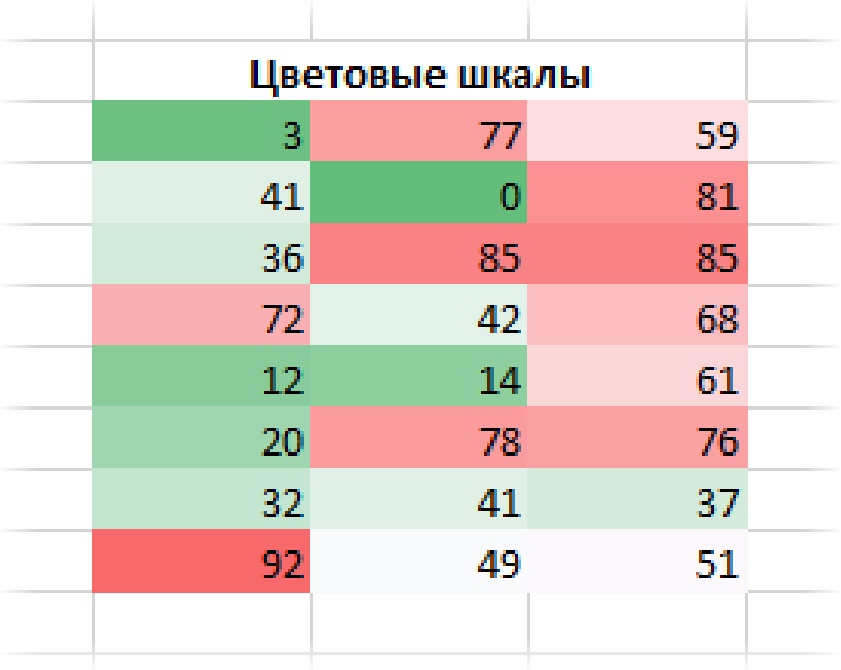

Рис. 47. Результат применения цветовой шкалы

Набор значков. В этом случае каждой ячейке присваивается свой значок в соответствии с выбранным стилем (рис.48). Внешний вид таблицы представлен на рис.49

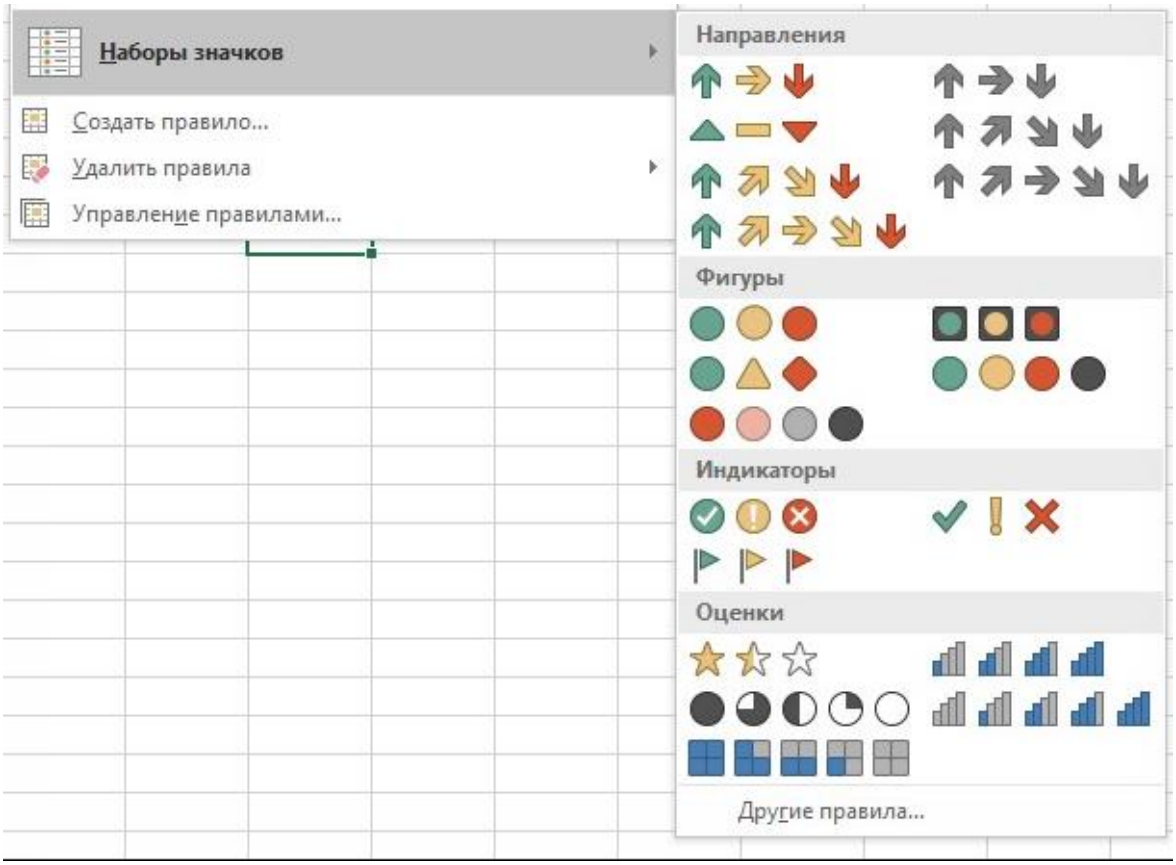

Рис. 48. Внешний вид вкладки Набор значков

	Значки				
2	✓	79	!	51	✗ 10
3	✗	9	✗	26	✓ 82
4	!	52	!	53	✗ 26
5	!	30	!	49	✗ 24
6	✗	23	✓	80	! 54
7	!	43	✗	6	! 43
8	!	46	✗	3	✗ 3
9	✓	66	!	30	✗ 24

Рис. 49. Результат применения инструмента Набор значков

Все представленные на рисунках таблицы были сделаны с помощью стилей по умолчанию. Чтобы внести изменения, нужно выделить диапазон и перейти на вкладку Управление правилами (рис.50). Откроется диалоговое окно, где можно создать новое, изменить или удалить правило. Также можно задать несколько правил (рис.51).

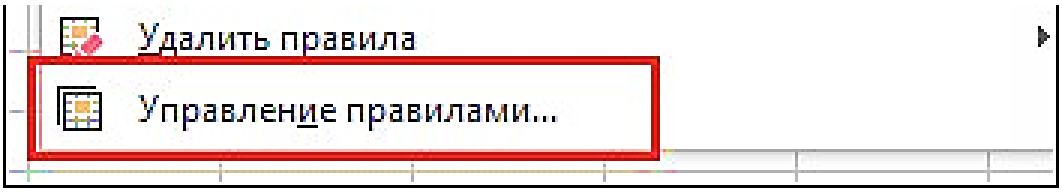

Рис. 50. Вкладка Управление правилами

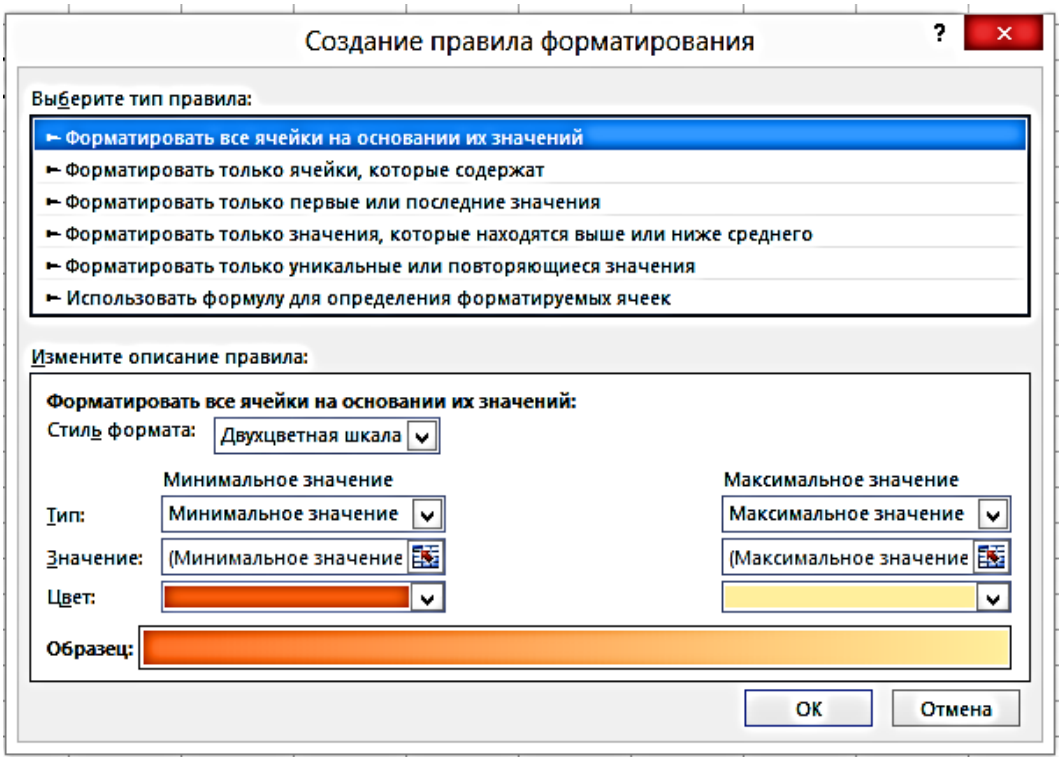

Рис. 51. Окно Диспетчера правил

После нажатия кнопки «Изменить правило…» откроется окно, вид которого зависит от редактируемого правила.

Для удаления правила условного форматирования необходимо выбрать команду Удалить правило (рис. 52).

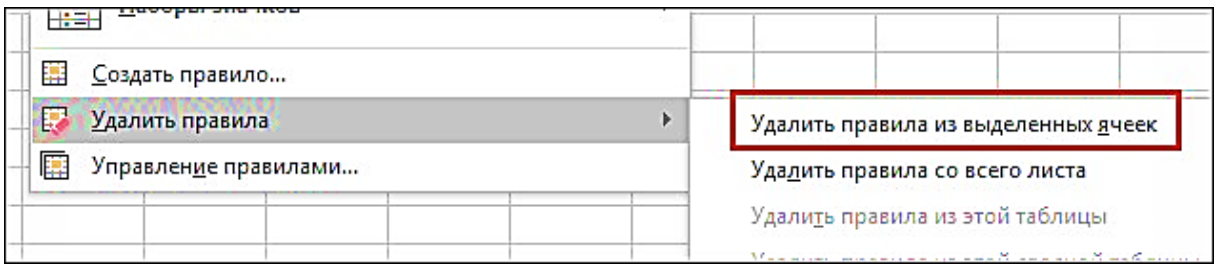

Рис. 52. Окно удаления правила форматирования

ВАЖНО! *При использовании любого форматирования всегда нужно помнить о цели его применения: облегчить восприятие информации и привлечь внимание к наиболее важным моментам. Фор-*

мат, представленный на рис. 53 – это результат неправильного применения условного форматирования.

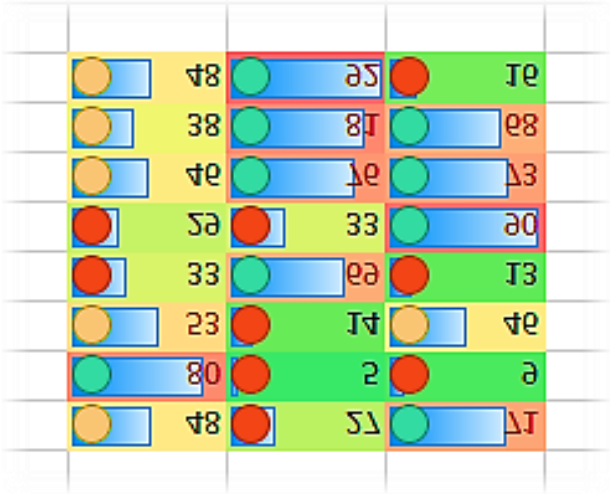

Рис. 53. Результат неправильного применения условного форматирования

Практическая работа № 6
«Использование условного форматирования в MS Excel»

Задание. Пусть нам известны сроки (начало и конец) работы сотрудников на некотором участке производства в апреле месяце. Требуется выделить цветом даты работы каждого сотрудника.

Рабочий график										
Ф.И.О.	Дата начала	Дата окончания	01.апр	02.апр	03.апр	04.апр	05.апр	06.апр	07.апр	Всего Отработано
Петров И.П.										
Иванов Г.А.										
Сидоров И.И.										

1.2.6. Календарные функции MS Excel

Календарные функции MS Excel позволяют работать с датами. Рассмотрим некоторые из них: СЕГОДНЯ (), ГОД (), ДОЛЯГОДА (), МЕСЯЦ ().

СЕГОДНЯ (). Возвращает текущую дату в числовом формате. Числовой формат даты – это код даты и времени, с помощью которого в Microsoft Excel производятся вычисления над датами и промежутками времени. Если до ввода этой функции для ячейки был задан формат Общий, результат будет отформатирован как Дата. Если должно отображаться число, выберите для ячейки Общий или Числовой формат.

Функция СЕГОДНЯ () полезна, если на листе требуется отображать текущую дату независимо от времени открытия книги. Она также используется для вычисления интервалов. Например, если известно, что кто-либо родился в 1963 году, узнать возраст этого человека можно с помощью следующей функции: = ГОД(СЕГОДНЯ())-1963

В этой формуле функция СЕГОДНЯ используется в качестве аргумента функции ГОД для получения текущего года, из которого вычитается 1963. Полученное значение и есть возраст человека.

ГОД (). Возвращает год, соответствующий заданной дате. Год определяется как целое число в диапазоне от 1900 до 9999.

Синтаксис: ГОД (дата_в_числовом_формате). Дата должны вводиться с помощью функции ДАТА или как результат вычисления других формул и функций. Например, для указания даты 23 мая 2008 г. используйте выражение ДАТА(2008,5,23). Если даты вводятся как текст, это может привести к возникновению проблем.

МЕСЯЦ (). Возвращает месяц для даты, заданной в числовом формате. Месяц возвращается как целое число в диапазоне от 1 (январь) до 12 (декабрь). Синтаксис: МЕСЯЦ(дата_в_числовом_формате). Дата должна быть введена с использованием функции ДАТА либо как результат других формул или функций. Например, чтобы задать дату 23 мая 2008 г., используйте выражение ДАТА(2008;5;23). Если ввести дату как текст, могут возникнуть проблемы.

Практическая работа № 7
«Календарные функции MS Ecxel»

Задание. Дан список сотрудников фирмы, содержащий паспортные данные: фамилия, имя, отчество, дата рождения, дата зачисления в состав фирмы (таблица 1). По этому списку необходимо составить другой список (таблица 2), содержащий следующие данные: фамилия и инициалы, возраст (количество полных лет), рабочий стаж в фирме (в годах).

1) Составьте таблицу сотрудников фирмы, содержащую следующие данные:

Таблица 1. Список сотрудников фирмы					
№ п/п	Фамилия	Имя	Отчество	Дата рождения	Дата зачисления
1.	Макаров	Сергей	Петрович	23.05.40	05.09.90
...	...	...	...	...	...

2. На основе данных таблицы 1. постройте таблицу 2. Рассчитайте возраст и стаж работы сотрудников фирмы.

Таблица 2. Список сотрудников фирмы			
№ п/п	Фамилия И.О.	Возраст	Стаж
1.	Макаров С.П.	58	8
...	...	...	...

1.2.7. Контроль ввода данных в MS Excel

Для того чтобы обезопасить себя от ошибок при вводе каких-то показателей или данных, работая в Excel, мы с помощью самой программы имеем возможность контролировать правильность занесения этих самых показателей в ячейки. Для приведения контроллера в действие необходимо задать ему определённые условия и установить необходимые параметры.

Допустим, мы собираемся вводить показатели, содержащие некоторое количество нулей. При больших объёмах и продолжительной работе с цифрами возможность ошибки реальна.

Для установления контроля при вводе данных необходимо последовательно выбрать вкладку «Данные», разделы Работа с данными, Проверка данных (рис.54,55,56)

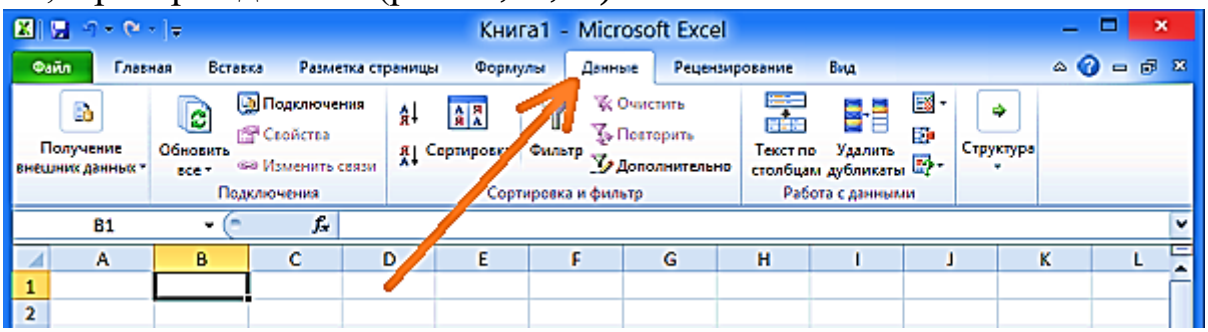

Рис. 54. Расположение вкладки Данные

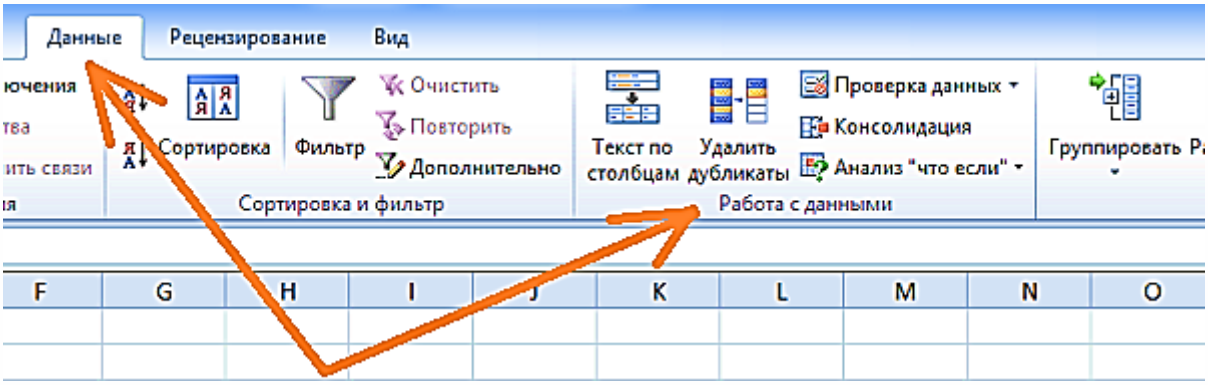

Рис. 55. Раздел Работа с данными

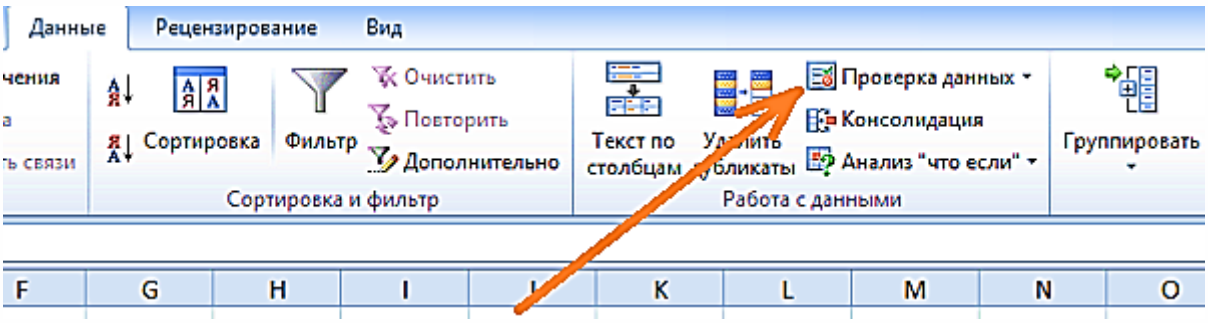

Рис. 56. Проверка данных

Требуется выбрать и зафиксировать те параметры из предлагаемых, которые и будут помогать при вводе каких-то данных (показателей). Процесс ввода может сопровождаться сообщением-предупреждением, которое можно задать, выбрав вкладку *Сообщения для ввода*.

Рассмотрим пример, демонстрирующий, как работает *Проверка данных*. Требуется ввести в ячейки показатели, содержащие не менее и не более 5-ти нулей. Ввод показателей произвести в ячейки столбца «В» (рис. 58).

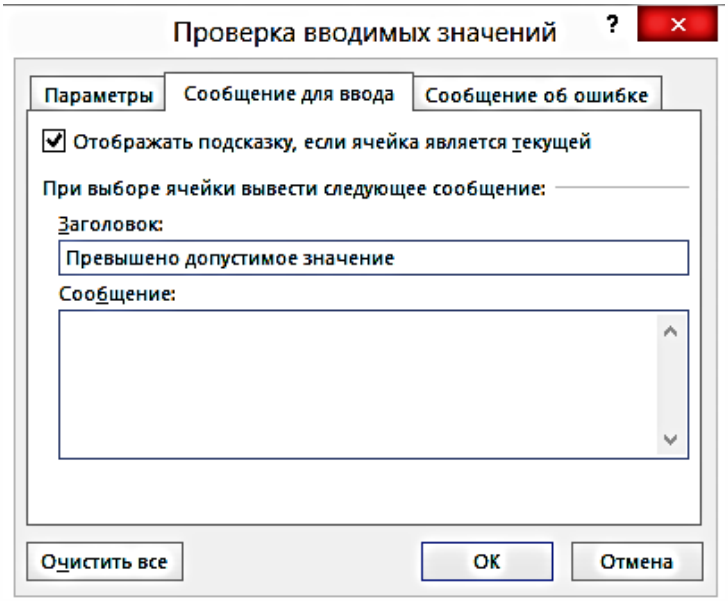

Рис. 57. Окно вкладки Сообщение для ввода

Откроем окно Проверка данных (рис.57). На вкладке *Параметры,* в поле *Тип данных* выберем из списка «Целое число». В поле «Значение» выберем из списка «между». В строку Минимум введем 100 000, в строку Максимум – 900 000. Зададим сообщение для ввода «Количество вводимых нулей равно 5» и сообщение об ошибке «введены неправильные данные» (рис. 58,59,60).

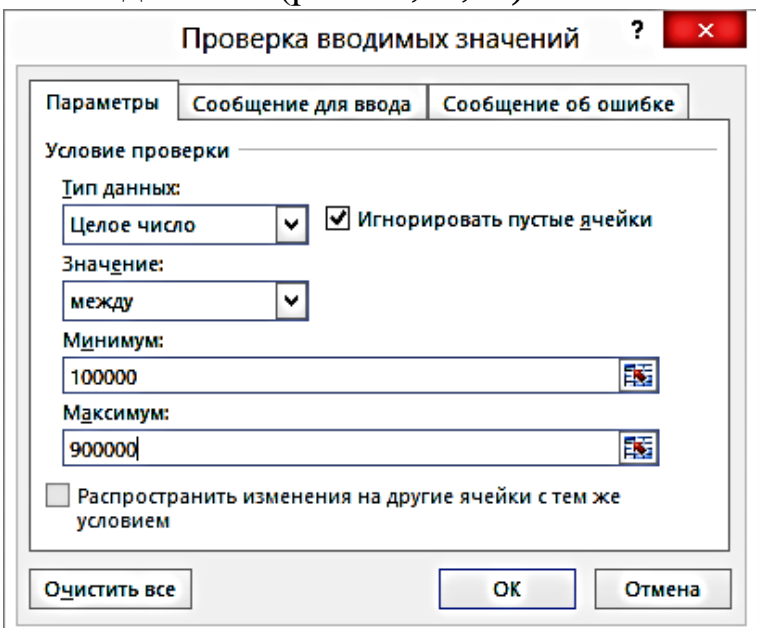

Рис. 58. Установка типа вводимых данных

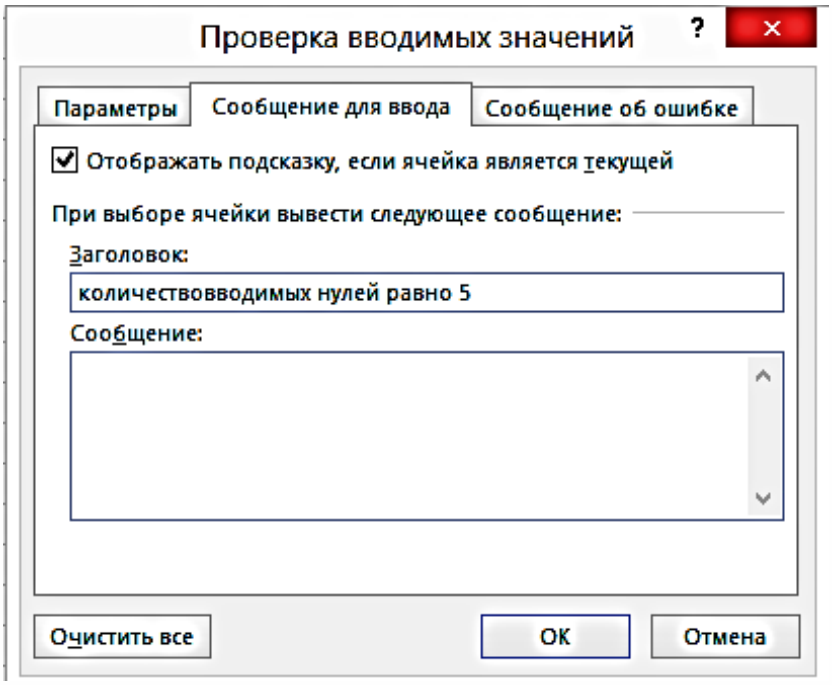

Рис. 59. Задание сообщения для вводимых значений

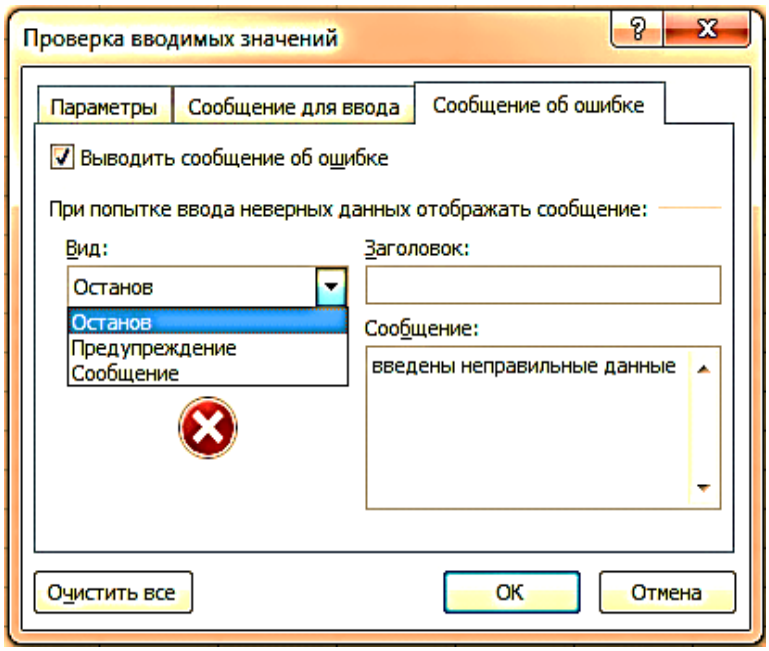

Рис.60. Задание сообщения об ошибке

При выборе вариантов «Предупреждение» или «Сообщение» раздела «Вид», каждый из них, выполнив свою задачу (предупредив или сообщив), всё же позволяет осуществить ввод показателя не соответствующего условию (рис.61).

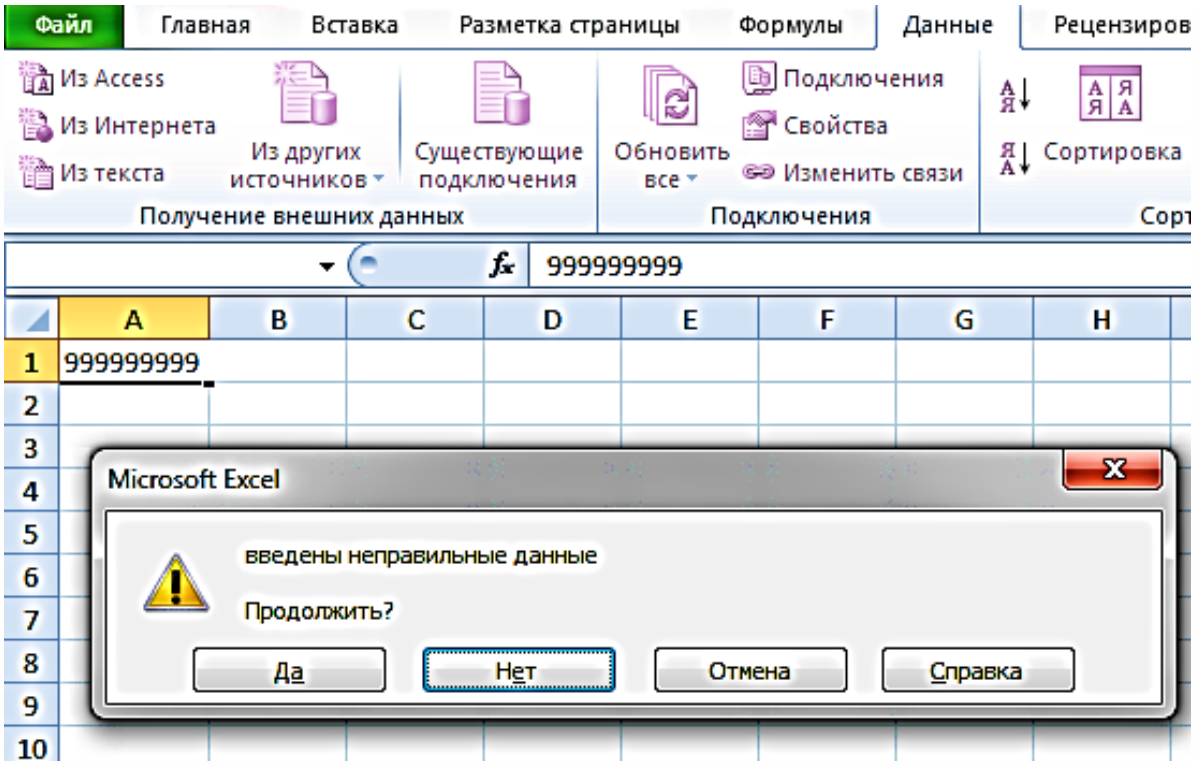

Рис. 61. Пример работы варианта «Сообщение»

В отличие от последних двух вариант «Останов» не позволит продолжать ввод данных в выделенные ячейки до тех пор пока с данными, внесёнными с нарушением условий, не будут произведены изменения или же ввод не будет прекращен.

Практическая работа № 8 «Контроль ввода данных в MS Excel»

Задание 1. Используя инструмент «Контроль ввода» ограничьте вводимые пользователем:

1) дату рождения согласно образцу ниже.

КОНТРОЛЬ ВВОДА

Задание 1

Введите дату рождения

	Дата
Значение:	**между**
Минимум:	**01.01.1989**
Максимум:	**01.04.20017**

2) стаж работы

Заполните вкладки *Сообщения для ввода* и *Сообщение об ошибке* самостоятельно.

Задание 2. Известно, что правильный артикул имеет длину 6 символов, начинается с латинской буквы, далее идут 4 цифры и заканчивается буквой русского алфавита. Используя инструмент «Контроль ввода», блокируйте ввод данных, не соответствующих этому шаблону.

Заполните вкладки *Сообщения для ввода* и *Сообщение об ошибке* самостоятельно.

Вопросы для самоконтроля

1. Назовите основные математические, логические, статистические, календарные функции MS Excel.
2. Назовите основные функции MS Excel, предназначенные для итоговых вычислений.
3. Назовите основные средства контроля правильности электронных таблиц.

Список литературы

1. Использование приложения MS Excel для моделирования различных задач [Электронный ресурс] / Кильдишов В.Д. - М.: СОЛОН-ПРЕСС, 2015. - http:// www.studentlibrary.ru/ book/ ISBN9785913591456.html.
2. Применение Excel в экономических и инженерных расчетах [Электронный ресурс] / В.А. Зеньковский - М. : СОЛОН-ПРЕСС, 2009. - http:// www.studentlibrary.ru /book/ ISBN5980032355. html

1.3. Информационные системы и базы данных

1.3.1. Основные понятия СУБД

СУБД – программное обеспечение, предназначенное для работы с базами данных (БД).

БАЗА ДАННЫХ – это организованная совокупность данных, предназначенная для длительного хранения во внешней памяти компьютера и постоянного использования.

ВИДЫ БАЗ ДАННЫХ:

По способу хранения:

- Фактографические БД – краткие сведения об объектах в строго определенном формате (каталог библиотеки, справочная картотека).
- Документальные БД – документы в различном формате: графические и мультимедиа объекты, текстовые документы, звуковая информация.
- Распределенные БД – хранение различных частей БД на множестве компьютеров, объединенных между собой сетью.

По организации данных:

- табличные (реляционные) – совокупность данных, представленная в виде таблицы;
- иерархические. Для иерархических структур (рис. 1) характерна подчиненность объектов нижнего уровня объектам верхнего уровня;
- сетевые. Сетевая база данных образуется обобщением иерархической за счет допущения объектов, имеющих более одного предка, т. е. каждый элемент вышестоящего уровня может быть связан одновременно с любыми элементами следующего уровня. Вообще, на связи между объектами в сетевых моделях не накладывается никаких ограничений. Сетевой базой данных фактически является Всемирная паутина глобальной компьютерной сети Интернет. Гиперссылки связывают между собой сотни миллионов документов в единую распределенную сетевую базу данных.

Примеры:

Реляционные БД. Предположим мы хотим собрать информацию про альбомы музыкальных групп. Пусть имеется информация о некоторых альбомах: 1965, Led Zeppelin 4, Lp, Help!, Atlantic, 1971. Lp(England), EMI. 1970, Flash Gordon, Parlophone, 1980, Led Zeppelin 3, Soundtrack, Lp, Atlantic. Этот список мало о чем говорит. Извлечь какую-либо информацию из этого набора данных практически невозможно. Представим данные в виде табл. 3.

Таблица 3. Информация об альбомах музыкальных групп

Название альбома	Год выпуска	Тип	Фирма альбома
Help!	1965	Lp (England)	Parlophone
Led Zeppelin 4	1971	Lp	Atlantic
Led Zeppelin 3	1970	Lp	Atlantic
Flash Gordon	1980	Soundtrack	EMI

В нашем примере объектами модели являются музыкальные альбомы. Свойства же этих объектов находятся в столбцах таблицы (“Название альбома”, “Год выпуска”, “Тип альбома”, “Фирма”), их называют атрибутами объектов. Таким образом, каждая строка таблицы - есть совокупность атрибутов объекта. Такую строку называют записью, а столбец - полем записи.

Иерархические БД (рис.62). Важно отметить, что в дереве, между верхними и нижними объектами, задано отношение “один ко многим” (т.е. одной группе соответствует много альбомов, одному альбому соответствует много песен).

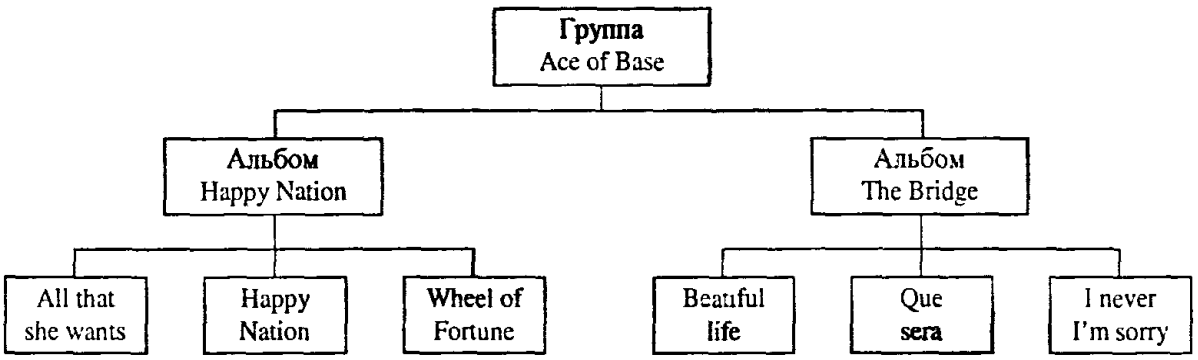

Рис. 62. Пример иерархической организации данных

Сетевую структуру данных можно представить в виде схемы, рис. 63.

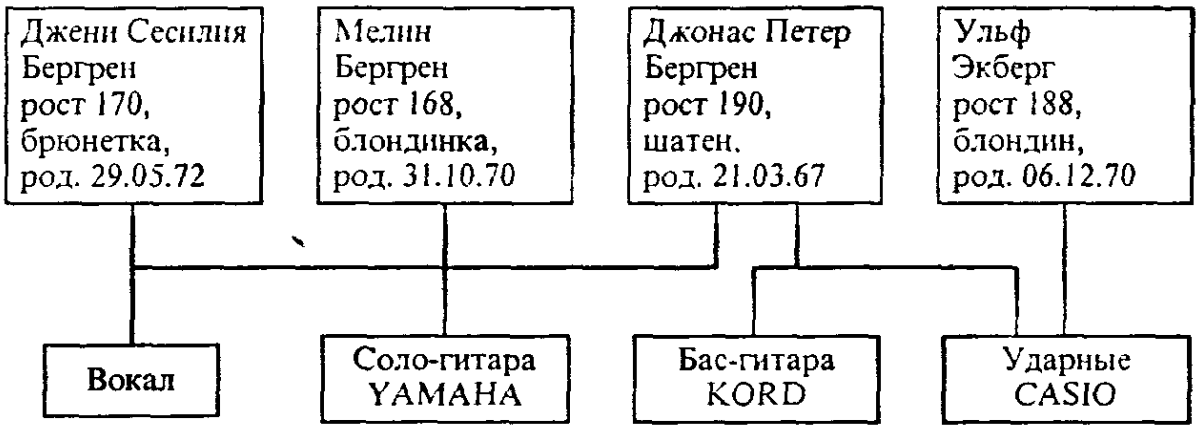

Рис. 63. Пример сетевой организации данных

1.3.2. Способы оптимизации работы таблиц

РЕЛЯЦИОННАЯ БАЗА ДАННЫХ – это база данных, основанная на реляционном способе организации данных.

Реляционные базы данных состоят из таблиц. Каждая таблица состоит из столбцов (их называют полями или атрибутами) и строк (их называют записями или кортежами). Таблицы в реляционных базах данных обладают рядом свойств. Основными являются следующие:

- В таблице не может быть двух одинаковых строк. В математике таблицы, обладающие таким свойством, называют отношениями - по-английски relation, отсюда и название - реляционные.
- Столбцы располагаются в определенном порядке, который создается при создании таблицы. В таблице может не быть ни одной строки, но обязательно должен быть хотя бы один столбец.
- На пересечении каждого столбца и строки может находиться только атомарное значение (одно значение, не состоящее из группы значений).

Таблицы, удовлетворяющие этим условиям, называют *нормализованными.*

Предположим, мы захотели создать базу данных для форума. У форума есть зарегистрированные пользователи, которые создают темы и оставляют сообщения в этих темах. Эта информация и должна

храниться в базе данных. Теоретически (на бумаге) мы можем все это расположить в одной таблице, например, так:

Имя	E-mail	Пароль	Созданные темы	Созданные сообщения

Но это противоречит свойству атомарности (одно значение в одной ячейке), а в столбцах Темы и Сообщения у нас предполагается неограниченное количество значений. Значит, нашу таблицу надо разбить на три: Пользователи, Темы и Сообщения.

Пользователи

Имя	E-mail	Пароль

Темы

Наименование	Автор

Сообщения

Текст	Автор

Типы полей

Все данные в БД разделены по типам. Вся информация полей, принадлежащих одному столбцу (домену), имеет один и тот же тип. Такой подход позволяет ЭВМ организовать контроль вводимой информации.

Рассмотрим основные типы полей в СУБД Access (рис.64).

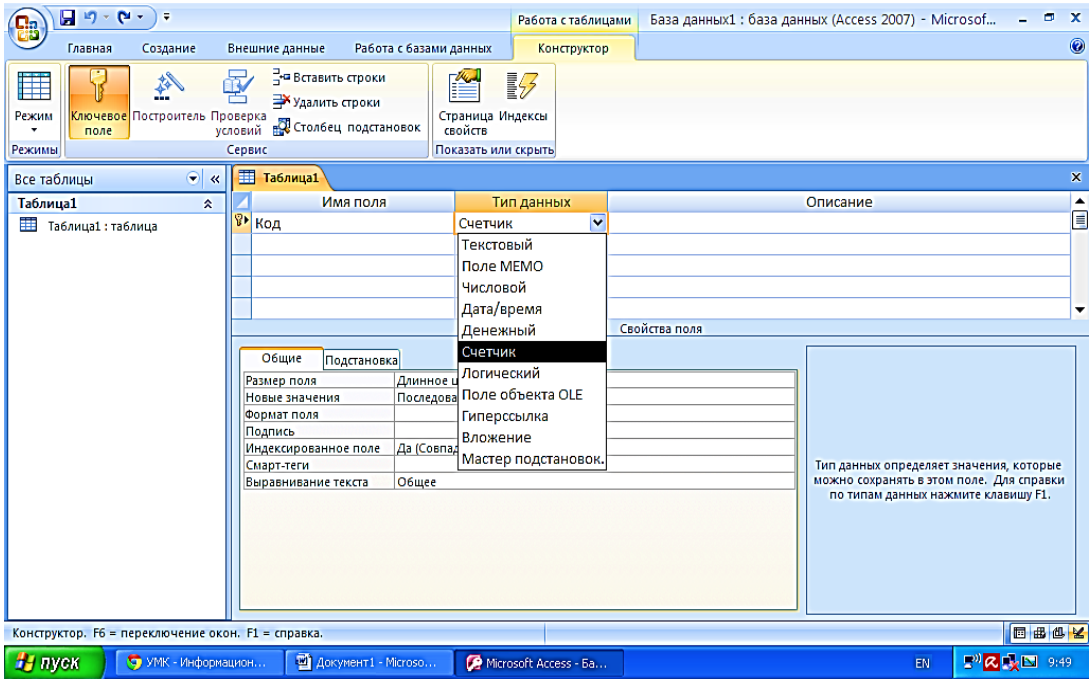

Рис. 64. Типы полей в СУБД Access

Основные типы полей баз данных:

• Символьный (текстовый). В таком поле по умолчанию может храниться до 256 символов.

• Числовой. Содержит числовые данные различных форматов, используемые для проведения расчетов.

• Дата / время. Содержит значение даты и времени.

• Денежный. Включает денежные значения и числовые данные до пятнадцати знаков целой части и четырех знаков дробной части.

• Поле примечание. Оно может содержать до 2^16 символов (2^16 = 65536).

• Счетчик. Специальное числовое поле, в котором СУБД присваивает уникальный номер каждой записи.

• Логический. Может хранить одно из двух значений: true or false.

• Поле объекта OLE (Object Linking and Embedding - технология вставки и связывания объекта). Это поле может содержать любой объект электронной таблицы, документ microsoft word, рисунок, звукозапись или другие данные в двоичном формате, внедренные или связанные с СУБД.

• Гиперссылка . Может содержать строку, состоящую из букв и цифр, представляющую адрес сайта или web - страницы.

• Мастер подстановок. Создает поле, в котором предлагается выбор значений из списка или содержащего набор постоянных значений.

Свойства полей базы данных

Поля базы данных не просто определяют структуру базы – они еще определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей.

Ниже перечислены основные свойства полей таблиц баз данных на примере СУБД Microsoft Access (рис.65):

• **Размер поля** — определяет предельную длину (в символах) данных, которые могут размещаться в данном поле.

Новые значения -

• **Формат поля** — определяет способ форматирования данных в ячейках, принадлежащих полю (основной, денежный, евро, фиксированный и т.д.).

• **Подпись** — определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство Имя поля).

• **Индексированное поле** – если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Существуют два режима индексирования: Совпадения допускаются (Duplicates OK) и Совпадения не допускаются (No duplicates). В первом случае поле может содержать повторяющиеся значения, во втором — нет. Кроме того, для индексированных полей можно сделать так, что значения в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных. Поскольку в разных полях могут содержаться данные разного типа, то и свойства у полей могут различаться в зависимости от типа данных. Так, например, список вышеуказанных свойств полей относится в основном к полям текстового типа. Поля других типов могут иметь или не иметь эти свойства, но могут добавлять к ним и свои. Например, для данных, представляющих действительные числа, важным свойством является количество знаков после десятичной запятой. С другой стороны, для полей, используемых для хранения рисунков, звукозаписей, видеоклипов и других объектов OLE, большинство вышеуказанных свойств не имеют смысла.

• **Смарт-теги** - по сути это значок, при щелчке на котором динамически отображаются некоторые сведения. Например, такой значок позволит вам получить данные из Интернет или выполнить определенные операции. Чаще всего смарт-теги применяются с целью автоматизации операций, связанных с обработкой данных других программ. Смарт-теги предоставляются Microsoft и некоторыми другими компаниями, причем, как правило, совершенно бесплатно. Более того, пользователи могут создавать и собственные смарт-теги. Например, встроенному смарт-тегу Дата соответствуют действия: запланировать собрания, показать мой календарь.

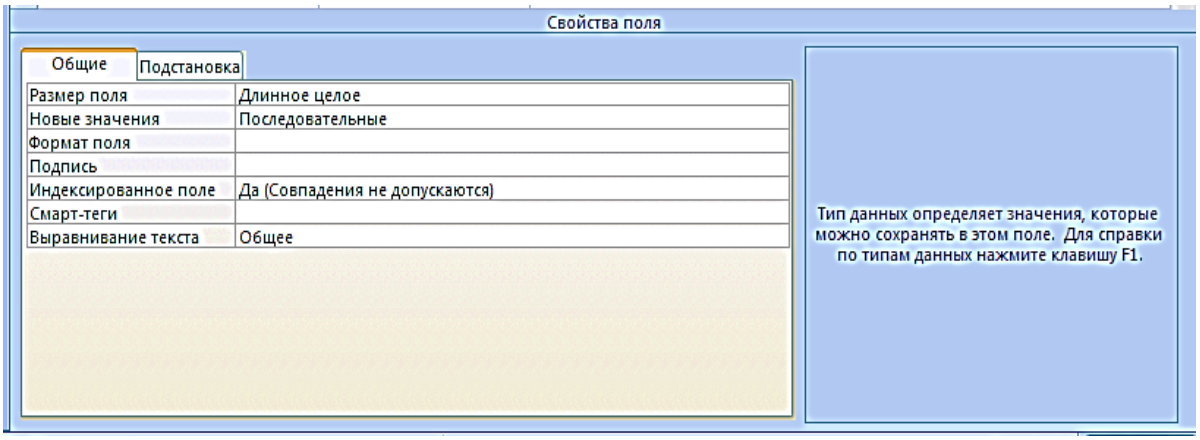

Рис. 65. Свойства полей таблицы

• **Выравнивание текста** – (по левому краю, по правому, по ширине и т.д).

Основные объекты СУБД (рис.66)

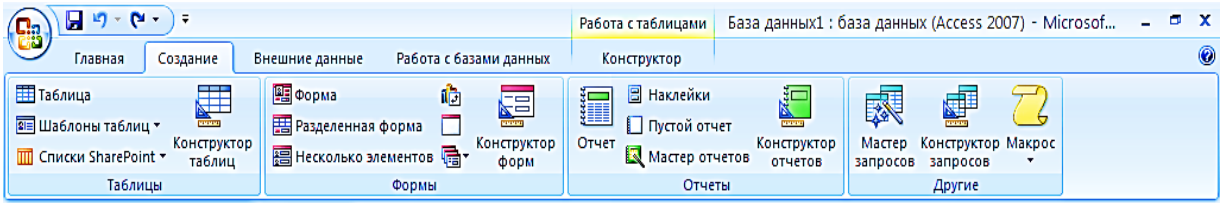

Рис. 66. Основные объекты СУБД Access

ТАБЛИЦЫ – основная часть реляционной БД. Строками таблицы являются записи, а столбцами - поля. Работать с хранимой информацией можно и в таблице (см. рис.67).

ЗАПРОСЫ – используются для выборки, поиска и сортировки данных. Запрос – это команда к базе данных, которая требует предоставить указанную информацию. Результатом выполнения запроса является производная таблица (см. рис.68).

ФОРМЫ – используются для удобного представления и ввода информации на экране. Форма похожа на обычный бланк с полями, который необходимо заполнить.

ОТЧЕТЫ – предназначены для вывода данных на печать. Позволяют группировать данные и вычислять промежуточные итоги в соответствии с заданными описаниями.

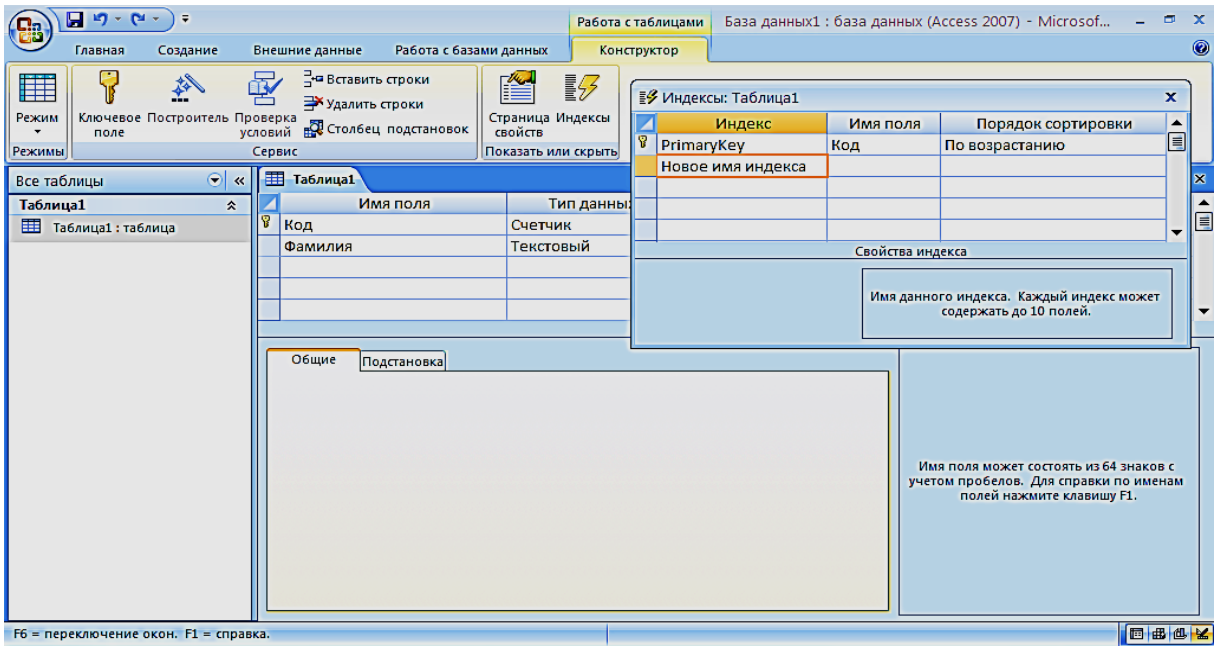

Рис. 67. Структура объекта Таблицы

1.3.3. Способы обработки информации в БД

Запросы – инструмент для работы с информацией базы данных: обновление, редактирование, отбор по определенным критериям, поиск данных.

Все запросы, имеющиеся в базе данных, отображены на вкладке Запросы окна Создание базы данных (см. рис.68).

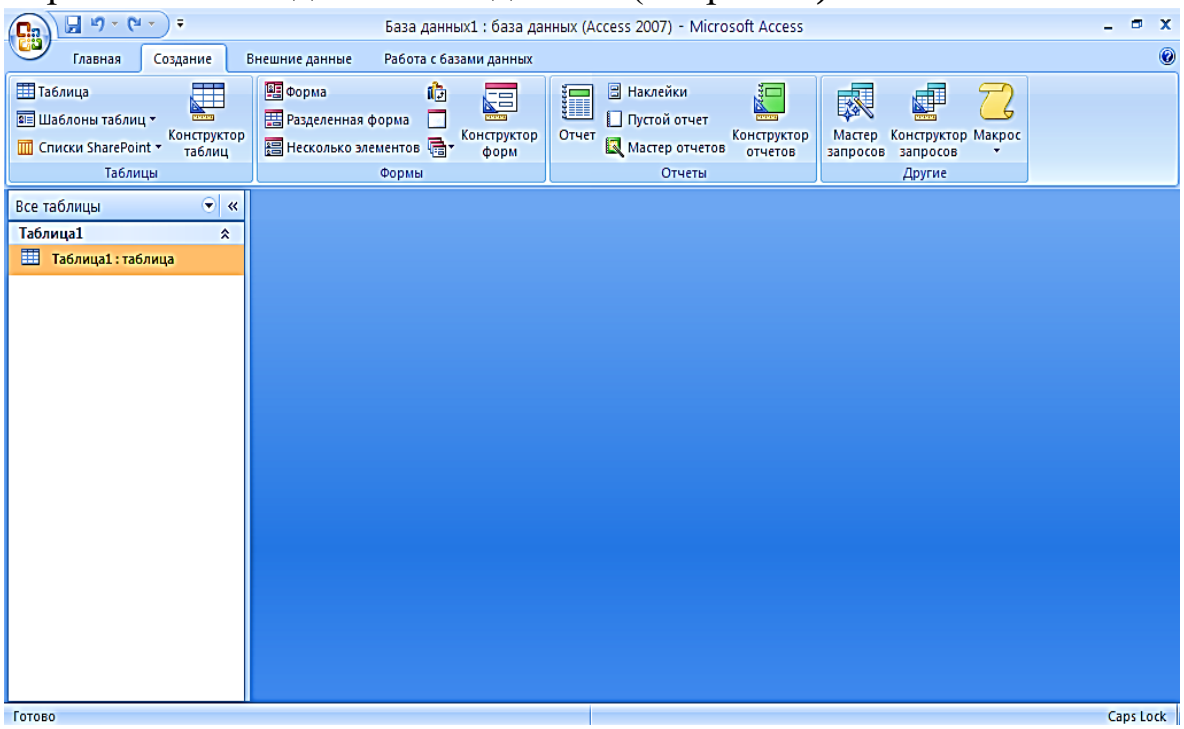

Рис. 68. Вкладка Запроса окна Создание

Запрос на выборку

Основная функция запроса на выборку – отбор данных из одной или нескольких таблиц или запросов по определенным критериям. В результате его выполнения формируется таблица, сведения которой передаются в другой объект базы данных.

Пример создания таблиц с запросом на выборку

Рассмотрим процедуру создания двух таблиц и формирования в них запроса на выборку. Для этой цели поставим перед собой задачу создать некоторый электронный журнал успеваемости учащихся. Таблицу успеваемости учащихся назовем Оценка, таблицу, содержащую список учащихся, назовем Список. Для этого выполним следующие действия:

1. Последовательно выберем в окне базы данных вкладки Создание, Конструктор таблиц.
2. В появившемся окне таблицы последовательно зададим название поля, тип данных и свойства поля (Порядковый номер, тип данных – числовой, размер – байт; Фамилия, тип данных – текстовый, размер – 50 символов; Имя, тип данных – текстовый, размер – 50 символов) см. рис. 69.
3.

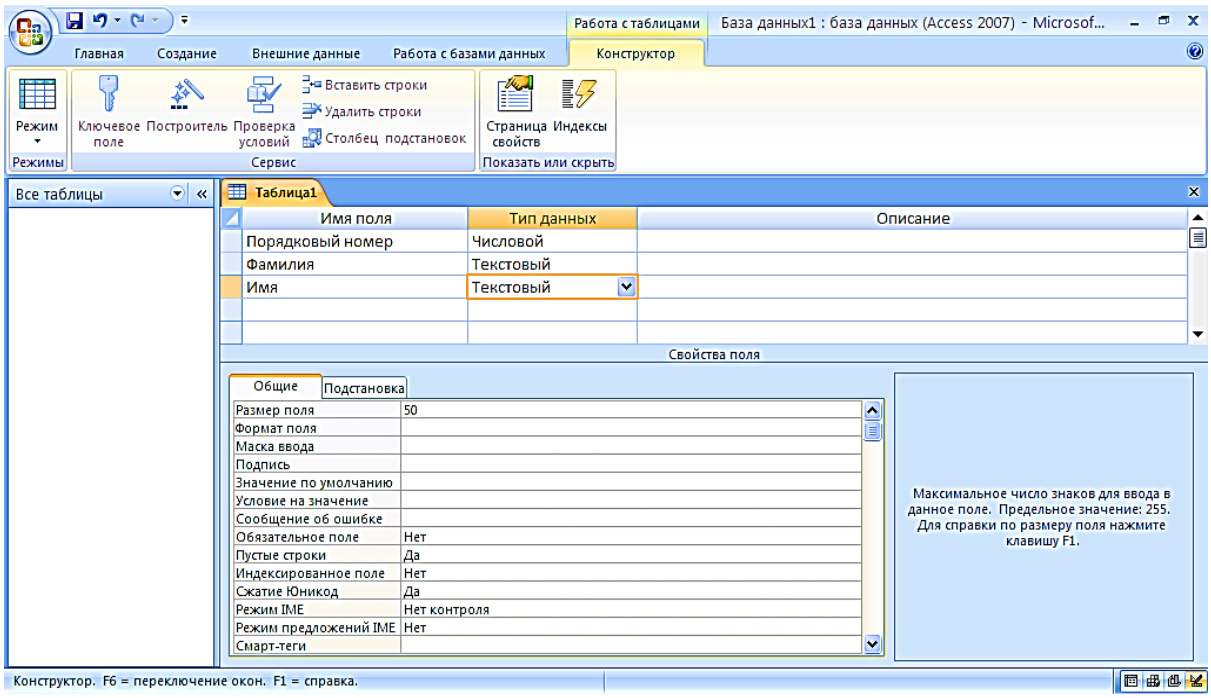

Рис. 69. Свойства поля

4. Сохраним таблицу под именем Список.

5. Аналогично создадим таблицу Оценки со следующими полями: Дата, тип данных – дата/время, Оценка, тип данных – текстовый.
6. Мы будем создавать связанные таблицы так, чтобы в таблице, содержащей список оценок, полученных за день, не нужно было дублировать информацию об учащемся. Для этой цели добавим ещё поле с именем Порядковый номер, тип данных и размер которого такой же, как в таблице Список.
7. Теперь нужно добавить ключ. Ключ – это специальный символ, однозначно идентифицирующий записи. Значения такого поля должны быть уникальными. Таким полем, очевидно, является поле Порядковый номер в таблице Список. Для того, чтобы добавить ключ:
    a. Откройте таблицу Список в режиме Конструктор;
    b. Сохраняя активной строку Порядковый номер, щелкните мышкой по кнопке Ключевое поле. В результате у строки Порядковый номер появится значок ключа;
    c. Закроем окно Конструктора. Сохранив изменения в таблице Список.
8. Заполним таблицы.
9. Свяжем таблицы. Для этого:
    a. Последовательно выберите вкладки Работа с базами данных, Схема базы данных.
    b. Добавьте таблицы на схему.
    c. Перетащите нужное поле из одной таблицы в другую.
10. Создадим запрос на выборку (рис.70,71). Для этого:
    a. Последовательно выберите вкладки Создание, Мастер запросов, Простой запрос.
    b. В качестве источника данных последовательно выберите таблицу Список, поле Фамилия, таблица Оценки, поля Дата и Оценка.
    c. Сохраните запрос.
    d. Откройте запрос в режиме конструктора. В строке Условие отбора наберите [Введите фамилию ученика].
    e. Сохраните запрос.
    f. Запустите запрос на выполнение. В окне ввода введите фамилию ученика.

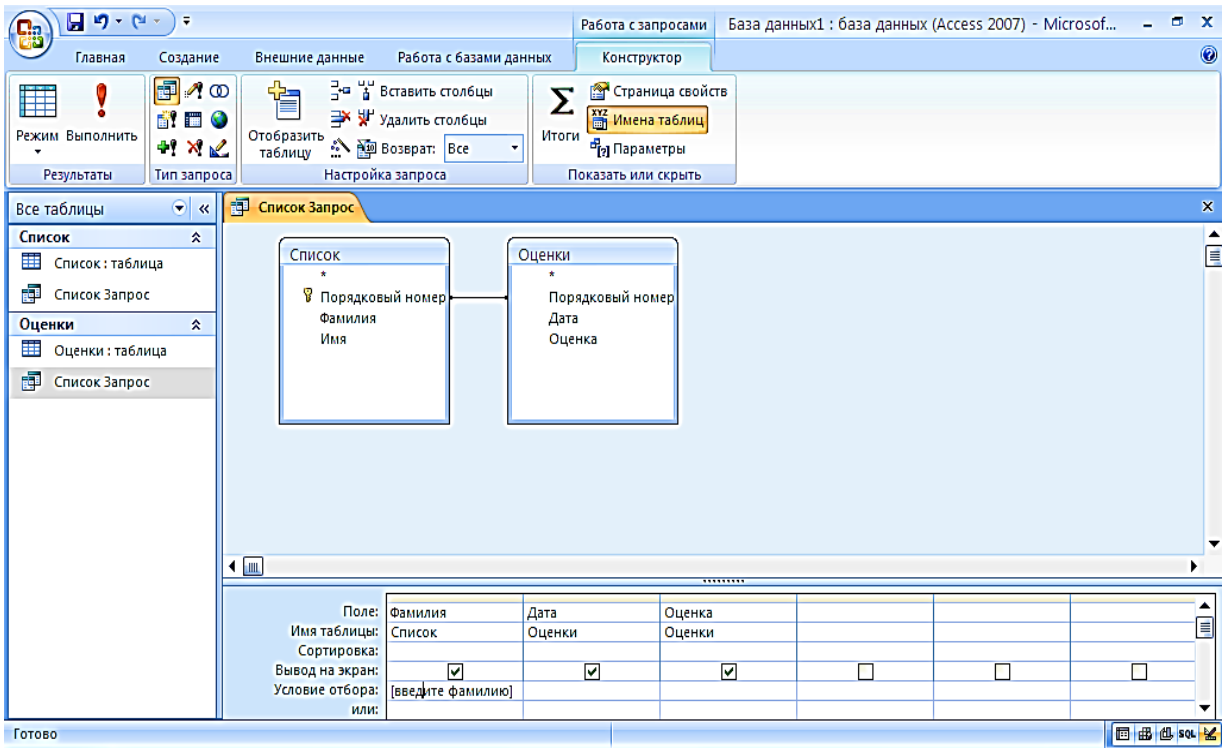

Рис. 70. Создание запроса на выборку

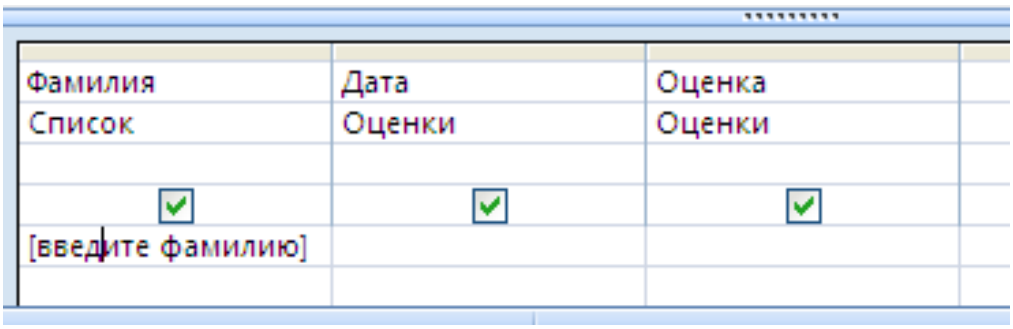

Рис. 71. Запрос на выборку в режиме Конструктор

Запрос на обновление. Находит некоторые записи и затем изменяет их. Возможные обновления:

– ввести новое значение в поле;

– перемещение данных из одной категории в другую;

– изменение текущих значений с помощью выражения.

Запрос на обновление. Введение нового значения в поле.

Рассмотрим процедуру создания запроса на обновление. С этой целью в таблицу Список введем запись о студенте по фамилии Бал-

дин. Обновим данные в таблице Список, заменим фамилию Балдин на Амафутский. Для это выполните следующую последовательность действий:

• Создайте простой запрос, включив в него все поля таблицы Список.

• Откройте запрос в режиме Конструктора.

• На вкладке Тип запроса выберите Запрос на обновление

• В появившейся строке *Обновление* задайте новое значение поля Фамилия, например, Амафутский.

• В строке *Условие отбора* – значение поля Фамилия, которое Вы хотите заменить, например, Балдин.

Перекрестные запросы - это запросы, в которых происходит статистическая обработка данных, результаты которой выводятся в виде таблицы. Могут служить для расчета сумм, средних значений, общего количества позиций и т.д.

Рассмотрим процедуру создания перекрестного запроса. Для этой цели поставим перед собой задачу вывести самую плохую оценку ученика:

• Создадим простой запрос, включив в него поле Фамилия из таблицы Список, поля Дата и Оценки из таблицы Оценки (рис.72,73,74,75).

• На вкладке Создание, Мастер запросов выберите Перекрестный запрос.

• Выберите поля, которые будут являться именами полей в перекрестном запросе. Очевидно, что значение поля Оценка должно быть на пересечении полей Фамилия и Дата.

• Выберите тип вычислений в ячейке, находящейся на пересечении строки (фамилия) и столбца (оценка). В нашем случае, для нахождения наихудшей оценки студента надо выбрать функцию Минимум.

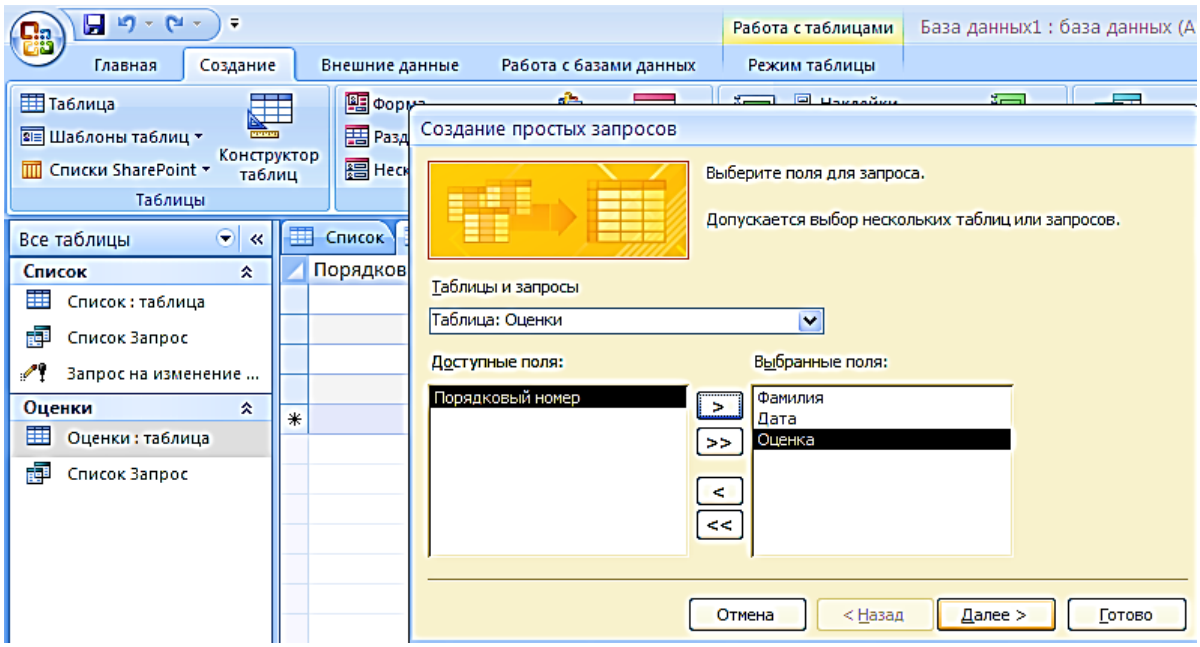

Рис. 72. Создание простого запроса

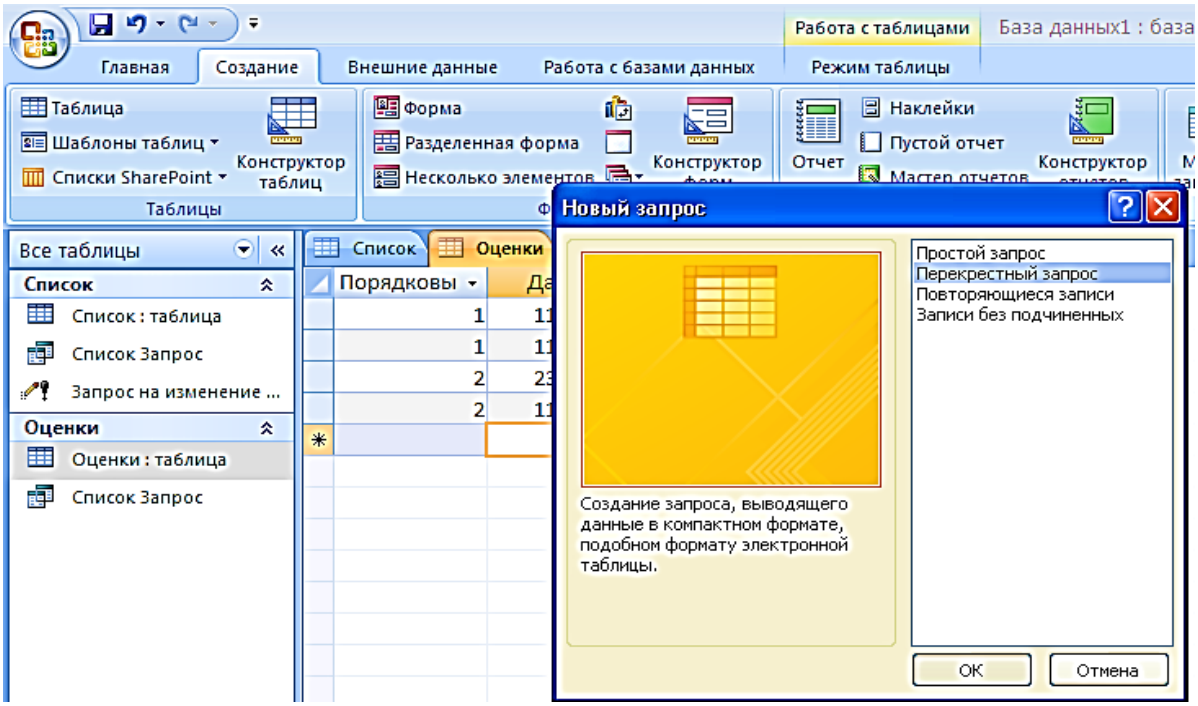

Рис. 73. Выбор перекрестного запроса

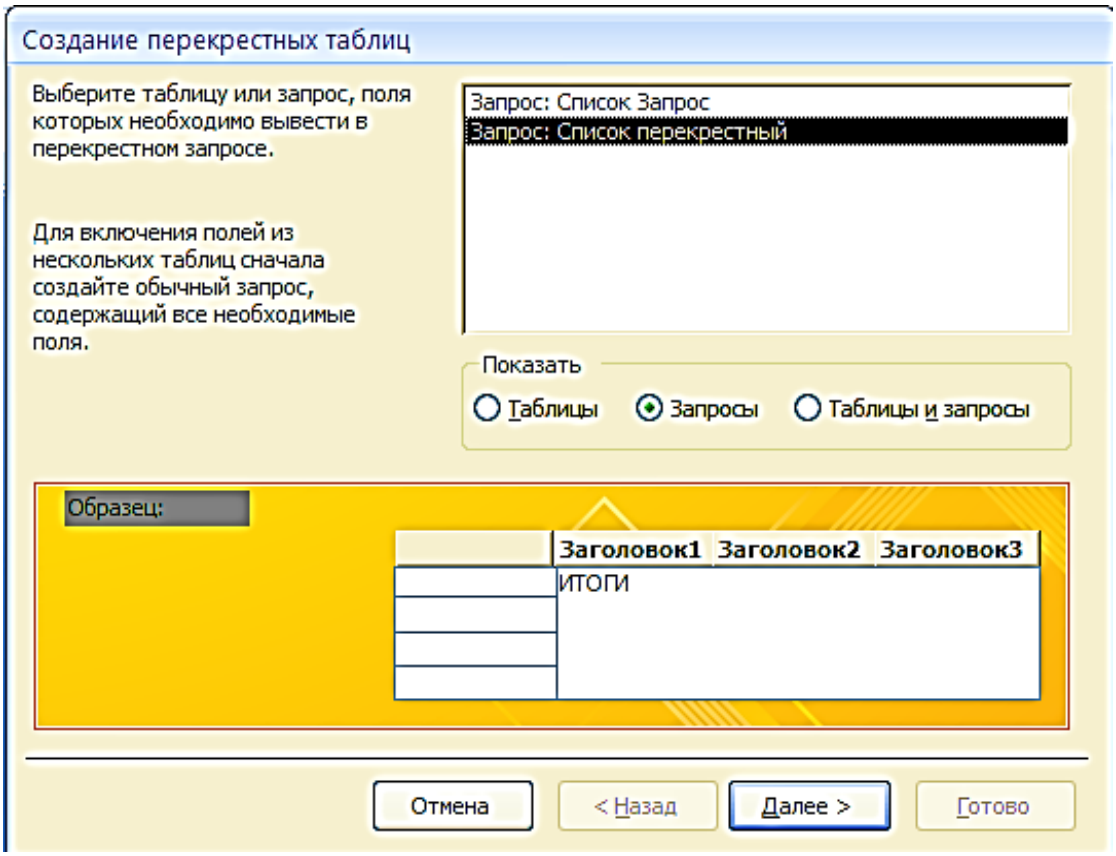

Рис. 74. Создание перекрестных таблиц

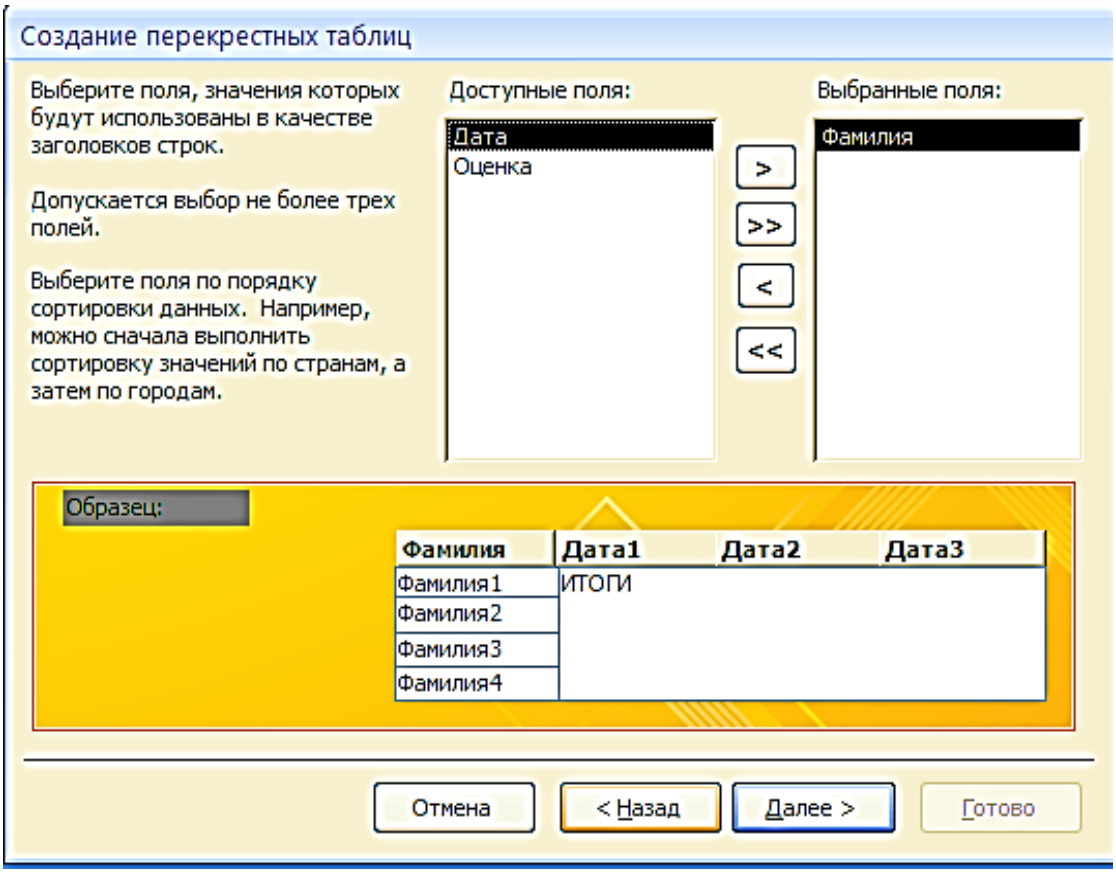

Рис. 75. Выбор заголовков строк

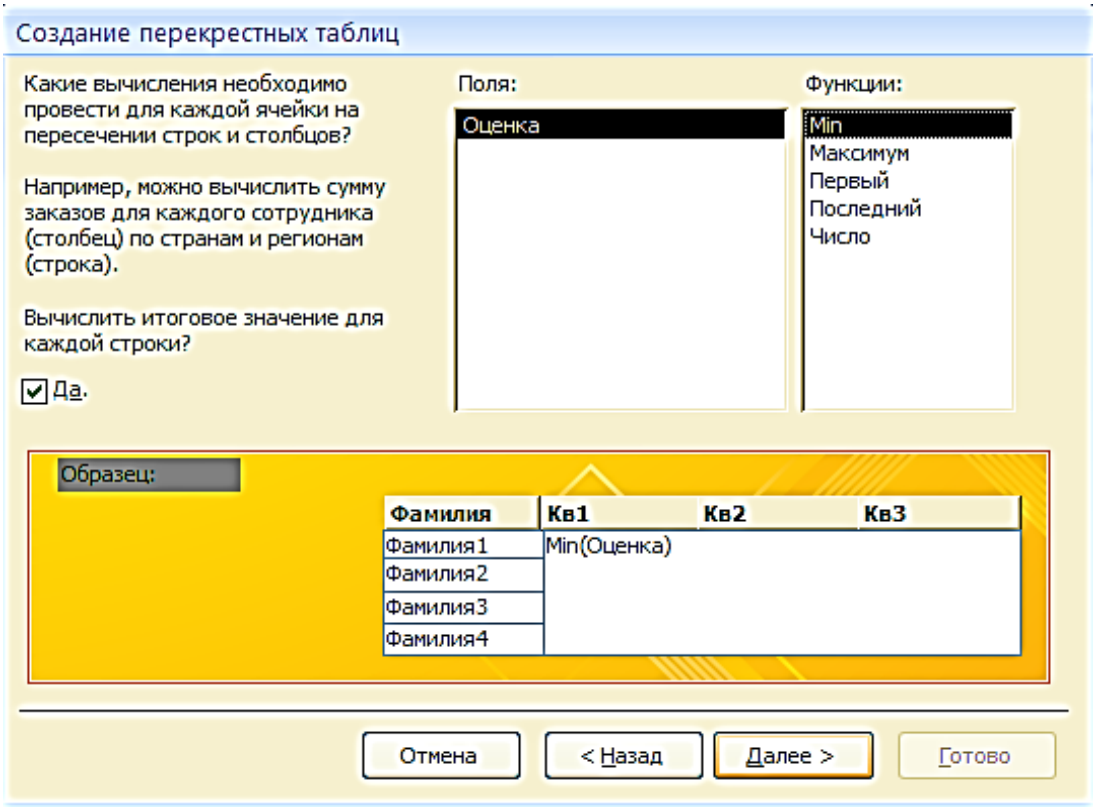

Рис. 76. Выбор вида вычислений

- Запустите запрос на выполнение

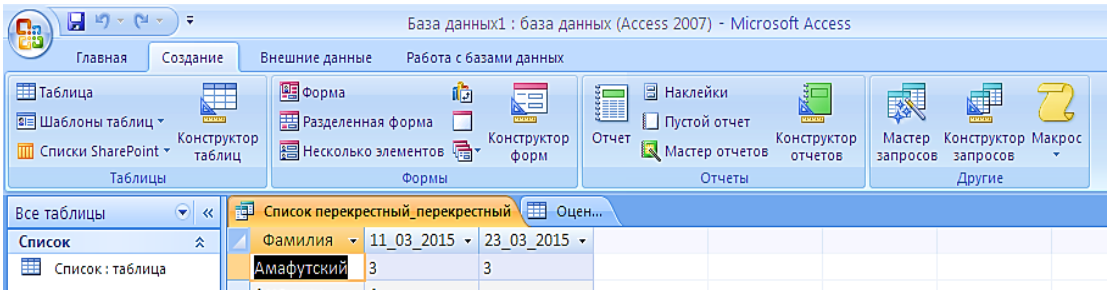

Рис. 77. Перекрестный запрос, выводящий минимальную оценку студента

Запросы на удаление действуют во многом так же, как запросы на выборку: задается ряд условий отбора, согласно которым записи в таблице отбираются и затем удаляются из БД.

Рассмотрим создание запроса на удаление на примере нашей БД. Для этой цели поставим перед собой задачу удалить данные о студенте по фамилии Амафутский. Для этого выполним следующие действия:

- Создайте простой запрос последовательно выбрав: Создание, Мастер запросов, Простой запрос.

• В появившемся диалоговом окне "Добавление таблицы" выберите таблицу, содержащую записи, которые вы хотите удалить.

• В Конструкторе запросов измените тип вашего запроса на запрос на удаление. Значение в появившейся строке Удаление измениться на Условие.

• В строке Условие отбора задайте условие, по которому будут удалены записи из таблицы. В нашем случае в поле Фамилия необходимо ввести «Амафутский».

• Выполните предварительный просмотр удаляемых записей. Для этого:

- откройте таблицу Список и два раза щелкните мышкой по строке, содержащей символ «*»;
- откройте в режиме Конструктор запрос на удаление. Значение в строке Удаление автоматически изменилось на FROM (Из).
- встаньте на заголовок таблицы Список, нажмите правую кнопку мыши и выберите Режим таблицы. Вы увидите записи, которые хотите удалить.

Если вы уверены, что получена корректная информация (в нашем случае, вы должны увидеть одну строку с фамилией Амафутский), выберите на ленте *Работа с запросами, Выполнить для удаления записей*.

Фильтры, как и запросы, предназначены для отбора определённых записей базы данных.

В отличие от запросов фильтры:

- требуют каждый раз задавать критерии отбора;
- не позволяют объединять таблицы;
- не сохраняются как отдельный объект в окне БД;
- не позволяют осуществлять обработку данных.

Фильтры целесообразно использовать при работе в режиме формы или таблицы для просмотра или изменения подмножества записей.

Фильтр по выделенному фрагменту

Данный фильтр используется для поиска записей, содержащих выделенный фрагмент.

В качестве примера установки этого типа фильтра обратимся к таблице Список нашей базы данных. Для этой цели поставим перед собой задачу отобрать записи, содержащие имя «Александр». Для этого выполним следующие действия:

1) откроем таблицу «Список»;
2) выделим имя «Александр», как представлено на рис.78;
3) последовательно выбрать в меню: Главная, Сортировка и фильтр, Выделение, Равно «Александр» как представлено на рис. 79;
4) в результате видимыми останутся только те записи, в поле Имя которых присутствует значение выделенного имени (см. рис.80);
5) для отмены влияния фильтра последовательно выберите на вкладке Сортировка и фильтр Фильтр, Снять фильтр с Имя (см. рис.81).

Аналогичным способом можно исключить записи в поле Имя которых присутствует значение выделенного имени.

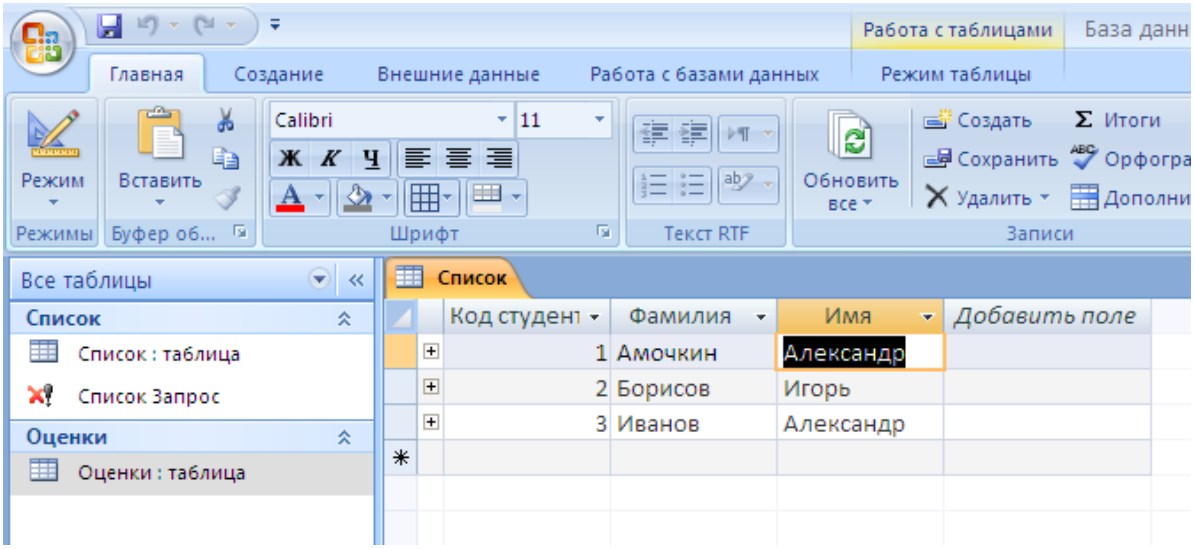

Рис. 78. Применение фильтра по выделенному фрагменту. Шаг 2

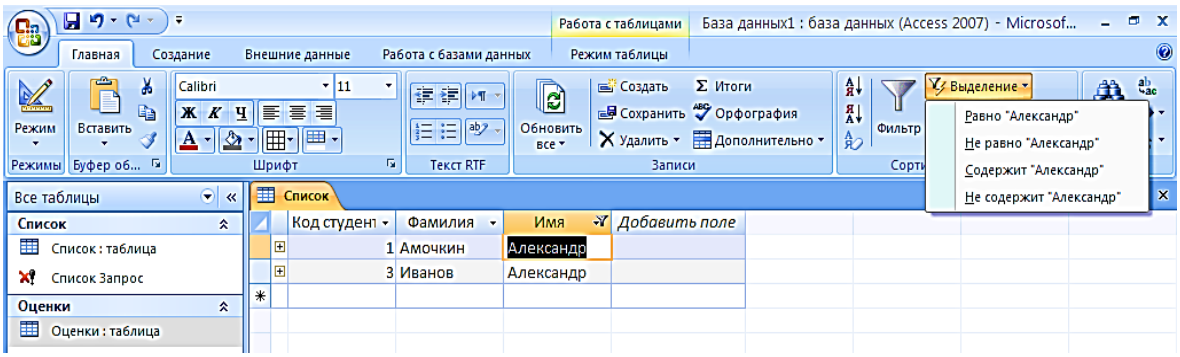

Рис. 79. Применение фильтра по выделенному фрагменту. Шаг 3

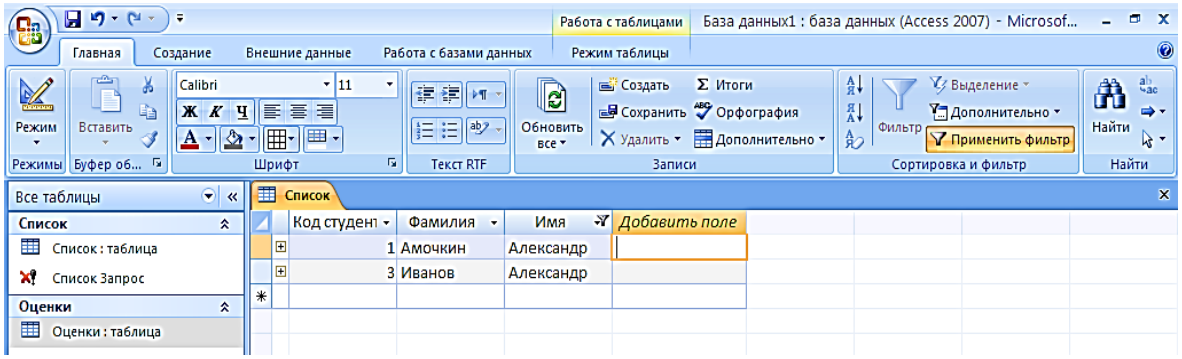

Рис. 80. Таблица после применения фильтра по выделенному фрагменту

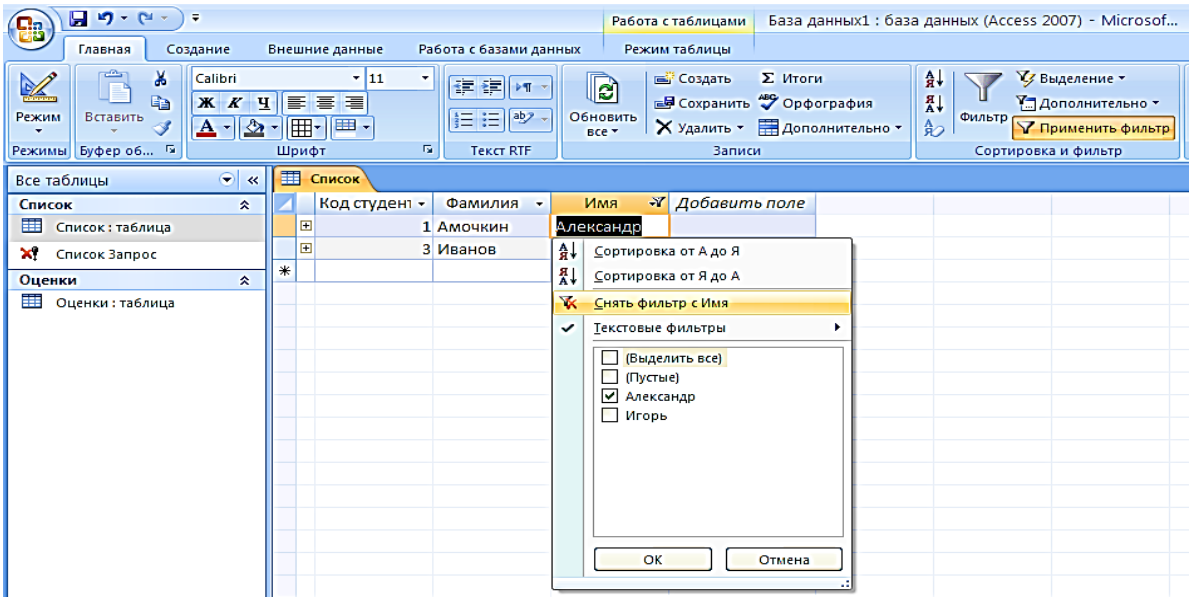

Рис. 81. Отмена влияния фильтра по выделенному фрагменту

Фильтр «Изменить фильтр»

Этот вид фильтров используется для отбора записей по нескольким критериям, в отличие от фильтра «По выделенному фрагменту», позволяющему задать только один критерий отбора. Критерии фильтра – значения, которые должны содержаться в отбираемых полях.

Для того, чтобы начать работать с фильтром «Изменить фильтр» необходимо последовательно выбрать вкладки Главная, Сортировка, Фильтр, Дополнительно, Изменить фильтр (см. рис.82).

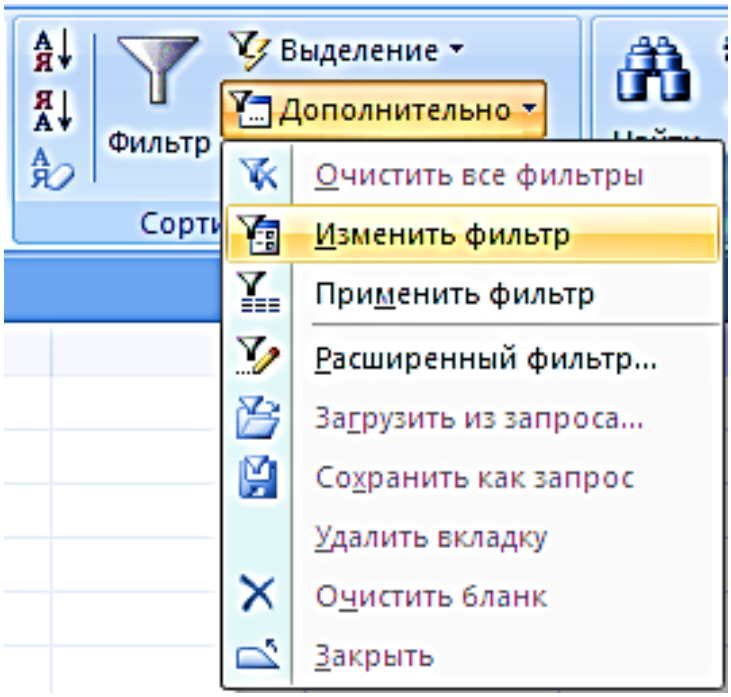

Рис. 82. Выбор фильтра «Изменить фильтр»

Для выбора критериев фильтра служит встроенный список в каждом столбце (см. рис.83). **Важно:** *В окне Изменить фильтр недоступны те типы полей, которые не могут быть использованы в фильтре.*

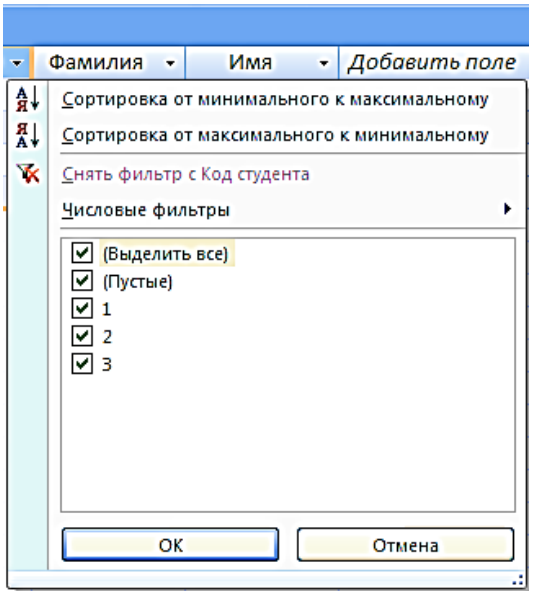

Рис. 83. Встроенные списки в столбцах таблицы

В качестве примера установки этого типа фильтра обратимся к таблице *Список* нашей базы данных. Поставим перед собой задачу отобрать записи, фамилия в которых начинается с буквы А, и содержащие имя «Александр». Для этого выполним следующие действия:

1) откроем таблицу «Список»;
2) откроем встроенный список в поле Фамилия, последовательно выберем строки Текстовые фильтры, Начинается с… (см. рис.84). нажмем ОК;
3) откроем встроенный список в поле Имя, поставим галочку на строке «Александр», нажмем ОК (см. рис.85).

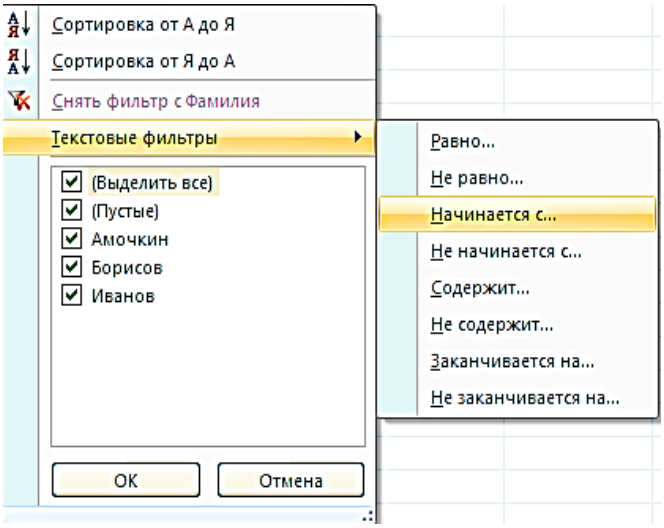

Рис. 84. Применение фильтра «Изменить фильтр» к полю Фамилия»

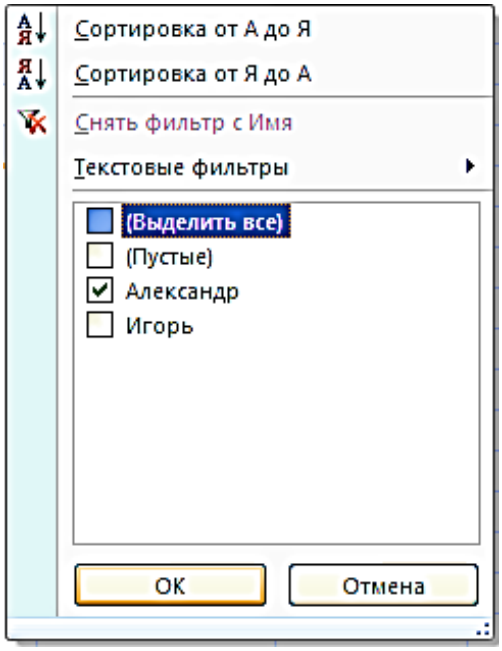

Рис. 85. Применение фильтра «Изменить фильтр» к полю Имя

В результате видимыми останутся только те записи, в поле Имя которых присутствует значение «Александр» и поле Фамилия содержит значения, начинающиеся на букву А (см. рис.86).

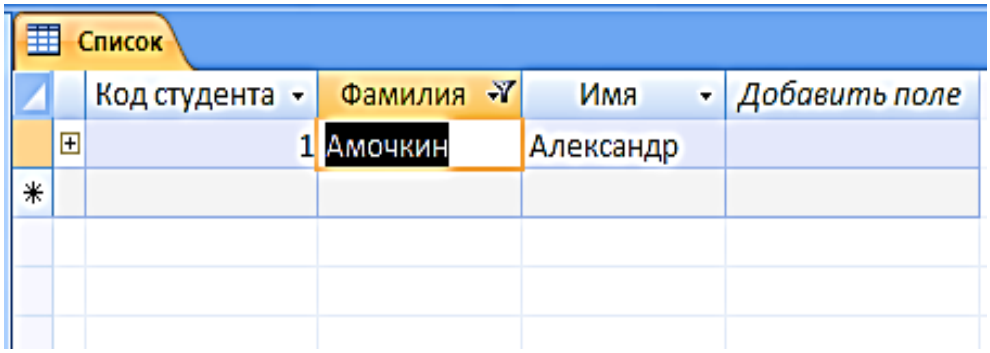

Рис. 86. Таблица после применения фильтра «Изменить фильтр»

Расширенный фильтр

Расширенный фильтр представляет более широкие возможности отбора данных, по сравнению с двумя предыдущими видами фильтров. Параметры расширенного фильтра задаются в диалоговом окне, которое вызывается командой Главная, Сортировка, Фильтр, Дополнительно, Расширенный фильтр.

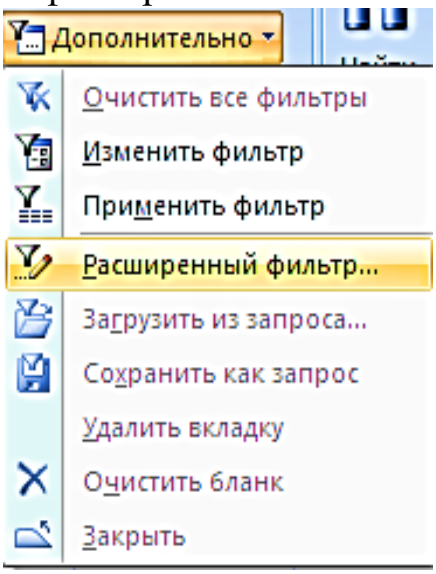

Рис. 87. Расширенный фильтр

Основное отличие расширенного фильтра от двух предыдущих видов фильтров состоит в том, что в данном фильтре можно задавать критерии с помощью логических выражений, что существенно облегчает выборку данных из таблиц.

Логические выражения и функции, которые можно использовать при задании критериев фильтрации

1. Использование текстовых значений в выражениях:
    - Between #номер месяца/день/год AND #номер месяца/день/год – отображает записи, содержащие дату между указанными значениями. Например, Between #1/17/03 AND #1 месяца/24/03 отображает записи, содержащие дату между 17 и 24 января 2003 года включительно (знак # указывает на то, что имеются в виду даты);
    - In (текстовое выражение1, текстовое выражение2, … текстовое выражение n) – оператор In используется для отбора записей, в которых встречается одно из текстовых выражений, указанных в скобках). Например, In («Москва», «Ростов») отберет записи, в которых встречается одно из текстовых выражений Москва, Ростов;
    - Like «*сочетание*» - оператор используется для отбора записей, содержащих сочетание. Например, Like «*мор*» используется для отбора записей, содержащих сочетание «мор»;
    - Right ([поле], n) = «m» - с помощью функции Right отбираются записи, в которых значения в поле заканчиваются m, n указывает на то, сколько знаков в поле должно совпадать с m. Например, Right ([Количество], 2) = «99» - отберёт записи, в которых значения в поле Количество заканчиваются цифрами 99. Причём правые два знака в поле должны совпадать с 99;
    - Len служит для измерения длины содержимого поля. Например, Len ([Фирма]) > Val (“30”) отобразит записи, где поле Фирма содержит строку текста длиной не более 30 знаков;
    - Not in (“текстовое выражение1” , ”текстовое выражение2”, …, ”текстовое выражение n”) – выражение отбирает записи, в которых значение соответствующего поля не содержит ни одного из указанных текстовых выражений. Например, выражение Not in (“Москва”, ”Ростов”) отберёт записи, в соответствующем поле которых нет текстовых выражений Москва и Ростов.

В качестве примера использования логических выражений в критериях фильтрации обратимся к таблице Оценки нашей базы данных. Для этой цели поставим перед собой задачу отобрать записи с оценками с 1 апреля по 5 апреля 2015 года. Для этого выполним следующую последовательность действий:

1) Откроем таблицу Оценки;
2) На вкладке Главная последовательно выберем Сортировка, Фильтры, Дополнительно, Расширенный фильтр;
3) В появившемся окне в строке Поле выберем Дата, в строке Условие отбора запишем выражение: Between#4/1/15#AND4/5/15# (см. рис. 88);
4) На вкладке Сортировка и фильтры выберите Применить фильтр. В результате видимыми останутся записи с указанным диапазоном дат (см. рис.89).

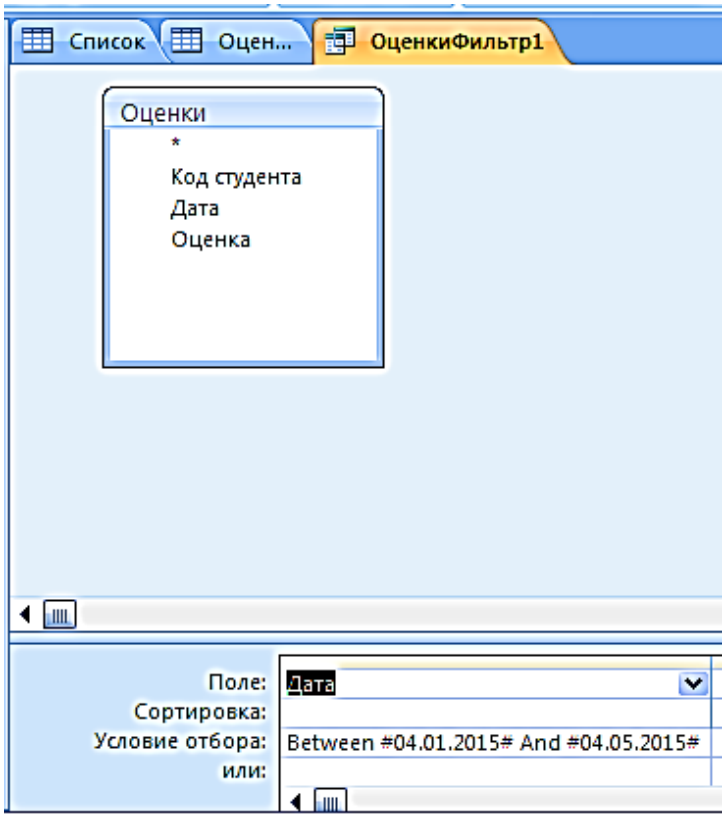

Рис. 88. Применение логических выражений при задании критериев

Важно: *Для отмены влияния фильтра можно воспользоваться кнопкой Удалить фильтр.*

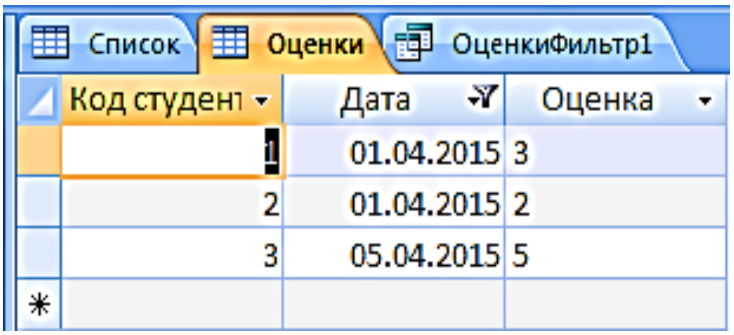

Рис. 89. Результат применения расширенного фильтра

Понятие макроса. Макросы представляют собой мощный и хорошо проработанный инструмент, помогающий решать достаточно сложные задачи, не прибегая к программированию.

Запускаются практически из любого объекта базы данных: назначены командными кнопками в форме или отчете, кнопками на панели инструментов, могут быть связаны с событиями в свойствах какого-либо из объектов, запущены из кода программы на языке Visual Basic либо из окна базы данных, где отображены все существующие поля макроса.

С помощью макроса можно автоматизировать такие операции как: отобразить или скрыть определенные панели инструментов, открыть объекты базы данных, создать собственные элементы меню.

Однако, существует ряд ситуаций, в которых использование макроса нежелательно:

- создание объектов базы данных, подлежащих дальнейшему копированию и использованию в других базах данных. В этом случае макрос, автоматизирующий какую-либо операцию с этим объектом, не скопируется вместе с ним, так как является самостоятельным объектом базы данных.
- создание собственных функций. Этого нельзя сделать с помощью макросов. Выражения, созданные с помощью макросов, имеют меньше возможностей, чем функции Visual Studio.
- программный код процедур позволяет работать с каждой записью таблицы по отдельности, а макрос может оперировать только таблицей целиком.

Список всех существующих макросов конкретной БД находится на панели Все объекты Access (см. рис. 90)

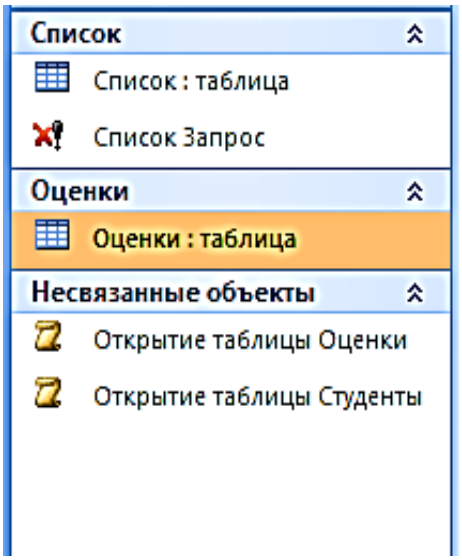

Рис. 90. Список существующих макросов в Базе данных

Создание макроса. Если вы хотите создать новый макрос, выберите вкладку Создание, Макрос. Появится диалоговое окно режима конструктора макроса, имеющее три столбца: Макрокоманда, Аргументы и Примечание (см. рис.91аб).

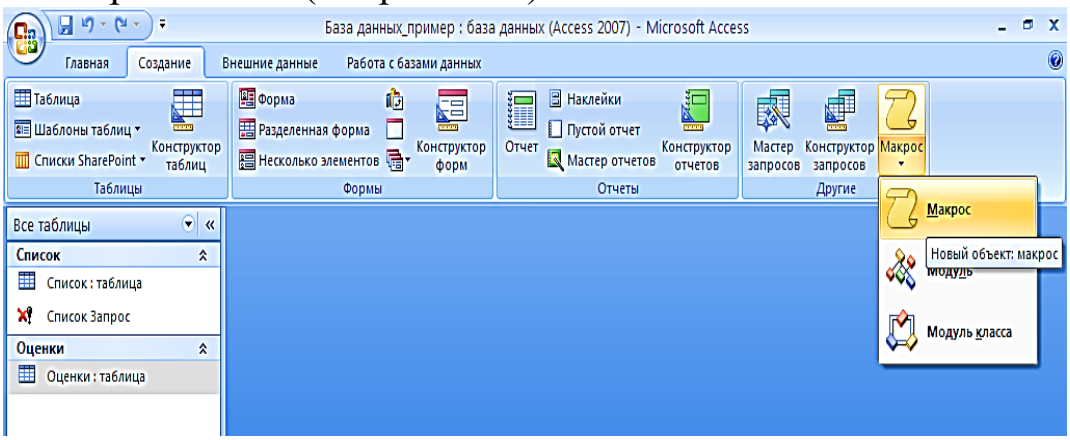

Рис. 91а. Создание нового макроса

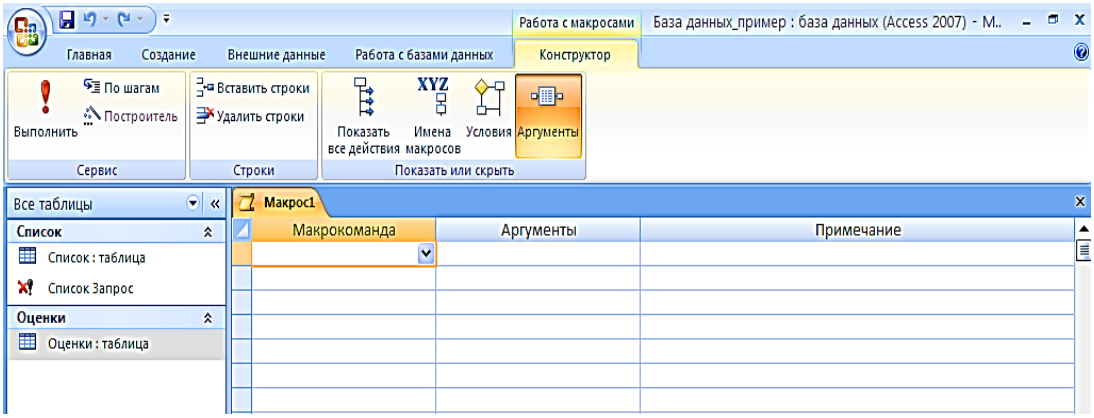

Рис. 91б. Диалоговое окно Макрос

После выбора макрокоманды на панели могут появиться строки, предназначенные для задания значений аргументов соответствующей макрокоманды. Заданные значения аргументов также появляются во втором столбце ***Аргументы*** панели описаний.

Разработаем макрос для открытия таблицы Список нашей базы данных. Последовательность выполнения:

1) откройте окно Конструктор макросов;
2) в поле Макрокоманда выберите действие Открыть таблицу (см. рис.92);
3) в поле аргументы макрокоманды в строке Имя таблицы выберите из списка таблицу Студенты (см. рис.93);

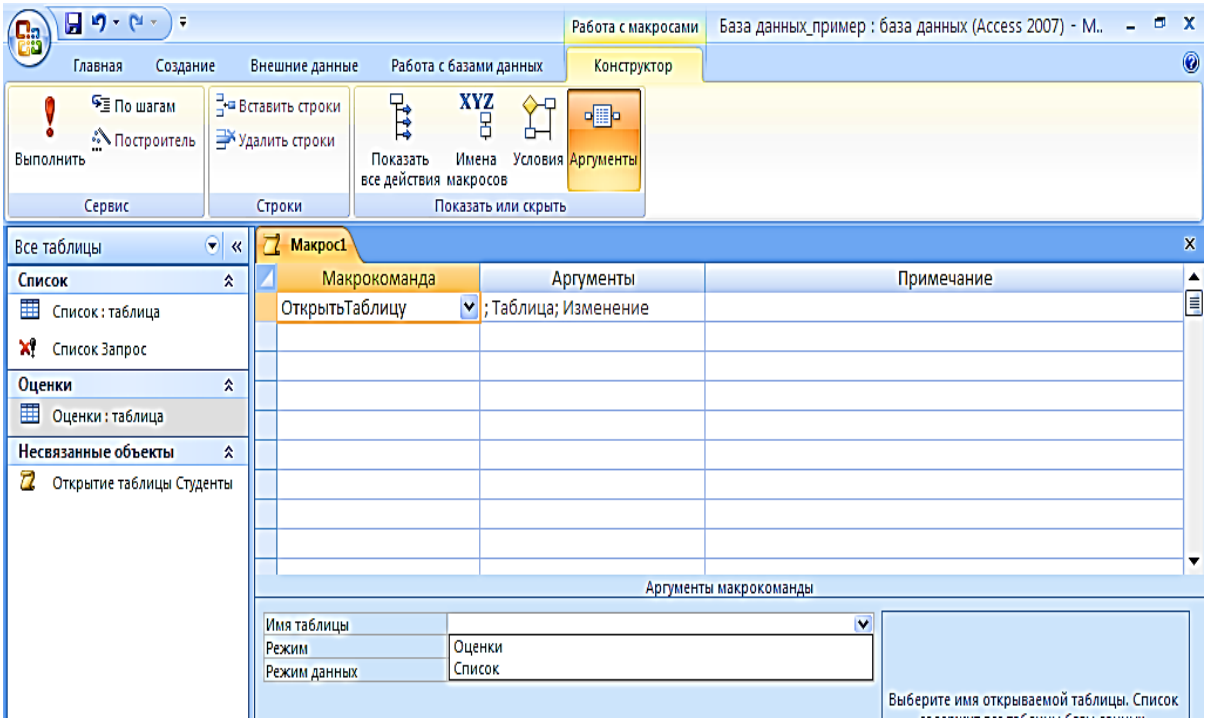

Рис. 92. Создание макроса, открывающего таблицу Студенты

4) сохраните макрос. Для этого закройте окно конструктора макросов, в появившемся окне Сохранение наберите имя макроса, например, открытие таблицы Студенты (см. рис.93).

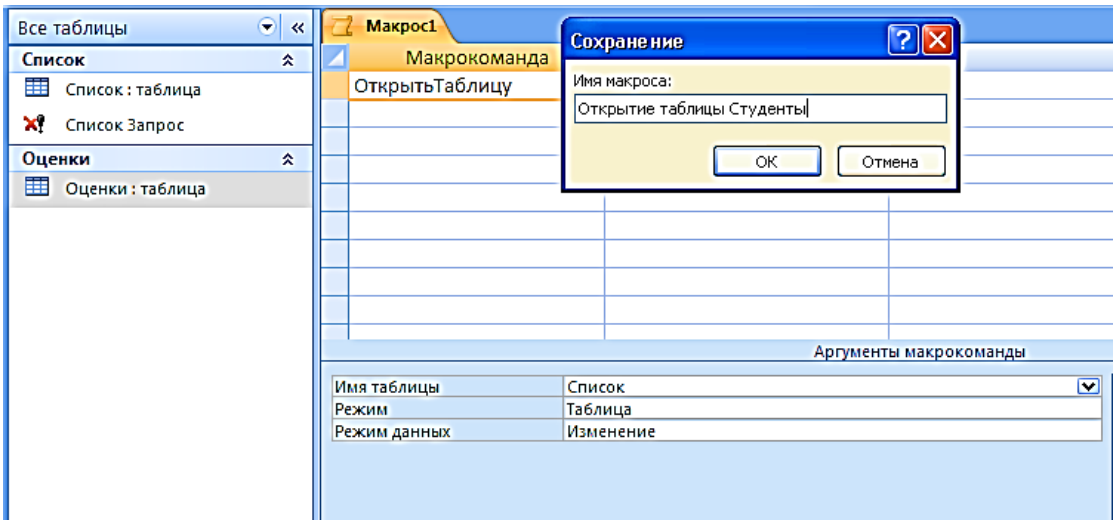

Рис. 93. Сохранение макроса

В столбце Макрокоманда указываются действия, выполняемые макросом в той последовательности, в которой они внесены в данный столбец. Число макрокоманд строго определено, и каждая из них отображена в списке, раскрывающемся при нажатии на кнопку со стрелкой в правой части поля Макрокоманда.

Практическая работа № 9
«Разработка базы данных средствами MS Microsoft Access»

Задание 1. Разработка базы данных

1. Выбрать тематику своей базы данных (БД).
2. Типовая структура БД:
   – две таблицы: главная и подчинённая,
   – три формы: загрузочная, главная и подчинённая,
   – три запроса: один на выборку, один на изменение и один на удаление.

   Допускается расширять структуру БД дополнительными таблицами, формами и запросами, а также реализовывать дополнительные возможности управления данными на форме в пределах индивидуального задания, согласованного с преподавателем.
3. Разработать структуру БД и построить схему данных в виде связанных списков полей таблиц. Поля должны быть не менее 4 различных типов данных. В одной из таблиц должно быть не менее двух полей числового или денежного типа.

4. Для каждого поля необходимо определить оптимальные размер и формат данных.
5. В соответствии со схемой данных создать связанные таблицы. Допускаются следующие типы связей: один-к-одному, один-ко-многим. Между главной и подчинённой таблицами должна быть связь типа один-ко-многим.
6. Создать подчинённую форму. На неё поместить все поля из подчинённой таблицы. Если подчинённых таблиц две или более, то создать столько же подчинённых форм. Поля, по которым подчинённые таблицы связаны с главной, на форму не выносить. На форме создать кнопки управления записями (не менее четырех).
7. Создать главную форму. На нее поместить все поля из главной таблицы и все подчинённые формы. Создать кнопки управления данными (не менее трех).
8. На главной форме одно из полей должно быть представлено в виде списка.
9. Создать загрузочную форму. На нее поместить следующую информацию: назначение БД, автор, краткое описание возможностей по управлению данными (согласно индивидуальному заданию), в БД. На этой форме также должна быть кнопка открытия главной формы.
10. Внести в главную таблицу не менее 5 записей. Внести в подчинённую таблицу не менее 15 записей.
11. Создать запрос на выборку данных по критериям, определяемым пользователем. Значение одного из критериев должно быть взято с одного из полей на форме.
12. Создать запрос на изменение данных с выборкой по критериям, заданным самостоятельно. Значение одного из критериев должно быть взято из списка на форме.
13. Создать запрос на удаление данных с выборкой по критериям, задаваемым самостоятельно.
14. Поместить на главную форму кнопки вызова запросов.

Задание 2. Разработка статистики по БД

В электронную таблицу (на 1-й лист) скопировать все данные из БД. На 2-м листе составить таблицу, содержащую данные о ходе изменения реального размера БД. Размер файла БД фиксировать на

каждом занятии и отмечать, какие элементы были добавлены в БД или какие изменения были внесены. На 3-м листе составить таблицу, содержащую данные об информационной емкости каждого поля из всех таблиц БД. С использованием стандартных функций табличного редактора найти 2-3 статистических показателя, логически корректных (недопустимо считать "среднюю температуру по больнице") по данным 1-го листа.

По данным 2-го и 3-го листа построить диаграммы.

Задание 3. Оформление отчета работе, который должен содержать:

- титульный лист с указанием темы работы и индивидуальной тематики;
- содержание с указанием страниц (2-й лист);
- общее задание практической работы (3-й лист);
- схему данных в текстовой или графической форме (4-й лист);
- индивидуальное задание – требования к содержанию форм и работе запросов БД в соответствии с индивидуальной тематикой (4-й – 5-й листы);
- краткие пояснения по выполнению каждого пункта общего и индивидуального задания, т.е. описание процесса построения таблиц, форм и запросов с указанием инструментов СУБД, использованных для их создания;
- внешний вид таблиц с данными;
- внешний вид форм;
- внешний вид запросов;
- статистические таблицы и диаграммы;
- краткое описание процесса выполнения задания в табличном редакторе, включая все формулы;
- список литературы, включающий источники, использованные в самостоятельной работе по созданию БД и источники информации, которая внесена в БД.

Вопросы для самоконтроля

1. Перечислите основные правила проектирования таблиц БД.
2. Расскажите, как выполняется оптимизация работы таблиц СУБД Access?
3. Назовите основные типы связей таблиц, используемых в MS Access.
4. Охарактеризуйте основные виды запросов, используемых в MS Access.

Список литературы

1. Базы данных и знаний. Проектирование баз данных в Microsoft Access [Электронный ресурс] / О.В. Чурбанова, А.Л. Чурбанов - Архангельск: ИД САФУ, 2015. - http:// www.studentlibrary.ru/ book/ ISBN9785261010296.html.
2. Искусство создания базы данных в Microsoft Office Access 2007 [Электронный ресурс] / В.В. Быкова - Красноярск : СФУ, 2011. - http://www.studentlibrary.ru/ book/ISBN9785763823554.html
3. Искусство создания базы данных в Microsoft Office Access 2007 [Электронный ресурс] / В.В. Быкова - Красноярск : СФУ, 2011. - http://www.studentlibrary.ru/book/ISBN9785763823554.html

Глава 2. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

2.1. Введение в практический курс

Правила программирования

Этот раздел посвящен рассмотрению универсальных правил, по которым пишутся программы, форматированию и правильной записи программ.

1. *Имена переменных.* Именам переменных необходимо давать *содержательные имена*, отражающие суть тех данных, для хранения которых они предназначены. Исключением могут быть только переменные, используемые в циклах, но не участвующие многократно в вычислениях. Например, переменной, используемой в программе для хранения чьего-либо имени, логично дать имя *name*.
2. *Объявление переменной.* Объявление переменной это определение ее типа и имени. Объявление переменной всегда должно предшествовать обращению к этой переменной. Если в C++ переменная не объявлена, при компиляции программы (переводе программы на язык, близкий к машинному) будет сгенерирована ошибка.
3. *Инициализация переменных.* После объявления переменной её рекомендуется инициализировать, т.е. присвоить ей какое-либо значение, единицу или ноль. Это очень актуально для переменных, используемых в вычислениях. Дело в том, при объявлении переменной для нее выделяется (резервируется) память. Резервирование памяти не очищает ячейки от значений, которые ранее в них хранились, поэтому, если за объявлением переменной не следует её инициализация, то текущее значение этой переменной будет непредсказуемым. При некоторых условиях компиляторы могут осуществлять очистку памяти, выделяемой под переменные.
4. *Стиль записи программы.* Рекомендуется придерживаться единого стиля оформления текста (интервалов, отступов, принципов записи конструкций языка, принципов именования пере-

менных и т.д.) в пределах всей программы. Основная цель – повышение читабельности и, следовательно, понятности программы.

Пример оформления фрагмента программы без использования стилей:

void main ( ) { cout<<"Hello, world"; _getch ();};

Пример оформления фрагмента программы с использованием стилей:

```
void main ( )
{
   cout <<"Hello, world";
   _getch ();
}
```

Некоторые рекомендации по использованию стилей:

- знаки (+ - = * /) пишутся через пробел;
- для зрительного разделения отдельных частей программы (больших фрагментов комментариев) используется штриховая линия типа:
  // ----------;
- скобки выравниваются вертикально по левой границе;
- включайте в программу комментарии. Они должны быть хорошо составлены, иметь правильную пунктуацию, по возможности без сокращений, и выровнены вертикально. Комментарии, в общем, воспринимаются лучше, когда помещаются в многострочных блоках, которые чередуются с блоками текста программы. Для этого комментарий должен описывать на высоком уровне, что делают несколько последующих строк кода. Не перегружайте программу комментариями – рекомендуется комментировать фрагменты программы (логически блоки кода, циклы и т.п.), а не каждый отдельный оператор. Например:

```
/*--------------------------------
Объявляем переменные:
a – номер дня недели
```

```
b–номер месяца года
c - год
-------------------------------- */
int a, b, c;

// вводим значения a, b, c
cout << "Введите a -"<<"\n";
cin>>a;
cout<< "Введитеb -"<<"\n";
cin>>b;
cout<< "Введите c -"<<"\n";
cin>>c;
```

Краткая историческая справка

Язык C++ развился из C, который, в свою очередь, был создан на основе двух предшествующих языков - BCPL и B. Язык BCPL был создан в 1967 году Мартином Ричардом как язык для написания компиляторов и программного обеспечения операционных систем. Кен Томпсон предусмотрел много возможностей в своем языке B- дубликате BCPL- и использовал B для создания более ранних версий операционной системы UNIX в BellLaboratories в 1970 году на компьютере DECPDP-7.

Язык C был развит из B Денисом Ритчи и первоначально реализован в 1972 году. Он использует многие важные концепции BCPL и B, а также добавляет типы данных и другие свойства. Первоначально C приобрел широкую известность как язык разработки операционной системы UNIX. Сегодня фактически все новые операционные системы написаны на C или на C++. Он независим от аппаратных средств. При тщательной разработке на C можно написать мобильные программы, переносимые на большинство компьютеров.

В конце 70-х годов C развился в то, что теперь относят к «традиционному C», «классическому C» или «C Кернигана и Ритчи».

Широкое распространение C на различных компьютерах (аппаратных платформах) привело ко многим вариациям языка. Некоторые из них были похожи, но несовместимы друг с другом. Это было серьезной проблемой для разработчиков программ, нуждавшихся в написании совместимых программ. Стало ясно, что необходима стандарт-

ная версия С. В 1983 году при Американском Национальном Комитете Стандартов (ANSI) в области вычислительной техники и обработки информации был создан технический комитет основной целью которого стало «обеспечить недвусмысленное и машинно-независимое определение языка». В 1989 году стандарт был утвержден. ANSI скооперировался с Международной Организацией Стандартов (ISO), чтобы стандартизировать С в мировом масштабе.Совместный стандарт был опубликован в 1990 году и назван ISO/IEC 9899:1990.

С++ – расширение С – был разработан Бьерном Страуступом в начале 80-х годов в BellLaboratories. С++ обеспечивает ряд свойств, которые «приводят в порядок» язык С, но, что более важно, он обеспечивает возможность объектно-ориентированного программирования. Это явилось революционной идеей в мире программного обеспечения. Объекты - это эффективные повторно используемые компоненты программного обеспечения, моделирующие элементы реального мира. Объектно-ориентированные программы легче понимать, корректировать и модифицировать.

С++ - гибридный язык, он предоставляет возможность программировать и в стиле С, и в объектно-ориентированном стиле, и в обоих стилях сразу. С середины 90-х годов С++ становится доминирующим системно-образующим языком.

Программа на языке *С++* представляет собой файл с расширением CPP (*.cpp). Процесс создания этого файла называют *кодированием* он, как правило, выполняется с помощью специального редактора кода (code editor).

По завершении редактирования исходный код программы необходимо перевести на машинный язык. Этот процесс называют компиляцией, он производится компилятором языка (compiler). Результат этой стадии - объектный файл с расширением OBJ (*.obj).

Завершает разработку программы фаза компоновки, в результате которой создается исполняемый файл с расширением EXE (*.exe), готовый к работе.

Фаза компиляции сопровождается проверкой синтаксиса программы, все найденные ошибки сообщаются пользователю.

Ошибки *Errors* означают нарушения синтаксиса языка С++, которые делают невозможной дальнейшую компоновку программы. Все ошибки должны быть вами устранены, для чего можно воспользо-

ваться сообщениями компилятора, в которых указывается номер строки и краткое описание ошибки.

Предупреждения *Warnings* означают подозрительные конструкции, которые могут функционировать неправильно - не так, как это задумано программистом. Наличие предупреждений не приводит к остановке процессов компиляции и компоновки. Однако, к ним также следует относиться внимательно, так как в большинстве случаев они являются следствием логических ошибок в программе, которые трудно поддаются обнаружению.

Современные системы программирования часто объединяют в себе все компоненты, необходимые для создания исполняемого файла, то есть выполняют одновременно функции редактора кода, компилятора, компоновщика, а также некоторые другие - библиотекаря (librarian), отладчика (debugger), профайлера (profiler) и т. д. Такие системы называют интегрированными средами разработки (IDE, integrated development environment). Программный пакет Visual Studio 2015 является примером такой IDE, который мы будем использовать в рамках настоящего курса.

2.2. Практическая работа № 1 «Создание консольного приложения»

1. Цель работы

Приобретение обучающимися умений и навыков в работе с оборудованием компьютерного класса, с системой программирования Microsoft Visual Studio,правилами безопасной работы. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

2. Порядок выполнения

Упражнение 1

1) Найдите на Рабочем столе своего компьютера ярлык 

, запустите программу.
2) Перед вами появиться окно программы (см. Рис.1). Первая строка экрана содержит все команды главного меню. В главном меню содержаться следующие команды: Файл, Правка (редактор), Вид, Отладка, Команда, Сервис, Тест, Анализ, Окно, Справка.

Все они имеют собственное подменю. Вызов функций подменю осуществляется перемещением курсора к нужному элементу и нажатием левой кнопки мыши.

3) Программные задачи оформляются в виде проектов. Обычно для каждой программы создается свой проект. Проект (project) представляет собой набор файлов, которые совместно используются для создания одной программы. Создайте и запустите первый проект. Для этого в главном меню выберите пункт Файл →Создать → Проект (см. Рис. 94).

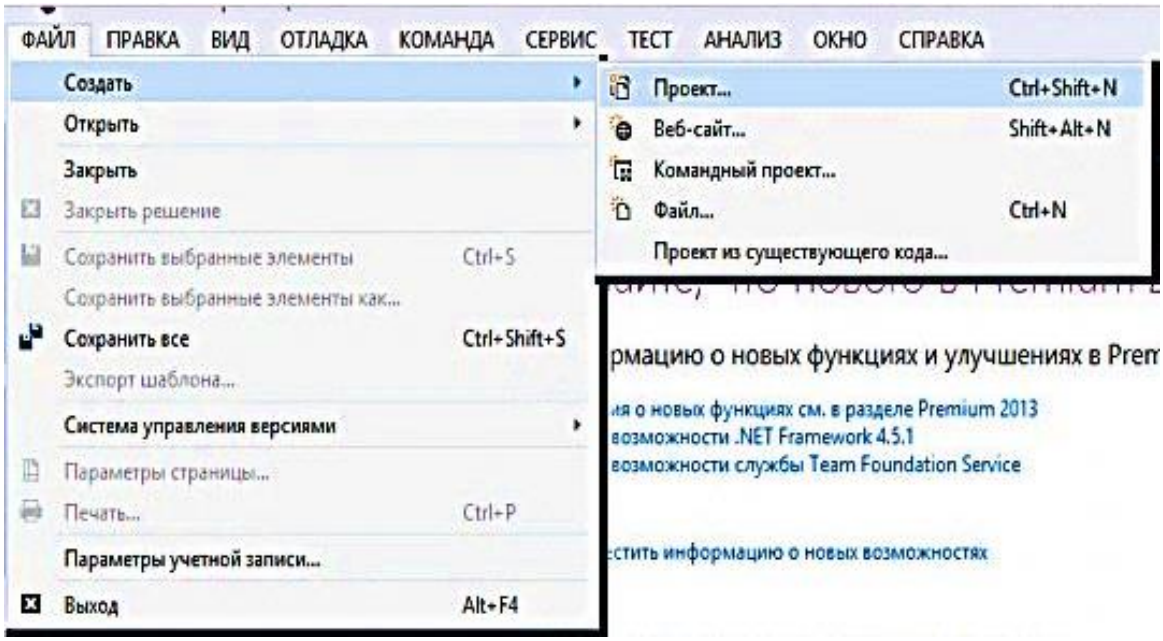

Рис. 94. Окно программы Visual Studio при создании проекта

4) Перейдите на вкладку Проект. В появившемся окне (см. рис.95) последовательно выберите Шаблон C++ → Консольное приложение Win32.
5) В строке Имя укажите имя проекта – zadanie1, в строке Расположение оставьте без изменения или укажите каталог для проекта, для чего воспользуйтесь кнопкой Обзор.
6) Нажмите кнопку ОК.

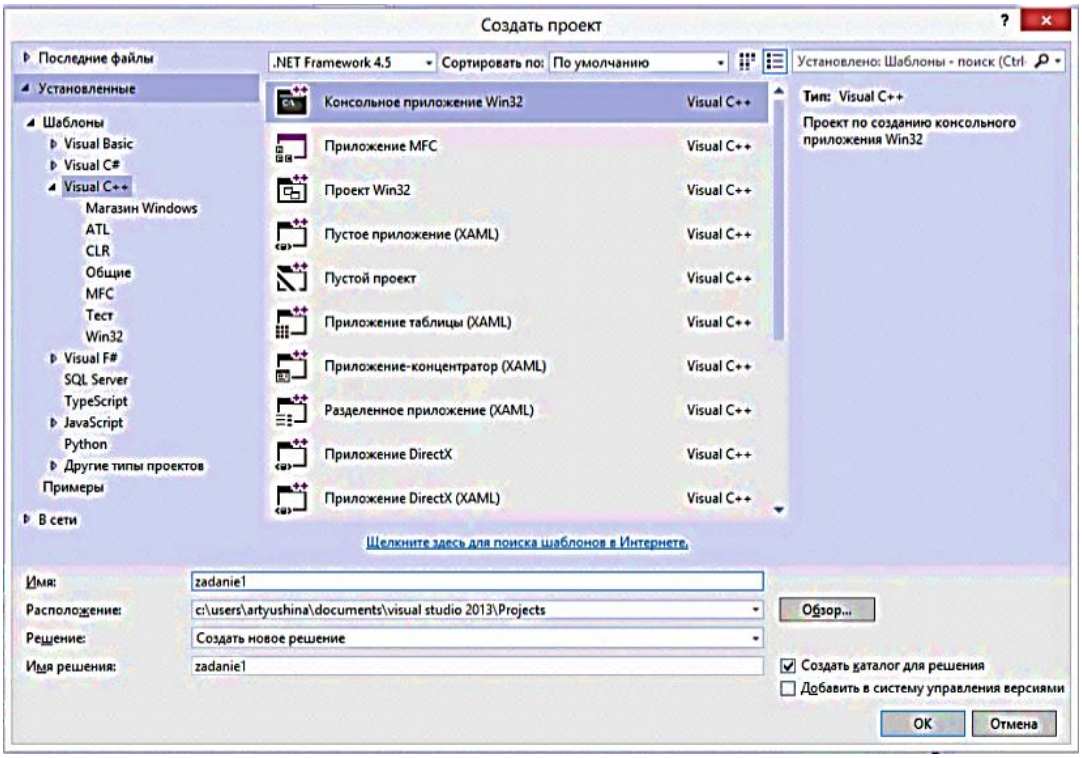

Рис. 95. Внешний вид вкладки Проект

7) Для текста программы вам понадобится файл zadanie1.cpp. Он открывается автоматически или может быть выбран в окне Обозреватель решений в правой части окна приложения (см. рис.96).

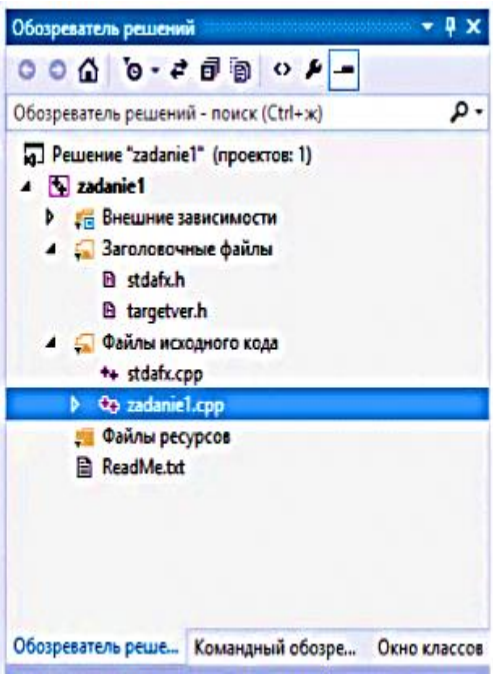

Рис. 96. Внешний вид окна Обозреватель решений

8) Перейдите в окно редактора и наберите листинг программы, приведенный ниже.

```
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
using namespace std;

int main()
{
    setlocale(LC_CTYPE, "russian");
    cout<< "Привет";
    _getch();
    return 0;
}
```

9) Сохраните текст программы. Для этого в Главном меню выберите последовательно пункты Файл→Сохранить zadanie1.cpp.
10) Откомпилируйте программу. Для этого в главном меню выберите Локальный отладчик Windows или нажмите комбинацию клавиш Alt +F9(см. Рис. 97,98).

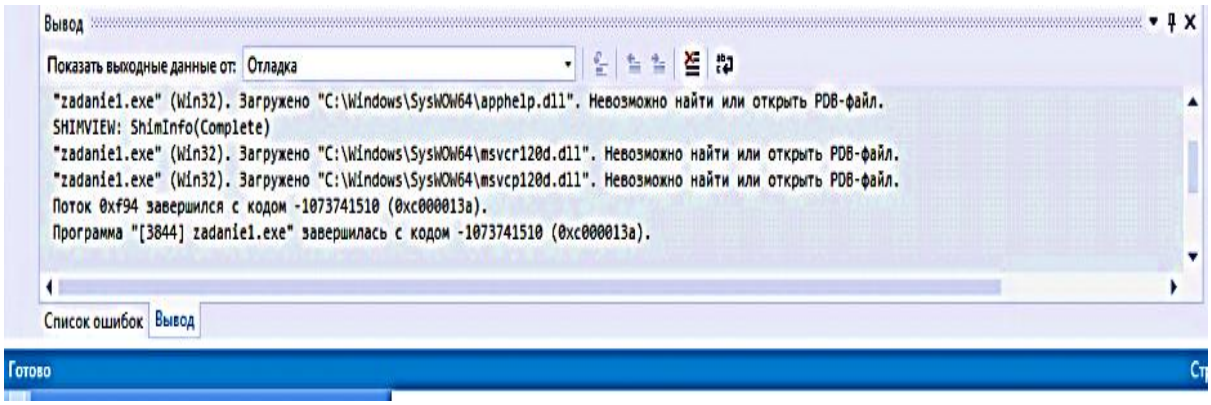

Рис. 97. Компиляция программы

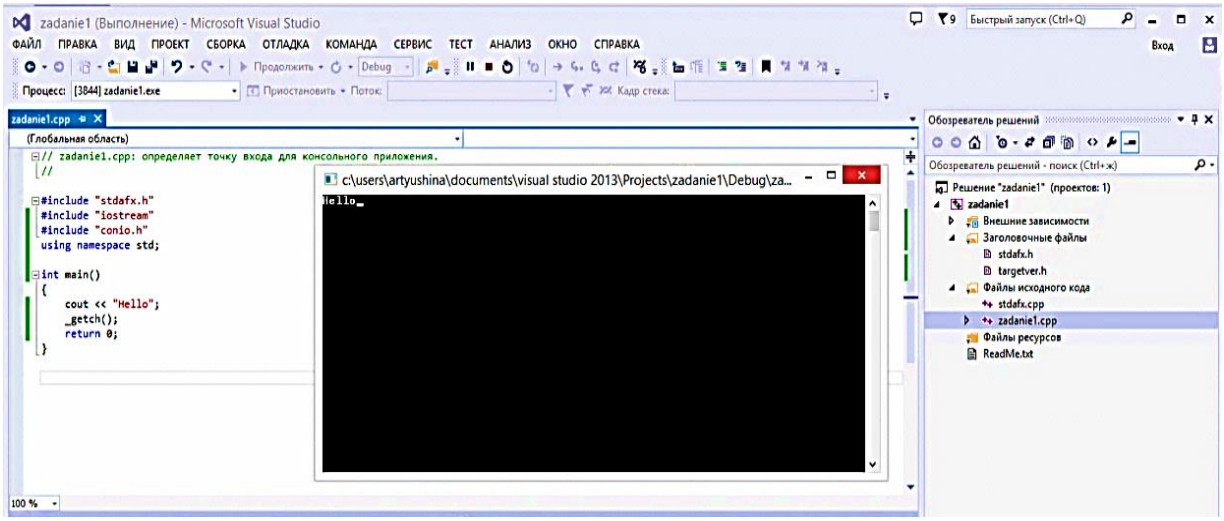

Рис. 98. Внешний вид окна запуска программы

Общая структура программы на языке C++

Рассмотрим детально набранную программу.

Программа начинается с конструкции #include "stdafx.h". Эта запись означает подключение заголовочного файла, в свою очередь содержащего в себе подключения общих файлов, образующих костяк любого проекта, созданного в среде Visual Studio.Все это сделано для ускорения компиляции проектов. Т.к. объём части общих заголовочных .h-файлов очень большой, они генерируются один раз, а затем многократно используются.

При создании приложения типа Conlsole Application в Visual Studio 2015 stdafx.h содержит следующие строки:

```
1   // stdafx.h : include file for standard system include files,
2   // or project specific include files that are used frequently, but
3   // are changed infrequently
4   //
5
6   #pragma once
7
8   #include "targetver.h"
9
10  #include <stdio.h>
11  #include <tchar.h>
12
....  // TODO: reference additional headers your program requires here
```

Аналогичные записи #include "iostream" и #include "conio.h" подключают библиотеки потокового ввода-вывода и консольного ввода-вывода соответственно (понятие консоль объединяет комплект устройств интерактивного ввода-вывода, подсоединенных непосредственно к компьютеру: монитор, клавиатуру, мышь и т.д.), директива #include <locale.h> подключает функцию setlocale (), позволяющую, в нашем случае, выводить сообщения на русском языке.

Подключение библиотек делает возможным вывод на экран текстовой строки с помощью операцииcout<<из состава библиотеки потокового ввода-вывода и использование функции _getch() из состава библиотеки консольного ввода-вывода для задержки завершения работы приложения до нажатия любой клавиши.

Программная библиотека представляет собой набор классов, функций, облегчающих работу пользователя. В таблицах 4,5 ниже приведены некоторые из часто используемых функций и манипуляторов.

Таблица 4. Некоторые функции и манипуляторы библиотеки iostream

Библиотека iostream	
Функции	Назначение
width(int x)	минимальное число знаков до следующего вывода
fill(char x)	устанавливает символ-заполнитель и возвращает предыдущий символ-заполнитель. По умолчанию в качестве символа-заполнителя используется пробел
precision(int x)	устанавливает число значащих знаков для чисел с плавающей точкой
Манипуляторы	Назначение
endl	перевод строки и вызов flush
flush	выгружает содержимое буфера в поток

Таблица 5. Некоторые функции библиотеки conio

Библиотека conio	
_getch()	Ожидание нажатия клавиши
clrscr()	Очистка экрана
gotoxy()	Перемещение курсора
textcolor()	Выбор цвета текста
kbhit()	Проверка нажатия клавиши
textmode()	Изменение режима

Например:

```
cout.width(10);
cout<<"ten"<<"four"<<"four";
```

устанавливает 10 знаков до вывода следующего слова.

Далее программа содержит строку usingnamespacestd;, обозначающую «зону видимости» тех классов с их методами и свойствами, которые будут использоваться в проекте.

Далее программа содержит заголовок функции с именем main (). Выполнение любой программы на C++ начинается с вызова функции main(). Поэтому каждая программа на языке C++ должна ее содержать.

Следующая строка

{

содержит открывающуюся фигурную скобку, обозначающую начало тела функции main (). Тело функции состоит из набора объявлений, определений и операторов. Каждое из них должно завершаться символом точки с запятой.

Строка, содержащая оператор return 0; необходима для возврата значения в вызывающую функцию, в нашем случае, в функцию main (). Ноль возвращается в вызывающую функцию, когда программа была выполнена успешно. Возможны и другие значения. Например, если произошла ошибка при открытии файла, возвращаемое значение будет равно 1.

Последняя строка программы

}

содержит закрывающуюся фигурную скобку. Она обозначает конец функции main () и конец основной части программы (в большинстве случаев – конец программы).

Сохранение, закрытие и открытие проектов

Проект сохраняется автоматически. Но можно сохранить проект, используя команды Сохранить все, Сохранить меню Файл.

Для открытия существующего проекта выберите последовательно пункты Открыть | Решение или проект из меню Файл.

2.3. Практическая работа № 2
«Работа со встроенным отладчиком Microsoft Visual C++ 2015»

1. Цель работы

Приобретение практических навыков в работе со встроенным отладчиком Visual Studio 2015. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Рекомендации по использованию встроенного отладчика Microsoft Visual C++ 2015

Отладка программ – процесс исправления ошибок в программе. Рассмотрим некоторые режимы отладки программ.

Чтобы запустить отладчик, необходимо нажать клавишу F10. В этом случае на экране появится желтая стрелка рядом с окном документа, указывающего на строку с открывающейся фигурной скобкой в main().

Если вы хотите начать трассировку не с начала, установите курсор на нужную строку.

Пошаговая трассировка. Этот режим позволяет следить за тем, как изменяются значения различных переменных. Нажимая клавишу F10, можно выполнять последовательно один оператор программы за другим. Значения переменных, выбранных компилятором, будут выводиться в окне закладки *Видимые*. Если вы хотите пропустить пошаговое выполнение некоторого куска программы, нажмите клавиши Shift+F11.

Просмотр переменных. Если требуется создать собственный набор просматриваемых переменных, внесите их в окно просмотра закладки Autos. Для этого щелкните правой кнопкой мыши на имени переменной в исходном коде. Из появившегося меню выберите Add-Watch (Добавить). Имя переменной и ее текущее значение появится в окне просмотра. Если переменная пока недоступна, окно просмотра выдаст сообщение об ошибке вместо значения переменной.

Пошаговая трассировка функций. Следует заметить, что режим трассировки по F10 рассматривает каждую встроенную функцию как одно выражение. Если вам необходимо осуществить пошаговую трассировку каждого выражения функции, нажмите F11. Если вы хотите

прервать пошаговую трассировку функции, воспользуйтесь клавишами Shift+F11.

Точки останова. Этот режим позволяет временно прерывать работу программы в указанных местах. Для того, чтобы вставить точки останова в листинги, нужно установить курсор на ту строку, в которой программа при трассировке должна остановиться. Затем нажать правую кнопку мыши и выбрать из меню Точка останова | Вставить точку останова. Напротив этой строки кода появится красный кружок. Теперь, если вы, например, запустите программу на выполнение, программа остановится на этой строке. На этом временном срезе можно проверить значения переменных, произвести пошаговую трассировку кода и т.д.

Чтобы удалить точку останова, встаньте на красный кружок, нажмите правую кнопку мыши и выберите из появившегося меню Удалить точку останова.

Задание 1. Наберите текст программы, представленный ниже. Построчно исполните код программы f1.cpp, используя встроенный отладчик Microsoft Visual C++ 2015. Проследите за тем, как изменяются значения переменных a, b, summa, зафиксировав значения в таблице.

```
    //f1.cpp
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
using namespace std;

int main()
{
        float a, b, summa;
        setlocale(LC_CTYPE, "russian");
        cout<<"введитеа,b\n";
        cin>>a>>b;
        summa = a + b;
        cout<<"\n сумма= "<<summa;
        _getch();
        return 0;
        }
```

Вопросы для самоконтроля

1. Перечислите основные правила программирования.
2. Назначение и синтаксис комментариев в программе.
3. От какого языка программирования произошел C++?
4. Какой основной фактор обусловил создание C++?
5. Структура простой программы на C++.
6. Объясните назначение функции main ().
7. Что такое пространство имён?

2.4. Практическая работа № 3 «Объявление и инициализация переменных. Стандартные типы данных»

1. Цель работы

Приобретение практических навыков в работе с основными типами переменных, в записи выражений на языке программирования C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Объявление и инициализация переменных. Стандартные типы данных

Основной задачей большинства компьютерных программ является быстрое выполнение большого количества вычислительных операций. В целях увеличения производительности промежуточные результаты расчетов хранятся в оперативной памяти. Программы C++ для размещения своих данных в оперативной памяти используют переменные. В языке C++ имена, которые используются для обозначения переменных, называются идентификаторами. Идентификатор может содержать латинские буквы, цифры и символ подчеркивания, и начинаться обязан с буквы или символа подчеркивания. В стандарте ANSI языка C++ идентификатор определяется своими первыми 32 символами. Строчные и прописные буквы рассматриваются в C++ как разные символы. Идентификатор не должен совпадать с ключевыми словами (командами, конструкциями языка).

В языке C++ все переменные должны быть объявлены до их использования. В нём определены 6 типов переменных, которые можно назвать базовыми (см. таблица 6).

Таблица 6. Стандартные типы C++

Тип	Название типа	Диапазон возможных значений
char	символьный	Символы ASCII, числа от -128 до 127
int	целый	от -32768 до 32767
float	вещественный	от 3,4 *10^{-38} до 3,4 *10^{+38}
double	вещественный двойной точности	от 1,7*10^{-308} до 1,7*10^{+308}
void	пустой, не имеющий значения	
bool	логический	true или false

Если исключить из представления целых чисел знак, то полученный тип данных будет представлять неотрицательные числа с удвоенной верхней границей диапазона представления (см. таблицу 7).

Таблица 7. Беззнаковые целые типы C++

Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Размер в байтах
unsigned char	0	255	1
unsigned int	0	65 535	2
unsignedlong	0	4 294 967 295	4

При объявлении переменная также может быть инициализирована (определено ее начальное значение) некоторой величиной из диапазона допустимых значений. Для этой цели используется оператор присваивания «=». Общая форма объявления переменной:

Тип_переменной идентификатор_переменной [=начальное значение];

В квадратных скобках указано необязательное выражение. Можно считать, что неинициализированная переменная не имеет определенного значения (точнее, ее значение непредсказуемо).

Объявление переменной может размещаться почти в любом месте программы. Однако оно всегда должно предшествовать первому

обращению к этой переменной. Одна и та же переменная может быть объявлена несколько раз в разных блоках программы. Нельзя объявить дважды одну переменную в одном блоке программы (в цикле, функции и т.д.)

Примеры объявления и инициализации переменных:

int x=10; // переменная x целого типа и начальным значением 10
float a,b,c; // неинициализированные вещественные переменные a,b,c
char s=’a’; // инициализируем символьную переменную s буквой «a»
char s [20]; // текстовая строка из 20 символов с именем s

При объявлении двух или более переменных одного типа в форме списка можно одну из них (или несколько) обеспечить начальными значениями. При этом все элементы списка разделяются запятыми.

*Пример:*int a, b = 8, c = 19, d;

В C++ определен широкий набор операций. Имеется четыре общих класса операций: арифметические, поразрядные, логические и операции отношений. В C++ определены следующие арифметические операции (см. таблицу 8).

Таблица 8. Арифметические операции

Название операции	Знак в C++	Запись на C++
сложение	+	a + 7
вычитание	-	p - c
умножение	*	b * m
деление	/	x / y
Остаток от деления (деление по модулю)	%	x % y

Кроме арифметических операций C++ дает удобные возможности использования математических функций (см. таблица 9). Большая их часть содержится в библиотеке math.h и для их пользования требуется подключение соответствующей библиотеки (директива #include <math.h>).

Таблица 9. Наиболее употребляемые математические функции

Название функции	Обозначение	Запись C++
синус	sin x	sin (x)
косинус	cos x	cos (x)
тангенс	tg x	tan (x)
квадратный корень	$\sqrt{x}$	sqrt (x)
возведение в степень	x^y	pow (x, y)
экспонента	e^x	exp (x)
натуральный логарифм	ln x	log (x)
модуль	|x|	fabs (x)
арксинус	arcsin x	asin (x)
арккосинус	arccos x	acos (x)
арктангенс	arctgx	atan (x)

Все перечисленные функции принимают в качестве аргумента вещественную переменную (или константу) и возвращают вещественный результат.

В библиотеке math.h также определены некоторые часто используемые математические константы, часть из которых приведена ниже (см. таблицу 10).

Таблица 10. Математические константы

Константа	Запись в C++
π	M_PI
E	M_E
ln2	M_LN2
ln10	M_LN10

Вопросы для самоконтроля

1. Объясните, почему переменную необходимо объявлять перед её использованием.
2. Чем объявление переменной отличается от её инициализации?
3. Требования, предъявляемые к идентификаторам в языке C++.
4. Назовите основные типы данных в C++.
5. Какое ключевое слово используется для объявления: целочисленной (вещественной, логической, символьной) переменной?

6. Какие типы целочисленных значений поддерживаются в C++?
7. Какие типы дробных значений поддерживаются в C++?
8. Почему в языке программирования для представления целых и дробных чисел существуют различные типы данных?
9. Чем характеризуется тип void?
10. Какие значения может принимать переменная типа bool?

Задания для самостоятельного выполнения

Порядок выполнения:

- составьте блок-схему решения задачи индивидуального задания;
- напишите программу на языке C++ для разработанного алгоритма решения задачи;
- выполните отладку и компиляцию программы, получите исполняемые файлы;
- выполните тестирование программы;
- оформите результаты работы в форме отчета, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования C++, тестовые данные.

1. Дана длина ребра куба. Найдите объем куба и площадь его боковой поверхности.
2. Дан радиус окружности. Найти длину окружности и площадь круга.
3. Даны катеты прямоугольного треугольника. Найдите его периметр и гипотенузу.
4. Вычислите значение функции $y = \frac{|a| + 2\sin b}{5{,}5a}$ при любых значениях a и b.
5. С начала суток прошло n секунд. Определить, сколько полных часов прошло с начала суток и сколько полных минут прошло с начала очередного часа.

2.5. Практическая работа № 4 «Функции и операторы ввода-вывода»

1. Цель работы

Приобретение практических навыков в программировании ввода-вывода в C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Организация ввода-вывода в C++

C++ дает пользователю различные возможности для программирования ввода-вывода. Эти возможности реализуются с помощью функций, входящих в состав различных библиотек. Далее мы рассмотрим два варианта – консольный и потоковый ввод-вывод.

Консольный ввод-вывод. Удобная компьютерная программа – это всегда программа, корректно взаимодействующая с пользователем. Такая программа запрашивает все необходимые для ее работы данные, используя операции ввода, и выдает результат с помощью операций вывода. Стандартным устройством ввода в ПК считается клавиатура, устройством вывода – экран монитора. Совокупность клавиатуры и монитора называется консолью.

Консольный ввод-вывод организуется с помощью функций библиотек stdio.h и conio.h, что предполагает наличие директив #include "stdio.h" и #include "conio.h" в заголовочной части программы.

Функции printf () и scanf_s () осуществляют форматированный вывод и ввод на консоль. Форматированный ввод и вывод означает, что функции могут читать и выводить данные в разном формате, которым вы можете управлять. Управляющая строка содержит два типа информации: символы, которые непосредственно выводятся на экран, и команды формата, определяющие, как выводить аргументы. Команды формата начинаются с символа % за которым следует код формата. Коды формата для стандартных типов данных указаны в табл. 11:

Таблица 11. Коды форматов стандартных типов данных языка C++

Переменная	Команда формата
Целое десятичное число со знаком	%d
Вещественное число	%f
Вещественное число двойной точности	%lf
Текстовый символ	%c
Целое число без знака	%u

Printf () – функция вывода информации на консоль. С ее помощью в окне приложения можно вывести как строку простого текста, так и значения переменных различных типов.

Общая форма записи функции:

printf (“форматная строка“[, перем1], [перемен2] […]);

Здесь в круглых скобках указаны обязательные параметры, а в прямоугольных скобках – параметры, которые указываются по необходимости.

Например, запись printf (“Hello!“)означает вывод на экран простой текстовой строки «Hello!».

После выполнения приведенного ниже кода программы на экран будет выведено значение вещественной переменной summa с точностью до двух знаков после запятой.

```
//primer1.cpp
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include "stdio.h"
#include <locale.h>
using namespace std;

int main()
{
	float a, b, summa=0;
	setlocale (LC_CTYPE, "russian");
	printf("введитеа,b\n");
	scanf_s ("%f%f",&a, &b);
	summa = a + b;
	printf ("\n сумма= %.2f", summa);
```

```
    _getch();
    return 0;
}
```

Функция printf () также дает возможности управления выводом с помощью эскейп-последовательностей, начинающихся с символа ESC (обратный слэш «\»). Некоторые из них приведены в таблице 12.

Таблица 12. Некоторые символы управления выводом

Управляющий символ	Название	Действие
\n	lf (line feed)	перевод строки
\a	bel (audible bell)	звуковой сигнал
\b	bs (backspase)	возврат на шаг (забой)
\t	ht (horizontal tab)	Табуляция
\v	vt (vertical tab)	вертикальная табуляция

Scanf_s() – функция ввода с консоли. Общая форма записи этой функции: scanf (“форматная строка”, &перем1 [, &перем2] [, …]);
Аргументы функции scanf_s аналогичны соответствующим аргументам функции prinf, за исключением того, что в качестве параметров scanf_s принимает не имена переменных, а их адреса. В силу этого перед именем каждой переменной в scanf_s должен стоять знак операции взятия адреса & (амперсанд). Например, команда вводит данные в переменные a, b, c через их адреса в целочисленном формате: *scanf_s("%d %d %d", &a, &b, &c);*

Потоковый ввод-вывод организуется с помощью библиотеки iostream.h, что предполагает наличие директивы #include "iostream" в заголовочной части программы. В библиотеке определены два потоковых объекта с именами cin и cout, которые связаны с клавиатурой и экраном компьютера соответственно. Для них определены следующие операции:

- извлечение из потока, т.е. ввод с клавиатуры cin;
- размещение в потоке, т.е. вывод на экран cout.

Общая форма записи этих операторов: cin>>переменная; cout<<текстовая строка или переменная.
Например:
// объявляется переменная a целого типа, значение переменной a*

*вводится с клавиатуры *//*
int a;
cout<<”Enter a”;
cin>>a;

Потоковый ввод-вывод также имеет возможности управления выводом с помощью методов и манипуляторов. Некоторые из них приведены в таблице 13.

Таблица 13. Некоторые методы управления выводом

Методы /Манипуляторы	Описание
endl	Помещение в выходной поток символа конца строки '\n'
dec	Установка основания 10-ой системы счисления
oct	Установка основания 8-ой системы счисления
hex	Установка основания 16-ой системы счисления
setbase	Вывод базовой системы счисления
setw (int w)	Устанавливает ширину поля вывода, равную w
fill ('символ')	Заполняет пустые знакоместа значением символа
precision (точность)	Устанавливает количество значащих цифр в числе (или после запятой) в зависимости от использования fixed
fixed	Показывает, что установленная точность относится к количеству знаков после запятой
showpos	Показывает знак + для положительных чисел
scientific	Выводит число в экспоненциальной форме
setiosflags(long f)	Устанавливает флаги, указанные в f
resetiosflags (long f)	Сбрасывает флаги, указанные в f

Для использования манипуляторов с параметрами в программу необходимо включить заголовочный файл iomanip.h. Все манипуляторы объявлены в классе ios, с которым вы познакомитесь позднее.

Ниже представлен пример программы, использующей манипуляторы для изменения формата вывода, а именно, вывода 2 чисел с плавающей точкой в поле шириной в 10 символов с двумя и четырьмя знаками после запятой.

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include "iomanip"
using namespace std;

int main()
{
	cout << setiosflags (ios::fixed);
	cout << setw (10) << setprecision (2) << 1000.243 << endl;
	cout << setw (10)<< setprecision (4) << 1000.243 << endl;
	_getch();
return 0;
}
```

Вопросы для самоконтроля:

1. Чем различаются консольный и потоковый ввод (вывод)?
2. Перечислите часто используемые функции библиотек iomanip, conio и stdio.
3. Какие операторы в C++ используется для ввода (вывода) данных?
4. Какие функции используются в C++ для ввода (вывода) данных?
5. Перечислите основные символы управления выводом.

Задания для самостоятельного выполнения:

1. Организовать средствами консольного (потокового) ввода-вывода вывод ряда из девяти чисел, записанных в шестнадцатеричной системе счисления, в десятичном формате в поле шириной в 4 символа с выравниванием вправо.
2. Используя управляющие последовательности, произведите следующий вывод на экран средствами консольного (потокового) ввода-вывода:

Области видимости и класс памяти переменных

	Локальная	Статическая локальная
Область видимости	функция	функция
Время жизни	функция	программа
Начальное значение	случайное	0
Область видимости	стек	динамическая

3. Используя управляющие последовательности, произведите следующий вывод на экран средствами консольного (потокового) ввода-вывода:

Стандартные типы C++

Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Точность
bool	false	true	нет
char	-128	127	нет
short	-32 768	32 767	нет
int	-2 147 483 648	2 147 483 647	нет
long	-2 147 483 648	2 147 483 647	нет

4. Используя управляющие последовательности, произведите следующий вывод на экран средствами консольного (потокового) ввода-вывода:

Беззнаковые типы данных C++

Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Размер в байтах
unsigned char	0	255	1
unsigned short	0	65 535	2
unsigned int	0	4 294 967 295	4
unsigned long	0	4 294 967 295	4

5. Используя управляющие последовательности, произведите следующий вывод на экран средствами консольного (потокового) ввода-вывода:

Приоритеты операций в C++

Тип операции	Операции	Приоритет
Унарные	!, ++, --, +, -	высший
Арифметические	*, /, %, +, -	
Отношения	<, >, <=, >=, ==, !=	
Логические	&&, ||	
Условная	?:	
Присваивания	=, +=, -=, *=, /=, %=	низший

2.6. Практическая работа № 5 «Сокращенные варианты записи»

1. Цель работы

Приобретение практических навыков в написании сокращённых вариантов записи на языке C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Сокращенные варианты записи

Инкремент, декремент. Оператор инкремента выполняет сложение операнда с числом 1. Оператор декремента вычитает 1 из своего операнда. Инструкция x=x+1 аналогична инструкции ++x. Инструкция x=x-1 аналогична такой инструкции - -x. Операторы инкремента и декремента могут стоять как перед своим операндом (префиксная форма), так и после него (постфиксная форма).

Если оператор применен в префиксной форме, то C++ сначала выполнит эту операцию, чтобы операнд получил новое значение, которое затем будет использовано остальной частью выражения.

Если оператор применен в постфиксной форме, то C++ использует в выражении его старое значение, а затем выполнит операцию, в результате которой операнд обретет новое значение. Некоторые сокращенные варианты записи представлены в таблице 14.

Таблица 14. Сокращенные варианты записи

Стандартная запись	Описание	Сокращенная запись
A=A+B	Увеличить	A+=B;
A=A-B	Уменьшить значение A на величину B	A-=B;
A=A*C	Увеличить A в C раз	A*=C;
A=A/D	Уменьшить A в D раз	A/=D

Порядок выполнения арифметических операций. Операторы одного уровня старшинства вычисляются компилятором слева направо. Для изменения порядка вычислений необходимо использовать скобки. В таблице 15 указан порядок выполнения арифметических операций.

Таблица 15. Приоритет арифметических операций

Приоритет	Операторы
Наивысший	++ - -
	* / %
Низший	+ -

Вопросы для самоконтроля

1. Перечислите основные средства языка C++ для сокращения размера кода. Приведите примеры.
2. В чём разница между записью операции инкремента в постфиксной и префиксной форме?
3. Есть ли разница между записями count+=1 и ++count?
4. Есть ли разница между записями count-=1 и --count?

Задания для самостоятельного выполнения

Указание: для выполнения заданий постройте трассировочную таблицу изменения значений переменных.

1.Чему будет равно значение переменных a,b после выполнения фрагмента программы?

```
int a, b;
a = 2; b = 1;
a++;
```

```
a = 2 + a--;
++a;
a *= 2;
b = a++;
b = ++a;
```

2.Вычислить значения, которые примут переменные после выполнения кода:

```
int a, b = 5, c = 7, d = 9;
a = b++ +c++ + ++d;
```

3.Чему будет равно значение переменой *к* после выполнения следующего кода?

```
int k = 1;
к+ = k++;
z = ++k;
z- = k+ --k;
```

4.Чему будет равно значение каждой переменой после выполнения следующего кода?

```
int i = 1, r;
i = ++i + i++ -3 + 8 - ++i;
r = sqrt (++i / 2);
```

5. Чему будет равно значение переменой *а* после выполнения следующего кода?

```
int variable = 1;
int a = 3;
-a--;
a += ++variable  +  1  +  ++variable * 2;
```

2.7. Практическая работа № 6 «Оператор условия IF-ELSE»

1. Цель работы

Приобретение практических навыков в работе с основными алгоритмическими конструкциями языка C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Ветвления

В C++ существует несколько типов ветвлений. Рассмотрим каждый из них. Условный оператор *if* служит для выбора направления работы программы в зависимости от условий, сложившейся в данной точке программы на момент ее выполнения.
Общая форма записи условного оператора:

```
if (условие)
{
	блок операторов 1;
}
else
{
	блок операторов 2;
}
```

Если на момент выполнения условие истинно, программа передает управление блоку операторов 1 и далее первому оператору за пределами конструкции if-else. При этом блок операторов 2 не выполняется. Если на момент выполнения условие ложно, выполняется блок операторов 2, а блок операторов 1 не выполняется. В таблице 16 указаны простейшие операции отношения.

Таблица 16. Операции отношения в C++

Операция	Запись на C++
больше	>
меньше	<
больше либо равно	>=
меньше либо равно	<=
равно	= =
не равно	!=

Вложенные операторы условия

Операторы условия могут быть вложенными друг в друга, в соответствии с тем программным алгоритмом, который они реализуют. Допускается произвольная степень их вложенности. Например:

```
if (a<=b)                          // начало внешнего оператора условия
{
      if (x!=0) cout<<”x!=0”       // начало вложенного оператора условия
else                               // начало ветви else, относящейся
                                   // к вложенному оператору условия
      {
      x=1;
y=0;
      }                            // конец ветви else, относящейся
                                   // к вложенному оператору условия
}
else                               // начало ветви else, относящейся
                                   // к внешнему оператору условия
{
      a=b;
      cout<<a;
}                                  // конец ветви else, относящейся
                                   // к внешнему оператору условия
```

Сокращенные варианты записи

При программировании обыденной является ситуация, когда требуется некоторое действие в ответ на сложившиеся условия. Например, если получены неверные исходные данные от пользователя, то выдать сообщение об ошибке и выйти из программы. В таких случаях используется сокращенная запись оператора условия с отсутствующим блоком else. Общая форма записи:

```
if (условие)
{
      блок операторов;
}
```

Здесь в случае истинности условия управление передается блоку операторов в фигурных скобках. В случае ложности условия этот блок пропускается.

Составные логические выражения

В программировании распространены двойные условия, которые в математике записываются в виде f<b<c. В программе такие условия должны быть переформулированы с использованием простых операций сравнения и логически операций «И», «ИЛИ», «НЕ». Обозначение логических операций приведено в табл. 17.

Таблица 17. Логические операции и их обозначения

Логическая операция	Знак C++	Наименование знака
И	&&	двойной амперсанд
ИЛИ	||	двойная вертикальная черта
НЕ	~ (!)	не

Например:

```
if ((a>b) && (a>c))          // еслиа больше bи а больше с
        cout<<"a="<<a;       // вывести значение переменной a
else                         // иначе
{
        if ((b>a) && (b>c))  //если b больше a и b больше c
                cout<<"b="<<b; // вывести значение переменной b
else
                cout<<"c="<<c;  // иначе вывести значение
}                            // переменной c
```

Оператор Switch используется в том случае, если в программе присутствует большое дерево ветвлений и все ветвления зависят от значения какой-либо одной переменной.

Общий формат записи:

```
switch (n)
{
case 1:
оператор 1;
break;
```

```
case 2:
    оператор 2;
break;
case 3:
    оператор 3;
break;
…….
}
```

где n – целочисленная или символьная переменная;
1, 2, 3 – целочисленная или символьная константа;
Оператор 1 – тело первого case;
Оператор 2 – тело второго case;
Оператор 3 – тело третьего case;
break – (прервать) оператор завершает выполнение ветвления switch, если значение переменной в операторе switch совпадает с одним из значений констант, указанных внутри ветвления. Если значение переменной в операторе switch не совпадет ни с одним из значений констант, то управление будет передано в конец switch без выполнения каких-либо действий. В случае отсутствия оператора break управление будет передано операторам, относящимся к другим веткам switch.

Условная операция

Существует распространенная в программировании операция: переменной необходимо присвоить одно значение в случае выполнения некоторого условия и другое значение в случае невыполнения этого условия. С помощью конструкции if … else это будет выглядеть следующим образом:

```
if (alfa < beta)
        min = alfa;
else
        min = beta;
```

Подобные действия на практике настолько распространены, что была специально разработана условная операция, выполняющая эти действия. Эта операция записывается с помощью двух знаков и использует три операнда.

С помощью условной операции можно записать предыдущий фрагмент следующим образом: *min = (alfa < beta) ?alfa : beta;*

Правая часть оператора *(alfa < beta) ?alfa : beta*представляет собой условное выражение. Знак ? и двоеточие : обозначают условную операцию. Условие стоит перед знаком вопроса *(alfa < beta)*и является условием проверки. Это условие вместе с операндами *alfa и beta* составляют тройку операндов условной операции.

Если значение проверяемого условия истинно, то условное выражение становиться равным значению *alfa*, в противном случае *beta*. Скобки в чтобы визуально упростить читаемость этого оператора.

Вопросы для самоконтроля

1. Напишите условный оператор, который бы присваивал переменной temp значение, равное 1, если значение переменной count больше 50, и 0 в противном случае.
2. Каким образом получается конструкция else … if из вложенных циклов if … else? Приведите примеры.
3. Когда лучше использовать последовательность if … else, а когда – switch ()?
4. Напишите условную операцию, присваивающую переменной result значение наибольшего из двух чисел.
5. Какого типа должно быть выражение, управляющее инструкцией switch?
6. Что происходит в случае, если результат вычислений switch-выражения совпадает с case-константой?
7. Что происходит, если case-последовательность не завершается инструкцией break?
8. Назначение ключевого слова default в конструкции switch (). Приведите примеры.

Задания для самостоятельной работы

Порядок выполнения:

1. составьте блок-схему решения задачи индивидуального задания;
2. напишите программу на языке C++ для разработанного алгоритма решения задачи;

3. выполните отладку и компиляцию программы, получите исполняемые файлы;
4. выполните тестирование программы;
5. оформите результаты работы в форме отчета, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования С++, тестовые данные.

Задание 1

Вычислите значение функции, заданной формулами:

1. $y=\begin{cases} x-2, & x>2.5 \\ 1+x^2, & 0\le x\le 2.5 \\ xln|cos(x)|, & x<0 \end{cases}$
2. $y=\begin{cases} \sin(2.3x-1), & x>2.5 \\ 1-3ln|1-x|, & 0\le x\le 2.5 \\ \frac{x^2}{2-x}, & x<0 \end{cases}$
3. $y=\begin{cases} \sqrt{(tg(x^2-1))}, & x>1 \\ -2x, & 0\le x\le 1 \\ e^{cos(x)}, & x<0 \end{cases}$
4. $y=\begin{cases} \mathrm{x}^2-3+2.5\mathrm{x}^2, & x>12.5 \\ e^x+5+\cos(0.001x), & 0\le x\le 12.5 \\ x^2, & x<0 \end{cases}$

Задание 2

1.Арифметические действия над числами пронумерованы следующим образом: 1 – сложение, 2 – вычитание, 3 – умножение, 4 – деление. Дан номер действия и два числа A и B (B не равно нулю). Выполнить над числами указанное действие и вывести результат.

2.Единицы длины пронумерованы следующим образом: 1 – дециметр, 2 –километр, 3 – метр, 4 – миллиметр, 5 – сантиметр. Дан номер единицы длины и длина отрезка L в этих единицах (вещественное число). Вывести длину данного отрезка в метрах.

3. Составить программу, которая по возрасту человека (вводится с клавиатуры как целое число) определяет его принадлежность к возрастной группе: от 0 до 13 – мальчик; от 14 до 20 – юноша; от 21 до 70 – мужчина;
более 70 – старец.

4. Локатор ориентирован на одну из сторон света («С» – север, «З» – запад, «Ю» – юг, «В» – восток) и может принимать одну из трех цифровых команд: -1 – поворот налево, 1 – поворот направо, 2 – поворот

на 180 градусов. Дан символ C – исходная ориентация локатора и число N – посланная ему команда. Вывести ориентацию локатора после выполнения команды.
5. Даны два целых числа: D (день) и M (месяц), определяющие правильную дату не високосного года. Вывести значения D и M для даты, следующей за указанной (например, дано D=1 M=1, надо вывести D=2 M=1; дано D=31 M=12 надо вывести D=1 M=1; дано D=28 M=2 надо вывести D=1 M=3).

2.8. Практическая работа № 7 «Циклы»

1. Цель работы

Приобретение практических навыков в работе с основными алгоритмическими конструкциями языка C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Циклы

Действие циклов заключается в последовательном повторении определенной части программы некоторое количество раз. Повторение продолжается до тех пор, пока выполняется соответствующее условие. Когда значение выражения, задающего условие, становиться ложным, выполнение цикла прекращается, а управление передается оператору, следующему непосредственно за циклом.

В C++ существует три типа циклов: for, while, do.

Цикл for

Цикл forорганизует выполнение фрагмента программы фиксированное число раз. Как правило (но не всегда), этот тип цикла используется тогда, когда число раз, за которое должно повториться исполнение кода, известно заранее.

Синтаксис:

for (<инициализация>; <условие продолжения>;<изменение счетчика>)
{
тело цикла;
}

Цикл forначинается с выполнения блока <инициализация>, где определяется начальное значение счетчика цикла. Далее выполняются операторы (оператор), образующие тело цикла. Затем проверяется <условие продолжения>, в случае, если это условие истинно, управление передается заголовку for и значение счетчика цикла автоматически изменяется в зависимости от параметра «изменение счетчика».

Рассмотрим следующий пример: напишите программу, выводящую на экран все целые числа от 0 до 99 включительно.

Можно предложить следующий алгоритм решения задачи:

1. Объявляем целочисленную переменную k и инициализируем ее значением 0.
2. Выводим на экран k.
3. Увеличиваем k на единицу.
4. Если k<100, возвращаемся к пункту 2.
5. Завершаем программу.

Переменная kназывается счетчиком цикла, а повторяющиеся в цикле операторы – телом цикла.

Для решения данной задачи воспользуемся циклом for.

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
using namespace std;
void main()
{
	for (int k = 0; k<100; k++)	// цикл по k  с шагом 1
		cout<< " " << k;		// задаем тело цикла

_getch();
	}
```

В качестве параметра цикла необязательно использовать целочисленный счетчик. Параметром могут выступать символы. Например, следующая программа выводит на экран буквы английского алфавита:

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
```

```
#include "conio.h"
using namespace std;
void main()
{
        char ch;
        for (ch = 'A'; ch<= 'Q'; ch++)
                cout<< " " <<ch;

_getch ();
}
```

Сокращенные варианты записи

Отдельные (или все) блоки в заголовке цикла formогут быть пустыми, однако разделительные точки с запятой являются обязательными. Например, следующий фрагмент программы, где в записи цикла for оператор инициализации пуст, так как основывается на значении переменной, которая была ранее объявлена и проинициализирована:

```
int i = 0;
int j;
int val1 = 0;
int val2;
i = 25;
j = i * 2;
for ( ; i < 100;  i++)
  val1 = i;
```

Вложенные операторы цикла

Аналогично условному оператору циклы могут быть вложены друг в друга, причем степень вложенности также может быть произвольной. В качестве счетчиков такие циклы, как правило, используют различные переменные.

Дополнительные средства управления циклами

К числу дополнительных средств управления циклами относятся операторы break (от англ. – прервать) и continue (от англ. – продолжить).

С помощью оператора break организуется досрочное окончание цикла с передачей управления оператору, следующему непосредственно за концом цикла. Пусть, например, нам предстоит суммирование элементов массива *a* до тех пор, пока не встретится первое отрицательное значение:

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;
void main()
{
	int s=0, j, a[5];
	for (j = 0; j<5; j++)
	{
		cin>> a[j];
		if (a[j]<0) break;
		s += a[j];
	}
	cout<< s;
_getch();
}
```

Однако если оператор break употреблен во внутреннем цикле, то с его помощью нельзя выйти за пределы внешнего даже в том случае, когда кажется, что тело внешнего цикла кончается там же, где и тело внутреннего. На самом деле, в конце каждого цикла незримо присутствуют системные вставки, обеспечивающие нормальный выход из цикла. В частности, такие вставки возвращают память, выделенную под переменные, объявленные в заголовке цикла. Кроме того, здесь же находятся команды, возвращающие управление в начало цикла при необходимости повторения итераций.

Цикл While

Этот тип цикла используется в том случае, если количество повторений заранее неизвестно.

Синтаксис:

while (условие выполнения)
{
тело цикла;
}

Предварительно проверяется условие выполнения, пока оно истинно, исполнение тела цикла продолжается. Как только оно становиться ложным, происходит выход из цикла. Если с самого начала условие выполнения ложно, то тело цикла не выполняется ни разу.

Например, пользователю предлагают ввести серию значений. В том случае, когда вводимое значение оказывается равным 0, происходит выход из цикла:

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;
void main()
{
	setlocale (LC_CTYPE, "russian");
	int n = 99;
	while (n != 0)
	{
		cout<< "введите n\n";
		cin>> n;
	}
_getch();
}
```

Цикл do

Синтаксис: *do*
{
тело цикла;
}
while (условие выполнения);

Оператор действует следующим образом: выполняются операторы циклической части, проверяется условие выполнения, если оно истинно, выполняется тело цикла. Если же оно ложно, то цикл заканчивается.

Например, следующий пример предлагает ввести два числа: делимое и делитель, а затем производит целочисленное деление с использованием операции /. После того, как произведено вычисление, программа спрашивает пользователя, хочет ли он произвести вычисления еще раз. Если в ответ программа получает символ “y”, то вводятся значения делимого и делителя. Если символ “n”, - происходит выход из цикла:

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;
int main()
{
	setlocale(LC_CTYPE, "russian");
	float result, a, b;
	char ch;
	do
	{
		cout << "введите делимое\n";
		cin >> a;
		cout << "введите делитель\n";
		cin >> b;
		cout << "частное = " << a / b << "\n";
		cout << "еще раз ? (y / n)\n";
		cin >> ch;
	} while (ch != 'n');
	_getch();
	return 0;
}
```

Вопросы для самоконтроля

1. В каких случаях используется цикл с параметром? Как он оформляется? Как он работает (что происходит при его выполнении)?
2. Нарисуйте графическую схему выполнения цикла с параметром.
3. В каких случаях используется цикл с пред-условием? Как он оформляется? Как он работает (что происходит при его выполнении)?
4. Нарисуйте графическую схему выполнения цикла с пред-условием.
5. В каких случаях используется цикл с пост-условием? Как он оформляется? Как он работает (что происходит при его выполнении)?
6. Нарисуйте графическую схему выполнения цикла с пост-условием.
7. Как должен быть оформлен оператор цикла, чтобы тело цикла выполнялось при уменьшающихся значениях параметра цикла. Как он работает (что происходит при его выполнении)?
8. Нарисуйте графическую схему выполнения цикла при уменьшающихся значениях параметра цикла.

Задания для самостоятельной работы

Порядок выполнения

1. составьте блок-схему решения задачи индивидуального задания;
2. напишите программу на языке C++ для разработанного алгоритма решения задачи;
3. выполните отладку и компиляцию программы, получите исполняемые файлы;
4. выполните тестирование программы;
5. оформите результаты работы в форме отчета, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования C++, тестовые данные.

Задание 1: Вывести на экран в виде таблицы значения функции F на интервале от $x_{нач}$ до $x_{кон}$ с шагом dx, где *a, b, c* – действительные числа. Значения *a, b, c,* $x_{нач}$, $x_{кон}$, dx ввести с клавиатуры. Решить задачи с помощью цикла с параметром.

1. $$F = \begin{cases} \frac{1}{ax} - b \text{ при } x + 5 < 0 \text{ и } c = 0 \\ \frac{x-a}{x} \text{ при x} + 5 > 0 \; b \; c \neq 0 \\ \frac{10x}{c-4} \text{ в остальных случаях} \end{cases}$$

2. $$F = \begin{cases} -ax^3 - b \text{ при } x + c < 0 \text{ и } a \neq 0 \\ \frac{x-a}{x-c} \text{ при } x + c > 0 \text{ и a} = 0 \\ \frac{x}{c} + \frac{c}{x} \text{ в остальных случаях} \end{cases}$$

3. $$F = \begin{cases} ax^2 - bx + c \text{ при x} < 3 \text{ и } b \neq 0 \\ x - \frac{a}{x-c} \text{при a} > 0 \text{ и } x = 0 \\ 1 + \frac{x}{c} \end{cases}$$

Задание 2

Указание: решить задачи задания 2 с использованием цикла с пред- или пост-условием.

1. Имеется серия измерений элементов треугольника. Группы элементов пронумерованы. В серии в произвольном порядке могут встречаться следующие элементы треугольника:

- основание и высота;
- две стороны и угол между ними (задан в радианах);
- три стороны.

Разработать программу, которая запрашивает номер группы элементов, вводит соответствующие элементы и вычисляет площадь треугольника. После вычисления результата программа должна отобразить результат и запросить пользователя о его желании произвести ещё одно вычисление.

2.Дано натуральное число N. Разработать программу, вычисляющую произведение первых N сомножителей.

3. Дано натуральное число N. Разработать программу, вычисляющую:

$$S = \frac{1}{sin1} + \frac{1}{sin1 + sin2} + \cdots + \frac{1}{sin1 + sin2 + \cdots + sinN}$$

4. Даны два натуральных числа m и n. Разработать программу, проверяющую, если в записи чисел одинаковые цифры.

5. Для любого натурального числа N разработать программу, вычисляющую произведение его первой и последней цифр.

2.9. Практическая работа № 8 «Структуры»

1. Цель работы

Приобретение практических навыков в работе с простыми структурами языка C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Структуры

Структура является объединением простых переменных. Эти переменные могут иметь различные типы: float, char, int. Переменные, входящие в состав структуры, называются *полями* структуры. Структуры являются одной из составляющих главных концепций языка – объектов и классов и используются, как правило, в качестве объединения данных.

Определение структуры

```
struct  имя структуры
{
  члены структуры;
};
```

Простая структура

Начнем рассмотрение со структуры, содержащей три поля, два из которых имеют целый тип и одно поле – вещественный тип. Эта структура предназначена для хранения информации о комплектующих деталях изделий, выпускаемых фабрикой. Компания производит несколько типов изделий, поэтому номер модели изделия включен в структуру как первое из ее полей. Номер самой детали представлен вторым полем, а ее стоимость – третьим полем.

```
//parts.cpp
#include <iostream.h>
#include <conio.h>
```

```
struct part
{
intmodelnumber;
intpartnumber;
floatcost;
};
```

Определение структуры part необходимо для того, чтобы создавать на его основе переменные типа part.

Определение структурной переменной

```
intmain ()
{
partpart1;
…
}
```

Первый оператор функции main () выглядит следующим образом: partpart1;

Он представляет собой определение переменной part1, имеющий тип part. Определение переменной означает, что под эту переменную выделяется память. Правило: под структурную переменную всегда отводится столько памяти, сколько достаточно для хранения всех ее полей.

Доступ к полям структуры

Когда структурная переменная определена, доступ к ее полям возможен с применением *операции точки (доступ к полю структуры)*. В выражении на первом месте ставиться имя структурной переменной, затем – операции точки, на третьем месте – имя поля.

```
part1. modelnumber = 6244;
part1. partnumber = 373;
part1. cost = 217.55;
```

С полями структурной переменной можно обращаться так же, как с обычными простыми переменными. В примере с помощью оператора поля структуры инициализируются и выводятся на экран.

```
cout<< "модель " << part1. modelnumber<< " ";
cout<<"деталь "<< part1. partnumber<< " ";
cout<<"стоимость " << part1. cost<< " ";
```

Структуры обладают достаточно широким набором возможностей. Рассмотрим некоторые из них.

Следующий пример демонстрирует способ, при помощи которого можно инициализировать поля предварительно определенной структурной переменной. В программе используются две структурные переменные part1, part2.

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;
struct part
	{
		int modelnumber;
		int partnumber;
		float cost;
	};

int main()
{
	setlocale (LC_CTYPE, "russian");
/*инициализация переменной структурной переменной в момент её объявления, первая из величин присваивается первому полю, вторая – второму и т.д. через перечисление */
	part part1 = { 6244, 373, 217.55 };

//объявление второй структурной переменной
		part part2;

//вывод полей первой переменной
	cout<< "model " << part1.modelnumber;
	cout<< "detal " << part1.partnumber;
	cout<< "cost " << part1.cost<<"\n";
//присваивание значений одной структурной переменной другой
	part2 = part1;
```

```
//вывод полей второй переменной
        cout<< "model " << part2.modelnumber;
        cout<< "detal " << part2.partnumber;
        cout<< "cost " << part2.cost<< "\n";
_getch();
return 0;
}
```

Другой, более длинный пример, использования структур в качестве элементов массива, в котором также присутствуют строковые переменные и функция, выводящая сообщение об ошибке в случае ввода некорректных оценок.

Задача: на вход подаются сведения о сдаче экзаменов учениками 9-х классов некоторой средней школы. В первой строке сообщается количество учеников N, которое не меньше 10, но не превосходит 100, каждая из следующих N строк имеет следующий формат:

<Фамилия> <Имя> <оценки>, где <Фамилия> - строка, состоящая не более чем из 20 символов, <Имя> - строка, состоящая не более чем из 15 символов, <оценки> - три целых числа, соответствующие оценкам по пятибальной системе. Требуется написать программу, которая будет выводит на экран фамилии и имена трех худших по общему баллу учеников. Если среди остальных есть ученики, набравшие тот же общий балл, что и худший из трех, то следует вывести и их фамилии и имена.

Листинг программы примера:

```
//primer 3_24.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
#include <stdio.h>
#include <string.h>

void func()
{
        cout<< "Input Error\n";
        _getch();
        exit(1);
}
```

```
struct student
{
	char familia [20];
	char imja [15];
	short int m [3];
	int sum;
} p[100];
using namespace std;
int main()
{
//введение исходных данных
	int n, i, j, s1, s2, s3;
	cout<<"input N\n";
	cin>>n;
	if (n<10 || n>100) func();

for (i = 0; i<n; i++)
{
	cout<<"input familia, imja\n";
	cin>> p[i].familia;
	if (strlen(p[i].familia) > 20) func();
	cin>> p[i].imja;
	if (strlen(p[i].imja)>15) func();
	for (j = 0; j<3; j++)
		{
			cout<<"input evalution\n";
			cin>>p[i].m[j];
			if (p[i].m[j]>5 || p[i].m[j]<1) func();
		}
	}
//вычисление общего балла
	for (i = 0; i<n; i++)
	{
		p[i].sum = 0;
		for (j = 0; j<3; j++)
		{
```

```
                p[i].sum = p[i].sum + p[i].m[j];
            }
        }
//поиск 3 худших результатов
        s1 = 20; s2 = 20; s3 = 20;
        for (i = 0; i<n; i++)
        {
            if (p[i].sum<s1) { s3 = s2; s2 = s1; s1 = p[i].sum; }
            else
                if (p[i].sum<s2) { s3 = s2; s2 = p[i].sum; }
                else
                    if (p[i].sum<s3) s3 = p[i].sum;
        }

/*поиск учеников, имеющих тоже количество баллов, что и худший из трех */
for (i = 0; i<n; i++)
        if (p[i].sum == s3)cout<<"\n"<< p[i].familia;
_getch();
return 0;
}
```

Вопросы для самоконтроля

1. Что объединяет структура?
2. Каким образом осуществляется доступ к полям структуры?
3. Выделяется ли память под переменную при определении структуры?
4. Может ли имя поля структуры совпадать с именем самой структуры?
5. Обязательно ли имена полей структуры должны быть различны?
6. Может ли структура содержать только одно поле?
7. Какими способами можно заполнить значение полей структуры? Приведите примеры.
8. Как можно вывести на экран значения полей структуры? Приведите примеры.

Задания для самостоятельной работы

Порядок выполнения

1. составьте блок-схему решения задачи индивидуального задания;
2. напишите программу на языке C++ для разработанного алгоритма решения задачи;
3. выполните отладку и компиляцию программы, получите исполняемые файлы;
4. выполните тестирование программы;
5. оформите результаты работы в форме отчета, оформите результаты работы в форме отчета, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования C++, тестовые данные.

Задания 1[1]

1. На вход программы подаются сведения об участниках массовки, пришедших на съемки фильма и получивших зарплату пропорционально отработанному времени. В первой строке задано текущее время начала съемки: через двоеточие два целых числа, соответствующие часам (от 00 до 23 – ровно 2 символа) и минутам (от 00 до 59 – ровно 2 символа). Во второй строке сообщается количество участников съемки N, которое не меньше 10, но не превосходит 1000. Каждая из следующих N строк имеет следующий формат: <Фамилия> <время начала съемки>, где <Фамилия> - строка, состоящая не более, чем из 20 символов, < время начала съемки > - через двоеточие два целых числа, соответствующие часам и минутам. Сведения отсортированы в порядке времени начала съемки. Требуется написать программу, которая выведет фамилии участников массовки, которые после 6 часов съемок должны освободиться в хронологическом порядке.
Пример входных данных:
10: 00
3
Иванов 14:00

[1] Использованы материалы Павловская Т.А., Щупак Ю.А. C/C++. Структурное программирование: Практикум. СПб.: Питер,2003. 240 с.

Петров 15: 00
Сидоров 11:30
Результат работы программы для этого примера
Петров
Иванов

2. Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов);

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию среднего балла;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4.0;
- если таких студентов нет, вывести соответствующее сообщение.
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, имеющих оценки 4 и 5;
- если таких студентов нет, вывести соответствующее сообщение.

3. На вход программе подаются сведения о номерах школ учащихся, участвовавших в районной олимпиаде по информатике. В первой строке сообщается количество учащихся N (N<=1000), каждая из следующих N строк имеет формат: <Фамилия> <Инициалы> <номер школы>, где <Фамилия> - строка, состоящая не более чем из 20 символов, <Инициалы> - строка, состоящая из 4 символов (буква, точка, буква, точка), <номер школы> - не более чем двузначный номер. Данные при вводе разделить одним пробелом. Пример входной строки: Иванов П. С. 57

Требуется написать программу, которая будет выводить на экран информацию, из какой школы было меньше всего участников (таких школ может быть несколько). При этом необходимо вывести информацию только по школам, пославшим хотя бы одного участника.

4. Описать структуру с именем AEROFLOT, содержащую следующие поля:

- название пункта назначения рейса;

– номер рейса;

– тип самолета;

Написать программу, выполняющую следующие действия:

– ввод с клавиатуры данных в массив, состоящий из 7 элементов типа AEROFLOT; записи должны быть упорядочены по возрастанию номера рейса;

– вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры;

– если таких рейсов нет, вывести соответствующее сообщение.

5. На вход программе подается последовательность целых чисел. В первой строке сообщается количество чисел N, которое должно быть не больше 100, во второй строке идут сами числа. Требуется написать программу, которая будет выводить на экран числа в следующем порядке: сначала отрицательные числа, потом положительные. При этом должна сохраняться исходное взаимное положение, как среди отрицательных, так и среди положительных чисел.

2.10. Практическая работа № 9 «Функции»

1. Цель работы

Приобретение практических навыков в работе с пользовательскими функциями в языке C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Функции

Понятие функции в C++ аналогично понятию подпрограмма, которое имеется в большинстве языков программирования. Их использование связано с принципами модульного программирования, в основе которого лежит разделение программы на автономные фрагменты или модули.

Таким образом, функции в C++ - это самостоятельные единицы программы, спроектированные для решения конкретных задач, обычно повторяющихся несколько раз. Примером являются все рассмотренные ранее функции стандартных библиотек stdio.h, conio.h, math.h.

Однако пользователь может не только использовать готовые функции, но и создавать собственные. Более того, несколько функций пользователя могут быть впоследствии объединены в библиотеку.

Определение функции

Определение содержит код функции. Основная форма определения функции имеет вид:

тип возвращаемого значения имя функции (список параметров)
{
тело функции
}

Тип возвращаемого значения указывается тип значения, возвращаемого функцией. Это может быть практически любой тип. Если функция не возвращает никакого значения, необходимо указать тип void. Если функция действительно возвращает значение, онодолжно иметь тип, совместимый с указанным в определении функции.

Имя функции, как правило, отражает выполняемое ею действие. В качестве имени можно использовать любой допустимый идентификатор, который еще не был задействован в программе. После имени функции в круглых скобках указывается *список параметров,* который представляет собой последовательность пар (состоящих из типа данных и имени), разделенных запятыми. Параметры — это, по сути, переменные, которые получают значение аргументов, передаваемых функции при вызове. Если функция не имеет параметров, элемент *список параметров* отсутствует, т.е. круглые скобки остаются пустыми.

В фигурные скобки заключено *тело функции*. *Тело функции* состоит из последовательности операторов. Когда происходит вызов функции, программа передает управление первому оператору тела функции. Затем исполняются операторы, находящиеся в теле функции, и когда достигается закрывающаяся фигурная скобкаили инструкции return, управление передается обратно вызывающей программе. Для *обращения к функции* (ее вызова) используется имя функции с указанием набора передаваемых ей параметров в круглых скобках.

Примером функции может служить функция, выбирающая наименьшее из трех чисел в приведенном ниже листинге программы:

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
using namespace std;
float min (float a, float b, float c)    // определяем функцию min
        {
// определяем локальную переменную m
          float m = a;

// выбираем наименьшее из двух чисел
          if (b<m) m = b;
          if (c<m) m = c;
// возвращаем результат к месту вызова
          return m;

        }

int main()
{
        setlocale(LC_CTYPE, "russian");
        float a, b, c; // задаем три аргумента для функции
        cout<< "введите числа a,b,c" << "\n";
        cin>> a >> b >> c;
        float minimal;              // объявляем переменную для результата

        minimal = min(a, b, c); // вызываем функцию
        cout<< "min=" <<minimal;   // выводим результат на экран
        _getch ();

return 0;
}
```

Прототип функции

Прототип функции – это ее опережающее объявление. В прототипе функции указывается только имя функции, тип результата и типы аргументов. Он не содержит реализации (тела) функции, а всего лишь указывает компилятору, что ее определение будет дано ниже в тексте программы. Если программа использует прототип функции, то обычно он располагается перед функцией main или внутри нее, а определение функции дается после функции main.

В следующем примере, рассмотренная ранее, функция min объявлена с помощью прототипа:

```
// primer.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;
// объявляем функцию min с помощью прототипа
float min(float a, float b, float c);

int main()
{
	setlocale(LC_CTYPE, "russian");

	float a = 2.5, b = 3.1, c = 5.8;// задаем три аргумента для функции
	float minimal;		// объявляем переменную для результата

	minimal = min(a, b, c);		// вызываем функцию
	cout<< "min=" <<minimal;	// выводим результат на экран
	_getch();
	}
```

```
float min(float a, float b, float c)     // определяем функцию min
        {
                float m = a;
                if (b<m) m = b;
                if (c<m) m = c;
                return m;
        }
```

Передача аргументов в функцию

Аргументом называют единицу данных (например, переменную типа int), передаваемую программой в функцию. Аргументы позволяют функции оперировать различными значениями или выполнять различные действия в зависимости от переданных ей значений.

В качестве аргумента могут быть константы, значения переменных, структурные переменные. В случае передачи констант при вызове функции вместо аргументов в скобках указывают их конкретные значения. В случае передачи значений переменных в функцию типы и имена этих переменных указываются в прототипе при определении функции (см. примеры выше). Структурные переменные, в качестве аргументов функций используются, как и любые другие переменные стандартного типа.

Рассмотрим пример, в котором аргументом функции будет структура Distance, представляющая длину в английской системе мер.

```
// primer.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;
// объявляем структуру
        struct Distance
        {
                int feet;
                float inches;
        };
```

```
//объявляем функцию
void eng (Distance);
int main ()
{
	setlocale(LC_CTYPE, "russian");
//объявляем переменные типа  Distance
	Distance d1, d2, d3;
	cout<< "введите число футов ";
	cin>> d1.feet;
	cout<< "введите число дюймов ";
	cin>> d1.inches;

	cout<< "введите число футов ";
	cin>> d2.feet;
	cout<< "введите число дюймов ";
	cin>> d2.inches;
//складываем дюймы
	d3.inches = d1.inches + d2.inches;
	d3.feet = 0;
	if (d3.inches >= 12.0)
		{
		  d3.inches -= 12.0;
		  d3.feet++;
		}
	d3.feet += d1.feet + d2.feet;
	eng(d1);
	eng(d2);
	eng(d3);
	_getch();

	return 0;
}
//определяем функцию  eng, выводящую результат каждый раз в новой строке
void eng (Distance dd)
{
	cout<< "\n";
	cout<<dd.feet<< "'-" <<dd.inches<< "" ";
}
```

Оператор return

Оператор return, с которым вы уже встречались, имеет два варианта использования. Во-первых, этот оператор может использоваться для возврата значения функции (см. примеры выше). Во-вторых, этот оператор вызывает немедленный выход из текущей функции и возврат в вызывающую программу. Продемонстрируем сказанное на примере функции *power()*, отображающей результат возведения целочисленного значения в положительную целую степень. Если показатель степени окажется отрицательным, инструкция return немедленно завершит эту функцию еще до попытки вычислить результат:

```
// power.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;
// объявляем и определяем функцию power
void power(int base, int exp)
{
	int i;
/* если показатель степени отрицательный, оператор return немедленно завершает функцию power */

	if (exp<0) return;
	i = 1;

// в теле функции организуем цикл по показателю степени
	for (; exp; exp--) i = base*i;
	cout<< "otvet=" << i;
}

void main()
{
// вызываем функцию и задаем ее аргументы
	power(10, 2);
```

```
    power(10, -2);
    _getch();
}
```

Область видимости и класс памяти

Изучив основы работы с функциями, рассмотрим два аспекта, касающихся взаимодействия переменных и функций: область видимости и класс памяти.

Область видимости определяет, из каких частей программы возможен доступ к переменной, а класс памяти – время, в течение которого переменная существует в памяти компьютера.

Рассмотрим типы области видимости: локальная область видимости, область видимости файла, область видимости класса. В этой лекции мы коснемся первых двух типов.

Переменные, имеющие локальную область видимости, доступны внутри того блока, в котором они определены. Блоком считается код, заключенный в фигурные скобки. Например, тело функции представляет собой блок.

Переменные, имеющие область видимости файла, доступны из любого места файла, в котором они определены.

Существуют два класса памяти: automatic (автоматический) и static (статический).

У переменных, имеющих класс памяти automatic, время жизни равно времени жизни функции, внутри которой они определены.

У переменных, имеющих класс памяти static, время жизни равно времени жизни всей программы.

Переменные, определяемые внутри функции, называются локальными, поскольку их область видимости ограничивается этой функцией. Локальные переменные имеют автоматический класс памяти.

Область видимости переменной определяет участки программы, из которых возможен доступ к этой переменной. Это означает, что внутри области видимости переменной операторы могут обращаться к ней по имени и использовать ее значение при вычислении выражений. За пределами области видимости попытки обращения к пере-

менной приведут к выдаче сообщения о том, что переменная неизвестна.

Например, имеются функция:

```
void somerfunc()
{
//локальные переменные
int somervar;
float othervar;

somervar=10;          //корректно
othervar=11;          //корректно
nextvar=12;           // некорректно, переменная невидима в функции
}
```

Рассмотрим классы памяти локальных переменных.

Локальная переменная не существует в памяти до тех пор, пока не будет вызвана функция, в которой она определена. Это означает, что в памяти нет места, выделенного для их хранения. Когда управление передается в функцию, переменные создаются и под них отводится место в памяти. Затем, когда выполнение функции завершается и управление передается вызывающей программе, переменные уничтожаются, а их значения теряются. Название автоматического класса памяти как раз указывает на то, что переменные автоматически создаются при входе в функцию и автоматически уничтожаются при выходе из нее.

В отличие от локальных переменных, определяемых внутри функции, глобальные переменные определяются вне каких-либо функций. Глобальная переменная видима из всех функций данного файла, потенциально, из других файлов. По этой причине определения глобальных переменных располагают в начале листинга.

Следующий пример демонстрирует использование глобальной переменной тремя функциями:

```
// global.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
```

```
#include <locale.h>
using namespace std;
charch = 'a';                    //глобальная переменная ch

//прототипы функций
void getachar();
void putachar();

/*пока не нажата клавиша Enter, вводим символ и он отражается на
экране, т.е. переменная ch, доступна двум функциям*/

void main()
{
while (ch != '\r')
        {
        getachar();
        putachar();
        }
}
void getachar()
{
ch = _getch ();
}
void putachar ()
{
cout<< ch;
}
```

Рассмотрим еще один тип переменных, называемый статическими локальными переменными. Существуют и статические глобальные переменные, но они используются в много-файловых программах, которые мы не рассматриваем.

Статическая локальная переменная имеет такую же область видимости, как и автоматическая. Однако, время жизни у статической локальной переменной совпадает с временем жизни глобальной переменной, с той разницей, что существование статической локальной переменной начинается при первом вызове функции, к которой она

принадлежит. Далее переменная существует на протяжении выполнения программы.

Статические локальные переменные используются в тех случаях, когда необходимо сохранить значение переменной в памяти после того, как выполнение функции будет завершено.

Следующий пример демонстрирует использование статической локальной переменной:

```
// pr.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
using namespace std;

float favg (float);
void main()
{
	setlocale (LC_CTYPE, "russian");
	float data = 1, avg;
	while (data != 0)
	{
		cout<< "введите число:"<<"\n";
		cin>> data;
		avg = favg (data);
		cout<< "\n среднее значение:" <<avg<<"\n";
	}
	_getch();
}
/* функция favg () находит среднее арифметическое всех введенных значений */

float favg (float newdata)
{
//инициализация статических переменных при первом вызове
	static float total = 0;
	static int count = 0;
```

```
//увеличение счетчика
        count++;
//добавление нового значения к сумме
        total + = newdata;
//возврат нового среднего значения
        return total / count;
}
```

Перегруженные функции

Перегруженная функция выполняет различные действия, зависящие от типов данных, передаваемых ей в качестве аргументов. Рассмотрим следующий пример: пусть имеется две функции. Функция starline (), выводящая на экран линию из 45 символов «*», и функция repchar (), выводящая заданный символ заданное число раз, используя два аргумента. Создадим третью функцию charline (), которая всегда печатает 45 символов, но позволяет задавать печатаемый символ. Все эти функции выполняют схожие действия, но их имена различны. Очевидно, было бы гораздо удобнее использовать одно и тоже имя для всех трех функций, несмотря на то, что они используют различные наборы аргументов. Программа overload демонстрирует, каким образом это можно осуществить:

```
// overload.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
using namespace std;

//прототипы функций
void repchar ();
void repchar (char);
void repchar (char, int);

void main()
{
//задаем различные наборы аргументов функций
        repchar ();
        cout<< "\n";
```

```
        repchar ('=');
        cout<< "\n";
        repchar ('+', 30);
        _getch();
}
//функция  выводит 45 раз символ *
void repchar ()
{
        for (int j = 0; j<45; j++)
                cout<< "*";
}

//функция выводит 45 заданных символов
void repchar (char ch)
{
        for (int j = 0; j<45; j++)
                cout<< ch;
}
//функция выводит заданный символ заданное количество раз
void repchar (char ch, int n)
{
        for (int j = 0; j<n; j++)
                cout<<ch;
}
```

Вопросы для самоконтроля

1. Общий формат C++ - функций. Пример.
2. Вызов функции и возврат из нее. Оператор return. Пример.
3. Как выполняется программа при вызове функции?
4. Прототип функции. Общая форма записи и назначение. Пример.
5. Чем аргумент функции отличается от параметра? Если функция использует параметр (аргумент), где он объявляется?
6. Приведите пример использования функции в выражениях.
7. Может ли void-функция возвращать значение (использоваться в выражении)?
8. Каковы основные различия между локальными и глобальными переменными?

Задания для самостоятельной работы

Порядок выполнения

1. составьте блок-схему решения задачи индивидуального задания;
2. напишите программу на языке C++ для разработанного алгоритма решения задачи;
3. выполните отладку и компиляцию программы, получите исполняемые файлы;
4. выполните тестирование программы;
5. оформите результаты работы в форме отчета, оформите результаты работы в форме отчета, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования C++, тестовые данные.

1. Составьте программу вычисления $z = \frac{a^5+a^{-5}}{2a^m}$, используя перегруженную функцию power из текста лекции, вычисляющую результат возведения целочисленного значения в положительную и отрицательные целые степени.

2. Напишите функцию, вычисляющую факториал заданного целого числа N. Используйте эту функцию для вывода на экран факториалов всех целых чисел от 0 до 10.

3. Модифицируйте программу Калькулятор так, чтобы каждая арифметическая операция (сложение, вычитание, умножение и деление) выполнялась с помощью функции. Каждая функция должна принимать два структурных аргумента и возвращать значения того же типа.

4. Напишите функцию с именем hms (), имеющую три аргумента типа int: часы, минуты и секунды. Функция должна возвращать эквивалент переданного ей временного значения в секундах (типа long). Создайте программу, которая будет циклически запрашивать у пользователя ввод значения часов, минут и секунд и выводить результат на экран.

5. Напишите функцию, вычисляющую расстояние между двумя точками на плоскости (x1,y1) и (x2,y2). Используйте эту функцию для расчета суммарной длины ломаной линии, заданной набором из 3 точек.

2.11. Практическая работа № 10 «Одномерные массивы»

1. Цель работы

Приобретение практических навыков в работе с простыми структурными типами данных языка С++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Одномерные массивы

В практической работе 2 были рассмотрены типы данных, которые называются базовыми или встроенными. На основе этих типов данных язык С++ позволяет строить другие типы и структуры данных. Массив – одна из наиболее простых и известных структур данных. Под массивом в языке С++ понимают набор данных одного и того же типа, собранных под одним именем. Каждый элемент массива определяется именем массива и порядковым номером элемента, который называется индексом. Индекс в языке С++ всегда начинается с нуля и является целым числом.

Объявление массива в программе

Как и обычная переменная, перед использованием массив должен быть объявлен. Основная форма объявления массива размерности n такова: тип данных *имя массива* [размер 1] [размер 2] …. [размер n];

В данной практической работе ограничимся рассмотрением одномерного массива. При описании одномерного массива объявляющая запись имеет вид: тип *имя массива* [размер];

где тип – базовый тип элементов массива;

размер – количество его элементов.

Примеры объявлений:

```
int a[15];          // массив из 15 целочисленных элементов
                    // с именем a
float x[3];         // массив из трех элементов вещественного
                    // типа с именем x
```

Структура одномерного массива может быть представлена в виде следующей таблицы:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	…	A[n-2]	A[n-1]
5	21	-3	10	0	15		15	-9

Выбор отдельного элемента одномерного массива осуществляется указанием имени массива и его индекса в квадратных скобках. Например, в результате выполнения команды *cout <<A[0];* на экран будет выведено число 5, являющееся значением первого элемента массива, из таблицы выше. А командой *A[3]=5.5;* четвертому элементу массива будет присвоено значение 5.5.

Передача массивов в функции

Массивы могут быть использованы как аргументы функций. Рассмотрим в качестве примера программу sale, в которой массив объема продаж передается в функцию, выводящую данные в виде таблицы.

```
 //sale.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;

//задаем размер массива
const int d = 4;
/* в объявлении функции массивы-аргументы представлены типом данных и их размером */
void display (double [d]);

void main ()
{
//инициализируем массив
	double sales[d] = { 1432.07, 234.50, 654.01, 322.00 };

/*вызываем функцию. При вызове функции в качестве аргумента используется только имя массива. Это имя в действительности
```

```
представляет собой адрес массива в памяти
 */
        display (sales);
        _getch ();
}
/* функция для вывода на экран массива. Для записи аргумента-
массива используются тип его данных, имя массива и его размерности
*/
void display (double sales[d])
{
        int m;
        cout << "\n mesaz\n";
        cout << "\t\t1" << "\t\t 2" << "\t\t 3" << "\t\t4\n";
        for (m = 0; m<d; m++)
                cout << "\t\t" << sales[m];
}
```

Массивы структур

Массивы могут содержать в себе не только данные основных типов, но и структуры. Массивы структур – это полезный тип данных, используемый в различных ситуациях. Можно хранить в массиве структур личные данные сотрудников, географические особенности городов и многое другое.

Программа primer_part.cpp демонстрирует использование массива структур. Пользователь вводит номер модели, номер части и стоимость части. Программа записывает эти данные в структуру, которая, в свою очередь, является одним из элементов массива структур.

//pr_part.cpp: определяет точку входа для консольного приложения.

```
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
const int Size = 4;
struct part
{
```

```
        int modelnumber;
        int partnumber;
        float cost;
};

using namespace std;
void main()
{
        setlocale (LC_CTYPE, "russian");
        int n;
/* имя типа part указывает на то, что массив содержит данные более сложного типа */

        part apart [Size];

/* доступ к данным, членам структуры, которые являются элементами массивов */
        for (n = 0; n < Size; n++)
        {
                cout << "\n введите номер модели: ";
                cin >> apart[n].modelnumber;

                cout << "\n введите номер части: ";
                cin >> apart[n].partnumber;
                cout << "\n введите стоимость части: ";
                cin >> apart[n].cost;
        }
        for (n = 0; n<Size; n++)
        {
                cout << "Model " << apart[n].modelnumber;
                cout << " Chast " << apart[n].partnumber;
                cout << " Stoimost " << apart[n].cost << "\n";
        }
        _getch();
}
```

Типовые алгоритмы обработки одномерных массивов

Существуют типовые алгоритмы обработки одномерных массивов. Приведем типовые программы на языке C++.

Поиск элемента в упорядоченном массиве. Рассмотрим алгоритм быстрого поиска элемента в упорядоченной совокупности a_1, ….., a_n. При этом будем считать, что a_1, ….., a_n – целочисленный массив, b – некоторое целое число.

Пусть $a_1 < \ldots.. < a_n$. Рассмотрим задачу: входит ли данное число b в массив a_1, ….., a_n, и если входит, то каково значение p, для которого a_p=b? Тривиальный алгоритм решения этой задачи основывается на последовательных сравнениях b с элементами a_1, ….., a_n. В этом случае среднее число требуемых сравнений можно считать равным n/2. Известен алгоритм, который требует гораздо меньших затрат.

Предположим, что в массиве a_1,..., a_n имеется элемент, равный b, т.е. существует такое p, что a_p=b. По результату любого сравнения a_s<b ($1 \le s \le b$) мы сразу определяем, лежит ли p в диапазоне от 1 до s или же в диапазоне от s+1 до n: второе будет иметь место, если неравенство a_s<b справедливо, а первое –если несправедливо. Если s находится примерно посередине между 1 и n, то сравнение a_s<b будет сужать диапазон поиска примерно вдвое. Этот прием можно использовать многократно. Получается алгоритм, называемый алгоритмом деления пополам. В соответствии с этим алгоритмом надо взять первоначально 1 и n в качестве границ поиска индекса элемента; далее до тех пор, пока границы не совпадут, шаг за шагом сдвигать эти границы следующим образом: сравнить b с a_s где s – целая часть среднего арифметического границ, если a_s<b, то заменить прежнюю нижнюю границу на s+1, оставив верхнюю границу без изменения, иначе оставить без изменения нижнюю границу, а верхнюю заменить на s. Поиск закончится, когда границы совпадут.

Сказанное можно записать в виде последовательности операторов (p и q – верхняя и нижняя границы), когда p и q совпадут, p дает результат выполнения.

Схематично процесс поиска для a_1,..., a_n соответственно равных 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 и b=19 представлен на рис. 99.

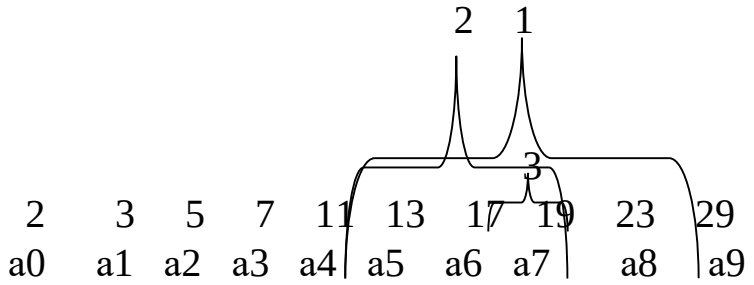

Рис. 99. Процесс поиска с использованием алгоритма деления пополам

Заметим следующее. Мы исходим из предположения, что среди элементов a_1,..., a_n имеется такой, который равен b. Если заранее неизвестно, имеется ли такой элемент, то получив p, необходимо дополнительно проверить, действительно ли a_p=b. Если обнаружится, что равенство не справедливо, то из этого будет следовать, что среди a_1, ….., a_n нет элемента, равного b. Программа primer 3_9.cpp демонстрирует применение этого алгоритма.

//primer 3_9.cpp: определяет точку входа для консольного //приложения.

```
#include "stdafx.h"
#include "iostream"
#include "conio.h"
using namespace std;

int main ()
{
	const int n = 10;
	int i, p, q, s, b = 19;
	int a [n] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
	p = 1;
	q = n;
	while (p < q)
	{
		s = (p + q) / 2;
		if (a[s] < b) p = s + 1;
		else q = s;
	}
```

```
    if (a[p] == b) printf ("%d ", p);
    else printf ("now ");
    _getch ();
    return 0;
}
```

Упорядочивание (сортировка) массива. Под *сортировкой* будем понимать размещение набора данных в определенном порядке, что используется для облегчения поиска нужного элемента или группы элементов. От эффективности алгоритма сортировки зависит, например, производительность работы баз и банков данных.

Имеются три способа сортировки: выбором, вставками и обменом. Рассмотрим алгоритм *сортировки простыми вставками*. Алгоритм сортировки методом вставок предназначен для решения задачи упорядочивания совокупности попарно различных чисел (a_1, a_2,..., a_n). Опишем этот алгоритм применительно к упорядочиванию по возрастанию элементов одномерного массива.

При сортировке вставкой сначала упорядочиваются два элемента массива. Затем делается вставка третьего элемента в соответствующее место по отношению к первым двум элементам. Затем делается вставка четвертого элемента в список из трех элементов. Этот процесс повторяется до тех пор, пока все элементы не будут упорядочены. На рис. 100 продемонстрировано, как этот алгоритм работает для массива A[6] = (5,2,4,6,1,3).

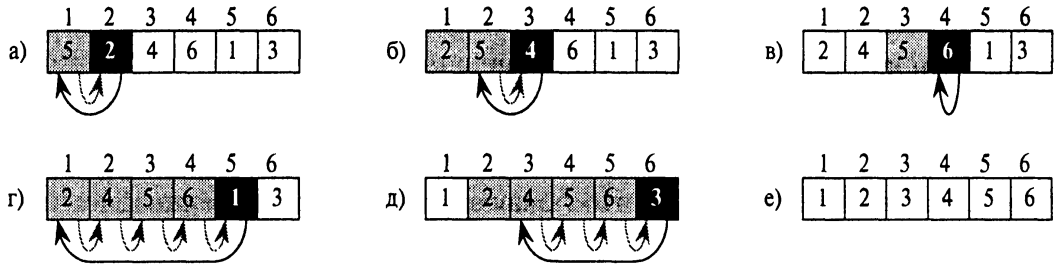

Рис. 100. Алгоритм сортировки простыми вставками

В алгоритме, работающем по методу вставок, применяется инкрементный подход: располагая отсортированным подмассивом A [l..j - 1], мы помещаем очередной элемент A[j] туда, где он должен находиться, в результате чего получаем отсортированный подмассив

A [l..j]. В программе primer 3_10.cpp показана реализация этого способа сортировки.

```
//primer 3_10.cpp: определяет точку входа для консольного
//приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
using namespace std;

int main()
{
	const int n = 6;
	int j, i, temp;
	int a[6] = { 5, 2, 4, 6, 1, 3 };

	for (i = 1; i<n; i++)
	{
		temp = a[i];
		for (j = i - 1; j >= 0; j--)
			if (temp<a[j])
			{
				a[j + 1] = a[j];

			}
		a[j + 1] = temp;
	}
	for (i = 0; i<n; i++)
		cout<<a[i]<<" ";

	getchar();
	return 0;
}
```

Метод «пузырька». Данный метод относится к сортировкам обменом. Метод «пузырька» получил своё название оттого, что продвижение максимальных элементов массива к его вершине происходит

постепенно, подобно всплытию пузырька на поверхность воды. Этот метод требует нескольких проходов массива. На каждом проходе сравнивается пара соседних друг с другом элементов.

Если пара расположена в порядке возрастания, переставляем эти элементы местами. В противном случае элементы остаются на исходных позициях. Процедура должна быть повторена n-1 раз для гарантированного достижения результата. Пример поэтапной работы алгоритма показан на рис.101. Изображенные на каждом проходе перестановки должны выполняться последовательно слева направо.

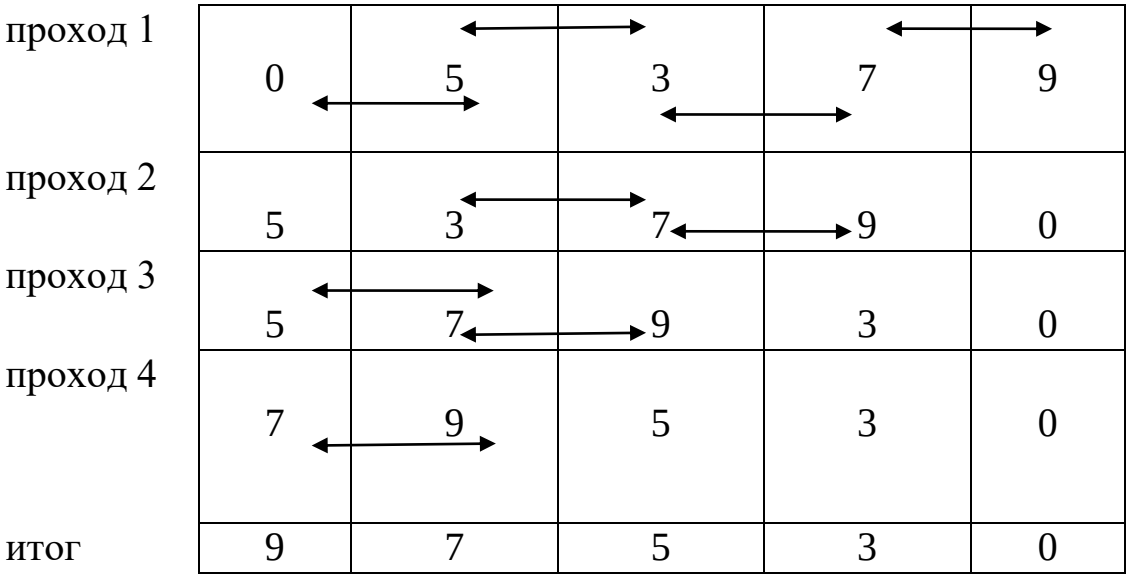

Рис. 101. Алгоритм сортировки «пузырьком»

Листинг программы primer 3_11.cpp демонстрирует реализацию этого метода.

```
//primer 3_11. cpp: определяет точку входа для консольного
// приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
using namespace std;

int main()
{
	int i, j, temp = 0;
```

```
    const int n = 5;
    int a[n] = { 0, 5, 3, 7, 9 };
    for (i = 0; i<n - 1; i++)
    {
        for (j = 0; j< n - 1; j++)
            if (a[j] < a[j + 1])
            {
              temp = a[j];
              a[j] = a[j + 1];
              a[j + 1] = temp;
            }
    }
    printf ("\n");
    for (i = 0; i < n; i++)
        printf ("%d ", a[i]);
    _getch();
    return 0;
}
```

Сортировка выбором. Алгоритм состоит в том, что выбирается наименьший элемент массива и меняется местами с первым элементом, затем рассматриваются элементы, начиная со второго, и наименьший из них меняется местами со вторым элементом, и так далее n-1 раз (при последнем проходе цикла при необходимости меняются местами предпоследний и последний элементы массива). Последовательность шагов при n=5 изображена на рис. 102 ниже.

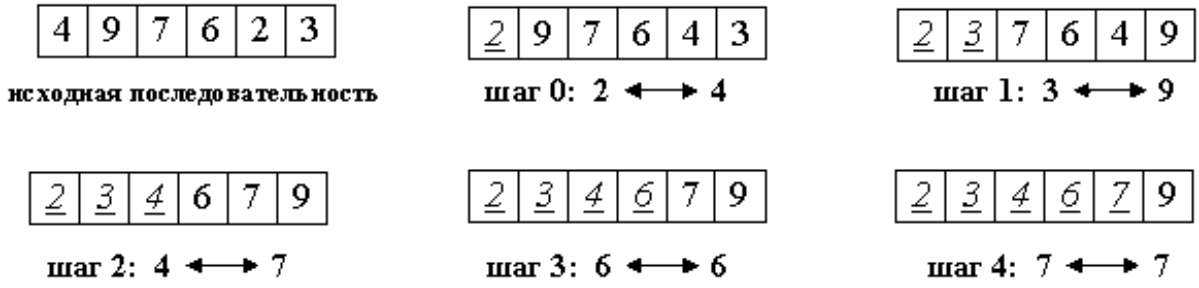

Рис. 102. Алгоритм сортировки выбором

Вне зависимости от номера текущего шага i, последовательность a[0]...a[i] (выделена курсивом) является упорядоченной. Таким образом, на (n-1)-м шаге вся последовательность, кроме a[n] оказыва-

ется отсортированной, а a[n] стоит на последнем месте по праву: все меньшие элементы уже ушли влево.

Листинг программы primer 3_12.cpp демонстрирует реализацию этого метода:

```
// 3_12.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;

int main()
{
const int n = 7;
int a[n] = { 1, 3, 4, 5, 6, 7, 1 };
int i, j, index, temp;
for (i = 0;  i< n - 1; i++)
{
        index = i;
        for (j = i + 1; j<n; j++)
                if (a[j]<a[i])
                {
                        index = j;
                        temp = a[i];
                        a[i] = a[index];
                        a[index] = temp;
                }
}
for (i = 0; I < n; i++)
        printf ("\n%d ", a[i]);
_getch ();
return 0;
}
```

Поиск минимального (максимального) элемента массива. Алгоритм поиска будет выглядеть следующим образом:

1. Будем считать минимальным (максимальным) элементом первый элемент массива.
2. Последовательно сравнить минимальный (максимальный) элемент с элементами массива, начиная от второго элемента. Если нашелся элемент меньший (больший) минимального (максимального), поместить его на место минимального (максимального) элемента.

Ниже приведен соответствующий код программы.

```
// array_min.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;
int main ()
{
	const int n = 5;
	int min, i;
	int a[n] = { 4,2,9,1,3 };
	min = a[0];
	for (i = 1; i<n; i++)
		if (a[i]<min) min = a[i];
	cout<<"\n min="<<min;
	_getch();
	return 0;
}
```

Объединение двух одномерных массивов. Типовой алгоритм объединения двух одномерных массивов заключается в применении дополнительного массива, что и продемонстрировано в программе primer 3_14.cpp. Новая деталь в этой программе – это использование числа вместо имени переменной при задании размера массива.

```
// primer 3_14.cpp: определяет точку входа для консольного
//приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;
int main ()
{
	int i;
	int mas [10];
	int a[5] = { 4, 2, 9, 1, 3 };
	int b[5] = { 5, 7, 8, 11, 12 };
	for (i = 0; I < 5; i++)
	{
		mas[i] = a[i];
		mas[5 + i] = b[i];
	}
	for (i = 0; i < 10; i++)
		printf("%d ", mas[i]);

	_getch();
	return 0;
}
```

Циклический сдвиг элементов массива. При циклическом сдвиге вправо выталкиваемые элементы с конца массива заполняют освобождающиеся места в начале массива. Например, при сдвиге вправо на 3 разряда массива A (1, 2, 3, 4, 5, 6, 7) получаем массив A (5, 6, 7, 1, 2, 3, 4). Ниже представлена соответствующая программа.

```
//array.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
using namespace std;
```

```
int main()
{
	int m, i, j;
	const int n = 7;
	int a[n] = { 1, 2, 3, 4, 5, 6, 7 };
	cout<<"input m"<<'\n';
	cin>>m;
	m = m % n;
	for (i = 0; i< m; i++)
	{
		int temp = a[n - 1];
		for (j = 0; j< n - 1; j++)
		{
			a[n - j - 1] = a[n - j - 2];
		}
		a[0] = temp;
	}
	for (i = 0; i<n; i++)
		cout<<" "<<a[i];
	_getch();
	return 0;
}
```

Вопросы для самоконтроля

1. Что такое одномерный массив? Для чего используются одномерные массивы? Как они описываются?
2. Как в программе использовать значение конкретного элемента одномерного массива?
3. Как называется номер элемента одномерного массива?
4. Напишите выражение, которое определяет одномерный массив, именованный как int Array, типа int, содержащий 56 элементов.
5. Напишите выражение, которое выводит i-ый элемент массива int Array с помощью cout.
6. Какой по счету элемент массива int Array [7]?
7. Как работает функция при передаче имени массива в функцию?
8. Что определяет выражение empl emp [100]?

Задания для самостоятельной работы[2]

В вариантах задания 1 под вставкой числа n в массив после k-ого элемента следует понимать:

1) увеличение количества элементов массива на 1, при этом исходный размер массива должен это допускать;
2) смещение всех элементов, начиная с (k+1)-ого на одну позицию вправо;
3) присваивание (k+1)-ому элементу массива значение n.

Для выполнения задания:

1. разработайте блок-схему решения задачи.
2. напишите программы на языке С ++ для разработанного алгоритма решения задачи;
3. выполните отладку и компиляцию программы, получите исполняемые файлы;
4. выполните тестирование программы, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования С++, тестовые данные.

Задания:

1. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1) сумму отрицательных элементов массива;
2) произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

2. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1) сумму положительных элементов массива;
2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

3. В одномерном массиве, состоящем из n целых элементов, вычислить:

[2] Использованы материалы Павловская Т.А., Щупак Ю.А. С/С++. Структурное программирование. СПб.: Питер, 2003.

1) произведение элементов массива с четными номерами;
2) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом — все отрицательные (элементы, равные 0, считать положительными).

4. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1) сумму элементов массива с нечетными номерами;
2) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

5. В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1) минимальный элемент массива;
2) сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом — все остальные.

2.12. Практическая работа № 11 «Двумерные массивы»

1. Цель работы

Приобретение практических навыков в работе с простыми структурными типами данных языка C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Двумерные массивы

Многомерные массивы задаются указанием каждого измерения в квадратных скобках, например, оператор int mass [6][8]: задает описание двумерного массива из 6 строк и 8 столбцов. В памяти такой массив располагается в последовательных ячейках построчно. Для доступа к элементу многомерного массива указываются все его индексы, например, mass [5][3]. Аналогом двумерного массива являются

квадратные и прямоугольные таблицы, которые часто называют *матрицами.*

Способы инициализации многомерного массива.

а) *С помощью оператора присваивания.* При инициализации многомерного массива он представляется либо как массив из массивов, при этом каждый массив заключается в свои фигурные скобки (в этом случае левую размерность при описании можно не указывать), либо задается общий список элементов в том порядке, в котором элементы располагаются в памяти:

```
int mass [] [2]={{1, 1}, {0, 2}, {1, 0}};
int mass [3] [2]={1, 1, 0, 2, 1, 0};
```

б) С помощью генератора случайных чисел

```
int a[n] [m], i, j;
srand (time (NULL));
for (i = 0; i< n; i++)
      for (  j=0; j< m; j++)
   a[i][j] = 5 + rand () % (25+1-5);  // интервал от 5 до 25
```

в) Ввод значений с клавиатуры

```
int a[n] [m], i, j;
 for (i=0; i< n; i++)
          for (  j = 0; j< m; j++)
    cin>>a[i][j];
```

Прямоугольные матрицы. Матрицы, в которых число строк не равно числу столбцов, называются прямоугольными.

Перечислим некоторые действия, которые можно выполнять над матрицами:

a. Суммой однотипных матриц A и B называют матрицу C, каждый элемент которой равен сумме соответствующих элементов матриц A и B.
b. Разностью матриц A и B называют матрицу C, каждый элемент которой равен разности соответствующих элементов матриц A и B.
c. Произведением двух матриц A и B называется такая матрица C, у которой элементы определяются по формуле $C_{ij} = \sum_{k=1}^{n} a_{ik}\, b_{kj}$, где i=1….m, j=1….p. То есть нужно перемножить соответствующие элементы i-ой строки матрицы A на

элементы j-ого столбца матрицы B и полученные произведения сложить. Примечание: число столбцов матрицы A должно равняться числу строк матрицы B.

Квадратные матрицы. Матрицы, в которых число строк равно числу столбцов, называются квадратными. Перечислим основные свойства квадратных матриц:

1. квадратные матрицы имеют главную и побочную диагонали (см. рис. 103). Рассмотрим массив A[4][4]. Если:

A[0,0]	A[0,1]	A[0,2]	**A[0,3]**
A[1,0]	**A[1,1]**	**A[1,2]**	A[1,3]
A[2,0]	**A[2,1]**	**A[2,2]**	A[2,3]
A[3,0]	A[3,1]	A[3,2]	**A[3,3]**

Побочная диагональ

Главная диагональ

Рис. 103. Диагонали матрицы

   1) i = j элементы расположены на главной диагонали;
   2) i > j элементы расположены ниже главной диагонали
   3) i < j элементы расположены выше главной диагонали
   4) i ≥ j элементы расположены на главной диагонали и ниже
   5) i ≤ j элементы расположены на главной диагонали и выше
   6) i + j = n-1 элементы расположены на побочной диагонали
   7) i + j < n-1 элементы расположены над побочной диагональю
   8) i + j> n-1 элементы расположены под побочной диагональю;
2. квадратная матрица, у которой все элементы, исключая элементы главной диагонали, равны нулю, называется диагональной матрицей:

$$D=\begin{matrix}4 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 3\end{matrix}$$

3. диагональная матрица, у которой все элементы, стоящие на главной диагонали, равны 1, называется единичной матрицей.

$$D=\begin{matrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{matrix}$$

4. Если в матрице A(m, n) поменять местами строки и столбцы, то получится матрица $A^t(m,n)$, которая называется транспонированной.

$$A=\begin{matrix} a11 & a12 & a1n \\ a21 & a22 & a2n \\ am1 & am2 & amn \end{matrix}$$

$$A^t=\begin{pmatrix} a11 & a21 & am1 \\ a12 & a22 & am2 \\ a1n & a2n & amn \end{pmatrix}$$

Приведем типовые алгоритмы обработки матриц на языке C++:

1. *Сумма элементов столбца (строки).*

```
// array.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;

int main()
{
	int i, j;
	int a[3][3] = { { 1,2 },{ 2,2 },{ 3,5 } };
	for (i = 0; i< 3; i++) {
		for (j = 0; j< 3; j++)
			cin>> a[i][j];
		cout<<"\n";
	}
	for (i = 0; i< 3; i++)
	{
		int s = 0;
		for (j = 0; j< 3; j++)
			s += a[i][j];
		cout<<"\n s="<<s;
	}
	_getch();
	return 0;
}
```

2. *Перестановка строк (столбцов) матрицы.*

Под перестановкой строк (столбцов) матрицы понимается упорядочивание (перестановка) строк (столбцов) согласно какому-либо условию.

Типовой алгоритм перестановки заключается в следующем:

1) поиск номера k строки (столбца) с элементом, отвечающим заданному условию;

2) перестановка k-ой строки (столбца) с i-ой строкой (столбцом) матрицы;

3) вывод строк (столбцов) упорядоченной матрицы.

Представленный ниже пример программы иллюстрирует алгоритм перестановки.

```
// array.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;

void func()
{
	printf ("Ошибка ввода данных\n");
	_getch ();
	exit (1);
}
int main ()
{
	setlocale (LC_CTYPE, "russian");
	const int n = 3;
	int x[n][n];
	int i, j, k, v, e, flag = 0;
//проверка, нет ли значений, выходящих за границы
	printf ("введите элемент массива x[i][j]\n");
	for (i = 0; i< n; i++)
	{
```

```
            for (j = 0; j<n; j++)
            {
                scanf_s ("%d", &x[i][j]);
                if (x[i][j] >32767 || x[i][j]< -32768)
                    flag = 1;
            }
        }

        if (flag == 1)
            func();

// проверка, нет ли одинаковых элементов в массиве
        flag = 0;
        for (i = 0; i<n; i++)
            for (j = 0; j<n; j++)
            {
                for (k = i; k<n; k++)
                {
                    for (e = j + 1; e<n; e++)
                        if (x[i][j] == x[k][e])
                            flag = 1;
                }
            }

        if (flag == 1)
            func();

// вывод элементов массива, если не было ошибок при вводе
        for (i = 0; i<n; i++)
        {
            for (j = 0; j< n; j++)
                printf ("%d  ", x[i][j]);
            printf ("\n");
        }
        printf ("\n");
```

```
/* поиск номера строки с наименьшим первым элементом, перестановка */
        for (i = 0; i<n - 1; i++)
        {
                for (j = i + 1; j< n; j++)
                        if (x[i][0] < x[j][0])
                        {
                                for (e = 0; e<n; e++)
                                {
                                        v = x[i][e];
                                        x[i][e] = x[j][e];
                                        x[j][e] = v;
                                }
                        }
        }

        // вывод упорядоченной матрицы
        for (i = 0; i< n; i++)
        {
                for (j = 0; j< n; j++)
                        printf("%d  ", x[i][j]);
                printf("\n");
        }

        _getch();
        return 0;
        }
```

3. Транспонирование матриц.

При транспонировании матрицы элементы, расположенные на главной диагонали исходной и транспонированной матриц, одни и те же. То есть, транспонировать матрицу – значит зеркально отразить ее элементы относительно главной диагонали. Сделать это можно, введя новый массив, например, как в программе, представленной ниже.

```
// array.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
```

```
#include "conio.h"

using namespace std;
int main()
{
	int i, j;
	int b[3][3];
	int a[3][3] = { { 1, 2, 3 },{ 4, 5, 6 },{ 7, 8, 9 } };
	for (i = 0; i< 3; i++)
	{
		for (j = 0; j< 3; j++)
			cout<<" "<< a[i][j];
		cout<<'\n');
	}
	cout<<'\n');
	for (i = 0; i< 3; i++)
		for (j = 0; j< 3; j++)
			b[i][j] = a[j][i];
	for (i = 0; i< 3; i++)
	{
		for (j = 0; j< 3; j++)
			cout<<" "<<b[i][j];

		cout<<'\n');
	}

	_getch();
	return 0;
    }
```

Повороты матриц.

Задачи такого типа встречаются очень часто. Рассмотрим метод их решения.

Пусть дана квадратная матрица A_{nn}, состоящая из целых чисел (рис.104). Повернем ее на 90^0 градусов по часовой стрелке. Для наглядности используем матрицу $A_{3,3}$:

$$A = \begin{pmatrix} a_{00}\, a_{01} a_{03} \\ a_{10}\, a_{11} a_{12} \\ a_{20} a_{21} a_{22} \end{pmatrix}$$

Рис. 104. Матрица до поворота

Матрица после поворота (рис.105):

$$\mathrm{A}' = \begin{pmatrix} a_{31}\, a_{21} a_{11} \\ a_{32}\, a_{22} a_{12} \\ a_{33} a_{23} a_{13} \end{pmatrix}$$

Рис. 105. Матрица после поворота

Установим соответствие между элементами матриц A и A'. Из рисунков 102,103 следует следующее отношение матриц A и A': $i = j', j + i' = n + 1 \rightarrow A'(i'j') = A(n + 1 - j, i)$.

Программа array.cpp реализует рассмотренное отношение матриц.

```
// array.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

using namespace std;

int main()
{
	const int n = 3;
	int i, j;
	int b[n][n];
	int a[n][n] = { { 1,2,3 },{ 4,5,6 },{ 7,8,9 } };
	for (i = 0; i<n; i++)
		{
		  for (j = 0; j<n; j++)
			cout<<" " <<a[i][j]);
			cout<<'\n';
		}
```

```
            cout<<'\n';
            for (i = 0; i< n; i++)
            {
                  for (j = 0; j< n; j++)
                  {
                        b[i][j] = a[n - 1 - j][i];
                        cout<<" ", b[i][j]);
                  }
                  cout<<'\n';
            }

      _getch();
      return 0;
}
```

Вопросы для самоконтроля

1. Что такое двумерный массив? Для чего используются двумерные массивы? Как они описываются?
2. Какую структуру данных описывает двумерный массив?
3. Как в программе использовать значение конкретного элемента двумерного массива?
4. Напишите выражение, которое определяет двумерный массив, именованный как int Array, типа int, содержащий 56 элементов.
5. Что определяет выражение empl emp [100] [50]?
6. Как можно заполнить двумерный массив? Приведите примеры.

Задания для самостоятельной работы

Порядок выполнения:

1. разработайте блок-схему решения задачи.
2. напишите программы на языке С ++ для разработанного алгоритма решения задачи;
3. выполните отладку и компиляцию программы, получите исполняемые файлы;
4. выполните тестирование программы

5. оформите результаты работы в форме отчета, содержащего: текст задания, блок-схему алгоритма решения задачи, текст программы на языке программирования C++, тестовые данные.

Задания:

1. Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

2. Дана целочисленная прямоугольная матрица. Определить номер строки, в которой находится самая длинная серия одинаковых элементов.

3. Дана целочисленная квадратная матрица. Определить максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

4. Написать программу которая определяет, является ли последовательность элементов главной диагонали двумерного массива упорядоченной по неубыванию. В случае отрицательного ответа должны быть напечатаны координаты первого элемента, нарушающего указанную упорядоченность.

5. Дана целочисленная прямоугольная матрица. Определить номера строк и столбцов всех седловых точек матрицы.

Примечание. Матрица A имеет седловую точку A_{ij}, если A_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

2.13. Практическая работа № 12 «Объекты и классы»

1. Цель работы

Приобретение практических навыков в создании классов и объектов языка C++. Работа состоит в последовательном изучении нижеследующих разделов, выполнении приведенных упражнений и заданий.

Объекты и классы

Основополагающей идеей объектно-ориентированного подхода является объединение данных и действий, производимых над этими данными, в единое целое, которое называется *объектом.*

Функции объекта, называемые в C++ методами или функциями-членами, обычно предназначены для доступа к данным объекта. Если необходимо считать какие-либо данные объекта, нужно вызвать соответствующий метод, который выполнит считывание и возвратит требуемое значение. Прямой доступ к данным невозможен. Данные сокрыты от внешнего воздействия, что защищает их от случайного изменения. Говорят, что данные и методы *инкапсулированы.* Термин инкапсуляция является ключевым в описании объектно-ориентированных языков.

Когда мы говори об объектах, мы говорим, что они являются экземплярами классов. *Класс* является своего рода формой, определяющей, какие данные и функции будут включены в объект класса. При объявлении класса не создаются никакие объекты этого класса, по аналогии с тем, что существование типа int еще не означает существование переменных этого типа.

Таким образом, класс является описанием совокупности сходных между собой объектов.

Определение класса

Перед использованием класс должен быть определен. Определение класса начинается с ключевого слова class, за которым следует имя класса. Тело класса заключается в фигурные скобки, после которых ставится точка с запятой (;).

Тело класса может содержать два ключевых слова: private и public. Ключевой особенностью объектно-ориентированного программи-

рования является возможность сокрытия данных. Для этой цели используется ключевое слово private. Термин «сокрытие» понимается в том смысле, что данные заключены внутри класса и защищены от несанкционированного доступа функций, расположенных вне класса. Данные, описанные с ключевым словом public, напротив доступны за пределами класса. Зачем это нужно? Сокрытие данных в толковании C++ означает ограждение данных от тех частей программы, которые не имеют необходимости использовать эти данные. В более узком смысле, это означает сокрытие данных одного класса от другого класса. Это позволяет уберечь программистов от своих ошибок. Программисты сами могут создавать средства доступа к закрытым данным, что значительно снижает вероятность случайного или некорректного доступа к ним.

Методы класса

Методы класса – это функции, входящие в состав класса. Если методы описаны с ключевым словом public, они доступны за пределами класса. Если методы описаны с ключевым словом private, они доступны только в пределах класса.

Определение объектов и вызов методов класса

Определение класса задает вид будущего объекта. Определение объекта подобно определению переменной. Например, определяются два объекта s1 и s2: smallobj s1, s2;

Доступ к методам класса возможен только через конкретный объект этого класса. Поэтому вызов метода класса имеет вид: объект . метод ();

Операцию точки называют *операцией доступа к члену класса.* Скобки позади имени метода говорят о том, что мы совершаем вызов функции. Иногда вызовы методов объектов называют *сообщениями.*

Следующий листинг программы представляет собой пример создания простого класса:

```
// klass.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>
```

```
using namespace std;
class smallobj              // определение класса
{
 private:
        int somedata;       //поле класса
 public:
  void setdata (int d)      //метод класса, изменяющий значение поля
        {
          somedata = d;
        }
  void showdata ()          //метод класса, отображающий значение поля

        {
           cout<< "значение поля равно " <<somedata<<"\n";
        }
};

int main ()
{
        setlocale (LC_CTYPE, "russian");
        smallobj s1, s2;            //определяем объекты класса

//вызов методов класса
        s1.setdata(1066);
        s2.setdata(1776);
        s1.showdata();
        s2.showdata();
        _getch();
        return 0;
}
```

Заметим, что объекты C++ могут выступать в качестве переменных типа, определенного пользователем.

Конструкторы

В предыдущем примере мы инициализировали поле объекта класса, явно вызывая соответствующий метод. Как правило, удобнее инициализировать поля объекта автоматически в момент его созда-

ния. Такой способ инициализации реализуется с помощью особого метода класса называемого *конструктором.*

У конструктора есть несколько особенностей, отличающих его от других методов класса:

1) Имя конструктора в точности совпадает с именем класса;
2) У конструкторов не существует возвращаемого значения. Это объясняется тем, что конструктор автоматически вызывает системой, и, следовательно, не существует вызывающей программы или функции, которой конструктор мог бы возвратить значение. Следовательно, отсутствует и тип возвращаемого значения.
1) Класс может иметь *несколько конструкторов* с разными параметрами при этом используется механизм перегрузки.
2) Конструктор, вызываемый без параметров, называется *конструктором по умолчанию.*
3) *Параметры конструктора* могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию. Их может содержать только один из конструкторов.
4) Если программист не указал ни одного конструктора, компилятор создает его *автоматически.* Такой конструктор берется из прародителя классов – класса Object, в котором он имеется.
5) *Конструкторы не наследуются.*
6) Конструкторы нельзя описывать с модификаторами const и static.
7) У конструктора две основные задачи: создавать в памяти экземпляр класса (если вы не писали конструктор, в этом случае он берется из прародителя классов – класса Object, в котором он имеется; вторая задача – придавать всем членам класса начальные значения.

В противоположность конструктору существует **программа-деструктор**, которая уничтожает экземпляр класса в памяти, освобождает память для других программных целей. Имя деструктора начинается со знака ~, за которым следует имя класса.

Свойства деструктора:

1) Не имеет аргументов и возвращаемого значения;
2) Не может быть объявлен с модификаторами const и static.
3) Не наследуется.

4) Деструктор определяется явным образом или создается автоматически.

```
class Ke
{
public:
   Ke(void);
   ~Ke(void);
};
```

Список инициализаций

Одна из наиболее часто возлагаемых на конструктор задач является инициализация полей объекта класса. Рекомендуется оформлять инициализацию следующим образом: инициализация располагается между прототипом метода и телом функции и предваряется двоеточием. Инициализирующее значение помещается в скобках после имени поля. Затем в фигурных скобках – пустое тело. Например: *some (): t (0)*

{ пустое тело конструктора }

Если необходимо инициализировать сразу несколько полей класса, то значения разделяются запятыми, и в результате образуется *список инициализаций*. Например: *some (): t (0), m2 (33), m3 (45)*

{ пустое тело конструктора }

В качестве примера создадим класс, объекты которого могут быть полезны для любой программы. Счетчик – это средство, предназначенное для хранения количественной меры какой-либо изменяющейся величины. Счетчик может хранить число обращений к файлу, число раз, которое пользователь нажал клавишу Enter или количество положительных чисел в массиве и т.д. обращение к счетчику происходит, как правило, для того, чтобы узнать текущее значение той величины, для измерения которой он предназначен.

В процедурных языках программирования, счетчик был бы представлен в виде глобальной переменной, однако, это не в идеологии C++. Пример COUNTER использует счетчик, значение которого может быть изменено только с помощью его собственных методов. В функции main () создаются два объекта класса counter с именами c1 и c2. В начале на экране выводятся начальные значения c1 и c2, равные 0. Затем значение счетчика c1 инкрементируется 1 раз, а значение

счетчика c2 – два раза, и программа вновь заставляет объекты вывести значение своих полей.

```
// counter.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <locale.h>

using namespace std;
class counter
{
private:
	unsigned int count;	//значение счетчика
public:
	counter() : count(0)	//конструктор
	{ }
	void inc_count()	//инкрементирование счетчика
	{
		count++;
	}
	int get_count()		//получение значения счетчика
	{
		return count;
	}
};
void main()
{
	//определение объектов класса с инициализацией
	counter c1, c2;
	cout<< "\nc1=" << c1.get_count(); //вывод начального значения c1
	cout<< "\nc2=" << c2.get_count(); //вывод начального значения c2

	//инкрементирование c1 и c2
	c1.inc_count();
	c2.inc_count();
	c2.inc_count();
	//вывод
```

```
	cout<< "\nc1=" << c1.get_count();
	cout<< "\nc2=" << c2.get_count();
	_getch();
}
```

Массивы как члены классов

Массивы могут быть использованы в качестве полей класса. Рассмотрим пример моделирования компьютерной структуры данных – стека.

```
// stack.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include "iostream"
#include "conio.h"

class Stack
{
private:
// определение переменной MAX, используемой внутри класса
	enum { MAX = 10 };
	int st[MAX];		//стек в виде массива
	int top;		//вершина стека
public:
	Stack()		//конструктор
	{
		top = 0;
	}
	void push (int var)		//поместить в стек
	{
		st[++top] = var;
	}
	int pop ()		//взять из стека
	{
		return st[top--];
	}
};
```

```
using namespace std;
void main ()
{
	Stack s1;
	s1.push(11);
	s1.push(22);
	cout << "1:" << s1.pop() << endl;
	cout << "2:" << s1.pop() << endl;
	s1.push(33);
	s1.push(44);
	s1.push(55);
	s1.push(66);
	cout << "3:" << s1.pop() << endl;
	cout << "4:" << s1.pop() << endl;
	cout << "5:" << s1.pop() << endl;
	cout << "6:" << s1.pop() << endl;
	_getch();
}
```

Вопросы и задания для самоконтроля

1. Для чего необходимо определение класса?
2. Доступность членов класса с ключевым словом private?
3. Доступность членов класса с ключевым словом public?
4. Напишите определение класса primer, включающего одно закрытое поле show типа int и одним открытым методом с прототипом void add () и конструктором, инициализирующим поле show нулем.
5. Для чего необходимо использование конструктора в программе?
6. Форма записи инициализации в конструкторе в случае одного поля (нескольких полей)?
7. Представьте пункт для взимания платежей за проезд по автостраде. Каждая проезжающая машина должна заплатить за проезд 50 центов, однако часть машин платит за проезд, а часть проезжает бесплатно. В кассе ведется учет числа проехавших машин и суммарная выручка от платы за проезд. Создайте модель такой кассы с помощью класса. Класс должен содержать

два поля. Одно из них типа unsigned int, предназначено для учета количества проехавших автомобилей, а второе, имеющее тип double, будет содержать суммарную выручку от платы проезда. Конструктор должен инициализировать оба поля нулевыми значениями. Один метод инкрементирует число машин и увеличивает на 0,50 суммарную выручку. Другой метод увеличивает на единицу число автомобилей, но оставляет без изменения выручку. Третий метод выводит оба значения на экран. Там, где это возможно, сделайте методы константными. Создайте программу, которая продемонстрирует работу класса. Программа должна предложить пользователю нажать одну клавишу для того, чтобы сымитировать заплатившего автолюбителя, и другую клавишу, чтобы сымитировать недобросовестного водителя. Нажатие клавиши q (quit – выход) должно привести к выдаче текущих значений количества машин, выручки и завершению программы.

8. Составить описание класса Vektor для объектов-векторов, задаваемых координатами начала и конца. Реализовать с помощью методов класса операции сложения и вычитания векторов, вычисления скалярного произведения векторов, вывода результатов вычисления на экран. Программа должна содержать меню для проверки всех методов класса.
9. Создайте класс matrix, упорядочивающий по возрастанию элементы каждой строки матрицы. Объектом класса matrix будет двумерный массив. Реализовать методы: инициализации массива значениями, упорядочивания элементов согласно условию задачи, вывода значений элементов массива на экран до и после упорядочивания. Программа должна содержать меню для проверки всех методов класса.

СПИСОК ЛИТЕРАТУРЫ

1. Программирование на C++ [Электронный ресурс] / Дейл Н., Уимз Ч., Хедингтон М. Пер. с англ. - М.: ДМК Пресс, 2000. - (Серия "Учебник") - http://www.studentlibrary.ru/book/ISBN5937000080.html.

2. От С к C++ [Электронный ресурс]: Учебное пособие для вузов / Каширин И.Ю., Новичков В.С. - 2-е изд., стереотип. - М.: Горячая линия - Телеком, 2012. - http://www.studentlibrary.ru/book/ISBN9785991202596.html.

3. Лафоре Р. Объектно-ориентированное программирование в C++. Классика Computer Science. 4-е изд. – СПб.: Питер, 2008. – 928 с.

4. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб.: Питер, 2003. – 461 с.

5. Страуструп Б. Язык программирования C++. Специальное издание. – Изд.-во «БИНОМ», 2017. – 1136 с.

6. Страуструп Б. Дизайн и эволюция.– СПб.: Питер, 2006. – 448 с.

7. Bjarne Stroustrup. The C++ Programming Language. 4-е изд. - Addison-Wesley Professional, 2013. – 1368 с.

ЗАКЛЮЧЕНИЕ

В результате изучения материалов учебно-практического пособия будущие специалисты будут способны использовать основные приемы обработки и представления экспериментальных данных; осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий; учитывать современные тенденции развития электроники, измерительной и вычислительной техники, информационных технологий в своей профессиональной деятельности; использовать навыки работы с компьютером, владеть методами информационных технологий, соблюдать основные требования информационной безопасности.

ПРИЛОЖЕНИЯ

Приложение 1

Microsoft Visual C++

Создание нового файла

Чтобы создать свой файл .cpp, закройте рабочую область открытого проекта (см. рис. П1). Выберите пункт *Файл* из меню *Создать* (см. рис. П1)

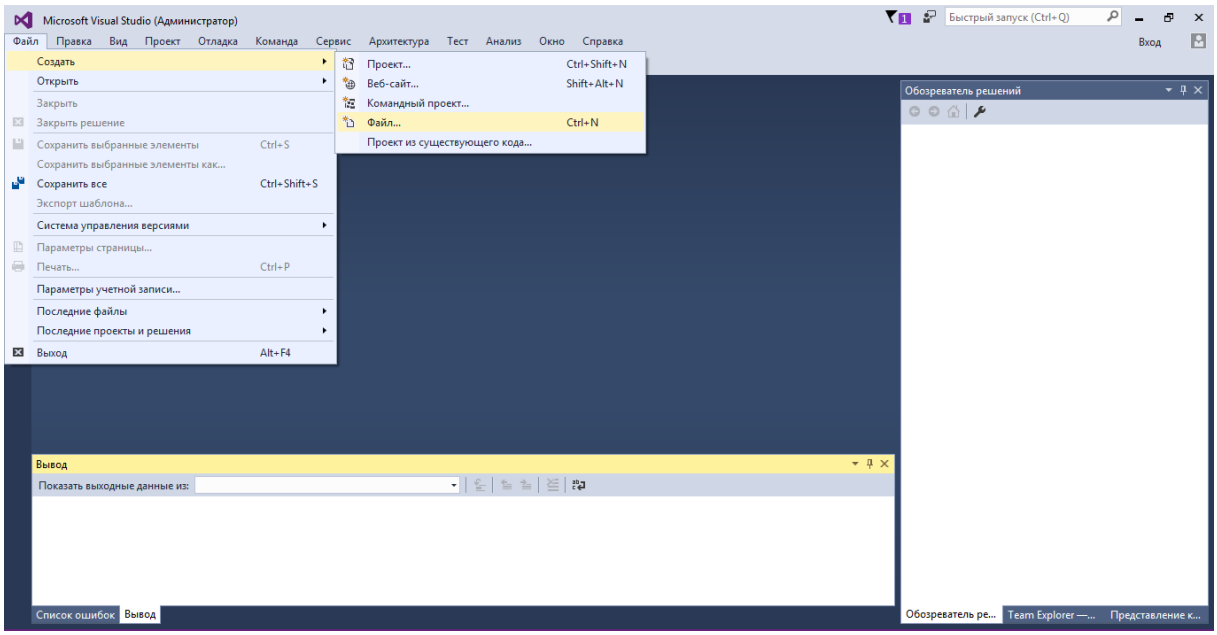

Рис. П1. Первый шаг создания нового файла .cpp

Последовательно выберите во вкладке Установленные – Visual C++, файл C++ (.cpp) (см. рис. П2).

Сохраните файл, предварительно выбрав нужный каталог и задав имя файла. Появиться чистое окно. В нём можно набирать код программы (рис. П3).

Добавление исходного файла

Файл можно добавить к существующему проекту. Для этого в меню вкладки *Обозреватель решений* последовательно выберите *Файлы исходного кода, Добавить, Существующий файл*. Файл появиться на вкладке Обозреватель решений (см. рис.П4).

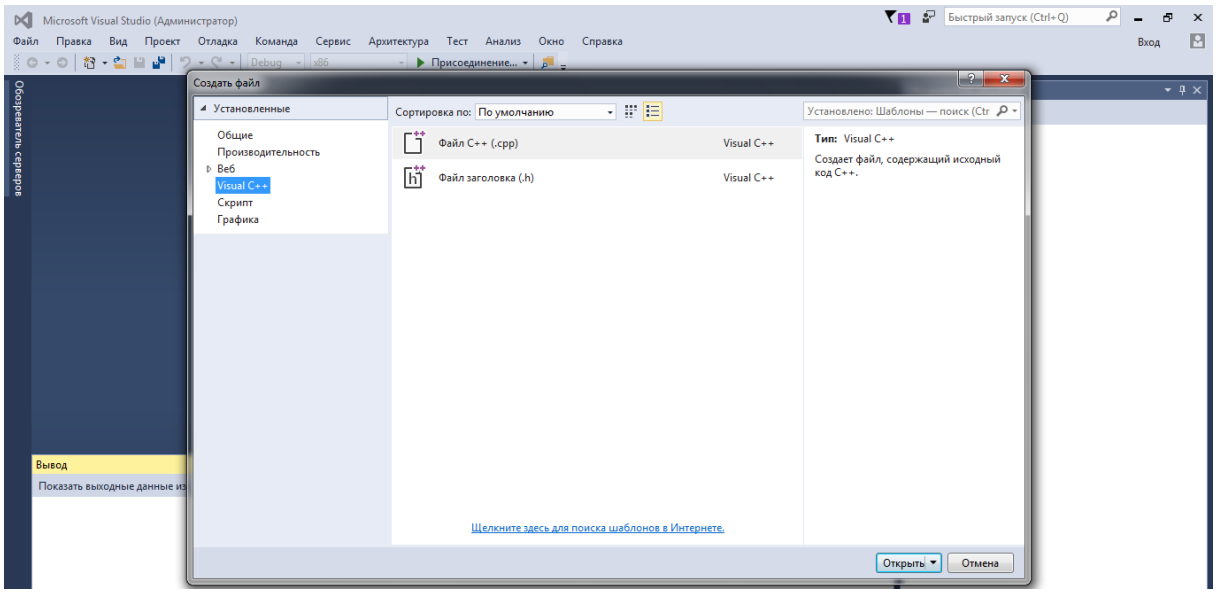

Рис. П2. Второй шаг создания нового файла

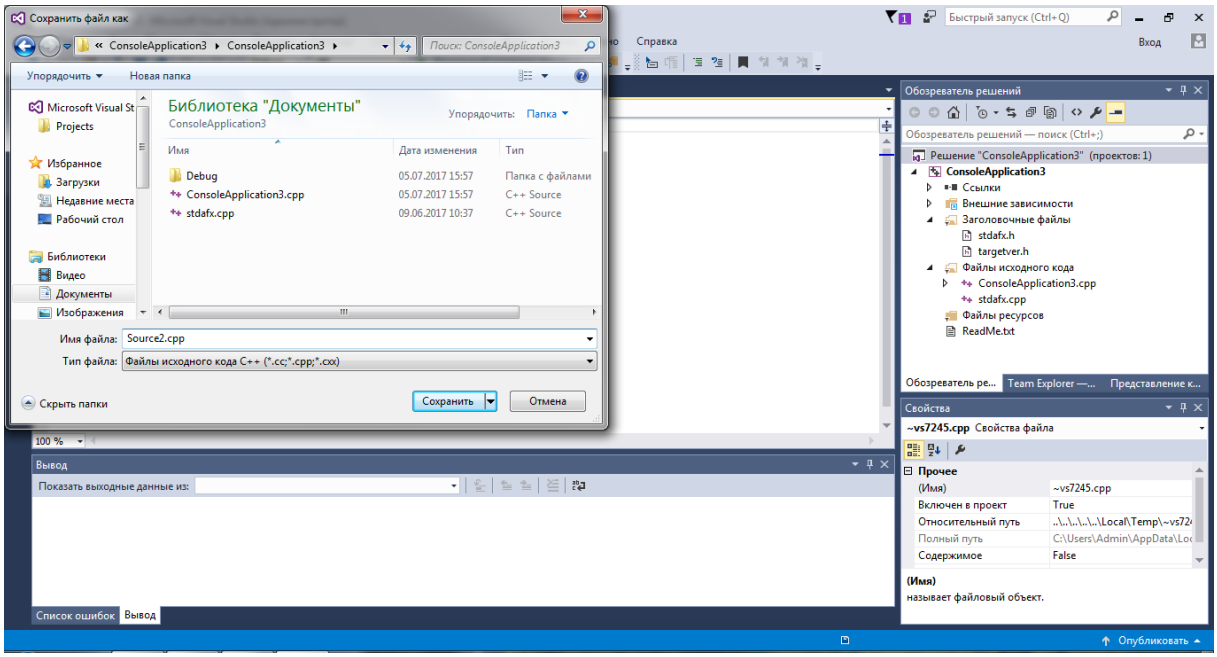

Рис. П3. Третий шаг создания нового файла

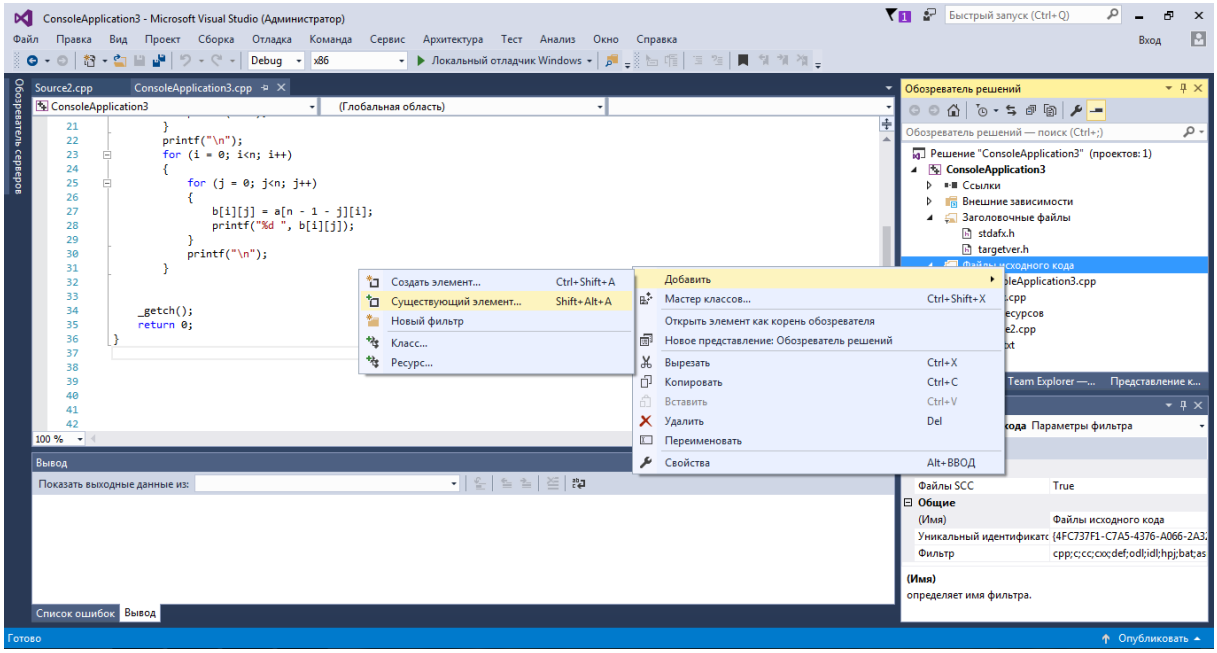

Рис. П4. Добавление нового файла к проекту

Ошибки

Если в программе имеются ошибки, сообщения об этом будут появляться в окне *Список ошибок* (см. рис. П5, П6) внизу экрана. Для этого необходимо нажать кнопку *Нет* в появившемся окне.

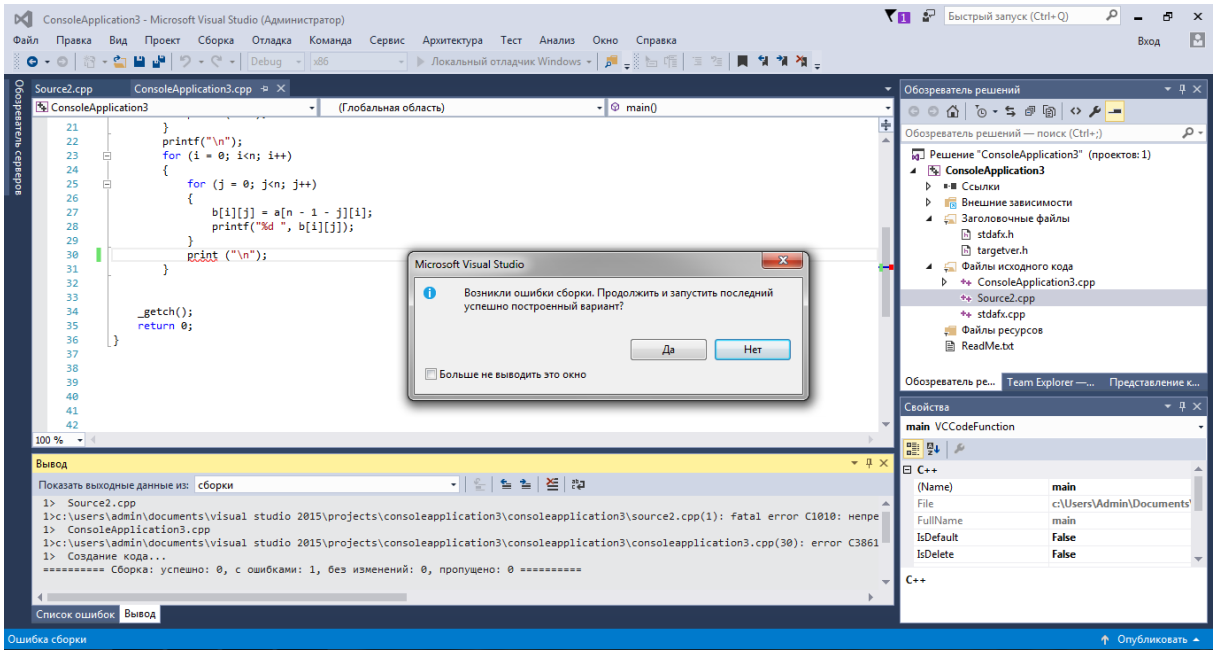

Рис. П.5. Окно Вывод процесса компоновки программы

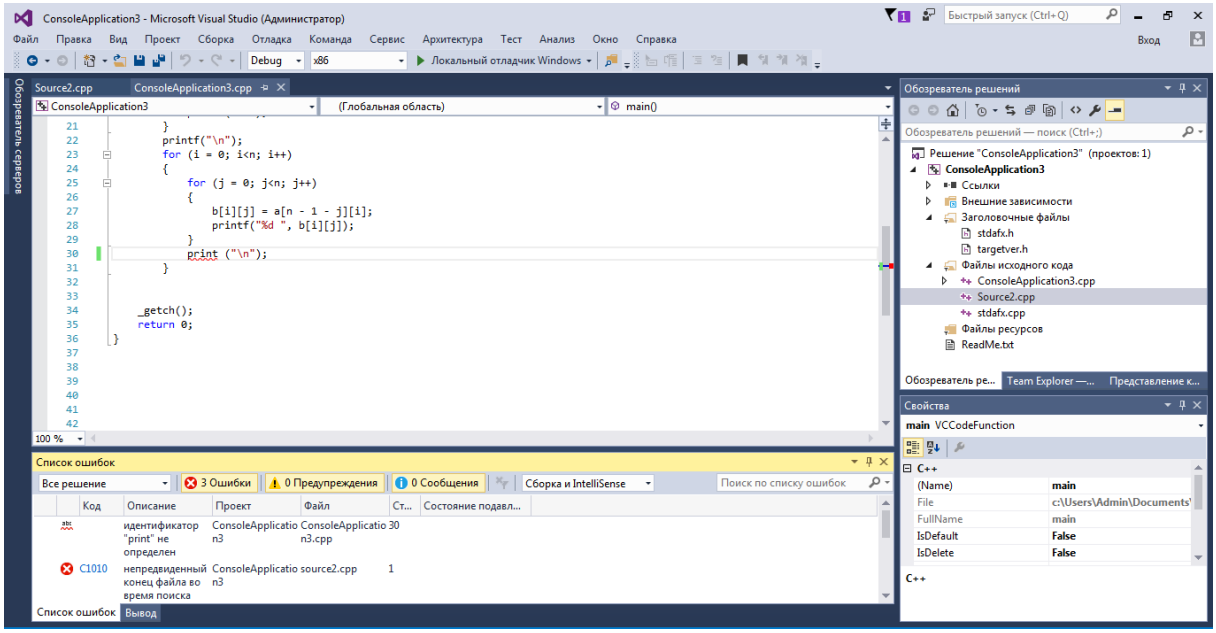

Рис. П6. Окно Список ошибок

Если дважды щёлкнуть на строке с какой-либо ошибкой, появиться стрелка в окне документа, указывающая на ту строку в коде, где эта ошибка произошла.

ГЛОССАРИЙ

AJAX (Asynchronous JavaSCript and XML) – это технология разработки web-приложений , в которой используются языки JavaScript, XML и другие механизмы; обычно сводится к применению объекта XMLHttp для обращения к серверу и изменения некоторых элементов страницы без ее полного обновления

API (Application Programming Interface) – интерфейс прикладного программирования – совокупность открытых данных и методов класса.

const – это ключевое слово, появившееся в PHP 5, которое употребляется для определения константных (постоянных) членов класса

DOM (Document Object Model – объектная модель документа) – это представление HTML или XML-документа в объектно-ориентированном виде

extends – это ключевое слово, обозначающее наследование классу, например: class Canary extends Bird

final – это модификатор, применяемый к методам или классам с целью ограничить возможность наследования. Класс с таким модификатором вообще нельзя наследовать, а метод нельзя переопределять в производных классах.

HTML (HyperTExt Markup Language) – это простой язык разметки, берущий начало от SGML и применяемый для составления web-страниц

implements – это ключевое слово, обозначающее наследование интерфейсу

Internet Explorer — программа фирмы Microsoft, входящая в состав Windows, служащая для просмотра информации, размещенной в Интернете.

Internet — всемирная глобальная Сеть (Сеть сетей). Была создана в 1995 г., на первых этапах контролировалась National Science Foundation (NSF). Представляет собой совокупность взаимосвязанных коммуникационных центров, к которым подключаются региональные

поставщики сетевых услуг и через которые осуществляется их взаимодействие.

Intranet — локальная (корпоративная) информационная сеть, построенная по принципам глобальной сети Internet.

IP (Internet Protocol) — межсетевой рабочий протокол, являющийся основой для Интернета. Благодаря наличию IP возможна маршрутизация пакетов информации между сетями и последующая сборка пакетов после того, как они достигнут места назначения. IP не отвечает за надежность доставки информации, за ее целостность, за сохранение порядка потока пакетов.

IPX/SPX — транспортные протоколы, применяемые в сетях Novell NetWare.

ISDN (Integrated Services Digital Network) — цифровая сеть с интеграцией услуг, обеспечивающая цифровое соединение между оконечными устройствами сети для предоставления широкого набора услуг, к которым пользователи получают доступ через ограниченное число стандартных многофункциональных интерфейсов; относится к классу сетей, изначально предназначенных для передачи как данных, так и голоса.

iterator – это встроенный в PHP интерфейс, который позволяет обходить (перебирать) объект

javadoc-формат – это формат внутренних комментариев; метод getDocComment в различных классах отражения позволяет извлекать отформатированные таким образом комментарии из исходного текста

MIME (Multipurpose Internet Mail Extensions) – это стандарт электронной почты в Интернете. Более широко – спецификация типа содержимого, например image/jpeg

OSI (Open Systems Interconnection reference model) — эталонная модель взаимодействия открытых систем.

PDO (PHP Data Object) – это группа классов, представляющая собой абстрагированный доступ к базам данных. По умолчанию включается в версии PHP 5.1 и старше; имеются драйверы для всех распространенных баз данных, часто используемых совместно с PHP

PEAR (PHP Extension and Application Repository – репозитарий расширений и приложений PHP) – это архив программ с открытыми исходными текстами, собранными в пакеты, которые легко установить с помощью интсаллятора PEAR

private (закрытый) – это ключевое слово, применяемое к методам и данным класса. Закрытые элементы доступны только внутри класса. Извне класса для доступа к ним применяются специальные методы. Они не наследуются производными классами.

protected (защищенный) – это ключевое слово, применяемое к методам и данным класса. Как и закрытые, защищенные члены класса недоступны извне класса, но, в отличие от закрытых, наследуются производными классами

public (открытый) - это ключевое слово, применяемое к методам и данным класса. Открытые методы класса, составляющие его интерфейс, могут вызываться от имени экземпляров класса и наследуются производными классами.

RSS (Really Simple Syndication или Reach Site Summary) – это новостной канал, который представляет собой XML-документ строго определенного формата

SDH (Synchronous Digital Hierarchy) — сети с синхронной цифровой иерархией, реализующие технологию синхронных волоконно-оптических сетей. SDH — высокоскоростные сети цифровой связи, отличающиеся высоким уровнем стандартизации, высокой надежностью, наличием полного программного контроля, возможностью оперативного предоставления услуг по требованию, сравнительно простой схемой развития сети.

SGML (Standard Generalized Markup Language) – международный стандарт представления документов; HTML и XML являются частными случаями SGML

SPL (Standard PHP Library) – набор классов, встроенных в PHP

static – это модификатор доступа применяемый к методу или свойству класса и позволяющий обращаться к этому элементу без со-

здания экземпляра класса. Статические свойства являются общими для всех экземпляров класса.

TCP (Transmission Control Protocol) — высокоуровневый протокол (Transmission Control Protocol — протокол управления передачей) с установлением логического соединения между отправителем и получателем. TCP обеспечивает сеансовую связь между двумя узлами с гарантированной доставкой информации, осуществляет контроль целостности передаваемой информации, сохраняет порядок потока пакетов.

TCP/IP — протокол для взаимодействия сетей в Интернете; представляет собой семейство программно реализованных протоколов старшего уровня, не работающих с аппаратными прерываниями; состоит из двух частей — TCP и IP.

W3C (WorldWideWebConsortium) – организация, ответственная за разработку стандартов для World Wide Web

WSDL (Web Services Definition language) – это язык на основе XML, на котором описываются web-сервисы

XHTML (eXtensible HyperTExt Markup Language) – это вариант HTML совместимый с XML

XML (eXtensible Markup Language) – это язык разметки, подобный HTML, который берет начало от SGML. XML-совместимые документы должны удовлетворять более строгим требованиям, чем предъявляются в HTML.

Zend – это языковой интерпретатор, лежащий в основе PHP; в PHP 5 был полностью переписан для поддержки объектно-ориентированных механизмов

Абстрактный метод – это метод, который объявлен, но не определен. Если в классе есть хотя бы один абстрактный метод, то ключевое слово abstract должно присутствовать и в определении класса. Все методы интерфейса абстрактны. Класс, в котором есть только абстрактные методы, называется абстрактным.

Агрегатный класс – это класс, содержащий хотя бы один член данных, также являющихся объектом.

Асимметричная операционная система (ОС) — система, которая целиком выполняется только на одном из процессоров системы, распределяя прикладные задачи по остальным процессорам.

Базовый класс – это класс, от которого произведены другие классы. Он также может называться родительским классом или суперклассом.

Безопасность ОС — означает, что операционная система должна обладать средствами защиты ресурсов одних пользователей от других.

Браузер — прикладная программа, позволяющая получать из Интернета различные документы, просматривать и редактировать их содержимое.

Будущая совместимость (forward compatibility) – написание кода с учетом будущих модернизаций языка, например, отказ от использования механизмов, объявленных устаревшими.

Верблюжья нотация – это соглашение о написании имен, согласно которому каждое слово в составном имени начинается с заглавной буквы. Имена классов при этом начинаются с заглавной буквы (DirectoryItems), а имена методов – со строчной (getName).

Волшебный метод – это метод, имя которого начинается с двух символов подчеркивания. Обычно вызывается неявно. Самые важные: __construct, __destruct, __clone.

Время вызова – это момент, в который вызывается функция или метод.

Гипертекст (гипертекстовая связь) — средство соединения информации, содержащейся в одном документе, с информацией из того же или любого другого документа, в том числе объектами нетекстовой природы (звук, изображение, видео), а также система, позволяющая читать такой текст, отслеживать ссылки, отображать картинки и проигрывать звуковые и видеовставки.

Глобальная вычислительная сеть (ГВС) — объединяет абонентские системы, рассредоточенные на большой территории, охватывающей различные страны и континенты; решает проблему объ-

единения информационных ресурсов всего человечества и организации доступа к ним.

Данные-члены – это переменные, объявленные внутри класса, но вне любого метода; называются также свойствами или переменными экземпляра.

Деструктор – это противоположность конструктору. Автоматически вызывается, когда объект покидает область видимости. Обычно применяется для того, чтобы гарантировать освобождение ресурсов, например описателей файлов.

Децентрализованная (одноранговая) сеть — локальная вычислительная сеть, функции управления в которой поочередно передаются от одной рабочей станции к другой и которая не имеет выделенных серверов.

Динамические свойства информации — свойства, которые характеризуют изменение информации во времени.

Знание — осознание, понимание и толкование определенной информации с учетом путей наилучшего ее использования для достижения конкретных целей.

Инкапсуляция – это процедура, позволяющая скрывать детали реализации, несущественные для программиста-клиента, то есть не раскрывать их в виде открытых методов и данных-членов. Соотносится с сокрытием данных, но более всеобъемлюща.

Интерфейс – это понятие имеет несколько значений. 1) ключевое слово в PHP, обозначающее класс, в котором методы объявлены, но не определены. В PHP разрешено наследовать нескольким интерфейсам. 2) совокупность открытых методов класса.

Информатизация – организационный, социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения потребностей и реализации прав граждан, органов государственной власти, органов местного самоуправления, организаций, общественных объединений на основе формирования и использования информационных ресурсов.

Информатика — отрасль знаний, изучающая общие свойства и структуру информации, а также закономерности и принципы ее создания, преобразования, накопления, передачи и использования в различных областях человеческой деятельности на базе современных средств вычислительной и телекоммуникационной техники.

Информационные процессы — процессы сбора, обработки, накопления, хранения, поиска и распространения информации.

Информационные ресурсы — отдельные документы и отдельные массивы документов, документы и массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, других информационных системах).

Источник информационного ресурса — определяет происхождение информации и в определенном смысле выступает как часть параметра охвата, ограничивающая содержание информации.

Каталоги Internet — средства хранения тематически систематизированных коллекций ссылок на различные сетевые ресурсы, в первую очередь на докумен-ты WWW.

Качество информационного ресурса — определяет (задает) совокупность свойств, отражающих степень пригодности конкретной информации об объектах и их взаимосвязях для достижения целей, стоящих перед пользователем, при реализации тех или иных видов деятельности.

Класс – это составной тип, обычно включающий данные и методы; самая фундаментальная концепция ООП.

Класс-потомок – см. производный класс.

Компьютерный вирус — программа (некоторая совокупность выполняемого кода и/или инструкций), которая способна создавать свои копии (не обязательно полностью совпадающие с оригиналом) и внедрять их в различные объекты и/или ресурсы компьютерных систем, сетей и т.д.

Конструктор – это специальный метод, который вызывается в момент создания объекта = __construct

Метаданные – это данные, с помощью которых описываются другие данные; например, информация о структуре базы данных.

Метод доступа (геттеры и сеттеры) – это открытый метод, служащий для получения или изменения данных-членов. Методы доступа называются также методами get и set. Считается хорошим стилем делать данные-члены закрытыми, а обращаться к ним для чтения или изменения только с помощью методов доступа.

Метод – это функция, определенная внутри класса.

Мировые информационные ресурсы — информационные ресурсы, которые рассматриваются как совокупность информационных ресурсов различных государств.

Многозадачные ОС — системы, которые управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Многонитевая ОС — система, которая разделяет процессорное время не между задачами, а между их отдельными ветвями (нитями).

Монолитное ядро ОС — компонуется как одна программа, работающая в привилегированном режиме и использующая быстрые переходы с одной процедуры на другую, не требующие переключения между привилегированным и пользовательским режимами.

Надежность и отказоустойчивость ОС — означает, что система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны быть способны наносить вред ОС.

Наличие нескольких прикладных сред — дает возможность в рамках одной ОС одновременно выполнять приложения, разработанные для нескольких ОС. Многие современные операционные системы поддерживают одновременно прикладные среды MS-DOS, Windows, UNIX (POSIX), OS/2 или хотя бы некоторого подмножества из этого популярного набора. Концепция множественных прикладных сред наиболее просто реализуется в ОС на базе микроядра, над которым работают различные серверы, часть которых реализует прикладную среду той или иной операционной системы.

Наследование – это способность объектно-ориентированного языка передавать методы и данные существующего класса новому, то есть от родителя к потомку.

Новый информационный ресурс — информационный ресурс, который порожден впервые и не представляет собой повторения тождественного или аналогичного.

Обертка, обертывающий метод – это метод, который просто заключает в себе вызов другого метода или функции.

Область видимости – это контекст, в котором возможен доступ к переменной. Переменная, определенная внутри метода, видима только в этом методе, а областью видимости переменной экземпляра/объекта является весь класс.

Обратная совместимость – это свойство версии языка программирования или приложения, позволяющая работать с предыдущими версиями или файлами, созданными предыдущими версиями программы.

Объем охвата информационного ресурса — общее количество информации по проблеме, доступной пользователю.

Однозадачные ОС — выполняют функцию предоставления пользователю виртуальной машины, делая более простым и удобным процесс взаимодействия пользователя с компьютером. Однозадачные ОС включают средства управления периферийными устройствами, средства управления файлами, средства общения с пользователем.

Одноранговая сеть — сеть, в которой два или несколько компьютеров могут взаимодействовать друг с другом, не прибегая к каким-либо промежуточным устройствам; в одноранговой сети компьютеры могут быть одновременно и клиентами, и серверами.

Оператор разрешения области видимости – это двойное двоеточие. Оператор :: употребляемый вместе с именем класса для ссылки на константы или статические методы класса.

Операционная система (ОС) — это программный комплекс, одной из важнейших задач которого является предоставление пользователю возможности использовать ресурсы компьютера по своему

усмотрению в максимально доступном объеме, не отвлекаясь на проблемы управления аппаратными ресурсами, находящиеся за гранью его возможностей.

ОС на базе микроядра — работает также в привилегированном режиме и выполняет только минимум функций по управлению аппаратурой, в то время как функции ОС более высокого уровня выполняют специализированные компоненты — ОС-серверы, работающие в пользовательском режиме.

Охват информационного ресурса — определяет, ограничивает и описывает содержание, уточняет или ограничивает его. В конкретном смысле охват можно рассматривать как часть параметра «содержание». Он как бы суживает и задает определенные рамки содержания.

Пара имя/значение – это формат строки запроса, передаваемой web-странице; любая строка запроса состоит из таких пар. Доступ к ним можно получить с помощью глобальных массивов $_POST и $_GET, причем имя выступает в качестве ключа.

Паттерн проектирования – это общее описание решения проблемы; нечто напоминающее абстрактный класс или интерфейс, но еще менее конкретное.

Перегруженный метод – это характеристика метода, когда речь идет о различном поведении в зависимости от параметров. В PHP этот термин обычно употребляется в отношении методов __call, __set, __get в том смысле, что один метод может обрабатывать различные свойства и методы. Поскольку PHP – слабо типизированный язык программирования, то в нем невозможна перегрузка в традиционном для других ОО-языков смысле – когда методы могут иметь одно имя, но разные сигнатуры.

Переменная класса – это статический член данных, принадлежащий классу в целом, а не его конкретному экземпляру.

Переносимость ОС — имеет место при условии, что код легко переносится с процессора одного типа на процессор другого типа и с аппаратной платформы (которая включает наряду с типом процессора

и способ организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа.

Переопределение – это изменение определения метода базового класса в производном.

Поверхностное копирование – это побитовое копирование объекта. Следует избегать при копировании агрегатных объектов.

Подсеть — часть сети TCP/IP, в которой все устройства имеют одинаковый префикс.

Поисковый сервер (search engine) — специальное программное обеспечение, которое, автоматически просматривая все ресурсы сети Internet, может найти запрашиваемые ресурсы и проиндексировать их содержание.

Полезность информационного ресурса — характеризует пригодность для определенной цели, способность функционировать в чьих-либо интересах, в соответствии с чьими-нибудь выгодами.

Полиморфизм – в строгом смысле, это возможность скопировать объект производного класса в переменную, принадлежащую базовому классу, так что при этом будут вызываться методы производного класса. В PHP применяется ослабленный вариант.

Полнота охвата информационного ресурса — соотношение между имеющейся информацией по проблеме и информацией, доступной пользователю (т.е. той ее частью, которую он может получить).

Построение ОС на базе объектно ориентированного подхода — дает возможность использовать все достоинства подхода, хорошо зарекомендовавшие себя на уровне приложений, внутри операционной системы, а именно: аккумуляцию удачных решений в форме стандартных объектов, возможность создания новых объектов на базе имеющихся с помощью механизма наследования, хорошую защиту данных за счет их инкапсуляции во внутренние структуры объекта, что делает данные недоступными для несанкционированного использования извне, структурированность системы, состоящей из набора хорошо определенных объектов.

Почтовые списки — списки рассылки (Mailing Lists) — представляют собой один из видов сервиса глобальной сети, когда в Сети выделяется адрес электронной почты, который является общим для многих пользователей — подписчиков определенного списка рассылки; пользователи-подписчики посылают свои сообщения по общему адресу, и эти сообщения рассылаются всем, кто подписался на данный список рассылки.

Прагматические свойства информации — свойства, которые характеризуют степень полезности информации для пользователя, потребителя и практики.

Прикладной процесс — различные процедуры ввода, хранения, обработки и выдачи информации, выполняемые в интересах пользователей и описываемые прикладными программами.

Программист-клиент – это пользователь класса в отличие от его автора; иногда называется программистом-пользователем.

Программные антивирусные средства, или антивирусы — специальные программы, которые находят и уничтожают вирусы на компьютере.

Производительность ОС — означает, что система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

Производный класс – это любой класс, у которого есть родительский или базовый класс. Также называется классом-потомком или подклассом.

Прототип – это объявление функции/метода, предшествующее ее определению (в некоторых языках это обязательно, но PHP к ним не относиться); может применяться и к объявлению методов, особенно методов интерфейса.

Процедурный – это вид языка программирования, в котором широко используются процедуры; например, язык С является процедурным, а PHP может рассматриваться и как процедурный, и как объектно-ориентированный язык.

Рабочая станция — обычный персональный компьютер, на котором пользователи Сети реализуют прикладные задачи.

Распределенная организация операционной системы — позволяет упростить работу пользователей и программистов в сетевых средах. В распределенной ОС реализованы механизмы, которые дают возможность пользователю представлять и воспринимать сеть в виде традиционного однопроцессорного компьютера. Характерными признаками распределенной организации ОС являются: наличие единой справочной службы разделяемых ресурсов, единой службы времени, использование механизма вызова удаленных процедур (RPC) для прозрачного распределения программных процедур по машинам, многонитевой обработки, позволяющей распараллеливать вычисления в рамках одной задачи и выполнять эту задачу сразу на нескольких компьютерах сети, а также наличие других распределенных служб.

Расширяемость ОС — имеет место при условии, что код написан таким образом, чтобы можно было легко внести дополнения и изменения, если это потребуется, и не нарушить целостность системы.

Родительский класс – см. базовый класс.

Сбор мусора – это схема автоматического управления памятью, позволяющая освобождать неиспользуемые участки памяти без вмешательства программиста.

Своевременность информационного ресурса — поступление информации в пределах того времени, когда она полезна для принятия решения и когда она еще может повлиять на результат принятия решения (деятельность).

Свойство – это синоним переменной экземпляра или члена данных.

Сервер — компьютер в Сети, обслуживающий другие компьютеры; его ресурсы доступны всем рабочим станциям Сети.

Сетевые серверы — аппаратно-программные системы, выполняющие функции управления распределением сетевых ресурсов общего доступа, имеющие возможности работать в качестве обычной абонентской системы.

Сеть — два или несколько компьютеров, соединенных друг с другом, благодаря чему они могут совместно использовать какие-либо ресурсы.

Сигнатура – это свойство, уникально характеризующее функцию или метод; состоит из имени метода, а также числа и типов его параметров. В PHP применяется в ослабленном варианте. В строго типизированных языках программирования возможно наличие методов с одинаковыми именами, но при условии, что число или типы их параметров/аргументов различны.

Симметричная ОС — полностью децентрализованная ОС, которая использует весь пул процессоров, разделяя их между системными и прикладными задачами.

Системы пакетной обработки — предназначены для решения задач в основном вычислительного характера, не требующих быстрого получения результатов.

Системы разделения времени — призваны исправить основной недостаток систем пакетной обработки — изоляцию пользователя-программиста от процесса выполнения его задач. Каждому пользователю системы разделения времени предоставляется терминал, с которого он может вести диалог со своей программой.

Системы реального времени — применяются для управления различными техническими объектами, такими, например, как станок, спутник, научная экспериментальная установка, или технологическими процессами, такими как гальваническая линия, доменный процесс и т.п.

Слабо типизированный – этот термин употребляется для характеристики языка, подобного PHP, в котором при объявлении переменной не обязательно указывать ее тип.

Служба каталогов — предоставляет средства для хранения данных о каталогах и делает эти данные доступными для пользователей и администраторов сети.

Совместимость ОС — означает, что ОС должна иметь средства для выполнения прикладных программ, написанных для других опе-

рационных систем. Кроме того, пользовательский интерфейс должен быть совместим с существующими системами и стандартами.

Содержание информационного ресурса — определяет проблемную область, охватываемую информационными ресурсами (тему, идею, теорию, методику). Границы проблемной области зависят от задач, решаемых пользователем. Причем различные группы пользователей, решающие аналогичные задачи и реализующие одинаковые цели, различным образом определяют границы проблемной области (свои потребности в информационных ресурсах), что ведет, как правило, к различию в результатах их деятельности.

Сокрытие данных – это возможность ограничить доступ к данным-членам. Также это называется защитой данных.

Сопоставление типов – это механизм, позволяющий в определении функции или метода указать перед именем его тип и тем самым запретить передачу параметров, принадлежащих типам, не совместимым с объявленным. В PHP 5.1 можно указывать также, что передается массив.

Строка запроса – это одна или несколько пар имя/значение, передаваемых web-странице в составе URL

Устаревший (deprecated) – это означает “более не рекомендуемый”. Устаревшие механизмы рано или поздно будут исключены из языка.

Ценность информационного ресурса — его важность, необходимость для принятия решений.

Экземпляр – это конкретное воплощение класса.

Учебное издание

АРТЮШИНА Лариса Андреевна
СПИРИНА Татьяна Венедиктовна
ТРОИЦКАЯ Елена Анатольевна

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
И ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Учебно-практическое пособие

Издается в авторской редакции

Подписано в печать 15.07.19.
Формат 60x84/16. Усл. печ. л. 11,86. Тираж 50 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.