МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

ОЛИМПИАДА «Я-БАКАЛАВР» ДЛЯ ОБУЧАЮЩИХСЯ 5-11 КЛАССОВ

ИНФОРМАТИКА

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПОДГОТОВКИ К ЗАКЛЮЧИТЕЛЬНОМУ ЭТАПУ ОЛИМПИАДЫ 2025/2026 УЧЕБНОГО ГОДА ДЛЯ **10** КЛАССА

ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП

Характер и уровень сложности олимпиадных задач направлены на достижение целей проведения олимпиады: выявить способных участников, твердо владеющих школьной программой и наиболее подготовленных к освоению образовательных программ ВУЗов, обладающих логикой и творческим характером мышления знанием, вопросов информатики, вычислительной техники, информационных и коммуникационных технологий.

Задания дифференцированы по сложности и требуют различных временных затрат на верное и полное решение. Задания направлены на выявление интеллектуального потенциала, аналитических способностей и креативности мышления участников и т.п.

Очный этап олимпиады проводится только в письменной форме. Каждый участник олимпиады получает бланк с заданием, содержащий 6 заданий. При выполнении заданий требуется:

1. владеть навыками работы с позиционными системами счисления;
2. владеть навыками моделирования;
3.владеть навыками работы в табличном редакторе;
4. владеть навыками программирования.

При подготовке к олимпиаде следует повторить приведенные ниже темы.

ПЕРЕЧЕНЬ ЭЛЕМЕНТОВ СОДЕРЖАНИЯ, ВКЛЮЧЕННЫХ В ЗАДАНИЯ ОЛИМПИАДЫ ЗАКЛЮЧИТЕЛЬНОГО ЭТАПА

2025/2026 УЧЕБНОГО ГОДА

Тема 1. Работа с табличным процессором и его графическими возможностями

Диаграмма — это представление данных таблицы в графическом виде, которое используется для анализа и сравнения данных. На диаграмме числовые данные ячеек изображаются в виде точек, линий, полос, столбиков, секторов и в другой форме. Группы элементов данных, отражающих содержимое ячеек одной строки или столбца на рабочем листе, составляют ряд данных. На одной диаграмме можно отображать несколько рядов данных. Диаграмма связана с данными, на основе которых она построена, и при их обновлении немедленно меняет свой вид.

Строятся диаграммы с помощью Мастера (Конструктора) диаграмм. Вызывается через меню ВСТАВКА ► ДИАГРАММА или нажатием кнопки  на панели инструментов. Построение (редактирование) диаграммы выполняется за четыре достаточно независимых друг от друга этапа, на

которых выбираются тип диаграммы, исходные данные, параметры оформления и размещения.

ТИП диаграммы задаёт форму представления данных, выбирается из доступных стандартных вариантов (Гистограмма  , График  , Круговая , Линейчатая  , Точечная  , Кольцевая  , Пузырьковая  , Лепестковая  и др.) или из файла шаблона.

ВЫБОР ДАННЫХ в простых случаях часто осуществляется заранее, путём выделения диапазона перед построением диаграммы. Но всегда остаётся возможность задания или уточнения диапазона (рядов данных) на данном этапе. Ряды данных могут быть не просто несмежными, но и вообще располагаться на разных листах.

ОФОРМЛЕНИЕ диаграммы заключается в выборе МАКЕТА (отображение Заголовков, Осей, Линий сетки, Легенды, Подписей данных, Таблицы данных и др.) и СТИЛЯ (форматирование – задание размера, цвета, формы, других эффектов отображения элементов диаграммы – текста, линий, фигур, фоновой заливки).

РАСПОЛОЖЕНИЕ диаграммы может задаваться как на имеющемся в книге рабочем листе, так и на специально создаваемом отдельном листе диаграммы.

Функции — заранее определенные формулы, которые выполняют вычисления по заданным величинам, называемым аргументами, и в указанном порядке, определяемом синтаксисом.

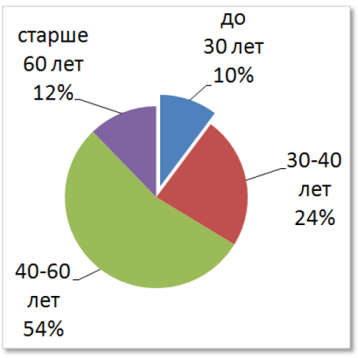
Функции MS Excel позволяют выполнять как простые, так и сложные вычисления, связанные с решением определенных задач. Некоторые вычисления могут быть выполнены как с помощью формул, так и с помощью аналогичных им функций. Например: Формула =C7+D7+E7 складывает содержимое ячеек C7, D7 и E7.

Элементы диаграмм

Пример задания:
Построить круговую диаграмму по указанным исходным данным

до 30 лет	30-40 лет	40-60 лет	старше 60 лет
20	48	108	24

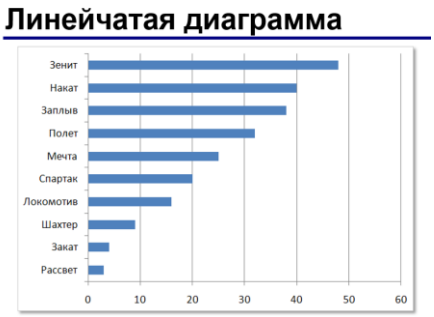
Ответ:



Пример задания:
Построить Линейчатую диаграмму по указанным исходным данным.

Команда	Мячи
Рассвет	3
Закат	4
Шахтер	9
Локомотив	16
Спартак	20
Мечта	25
Полет	32
Заплыв	38
Накат	40
Зенит	48

Ответ:



Тема 2. Логические основы компьютера

Логические операции могут производиться не только над булевыми величинами, но и над битами операндов. В этом случае логическая операция возвращает поразрядный результат, который либо истинен (1), либо ложен (0). В языках программирования могут существовать специальные операторы побитового выполнения логических операций. Например, в «Си++» и «Ява» поразрядным (побитовым) операциям НЕ, И, ИЛИ соответствуют операторы ~, &, | (сравните с операторами таблицы 7).

В Бейсике используются только побитовые логические операции, а операнды представляются в восьми-, шестнадцати- или тридцатидвухразрядном дополнительном коде. При этом булевым значениям False и True соответствуют десятичные значения 0 и -1, так 0 — число, в котором все биты обнулены, а -1 — двоичное число, все биты которого установлены в 1 (таблица 3).

Операциям *исключающее ИЛИ* (неравнозначность), *эквивалентность* (равнозначность) и *импликация* в Бейсике соответствуют операторы XOR, EQV и IMP. Результат логической операции определяется поразрядно согласно таблице 8. Операторы приведены в порядке убывания их приоритета.

Таблица 8 - Результаты, возвращаемые логическими операциями

Операнды		Результаты операций					
X	Y	NOT X	X AND Y	X OR Y	X XOR Y	X EQV Y	X IMP Y
1	1	0	1	1	0	1	1
1	0	0	0	1	1	0	0
0	1	1	0	1	1	0	1

0	0	1	0	0	0	1	1

Пример задания:

Воспользуемся приемами для упрощения записи сформулированного выражения из алгебры логики. В процессе преобразований вынесем ¬ у выражения A за знаки скобок. Далее целесообразно воспользоваться закономерностью исключенного третьего и распределительным способом. В итоге вычислений получим следующее равенство. Произведем необходимые алгебраические операции в уравнении и запишем результирующий ответ:

$$A + A \cdot B = A (\text{т.к. } A + A \cdot B = A \cdot 1 + A \cdot B = A \cdot (1 + B) = A \cdot 1 = A)$$

Ответ: A

Пример задания:

Вычислить логическое выражение.

Y = (38 OR &H1C) AND &H15 IMP NOT &O5

Y1=1 Y2=10100 Y3=1111111111111011 Y4=111110

Р а с ч е т з а д а н и я

Переводим все операнды в двоичную систему счисления:

$38_{(10)} = 100110_{(2)}$

$1C_{(16)} = 11100_{(2)}$

$15_{(16)} = 10101_{(2)}$

$5_{(8)} = 101_{(2)}$

Указываем приоритет выполнения операций:

Y = (38 OR &H1C) AND &H15 IMP NOT &O5

1 2 3 4

Определяем результат выполнения каждой операции побитно, используя для представления операндов шестнадцатиразрядный дополнительный код:

1) 38 **OR** &H1C

0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0

0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0

2) **NOT** &O5

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0

3) (38 OR &H1C) **AND** &H15

0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

4) (38 OR &H1C) AND &H15 **IMP** NOT &O5

0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1

Ответ: Y = Y3 = $1111111111111011_{(2)}$ = $177773_{(8)}$ = $FFFB_{(16)}$ = $-5_{(10)}$

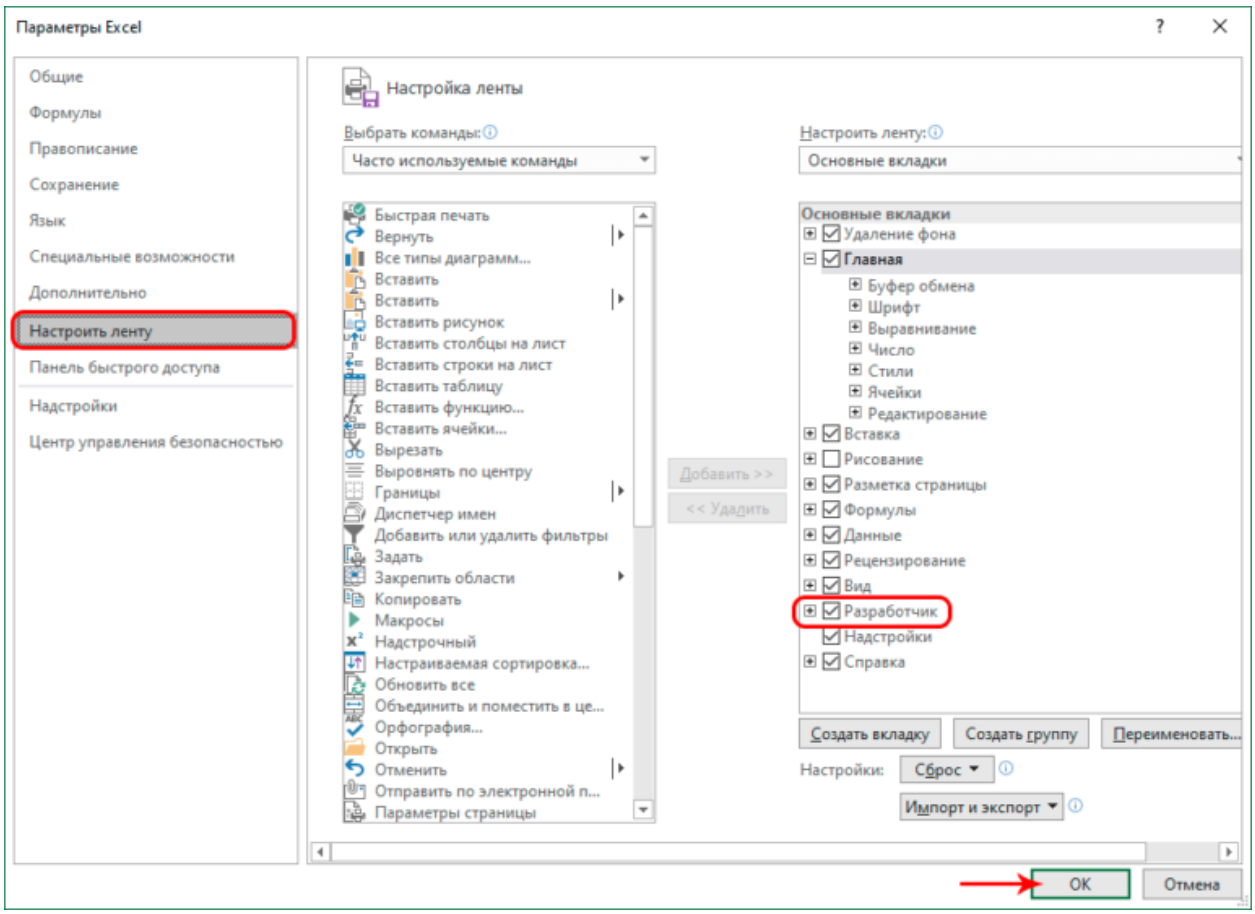
Тема 3. Работа с табличным процессором и его графическими возможностями

В состав пакета Microsoft Office входит мощный инструмент автоматизации обработки табличной информации – Microsoft Office Excel. Эта программа обеспечивает эффективную обработку числовых данных, позволяет выполнять вычисления, анализировать и визуализировать данные в электронных таблицах, находит широкое применение, особенно при выполнении бухгалтерских и экономических расчетов. С помощью электронных таблиц можно решать задачи планирования и бухгалтерского учёта, строить расписания и графики проведения работ, выполнять самые разные расчёты.

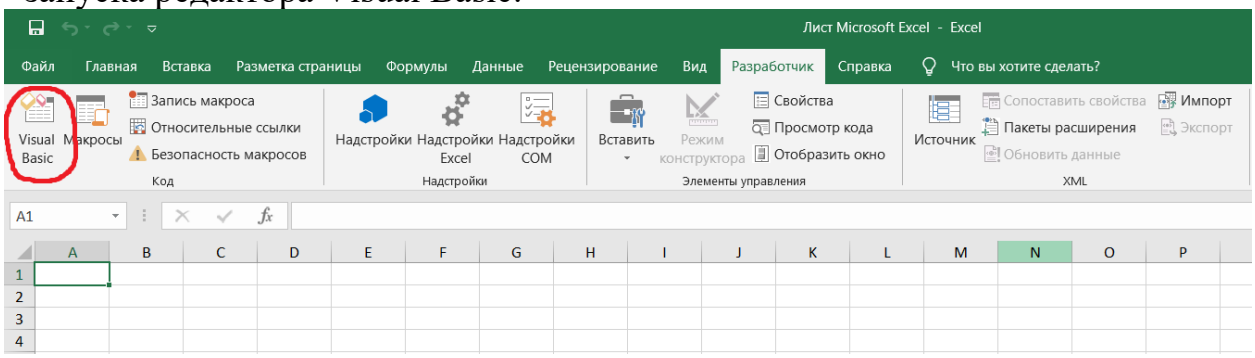
Кроме множества встроенных функций (финансовых, математических, текстовых, логических и других), в Excel реализован первичный статистический анализ, сортировка данных, формирование выборки по различным критериям, построение таблиц результатов, диаграмм. Кроме того, в Excel имеется возможность создания функций, определённых пользователем.

Макрос — это действие (или последовательность действий), записанное на языке программирования Visual Basic for Applications (VBA), которое можно выполнить сколько угодно раз.

Для работы с макросами необходимо открыть специальное окно – редактор программ на VBA, встроенный в Microsoft Excel, который доступен при включенном **режиме Разработчика**. Чтобы включить режим разработчика, нужно перейти по пути: Файл – Параметры – Настроить ленту и поставить в правой части окна флажок Разработчик.



В ленте появится вкладка «Разработчик», в которой есть кнопка для запуска редактора Visual Basic.



Макрос можно записать двумя способами

1. Вручную в редакторе (для сложных макросов и процедур);
2. С помощью функции «Запись макроса», которая находится во вкладке Разработчик.

Для начала, разберем **первый способ** – запишем макрос вручную. Например, у нас есть код макроса для печати активного листа:

```
Sub Print_1()
ActiveSheet.UsedRange.PrintOut
End Sub
```

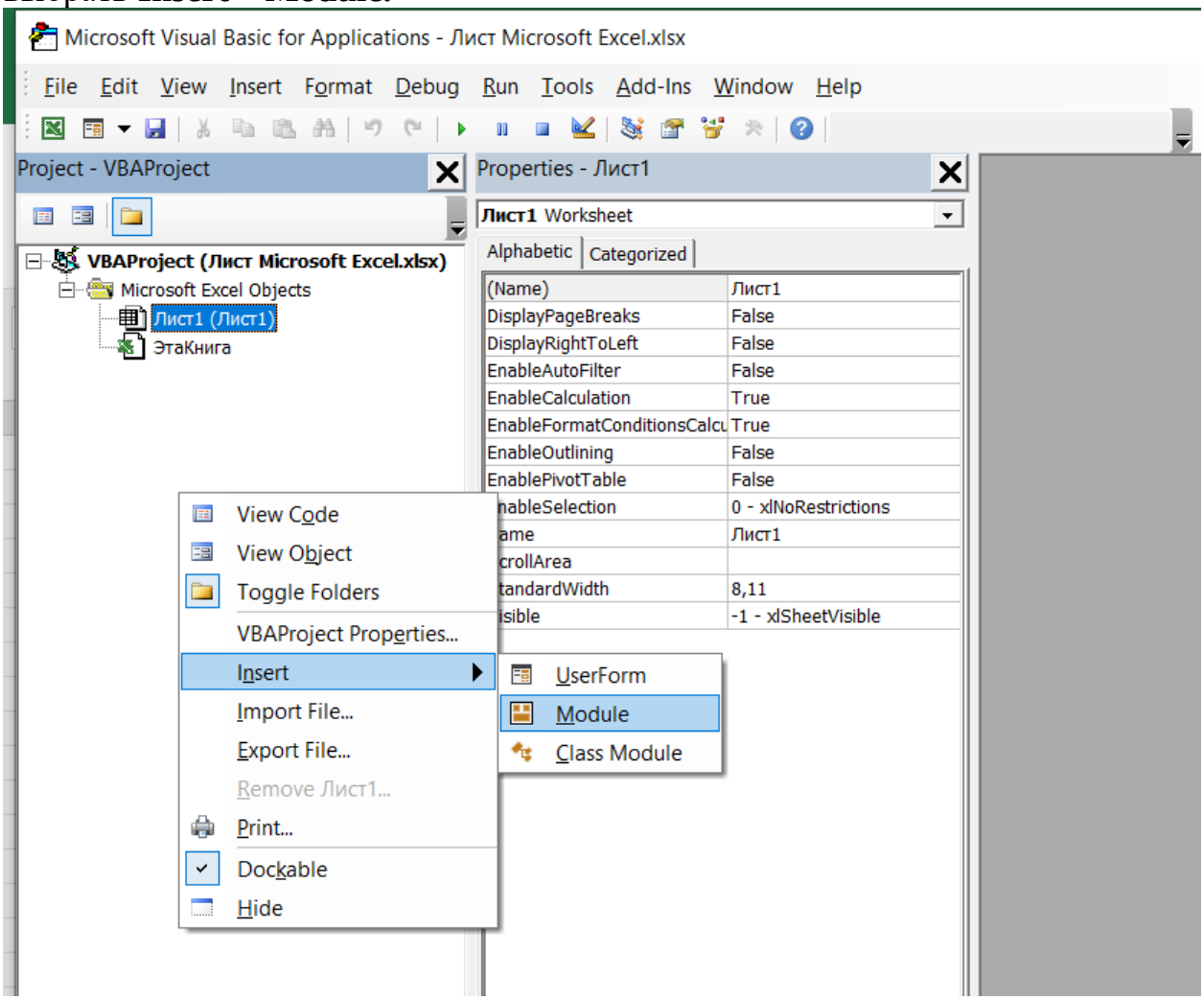
– Каждый макрос начинается с оператора Sub, за которым идет имя макроса и список аргументов (входных значений) в скобках. Если аргументов нет, то скобки нужно оставить пустыми.

– Каждый макрос заканчивается оператором End Sub.

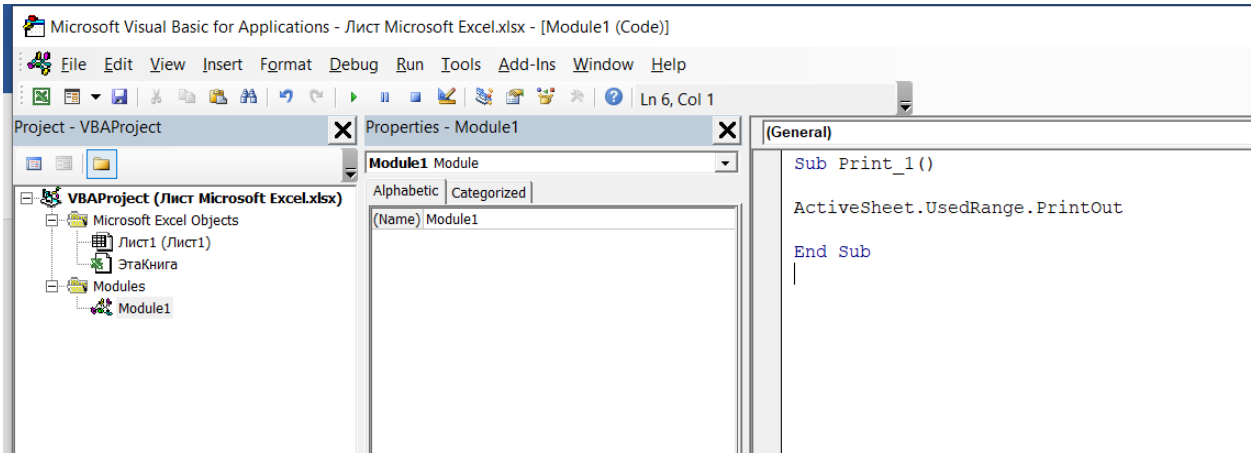
– Все, что находится между Sub и End Sub – тело макроса, т.е. команды, которые будут выполняться при запуске макроса. В данном случае наш макрос при запуске будет распечатывать активный лист, либо заданную пользователем область печати на активном листе.

Макросы хранятся в модулях. В любой книге Excel мы можем создать любое количество модулей и разместить там наши макросы. Один модуль может содержать любое количество макросов. Доступ ко всем модулям осуществляется с помощью окна VBA Project в левом верхнем углу редактора

Чтобы создать модуль, нужно открыть во вкладке «Разработчик» редактор Visual Basic, в окне VBA Project щелкнуть правой кнопкой мыши и выбрать **Insert – Module.**

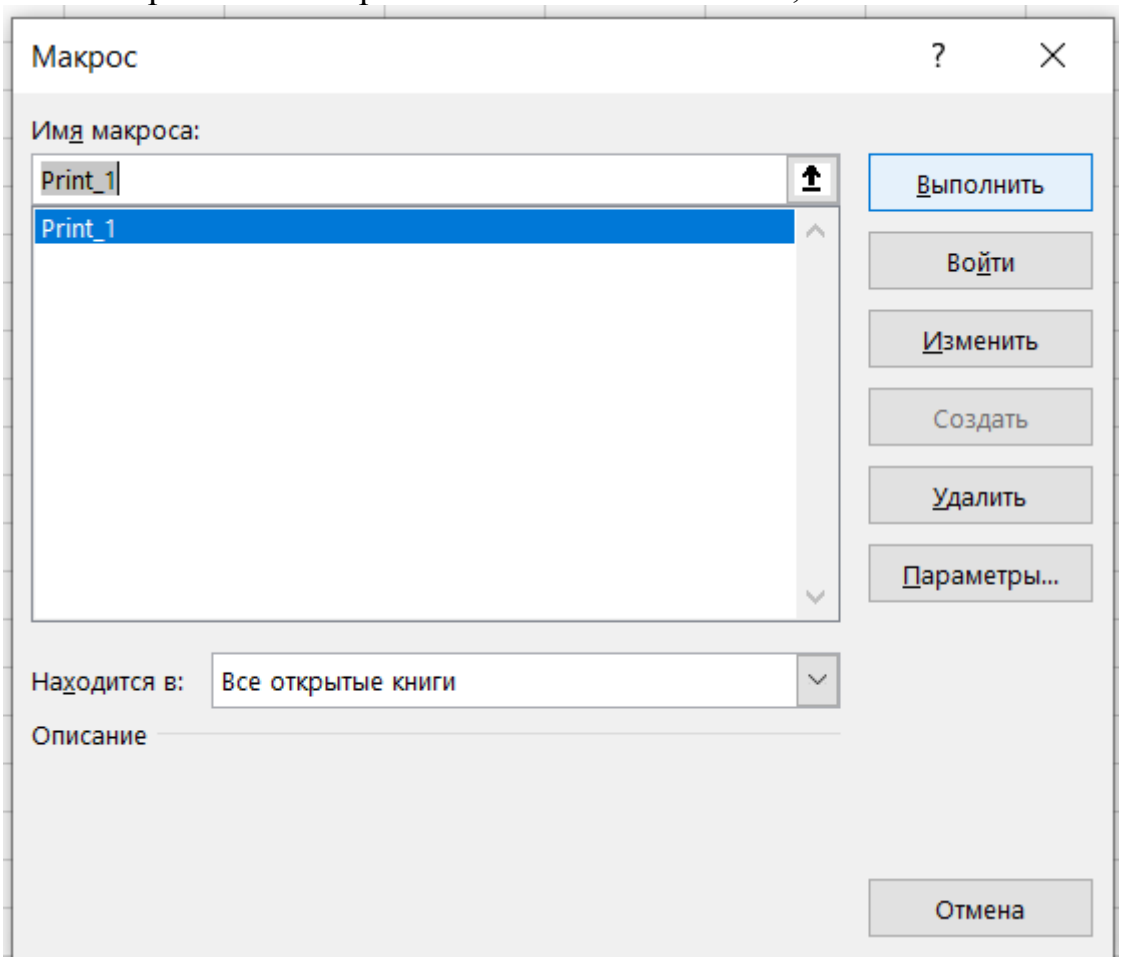


Появится вкладка «Modules», в которой находится наш модуль «Module1». Копируем код макроса и вставляем его в наш модуль.



Готово, макрос записан. Теперь нужно определиться с запуском макроса. Макрос можно запустить несколькими способами:

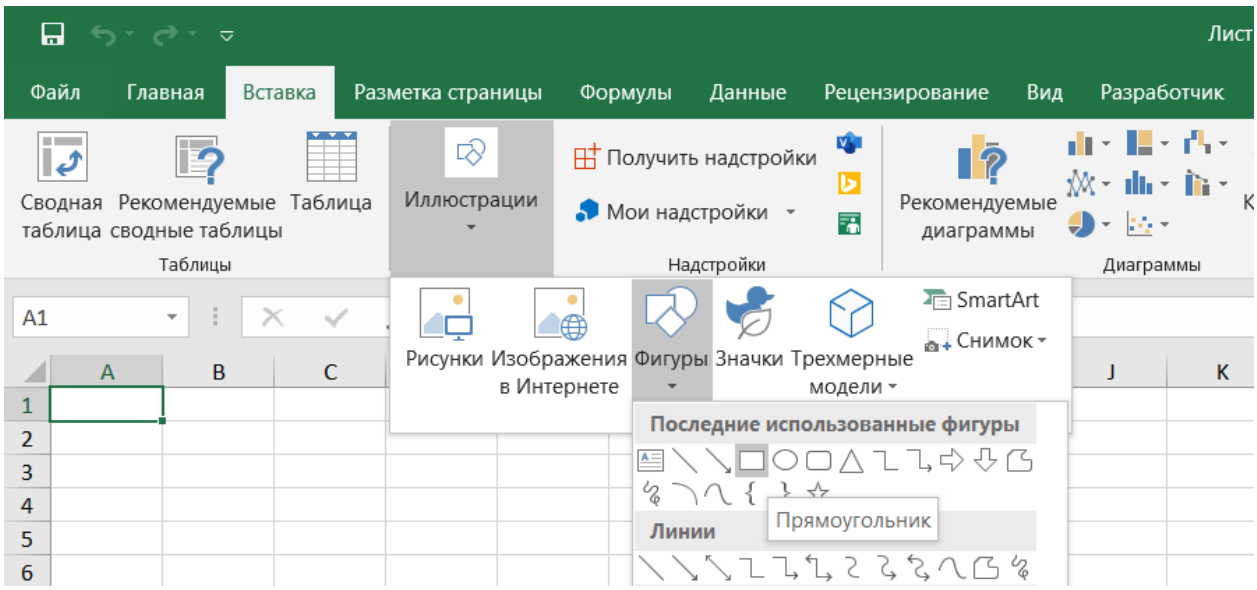
– Во вкладке «Разработчик» нажать кнопку «Макросы», в появившемся окне выбрать наш макрос и нажать «Выполнить»;



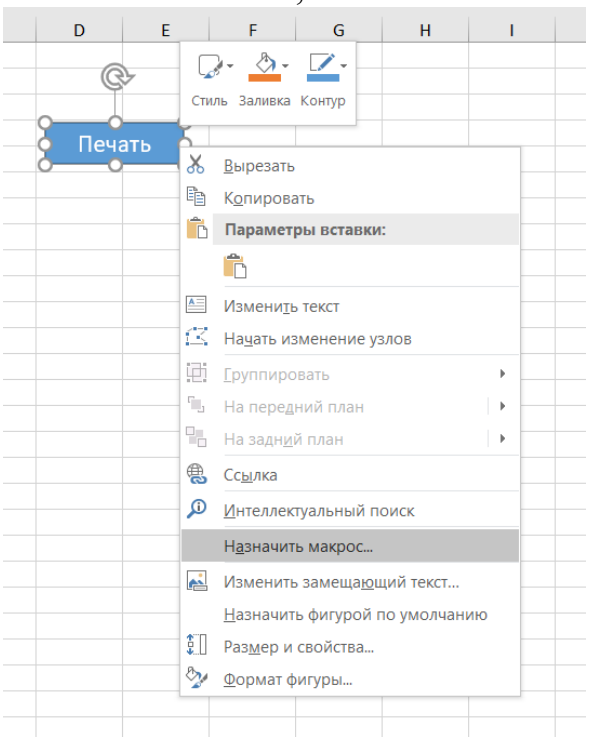
– С помощью горячих клавиш, сочетание которых можно установить в том же окне, нажав кнопку «Параметры»;

– Назначить макрос на кнопку, размещенную на листе. Этот способ рассмотрим более подробно.

Для начала, добавим кнопку на лист, который нужно распечатать. Для этого откроем вкладку «Вставка» и выберем Иллюстрации-Фигуры.



Выбираем любую фигуру (я выбрал прямоугольник), затем щелкаем левой кнопкой мыши на листе и растягиваем фигуру. Получилась кнопка, которую можно отформатировать (поменять цвет, вставить надпись и т.д.). Щелкаем по кнопке правой кнопкой мыши, выбираем «Назначить макрос», выбираем наш макрос Print1 из списка, жмем «Ок».

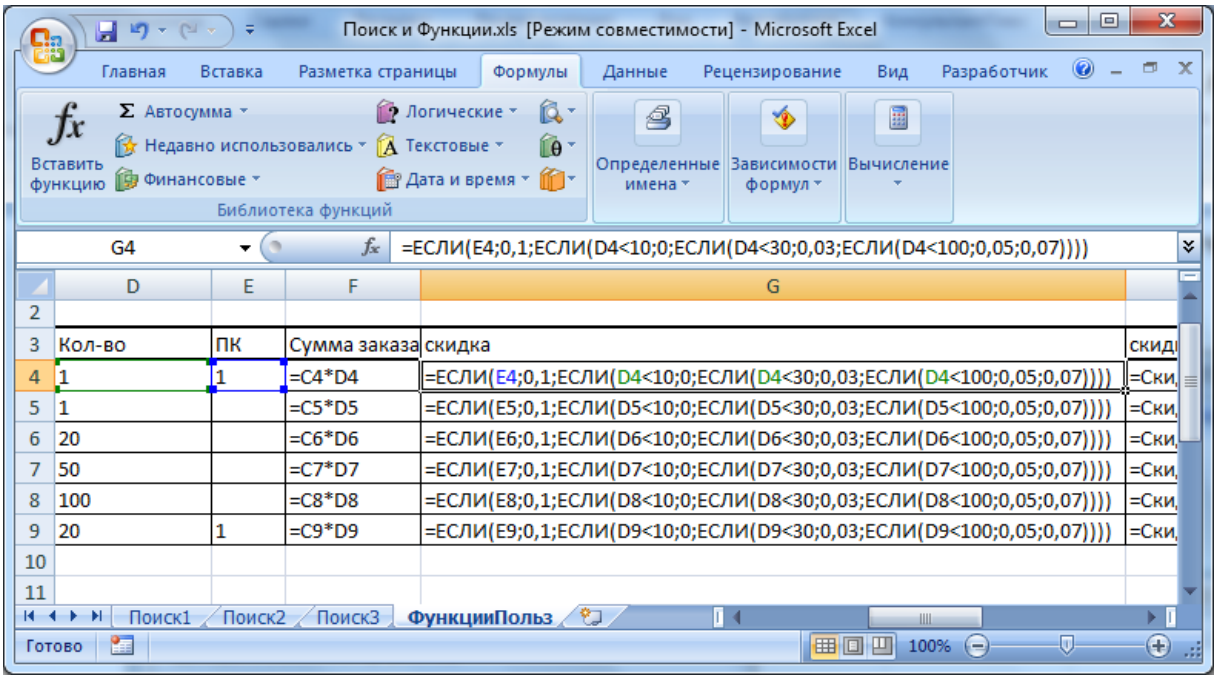


Макрос назначен на кнопку. Теперь, при нажатии кнопки, активный лист (или установленная на нем область печати) будет мгновенно распечатана.

Рассмотрим возможности реализации в Excel функций, определённых пользователем, на примере следующей задачи: «В рознично–оптовом магазине действует система скидок. При заказе от 10 до 30 товаров скидка составляет 3%; от 30 до 100 – 5%; от 100 и более – 7%. Постоянным покупателям, при предъявлении соответствующей карточки, предоставляется десятипроцентная скидка».

Создадим электронную таблицу для решения данной задачи, как с использованием встроенных функций, так и с применением функций, определённых пользователем.

Один из вариантов использования встроенных функций – логическая функция ЕСЛИ –представлен на рисунке 4.1. Особое внимание следует уделить формуле, представленной в ячейке G4, которую затем следует скопировать в остальные ячейки этого столбца.



	D	E	F	G
2				
3	Кол-во	ПК	Сумма заказа	скидка
4	1	1	=C4*D4	=ЕСЛИ(E4;0,1;ЕСЛИ(D4<10;0;ЕСЛИ(D4<30;0,03;ЕСЛИ(D4<100;0,05;0,07))))
5	1		=C5*D5	=ЕСЛИ(E5;0,1;ЕСЛИ(D5<10;0;ЕСЛИ(D5<30;0,03;ЕСЛИ(D5<100;0,05;0,07))))
6	20		=C6*D6	=ЕСЛИ(E6;0,1;ЕСЛИ(D6<10;0;ЕСЛИ(D6<30;0,03;ЕСЛИ(D6<100;0,05;0,07))))
7	50		=C7*D7	=ЕСЛИ(E7;0,1;ЕСЛИ(D7<10;0;ЕСЛИ(D7<30;0,03;ЕСЛИ(D7<100;0,05;0,07))))
8	100		=C8*D8	=ЕСЛИ(E8;0,1;ЕСЛИ(D8<10;0;ЕСЛИ(D8<30;0,03;ЕСЛИ(D8<100;0,05;0,07))))
9	20	1	=C9*D9	=ЕСЛИ(E9;0,1;ЕСЛИ(D9<10;0;ЕСЛИ(D9<30;0,03;ЕСЛИ(D9<100;0,05;0,07))))
10				
11				

Рисунок 4.1 – Таблица с отображением формул

А теперь для определения размера скидки опишем функцию пользователя на языке *Visual Basic for Applications*. Для запуска редактора *VBA* используем команду *Visual Basic*, расположенную на ленте, вкладка *Разработчик*, группа *Код* как на рисунке 4.2, или нажимаем сочетание клавиш *Alt+F11*.

В открывшемся окне создаём модуль (командой меню *Insert – Module*), в котором размещаем описание функции, что показано на рисунке 4.3.

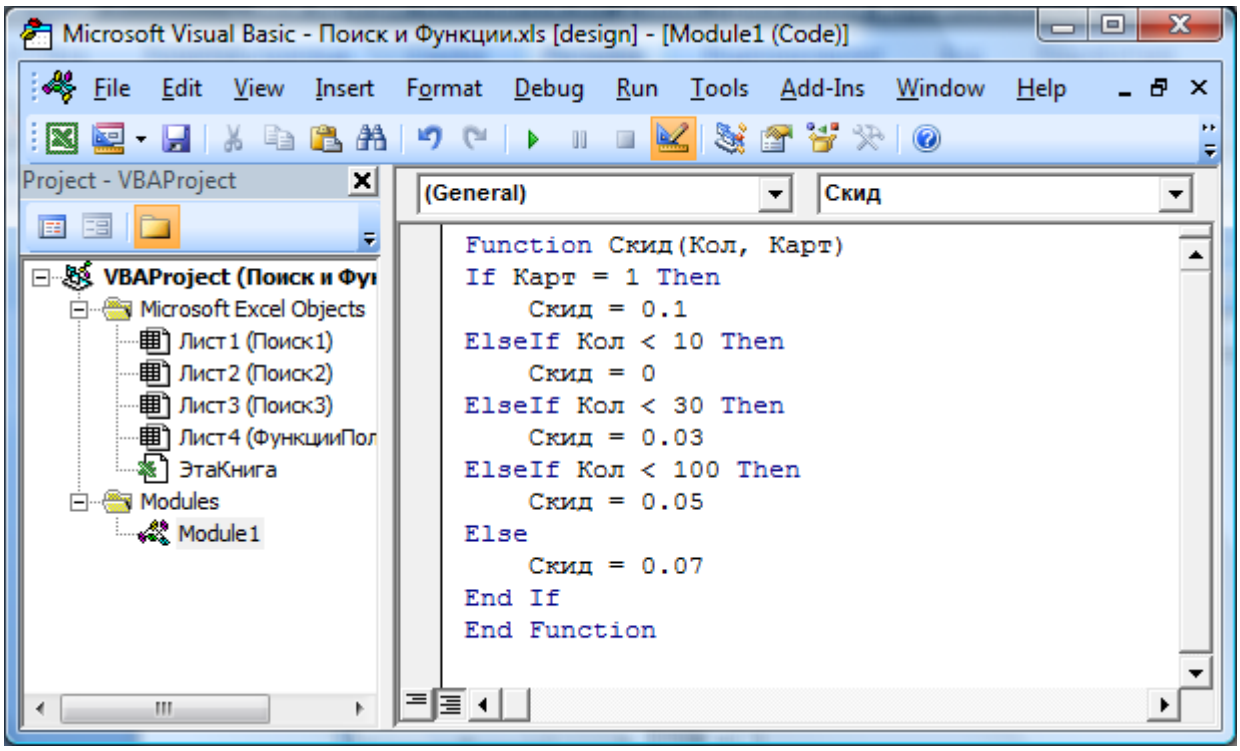


Рисунок 4.3 – Создание функции в модуле VBA

После ключевого слова *Function* следует задать имя функции (назовём её *Скид*, чтобы имя не совпадало со встроенной финансовой функцией *Скидка*) и формальные аргументы (Кол, Карт), определяющие, соответственно, количество заказанных единиц продукции и наличие карточки постоянного покупателя. Затем описывается тело функции, реализующее логику вычислений.

Теперь, если позволяют настройки безопасности данной книги (макросы должны быть включены), созданную функцию можно использовать в формулах, она становится также доступной в мастере вставки функций (категория Определенные пользователем – Рисунок 4.4).

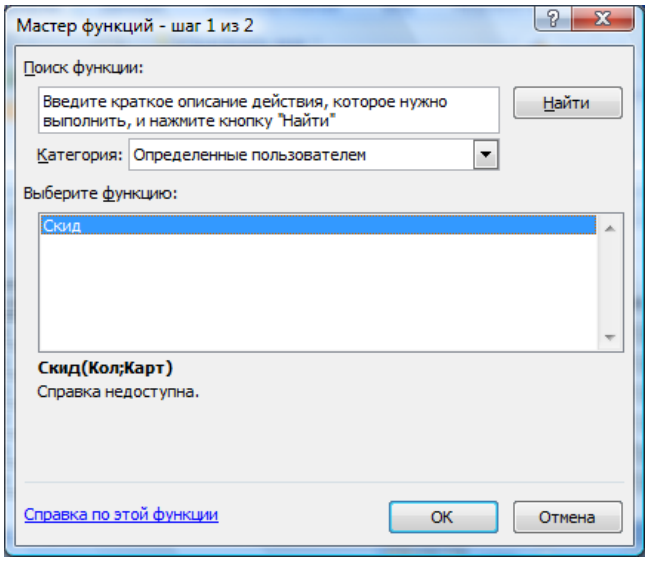
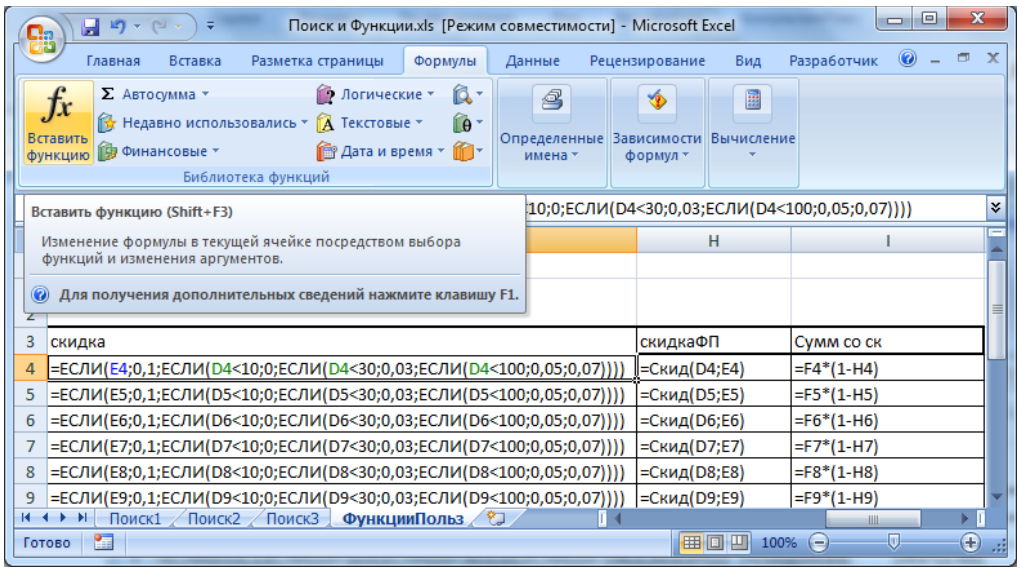


Рисунок 4.4 – Окно мастера функций

Введём соответствующую формулу в графу СкидкаФП на рабочем листе, как показано на рисунке 4.5. Как видно, запись этой формулы осуществляется намного проще, при её вводе допускается меньше ошибок, чем при заполнении графы Скидка, а при необходимости её легче отредактировать.

Ответ:



Тема 4. Программирование

Программирование — это процесс и искусство создания компьютерных программ с помощью языков программирования. В более широком смысле оно представляет собой разработку программного обеспечения. Программирование сочетает в себе элементы искусства, науки, математики и инженерии и подразумевает написание исходного кода, который потом компилируется или интерпретируется в машинный код для выполнения на компьютере.

Основная задача программирования — это реализация алгоритмов, то есть последовательностей действий, которые компьютер должен выполнить для решения конкретной задачи. При этом программисты выбирают подходящий язык программирования, учитывая тип задачи и требования к производительности и удобству разработки.

Основные понятия в программировании включают переменные (хранение данных), инструкции (команды для компьютера) и выражения (вычисления). Программирование широко используется в самых разных сферах: от разработки сайтов и приложений до автоматизации бизнес-процессов, медицины и образования

Каждый язык имеет свои собственные особенности и тонкости реализации, которые можно узнать на официальных страницах этих языков.

Например:

Python https://www.python.org/

Java https://www.java.com/ru/

C# https://dotnet.microsoft.com/ru-ru/languages/csharp

C++ https://learn.microsoft.com/ru-ru/cpp/?view=msvc-170

Эти ссылки являются основными для получения теоретических знаний о построении и особенностях применения конкретного языка программирования.

Для подготовки к заключительному этапу необходимо повторить:

Общие вопросы программирования.

Для успешного освоения программирования необходимо изучить широкий спектр общих вопросов. Начните с основ. Изучите, что такое алгоритм, как его представить в виде блок-схемы или псевдокода, и освойте основные алгоритмические структуры, такие как последовательность, ветвление и циклы, а также базовые алгоритмы поиска и сортировки.

Углубитесь в понимание типов данных, включая основные (целые числа, числа с плавающей точкой, символы, строки, логические значения) и структуры данных (массивы, связные списки, стеки, очереди, деревья, графы, хэш-таблицы). Необходимо понимать, как объявлять и инициализировать переменные и константы, а также учитывать область их видимости. Освойте арифметические, логические операторы, операторы сравнения и присваивания.

Изучите парадигмы программирования, такие как императивное, декларативное, структурное, объектно-ориентированное и функциональное программирование, а также принципы SOLID. Необходимо понимать структуры управления программой, включая условные операторы, циклы, функции, процедуры и обработку исключений.

Необходимо ориентироваться в терминах, что такое алгоритм, компилятор, интерпретатор, синтаксис, исходный код, парадигма программирования, отладка программы, API, IDE, и т.д. Знать основы работы с программным кодом.

Типы и структуры данных.

В программировании типы и структуры данных играют ключевую роль в организации и эффективной обработке информации. Фундаментом являются примитивные типы данных, такие как целые числа (int, short, long, byte), числа с плавающей точкой (float, double), символы (char), строки (String) и логические значения (boolean). Важно понимать их особенности и диапазоны значений.

Составные типы, такие как массивы и структуры, строятся на основе примитивных типов, позволяя группировать связанные данные.

Структуры данных, в свою очередь, предоставляют различные способы организации и хранения, влияя на производительность программы.

Линейные структуры включают связные списки, стеки, очереди и деки, каждая из которых имеет свои особенности и принципы работы (LIFO, FIFO).

Нелинейные структуры, такие как деревья и графы, предоставляют иерархические и сетевые способы организации данных.

Ассоциативные структуры, такие как хэш-таблицы, словари и множества, обеспечивают быстрый поиск и хранение уникальных элементов.

Абстрактные типы данных (ADT) определяют что структура данных делает, а не как. Список, стек, очередь, дерево и граф являются примерами ADT.

Выбор подходящей структуры данных зависит от требований к производительности, объема данных, типов операций и простоты реализации.

При изучении необходимо понимать концепции указателей и ссылок, динамического выделения памяти, рекурсии, обобщенного программирования и сложности алгоритмов. Рекомендуется изучать теорию, практиковаться в реализации структур данных, рассматривать примеры их использования и анализировать эффективность различных структур для конкретных задач.

Условные конструкции.

Разберитесь с базовым синтаксисом работы условных конструкций. Для этого рассмотрите операторы:

if (если): Самый простой условный оператор. Выполняет блок кода только в том случае, если условие истинно.

```
if условие:
    # Код, который выполняется, если условие истинно
```

else (иначе): Выполняет блок кода, если условие if ложно.

```
if условие:
    # Код, который выполняется, если условие истинно
else:
    # Код, который выполняется, если условие ложно
```

elif (иначе если) / else if: Позволяет проверять несколько условий последовательно. Выполняется блок кода, соответствующий первому истинному условию.

```
if условие1:
    # Код, если условие1 истинно
elif условие2:
    # Код, если условие2 истинно
```

else:

```
    # Код, если ни одно из условий не истинно
```

Условия (логические выражения).

Операторы сравнения:

== (равно)

!= (не равно)

> (больше)

< (меньше)

>= (больше или равно)

<= (меньше или равно)

Логические операторы:

and / && (логическое И). Условие истинно, если оба операнда истинны.

or / || (логическое ИЛИ). Условие истинно, если хотя бы один операнд истинен.

not / ! (логическое НЕ). Инвертирует значение операнда (истина становится ложью, и наоборот).

Оператор **switch** (в некоторых языках):

Альтернатива цепочке **if-elif-else** для проверки равенства переменной с несколькими конкретными значениями.

Пример для c++

```
switch (переменная) {
    case значение1:
        // Код для значения1
        break;
    case значение2:
        // Код для значения2
        break;
    default:
        // Код, если ни одно из значений не совпало
}
```

Важно помнить про break, чтобы избежать "проваливания" в следующие case.

Массивы.

Массив – это упорядоченная структура данных, содержащая элементы одного типа, которые расположены в памяти последовательно. Каждый элемент доступен по своему индексу, начиная, как правило, с нуля.

Массивы бывают статические, размер которых фиксируется во время компиляции и не может быть изменен, и динамические, размер которых можно изменять во время выполнения программы. Также существуют многомерные

массивы, представляющие собой массивы массивов и используемые для представления, например, таблиц.

Основные операции над массивами включают создание, доступ к элементам по индексу (чтение и запись), итерацию (перебор всех элементов), изменение размера (для динамических массивов), поиск элементов и сортировку. Важно помнить об ошибке выхода за границы массива.

Существуют различные алгоритмы для работы с массивами, такие как линейный и бинарный поиск, а также алгоритмы сортировки: пузырьковая сортировка, сортировка вставками, выбором, быстрая сортировка и сортировка слиянием. Важно понимать, как сложность алгоритмов (оценка их эффективности) влияет на время выполнения операций с массивами, особенно с большими объемами данных.

В разных языках программирования массивы имеют свои особенности. В C/C++ важен контроль типов и ручное управление памятью (или использование умных указателей), в Java массивы являются объектами, а в Python списки предоставляют удобные встроенные методы для работы с данными.

В Python и JavaScript поддерживаются динамические массивы с гибкой типизацией и автоматическим управлением памятью.

Для изучения массивов рекомендуется начать с теории, затем перейти к практике решения задач, использовать отладчик для понимания работы кода и анализировать существующие примеры кода.

Циклы.

Цикл в программировании – это конструкция, позволяющая многократно выполнять определенный блок кода, который называется телом цикла. Повторение происходит до тех пор, пока выполняется заданное условие цикла.

В циклах часто используется счетчик (итератор) – переменная, которая изменяется на каждой итерации и помогает контролировать выполнение цикла.

Существуют три основных типа циклов:

— for (цикл со счетчиком);

— while (цикл "пока");

— do-while (цикл "делай-пока").

Синтаксис циклов немного отличается в разных языках программирования, но основная логика остается одинаковой.

Управление циклом осуществляется с помощью операторов break (немедленный выход из цикла) и continue (пропуск текущей итерации). Важно избегать создания бесконечных циклов, которые могут возникнуть, если неверно задано условие выхода.

Вложенные циклы – это циклы, помещенные внутри других циклов, которые полезны для обработки многомерных массивов и других сложных структур данных.

Циклы находят широкое применение в программировании, например, для обработки массивов, чтения данных из файлов, создания графических интерфейсов и реализации различных алгоритмов. Важно уметь оптимизировать циклы, уменьшая количество итераций и вынося инвариантный код за их пределы.

При работе с циклами часто возникают ошибки, такие как бесконечные циклы, смещение индекса, неправильная инициализация счетчика и неверное использование break и continue.

Использование отладчика и анализ существующего кода помогают избежать этих ошибок и лучше понять работу циклов. В некоторых языках существуют продвинутые концепции, такие как итераторы и генераторы в Python, которые предоставляют более эффективные способы перебора последовательностей. Также бывают доступны циклы foreach и параллельные циклы для упрощения работы с коллекциями и повышения производительности.

Пример задания: «Анализатор температурного журнала».

Необходимо разработать программу, которая обрабатывает данные о ежедневной температуре воздуха в течение месяца, выполняет базовый статистический анализ и предоставляет результаты.

Входные данные:

Массив (список) из 30 целых чисел, представляющих ежедневную температуру воздуха за месяц в градусах Цельсия. Температуры вводятся вручную.

Требуется выполнить:

Вычислить и вывести среднюю температуру за месяц.

Подсчитать и вывести количество дней в месяце, когда температура была ниже 0 градусов Цельсия.

Найти и вывести максимальную температуру за месяц и день (индекс в массиве + 1), когда она была зафиксирована впервые.

Найти и вывести минимальную температуру за месяц и день (индекс в массиве + 1), когда она была зафиксирована впервые.

Выходные данные:

Программа должна вывести на экран следующие результаты:

Средняя температура: [значение]

Количество дней с отрицательной температурой: [значение]

Максимальная температура: [значение], День: [значение]

Минимальная температура: [значение], День: [значение]

Решение.

```
temperatures = []
for i in range(30):
```

```
    while True:
        try:
            temp = int(input(f"Введите температуру за день {i+1}: "))
            temperatures.append(temp)
            break
        except ValueError:
            print("Некорректный ввод. Пожалуйста, введите целое число.")

# Анализ данных
total_temperature = 0
negative_days = 0
max_temp = temperatures[0]
max_temp_day = 1
min_temp = temperatures[0]
min_temp_day = 1

for i, temp in enumerate(temperatures):
    total_temperature += temp

    if temp < 0:
        negative_days += 1

    if temp > max_temp:
        max_temp = temp
        max_temp_day = i + 1

    if temp < min_temp:
        min_temp = temp
        min_temp_day = i + 1

average_temperature = total_temperature / 30

# Вывод результатов
print(f"Средняя температура: {average_temperature}")
print(f"Количество дней с отрицательной температурой: {negative_days}")
print(f"Максимальная температура: {max_temp}, День: {max_temp_day}")
print(f"Минимальная температура: {min_temp}, День: {min_temp_day}")
```

Описание кода.

Первый шаг – получение данных о температуре от пользователя. Для этого создается пустой список под названием temperatures, в который будут добавляться значения температуры для каждого дня. Затем запускается цикл, который повторяется 30 раз – по разу для каждого дня месяца. Внутри этого цикла используется еще один цикл, while True, который гарантирует, что пользователь введет корректные данные, а именно – целое число.

Для обработки возможных ошибок при вводе используется конструкция try-except. В блоке try программа пытается преобразовать введенную пользователем строку в целое число с помощью функции int(). Если преобразование проходит успешно, то полученное число, представляющее температуру за определенный день, добавляется в список temperatures с помощью метода append(). После добавления температуры в список, внутренний цикл while True прерывается оператором break, и программа переходит к следующему дню. Если же пользователь вводит что-то, что не

может быть преобразовано в целое число, например, текст или дробное число, возникает ошибка ValueError.

В этом случае, блок except перехватывает эту ошибку, выводит информационное сообщение о том, что нужно ввести целое число, и программа возвращается к началу цикла while True, чтобы пользователь мог попробовать ввести данные снова. Таким образом, обеспечивается надежный ввод данных.

После того, как все данные о температуре успешно введены и сохранены в списке temperatures, начинается этап анализа данных. Сначала, инициализируются несколько переменных: total_temperature для подсчета общей суммы температур (она начинается с 0), negative_days для подсчета количества дней с отрицательной температурой (тоже начинается с 0), а также max_temp и min_temp для хранения максимальной и минимальной температур соответственно.

Важно отметить, что переменные max_temp и min_temp изначально устанавливаются равными первому элементу списка temperatures, и предполагается, что в списке есть хотя бы один элемент. Так же хранятся переменные max_temp_day, min_temp_day -- день, в котором зафиксирована максимальная и минимальная температуры соответственно. При этом max_temp_day и min_temp_day инициализируются значением 1, поскольку отсчет дней начинается с 1.

Затем запускается цикл for, который перебирает все элементы списка temperatures. Функция enumerate позволяет получить индекс i текущего элемента, который используется для определения номера дня. Для каждой температуры в списке выполняется несколько проверок. Сначала, текущая температура добавляется к общей сумме температур total_temperature. Затем, проверяется, является ли текущая температура отрицательной.

Если да, то увеличивается счетчик отрицательных дней negative_days. Далее, проверяется, больше ли текущая температура, чем текущее значение max_temp. Если это так, то max_temp обновляется текущей температурой, а также номер дня, в который она зафиксирована, сохраняется в переменной max_temp_day. Это происходит только в том случае, если найдена новая максимальная температура. Аналогично, проверяется, меньше ли текущая температура, чем текущее значение min_temp. Если это так, то min_temp обновляется, и только если найдена температура меньше текущей минимальной, в min_temp_day сохраняется номер дня, когда была зафиксирована минимальная температура.

После завершения цикла анализа, вычисляется средняя температура, путем деления общей суммы температур total_temperature на количество дней (30). Этот результат сохраняется в переменной average_temperature.

Наконец, программа выводит результаты анализа на экран. Используя f-строки, программа выводит значения средней температуры, количества дней с отрицательной температурой, максимальной температуры и дня, когда она

была зафиксирована, а также минимальной температуры и дня, когда она была зафиксирована.

F-строки позволяют легко встраивать значения переменных в текст вывода, делая его более читаемым и понятным."

Литература для подготовки

1. Информатика: 10-й класс: учебник. Босова Л.Л., Босова А.Ю., АО «Издательство «Просвещение».
2. Информатика: 10-й класс: учебник. Гейн А.Г. АО «Издательство «Просвещение»

Информационные ресурсы:

1 Единая коллекция цифровых образовательных ресурсов [Электронный ресурс]. – Режим доступа http://school-collection.edu.ru
2 Сайт журнала информатика. [Электронный ресурс]. – Режим доступа Журнал «Информатика и образование» — Издательство "Образование и Информатика"