МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЁЖИ И СПОРТА УКРАИНЫ
НАЦИОНАЛЬНАЯ МЕТАЛЛУРГИЧЕСКАЯ АКАДЕМИЯ УКРАИНЫ

Г.Г. ШВАЧИЧ, Е.А. ГУЛЯЕВА

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

И ПРОГРАММИРОВАНИЕ

Днепропетровск НМетАУ 2011

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЁЖИ И СПОРТА УКРАИНЫ
НАЦИОНАЛЬНАЯ МЕТАЛЛУРГИЧЕСКАЯ АКАДЕМИЯ УКРАИНЫ

Г.Г. ШВАЧИЧ, Е.А. ГУЛЯЕВА

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

И ПРОГРАММИРОВАНИЕ

Утверждено на заседании Учёного совета академии
в качестве конспекта лекций

Днепропетровск НМетАУ 2011

УДК 004(075.8)

Швачич Г.Г., Гуляева Е.А. Компьютерные технологии и программирование: Конспект лекций. − Днепропетровск: НМетАУ, 2011. – 78 с.

Рассмотрены основные средства и приёмы вычислений в среде Excel, а также методы программирования на языках VBA, C, C++.

Предназначен для студентов специальности 6.050202 − автоматизация и компьютерные технологии (АВ01).

Илл. 12. Библиогр.: 6 наим.

Издаётся в авторской редакции.

Ответственный за выпуск Г.Г. Швачич, канд. техн. наук, проф.

Рецензенты:

Содержание

Тема 1. Компьютерные технологии

1. Данные

Кодирование данных двоичным кодом

Благодаря кодированию компьютер может обрабатывать информацию различного рода. В вычислительной технике существует двоичное кодирование, то есть данные представляются с помощью двух знаков 0 и 1.

Бит – двоичный разряд, наименьшая порция информации. Одним битом может быть представлено два понятия 0 или 1 (да или нет).

С помощью 2-х битов может быть представлено 4 различных понятия:

00 01 10 11 (то есть 2^2).

С помощью 3-х битов может быть представлено 8 различных понятий (2^3), и т.д.

При увеличении на 1 единицу количества разрядов мы в 2 раза увеличиваем количество значений, которые могут быть выражены в данной системе.

Общая формула имеет вид:

$\mathbf{N=2^m}$, где N – количество независимых кодируемых значений
m – разрядность процессора

Первые ПК были 8-разрядными. В последствии появились 16, 32, 64, 128 разрядные процессоры и как следствие возросли возможности ПК.

Для кодирования целых чисел от 0 до 255 достаточно иметь 8 разрядов двоичного кода (8 бит). 16 бит позволяют закодировать целые числа от 0 до 65535 и т.д.

Кодирование текстовых данных

Каждому символу алфавита ставится в соответствие целое число (на пример порядковый номер). Восьми двоичных разрядов достаточно для кодирования 256 различных символов.

За основу кодировки символов взята система кодирования ASCII (American Standard Code for Information Interchange). Эта система состоит из 2-х таблиц – базовой и расширенной.

Базовая таблица закрепляет значения кодов от 0 до 127. Это управляющие коды, коды символов английского языка, знаки препинания, цифры, арифметические действия.

Расширенная часть определяет значения кодов от 128 до 255, это национальные системы кодирования (русский алфавит).

Универсальная система кодирования текстовых данных UNICODE основана на 16-разрядном кодировании, это позволяет обеспечить уникальные

коды для 2^{16} =65536 символов, этого достаточно для размещения в одной таблице символов большинства языков планеты.

Устройство ввода преобразует символы (текст) в двоичные коды. При выводе информации выполняется обратное преобразование: двоичный код → обычное представление.

Кодирование графических данных

Если рассмотреть с помощью увеличительного стекла чёрно-белое изображение, напечатанное в газете или книге, то можно увидеть, что оно состоит из мельчайших точек, образующих характерный узор, называемый *растром*.

Яркость любой точки может быть выражена целым числом, а значит двоичным кодом. Для представления чёрно-белых иллюстраций достаточно 8 разрядов (256 оттенков серого цвета).

Для кодирования цвета одной точки необходимо 24 разряда, то есть 16,5 млн различных цветов, что близко к чувствительности человеческого глаза. Существует и 32-х разрядная система кодирования.

Единицы измерения данных

Наименьшей единицей измерения является байт. Более крупные единицы образуются добавлением префиксов кило-, мега-, гига-, тера-.

1 Кбайт=1024 байт
1 Мбайт=1024 Кбайт
1 Гбайт=1024 Мбайт
1 Тбайт=1024 Гбайт

2. Текстовый процессор Microsoft Word. Создание комплексных текстовых документов

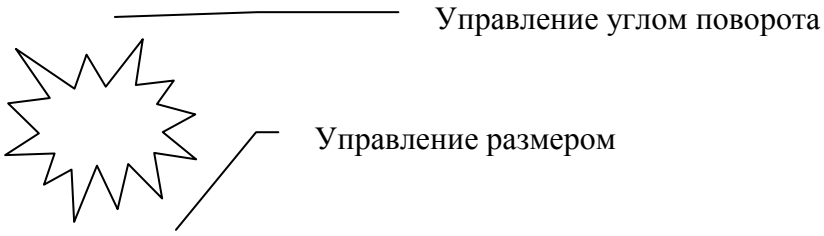
Рассмотрим приёмы создания *комплексных текстовых документов*, то есть документов, содержащих объекты, встроенные в текст.

Среди встроенных объектов могут быть стандартные объекты, созданные другими программами (например рисунки, звуковые клипы и др.), а также объекты, созданные средствами самого процессора: геометрические фигуры, художественные заголовки, диаграммы, формулы.

Пусть в текст вставлен объект созданный редствами самого процессора. Когда объект выделен, вокруг него видны восемь маркеров (см. рис.).

При наведении указателя мыши на сам объект указатель превращается в ↕, в таком состоянии объект можно перетаскивать. Вручную можно управлять только размером, поворотом и положением объекта на странице. Остальные

свойства объекта можно найти в *контекстном меню объекта→Формат объекта.*



Группировка объектов

Если на странице представлено несколько объектов и при этом важно строго зафиксировать их взаимное расположение, то их объединяют в группу, то есть в один объект. Для этого необходимо:

- выделить объекты при нажатой клавише SHIFT или рамкой;
- щёлкнуть на любом из объектов группы правой кнопкой мыши и выбрать в контекстном меню команду **Группировка →Группировать.**

Сгруппированные объекты можно перемещать как единое целое.

Чтобы разгруппировать объекты и получить доступ к индивидуальным свойствам каждого необходимо:

- выделить группу
- дать команду **Группировка →Разгруппировать.**

Ввод формул

Средством для ввода формул является редактор формул **Microsoft Equation 3.0.** Для запуска редактора формул служит команда пункта меню **Вставка → Объект → Microsoft Equation 3.0.**

Введенная формула автоматически вставляется в текст в качестве объекта. Далее её можно переместить в любое иное место документа через буфер обмена (CTRL+X — вырезать; CTRL+V — вставить).

Для редактирования формулы в документе достаточно выполнить на ней двойной щелчок. При этом автоматически открывается окно редактора формул.

Работа с таблицами

Три основных средства создания таблиц:

1. кнопка **Вставить таблицу** на панели инструментов *Стандартная;*
2. *пункт меню* **Таблица→Вставить→ Таблица;**
3. средство рисования таблиц *пункт меню* **Таблица→ Нарисовать таблицу**

Редактирование структуры таблицы: пункт меню **Таблица →**

- добавление строк (столбцов);
- удаление строк (столбцов);
- объединить выделенные ячейки и др.

Форматирование таблиц: пункт меню **Таблица → Свойства таблицы**. Вкладки окна позволяют осуществлять:

- оформление внешних и внутренних рамок таблицы (*границы и заливка*);
- изменение размеров ячеек, строк, столбцов;
- взаимодействии таблицы с окружающим текстом (*обтекание*) и др.;

Работа с графическими объектами. Создание и редактирование рисунков

В документах MS Word можно использовать два типа графических объектов: *рисунки и изображения*.

Рисунки — объекты векторной природы (линии, прямые и кривые, геометрические фигуры). Для работы с векторными рисунками служит панель инструментов *Рисование* **(Вид → Панели инструментов → Рисование).** Основным средством этой панели является раскрывающийся список **Автофигуры**.

Для размещения текста в поле автофигуры пользуются командой **Добавить текст** в контекстном меню автофигуры.

Изображения — растровые объекты. Текстовый процессор не имеет средств для их создания, поэтому они вставляются как внешние объекты из других программ. Пункт меню **Вставка →Рисунок →Из файла**

Создание графических заголовков

Для создания художественных графических надписей, например заголовков, существует программное средство **WordArt**. Доступ к нему осуществляется двумя способами:

1. через панель инструментов **WordArt (Вид→Панели инструментов→ WordArt)**
2. с помощью кнопки **Добавить объект WordArt** на панели инструментов *Рисование*

Тема 2. Табличный процессор Excel

2.1. Основные понятия

Программа Microsoft Excel предназначена для работы с таблицами данных. Документ Excel называется книгой. Книга представляет собой набор листов, каждый из которых имеет название, которое отображается на ярлыке листа.

Имена столбцов A, B, C, ….. IV (всего 256). Имена строк от 1 до 65536.

Файлы рабочих книг имеют расширение .xls.

2.2. Ввод, редактирование данных

Ввод данных осуществляется в текущую ячейку. Ячейка может содержать данные 3-х типов: текст, число, формула. Текстовые данные по умолчанию выравниваются по левому краю, а числа – по правому.

Редактирование данных в ячейке: пункт меню **Формат → Ячейка**. Вкладки этого диалогового окна позволяют:

выбирать формат записи данных

- количество знаков после запятой
- указание денежной единицы
- способ записи даты и др.

задавать направление текста и метод его выравнивания

определять шрифт и начертание символов

задавать фоновый цвет и др.

2.3. Копирование, перемещение содержимого ячеек

Копирование и перемещение в Excel можно осуществить методом перетаскивания или через буфер обмена.

Метод специального перетаскивания:

- навести указатель мыши на рамку текущей ячейки или выделенного диапазона (указатель примет вид стрелки)
- перетащить с помощью нажатой правой кнопки мыши. При отпускании кнопки появляется меню, в котором можно выбрать конкретную операцию (копировать, переместить).

Применение буфера обмена:

- выделить копируемый (вырезаемый) диапазон
- поместить его в буфер, выполнив команду **Правка → Копировать** (**Правка → Вырезать)**
- вставить данные в лист, выполнив команду **Правка → Вставить**.

2.4. Автозаполнение

В правом нижнем углу рамки текущей ячейки находится маркер заполнения.

Пример 2.1. Заполнить диапазон ячеек A1:A10 нечётными числами *с помощью маркера заполнения.*

Порядок действий:

- ввести в ячейку A1←1, в ячейку A2←3
- выделить диапазон A1:A2 при нажатой левой кнопке мыши
- протянуть за маркер заполнения вниз до ячейки A10.

Пример 2.2. Заполнить столбец **B** значениями из диапазона [-2;5] с шагом 0,5 **с помощью средств прогрессии.**

Порядок действий:

- в ячейку B1 ← -2 (начальное)
- выполнить команду **Правка → Заполнить → Прогрессия**
- в появившемся окне установить переключатели:
  - расположить по столбцам
  - арифметическая
  - ввести шаг = 0,5
  - ввести предельное значение 5
  - щелкнуть **ОК**

Аналогичным образом может быть построена геометрическая прогрессия.

2.5. Формулы

Вычисления в таблицах Excel осуществляются при помощи формул. Признак формулы — знак =. Приоритет выполнения операций в выражении:

( )	действия в скобках
sin( ),….	вычисление функций
—	унарный минус
%	взятие %
^	возведение в степень
* /	умножение, деление
+ —	сложение, вычитание
&	объединение текста
= < > >= <=	операции сравнения

2.6. Стандартные функции Excel

Вызов стандартной функции

имя функции (список аргументов)

Функция может быть записана вручную или с помощью **Мастера функций**. Вызов Мастера функций: пункт меню **Вставка → Функция.** Иначе, щелчком на кнопке f_x в строке формул.

В раскрывающемся списке **Категория** выбирается категория, к которой относится функция (если определить категорию затруднительно, используют пункт **Полный алфавитный перечень**). В списке **Выберите функцию** — конкретная функция.

В качестве аргументов может использоваться число, адрес ячейки или произвольное выражение.

Примеры записи формул.

$e^{2,4} - \sin 3,5$	$= \exp(2,4) - \sin(3,5)$
π	$= ПИ()$
$\sin 30^0$	$= \sin(30 * ПИ(\ )/180)$
$\sqrt[3]{\cos 14,5} + 45,6^2$	$= \cos(14,5)\text{^}(1/3) + 45,6\text{^}2$

2.7. Копирование формулы на блок ячеек. Относительные и абсолютные ссылки

Ссылка — это *адрес* ячейки, которая используется. По умолчанию ссылки на ячейки в формулах рассматриваются как *относительные.* Это означает, что при копировании формулы адреса автоматически изменяются в соответствии с относительным расположением исходной ячейки.

При *абсолютной адресации*, ссылки при копировании не изменяются.

A1	относительная ссылка
$A1	ссылка абсолютная по столбцу
A$1	ссылка абсолютная по строке
$A$1	абсолютная ссылка

Для изменения способа адресации надо выделить ссылку и нажать клавишу F4.

Пример 2.3. Нахождение суммы значений двух диапазонов-столбцов.

Порядок действий:

- заполнить числами диапазоны A1:A4 и B1:B4
- в ячейку C1← =A1+B1, протянуть маркер заполнения на диапазон C2:C4
- в ячейку D1← =$A$1+B1, протянуть маркер заполнения на диапазон D2:D4

	A	B	C	D
1	1	4	=A1+B1	=$A$1+B1
2	2	5	=A2+B2	=$A$1+B2
3	3	6	=A3+B3	=$A$1+B3
4	7	8	=A4+B4	=$A$1+B4

Рис. 1. Абсолютные и относительные ссылки

В режиме просмотра формул **(Сервис → Параметры, вкладка Вид→ параметры окна)** видно, что в диапазоне C2:C4 адреса ячеек настраиваются на новое месторасположение, в диапазоне D2:D4 адрес ячейки A1 не изменяется.

Пример 2.4. Дан массив X = (3,6; -4; 8; 1,6; 2). Вычислить

$$a) \quad y = \cos x + \frac{\sqrt[3]{x}}{2}$$

$$b) \quad s = \sum_{i=1}^{5} x_i$$

Результаты вычислений

	A	B	C
1	**x**	**y**	**s**
2	3,6	-0,13045	11,2
3	-4	-1,44734	
4	8	0,8545	
5	1,6	0,555604	
6	2	0,213814	

B2← = **COS(A2)+A2^(1/3)/2**

Вычисление Суммы:

C2← = **СУММ(A2:A6)**

Рис. 2. Одномерные массивы.

2.8. Присвоение имени ячейке (диапазону ячеек)

Если ячейка имеет имя, то в дальнейшем, вместо её адреса может быть использовано это имя.

Для того, чтобы **присвоить ячейке A1 имя x** достаточно

- сделать A1 активной
- в *поле имени* ввести **x**, нажать **Enter**.

Присвоение имени диапазону ячеек:

- выделить диапазон
- в *поле имени* ввести *имя*, нажать **Enter.**

Иначе: пункт меню **Вставка→Имя→Присвоить**, ввести *имя*, щелкнуть кнопку **Добавить**, *OK*.

Пример 2.5. Вычислить значения функции $Y = ax^2+bx$,
если **x** принадлежит отрезку [-2;0], $\Delta x=0,5$; $a=8$, $b=-4,6$.

Порядок действий:

Заполнить значениями столбец X

B2 ← = $C$2*A2^2+$D$2*A2

Присвоить ячейкам C2 и B2 имена **a** и **b** соответственно, заменить абсолютные адреса ячеек на их имена. Формула примет вид =a*A2^2+b*A2

Результаты вычислений

	A	B	C	D
1	**x**	**y**	**a**	**b**
2	-2	41,2	8	-4,6
3	-1,5	24,9		
4	-1	12,6		
5	-0,5	4,3		
6	0	0		

Рис. 3. Вычисление значений функции

2.9. Построение и редактирование диаграмм

В программе Excel термин «диаграмма» используется для обозначения всех видов графического представления числовых данных. Построение графического изображения производится на основе *ряда данных*.

Диаграмма сохраняет связь с данными, на основе которых она построена, и при обновлении этих данных немедленно изменяет свой вид.

Для построения диаграммы используют **Мастер диаграмм**, запускаемый щелчком по кнопке *Мастер диаграмм* на стандартной панели инструментов.

Удобно заранее выделить область, содержащую данные, которые будут отображаться, но задать эту информацию можно и в ходе работы *мастера*.

Под управлением **Мастера диаграмм** выполняются 4 шага:

Шаг 1: выбор типа диаграммы

Шаг 2: задание диапазона с данными, если они не были выделены ранее. Если диапазон с данными был выбран заранее, то в области предвариельного просмотра появится приблизительное отображение будущей диаграммы.

Шаг 3: оформление диаграммы:

- название диаграммы, подписи осей (вкладка *Заголовки*)
- отображение и маркировка осей координат(вкладка *Оси*)
- описание построенных графиков (вкладка *Легенда*)
- отображение надписей, соответствующих отдельным элементам данных на графике (вкладка *Подписи данных*) и др.

Шаг 4: указать, месторасположения диаграммы: новый рабочий лист или текущий.

Редактирование диаграммы

Диаграмма состоит из набора отдельных элементов, таких как сами графики (ряды данных), оси координат, заголовки, область построения и др.

При щелчке на элементе диаграммы он выделяется маркерами. Открыть окно для форматирования элемента диаграммы можно через меню *Формат* или через контекстное меню →*Формат*.

Что бы удалить диаграмму нужно её выделить и нажать клавишу Delete.

Пример 2.6. Построить график функции Y=cos$x$ для значений x из диапазона $[-\frac{\pi}{2}; \frac{3\pi}{2}]$ с шагом $\frac{\pi}{4}$.

Порядок действий:

1. Заполнить значениями столбцы X и Y (см. рис. 4).
   - A2 ← = -ПИ()/2
   - A3 ← = A2+ПИ()/4, скопировать формулу на диапазон A4:A10
   - B2 ← = cos(A2), скопировать формулу на диапазон B3:B10
2. Выделить диапазон A1:B10.
3. Вызвать *Мастер диаграмм* и проделать рассмотренные ранее 4 шага.

При построении графика функции удобно пользоваться типом **Точечный.** При этом, по умолчанию, 1-й столбец данных представляет собой ось X, 2-й столбец — ось Y. Получим график, представленный на рис. 4.

	A	B
1	**x**	**y**
2	-1,5708	6,12574E-17
3	-0,7854	0,707106781
4	0	1
5	0,785398	0,707106781
6	1,570796	6,12574E-17
7	2,356194	-0,707106781
8	3,141593	-1
9	3,926991	-0,707106781
10	4,712389	-1,83772E-16

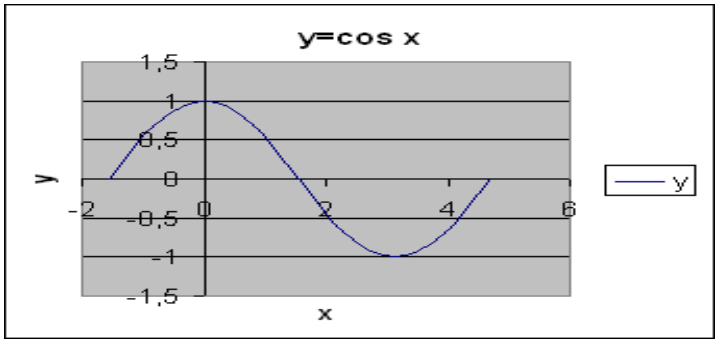


Рис. 4. Построение графика функции Y=cos$x$

2.10. Построение графика функции, заданной параметрически

Пример 2.7. Построить график зависимости y(x), если

$$x = \sin^3 t$$
$$y = \cos^3 t$$
$$t \in [0; 2\pi], \quad \Delta t = 0{,}2$$

Порядок действий (см. рис.5):

- заполнить значениями столбец **t**
- присвоить диапазону A2:A33 имя **t**
- заполнить значениями столбец X
  B2 ← =sin(t)^3
- заполнить значениями столбец Y
  C2 ← =cos(t)^3

- выделить диапазон B1:C33, вызвать *мастер диаграмм*, проделать 4 шага, тип графика — точечный.

	A	B	C
1	t	x	y
2	0	0	1
3	0,2	0,007841	0,941384
4	0,4	0,059054	0,781385
5	0,6	0,18002	0,562201
6	0,8	0,369151	0,338182
7	1	0,595823	0,157729
8	1,2	0,809659	0,047579
9	1,4	0,956981	0,00491
10	1,6	0,998721	-2,5E-05
11	1,8	0,923577	-0,01173
	…….	……	……
31	5,8	-0,10029	0,694376
32	6	-0,02181	0,885207
33	6,2	-0,00057	0,989662

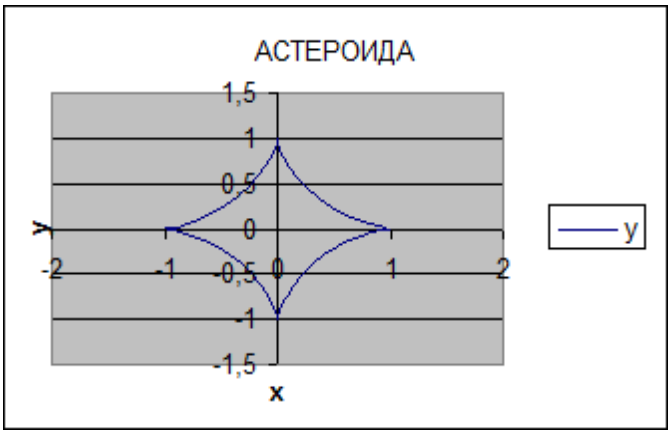


Рис.5. Астероида

2.11. Построение поверхности

Пример 2.9. Построить график функции двух переменных

$$z(x,y) = x^2 - y^2$$

$$x \in [-7{,}5;7{,}5], \qquad \Delta x = 1{,}5$$

$$y \in [-5;5], \qquad \Delta y = 1$$

Гиперболический параболоид (см. рис. 6)

	A	B	C	D	E	F	G	H	L
1	y x	-7,5	-6	-4,5	-3	-1,5	0	1,5	7,5
2	-5	31,25	11	-4,75	-16	-22,75	-25	-22,75	31,25
3	-4	40,25	20	4,25	-7	-13,75	-16	-13,75	40,25
4	-3	47,25	27	11,25	0	-6,75	-9	-6,75	47,25
5	-2	52,25	32	16,25	5	-1,75	-4	-1,75	52,25
6	-1	55,25	35	19,25	8	1,25	-1	1,25	55,25
7	0	56,25	36	20,25	9	2,25	0	2,25	56,25
8	1	55,25	35	19,25	8	1,25	-1	1,25	55,25
9	2	52,25	32	16,25	5	-1,75	-4	-1,75	52,25
10	3	47,25	27	11,25	0	-6,75	-9	-6,75	47,25
11	4	40,25	20	4,25	-7	-13,75	-16	-13,75	40,25
12	5	31,25	11	-4,75	-16	-22,75	-25	-22,75	31,25

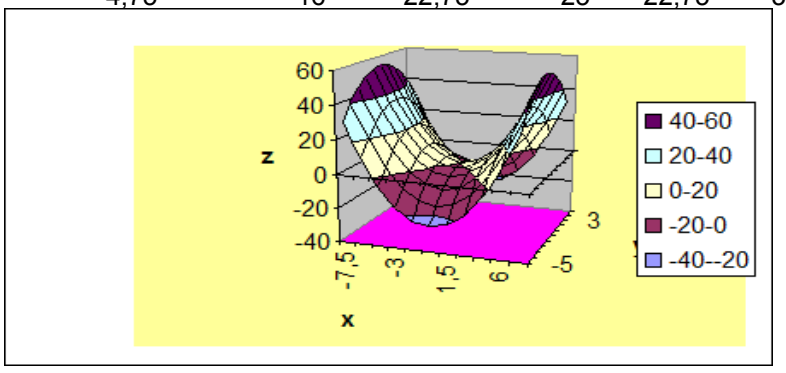


Рис. 6. Гиперболический параболоид

Порядок действий:

- заполнить 1-ю строку значениями X
- заполнить 1-й столбец значениями Y
- B2←=B1^2-A2^2, скопировать формулу по столбцу, а затем по строке (т.к. x берётся из 1-ой строки, а y из 1-ого столбца)
- выделить всю таблицу данных, диапазон A1:L12
- вызвать *Мастер диаграмм*, проделать 4 шага:
  тип — поверхность, ряды данных в строках.

Тема 3. Excel. Организация вычислений

3.1. Приближённое решение нелинейных уравнений. Подбор параметра

Решается уравнение вида f(x)=0. Вопрос о нахождении приближённого решения распадается на 2 этапа:

- нахождение интервала изоляции корня (если он не задан)
- уточнение корня

Пример 3.1. Найти корень уравнения $8x^3 - 17x^2 + 8{,}5x - 8{,}25 = 0$
Это уравнение имеет либо 3, либо 1 действительный корень.

Нахождение интервала изоляции корня:

- в области определения функции выбрать произвольным образом диапазон изменения X, например [0;3] с шагом 0,5. Заполнить значениями столбец X (см. рис.7).
- заполнить значениями столбец Y=f(x). Для этого в ячейку B2 занести формулу B2← = 8*A2^3-17*A2^2+8,5*A2-8,25. Скопировать формулу на диапазон B3:B8.
- выбрать отрезок, на концах которого функция принимает значения разных знаков, это и будет интервал изоляции корня. В нашем случае это отрезок [1,5; 2]. Если такого отрезка нет, то изменить диапазон изменения X.

Уточнение корня

- в качестве начального приближения к корню выбрать значение из отрезка [1,5; 2], пусть x_0=1,7;
  C2 ← 1,7
- вычислить значение функции в этой точке, т.е. $f(x_0)$;
  D2 ← =8*C2^3-17*C2^2+8,5*C2-8,25

- уточнить это значение с помощью команды пункта меню **Сервис → Подбор параметра**

В окне диалога:
Установить в ячейке ввести D2
Значение ввести 0
Изменяя значение ячейки ввести $C$2 (щелчок на ячейке C2), **ОК**

В окне диалога **Результат подбора параметра:**
- приближённое значение к корню находится в ячейке C2, x ≈ 1,851961. в ячейке D2 — погрешность корня 8,8293E-0,6.
- для сохранения результата щёлкнуть **ОК**.

	A	B	C	D
1	**x**	**y**	**корень**	**погрешность**
2	0	-8,25	1,851961	8,8293E-06
3	0,5	-7,25		
4	1	-8,75		
5	1,5	-6,75		
6	2	4,75		
7	2,5	31,75		
8	3	80,25		

Рис.7. Приближённое решение нелинейного уравнения

При подборе параметра используется итерационный процесс, при котором проверяется одно значение за другим, пока не получится нужное решение.

3.2. Аппроксимация (приближение) функций эмпирическими формулами. Построение линии тренда

Линия тренда обычно используется в задачах прогнозирования, она позволяет графически отображать тенденцию данных и прогнозировать их дальнейшие изменения. Этот анализ называется **регрессионным анализом.**

Пример 3.2. **Постановка задачи**. Пусть в результате исследований получены данные, представленные в таблице

месяц	1	2	3	4	5	6	7	8	9	10
продажа ед. товара	250	260	265	300	350	370	380	420	440	500

Необходимо подобрать формулу наилучшего приближения для функции, заданной таблично.

Для решения этой задачи необходимо выполнить следующее:

- в среде Excel построить таблицу исходных данных
- построить график функции, заданной таблично (см. рис.8)
- из контекстного меню графика выбрать **Добавить линию тренда.**

В открывшемся диалоговом окне во вкладке **тип** представлено 6 различных видов линии тренда, которые могут быть добавлены в диаграмму:

линейная	логарифмическая
полиномиальная	степенная
экспоненциальная	скользящее среднее

Мы выбираем вид в зависимости от типа данных. Для нашего случая это **линейная** линия тренда.

На вкладке *Параметры* выбрать

√ **Показать уравнение на диаграмме**

√ **Поместить на диаграмме величину достоверности аппроксимации R^2**

Поле **Прогноз** определяет продолжение линии тренда.

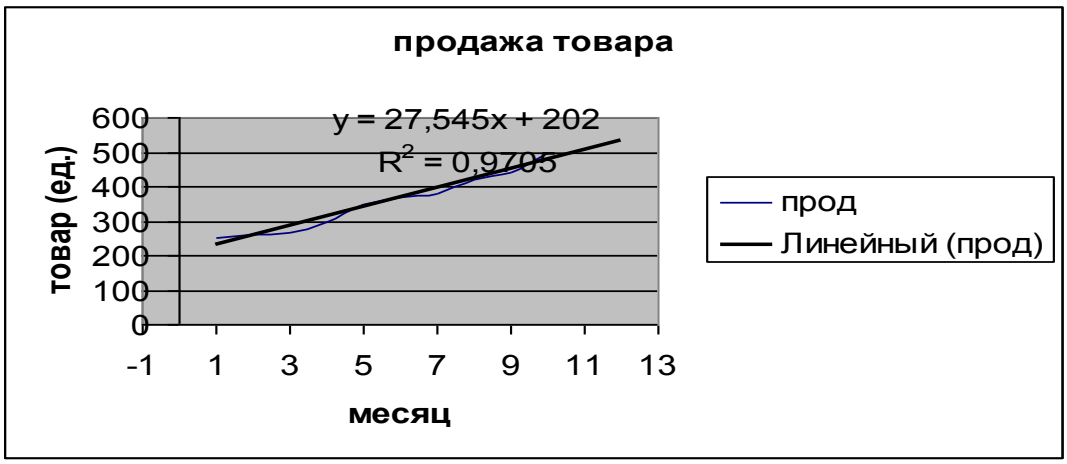


Рис. 8. Построение линии тренда

Линия тренда может быть продолжена за пределы реальных данных для представления будущих значений (см. рис. 8).

R^2 — число от 0 до 1, которое обозначает близость линии тренда к фактическим данным. Таким образом, формула наилучшего приближения для функции заданной таблично имеет вид Y = 27,545x +202.

Редактирование линии тренда может быть осуществлено с помощью контекстного меню.

Тема 4. Разветвляющийся вычислительный процесс

4.1. Функция ЕСЛИ

Функция **ЕСЛИ** используется для проверки условий и может быть вызвана с помощью *Мастера функций* из категории **Логические**.

Общий вид функции ЕСЛИ:

ЕСЛИ (лог. выражение; **значение если истина**; **значение если ложь**)

Эта функция возвращает значение (1), если *лог. выражение* при вычислении дает значение *истина*, и значение (2)— если *ложь*.

Пример 4.1. Вычислить значение функции

$$z = \begin{cases} t\sin x, & \text{если} \quad x \geq 0 \\ \cos y + x, & \text{если} \quad x < 0 \end{cases}$$

$x = 2\alpha, \quad y = 3\alpha, \quad \alpha_н = -3, \quad \alpha_к = 1, \quad \Delta\alpha = 0{,}5; \quad t = 3{,}08$

Построить график зависимости Z(α).

Последовательность действий (см. рис**.** 9**)**:

- ввести значение t;
  заполнить значениями столбец L (один из вариантов заполнения):
  A2 ← -3
  A3 ← =A2+0,5
  Скопировать формулу на диапазон A4:A10 с помощью маркера заполнения;
- присвоить диапазону A2:A10 имя L:
  - выделить A2:A10
  - в поле имени ввести L
- заполнить значениями столбец X:
  - B2 ← = 2*L
  - скопировать формулу на B3:B10
- заполнить значениями столбец Y:
  - C2 ← = 3*L
  - скопировать формулу на C3:C10
- заполнить значениями столбец Z: D2 ← =ЕСЛИ

  В открывшемся диалоговом окне заполнить поля:

  Логическое выражение B2>=0
  Значение, если истина $E$2*sin(B2)
  Значение, если ложь cos(C2)+B2

  нажать **ОК**.

  В строке формул:
  =ЕСЛИ (B2>=0; $E$2*sin(B2); cos(C2)+B2)
- скопировать формулу на диапазон ячеек D2:D10.

Для построения графика зависимости Z(α) необходимо выделить диапазон A1:A10, при нажатой клавише [CTRL] выделить диапазон D1:D10.

Вызвав *Мастер диаграмм* удобно выбрать тип *Точечный*, при этом по умолчанию, ось X будет заполнена значениями α, а ось Y — значениями Z.

	A	B	C	D	E
1	**L**	**X**	**Y**	**Z**	**T**
2	-3	-6	-9	-6,91113	3,08
3	-2,5	-5	-7,5	-4,65336	
4	-2	-4	-6	-3,03983	
5	-1,5	-3	-4,5	-3,2108	
6	-1	-2	-3	-2,98999	
7	-0,5	-1	-1,5	-0,92926	
8	0	0	0	0	
9	0,5	1	1,5	2,591731	
10	1	2	3	2,800636	

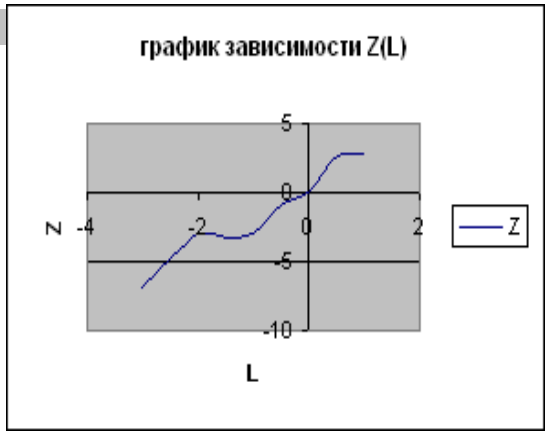


Рис. 9. Разветвляющийся вычислительный процесс. График зависимости Z(α)

4.2. Функции И, ИЛИ

Функции И, ИЛИ как и функция ЕСЛИ используются при проверке условий и могут быть вызваны с помощью *Мастера функций* из категории *Логические*.

Общий вид функции И:

И (логическое выражение1; логическое выражение2; ...)

Функция возвращает значение ИСТИНА, если все аргументы имеют значение ИСТИНА; возвращает значение ЛОЖЬ, если хотя бы один аргумент имеет значение ЛОЖЬ.

Общий вид функции ИЛИ:

ИЛИ (логическое выражние1; логическое выражение2; ...)

Функция возвращает значение ИСТИНА, если хотя бы один из аргументов имеет значение ИСТИНА; возвращает ЛОЖЬ, если все аргументы имеют значение ЛОЖЬ.

Таблица истинности

x	**y**	**и**	**или**
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

1 — истина

0 — ложь

Пример 4.2. математическая запись запись в среде Excel:

$x \in [0; 3]$ И(x>=0; x<=3)

$x<-10 \bigcup x>10$ ИЛИ (x< -10; x>10)

Пример 4.3. Вычислить значение функции

$$z = \begin{cases} \sin x, & x \le -1 \\ x+1, & -1 < x < 1 \\ x^2, & x \ge 1 \end{cases}$$

$x \in [-5; 5], \ \Delta x = 1$

Присвоив диапазону значений *X* имя **x,** формула может быть записана в виде:

ЕСЛИ (x<=-1; sin(x); ЕСЛИ (И(x>-1; x<1); x+1; x^2))

иначе ЕСЛИ (x<=-1; sin(x); ЕСЛИ (x>=1; x^2; x+1))

Тема 5. Одномерные и двумерные массивы

5.1. Матричные функции. Решение системы линейных алгебраических уравнений матричным способом

Матричные функции выбираются с помощью *Мастера функций* из категории *Математические*. К этим функциям относятся:

МОБР(массив)

Возвращает обратную матрицу для матрицы, хранящейся в массиве.

Массив — числовой массив с равным количеством строк и столбцов.

МОПРЕД(массив)

Возвращает определитель матрицы, хранящейся в массиве.

Массив — числовой массив с равным количеством строк и столбцов.

МУМНОЖ(массив1;массив2)

Возвращает произведение матриц (матрицы хранятся в массивах). Результатом является массив с таким же числом строк, как **массив1** и с таким же числом столбцов, как **массив2.**

Массив1, массив2 — перемножаемые массивы.

Пример 5.3. Найти решение системы линейных алгебраических уравнений матричным способом.

$$\begin{cases} 2x_1 + 3x_2 + 7x_3 + 6x_4 = 1 \\ 3x_1 + 5x_2 + 3x_3 + x_4 = 3 \\ 5x_1 + 3x_2 + x_3 + 3x_4 = 4 \\ 3x_1 + 3x_2 + x_3 + 6x_4 = 5 \end{cases} \qquad (1)$$

Система (1) в матричной форме имеет вид:

AX=B, где $A = \begin{pmatrix} 2 & 3 & 7 & 6 \\ 3 & 5 & 3 & 1 \\ 5 & 3 & 1 & 3 \\ 3 & 3 & 1 & 6 \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 5 \end{pmatrix}$, $X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$

Решение системы ищем в виде $X = A^{-1}B$**.**

Порядок действий:

1. Ввод исходных данных
   - в диапазон A3:D6 ввести элементы матрицы A
   - в диапазон E3:E6 ввести элементы вектора B
2. Вычисление обратной матрицы A^{-1}
   - выделить диапазон, в котором будет находиться обратная матрица A11: D14
   - с помощью *Мастера функций* вызвать функцию **МОБР(массив)**
   - в поле *массив* ввести диапазон A3:D6
   - нажать **CTRL+SHIFT+ENTER**

   формула вычисления обратной матрицы: {=МОБР(A3:D6)}
3. Нахождение решения системы
   - выделить диапазон, где будет находиться решение системы G3:G6
   - с помощью *Мастера функций* вызвать функцию **МУМНОЖ**
   - в поле **Массив1** ввести A11:D14
   - в поле **Массив2** ввести E3:E6
   - нажать **CTRL+SHIFT+ENTER**

   формула нахождения решения системы: {= МУМНОЖ(A11:D14;E3:E6)}

	A	B	C	D	E	F	G
1	**Решение системы линейных уравнений**						
2	**матрица A**				**вектор B**		**решение системы**
3	2	3	7	6	1		0,185714286
4	3	5	3	1	3		0,778571429
5	5	3	1	3	4		-0,635714286
6	3	3	1	6	5		0,457142857
7							
8							
9							

10	обратная матрица A^{-1}					проверка	
11	0,028571	-0,12857	0,385714	-0,2		1	
12	-0,12381	0,307143	-0,25476	0,2		3	
13	0,171429	-0,02143	0,064286	-0,2		4	
14	0,019048	-0,08571	-0,07619	0,2		5	

Рис. 10. Решение системы линейных алгебраических уравнений матричным способом

4. В результате вычислений в диапазоне G3:G6 получаем координаты вектора X — это и будет искомое решение системы (1).
5. Сделаем проверку, подставив решение системы в исходное матричное уравнение AX=B. Для этой цели выполним следующие действия:
   - выделить диапазон F11:F14, здесь будет находиться результат AX
   - с помощью *Мастера функций* вызвать функцию **МУМНОЖ**
   - в поле **Массив1** ввести A3:D6
   - в поле **Массив2** ввести G3:G6
   - нажать **CTRL+SHIFT+ENTER**

В результате подстановки найденного решения в исходное уравнение в диапазоне F11:F14 получаем правую часть системы (1).
Формула, по которой осуществляется проверка: {=МУМНОЖ(A3:D6;G3:G6)}

5.2. Статистические функции. Одномерные массивы

Статистические функции выбираются с помощью *Мастера функций* из категории *Статистические*. К этим функциям относятся:

МАКС(число1;число2; ...)

Эта функция возвращает наибольшее значение из своих аргументов. Логические значения и текст игнорируются.

МИН(число1;число2; ...)

Эта функция возвращает наименьшее значение из своих аргументов.

СРЗНАЧ(число1; **число2;** ...)

Возвращает среднее (арифметическое) своих аргументов.

СЧЁТ(значение1; **значение2;** ...)

Подсчитывает количество чисел в списке аргументов.

Значение1, значение2, ... — это от 1 до 30 аргументов, которые могут содержать данные различных типов, но в подсчете участвуют только числа.

СЧЁТЕСЛИ(диапазон; **критерий)**

Подсчитывает количество ячеек внутри диапазона, удовлетворяющих заданному критерию.

СУММЕСЛИ (диапазон; критерий; диапазон суммирования)
(категория *математические*)

Суммирует ячейки, удовлетворяющие заданному критерию.

Диапазон — диапазон проверяемых ячеек

Критерий — критерий в форме числа, выражения или текста, который определяет, какие ячейки надо подсчитывать

Диапазон суммирования — фактические ячейки для суммирования

М.б. иначе: **СУММЕСЛИ (диапазон; критерий)**

Пример 5.1.

	A	B
1	**Стоимость имущества**	**Комиссионные**
2	100 000	7 000
3	200 000	14 000
4	300 000	21 000
5	400 000	28 000
6	**Формула**	**Описание (результат)**
7	СУММЕСЛИ(A2:A5;">160000";B2:B5)	Сумма комиссионных для стоимости имущества более 160000 (63 000)

Пример 5.2. В массиве X = (-3; 16; 24; -0,1;-8) найти

а) количество элементов, принадлежащих отрезку [-5; 5];

Результаты вычислений:

	A	B	C	D	E
1	**массив X**	**[-5;5]**	**количество**	**отриц. эл.**	**сред. ар.**
2	-3	-3	2	-3	
3	16	ЛОЖЬ		ложь	
4	24	ЛОЖЬ		ложь	
5	-0,1	-0,1		-0,1	
6	-8	ЛОЖЬ		-8	

Рис. 11. Нахождение количества элементов, принадлежащих отрезку [-5; 5].

B2 ← =ЕСЛИ(И(A2>=-5;A2<=5);A2), скопировать формулу по столбцу

C2 ← =СЧЁТ(B2:B6)

б) среднее арифметическое отрицательных элементов.

D2←=ЕСЛИ(A2<0;A2),скопировать по столбцу

E2←СРЗНАЧ(D2:D6)

Подключение пакета анализа

Сервис → Надстройки → √ Пакет анализа

Этот пакет содержит функции для анализа научных и финансовых данных. При этом в *Мастер функций* добавляются категории:

- информационные →функции ЕЧЁТН, ЕНЕЧЁТ

- инженерные

Функция ЕЧЁТН(число) возвращает значение истина, если число чётное.
Функция ЕНЕЧЁТ(число) возвращает значение истина, если число нечётное.

5.3. Двумерные массивы

Пример 5.3. Задан двумерный массив A(3,4) (рис. 12). Найти:

1) сумму положительных элементов в каждой строке массива A
Прядок действий:

- E2← =СУММЕСЛИ(A2:D2,”>0”)
- скопировать формулу на диапазон E3:E4

2) среднее арифметическое положительных элементов
Прядок действий:

- сформировать массив из положительных элементов
  A8← =ЕСЛИ(A2>0; A2), скопировать формулу на диапазон A8:D10
- найти среднее значение
  E8← =СРЗНАЧ(A8:D10)

	A	B	C	D	E	F
1	***Исходный массив***				***S***	
2	-2	3	7	0	10	
3	0	5	-6	7,8	12,8	
4	7	-6	5	3	15	
5						
6						
7	**массив из положит. элементов**				**ср.знач.**	**ср.геом.**
8	ЛОЖЬ	3	7	ЛОЖЬ	5,4	5,069029
9	ЛОЖЬ	5	ЛОЖЬ	7,8		
10	7	ЛОЖЬ	5	3		

Рис. 12. Решение задач с двумерным массивом

3) среднее геометрическое положительных элементов
Прядок действий:

- воспользоваться уже полученным массивом из положительных элементов
- найти среднее геометрическое элементов массива
  F8 ← =СРГЕОМ(A8:D10) (категория *статистические*)

Тема 6. Макросы

6.1. Технология построения макросов

Вычисления, которые мы выполняем в Excel можно запомнить, а затем вызвать в нужный момент. Действия запоминаются в виде команд на языке VBA.

Макрос (macro) — это последовательность команд, написанная на языке VBA. Эти команды не пишутся вручную, они получаются как результат работы с мышью и клавиатурой.

Рассмотрим на примере правила и последовательность действий при создании макросов.

Пример 6.1. Создание информационной системы в среде Excel.
Лист 1 переименуем в *Главное меню* (контекст. меню→переименовать)
Лист 2 — в З*адание 1*
Лист 3 — в З*адание 2*
Лист 4 — в З*адание 3*
Лист *Главное меню.* Разработка дизайна (один из вариантов)

ИНФОРМАЦИОННАЯ СИСТЕМА

Задание 1	**Задание 2**	**Задание 3**

Порядок действий:

- с помощью объекта WordArt (панель инструментов *Рисование)* ввести заголовок;
- изобразить объекты (панель инструментов *Рисование*)
- дать название объектам (контекстное меню объекта → **Добавить текст**)

Мы хотим, что бы по щелчку на кнопке осуществлялся переход на соответствующий Лист.

Проектирование макросов

Примем единый принцип создания макросов, который заключается в следующем:

- макрос должен начинаться и заканчиваться со щелчка на одной и той же ячейке, например А1;
- каждое действие пользователя должно завершаться нажатием клавиши **enter;**
- в процессе создания макросов не допускать лишних действий.

1. Создание макроса для перехода на Лист З*адание 1*

- щёлкнуть в ячейке А1
- выбрать пункт меню *Сервис → Макрос → Начать запись*
- в окне *Запись макроса* заполнить поля (обязательным является только поле *Имя макроса*)

Имя макроса ввести *Задание1* (по умолчанию Макрос1)
Описание ввести комментарий, например «Переход на Лист2»

Сочетание клавиш CTRL+A
Сохранить в «эта книга» или может быть «новая книга»
Нажать кнопку *ОК*

Появится плавающая панель инструментов с кнопкой *Остановить запись*. С этого момента начинается запись наших действий в пам'ять.

Действия:
Щёлкнуть на Лист *Задание 1*

2. Остановить макрос

Выполнить команду пункта меню *Сервис* → *Макрос* → *Остановить запись* (или щёлкнуть на кнопку *Остановить запись*)

3. Назначить макрос объекту *Задание 1*

- перейти на Лист *Главное меню*
- контекстное меню объекта → *Назначить макрос*
- выбрать имя *Задание1, ОК*

4. Проверить работу макроса

- щелкнуть на объекте *Задание 1*

возможно иначе, с помощью горячих клавиш CTRL+A
либо через верхнее меню *Сервис* → *Макрос* → *Макросы*, выбрать нужный, щёлкнуть *выполнить*.

Аналогичным образом создать макросы для переходов на Лист *Задание 2* и Лист *Задание 3*, дав им имена *Задание2, Задание3* соответственно.

Лист *Задание 1*. Разработка дизайна
Задание 1. Вычислить значение функции

$$z=\begin{cases} t\sin x, & \text{если} \quad x\ge 0 \\ \cos y + x, & \text{если} \quad x<0 \end{cases}$$

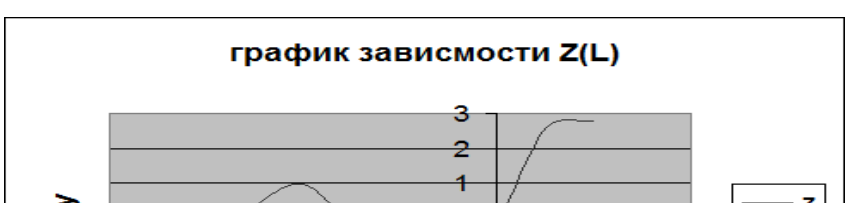
$x=2\alpha, \quad y=3\alpha, \quad \alpha_н=-3; \quad \alpha_к=1; \quad \Delta\alpha=0{,}5; \quad t=3{,}08$

Построить график зависимости $Z(\alpha)$.

	A	B	C	D	E
12	**L**	**t**	**X**	**Y**	**Z**
13	-3	3,08	-6	-9	-6,91113
14	-2,5		-5	-7,5	-4,65336
15	-2		-4	-6	-3,03983
16	-1,5		-3	-4,5	-3,2108
17	-1		-2	-3	-2,98999
18	-0,5		-1	-1,5	-0,92926
19	0		0	0	0
20	0,5		1	1,5	2,591731
21	1		2	3	2,800636

вычислить x
вычислить y
вычислить z и построить график
очистить

Рис. 13. Вычисление значения функции



главное меню

Порядок действий

1. Ввести условие задания, вызвав редактор формул MS Equetion 3.0.
2. Ввести исходные данные (см. рис. 13).
   - заполнить значениями столбец L, ввести значение переменной t
   - присвоить диапазону A13:A21 имя L
3. Изобразить объекты, воспользовавшись панелью инструментов *рисование*. Дать им имена: *вычислить x, вычислить y, вычислить z и построить график, очистить, главное меню*
   - контекстное меню объекта → *добавить текст*
4. Изобразить объект Диаграмма MS Graph
   - выделить диапазон A12:A21, нажать (удерживать) клавишу Ctrl, выделить диапазон E12:E21
   - вызвать *Мастер диаграмм* и выполнить 4 шага

Проектирование макросов

1. **Создание макроса для вычисления значений X**
   - сделать щелчок на ячейке A1
   - выбрать пункт меню *Сервис → Макрос → Начать запись*
   - в окне *Запись макроса* заполнить поля, имя макроса *вычислить x*

Выполнить действия:

- C13 → 2*****L, скопировать формулу по столбцу
- щелкнуть в A1

2. **Остановить макрос**

Выполнить команду пункта меню *Сервис → Макрос → Остановить запись* (или щёлкнуть на кнопку *Остановить запись*)

3. **Назначить макрос объекту** *Вычислить x*
   - из контекстного меню объекта выбрать *Назначить макрос объекту*

4. **Проверить работу макроса**
   - очистить диапазон C13:C21
   - щёлкнуть на объекте *вычислить x*

Макросы для вычисления значений Y, значений Z, очистки диапазона C13:E21, а также макрос для возврата в главное меню создаются аналогично (см. пункты 1, 2, 3, 4), при этом

D13 ← =3*l

E13 ← =ЕСЛИ(C13>=0; $B$13*sin(C13); cos(D13)+C13)

График строится автоматически.

Макросы дают возможность автоматизировать процесс вычислений.

6.2. Чтение программы VB Script

Просмотреть код макроса можно, выполнив команду *Сервис→ Макрос→Макросы*. В диалоговом окне выбрать имя макроса и щёлкнуть кнопку *Изменить***.**

Данное действие вызовет Редактор VB с текстом программы выбранного Макроса.

Код макроса *Задание1* имеет вид:

```
Sub Задание1()
' Задание1 Макрос                ‘комментарий
   Sheets("Задание 1").Select    ‘Лист Задание 1 выбран
End Sub
```

Код макроса *ВычислитьX* имеет вид:

```
Sub ВычислитьX ()
' ВычислитьX Макрос

  Range("C13").Select                ‘выбирается ячейка C13
  ActiveCell.FormulaR1C1 = "=2*L" ‘в выбранную ячейку вводится формула
  Selection.AutoFill Destination:=Range("C13:C21"), Type:=xlFillDefault
  Range("C13:C21").Select            ‘выбирается диапазон C13:C21
  Range("A1").Select                 ‘выбирается ячейка A1
End Sub
```

6.3. Печать документов в Excel

Границы печатной страницы выделены на рабочем листе мелким пунктиром.

Перед печатью рабочего листа следует перейти в режим *Предварительного просмотра* (кнопка *Предварительный просмотр* на панели инструментов Стандартная). В диалоговом окне выбрать вкладку **Страница**, затем сделать выбор на вкладках:

Страница

- ориентация листа (книжная, альбомная)

Лист

- сетка (включить или отключить сетку)
- заголовки строк и столбцов

Поля — позволяет изменить величину полей страницы (верхнее, нижнее, левое, правое).

Для вывода на печать щёлкнуть кнопку *Печать*.

Печать происходит по листам.

Тема 7. Среда VBA (Visual Basic for Application)

*Объектно-ориентированное программирование (**ООП**) является одной из лучших технологий при создания крупных программных проектов.*

7.1. Интерфейс редактора VBA

VBA функционирует только в составе Office приложений, таких как Excel, Word, Access и др. Запуск редактора VBA из среды Excel возможен следующими способами:

- пункт меню **Сервис → Макрос → Редактор Visual Basic**
- с помощью горячих клавиш **[ALT+F11]**
- панель инструментов **Visual Basic → Редактор Visual Basic**
- панель инструментов *Элементы управления* → **исходный текст**

Возвратиться из редактора **Visual Basic** в рабочую книгу можно нажатием кнопки **View Microsoft Excel** на панели инструментов **Standard**.

Основные компоненты **VBA:**

- окно проекта Prodject - VBA Prodject
- окно кода Code
- окно свойств Properties
- окно формы UserForm
- панель инструментов ToolBox

Вызов окон может быть осуществлён с помощью пункта меню **View**.

Excel и VBA могут находиться в двух режимах:

Режим конструктора (дизайна). В этом режиме мы располагаем объекты, редактируем их свойства, пишем программный код. Переход в режим конструктора: панель инструментов *Элементы управления* → п/г

Режим выполнения — это обычный режим, в котором выполняются вычисления.

7.2. Элементы управления. Свойства, события и методы элементов управления

Элементы управления — это видимые объекты. Они расположены
в Excel: п/и *Элементы управления*
в VBA: п/и **ToolBox**. Некоторые из них:

- **Кнопка** (**CommandButton**) – предназначена для инициализации, окончания или прерывания каких-либо действий
- **Поле** (**TextBox**) – предназначено для отображения информации, вводимой пользователем, а также для отображения результатов вычислений
- **Надпись** (**Label**) – предназначена для отображения заголовков или короткого пояснительного текста
- **Рисунок** (**Image**) – предназначен для вывода растровых изображений

- **Полоса прокрутки (ScrollBar)** – позволяет установить числовые значения, основываясь на положении ползунка

Объекты могут быть расположены
в Excel: на рабочем Листе,
в VBA: на **Форме**. **Форма** – это объект **UserForm.**

Добавление Формы в проект:

- перейти в редактор VB
- пункт меню **Insert → UserForm**

Каждый объект имеет свойства (характеристики), события, на которые он реагирует и методы (обработчики событий).

Вызов окна свойств:

- в окне *кода* VBA п/м **View → Properties**
- на Листе Excel п/и *Элементы управления* → **свойства**

Некоторые свойства объектов:

- **Name** – имя объекта
- **Caption** – заголовок объекта
- **BackColor** – цвет объекта
- **ForeColor** – цвет заголовка
- **Font** – шриф

и др.

События и обработчики событий

В VBA предопределены специальные **процедуры обработки событий**. Например процедура обработки события щелчка по объекту:

```
Private Sub CommandButton1_Click()
‘ строки программы, написанные пользователем
End Sub
```

Проект VBA по умолчанию содержит модуль всей книги, а также каждый Лист Excel имеет собственный модуль.

Все объекты, установленные на Листе**,** будут принадлежать данному Листу, а значит их обработчики событий будут реализоваться в модуле данного листа.

В VBA ячейка A2 как объект может быть записана двумя способами
Range (“A2”) или **Cells(2,1)** , где 2 — номер строки
1 — номер столбца

Пример 7.1.

```
Cells(2,1) = 14.6
Range("A1") = Sin(0.5)
Range("A3") = Range(“A1”)*Cells(2,1)^2
```

Оформление внешнего вида ячеек может быть осуществлено посредством свойств **Interior** (интерьер), **Font** (шрифт) и вложенных в них свойств:

Color	цвет
Pattern	узор
Size	размер
Bold	жирный
Italic	курсив

и др.

Пример 7.2.

```
Range("A6") = "текст"
Range("A6").Interior.Color = vbGreen ‘заливка диапазона зелёным цветом
Range("A6").Font.Color = vbBlue ’цвет шрифта голубой
Range("A6").Font.Bold=True ‘шрифт жирный
```

Текст, следующий за символом (‘) игнорируется компилятором и представляет собой комментарий.

7.3. Типы данных.

В VBA имеются следующие базовые типы данных:

Integer	целый
Single	с плавающей точкой обычной точности
Double	с плавающей точкой двойной точности
String	строка
Variant	тип, используемый по умолчанию

Переменная типа Variant – основной тип данных среды VBA

Переменная должна быть объявлена до ее применения.

Оператор объявления переменных **Dim**.

Если переменной не присвоено никакое значение, то она имеет значение нуль. Если тип переменной не задан , то он по умолчанию подразумевается как **Variant**.

Локальная переменная доступна только в той процедуре, в которой она объявлена. Глобальная переменная доступна на урове модуля.

Пример 7.3. Объявление простых переменных

```
Dim a, c  ‘по умолчанию тип Variant
a = 100.5: c = 0
```

Простую переменную типа Variant можно переобъявить в одномерный массив, используя функцию **Array**.

Пример 7.4.

```
Dim a
a = Array(10, 20, 30)  ‘нумерация элементов идёт с нуля
   Range (“A1”)=a(0)  ‘отобразится 10
   Range (“A2”)=a(2)  ‘отобразится 30
```

Пример 7.5. Объявление одномерного массива и присвоение значений элементам массива.

```
Dim A(2) ‘верхнее значение индекса равно 2
A(0)= 3.6 : A(1)= 4.82
A(2)= -12.7
```

Пример 7.6. Объявление двумерного массива

```
Dim B(1,1)
B(0,0)=2:  B(0,1)=4
B(1,0)=1:  B(1,1)=6
```

7.5. Математические функции VBA

Среда VBA предоставляет пользователю большой список встроенных математические функций. Некоторые из них:

функция	**описание**
Abs	модуль (абсолютная величина)
Atn	арктангенс
Cos	косинус
Exp	экспонента, т.е. возведения числа **e** (основание натурального логарифма) в указанную степень
Log	натуральный логарифм
Int	целая часть числа
Rnd	возвращает случайное число из интервала [0;1)
Sin	синус
Sgn	знак числа
Sqr	квадратный корень
Tan	тангенс
Mod	Остаток от деления

Тема 8. Основные конструкции языка VBA

8.1. Операции отношения

При проверке условий могут использоваться**:**

=	равно	>	больше
< >	неравно	>=	больше или равно
<	меньше	<=	меньше или равно

Логические операции:

And (аналог И**), Or (**аналог ИЛИ**), Not**, возвращающие логические значения.

Таблица истиности

x	y	AND	OR	NOT x
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

1 — истина
0 — ложь

Например, принадлежность элемента массива промежутку [-1;1] может быть записана следующим образом:

x(i) >= -1 **And** x(i) < =1

8.2. Разветвляющийся вычислительный процесс. Оператор условного перехода IF

Оператор **IF** задаёт выполнение групп операторов в зависимости от выполнения условия. Существует несколько вариантов этого оператора:

а) **IF** условие **Then** операторы

В данном случае, если условие принимает значение ИСТИНА, то выполняются операторы, следующие за **Then,** в противном случае — оператор, следующий за оператором **IF**.

б) **IF** условие **Then** операторы1 **Else** операторы2

В данном случае, если условие принимает значение ИСТИНА, то выполняются *операторы1*, следующие за **Then,** в противном случае — *операторы2*, следующие за **Else**.

в) блок **IF**

IF условие **Then**
операторы1
[**Else**
операторы2]
End IF

Выполняется аналогично случаю б). Группа [**Else** операторы2] может отсутствовать.

Пример 8.1. Вычислить значение функции

$$z = \begin{cases} a+b, & при \quad a > b \\ a-b, & при \quad a < b \\ a*b, & при \quad a = b \end{cases}$$

Пусть исходные данные a=8**,7** *и* b=5,3 находятся в ячейках листа Excel A1 и B1 соответственно.

Процедура на VBA имеет вид:

```
Sub f1()
 Dim a, b, z
 a = Range("A2"): b = Range("B2")
 If a > b Then z = a + b
 If a < b Then z = a - b
 If a = b Then z = a * b
 Range("C1") = z ‘результат помещается в ячейку C2
End Sub
```

	A	B	C
1	**a**	**в**	**z**
2	**8,7**	**5,3**	**14**

8.3. Ввод данных с клавиатуры. Функция InputBox

Функция **InputBox** выводит на экран окно, содержащее сообщение и поле ввода. Возвращает значение типа *String*. Для преобразования текстового типа в числовой используется функция **Val,** например:

a=val(InputBox(“a”))

Пример 8.2. Найти действительные корни квадратного уравнения $\mathbf{ax^2+bx+c=0}$, $a \neq 0$. Значения **a, в, c** вводятся с клавиатуры, результат выводится в ячейки рабочего Листа.

Порядок действий:

- разместить два объекта Кнопка для запуска проекта и для очистки диапазона, дать им заголовки *Решение уравнения* и *Очистить* соответственно.

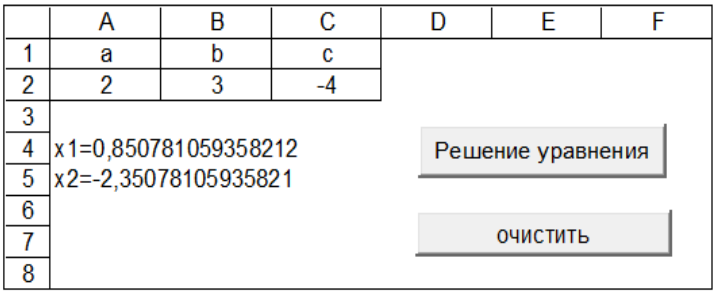


Рис. 15. Интерфейс решения квадратного уравнения

- обработчик события *Щелчок по кнопке Решение уравнения* имеет вид:

```
Private Sub CommandButton1_Click()
Dim a, b, c, x1, x2, d
a = Val(InputBox("a")) ‘перем. a присв. значение, вводимое с клавиатуры
b = Val(InputBox("b"))
c = Val(InputBox("c"))
Range("A2") = a          ‘запись значений переменной в ячейку Листа
Range("B2") = b
Range("C2") = c
d = b ^ 2 - 4 * a * c
If d >= 0 Then
  x1 = (-b + Sqr(d)) / (2 * a)
  x2 = (-b - Sqr(d)) / (2 * a)
  Range("A4") = "x1=" & x1
  Range("A5") = "x2=" & x2
Else
  Range("A4") = "действительных корней нет"
End If
End Sub
```

- обработчик события *Щелчок по кнопке Очистить* имеет вид:

```
Private Sub CommandButton2_Click()
Range("A4:C5").ClearContents
End Sub
```

- вернуться в Excel, выйти из режима конструктора запустить проект на выполнение. Проверить работу проекта щелчками по кнопкам.
  С клавиатуры ввести значения a=2, в=3, с=-4. Результат см. на рис. 15.

8.4. Циклический вычислительный процесс.

Операторы цикла FOR … NEXT

Общий вид:

FOR *переменная* **= A1 TO A2 [step A3]**
[операторы]
NEXT [переменная]

где *переменная* — параметр цикла

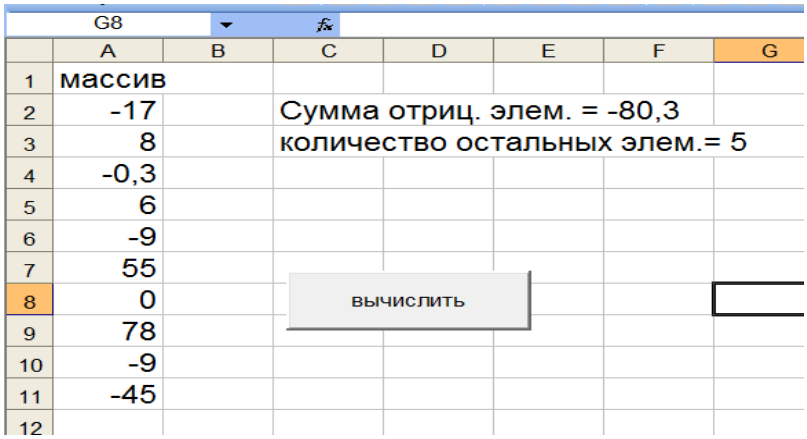
A1 — начальное значение параметра цикла
A2 — конечное значение параметра цикла
A3 — шаг изменения параметра цикла (если шаг равен 1, то может отсутствовать)

8.5. Одномерные массивы

Пример 8.5. Дан массив X=(-17; 8; -0,3; 6; -9; 55; 0; 78; -9: -45). Найти сумму отрицательных элементов и количество остальных.

Порядок действий:

1. в режиме *Конструктор* установить объект CommandButton1, дать ему заголовок.



2. обработчик событий *Щелчок по кнопке Вычислить* имеет вид:

```
Private Sub CommandButton1_Click()
Dim x, k, s, i
x = Array(-17, 8, -0.3, 6, -9, 55, 0, 78, -9, -45)
s = 0: k = 0
For i = 0 To 9
Cells(i + 2, 1) = x(i)
If x(i) < 0 Then
s = s + x(i)
Else
```

```
k = k + 1
End If
Next
Range("c2") = "Сумма отриц. элем. = " & s
Range("c3") = "количество остальных элем.= " & k
End Sub
```

Генерация случайных чисел

Rnd()	возвращает случайное число из интервала [0;1)
100***Rnd()**	возвращает действит. случ. число из диап. [0;100)
100***Rnd()**-50	возвращает действит. случ. число из диап. [-50;50)
Int(6***Rnd()**)+1	возвращает целое случайное число из диапазона [1;6]

Randomize Инициализирует генератор случайных чисел

Пример 8.4. Заполнение диапазона ячеек случайными числами

```
For i = 1 To 10
  Cells(2, i) = 100*Rnd()-50  'случ. числа из [-50;50]
Next
```

Динамические массивы

Если размерность массива в программе не постоянна, то такой массив называется *динамическим*. Динамические массивы объявляются с помощью оператора **ReDim.**

Пример 8.5. Создать процедуру для генерирования одномерного массива целых случайных чисел любой размерности. Найти среднее геометрическое положительных элементов этого массива.

Порядок действий:

- разработать дизайн

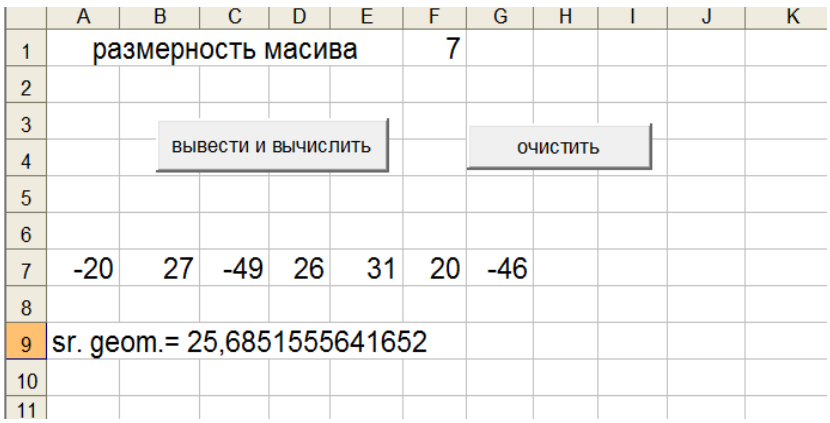


Рис.17. Результат работы проекта примера 8.5.

- в *режим конструктора* установить элементы управления *CommandButton1* и *CommandButton2,* дать им заголовки;
- обработчик события *щелчок по кнопке Вычислить* имеет вид:

```
Private Sub CommandButton1_Click()
Dim p, k, n, i
n = Range("f1")
ReDim x(n - 1)
Randomize
For i = 0 To n - 1
x(i) = int(100 * Rnd() - 50)        ‘диапазон [-50;50]
Cells(7, i + 1) = x(i)
Next
p = 1: k = 0
For i = 0 To n - 1
If x(i) > 0 Then
p = p * x(i): k = k + 1
End If
Next
Range("a9") = "sr. geom.= " & p ^ (1 / k)
End Sub
```

- обработчик события *щелчок по кнопке Очистить* имеет вид:

```
Private Sub CommandButton2_Click()
Range("A7:p7").ClearContents
Range("A9").Clear
End Sub
```

- запустить на выполнение. Результат работы на рис. 17.

8.6. Двумерные масивы

Пример 8.6. Заполнить массив X(3,4) целыми случайными числами, вывести его в ячейки Excel.

```
For i = 0 To 2
For j = 0 To 3
x(i, j) = int(100 * Rnd() - 50)
Cells(i + 1, j + 1) = x(i, j)
Next
Next
```

Пример 8.7. В среде Excel заполнить таблицу размерностью 3*3 целыми случайными числами. Найти сумму элементов в каждой строке.

Порядок действий:

- перейти в *режим конструктора*, установить элементы управления *CommandButton1* и *CommandButton2*, дать им заголовки *Вычислить* и *Очистить* соответственно.
- обработчик события *щелчок по кнопке Вычислить* имеет вид:

```
Private Sub CommandButton1_Click()
Dim i, j, x(2, 2)  ) ‘объявление переменных и массива размерностью 3*3
Randomize
For i = 2 To 4
For j = 2 To 4
Cells(i, j) = Int(100 * Rnd() - 50) ‘целые случ. числа из диап. [-50;50]
Next j
Next i
For i = 2 To 4
  s = 0
For j = 2 To 4
  s = s + Cells(i, j)
Next j
  Cells(i, 5) = s
Next i
End Sub
```

- обработчик события *щелчок по кнопке* Очистить имеет вид:

```
Private Sub CommandButton2_Click()
Range("b2:e4").ClearContents
End Sub
```

Двумерные массивы динамические

Пример 8.7. Создать процедуру для генерирования двумерного массива целых случайных чисел любой размерности. Найти сумму элементов этого массива.

Порядок действий:

- разработать дизайн (см. рис.19)
- перейти в *режим конструктора*, установить элементы управления *CommandButton1* и *CommandButton2* дать им заголовки *вывести и вычислить* и *очистить* соответственно

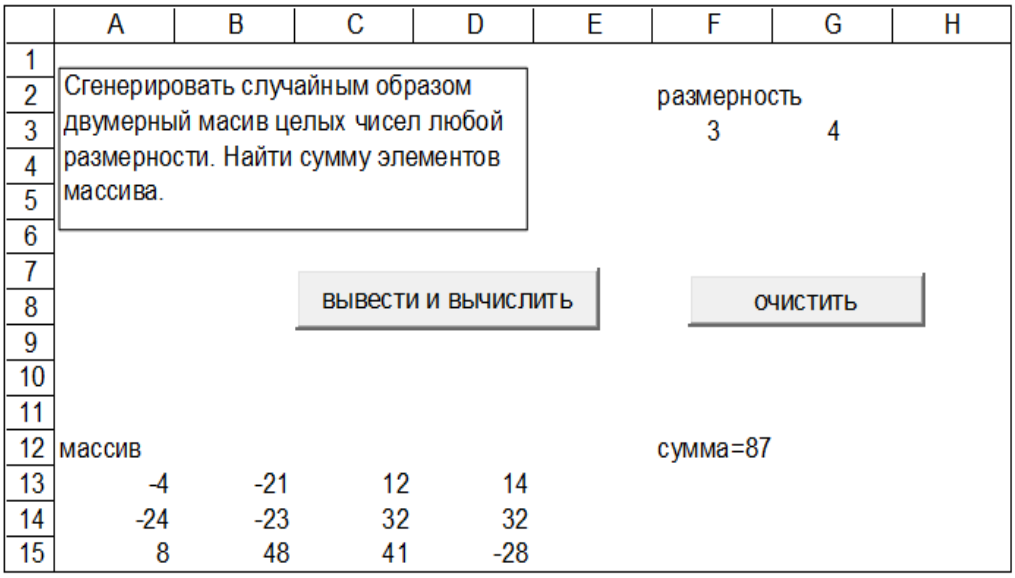


Рис.19. Результат работы проекта примера 8.6.

- обработчик события *щелчок по кнопке вывести и вычислить* имеет вид:

```
Private Sub CommandButton1_Click()
Dim i, j, n, m, s
m = Range("F3")
n = Range("G3")
ReDim x(m-1, n-1)
Randomize
For i = 0 To m - 1
For j = 0 To n - 1
x(i, j) = Int(100 * Rnd() - 50)
Cells(i + 13, j + 1) = x(i, j)
s = s + x(i, j)
Next
Next
Range("F12") = "сумма=" & s
End Sub
```

- обработчик события *щелчок по кнопке Очистить* имеет вид:

```
Private Sub CommandButton2_Click()
Range("a13:D15").ClearContents
Range("f12").Clear
Range("f3:g3").Clear
End Sub
```

- запустить проект на выполнение. Результат работы см. на рис. 19.

• Тема 9. Среда Borland C++ Builder

Borland C++ Builder — это среда программирования, созданная компанией Borland, позволяющая быстро создавать приложения на языке C++. Это среда, в которой может быть осуществлено объектно-ориентированное программирование. Каждый объект имеет свойства (характеристики), события, на которые он реагирует и методы (обработчики событий).

На рынке программных продуктов существует много сред для автоматизации программирования. По мощности среда Builder может соперничать только с Borland Delphi.

Предком языка C++ явился язык C, созданный во 2-ой половине 1970-х годов Денисом Ритчи (США), как язык системного программирования.

В 80-х годах Бьерном Строуструпом создаётся язык C++, как язык объектно-ориентированного программирования. Символами **++** обозначается операция увеличения на единицу, то есть C++ задуман как язык **C** с расширенными возможностями.

Главные составные части среды C++ Builder

Дизайнер Форм (Form1) — это пустая форма, которая автоматически появляется на экране при создании проекта. Она заполняется нужными объектами, выбранными в палитре компонентов.

Палитра компонентов — здесь расположены компоненты среды, используется их постраничная группировка. Существует библиотека визуальных компонентов VCL. Основные визуальные компоненты находятся на страницах Standard, Addition, Win32.

Инспектор объектов (Object Inspector) — это окно, позволяющее увидеть основные свойства и события объекта, помещённого в форму.

Инспектор объектов состоит из двух страниц. Первая страница — это список свойств (*Properties*), вторая — список событий (*Events).*

Окно редактора кода — при создании новой формы, к ней создаётся программный модуль (по умолчанию имя Unit1.cpp) Попасть редактор кода из окна Form1 можно с помощью клавиши [F12].

Проект Builder

Все пользовательские программы в среде Builder оформляются в виде *проектов*. Результатом работы является исполнимый файл (приложение).

Разработка любого проекта начинается с сохранения пустого проекта в новой папке. Для этого выполняются следующие действия:

- пункт меню **File → Save Project As**
  создать новую папку, в которой сохранить два файла:
  - программный модуль: по умолчанию имя **Unit1** (.cpp) → сохранить
  - модуль проекта: по умолчанию имя **Project1** (.bpr). → сохранить.

Во время разработки приложения полезно делать промежуточные сохранения:

пункт меню **File→ Save All** — сохраняются все исходные файлы под текущими именами.

Папки, содержащие файлы проектов, можно перемещать, переименовывать, переносить на другие компьютеры, но файлы, относящиеся непосредственно к проекту переименовывать нельзя.

Состав и структура проекта

Среда Builder связывает с каждым своим приложением следующие файлы:

Unit1.cpp — C++ файл (текст программы). Если появятся новые формы, то для каждой из них будет построен свой программный модуль: Unit2.cpp, Unit3.cpp и т.д.

Unit1.h — модуль с объявлениями компонентов, которые расположены в данной форме. Для каждой новой формы будет построен свой модуль: Unit2.h, Unit3.h, и т.д.

Unit1.dfm — здесь хранится описание формы и всех расположенных в ней компонентов.

Project1.cpp — файл проекта. Он содержит код главной программы, написанной на языке C++. В файле проекта содержатся ссылки на все формы проекта и относящиеся к ним модули.

Содержимое файла Project1.cpp можно просмотреть выполнив команду п. м. **View → Project Source**.

Project1.exe — исполнимый файл (приложение), готовая к выполнению программа, которая может функционировать под управлением операционной системы Windows.

Project1.res — файл ресурсов проекта, представляет собой двоичный файл. В нём хранятся различные значки, графические изображения, используемые в проекте.

Project1.bpr — этот файл отвечает за проведение процесса сборки и компиляции проекта.

Если просмотреть папку, в которой находится весь проект, то обнаружим ещё и другие файлы, например файлы временного хранения ~cpp, ~dfm.

Тема 10. Элементы языка С++. Консольный режим. Линейный вычислительный процесс

10.1. Символы языка

1.1. Прописные буквы латинского алфавита A, B, C, D, …X, Y, Z
Строчные буквы латинского алфавита a, b, c,…x, y, z
Символ подчёркивания _
1.2. Прописные и строчные буквы русского алфавита
Одинаковые прописные и строчные буквы считаются различными символами.
1.3. Арабские цифры 0, 1, 2, …9
1.4. Специальные символы . , ; ( ) < > / \ * = - % и др.
1.5. Управляющие символы, используемые в функциях ввода-вывода:
\n — переход на новую строку
\t — горизонтальная табуляция
\o — нулевой символ
\v — вертикальная табуляция и другие.

10.2. Константы

Различают четыре вида констант:

Целые константы

- *десятичные*: например 16; 240
- *восьмеричные:* в качестве цифр используются символы 0, 1, 2, 3, 4, 5, 6, 7. Восьмеричные константы всегда начинаются с нуля
- *шестнадцатиричные:* в качестве цифр используются 16 символов — 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Шестнадцатиричные константы всегда начинаются с пары символов 0x.

Например:

Шестнадцатиричные константы	*Восьмеричные константы*
0x10	01
0xFFF	065
0x1F1A	0777

Действительные константы (с плавающей точкой)

Например: 3.45; 1.5E-2; -1.85E12; -.56

Символьные константы

Представляют собой символ, заключённый в апострофы. В таблице кодов ASCII каждому символу ставится в соответствие целое положительное число (код), поэтому *значением символьной константы является числовой код символа.*

Примеры символьных констант: ‘Q’; ‘a’; ‘\n’

Строковые константы — это последовательность символов, заключённая в кавычки. Например: “Borland C++”

В конце строковой константы всегда стоит признак конца строки ‘\0’, который формирует компилятор: “ Borland C++‘\0’ ”

10.3. Комментарии

Если текст комментариев занимает одну строку, то используется //……..
Если текст комментариев занимает более одной строки, то используется
/* */

10.4. Типы данных

Особенностью языка С является отсутствие принципа умолчания. Поэтому типы всех переменных должны быть объявлены.

Базовые типы данных

int	целый тип
float	тип с плавающей точкой
double	с плавающей точкой двойной точности
char	символьный тип
void	пустой тип

На основе этих типов строятся другие типы с помощью модификаторов:

short	короткий
long	длинный
unsigned	без знака
signed	знаковый
bool	логический тип

Таблица основных типов данных

ТИП	РАЗМЕР(байт)	ЗНАЧЕНИЯ
bool	1	true, false
int	2	-32768 — 32768
short int	2	-32768 — 32768
long int	4	-2147483648 — 2147483648
unsigned int	2	0 — 65535
float	4	-1,2e-38 — 34e38
double	8	2,2e-308 — 1.8e308
char	1	256 значений символов

Объявление и инициализация переменных

а) **int** a = 24, i = 5; //переменным целого типа **a, i** присваиваются значения

б) **char** c=’c’; //переменной символьного типа **c** присваивается значение

Переменная **c** содержит один символ **c** (точнее его код по таблице кодирования ASCII), тип **char** и **int** взаимозаменяемы.

в) **String** a, b; // объявляются переменные строкового типа **a** и **b**
a = ”Программирование”;
b = ”на C++”;

Переменная любого типа может быть объявлена как неизменяющаяся. Это достигается добавлением зарезервированного слова **const**.
Например: **const double** A=2.12E-2

10.5. Массивы

Объявление массивов:

а) **int** a[30]; //массив из 30 элементов, нумерация с нуля
Это элементы a[0], a[2], …..a[29]. Количество байт в памяти 2*30=60 байт.

б) **double** b[2][3]; //двумерный массив из 6 элементов
Это элементы b[0][0], b[0][1], b[0][2], b[1][0], b[1][1], b[1]][2]]

Инициализация массивов

Одномерный массив

int m[2]={1,8};
иначе **int** m[2];
m[0]=1; m[1]=8;

Двумерный массив $A=\begin{pmatrix} 2 & 1 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

int a[2][3]={{2, 1, 3}, {4, 5, 6}};
возможны варианты **int** a[2][3]={2, 1, 3, 4, 5, 6};
int a[ ][3]={2, 1, 3, 4, 5, 6};

Массив символов

char str[3]={‘a’,’b’,’c’}; //одномерный массив символов

10.6. Арифметические операции

+	сложение	**/**	целочисленное деление
-	вычитание	**%**	остаток от деления
*****	умножение	унарные операции	**+** и **-**

Например:

операция	**результат**
11/3	3
11./3.	3.666….
11%3	2
-x*y	(-x)*y, так как приоритет унарных операций выше, чем у бинарных.

10.7. Математические функции

функция	описание
fabs	модуль (абсолютная величина)
abs	модуль (для целых чисел)
sin	синус
cos	косинус
tan	тангенс
exp	экспонента, т.е. возведения числа **e** (основание натурального логарифма) в указанную степень
log	натуральный логарифм
log10	десятичный логарифм
sqrt	корень квадратный
pow(x, y)	возведение в степень x^y
pow10(x)	возведение в степень 10^x
asin	арксинус
acos	арккосинус
atan	арктангенс
fmode(x, y)	остаток от деления x на y
M_PI	Число π

Все функции кроме **abs** возвращают значение типа **double,** типы аргументов тоже **double**.

Примеры записи арифметических выражений на языке C++ :

$$\frac{e^{2x}+\sin(x-y)^2}{\sqrt{xy}-\ln\frac{x}{2}}$$ (exp(2*x)+sin((x-y)*(x-y))) / (sqrt(x*y)-log(x/2.))

$\sin^2 x$ pow(sin(x), 2)

$\sqrt[3]{x}$ pow(x, 1./3.)

10.8. Функции в C++

Все программы на C++ строятся из функций. Одна часть функций находится в библиотеках. Другая − создаётся самим пользователем (пользовательские функции).

Согласно общему правилу, каждая функция перед использованием должна быть объявлена. Объявления библиотечных функций находятся в заголовочных файлах, которые подключаются к программе с помощью директивы **#include** <имя файла.h>.

Объявления математических функций подключаются с помощью директивы **#include <math.h>.**

10.9. Консольный режим. Возможности ввода-вывода C++

Консольное приложение — это программа на языке C++ в среде Builder, которая запускается без графического интерфейса в консольном окне.

Ввод, идущий с клавиатуры

cin >>

cin — стандартное имя потока ввода

Например:

а) **cin** >> a; // данные вводятся в переменную **a.**

б) **cin** >> i >> j >> s;

Данные с клавиатуры вводятся в три переменные i, j, s.

Вывод, идущий на экран

cout <<

cout — стандартное имя потока вывода.

Например:

а) **cout** <<b; // значение переменной **b** будет выводиться на экран.

Объявления функций *cin, cout* подключаются к проекту с помощью директивы

#include <iostream.h>

10.10. Создание консольного приложения

Работая в консольном режиме, всегда должна быть функция с именем **main** (главная), именно с неё начинается выполнение программы, в каком бы месте она не находилась.

Пример* 1.** Разработать проект для вычисления значения функции ***y при различных значениях ***x.***

$y = a \cdot e^x + \ln c$, где

$c = \sqrt{b + s \cdot \sin\left(\frac{\pi}{4}\right)}$

a = 3,112; b = 5,85; s = 9, 48**;** значения **x** вводятся с клавиатуры.

Порядок действий:

1. создать новый проект, используя пункт меню **File→New**
   в открывшемся окне выбрать **Console Wizard** → OK,
   активизировать переключатель C++, затем **Console Application** → OK

   На экране появится окно **Unit1.cpp** и заготовка для ввода функции:

```
int main(int argc, char* argv[])
{
    return 0;
}
```

2. сохранить проект: пункт меню **File → Save Project As**
   в новой папке сохранить 2 модуля:
   - имя программного модуля по умолчанию **Unit1**
   - имя модуля проекта **Project1**

На этом организационная часть закончена, сформируем модуль Unit1.

3. внести изменения в заготовку, программа на C++ в консольном режиме имеет вид:

```
#include <math.h>            //для мат. функций
#include <iostream.h>        //для cin, cout
#include <conio.h>           //для getch(),clrscr ()
//---------------------------------------------------------------------------
main()
{
double a=3.112, b=5.85, s=9.48;   //инициализация переменных
double  x,y,c;                    //объявление переменных
clrscr ();                        //функция очистки экрана
cout<<"vvedite x"<<endl;  //вывод на экран и спуск на новую строку
cin>>x;                           //ввод с клавиатуры значения x
c= sqrt(b+s*sin(M_PI/4));
y=a*exp(x)+log(c);
cout<<"y="<<y;
getch();
}
```

4. запустить на выполнение RUN [F9]
   ввести с клавиатуры 5.5
   результат y=762,421…
5. сохранить отлаженную программу **File→ Save All**

Функция getch() ждёт ввода с клавиатуры любого символа, делая при этом задержку экрана вывода.

10.11. Стадии прохождения программы

1. *Исходный модуль* (*.cpp) — программа на алгоритмическом языке, передаётся препроцессору, который выполняет директивы, находящиеся в тексте. В текст программы включается содержимое указанных файлов.

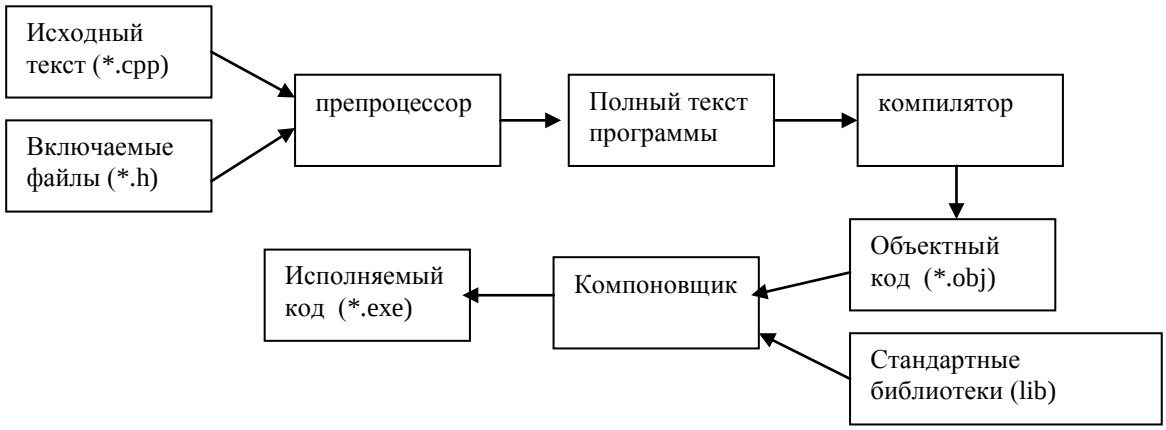
#include <имя файла.h> — поиск файла осуществляется в библиотеке ОС (библиотечные файлы).

#include “имя файла.h” — поиск файла осуществляется в текущем каталоге (функции, созданные пользователем).

Полученный текст передаётся на вход компилятора.

3. Компилятор выявляет синтаксические ошибки и в случае их отсутствия строит *объектный модуль* (*.obj).
4. Компоновщик подключая к объектному модулю другие объектные модули (библиотечные файлы), формирует *исполнимый модуль*.

Исполнимый модуль имеет расширение *.exe — это готовая к исполнению программа.



Тема 11. Windows приложения в графической среде. Линейный вычислительный процесс.

11.1. Функции приведения типов

StrToInt(строка) — преобразование строки в целое число
StrToFloat(строка) — преобразование строки в число с плавающей точкой
IntToStr(целое число) — преобразование целого10-го числа в строковый тип
FloatToStr(число) — преобразование значения с плавающей точкой в строковое представление
IntToHex(целое число) — преобразование целого 10-го числа в 16-е

11.2. Операции со строками

1. Слияние строк. Вывод осуществляется в объект Label1:
   Label1->Caption=”Язык” “– “ “C++”;
   Выведется Язык – C++
2. Слияние строки и числового выражения:
   Label2 ->Caption =” Результат=” + IntToStr(4*6);
   Выведется Результат=24

10.3 Ввод данных с клавиатуры

В графическом режиме ввод данных с клавиатуры осуществляется с помощью функции **InputBox,** которая подключается к проекту с помощью директивы: **#include <Stdlib.h>**

Функция возвращает значение типа **String.**

Общий вид:

InputBox (“сообщение”, “подсказка”, “Default”)

Default — строковое выражение, используемое по умолчанию, если пользователь не введёт строку. В общем случае может быть пробел.

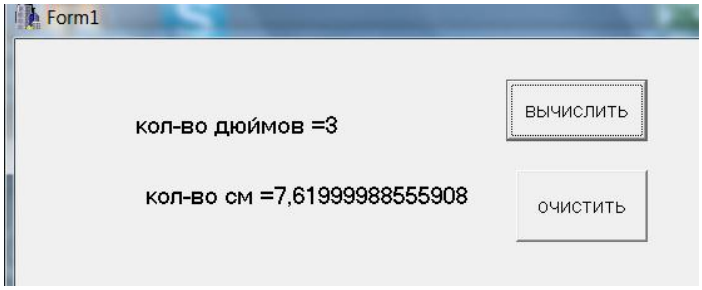
Например, фрагмент кода:

```
{
  float a,b;              // объявление переменных
  a = StrToFloat(InputBox(" введите", " a ", " "));
  b = StrToFloat(InputBox(" введите", " b ", " "));
  Label1->Caption = a+b;
}
```

Пример 11.1. Создать проект для перевода вводимого с клавиатуры количества дюймов в сантиметры, 1дюйм = 2,54см.

Порядок действий:

- создать новый проект, используя пункт меню **File → New Application**
- сохранить проект: пункт меню **File → Save Project As**
- разместить объекты, придать им необходимые свойства в *окне свойств*



На странице **Standard** палитры компонентов:

объект **Label1**— для вывода количества дюймов
объект **Label2** — для вывода количества см
объект **Button1** — для запуска проекта, свойство *Caption* → вычислить
объект **Button2** — для очистки полей вывода, свойство *Caption* → очистить

- обработчик события *щелчок по кнопке вычислить* имеет вид

```
#include <Stdlib.h>
//=========================================
void __fastcall TForm1::Button1Click(TObject *Sender)
{
float a,inch;
inch=StrToFloat(InputBox("vv","kd"," ")); //ввод количества дюймов
a=inch*2.54;                              //количество сантиметров
Label1->Caption="количество дюймов ="+FloatToStr(inch);
Label2->Caption="количество см ="+FloatToStr(a);
}
```

- обработчик события *щелчок по кнопке очистить* имеет вид

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Label1->Caption="";
  Label2->Caption="";
}
```

- запустить программу на компиляцию и выполнение **[F9] (Run)**, в открывшееся окно ввести с клавиатуры 3 (количество дюймов), результат появится в объекте Label2
- сохранить отлаженную программу, выполнив команду **Save All**.

Ввод значений в компонент **Edit** (редактируемое поле, аналог Textbox в VBA)

Внесём изменения в проект:

- установить объект **Edit1** для ввода данных строкового типа (страница *Standard* палитры компонентов)
- внести изменения в код:
  строку inch=StrToFloat(InputBox("vv","kd"," "));
  заменить на inch=StrToFloat(Edit1->Text);
- запустить проект на выполнение *Run*, значения дюймов ввести в объект *Edit1* с клавиатуры (например 3,5), сделать щелчок на кнопке *вычислить*

Пример 11.2. Разработать проект для вычисления значения функции

$$y = a \cdot e^{x} + \ln c$$

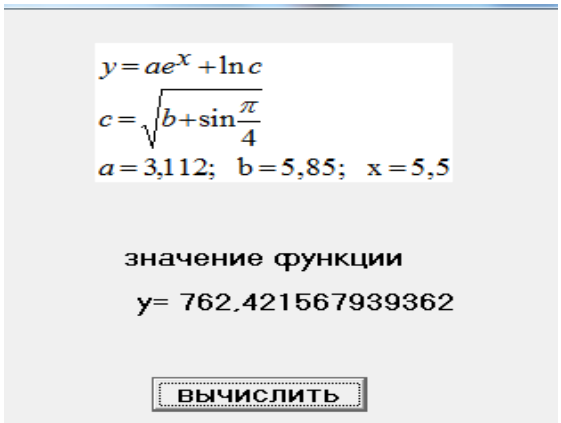
$$c = \sqrt{b + \sin\left(\frac{\pi}{4}\right)}$$, где a = 3,112; b = 5,85; x = 5,5

Порядок действий:

- создать новый проект, сохранить его
- разместить объекты и придать им необходимые свойства:

объект **Label1**— свойство *Caption* →значение функции
объект **Label2** — для вывода результата
объект **Button1** — для запуска проекта, свойство *Caption* → вычислить
объект **Image1** для вставки рисунка (страница **Addition** палитры компонентов)

Рисунок вставляется в формате .bmp в свойство **Picture** объекта Image1.



- обработчик события *щелчок по кнопке вычислить*

```
#include <math.h>
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  double a=3.112, b=5.85, x=5.5; //инициализация переменных
  double  y, c;                  //объявление переменных
  c= sqrt(b+sin(M_PI/4));
  y=a*exp(x)+log(c);
  Label2->Caption="y= "+FloatToStr(y);
}
```

- запуск на выполнение [F9]
- сохранить отлаженную программу, выполнив команду **Save All**.

11.4. Операции отношения

= =	равно	**!=**	неравно
>	больше	>=	больше или равно
<	меньше	<=	меньше или равно

Если две переменные сравниваются с помощью операций отношения, то результат всегда будет логического типа.

Истина (true) —вырабатывается значение отличное от нуля, чаще всего единица.

Ложь (false) — вырабатывается значение равное нулю.

11.5. Логические операции

Логические операции в С++ соответствуют классическим логическим операциям

&& логическое И
|| логическое ИЛИ
! логическое НЕ

Таблица истинности

x	y	x && y	x || y	!x
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

1 — истина
0 — ложь

Приоритет логических операций ниже приоритета операций отношения.
Например запись на языке С++ условия $x \in [0;100]$ имеет вид

x >= 0 && x <=100

11.6. Операция присваивания

Обозначается знаком **=**

1. Выражение вида i = i + 2 может быть записано в виде i+=2
   i = i*2 эквивалентно i*=2
   i=i/10 эквивалентно i/=10
   i=i%10 эквивалентно i%=10
2. Многократное присваивание a = b = c = x*y происходит справа налево

Сначала вычисляется **x*y**, затем его значение присваивается переменной **c**, потом **b**, и лишь затем переменной **a**.

11.7. Преобразование типов

Если в программе используются переменные различных типов, то компилятор автоматически осуществляет преобразования.

Правила преобразований

1. В операторе присваивания тип значения правой части всегда преобразуется в тип значения левой части.

Например

int i = 3.14; // 3.14 преобразуется к целому типу, в результате i =3

2. В арифметическом выражении низший тип всегда преобразуется к высшему, например

float → double, int → long int

3. Любое выражение может быть приведено к желаемому типу с помощью конструкции **(тип) выражение**

Например int x=5;

Значением выражения (float) x/2; будет 2.5

11.8. Операции увеличения и уменьшения на единицу

++ увеличить на единицу

-- уменьшить на еданицу

Пример 11.3**.**

```
int c;
  ++ c; //  значение c увеличивается на единицу, а затем используется
  c++; //  значение c используется, а затем увеличивается на единицу
```

Примр 11.4. Что будет результатом выполнения программы?

```
main () {
int x=5;
 int y=60;
    x++;
    ++y;
    cout<<x<<y;
    cout <<x++<<++y;
}
```

Результат 6 61

6 62

Пример 11.5. Что будет выведено на экран?

```
c=5;
 x=c++;
cout<<x<<c;
```

результат 5 6

Тема 12. Разветвляющийся вычислительный процесс

12.1. Условная операция

Общий вид **e1? e2:e3** , где e1, e2, e3 — выражения

Если **e1** принимает значение истина (то есть ≠ 0), то значением этой конструкции будет **e2**, в противном случае **e3**.

Например, нахождение наибольшее из 2-х чисел **a** и **b** может быть записано виде max = (a > b) ? a : b;

Пустой оператор ;

Этот оператор используется там, где по синтаксису языка требуется оператор, а по смыслу никаких действий не выполняется.

Составной оператор (блок) заключается в фигурные скобки **{ }**, блок эквивалентен одному оператору. **;** после блока не ставится.

12.2 Условный оператор IF

Рассмотрим два варианта оператора IF:

а) **if (условие) оператор1;**

Если условие принимает значение истина (то есть ≠ 0), то выполняется *оператор1*, в противном случае — следующий оператор программы.

б) **if (условие) оператор1;**
else оператор2;

Если условие принимает значение истина (то есть ≠ 0), то выполняется *оператор1*, в противном случае выполняется *оператор2*.

Пример 12.1. Найти действительные корни квадратного уравнения $ax^2 + bx + c = 0$, где a, b, c — заданные числа, $a \neq 0$.
В случае D < 0 вывести сообщение *действительных корней нет.*

Программа в консольном режиме:

```
#include<iostream.h>
#include<conio.h> // для clrscr ()
#include<math.h>
//---------------------------------------------------------
main() {
double a, b, c, d, x1, x2;     //объявление переменных
clrscr ();                     //функция очистки экрана
cout << "  vvedite  a, b, c " << endl;
cin >> a >> b >> c;            //ввод исходных данных с клавиатуры
d=b*b-4*a*c;                          //вычисление дискриминанта
if (d >= 0) {
  x1=(-b + sqrt(d))/( 2*a ); //нахождение корней
  x2=(-b - sqrt(d))/( 2*a );
  cout << "x1=" << x1 << endl;
  cout << "x2=" << x2;
}
   else    cout << "deystvit. korney no";
   getch();
}
```

Пример 12.2. Вычислить значение функции **y** при различных значениях **x**

$$y = \begin{cases} \cos x, & x \le 0 \\ \arcsin x, & 0 < x \le \dfrac{\pi}{2} \\ \log_4 x, & \dfrac{\pi}{2} < x \le 64 \\ \dfrac{1}{x^2}, & x > 64 \end{cases}$$

Программа в консольном режиме:

```
#include<math.h>
#include<conio.h>
#include<iostream.h>
//---------------------------------------------------------------------------
void main() {
double x,y;
  clrscr();
  cin>>x;
  if (x <= 0) y = cos(x);
  if (x > 0 && x <= M_PI/2)   y = asin(x);
  if (x > M_PI/2 && x <= 64)  y=log(x)/log(4);
  if (x > 64)   y=1/pow(x, 2);
  cout << "\n y=" << y;
 getch();
}
```

Запустить программу на компиляцию и выполнение
при x= -3 получим результат y= -.989972
при x = 1 получим результат y= 1.5702

Пример12.3. Создать в графической среде Builder проект для вычисления значения функции **y** при различных значениях **x**

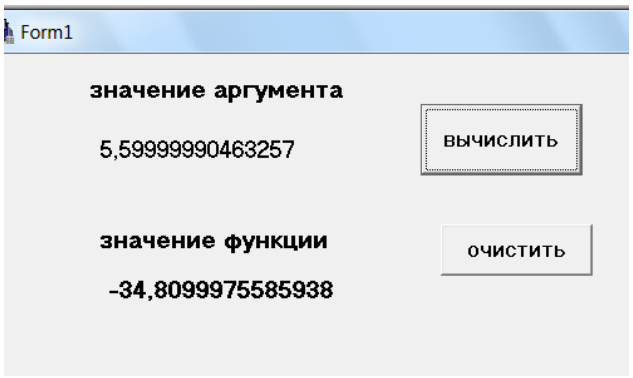
$$y = \begin{cases} a + b - x, & x \ge 1{,}5 \\ \sin^2 x + \cos(a + b), & x < 1{,}5 \end{cases}$$

$a = 1{,}3 - x^2$; b = 0,85

Порядок действий:

- создать новый проект, сохранить его
- разместить объекты, задать им необходимые свойства
  на странице Standard палитры компонентов:
  объект **Label1**, свойство *Caption → значение аргумента*
  объект **Label2** для вывода значений **x**
  объект **Label3**, свойство *Caption → значение функции*

объект **Label4**, для вывода значений **y**
объект **Button1** для запуска проекта, свойство *Caption → вычислить*
объект **Button2** для очистки полей, свойство *Caption → очистить*



- обработчик события *щелчок на объекте вычислить*

```
#include <math.h>
//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float b=0.85;
    float x,y,a;
    x=StrToFloat(InputBox("vvedite","x",""));
    a=1.3-x*x;
    if(x>=1.5) y=a+b-x;
    else
    y=sin(x)*sin(x)+cos(a+b);
    Label2->Caption=x;
    Label4->Caption=y;
}
```

- обработчик события *щелчок на объекте очистить*

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 Label2->Caption="";
 Label4->Caption="";
}
```

- Запустить на выполнение, щелчками по кнопкам проверить работу проекта.

12.3 Оператор выбора switch
(оператор множественных разветвлений)

Синтаксис:

```
switch (выражение) {
   case const1:  последовательность операторов;   break;
   case  const2:  последовательность операторов;   break;
   ……..
   case constN:  последовательность операторов;   break;
   [ default:  последовательность операторов;   break;]
  }
```

Выражение, const1, const2,… — должны быть целого типа.

Правила выполнения оператора:

- вычисляется выражение в заголовке и сравнивается последовательно с const1, const2, … constN. Как только будет найдено соответствие, выполнятся операторы, следующие за двоеточием;
- если соответствие нигде не установлено, то выполнятся операторы, следующие за ключевым словом **default**.

break — выход из оператора *switch* и переход к следующему оператору программы.

Пример 12.4. Программа, обеспечивающая вывод названий дней недели по их номеру.

Консольный режим:

```
#include <conio.h>
#include <iostream.h>
//---------------------------------------------------------------------------
void main() {
int dn;          //dn — день недели
  cout<<"введите номер дня недели";
  cin>> dn;    //в переменную dn  вводится номер дня недели
  switch (dn) {
     case   1 : cout<<"понедельник"; break;
     case   2 : cout<<" вторник"; break;
     case   3 : cout<<" среда"; break;
     case   4  :cout<<" четверг"; break;
     case    5 :cout<<" пятница"; break;
     case    6 :cout<<" суббота"; break;
     case    7 :cout<<" воскресенье "; break;
     default: cout<<" нужно ввести целое число   1-7 " ; break;
   }
    getch();
}
```

Тема 13. Операции над двоичным кодом

13.1. Поразрядное (побитовое) сложение

В вычислительной технике вся информация (числа, текст, графические данные, звук) кодируется двоичным кодом в виде {0; 1}. Коды чисел от 1 до 16 в 10-ой, 2-ой, 16-ой системах счисления представлены в таблице.

Decimal	Binary	Hex
1	0000 0001	01
2	0000 0010	02
3	0000 0011	03
4	0000 0100	04
5	0000 0101	05
6	0000 0110	06
7	0000 0111	07
8	0000 1000	08

Decimal	Binary	Hex
9	0000 1001	09
10	0000 1010	0A
11	0000 1011	0B
12	0000 1100	0C
13	0000 1101	0D
14	0000 1110	0E
15	0000 1111	0F
16	0001 1111	10

Пример 13.1. Сложение двоичных чисел. Переход от двоичной системы счисления в десятичную.

0110 1101 $\quad 2^6 + 2^5 + 2^3 + 2^2 + 2^0 = 109_{10}$

<u>0010 1110</u> $\quad 2^5 + 2^3 + 2^2 + 2^1 = 46_{10}$

1001 1011 $\quad 2^7 + 2^4 + 2^3 + 2^1 + 2^0 = 155_{10}$

Когда оперируют большими числами, то двоичная система счисления неудобна, поэтому переходят к более сжатой системе — 16-ой.
Цифры 16-ой системы счисления: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Например $\quad 0101\ 0001\ 1111_2 \rightarrow$ 0x51F

13.2. Поразрядные логические операции

Поразрядные логические операции выполняются над соответствующими битами целых чисел. Каждый бит имеет значение 0 или 1.
Поразрядными логическими операциями являются:

& логическое умножение (И)
| логическое сложение (ИЛИ)
^ исключающее ИЛИ
~ отрицание
<< сдвиг влево
>> сдвиг вправо

Поразрядные логические операции позволяют обеспечить доступ к каждому биту информации.

Таблица истинности

значения битов		результат операции			
e1	**e2**	**e1 & e2**	**e1 | e2**	**e1 ^e2**	**~e1**
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

1 — истина

0 — ложь

Пример 13.2. Логическое умножение (&).
Результат принимает значение 1, если оба бита имеют значение 1.

В 2-ой системе счисления
1100 0001
0100 0010
0100 0000

В 16-ой системе счисления: 0xF0 & 0xFF → $F0_{16}$

Из примеров видно, что если необходимо установить значение разряда равное нулю, то пользуются операцией &.

Пример 13.3. Логическое сложение (|).
Результат принимает значение 1, если хотя бы один из битов имеет значение 1.

В 2-ой системе счисления
1100 0001
0100 0010
1100 0011

В 16-ой системе счисления 0xF0 | 0xFF → FF_{16}

Если необходимо установить значение разряда равное 1, то пользуются операцией | .

Пример 13.4. Исключающее ИЛИ (^)
Результат принимает значение 1, если значение только одного из битов равно 1 (один и только один).

В 2-ой системе счисления
1100 0001
0100 0010
1000 0011

В 16-ой системе счисления 0xF0 ^ 0xFF → $0F_{16}$

Пример 13.5. Логическое отрицание (~).
В 2-ой системе счисления: ~0100 0001 → 1011 1110

13.3. Таблица приоритетов операций

операция	назначение
++ --	увеличение (уменьшение) на 1
~	побитовое НЕ
!	логическое НЕ
+ -	унарный +, унарный -
( )	приведение типа
* / %	умножение, деление, вычисление остатка
+ -	сложение, вычитание
>> <<	сдвиги
< <= > >= == !=	операции отношения
&	побитовое И
^	побитовое исключающее ИЛИ
|	побитовое ИЛИ
&&	логическое И
||	логическое ИЛИ
?:	условная операция
= += *=	присваивание

13.4. Поразрядные сдвиги

Операции сдвига применяются только к целочисленным переменным.

При использовании операций сдвига может происходить потеря старших или младших разрядов, недостающие значения битов при этом дополняются нулями, значения переменных изменяются.

Общий вид операции: **переменная >> число позиций**

Все биты переменной сдвигаются вправо на указанное число позиций (SHR — общепринятое обозначение).

переменная << число позиций

В данном случае все биты переменной сдвигаются влево на указанное число позиций (SHL — общепринятое обозначение).

Пример 13.6. Левый сдвиг на 1 позицию

до сдвига $0010\ 0000 \rightarrow 2^5 = 32_{10}$

после сдвига $0100\ 0000 \rightarrow 2^6 = 64_{10}$

Левый сдвиг на 1 разряд умножает число на 2.

Пример 13.7. Правый сдвиг на 1 позицию

до сдвига $0010\ 0000 \rightarrow 2^5 = 32_{10}$

после сдвига $0001\ 0000 \rightarrow 2^4 = 16_{10}$

Правый сдвиг на 1 разряд делит число на 2.

Пример 13.8. Фрагмент программы на C++

```
unsigned char bits =9; // 00001001₂
      bits=bits << 3; // 0100 1000₂
      bits=bits >> 5; // 0000 0010₂
```

Тема 14. Циклический вычислительный процесс. Генерация случайных чисел. Одномерные массивы

Существует три вида взаимозаменяемых операторов цикла: **for, while, do…while**

14.1. Оператор цикла FOR

Общий вид оператора:

for (выражение1; выражение2; выражение3) тело цикла;

выражение1 — задаёт начальное значение параметру цикла
выражение2 — задаёт условие продолжения цикла
выражение3 — задаёт изменение параметра цикла
тело цикла — может быть либо простой, либо составной оператор

Алгоритм работы цикла:

1. присваивается значение параметру цикла
2. проверяется условие продолжения цикла. Если условие принимает значение *истина* (≠0), то выполняется тело цикла и переход на п.1, в противном случае выполняется оператор, следующий за оператором **for.**

Таким образом, перед каждым повторением цикла происходит изменение параметра цикла и проверка условия.

Пример 14.1. for (i= 0; i < 10; i ++) cout << i << "\n";
В результате работы цикла в столбец выведутся цифры от 0 до 9.

Пример 14.2. for (i = 9; i >= 0; i --) cout << i << "\n";
В данном случае в столбец выведутся цифры от 9 до 0.

Пример 14.3. Примеры бесконечных циклов (могут быть получены случайно).
for (i = 1; 1; i ++) оператор;
for (i=10; i>6; i++) оператор;
Оператор ***for*** удобен при организации циклов, когда количество повторений известно.

Пример 14.4. Вычислить значение функции

$$z = \begin{cases} ax^2 + \lambda, & x > \lambda \\ ay + x, & x \le \lambda \end{cases}$$

$$x = 3\lambda + \cos\lambda$$

$$y = 2\sin\lambda, \quad \alpha = 3{,}0; \quad \lambda \in [-5;5]; \quad \Delta\lambda = 0{,}5$$

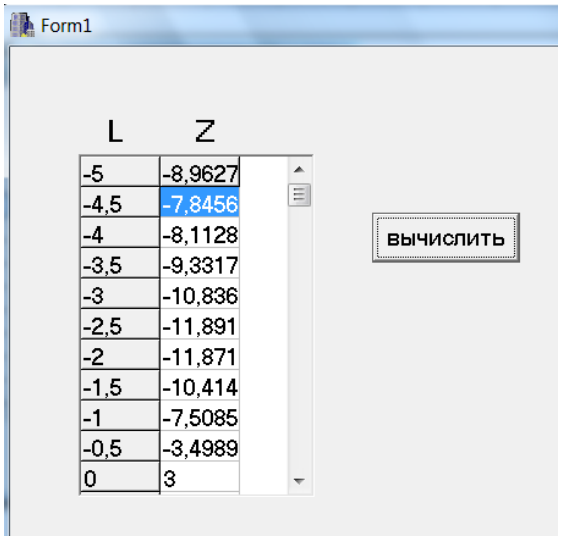
Результат вывести в виде таблицы

L	**Z**
.....	.....

Порядок действий (графический режим):

- создать и сохранить проект
- для вывода результатов в виде таблицы необходимо установить на форме компонент **StringGrid** (страница **Additional**). Определить количество строк в таблице по формуле $\left[\frac{\lambda_к - \lambda_н}{\Delta\lambda}\right] + 1 = 21$
- задать свойства компонента *StringGrid:*
  ColCount → 2 (количество столбцов)
  RowCount → 21 (количество строк)
- установить в форму два компонента Label, дать им заголовки **L** и **Z** соответственно
- установить компонент **Button** для запуска проекта, свойство *Caption* → *вычислить*

Интерфейс:



Обработчик события *щелчок по кнопке вычислить* имеет вид:

```
  #include<math.h>
void __fast call TForm1::Button1Click(TObject *Sender)
{
  float x, y, l, z, a=3.0;              //объявление переменных
  int i = 0;                        //задание начального номера строки
  for (l = -5; l <= 5; l += 0.5)   {
    x=3*l+cos(l);
    y=2*sin(l);
    if (x > l) z = a*x*x+l;
    else
    z = a*y+x;
    StringGrid1->Cells[0][i]=l;          //Cells[Col][Row]
    StringGrid1->Cells[1][i]=z;          //вывод результатов в таблицу
    i++;                                 // изменение номера строки в таблице
  }
}
```

Все результаты вычислений могут быть просмотрены с помощью вертикальной линейки компонента StringGrid.

14.2. Область действия переменных

Переменная должна быть объявлена до её использования. Существуют переменные локальные (внутренние) и глобальные.

Локальная переменная существует только в том блоке, в котором объявлена. При выходе из блока эта переменная и её значение теряются.

Область действия локальной переменной — блок.

Например:

```
for ( int i=0; i<10; i++)
{тело цикла}
 i=0;
```

Первая переменная *i* известна только в цикле *for* , а в выражении *i=0***;** это будет уже другая переменная, то есть ей компилятор присвоит совсем другой адрес.

Глобальная переменная — это переменная, объявленная вне какой-либо функции и может быть использована в любом месте программы.

Область действия глобальной переменной — вся программа.

Например:
```
int a;
int main() {
}
```
Совет: переменная должна иметь минимальную из возможных областей действия.

14.3. Вычисление суммы (произведения) ряда

Пример 14.5. Вычислить n!=1*2*3*…*n при различных значениях n.

Консольный режим.

0!=1; 1!=1; n!→fact

```
#include <iostream.h>
#include <conio.h>
//---------------------------------------------------------------------------
void main() {
int n;
long int fact;
  cout<<"vvedite  n"<<endl;
  cin>>n;
  if (n==0 || n==1) fact=1;
  else  {
     fact=1;
     for (int i=1; i<=n; i++)
      fact*=i;
  }
  cout<<n<<"!="<<fact;
  getch();
}
```

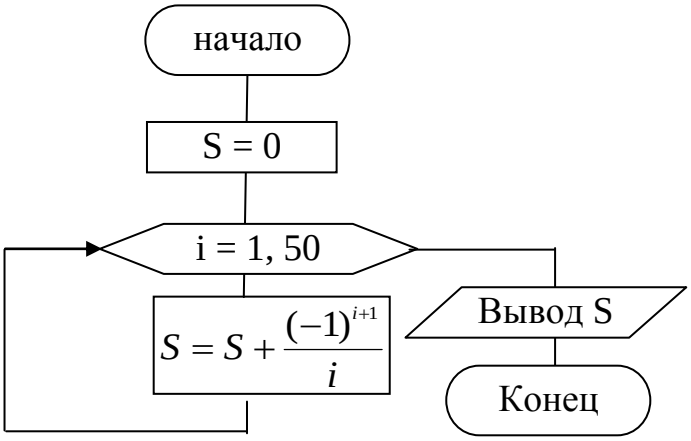
Запустить на выполнение, ввести n=10.

Результат n!=3628800.

Пример 14.6. Вычислить сумму ряда

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{1}{49}-\frac{1}{50}$$

Блок-схема алгоритма:



Программа в консольном режиме имеет вид:

```
#include<iostream.h>
#include<conio.h>
//---------------------------------------------------------------------------
void main() {
double s=0;
   for (int i=1; i<=50; i++)
   s+=pow(-1, i+1) / i;
   cout<<"s="<<s;
   getch();
}
Результат S=0,683247
```

14.4. Оператор цикла WHILE

Общий вид:

while (условие продолжения цикла) тело цикла;

Тело цикла — простой оператор или составной.

Цикл выполняется до тех пор, пока условие принимает значение *истина* (≠0). Если условие пронимает значение *ложь*, то управление передаётся следующему оператору программы.

Так же как и в цикле *FOR* сначала проверяется условие, а затем выполняется тело цикла. Это циклы с предусловием

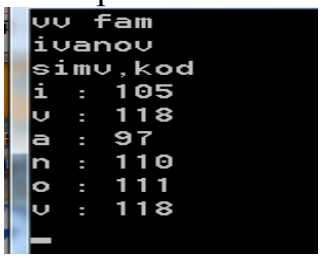
Пример 14.7. Для заданной строки (фамилия пользователя, заданная латинскими буквами) отобразить ASC коды каждого символа.

В С++ строка − это массив символов. Признак конца строки ‘\0’. Нулевой байт автоматически добавляется компилятором.

Алгоритм решения задачи. Консольный режим. name — фамилия

```
#include<iostream.h>
#include<conio.h>
void main() {
char name [10];              //длина массива с учётом '\0'
      cout << "введите фамилию "<<endl;
      cin >> name;
      cout << " символы фамилии и соответствующие им коды "<<"\n";
 int i=0;                 //начальное значение параметра цикла
       while (name[i]!='\0')  {
           cout <<name[i]<<":"<< int(name[i])<<"\n";
           i++;                 //изменение параметра цикла
        }
       getch();
}
```

Цикл будет работать до тех пор, пока не встретится признак конца строки '\0'.
Результат работы:



14.5. Оператор цикла DO…WHILE

Общий вид:

do
тело цикла
while (условие продолжения цикла **);**

Это цикл с постусловием, то есть вначале выполняется тело цикла, а потом оценивается условие продолжения цикла. Если в результате оценки получается значение *истина*, то цикл повторяется. Если получено значение *ложь*, то цикл завершается.

Таким образом, цикл выполнится хотя бы один раз.

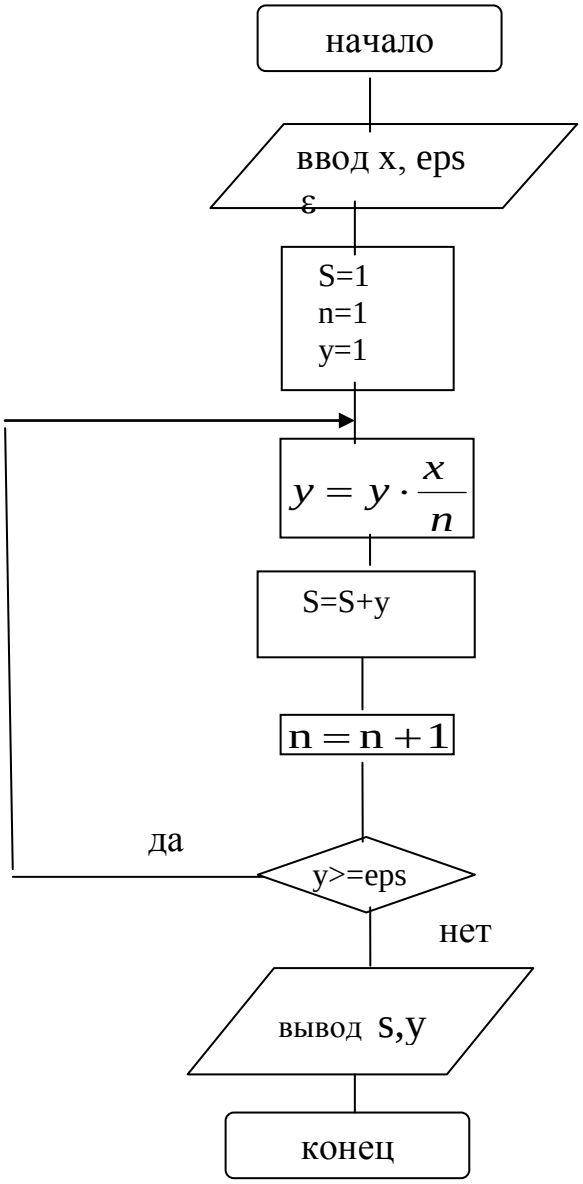
Пример 14.6. Вычислить сумму членов бесконечного ряда

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots + = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

с точностью до ε, где ε – заданная сколь угодно малая величина.

Вычисления проводить до тех пор, пока не выполнится условие $\frac{x^n}{n!} \le \varepsilon$.

Блок-схема алгоритма:



Алгоритм решения в консольном режиме:

```
#include<conio.h>
#include<iostream.h>
//-----------------------------------------
void main() {
float x, s, eps, y=1;
int n=1;
   cout <<"\n vvedite  x i eps\n" ;
   cin>>x>>eps;
   s=1;
 do  {
    y=y*x/n;
    s+=y;
```

```
    n+=1;
  }
  while(y>=eps);
  cout<<"summa="<<s<<endl;
  cout<<"element ryada="<<y;
   getch();
}
```

Результаты вычислений:

при x=2,5; ε =0,001 summa=12.1823
element ryada=0.000597

14.6. Генерация случайных чисел

Функция **rand()** генерирует целое число в диапазоне между **0** и символьной константой **RAND_MAX,** определённой в заголовочном файле **<stdlib.h>**. Для того чтобы выбрать целые числа в конкретном диапазоне, используется операция вычисления остатка **%.**

Например: **rand() % 6** целые числа в диапазоне от 0 до 5
1+rand() % 6 целые числа в диапазоне от 1 до 6
rand() % 9 - 4 целые числа в диапазоне от -4 до 4

Совместно с функцией **rand** используется библиотечная функция **srand**, которая задаёт точку входа в таблицу случайных чисел.

Например **srand (time (NULL)); (**м.б. randomize - аналог VBA**)**

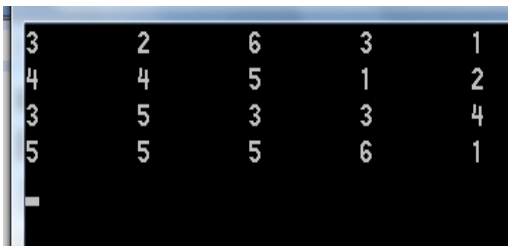
Функция **time** возвращает текущее время в секундах.

В результате, при каждом запуске программы генерируются различные числа.

Пример 14.7. Моделирование бросания игральной кости. Вывести 20 целых случайных чисел из диапазона [1;6] по 5 в строке.

```
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
void main() {
    srand ( time (NULL) );              //вход в таблицу случайных чисел
    for ( int i = 1; i <= 20 ; i++) {   //цикл работает 20 раз
       cout << 1+rand() % 6 << ”\t”; //вывод случайных чисел  в строку
       if ( i%5 = =0 )                  //числа выводятся по 5 в строку
       cout<<end1;                              //спуск на новую строку
    }
     getch();
}
```

Результат работы программы:



14.7. Одномерные массивы

Пример 14.8. Ввод массива A(10) с клавиатуры и вывод на экран:

```
for (i=0; i<10; i++) cin>>a[i];
for (i=0;I <10; i++) cout << a[i];
```

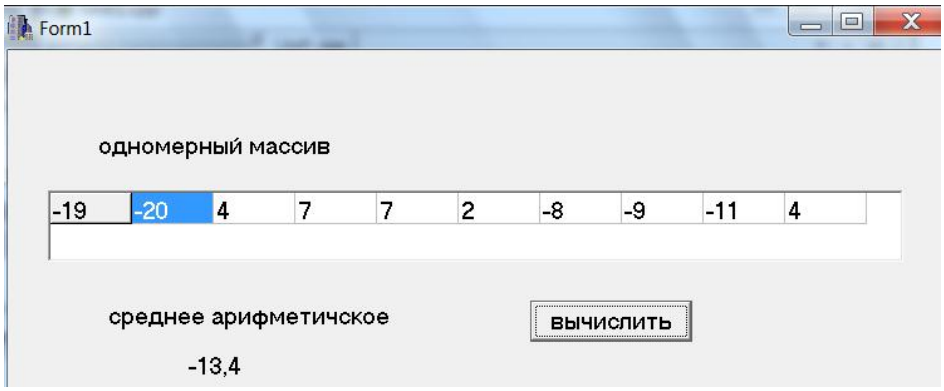
Пример 14.9. Сгенерировать одномерный массив целых случайных чисел размерностью 7 в диапазоне [-10;10]. Графический режим:

```
srand(time(NULL));
for (i=0; i<7; i++) {
    x[i]=rand()%21-10;    //заполнение массива случайными числами
    StringGrid1->Cells[i][0] =x[i];      //вывод массива в строку
 }
```

Пример 14.10. Заполнить одномерный массив целыми случайными числами из диапазона [-20;20] размерностью 10. Найти среднее арифметическое отрицательных элементов массива.

Реализация алгоритма в графической среде. Порядок действий:

- создать и сохранить проект



- разместить объекты, задать им необходимые свойства:
  - ✓ объект **Label1**, свойство *Caption* → *одномерный массив*
  - ✓ объект **StringGrid1** (страница **Addition**)
    свойства **ColCount** → 10 (количество столбцов)
    RowCount → 1 (количество строк);
  - ✓ объект **Button1** для запуска проекта, свойство *Caption* → *вычислить*

- ✓ объект **Label2,** свойство *Caption* → с*реднее арифметическое*
- ✓ объект **Label3** для вывода результата

- обработчик события *щелчок по кнопке вычислить* имеет вид*:*

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int x[10], sum=0, k=0;
  double sa;
     srand(time(NULL));
     for (int i=0;i<=9;i++) {
          x[i]=rand()%41-20;        //заполнение массива случайными числами
          StringGrid1->Cells[i][0] =x[i]; //вывод массива
          if(x[i]<0) {
             sum+=x[i];
             k++;
           }
      }
  sa=(double)sum/(double)k;       //нахождение среднего арифметического
  Label3->Caption=sa;
  }
```

14.6 Динамическое распределение памяти

Если размерность массива задаётся во время выполнения программы, например введена с клавиатуры, то такие массивы называются *динамическими*. Память под них выделяется с помощью оператора **new.**

Например, пусть arr — имя массива,

*arr — адрес массива,

n— размерность массива (количество элементов).

тогда

```
int n=9;       //или cin>>n
int *arr=new int[n];
```

В данном случае, по адресу массива с именем ***arr*** размерностью **n**, оператором **new** будет выделена непрерывная область памяти. Памяти выделится столько, сколько необходимо для хранения *n* величин типа *int.*

Пример 14.11. Сгенерировать случайным образом массив целых чисел любой размерности. Найти сумму элементов с номерами кратными 3 и сумму остальных элементов.

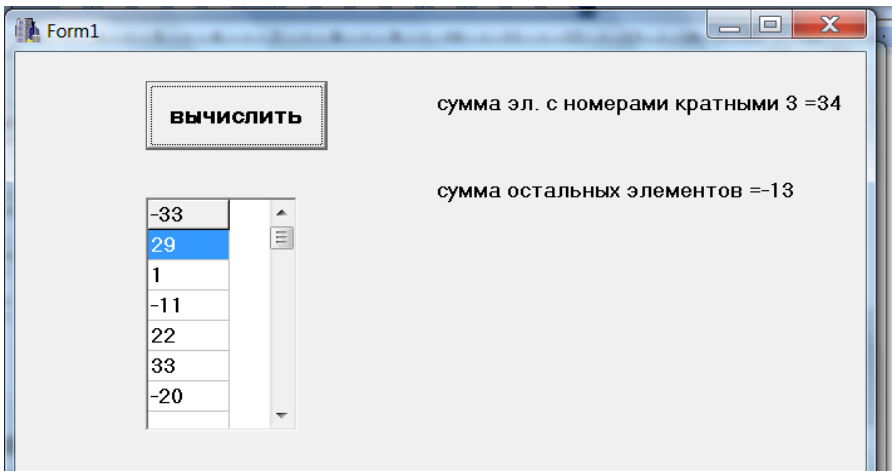
Порядок действий:

- установить

  объекты Label1, Label2 для вывода результатов

  объект Button1 для запуска проекта, свойство *Caption→вычислить*

  объект StringGrid1 для вывода элементов массива



- обработчик события *щелчок по кнопке вычислить* имеет вид:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  float s1=0, s2=0;
  int size;
  size=StrToFloat(InputBox("vv","size","")) ;        //ввод размерности массива
  int*arr=new int[size];                          //динамическое выделение памяти
  srand (time(NULL));
  for ( int i=0;i<size;i++) {
    arr[i]=rand()%100-50;          //заполнение массива случайными числами
    StringGrid1->Cells[0][i]=arr[i]; //вывод массива
  }
    for (int i=0; i<size ;i++)
    if((i+1) %3==0 )
    s1=s1+arr[i];
    else
    s2=s2+arr[i];
    Label1->Caption="сумма элем. с ном. кратными   3 ="+FloatToStr(s1);
    Label2->Caption="сумма остальных элементов="+FloatToStr(s2);
}
```

- запустить проект на выполнение. Ввести с клавиатуры размерность массива, например 7. Результат см. на Форме.

Тема 15. Двумерные массивы

Пример 15.1. Заполнить двумерный массив A(4,5) случайными целыми числами из диапазона [0; 10].

```
int a[4][5];
for ( i = 0; i<4; i++)
for ( j = 0; j<5; j++)
a[i][j] = rand() % 11;
```

Пример 15.2. Ввод двумерного массива A(2,3) с клавиатуры. Консольный режим.

```
double a[2][3];
for ( int i=0; i<2; i++)
for ( int j=0; j<3; j++)
cin >> a[i][j];
```

Пример 15.3. Вывод двумерного массива A(4,5) по строкам. Консольный режим.

```
for ( i = 0; i<4; i++)
{
for ( j = 0; j<5; j++)
cout<<a[i][j];
cout<<end1;          //спуск на новую строку
}
```

Пример 15.4. Найти сумму положительных элементов двумерного массива

$$A=\begin{pmatrix}1 & 0 & 3\\ 5 & -4 & 9\end{pmatrix}$$

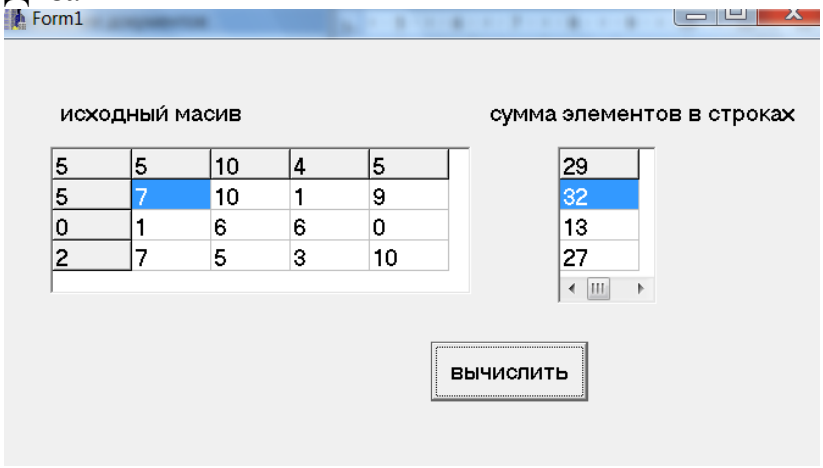
Алгоритм решения задачи. Консольный режим.

```
#include <iostream.h>
#include <conio.h>
//--------------------------------------------------------------------------
void main() {
int a[2][3]={{1,0,3},{5,-4,9}};
int s=0;
  for (int i=0; i<2; i++)
  for (int j=0; j<3; j++)
  if (a[i][j]>0)  s+=a[i][j];
  cout <<"s="<<s<<"\n";
getch();
}
```

Пример 15.5. Заполнить двумерный массив A(4,5) случайными целыми числами из диапазона [-10;10]. Найти сумму элементов в каждой строке.

Создать проект в графической среде.

Порядок действий:
Дизайн



- разместить объекты, задать им необходимые свойства
  - ✓ объект **Label1,** свойство *Caption* → *исходный массив*
  - ✓ объект **Label2,** свойство *Caption* → *сумма элементов в строках*
  - ✓ объект **StringGrid1** для вывода элементов массива виде таблицы: свойства **ColCount** → 5 (количество столбцов)
    RowCount → 4 (количество строк);
  - ✓ объект **StringGrid2** для вывода результатов:
    свойства **ColCount** → 1
    RowCount → 4
  - ✓ объект **Button1** для запуска проекта, свойство *Caption*→ *вычислить*

- обработчик события *щелчок по кнопке вычислить* имеет вид:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 int s;
 const int nrow=4, ncol=5;          //размерность массива
 srand(time(NULL));          //инициализация генератора случайных чисел
 for (int i=0; i<nrow; i++)
 for (int j=0; j<ncol; j++)   {
  a[i][j]=rand()%21-10;              //заполнение массива случайными числами
  StringGrid1->Cells[j][i]=a[i][j]; //заполнение таблицы случайными числами
  }
  for (int i=0; i<nrow; i++)  {
    s=0;
    for (int j=0;j<ncol; j++)
    s+=a[i][j];                          //накопление суммы в строке
    StringGrid2->Cells[0][i]=s;       //вывод результатов в таблицу
   }
}
```

- запустить на выполнение RUN

Размерность массива задаётся именованными константами, что позволяет их легко изменять.

Тема 16. Функции пользователя в C++. Рекурсивные функции

С использованием функции связано три понятия:

- описание функции — описание действий, которые выполняет функция
- объявление функции
- обращение к функции

Описание функции

[тип] имя функции ([список формальных параметров]) {
тело функции
}

Первая строка **— заголовок функции.**

Тип — это тип возвращаемого значения. Если тип не задан, то по умолчанию подразумевается *int*. Если функция не возвращает значение, то тип *void*.

Формальные параметры — это переменные, используемые внутри тела функции. Они получают значения при вызове функции путём копирования в них значений соответствующих фактических параметров.

Для каждого формального параметра должен быть указан тип.

Оператор return

Общий вид оператора: **return [выражение];**

Этот оператор обеспечивает:

- выход из функции и возврат результата вычислений в вызывающую программу (в точку вызова).

В теле функции может быть несколько операторов *return* или ни одного.

Пример 16.1. Функция пользователя для нахождения наибольшего из 2-х целых чисел. Возможны варианты:

а)
```
max (int a, int b)  {
  if (a>b) return a;
  else return b;
 }
```

б)
```
max (int a, int b) {
  return (a>b)? a:b;
 }
```

Если оператор *return* отсутствует, то возврат в вызывающую программу происходит после выполнения последнего оператора тела функции, а возвращаемое значение не определено.

Объявление функции

До первого вызова функции *необходимо сообщить тип возвращаемого результата, а так же количество и типы аргументов*.

Объявление функции совпадает с заголовком функции:

[тип] имя функции ([список формальных параметров]);

Объявление функции может иметь вид

```
max (int a, int b);      //совпадает с заголовком в описании функции
```

или

```
max (int , int );   /* имена формальных параметров не играют никакой роли и игнорируются компилятором*/
```

Обращение к функции:

имя функции ([список фактических параметров])

Пример 16.2. Создать функцию для вычисления квадрата числа. Воспользоваться ею при вычислении квадратов чисел из диапазона [-3; 1] с шагом 0,5.

Пусть kv — имя функции, a — *формальный параметр*

Описание функции:

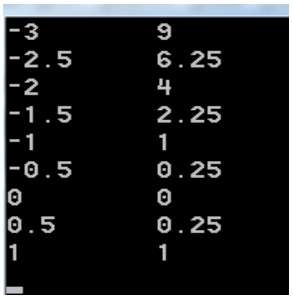
```
float kv(float a)  {
  return a*a;
}
```

Алгоритм решения задачи. Консольный режим.

```
#include <iostream.h>
#include <conio.h>
//---------------------------------------------------------------------------
float kv(float a);                    //это объявление функции
main () {                             //это главная функция
float b;
  for (b=-3; b<=1;b+=0.5)
  cout << b << "\t " <<kv(b)<<"\n"; //b-фактический параметр
  getch();
}
float kv(float a)  {          // это описание функции
  return a*a;
}
```

В данном примере объявление функции kv находится перед главной функцией *main*, а описание функции — после.

Результат работы программы



Пример 16.3. Нахождение наибольшего из 2-х целых чисел **a** и **b**.

Алгоритм решения задачи. Консольный режим.

```
#include<iostream.h>
#include<conio.h>
//-------------------------------------
int max(int x, int y)  { //описание функции  max  совпадает с объявлением
  return (x>y)? x:y;
}
void main() {          //это главная функция
int a,b;
   cin>>a>>b;
   cout<<max(a,b);    //обращение к функции
   getch();
}
```

Фактические и формальные параметры соответствуют друг другу по количеству и порядку следования.

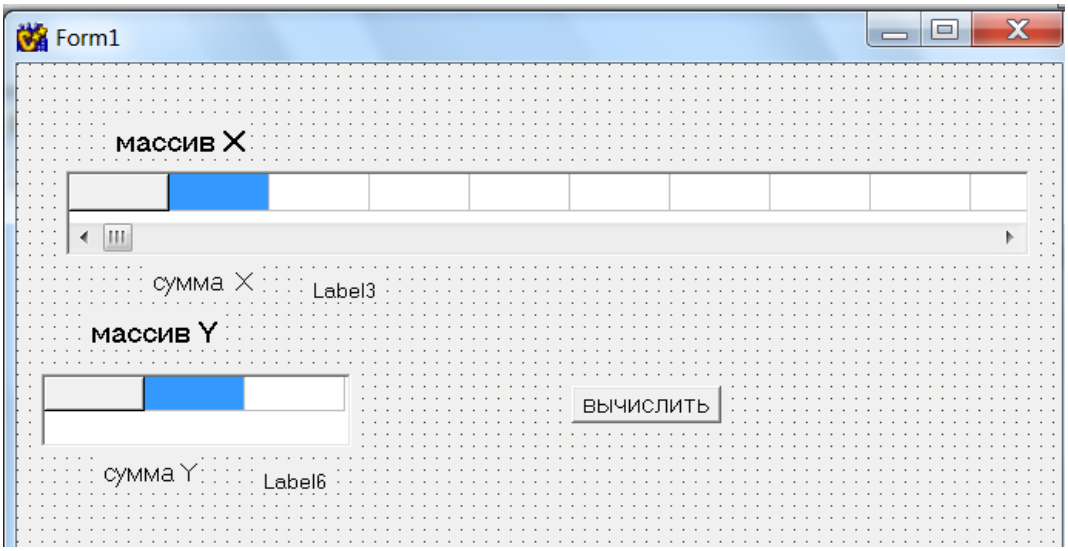
Пример 16.4. Создать функцию, возвращающую сумму элементов массива. Воспользоваться ею при вычисления сумм элементов массивов X и Y, содержащих 10 и 3 элемента соответственно.
Создать проект в графической среде.

Описание функции:

```
float sum(float a[], int n) {
float s=0;
  for (int i=0; i<n;i++)
  s+=a[i];
  return s;
 }
```

Порядок действий:

- разместить объекты, задать им необходимые свойства



- обработчик события *щелчок по кнопке вычислить* имеет вид:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
float sum(float a[], int n);                    //объявление функции
float x[10]={7.8, 6, 9, 0, 5.5, 6.6, 7, 8, 9, 10};
float y[3]={1.1, 2.2, 3.3};
int i;
  for (i=0; i<=9; i++)
  StringGrid1->Cells[i][0]=x[i];                //вывод массива x
  Label3->Caption=sum(x,10);                    //обращение к ф-ии, вывод результата
  for (i=0; i<=2;i++)
  StringGrid2->Cells[i][0]=y[i];                //вывод массива y
  Label6->Caption=sum(y,3);
}
float sum(float a[], int n) {                   //описание функции
float s=0;
   for (int i=0; i<n;i++)
   s+=a[i];
   return s;
 }
```

16.3. Рекурсивные функции

В языке C++ функции могут вызывать сами себя. Функция называется рекурсивной, если оператор в теле функции содержит вызов этой же функции.

Пример 16.5. Классический пример рекурсивной функции — вычисление факториала числа. Вычислить n! при $n \in [0; 10]$.

n!=1*2*3*…*n.

Пусть эта функция называется **factorial**.

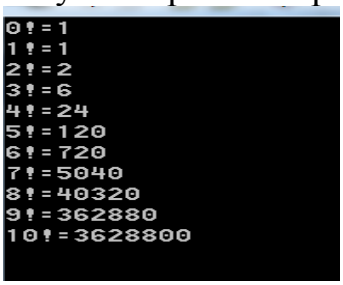
Описание функции:

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

Программа на С++ в консольном режиме имеет вид:

```
#include<iostream.h>
#include<conio.h>
//……
long int factorial (int n);                    //это объявление функции
void main() {                                  //это главная функция
  for (int i=0; i<=10; i++)
  cout<<i<<"!="<<factorial(i)<<"\n"; //обращение к функции
  getch();
}
long int factorial(int n) {                    //это описание функции
  if (n==0 | n==1) return 1;
   return factorial(n-1)*n;
}
```

Результат работы программы



Литература

1. Шалин П. А. Энциклопедия Windows XP. – СПб.:Питер, 2003, 688 с.
2. Самоучитель VBA. Гарнаев А. Ю. изд. 2, перераб. И доп. :БХВ-Петербург, 2004. – 560 с.
3. Швачич Г.Г., Овсянніков О.В. та ін. Інформатика та комп'ютерна техніка. Елементи об'єктно-орієнтованого програмування. Розділ «Реалізація концепції об'єктно-орієнтованого програмування в мові Visual Basic for Application»:Навчальний посібник. – Дніпропетровськ: НметАУ, 2006.- 52 с.
4. Информатика. Базовый курс: Учебник для вузов / Под ред. С. В. Симоновича. Изд.2:Питер, 2009.- 639с.
5. Березин Б.И., Березина С.Б. Начальный курс С и С++,-М.:Диалог – МИФИ, 2001.-288с.
6. Павловская Т.А. С/С++. Программирование на языке высокого уровня. Учебник для вузов. – СПб.:Питер, 2003, 400 с.
7. Пахомов Б.И. С/С++ и Borland C++ Builder для студентов. СПб.: БХВ-Петербург, 2006.-448с.